ISMIR 2009

Proceedings of the 10th International Society
for Music Information Retrieval Conference

Celebrating the 10th Anniversary:

Increasing Diversity and Toward the Future

October 26–30, 2009

Kobe, Japan

ISMIR 2009 is organized by the International Society for Music Information Retrieval.
Website: http://ismir2009.ismir.net

Cover and logo design by Toshie Matsui (Kwansei Gakuin University, Japan)

Edited by:
Keiji Hirata (NTT Communication Science Laboratories, Japan)
George Tzanetakis (University of Victoria, Canada)
Kazuyoshi Yoshii (AIST, Japan)

Printed by ISEBU
Website: http://www.isebu.co.jp

Published by the International Society for Music Information Retrieval
Website: http://www.ismir.net
ISBN 978-0-9813537-0-8

Copies may be ordered from:
Ichiro Fujinaga
Schulich School of Music, McGill University
555 Sherbrooke Street West, Montreal, QC, H3A 1E3
Mail: ich@music.mcgill.ca

Corporate Support

ISMIR 2009 gratefully acknowledges support from the following sponsors:

Sun Microsystems: http://www.sun.com (Gold sponsor)

Gracenote: http://www.gracenote.com (Gold sponsor)

Yamaha Corporation: http://www.yamaha.com (Silver sponsor)

KDDI R&D Laboratories: http://www.kddilabs.jp/english (Bronze sponsor)

Sony Corporation: http://www.sony.net (Bronze sponsor)

KORG Inc.: http://www.korg.com (Bronze sponsor)

ProQuest: http://www.proquest.com (Bronze sponsor)

Conference Committee

General Chairs:

Masataka Goto (AIST, Japan)
Ichiro Fujinaga (McGill University, Canada)

Program Chairs:

Keiji Hirata (NTT Communication Science Laboratories, Japan)
George Tzanetakis (University of Victoria, Canada)

Tutorials Chair:

Kunio Kashino (NTT Communication Science Laboratories, Japan)

Finance Chair:

Takuichi Nishimura (AIST, Japan)

Publications Chair:

Kazuyoshi Yoshii (AIST, Japan)

Publicity Chairs:

Hiromasa Fujihara (AIST, Japan)
Hirokazu Kameoka (NTT Communication Science Laboratories, Japan)

Registration & Accommodation Chairs:

Tomoyasu Nakano (AIST, Japan)
Akinori Ito (Tohoku University, Japan)

Local Organizing Committee Chair:

Tetsuro Kitahara (Kwansei Gakuin University, Japan)

Local Organizing Committee:

Takuya Fujishima (Yamaha Corporation, Japan)
Masatoshi Hamanaka (University of Tsukuba, Japan)
Keiichiro Hoashi (KDDI R&D Laboratories, Japan)
Katsutoshi Itoyama (Kyoto University, Japan)
Katsuhiko Kaji (NTT Communication Science Laboratories, Japan)
Kazuhiro Nakadai (Honda Research Institutes, Japan)
Mitsunori Ogihara (University of Miami, USA)

Reviewers

Ahonen, Teppo
Anagnostopoulou, Christina
Anglade, Amélie
Arcos, Josep Lluís
Assayag, Gérard
Aucouturier, Jean-Julien
Auhagen, Wolfgang

Baccigalupo, Claudio
Bainbridge, David
Barbieri, Gabriele
Barrington, Luke
Basu, Sumit
Bauman, Stephan
Bay, Mert
Bello, Juan Pablo
Bertin-Mahieux, Thierry
Betser, Michael
Boehm, Carola
Bosteels, Klaas
Brattico, Elvira
Burred, Juan José
Byrd, Donald

Cambouropoulos, Emilios
Cartwright, Mark
Casagrande, Norman
Casey, Michael
Celma, Òscar
Chew, Elaine
Chordia, Parag
Christensen, Mads Grasboll
Chuan, Ching-Hua
Cohen, Annabel
Cont, Arshia
Cook, Perry
Crawford, Tim
Cunningham, Sally Jo

Dannenberg, Roger
Daudet, Laurent
Davies, Matthew
Davis, Elizabeth
Deliège, François
Devaney, Johanna
Dixon, Simon
Downie, J. Stephen
Duan, Zhiyao
Duane, Ben
Dunn, Jon
Durrieu, Jean-Louis

Eck, Douglas
Eerola, Tuomas
Eggink, Jana
Ehmann, Andreas
Eleftheriadis, Alexandros
Ellis, Dan
Esquef, Paulo
Essid, Slim

Farbood, Morwaread
Fazekas, György
Fields, Benjamin
Fingerhut, Michael
FitzGerald, Derry
Flexer, Arthur
Flossmann, Sebastian
François, Alexandre
Frank, Jakob
Franklin, Judy
Fujihara, Hiromasa

Gärtner, Markus
Gasser, Martin
Gerhard, David
Gómez, Emilia
Gómez, Francisco
Good, Michael
Gouyon, Fabien
Grachten, Maarten
Grecu, Andrei
Green, Stephen
Grindlay, Graham

de Haas, Bas
Han, Jinyu
Harte, Christopher
Heittola, Toni
Helen, Marko
Henry, Stephen
Herrera, Perfecto
Honingh, Aline
Hu, Xiao
Huq, Arefin

Inskip, Charlie
Iñesta, José Manuel
Ito, Akinori
İzmirli, Özgür

Jacobson, Kurt
Jang, Roger
Jehan, Tristan
Jensen, Jesper Højvang
Jensen, Kristoffer
Jones, M. Cameron

Kaji, Katsuhiko
Kameoka, Hirokazu
Kashino, Kunio
Katayose, Haruhiro
Keller, Robert
Kim, Youngmoo
Kitahara, Tetsuro

Klapuri, Anssi
Knees, Peter
Kroger, Pedro
Kurth, Frank
Kuuskankare, Mika

Lagrange, Mathieu
Lamere, Paul
Lanckriet, Gert
Lartillot, Olivier
Lassfolk, Kai
Laurier, Cyril
LeBoeuf, Jay
Lee, Eric
Lee, Jin Ha
Lee, Kyogu
Lemström, Kjell
Leveau, Pierre
Lewis, David
Lidy, Thomas
Little, David
Liutkus, Antoine

Madsen, Søren Tjagvad
Mäkinen, Veli
Mandel, Michael
Mardirossian, Arpi
Marolt, Matija
Marsden, Alan
Martins, Luís Gustavo
Mauch, Matthias
McFee, Brian
McKay, Cory
Mckinney, Martin
Mesaros, Annamaria
Moelants, Dirk
Müller, Meinard

Nakano, Tomoyasu
Nakra, Teresa Marrin
Ng, Kia
Niedermayer, Bernhard
Noland, Katy

Ong, Bee Suan
Orio, Nicola
Oswell, Michelle
Ozerov, Alexey

Pachet, François
Paiva, Rui Pedro
Pampalk, Elias
Pardo, Bryan
Paulus, Jouni
Pauws, Steffen
Peeling, Paul
Peeters, Geoffroy
Pikrakis, Aggelos
Pohle, Tim
Pugin, Laurent

Quinn, Ian

Rafii, Zafar
Raimond, Yves
Raphael, Christopher
Rauber, Andreas
Reed, Jeremy
Reiss, Josh
Rhodes, Christophe
Richard, Gaël
Riley, Jenn
Rizo, David
Röbel, Axel
Rowe, Robert
Roy, Pierre

Sabin, Andrew
Sagayama, Shigeki
Sancho, Carlos Pérez
Sandler, Mark
Sapp, Craig
Schedl, Markus
Schmidt, Erik
Schoner, Bernd
Serra, Xavier
Serrà, Joan
Sethares, William
Seyerlehner, Klaus
Slaney, Malcolm
Smaragdis, Paris
Smit, Christine
Sordo, Mohamed
Spiro, Neta
Suzuki, Kenji

Takeda, Haruto
Temperley, David
Tenkanen, Atte
Terasawa, Hiroko
Tidhar, Dan
Toiviainen, Petri
Tomita, Yo
Turnbull, Douglas
Typke, Rainer

Veltkamp, Remco
Vignoli, Fabio
Vincent, Emmanuel
Virtanen, Tuomas

Wang, Ye
West, Kris
Weyde, Tillman
Whiteley, Nick
Whitman, Brian
Widmer, Gerhard
Wiering, Frans
Wiggins, Geraint

Yoshii, Kazuyoshi

Contents

General Chairs' Preface

We would like to welcome all participants to Kobe, Japan for the 10th International Society for Music Information Retrieval Conference (ISMIR 2009). ISMIR has grown continuously and rapidly in the past 10 years. Starting off as the International Symposium on Music Information Retrieval in Plymouth, USA (2000) and in Bloomington, USA (2001), it was renamed the International Conference on Music Information Retrieval in Paris, France (2002). Since then, six successful conferences have been held in Baltimore, USA (2003), Barcelona, Spain (2004), London, UK (2005), Victoria, Canada (2006), Vienna, Austria (2007), and Philadelphia, USA (2008). Until now, the nine previous conferences have been held in either American or European countries. We are proud to host the first ISMIR in Asia, and are pleased to be able to commemorate the 10th anniversary conference in the international history of ISMIR. Furthermore, ISMIR 2009 is the first conference to take place after the official birth of the "International Society for Music Information Retrieval" (ISMIR) on 4 July 2008. Starting this year, our annual conference is called the International Society for Music Information Retrieval Conference (also, ISMIR).

The motto of ISMIR 2009 is "Celebrating the 10th Anniversary: Increasing Diversity and Toward the Future." With the recent rapid growth of ISMIR conferences and the expansion of the music information retrieval (MIR) community, MIR participants may no longer consider MIR strictly as "music information retrieval," but rather consider more broadly as "music information research." Correspondingly, MIR also draws considerable attention from other research and application fields. ISMIR therefore aims to open new horizons for the exchange and discussion of ideas, issues, results, and perspectives for people of diverse backgrounds. Researchers, developers, educators, librarians, students, and professional users from academia, industry, entertainment, and education all benefit from the diversity of ISMIR. Looking toward the future, such diversity will certainly be increased further.

The conference will take place at the Kobe International Conference Center in Kobe, Japan on 26–30 October 2009 (Monday through Friday). The packed conference program starts on Monday with four tutorials providing in-depth coverage of timely, hot topics such as large-scale data on the Web, the Social Web, music discovery based on visualization, and the Semantic Web. As in previous years, the program continues with scientific presentations during three full days and a half day. This year the conference features two keynote presentations. The first keynote, *Ten Years of ISMIR: Reflections on Challenges and Opportunities*, by J. Stephen Downie, Donald Byrd, and Tim Crawford on Tuesday morning will introduce the 10-year history of ISMIR and provide thoughtful discussions for the future, which will be a great opening for this 10th anniversary. The second keynote, *Wind Instrument-Playing Humanoid Robots*, by Atsuo Takanishi on Thursday morning will introduce humanoid robots that can perform musical instruments, which will reflect Japan's strength in and fascination with the robotic entertainment and industry. The conference this year will also feature an industrial panel discussion by practitioners from companies using MIR techniques, such as The Echo Nest, Microsoft Corporation, Last.fm, Barcelona Music and Audio Technologies (BMAT), Gracenote, KDDI R&D Laboratories, NTT Communication Science Laboratories, and Yahoo! Research.

	Gold sponsor	Silver sponsor	Bronze sponsor
Donation	500,000JPY or 5,000USD	300,000JPY or 3,000USD	100,000JPY or 1,000USD
Exhibition space	Exhibition space during demonstration session		
Logo on conference banner	Large and prominent	Medium	Small
Logo on conference program	Half page	Quarter page	Eighth page
Logo on printed proceedings	Large and separated	Medium and separated	Small
Complimentary registrations	Two	One	None

We shall also carry on two fine traditions of previous ISMIR conferences. For five consecutive years, panel and poster sessions for the annual Music Information Retrieval Evaluation eXchange (MIREX) have taken place. This year these events will take place on Wednesday afternoon. Ever since the Audio Description Contest (ADC) in Barcelona in 2004 followed by MIREXs, they serve as the standard community-based framework for the formal evaluation of algorithms and techniques related to MIR. For the second consecutive year, a late-breaking/demo session without peer review will be held on Friday morning to present preliminary results and ideas and to demonstrate MIR applications.

There will be several new planned activities in addition to being the first Asian ISMIR and society-based conference. The first Annual General Meeting of the International Society for Music Information Retrieval will be held on Thursday afternoon before the banquet. The ISMIR Board of Directors will be elected during this meeting. In addition, the first Workshop on the Future of MIR (f(MIR)) will be held as a special session on Friday morning. This session was proposed and organized only by students and its goal is to discuss what MIR research might be like in 10, 20, or even 50 years. Four oral and two poster presentations were selected through a separate student-run review process independently of the regular ISMIR peer-review process.

This is also the first time that the conference sponsorship program was officially designed at three different levels indicated in the table above. A special word of thanks goes to seven corporate sponsors: Sun Microsystems and Gracenote (Gold sponsors), Yamaha Corporation (Silver sponsor), and KDDI R&D Laboratories, Sony Corporation, KORG Inc., and ProQuest (Bronze sponsors). A part of Sun Microsystems' sponsorship is used for the "Sun Microsystems Student Travel Award" to assist students with travel to ISMIR 2009. The General Chairs of ISMIR 2008, Dan Ellis and Youngmoo Kim, have also donated funds from ISMIR 2008 to significantly increase travel support for students through the "Drexel-ISMIR Student Travel Award." In addition, we gratefully acknowledge financial support from Tsutomu Nakauchi Foundation and "Portopia '81 Memorial Fund" of Kobe Convention & Visitors Association, and service support from the "MEET IN KOBE 21st Century" program.

We would like to thank everyone who contributed to the planning, preparation, and administration of ISMIR 2009, including the members of the Conference Committee and the ISMIR Steering Committee, and, of course, the conference participants and presenters. We hope that ISMIR 2009 will be a fruitful and memorable meeting and we wish you a very pleasant stay in Japan. Let's celebrate the 10th anniversary together!

Masataka Goto and Ichiro Fujinaga
General Chairs, ISMIR 2009

Program Chairs' Preface

We are proud to present the proceedings of ISMIR 2009, the 10th International Society for Music Information Retrieval Conference which will take place in Kobe, Japan. It was with a great sense of responsibility that we accepted to chair the program of this conference. Our goal was to continue building on the strong record of previous ISMIR conferences and support the vibrant interdisciplinary community of researchers and practitioners in the field of Music Information Retrieval (MIR). This was not an easy task as we received a large number of high quality submissions and had to make some difficult decisions assisted by the hard work of the Program Committee (PC) members and the reviewers.

Before starting the reviewing process we had extensive discussions with previous Program chairs and committee members as well as ISMIR participants to learn from their experience and receive suggestions and feedback. Based on their input we decided to have a thorough double-blind reviewing process as well as provide the authors with a chance to respond to the reviewer comments and suggestions. We retained the emphasis on cross-disciplinary and inter-disciplinary research and did our best to ensure that the final program appeals to the widest possible portion of the community. All papers published in the proceedings have uniform status and have up to 6 pages. The mode of presentation (oral or poster) was decided after the accept/reject decisions were made based on the topic and broad appeal of the work rather than being an indication of quality. Because of the double-blind review process we did not impose any authorship limitations. However we did take into account the identity of authors for the oral presentation decisions in order to have a well-balanced public forum for scholars to stand before an international audience to present their ideas.

ISMIR 2009 received a total of 212 submissions from 29 different countries out of which 123 were accepted. 38 were selected for oral presentation and 85 for poster presentation. The table at the top of the next page shows some statistics of the ISMIR conferences up to this year. The table has been produced by adding the ISMIR 2009 data to the table from last year. In 2007, 127 submissions were accepted out of 217 submissions (acceptance rate of 58%), in 2008, 105 out of 175 (acceptance rate 60%) and in 2009, 123 out of 212 (acceptance rate 58%). The 22-member Program Committee and the two Program Chairs coordinated the reviewing efforts of 214 reviewers. Before paper assignments, both reviewers and PC members were invited to indicate their preferences for papers to review. These preferences directly informed the paper assignment process, thus ensuring knowledgeable reviews and assessor confidence. The double-blind reviewing required more effort in resolving conflicts of interest which were handled through a combination of automatic detection as well as input from the PC members. Each submission received at least three reviews and a meta-review by a PC member. Moreover, the authors were given the opportunity to respond to the reviews in the so-called "rebuttal" phase. Their comments were taken into considerations during a final discussion phase between the PC members and reviewers. The final decisions were made by the Program Chairs take into account primarily the recommendation of the PC members and the reviewers as well as relevance to the conference, inter-disciplinary nature and balance of the overall program.

To encourage and recognize excellence in Music Information Retrieval research we decided to give out a best paper award (for which all accepted submissions were considered) and a best student paper award (for

Year	Location	Presentations		Total	Total	Total	Unique	Pages /	Authors /	U. Authors
		Oral	Posters	Papers	Pages	Authors	Authors	Paper	Paper	/ Paper
2000	Plymouth	19	16	35	155	68	63	4.4	1.9	1.8
2001	Indiana	25	16	41	222	100	86	5.4	2.4	2.1
2002	Paris	35	22	57	300	129	117	5.3	2.3	2.1
2003	Baltimore	26	24	50	209	132	111	4.2	2.6	2.2
2004	Barcelona	61	44	105	582	252	214	5.5	2.4	2.0
2005	London	57	57	114	697	316	233	6.1	2.8	2.0
2006	Victoria	59	36	95	397	246	198	4.2	2.6	2.1
2007	Vienna	62	65	127	486	361	267	3.8	2.8	2.1
2008	Philadelphia	24	105	105	630	296	253	6.0	2.8	2.4
2009	**Kobe**	**38**	**85**	**123**	**729**	**375**	**292**	**5.9**	**3.0**	**2.4**

which only accepted submissions with student authors were considered). Based on the reviewer scores five candidate papers were selected for the best paper award and four papers were selected for the best student paper award. PC members as well as the Program Chairs ranked the candidates for each category and the results were tabulated to select the final winner. We are pleased to announce that *Musical Instrument Recognition in Polyphonic Audio Using Source-Filter Model for Sound Separation* by Toni Heittola, Anssi Klapuri and Tuomas Virtanen is the winner of the best paper award and *Easy as CBA: A Simple Probabilistic Model for Tagging Music* by Matthew Hoffman, David Blei and Perry Cook is the winner of the best student paper award.

We both spent many hours agonizing over decisions, reading reviews and discussions and frequently reading the entire submission in order to ensure a diverse, balanced, high-quality program. Unfortunately many good submissions had to be excluded but we were encouraged by some of the authors of rejected submissions who thanked us for the thorough feedback they received. Finally, we sincerely ask that the authors who did not have their submissions accepted this year consider attending ISMIR and submitting in the future. The continuing participation of the ISMIR community will ensure that the tradition of high-quality, inter-disciplinary research continues in forthcoming ISMIR conferences.

To complement the peer-reviewed program, ISMIR 2009 also includes a late-breaking/demo session to showcase preliminary research and technical demonstrations. The 24 submissions received in the form of abstracts are published on the conference website and will be presented as posters on the last day of the conference.

Keiji Hirata and George Tzanetakis
Program Chairs, ISMIR 2009

Keynote Talk 1

Ten Years of ISMIR:

Reflections on Challenges and Opportunities

J. Stephen Downie (University of Illinois at Urbana-Champaign, USA)
Donald Byrd (Indiana University at Bloomington, USA)
Tim Crawford (Goldsmiths College, University of London, UK)

Biographies

J. Stephen Downie is an Associate Professor at the Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign (UIUC). He is Director of the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL). He is Principal Investigator on the Networked Environment for Music Analysis project (NEMA). He has been very active in the establishment of the Music Information Retrieval (MIR) community through his ongoing work with the International Symposium on Music Information Retrieval (ISMIR) conferences as a member of the ISMIR steering committee. He holds a BA (Music Theory and Composition) along with a Master's and a PhD in Library and Information Science, all earned at the University of Western Ontario, London, Canada.

Donald Byrd studied music composition at Indiana University in the late 1960's, and then became interested in computers and their potential to help musicians. After spending a number of years as a programmer and consultant at the University's academic computing support services, he received a PhD in Computer Science with a dissertation on music notation by computer. Since then, He has worked extensively both in industry and academia. He was one of the principal sound designers and sound-design software developers for the Kurzweil 250, arguably the first synthesizer to reproduce sounds of acoustic instruments convincingly. He was also the principal designer of the influential music-notation program Nightingale. His academic background includes research on music notation by computer (at Princeton University); work on information retrieval in text, especially visualization and human/computer interaction aspects (at the University of Massachusetts); and work on music information retrieval, digital music libraries, and optical music recognition (at the University of Massachusetts and Indiana University). Most recently, he has been working on the "General Temporal Workbench," a timeline-based system for visualizing, exploring, creating, and "playing" temporal phenomena: a system general enough for use on any timescale from fractions of an attosecond to billions of years. He is currently senior scientist and adjunct associate professor in the School of Informatics at IU.

Tim Crawford is a member of the Intelligent Sound and Music Systems group in the Computing Department at Goldsmiths College, University of London. He worked for 15 years as a professional musician before turning to academic research. He is active as a musicologist, being internationally recognized as a leading authority on the history and music of the European lute, and is currently Editor of the Complete Works of the lutenist Silvius Leopold Weiss (1687-1750). Otherwise he is mostly engaged in the application of computational methods to music-related research. He managed the UK effort for the original OMRAS project (Online Music Recognition and Searching, 1999-2003), which was the precursor of the currently-running OMRAS2 project on which he currently works. He also conceived, led and managed ECOLM (Electronic Corpus of Lute Music, 1999-2006), and is currently Principal Investigator of the Purcell Plus project which is investigating the application of eScience in musicology and the longer-term methodological implications of technology for the discipline. He is one of the founders of ISMIR and frequent contributor as author or organizer and has recently jointly edited one of a series of books on "Humanities Computing: Modern Methods for Musicology: Prospects, Proposals and Realities," ISBN 978-0-7546-7302-6 (Farnham: Ashgate 2009), in which several ISMIR authors are represented.

TEN YEARS OF ISMIR: REFLECTIONS ON CHALLENGES AND OPPORTUNITIES

J. Stephen Downie
University of Illinois
at Urbana-Champaign
jdownie@illinois.edu

Donald Byrd
Indiana University
at Bloomington
donbyrd@indiana.edu

Tim Crawford
Goldsmiths College
University of London
t.crawford@gold.ac.uk

ABSTRACT

The *International Symposium on Music Information Retrieval* (ISMIR) was born on 13 August 1999. This paper expresses the opinions of three of ISMIR's founders as they reflect upon what has happened during its first decade. The paper provides the background context for the events that led to the establishment of ISMIR. We highlight the first ISMIR, held in Plymouth, MA in October of 2000, and use it to elucidate key trends that have influenced subsequent ISMIRs. Indicators of growth and success drawn from ISMIR publication data are presented. The role that the Music Information Retrieval Evaluation eXchange (MIREX) has played at ISMIR is examined. The factors contributing to ISMIR's growth and success are also enumerated. The paper concludes with a set of challenges and opportunities that the newly formed *International Society for Music Information Retrieval* should embrace to ensure the future vitality of the conference series and the ISMIR community.

1. ORIGINS OF ISMIR

In mid-August 1999, Byrd and Downie were at the Radisson Hotel Berkeley Marina conference center in Berkeley, California: Byrd for the ACM (Association for Computing Machinery) Digital Library Conference (DL '99), Downie for the ACM SIGIR conference, which immediately followed DL '99. We had not met before, but our paths had been converging for some time, and in retrospect, it is hardly surprising that something special came out of our face-to-face encounter. Crawford was in England at the time, but he and Byrd had been collaborating since the early 1990s. Crawford and Byrd had recently received word that their "Online Music Recognition and Searching" (OMRAS) project [1] would be jointly funded by the Joint Information Systems Committee (JISC) of the UK and the National Science Foundation (NSF) of the USA. Steve Griffin, the project's NSF program officer, had already suggested to Byrd and Crawford independently that a music-IR workshop be organized in conjunction with OMRAS. Furthermore, Crawford was organizing another workshop on music IR, as part of the "Digital Resources for the Humanities" conference to be held in London in September 1999. Finally, Downie, with the assistance of David Huron (Ohio State University) and Craig Nevill-Manning (then of Rutgers University), had organized "The Exploratory Workshop on Music Information Retrieval" at SIGIR '99.[1] Before going to Berkeley, Downie was already thinking of a larger-scale follow-up event as this was an explicit goal of his SIGIR workshop. One of the workshop presenters, Michael Fingerhut of IRCAM, would later play a pivotal role in the success of ISMIR through his establishment and maintenance of vital community resources (see Section 3.2).

With the encouragement of Bruce Croft (University of Massachusetts, Amherst)—a very well-known researcher in the text IR world, and Byrd's boss at the time—Downie and Byrd decided on the spot to join forces to plan a larger-scale event instead of a workshop in the normal sense, and they came up with the name "International Symposium on Music Information Retrieval."

Most of the above has been described in print before [2]. Previously unreported, however, are some informal meetings convened in Berkeley, which variously included Byrd, Downie, Nevill-Manning, David Bainbridge (University of Waikato), Matthew Dovey (University of Oxford), and Massimo Melucci (University of Padua). It is interesting that Byrd's notes of these meetings show a heavy emphasis on music in symbolic form over audio, and quite a bit of discussion of TREC[2]-like evaluations of music-IR systems.

1.1 What's in a Name?: Evolution of "ISMIR"

The ISMIR acronym, decided upon during the August 1999 meetings, was carefully crafted. First, both Byrd and Downie wanted to strongly encourage the participation of researchers from around the world, so *International* was chosen without hesitation. Second, the word *symposium* has its roots in the Greek verb, *sympotein*,

[1] See http://nema.lis.uiuc.edu/sigir99_mir_wshop.pdf.

[2] The Text Retrieval Conference upon which MIREX is based.

which means "to drink together." [1] As many know, Downie is particularly fond of symposia. Besides its social connotations, the term *Symposium* was agreed upon as it indicated a certain academic middle-ground between a workshop and a full-fledged conference. Before long, however, some participants noted that they were having difficulties obtaining travel funding to attend "a mere symposium," and in 2002 ISMIR became the "International *Conference* on Music Information Retrieval." Over the years, ISMIR organizers explored affiliation opportunities with such organizations as the Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), and the International Computer Music Association (ICMA); none worked out. Undeterred, Ichiro Fujinaga of McGill University led the way to formally establishing ISMIR as an independent society. On 4 July 2008, the "International *Society* for Music Information Retrieval" was officially born. By the time ISMIR 2009 in Kobe concludes, the music-IR community will have elected its first roster of ISMIR executive officers and held its first Annual General Meeting.

2. ISMIR 2000 AT PLYMOUTH, MA: LANDING OF THE MUSIC-IR PILGRIMS

In accordance with the events of 1999 described above, ISMIR 2000[2] was held in Plymouth, Massachusetts (the site of the Pilgrims' 1620 arrival in the New World) from 23 to 25 October 2000. Byrd was general chair and Downie was program chair. The other organizing committee members were Crawford, Croft, and Nevill-Manning. In addition, Jeremy Pickens, then a PhD student working on the OMRAS project, became, by virtue of his good nature, *the* local organizer—i.e., audio-visual person and general helper—during the conference.

In terms of statistics, 88 people attended ISMIR 2000: not bad at all for a first conference in the field, and about twice the attendance at the first computer-music conference (which Byrd had attended in 1974). Furthermore, attendance was already very international: 29 attendees (33%) came from 11 countries outside the United States. ISIMIR 2000 was very heavy on invited papers, of which there were nine. An additional 33 papers were submitted; 10 were accepted as papers, 16 as posters.

2.1 ISMIR 2000: Highlights and Commentary

Many of the intellectual themes, challenges, and opportunities that would resonate throughout subsequent convenings of ISMIR were already evident in Plymouth. To illustrate this, a selection of ISMIR 2000 highlights with *editorial comments* follows:

- Marvin Minsky delivered the keynote address. *His talk was uniquely creative and pointed out several connections that are still relevant, e.g., to artificial intelligence, improvisation vs. written-out music, and even to his institution, MIT.*
- Beth Logan gave one of the first papers formally examining the implications of using Mel Frequency Cepstral Coefficients (MFCC) for music; this created a fair amount of controversy. *We wish we had a penny for each MFCC calculated since 2000!*
- There were two papers on music digital library applications: Jon Dunn spoke on the "Variations" system; David Bainbridge talked about the "New Zealand Digital Music Library." *Downie, as a library science professor, notes with some sadness that the digital library theme has not gained much traction in subsequent ISMIRs.*
- Byrd, Crawford, and Steve Larson led a "Lecture, Recital, Discussion, and Survey" session on music similarity. Centered on Mozart's piano piece *Variations on Ah! Vous dirai-je, maman* (the melody English speakers call "Twinkle, Twinkle, Little Star"), Larson played the piece, and attendees filled out survey forms to say how similar they felt each of the selected variations was to the theme. *This session led to our choosing three measures from the Mozart variations for the ISMIR logo (Figure 1). The "similarity problem" remains a huge challenge, not least because of the difficulty of establishing "ground-truth" in this subjective area.*

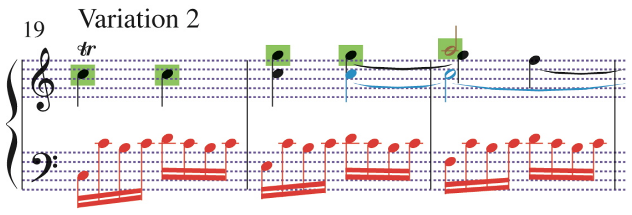


Figure 1. The Mozart-based official ISMIR logo.

- Mary Levering of the U.S. Patent and Trademark Office talked about "Intellectual Property Rights in Musical Works." *This is a problem that continues to plague many music-related activities, including music-IR research.*
- George Tzanetakis and Perry Cook gave a paper on audio-IR tools. *Tzanetakis's MARSYAS is now one of the most widely used music-IR toolkits.*
- Jonathan Foote gave a paper on recognizing pieces of orchestral music regardless of performance differences. *Foote's approach looked solely at low-level (though long-term) audio features. Numerous music-IR papers since 2000 ignore musical knowledge and instead employ low-level features that seem to work; this paper foreshadows the trend.*
- There were papers on musicology applications, transcription from audio, retrieval from audio, Optical Mu-

[1] See http://en.wikipedia.org/wiki/Symposium.
[2] See http://ismir2000.ismir.net/.

sic Recognition (OMR), language modeling, and XML representation of music notation. *All of these except language modeling have been the subjects of numerous ISMIR papers since.*

- Eric Allamanche of Fraunhofer gave an informal demo of an audio fingerprinting application designed to identify broadcast music in real time. *Similar systems are now widely available in the commercial sphere.*
- The creators of MusicXML and MEI each gave posters on early versions of their representations. *Since 2000 MusicXML has become the most popular XML form for music content, but MEI has recently been the subject of development for specific applications, particularly in the musicological domain.*
- Suzanne Lodato of the Andrew W. Mellon Foundation took an active role in the plenary planning and discussions sessions of the symposium. *The Mellon Foundation would go on to provide critical funding to prepare for, establish, and run MIREX.*

Of the 19 full papers presented, six were mostly or entirely on audio; nine mostly or entirely on music in symbolic form (including metadata); and four about equally on audio and symbolic music. As is obvious to anyone who has attended the last five or six ISMIRs, the predominance of symbolic music (reflecting the backgrounds of the original organizers) has not persisted; we will say more about this later.

Despite the inexperience of the organizers and the novelty of the subject area, ISMIR 2000 was universally regarded as a resounding success.

3. SUCCESS AND GROWTH OF ISMIR: 2000–2009

The ongoing success and growth of ISMIR since 2000 is both remarkable and encouraging. The vitality of the community is readily apparent from even the most cursory examination of the statistics. For example, Table 1 presents the number of published items (both posters and papers), number of pages published, and the number of unique authors represented in the proceedings of each ISMIR from 2000 to 2009. The table shows a 251% increase in the number of published items per year, from 35 to 123. The number of pages went up even more: 155 to 729 is a 370% increase.

The number of unique authors represented also grew tremendously, by 363% from 2000 (63) to 2009 (292). On average, 183 unique authors made contributions to each of the 10 ISMIRs under consideration. For us, the growth in the number of unique authors is the best statistic of the set, since it indicates that ISMIR has attracted the most important asset of any conference: active, engaged and publishing researchers.

Mailing list statistics also confirm ISMIR's success. The *music-ir@ircam.fr* list, established in October 2000, is the ISMIR community's primary communications mechanism. This list, as of 22 August 2009, has 1190 registered subscriptions. It has broadcast nearly 3000 messages for an average of 28 per month. These are strong numbers for such a specialized research area as music IR.

YEAR	LOCATION	ITEMS	PAGES	UNIQUE AUTHORS
2000	Plymouth, MA	35	155	63
2001	Bloomington, IN	41	222	86
2002	Paris, FR	57	300	117
2003	Baltimore, MD	50	209	111
2004	Barcelona, ES	105	582	214
2005	London, UK	114	697	233
2006	Victoria, BC	95	397	198
2007	Vienna, AT	127	486	267
2008	Philadelphia, PA	105	630	253
2009	Kobe, JP	123	729	292
TOTALS	----	852	4407	----

Table 1. ISMIR publication and author data 2000–2009.[1]

3.1 The Audio Description Contest and MIREX

As mentioned in Section 1, the formal evaluation of music-IR systems has been part of the ISMIR "wish list" since its inception. However, notwithstanding strong community interest, it was surprisingly difficult to institute a formal evaluation framework along the lines of TREC. There were many challenges to overcome, the greatest of which was the lack of high-quality test collection and ground-truth data caused primarily by the very restrictive intellectual property regimes governing music [4]. After a series of exploratory workshops led by Downie and funded by Mellon and NSF [5], the organizers of ISMIR 2004 in Barcelona were able to put together the "Audio Description Contest" (ADC) [6]. Many valuable lessons were learned in the running of ADC and these were subsequently incorporated into MIREX. After receiving substantial long-term funding from both Mellon and NSF, MIREX established itself as a permanent fixture in time for ISMIR 2005 [7].

	2005	2006	2007	2008
Number of Task (and Subtask) "Sets"	10	13	12	18
Number of Individuals	82	50	73	84
Number of Countries	19	14	15	19
Number of Runs	86	92	122	169

Table 2. MIREX descriptive data 2005–2008 [8].

Like the publication data examined previously, the MIREX data are quite encouraging. Table 2 summarizes the key descriptive data for MIREX between 2005 and

[1] 2000–2008 data sourced from the *Preface* of the ISMIR 2008 proceedings [3].

2008. A fuller explication of the MIREX data can be found in [4, 8].

The number of task and subtask sets grew by 80% from 2005 (10) to 2008 (18). This growth can be attributed to growing interest in MIREX and the donation of new high-quality data sets from community members.

In keeping with ISMIR's international mission, the number of countries represented was a strong, but flat, 19 for both 2005 and 2008 with an average of 17 per year. Most of these numbers come from European countries with Japan, China, and Taiwan also represented. Likewise, the number of individual participants has not appreciably increased between 2005 (82) and 2008 (84). We do note the lack of growth in the country and participant numbers as something that needs addressing.

The most heartening MIREX statistic concerns the number of individual runs performed: this went from 86 to 169, an increase of 96%. Note that the increase is greater than the increases in both participants and tasks: participants are more likely now to submit multiple variations on their algorithms. This fact suggests to us that MIREX has been successful in its message that MIREX exists as an exploratory mechanism designed to try out new ideas and *not* a "contest" to be won or lost.

In total, MIREX has run 469 algorithms. It is interesting to note the distribution of runs over areas of interest:

- 129 (28%) can be categorized as "train-test" machine-learning classification experiments (e.g., Audio Genre Classification, Audio Mood Classification, etc.).
- 139 (30%) can be categorized as "search" experiments (e.g., Audio Cover Song Identification, Audio Music Similarity, etc.)
- 201 (43%) can be categorized as "low-level" feature experiments (e.g., Audio Onset Detection, Audio Beat Tracking, etc.)

We must also note that, of the 22 unique task sets run over 2005 to 2008, only three (14%) have dealt exclusively with symbolic music data (i.e., Symbolic Genre Classification, Symbolic Key Finding, and Symbolic Melodic Similarity). Not one of these symbolic tasks was run in 2008 and not one proposed for MIREX 2009. 16 task sets (73%) have been exclusively audio-based (e.g., Audio Tempo Extraction, Audio Key Finding, etc.), and three tasks have involved a combination of audio and symbolic data (i.e., Query-by-Singing/Humming, Query-by-Tapping, and Score Following). As these data show, MIREX has been quite successful in growing evaluation activity in the audio domain, but not at all successful in helping the symbolic sub-community to flourish: this is perhaps MIREX's most serious weakness.

3.2 Success and Growth Factors

Many factors have contributed to the success and growth of ISMIR over the years. These factors are both external and internal to ISMIR. Like many things in life, ISMIR has been successful through a combination of good timing, thoughtfulness, and hard work.

From the beginning, ISMIR's timing was good; it has benefitted from several important external opportunities and trends that developed in parallel. These developments have provided ISMIR with a larger body of researchers and research themes to draw upon than we could have anticipated, especially in the audio domain. We believe these external factors include:

- The success of the audio compression research community in developing techniques specifically designed for, and tested against, music. It was this success and the subsequent acceptance of these approaches that afforded the opportunity to create, share, and store large collections of music audio.
- The explosive growth in the availability of audio files, mostly MP3's, via the Internet. This growth resulted to a great extent from the audio-compression research described above, but in turn it created a demand for better search and retrieval mechanisms. Napster, for example, was established in 1999.
- The work of such standards bodies as the MPEG-7 group that brought together important industry players with leading academic research groups. The MPEG-7 first working draft came out in December 1999.[1]
- The success of such search engines as Google, Yahoo, etc., that encouraged researchers to seek fame and fortune in the music domain. The great "dot.com bubble" of 1998–2001 was contemporaneous with ISMIR's early development.

The internal factors that have contributed to ISMIR's success are founded upon the thoughtful actions, goodwill, and hard work of community members acting either as individuals, in small groups, or collectively. These factors include:

- The establishment of the communication resources housed at IRCAM. The *music-ir@ircam.fr* mailing list, the hosting of the conference websites, and the archiving of the collected ISMIR proceedings are resources without which ISMIR might not exist today. Each of these has contributed inestimably to the openness, continuity, and intellectual life of the ISMIR community. We applaud Michael Fingerhut for his continued service.
- The diversity of backgrounds and disciplines represented on the ISMIR Steering Committee (SC). The SC has worked hard over the years to ensure that the broadest possible range of research interests is present at each ISMIR. Ichiro Fujinaga has been the SC's coordinator for years, and he is especially commended for his ability to guide the SC through its deliberations.
- The great fortune ISMIR has had in the quality of the chairs and program committee (PC) members for each

[1] See http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

conference. We have nothing but praise for the ISMIR PC teams; each ISMIR has been organized and run with enthusiasm, integrity, and efficiency.

- The implicit policy of inclusiveness that has pervaded the conference programming ethos of each ISMIR. Unlike other technology-related conference series, ISMIR has not measured its intrinsic value through high rejection rates. In fact, the ISMIR PCs are to be applauded for finding mechanisms like expanded poster presentation opportunities to allow for the maximum level of participation yet maintaining academic research quality through strong peer-reviewing. We believe that it is precisely this policy of inclusiveness that has allowed for the all-important growth in unique author participation noted in Section 3. The ISMIR community as a whole is also to be praised for its consistent efforts to make the peer-review process simultaneously as fair, open-minded, and rigorous as possible.
- The ongoing PC and general community support for ADC and MIREX. This support has contributed to ISMIR by fostering a sense of common purpose and exploration among researchers in many of ISMIR's sub-fields. MIREX has also helped to set standards in many sub-fields for what constitutes proper evaluation. Finally, MIREX has provided an extra opportunity for participation in ISMIR for those researchers whose work could not be included in the official proceedings. We must acknowledge here the extra-special efforts made by Kris West, M. Cameron Jones, Andreas F. Ehmann, and Mert Bay in making MIREX run well.

4. CHALLENGES AND OPPORTUNITIES

In its first 10 years, ISMIR has grown into a vibrant and enthusiastic research community. We now need to turn our attention to making ISMIR's next 10 years, its "teen" years, even more rewarding and successful. Like a teenager, ISMIR will undoubtedly stop growing in size at some point; this is only natural. But if ISMIR—both as a conference series and as a society—is to have a successful "adulthood," it will need to address some challenges that it has not fully engaged with before. It must recast these challenges as opportunities and engage them with its growing maturity and its youthful vigor. Five of the most important challenges are:

1. ISMIR needs to more actively encourage the participation of potential users of music-IR systems. Notwithstanding the laudable efforts made by the ISMIR Steering Committee, ISMIR has tended to focus much less on the potential *users* of music-IR technology than on its developers. These users might include, for example, performing musicians, film-makers, musicologists, music librarians, sound archivists, music educators, and music enthusiasts of all types. The knowledge acquired by interacting with users like these can only improve the quality of the community's research output. It will also go a long way to helping ISMIR researchers create truly useful music-IR systems.

2. ISMIR research projects must dig deeper into the music itself. Notwithstanding some recent—and heartening—developments in such areas as, for example, chord detection, cover song detection, and structural analysis, etc., a large amount of ISMIR research effort, especially in timbre-based audio matching, has gone into attempts to enhance a few basic features and matching algorithms. However, it seems likely that there is a point beyond which improved matching performance using any single feature cannot be achieved [9]. On the other hand, the incorporation of multiple features in what might be thought of as "hybrid" matching tends to be more successful. But such combining of features needs to be done in a way that is understood and principled, and much more research needs to be done in understanding what such combinations *actually represent in musical terms*. The integration of symbolic music data to create hybrid audio + symbolic music-IR systems could help in this regard.

3. Time has come for ISMIR to expand its musical horizons. The vast majority of ISMIR's collective music-IR research has been conducted on Western popular musics of the late-20th and early-21st centuries. This is a serious problem because there is an enormous amount of music in existence that is utterly different from these corpora. There is no reason to assume algorithms that work superbly for the *Beach Boys* will do anything useful with Tuvan throat singing, *musique concrète*, or Indian Raga.

4. ISMIR must rebalance the portfolio of music information types with which it engages. Music information is inherently multifaceted. Each of its manifestations—audio, symbolic, and metadata—contributes different but equally important features to the experience of music. We celebrate the accomplishments of ISMIR's audio researchers but, as noted before, research exploiting the symbolic aspects of music information has not thrived under ISMIR. We are thrilled to see, however, the growing body of work that strives to unite social metadata and audio information. Rather than "pushing down" on the audio side of ISMIR research, we challenge ISMIR to make special efforts to "pull up" symbolic and metadata research to create a more productive, synergistic, and harmonious balance among the three.

5. ISMIR must encourage the development and deployment of full-featured, multifaceted, robust, and scalable music-IR systems with helpful user-interfaces. During ISMIR's first decade, we have seen a great deal of effort expended on the development of the various sub-components of music-IR systems. Unfortunately, we have not yet seen much in the way of a successful integration of these sub-systems into real-world-useable re-

sources. This state of affairs cannot be sustained for the next decade as the community needs these full-featured systems to exist in order to inspire the development of the next generation of refinements and improvements. In the text IR world, and starting in the 1960s, such systems as "SMART,"[1] "Managing Gigabytes,"[2] and "Terrier,"[3] have fulfilled this important, if not imperative, role.

4.1 The Grand Challenge

We see our "complete system" challenge as "*The* Grand Challenge" for ISMIR's second decade. By embracing this challenge, the preceding ones will necessarily have to be engaged. We do recognize, however, that meeting this "Grand Challenge" will not be easy. We believe there will be difficulties because academic researchers traditionally have obtained little academic credit for comprehensive system development. Future ISMIR program committees need to find a mechanism through which the developers of such systems can acquire full academic credit for accomplishments. One possibility is to have ISMIR create a rigorous set of peer-reviewing criteria specifically designed to handle this type of work. Along these lines, the demonstration of complete systems should receive the same status now afforded to paper presentations. Special awards should also be considered.

5. CLOSING REMARKS

As we noted in the beginning of this paper, the founders of ISMIR, because of their backgrounds, had conceived of music IR as an intersection of music and symbolic IR techniques. As early as ISMIR 2000, it became readily apparent that this conception was much, much too limiting. ISMIR research papers now cover a wide range of activities and recent "Calls for Papers" have reflected this broadening of scope explicitly. We now challenge the ISMIR community to consider whether the term "music IR" has outlived its usefulness. Is it possible that "information retrieval" is too narrow a concept to fully encapsulate what ISMIR researchers actually do? Byrd has proposed several times making the "R" in "ISMIR" stand for "Research" instead of "Retrieval" which could better describe the breadth of the organization without losing ISMIR's name recognition. A related idea is to refer to "music informatics" instead of "music information."

We will leave these questions open in the hope that they will inspire some healthy, self-reflective, debate about the future of ISMIR. It will be through such reflections that ISMIR will continue to be vibrant, energetic, and successful well past its second decade.

[1] See http://en.wikipedia.org/wiki/SMART_Information_Retrieval_System.
[2] See http://www.cs.mu.oz.au/mg/.
[3] See http://ir.dcs.gla.ac.uk/terrier/.

6. ACKNOWLEDGEMENTS

We thank Jeremy Pickens, Michael Casey, Ichiro Fujinaga, Masataka Goto, Keiji Hirata, and the Kobe 2009 Organizing Committee, for their assistance in writing this paper. We also thank the NSF and the Mellon Foundation for their financial support of MIREX. We are especially grateful to Steve Griffin of NSF for the financial support he arranged for ISMIR 2000 and 2001.

7. REFERENCES

[1] D. Byrd and T. Crawford. "Problems of Music Information Retrieval in the Real World," *Information Processing and Management*, Vol. 38, No. 2, pp. 249–272, 2002.

[2] D. Byrd and M. Fingerhut. "The History of ISMIR - A Short Happy Tale." *D-Lib Magazine*, Vol. 8, No. 11, 2002. See http://www.dlib.org/dlib/november02/11inbrief.html#BYRD.

[3] D. P. W. Ellis, Y. Kim, J. P. Bello, and E. Chew. "Preface," *Proceedings of the International Conference on Music Information Retrieval (ISMIR 2008)*, pp. 9–11, 2008.

[4] J. S. Downie. "The Music Information Retrieval Evaluation Exchange (2005-2007): A Window into Music Information Retrieval Research," *Acoustical Science and Technology*, Vol. 29, No. 4, pp. 247–255, 2008. See http://dx.doi.org/10.1250/ast.29.247.

[5] J. S. Downie. "The Scientific Evaluation of Music Information Retrieval Systems: Foundations and Future," *Computer Music Journal*, Vol. 28, No. 3, pp. 12–23, 2004.

[6] P. Cano, E. Gomez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack. *ISMIR 2004 Audio Description Contest. MTG Technical Report, MTG-TR-2006-02*, Music Technology Group, Barcelona, 2004.

[7] J. S. Downie, K. West, A. F. Ehmman, and E. Vincent. "The 2005 Music Information Retrieval Evaluation eXchange (MIREX 2005): Preliminary Overview," *Proceedings of the International Conference on Music Information Retrieval (ISMIR 2005),* pp. 320–323, 2005.

[8] J. S. Downie, A. F. Ehmann, M. Bay, and M. C. Jones. "The Music Information Retrieval Evaluation eXchange: Some Observations and Insights," *Advances in Music Information Retrieval,* Springer, New York, in press.

[9] J.-J. Aucouturier and F. Pachet, "Improving Timbre Similarity: How High is the Sky?" *Journal of Negative Results in Speech and Audio Sciences*, Vol. 1, 2004. See http://www.csl.sony.fr/downloads/papers/uploads/aucouturier-04b.pdf.

Keynote Talk 2

Wind Instrument-Playing Humanoid Robots

Atsuo Takanishi (Waseda University, Japan)

Abstract

Even though the market size is still small at this moment, applied fields of robots are gradually spreading from the manufacturing industry to others in recent years. One can now easily expect that applications of robots will expand into the first and the third industrial fields as one of the important components to support our society in the 21st century. There are also strong anticipations in Japan that robots for the personal use will coexist with humans and provide support such as assistance for housework and care for the aged and the physically handicapped, since Japan is one of the fastest aging societies in the world. Consequently, humanoid robots and/or animaloid robots have been treated as subjects of robotics researches in Japan such as a research tool for human/animal science, an entertainment/mental-commit robot or an assistant/agent for humans in the human living environment. Over the last couple of years, some manufactures including famous global companies started to develop prototypes or even to sell mass-produced robots for the purposes mentioned above, such as HONDA, TOYOTA, Mitsubishi Heavy, TMSUK, etc. On the other hand, Waseda University, where the author belongs to, has been one of the leading research sites on humanoid robot research since the late Prof. Ichiro Kato and his colleagues started the WABOT (WAseda roBOT) Projects and developed the historical humanoid robots that were WABOT-1 and WABOT-2 in the early 70s and 80s respectively. One of the most important aspects of our research philosophy is as follows: By constructing anthropomorphic/humanoid robots that function and behave like a human, we are attempting to develop the design method of humanoid robots to coexist with humans naturally and symbiotically, as well as to scientifically build not only the physical model of a human but also its mental model from the engineering view point. Based upon the philosophy, I and my colleagues have been developing the flute-playing humanoid robots as WF (Waseda Flutist) series as well as the bipedal walking robots WABIAN series, the emotion expression robots WE series and the talking robots WT series, etc. Especially, the purpose of the flute playing robot research is to build the model of the human flute play and to clarify the model from the engineering viewpoint by reproducing the human-like flute play using a humanoid robot having the human-like respiratory organs for the flute play. By using the robot, we will be able to experimentally confirm the model of the human flute play quantitatively. The flute-playing robot/model is useful for the flute-playing beginners to show how to use/move the organs or it will be used for the evaluation of the flute instrument production in the industry. We also started the development of saxophone-playing humanoid robots recently. In my keynote talk, I will introduce the research philosophy of my humanoid robots in general by showing examples, the technical aspects of the wind instrument playing humanoid robots, and the other humanoid robots related to music.

Biography

Atsuo Takanishi is a Professor of the Department of Modern Mechanical Engineering, Waseda University and a concurrent Professor and one of the core members of the HRI (Humanoid Robotics Institute), Waseda University. He received the B.S.E. degree in 1980, the M.S.E. degree in 1982 and the Ph.D. degree in 1988, all in Mechanical Engineering from Waseda University.

His current researches are related to Humanoid Robots and its applications in medicine and well-being, such as the biped walking robots for modeling human biped walking as WABIAN (WAseda BIpedal humA-Noid) series, the biped locomotors for carrying handicapped or elders as WL (Waseda Leg) series, the mastication robots WJ (Waseda Jaw) series to mechanically simulate human mastication for clarifying the hypotheses in dentistry, the jaw opening-closing trainer robots WY (Waseda Yamanashi) series for patients having difficulties in jaw opening or closing, the flute-playing robots as WF (Waseda Flutist) series and the saxophone-playing robots WS (Waseda Saxophonist) series to quantitatively analyze human flute/saxophone playing by collaborating with a professional flutists/saxophonists, and the anthropomorphic talking robots WT (Waseda Talker) series which mechanically speak Japanese vowels and consonant sounds, and the other robots/systems related to his research area. His interest in humanoid robots has extended to the emotion of human that he started to develop the emotion expression humanoid robots WE (Waseda Eye) series and KOBIAN/HABIAN which emotionally behave like a human based upon the "Equations of Emotion." His humanoid robot WABIAN-2R was exhibited in the 2005 World Exposition in Aichi, Japan to demonstrate the knee extended walking using the human-like pelvis and seven DOF leg mechanisms. The emotion expression humanoid KOBIAN is developed based on WABIAN-2R. The latest model WL-16 carries humans and virtually any heavy load weighing up to 80 kg. This project is aiming at developing a practical personal vehicle which supports the society of Japan rapidly becoming an aging society. He recently developed suture/ligature evaluation system WKS series which shows surgeon trainees the quantitative scores of their suture/ligature skills. This system is commercially available from a medical model and training simulator company, Kyoto Kagaku Co. Ltd., in Japan. He is also developing the airway management robot for anesthetist/paramedic trainees collaborating with the company. Refer to www.takanishi.mech.waseda.ac.jp for more details.

He is a member of Robotics Society of Japan (a board member in 1992 and 1993), Japanese Society of Biomechanisms, Japanese Society of Mechanical Engineers, Japanese Society of Instrument and Control Engineers and Society of Mastication Systems (a major board member from 1996 to current), IEEE and other medicine and dentistry related societies in Japan.

He received the Best Paper Award from Robotic Society Japan (RSJ) two times in 1998 and in 2005, the Finalist of Best Paper Award two times in the IEEE International Conference on Robotics and Automation (ICRA) in 1999 and in 2006, the Best of Asia Award from BusinessWeek Magazine in 2001, the Distinguished Research Activity Award in Robotics and Mechatronics from Japan Society of Mechanical Engineers (JSME) in 2003, the Best Paper Award – Application in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in 2004, the Excellent Research Award in 2005 from the Japan Society for Artificial Intelligence (JSAI), the Industrial Application Division Promotion Award in 2005 from the Society of Instrument and Control Engineers (SICE), the Best Paper Award in 2006 from JSME, the Best Conference Paper Award in IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) in 2009, etc.

Website: http://www.takanishi.mech.waseda.ac.jp

Panel Discussion

Industrial Panel Discussion

Organizer

Paul Lamere (The Echo Nest, USA)

Panelists

Tom Butcher (Microsoft Corporation, USA)
Norman Casagrande (Last.fm, UK)
Òscar Celma (Barcelona Music and Audio Technologies, Spain)
Markus Cremer (Gracenote, USA)
Keiichiro Hoashi (KDDI R&D Laboratories, Japan)
Kunio Kashino (NTT Communication Science Laboratories, Japan)
Malcolm Slaney (Yahoo! Research, USA)

Abstract

In this panel practitioners from industry will discuss how their companies are currently using music information retrieval (MIR) techniques to solve problems for their customers. Panelists will also discuss emerging areas of MIR research that are particularly relevant for commercial applications. Audience members will have the opportunity to ask questions of the panelists.

Biographies

Paul Lamere is the Director of Developer Community at The Echo Nest, a research-focused music intelligence startup that provides music information services to developers and partners through a data mining and machine listening platform. He is especially interested in hybrid music recommenders and using visualizations to aid music discovery.

Tom Butcher joined Microsoft in 2006 to build large-scale web services for computing and delivering media experiences. His interests include digital media, artificial intelligence, the Internet, and various intersections thereof. Currently, He is a senior engineer in the Zune group at Microsoft creating data-driven media experiences, which include automatic playlist generation, social discovery, recommendations, and more. Prior to joining Zune, his work encompassed automatic tagging, indexing, and recommendations at MSN Video. An avid music enthusiast, he records electronic music in his spurious free time using the moniker Codebase.

Norman Casagrande joined Last.fm in 2006 as the head of music research. Since then he has been working on a wide range of problems, including collaborative filtering for user/item similarity and recommendation, dealing with scalability, dynamic playlist generation, users insight, audio and semantic analysis, fingerprint, spam fighting, and many other related topics.

Òscar Celma is the Chief Innovation Officer at Barcelona Music and Audio Technologies (BMAT), a spin-off of the Music Technology Group (MTG). BMAT offers solutions for music discovery and recommendation, musical edutainment, and music copyright detection. In 2008, he obtained his PhD in Computer Science and Digital Communication, in the Pompeu Fabra University (Barcelona, Spain). He worked in the MTG from 2000 until 2008 as a Researcher and Project Manager. In 2006, he received the 2nd prize in the International Semantic Web Challenge for the system named “Foafing the Music,” a personalized music recommendation and discovery application.

Markus Cremer joined Fraunhofer Institute for Integrated Circuits (IIS) in 1996 after graduating from Friedrich-Alexander University in Erlangen, Germany, where he contributed to the design of embedded audio codec architectures and digital radio broadcast systems. In 2000, he co-founded the department Metadata at the Fraunhofer Institute for Digital Media Technology in Ilmenau, Germany. Since 2005, he has been directing Gracenote’s Media Technology Lab in Emeryville, California. He is a member of IEEE, ACM, and AES, respectively.

Keiichiro Hoashi joined KDDI R&D Laboratories in 1997. His main research interest is in the area of content-based multimedia information analysis and retrieval, namely music, images, and video. Currently, he is working to implement multimedia content analysis technologies in practical applications and services. He is also working on research projects in data mining, and recommender systems. He was a lecturer at Waseda University from 2002 and 2005, and has received his Dr. Eng. degree from Waseda University in 2007.

Kunio Kashino is Distinguished Technical Member, Supervisor, leading Media-search Research Team at NTT Communication Science Laboratories, and Visiting Professor at National Institute of Informatics (NII), Japan. His team has been working on audio and video analysis, search, retrieval, and recognition algorithms. Its activities include development of basic theories as well as their commercial applications such as Internet content monitoring. He received his PhD from University of Tokyo for his work on “music scene analysis” in 1995.

Malcolm Slaney is a principal scientist at Yahoo! Research Laboratory. He received his PhD from Purdue University for his work on computed imaging. He is a coauthor, with A. C. Kak, of the IEEE book “Principles of Computerized Tomographic Imaging.” This book was recently republished by SIAM in their “Classics in Applied Mathematics” series. He is coeditor, with Steven Greenberg, of the book “Computational Models of Auditory Function.” Before Yahoo!, he has worked at Bell Laboratory, Schlumberger Palo Alto Research, Apple Computer, Interval Research, and IBM’s Almaden Research Center. He is also a (consulting) Professor at Stanford’s CCRMA where he organizes and teaches the Hearing Seminar. His research interests include auditory modeling and perception, multimedia analysis and synthesis, compressed-domain processing, music similarity and audio search, and machine learning. For the last several years he has lead the auditory group at the Telluride Neuromorphic Workshop.

Tutorial 1

MIR at the Scale of the Web

Malcolm Slaney (Yahoo! Research, USA)
Michael Casey (Dartmouth College and University of London, UK)

Abstract

In the last couple of years we have received access to music databases with millions of songs. This massive change in the amount of data available to researchers is changing the face of MIR. In many domains, speech recognition is most notable, people have observed that the best way to improve their algorithm's performance is to add more data. Starting with hidden-Markov models (HMMs) and support-vector machines, people have applied ever greater amounts of data to their problems and been rewarded with new levels of performance. What are the algorithms and ideas that are necessary to work with such large databases? How do we define the scope of a problem, and how do we apply modern clusters of processors to these problems? What does it take to collect, manage, and deliver solutions with millions of songs and terabytes of data?

In this tutorial we will talk about a range of algorithms and tools that make it easy/easier to scale our work to Internet-sized collections of music. The field is just developing so this tutorial will talk about a range of techniques that are in use today. Millions of songs fit into a small number of terabytes, which is just a few hundred dollars of disk space. This tutorial will give attendees the tools they need to make use of this data. This tutorial will give attendees an overview and pointers to the tools that will allow them to scale their work to modern datasets. The tutorial will discuss the theoretical and practical problem with large data, applications where large amounts of data are important to consider, types of algorithms that are practical with such large datasets, and examples of implementation techniques that make these algorithms practical. The tutorial will be illustrated with many real-world examples and results.

Biographies

Malcolm Slaney is a principal scientist at Yahoo! Research. He received his PhD from Purdue University for his work on computed imaging. There he has been working on music- and image-retrieval algorithms in databases with billions of items. He has given successful tutorials at ICASSP 1996 and 2009 on "Applications of Psychoacoustics to Signal Processing" and on "Multimedia Information Retrieval" at SIGIR and ICASSP.

Michael Casey is Professor of Music and director of the graduate program in Digital Music at Dartmouth College, USA, and Professor of Computer Science at Goldsmiths, University of London, UK. He received his PhD from the MIT Media Laboratory in 1998 in the fields of statistical audio. His recent activities include forming the OMRAS2 (Online Music Recognition and Searching) group at Goldsmiths.

Tutorial 2

Mining the Social Web for Music-Related Data: A Hands-on Tutorial

Claudio Baccigalupo (Spanish Council for Scientific Research, Spain)
Benjamin Fields (University of London, UK)

Abstract

The social web is a useful resource for those conducting research in music informatics. Yet there exists no "standard" way to integrate web-based data with other more common signal-based music informatics methods. In this tutorial we go through the entire process of retrieving and leveraging data from the social web for MIR tasks. This is done through the use of hands-on examples intended to introduce the larger ISMIR community to web-mining techniques.

The intended audience is formed of people who are familiar with other MIR techniques (principally content-based) and who can benefit from knowledge available on the web to improve their algorithms and evaluation processes. The tutorial presents a series of short snippets of code to rapidly retrieve musical information from the web in the form of genre-labeled audio excerpts, tags, lyrics, social experiences, acoustic analyses or similarity measures for millions of songs.

Tutorial Website: http://ismir2009.benfields.net

Biographies

Claudio Baccigalupo is a PhD candidate at the Artificial Intelligence Research Institute (IIIA-CSIC), with the thesis discussion expected in November 2009. He holds a 5-year degree in Computer Technology with top marks and distinction. His research focuses on recommender systems in a musical context: he investigated how to extract musical knowledge from the analysis of playlists and how to customize radio channels for groups of listeners.

Benjamin Fields is a PhD candidate with the Intelligent Sound and Music Systems (ISMS) research group at the Department of Computing, Goldsmiths, University of London, with his dissertation submission anticipated in late spring 2010. His current research centers on applications to understand and exploit the semantic gap between the social relationships of artists and the acoustic similarity of works these artists produce.

Tutorial 3

Using Visualizations for Music Discovery

Justin Donaldson (Indiana University, USA)
Paul Lamere (The Echo Nest, USA)

Abstract

As the world of online music grows, tools for helping people find new and interesting music in these extremely large collections become increasingly important. In this tutorial we look at one such tool that can be used to help people explore large music collections: information visualization. We survey the state-of-the-art in visualization for music discovery in commercial and research systems. Using numerous examples, we explore different algorithms and techniques that can be used to visualize large and complex music spaces, focusing on the advantages and the disadvantages of the various techniques. We investigate user factors that affect the usefulness of visualization and we suggest possible areas of exploration for future research.

Biographies

Justin Donaldson is a PhD candidate at Indiana University School of Informatics, as well as a regular research intern at Strands, Inc. Justin is interested with the analyses and visualizations of social sources of data, such as those that are generated from playlists, blogs, and bookmarks.

Paul Lamere is the Director of Developer Community at The Echo Nest, a research-focused music intelligence startup that provides music information services to developers and partners through a data mining and machine listening platform. Paul is especially interested in hybrid music recommenders and using visualizations to aid music discovery.

Tutorial 4

Share and Share Alike, You Can Say Anything about Music in the Web of Data

Kurt Jacobson (University of London, UK)
Yves Raimond (BBC, UK)
György Fazekas (University of London, UK)
Michael Smethurst (BBC, UK)

Abstract

Linked Data provides a powerful framework for the expression and re-use of structured data. Recent efforts have brought this powerful framework to bear on the field of music informatics. This tutorial will provide an introduction to Linked Data concepts and how and why they should be used in the context of music-related studies. Using practical examples we will explore what data sets are already available and how they can be used to answer questions about music. We will also explore how signal processing tools and results can be described as structured data. Finally, we will demonstrate tools and best practice for researchers who wish to publish their own data sets on the Semantic Web in a Linked Data fashion.
Tutorial Website: http://ismir2009.dbtune.org

Biographies

Kurt Jacobson is a PhD candidate at the Centre for Digital Music. As assistant administrator of DBTune.org he has worked to create Semantic Web services for music including a service publishing structured data about music artists on MySpace and musicological data about classical music composers.

Yves Raimond is a Software Engineer at BBC Audio & Music interactive, after completing a PhD at the Centre for Digital Music, Queen Mary, University of London. He is one of the editors of the Music Ontology specification, and the creator and head administrator of the DBTune.org service. He is now working on http://www.bbc.co.uk/programmes, publishing a wide range of structured data about BBC programs.

György Fazekas is a PhD candidate at the Centre for Digital Music. His main research interest includes the development of semantic audio technologies and their application to creative music production. He is working on ontology-based information management for audio applications.

Michael Smethurst is an Information Architect at BBC Audio & Music. He is currently working on BBC Programs, BBC Music, and BBC Events, publishing and interlinking data in a number of overlapping domains. He writes on the BBC Radio Labs blog about Linked Data and web publishing.

INTEGRATING MUSICOLOGY'S HETEROGENEOUS DATA SOURCES FOR BETTER EXPLORATION

David Bretherton, Daniel Alexander Smith, mc schraefel, Richard Polfreman, Mark Everist, Jeanice Brooks, and Joe Lambert
University of Southampton, Southampton, UK, SO17 1BJ
D.Bretherton@soton.ac.uk; {ds, mc}@ecs.soton.ac.uk; {R.Polfreman, M.Everist, L.J.Brooks}@soton.ac.uk; jl2@ecs.soton.ac.uk

ABSTRACT

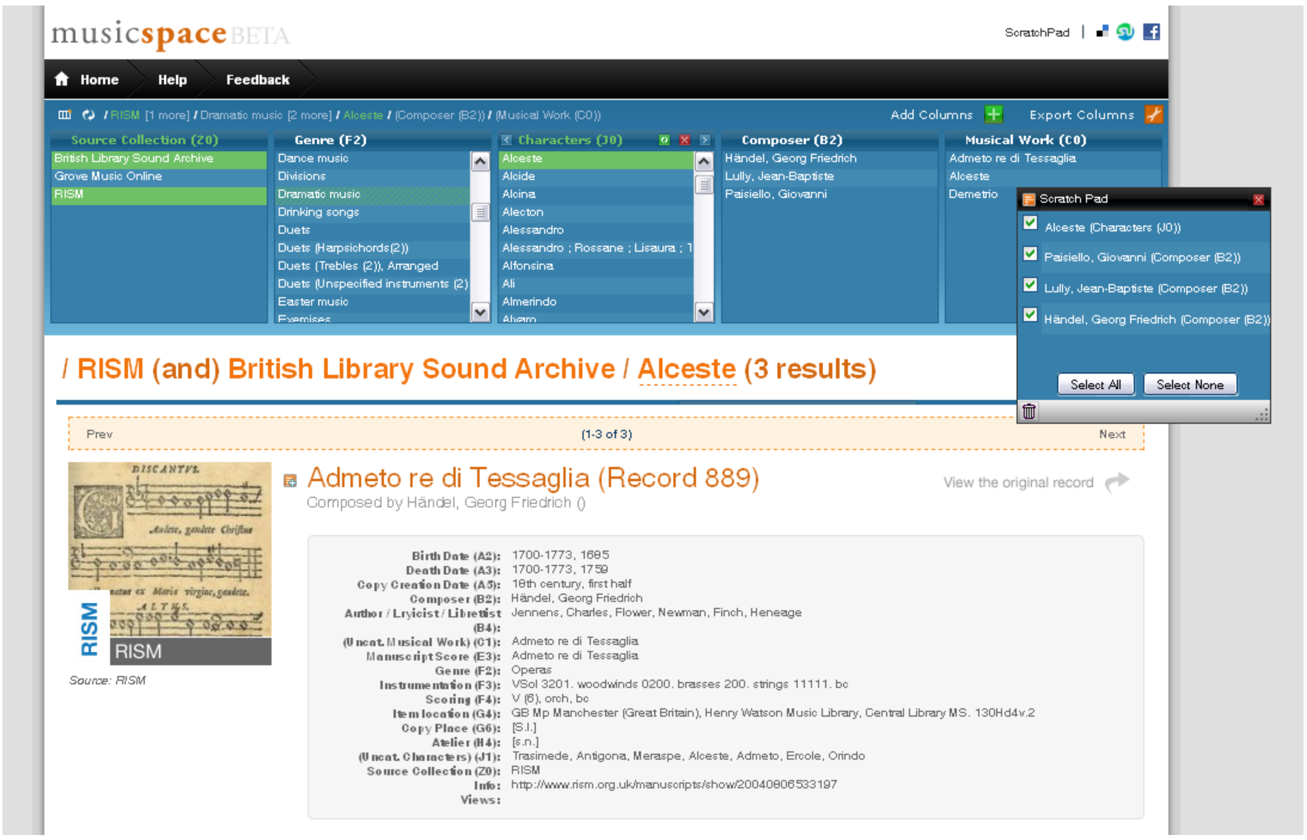
Musicologists have to consult an extraordinarily heterogeneous body of primary and secondary sources during all stages of their research. Many of these sources are now available online, but the historical dispersal of material across libraries and archives has now been replaced by segregation of data and metadata into a plethora of online repositories. This segregation hinders the intelligent manipulation of metadata, and means that extracting large tranches of basic factual information or running multi-part search queries is still enormously and needlessly time consuming. To counter this barrier to research, the "musicSpace" project is experimenting with integrating access to many of musicology's leading data sources via a modern faceted browsing interface that utilises Semantic Web and Web2.0 technologies such as RDF and AJAX. This will make previously intractable search queries tractable, enable musicologists to use their time more efficiently, and aid the discovery of potentially significant information that users did not think to look for. This paper outlines our work to date.

1. INTRODUCTION

A significant barrier to the research endeavours of musicologists is the sheer volume of potentially relevant information that has accumulated over centuries. Researchers once faced the daunting prospect of manually scouring through seemingly endless primary and secondary sources in order to answer the basic whats, wheres and whens of musicology, particularly when making lists of people or repertoire according to specific criteria. Many of the sources needed to address these queries are becoming available online. Yet the dramatic increase in the online availability of data, the variety of data subjects, the growing number of data providers, and, moreover, the inability of current mainstream search tools to manipulate the associated metadata in useful ways, means that extracting large tranches of basic factual information (e.g. manuscripts once owned by "a," opera roles performed by "b") or running multi-part search queries (e.g. composers from place "c" that were active during decade "d") is still enormously and needlessly time consuming.

Accordingly, the "musicSpace" project <http://www.mspace.fm/projects/musicspace> is exploiting Semantic Web [1] and Web2.0 technologies to develop an experimental innovative search interface that integrates access to some of musicology's largest and most significant online data and metadata repositories, including the British Library Music Collections catalogue, the British Library Sound Archive catalogue, Cecilia, Copac, Grove Music Online, Naxos Music Library, RILM, and RISM UK and Ireland. We anticipate that integrating heterogeneous metadata sources into one exploratory search user interface will allow our users to spend their research time more efficiently, make previously intractable search queries tractable, and ultimately open up new avenues for musicological study.

musicSpace is exploring and developing numerous methods for enhancing and generating additional metadata from our data partners' particularly heterogeneous data sets, and a primary focus is the development of web-based UIs and the longitudinal analysis of their effects on musicological scholarship and human-computer interaction. This distinguishes our work from that of previous notable projects concerned with music data source integration, such as Variations2 <http://variations2.indiana.edu> and EASAIER <http://www.easaier.org> [2, 3]. The "mSpace" framework and interaction layer of musicSpace has been designed and evaluated [4, 5] specifically to support multiple browsing and exploratory search tactics that go beyond common keyword search. Our user interface gives the provenance of all records, and is designed not only to help musicologists discover relevant resources, but also to enable them to go from musicSpace to those resources in their original context in a single click. Beyond these core features, there are numerous support services based on related usability research to assist with collecting,


musicspace BETA

ScratchPad |

Home | Help | Feedback

/ RISM [1 more] / Dramatic music [2 more] / Alceste / (Composer (B2)) / (Musical Work (C0))

Add Columns | Export Columns

Source Collection (Z0)	Genre (F2)	Characters (J0)	Composer (B2)	Musical Work (C0)
British Library Sound Archive	Dance music	Alceste	Händel, Georg Friedrich	Admeto re di Tessaglia
Grove Music Online	Divisions	Alcide	Lully, Jean-Baptiste	Alceste
RISM	Dramatic music	Alcina	Paisiello, Giovanni	Demetrio
	Drinking songs	Alecton		
	Duets	Alessandro		
	Duets (Harpsichords(2))	Alessandro ; Rossane ; Lisaura ; 1		
	Duets (Trebles (2)), Arranged	Alfonsina		
	Duets (Unspecified instruments (2))	Ali		
	Easter music	Almerindo		
	Exercises	Alvaro		

Scratch Pad
- ✓ Alceste (Characters (J0))
- ✓ Paisiello, Giovanni (Composer (B2))
- ✓ Lully, Jean-Baptiste (Composer (B2))
- ✓ Händel, Georg Friedrich (Composer (B2))

Select All | Select None

/ RISM (and) British Library Sound Archive / Alceste (3 results)

Prev (1-3 of 3) Next

Admeto re di Tessaglia (Record 889)
Composed by Händel, Georg Friedrich ()
View the original record

Birth Date (A2):	1700-1773, 1685
Death Date (A3):	1700-1773, 1759
Copy Creation Date (A5):	18th century, first half
Composer (B2):	Händel, Georg Friedrich
Author / Lryicist / Libretist (B4):	Jennens, Charles, Flower, Newman, Finch, Heneage
(Uncat. Musical Work) (C1):	Admeto re di Tessaglia
Manuscript Score (E3):	Admeto re di Tessaglia
Genre (F2):	Operas
Instrumentation (F3):	VSol 3201. woodwinds 0200. brasses 200. strings 11111. bc
Scoring (F4):	V (6), orch, bc
Item location (G4):	GB Mp Manchester (Great Britain), Henry Watson Music Library, Central Library MS. 130Hd4v.2
Copy Place (G6):	[S.l.]
Atelier (H4):	[s.n.]
(Uncat. Characters) (J1):	Trasimede, Antigona, Meraspe, Alceste, Admeto, Ercole, Orindo
Source Collection (Z0):	RISM
Info:	http://www.rism.org.uk/manuscripts/show/20040806533197
Views:	

RISM
Source: RISM

Figure 1. The musicSpace interface in use.

organising, exporting, and sharing information relevant to a particular query. It should also be noted that as musicSpace is a Web2.0 application, a web browser is all that is required to access the interface, a screenshot of which is given in Figure 1.

In this paper we give an overview of our work so far and outline the findings of our initial trial of the musicSpace browser interface. To begin, we review the motivation for our approach to supporting musicological knowledge building.

2. MOTIVATION: BARRIERS TO EFFICIENCY

2.1 Database Heterogeneity

The digitisation of musicology's central resources has revolutionised the research process, yet dispersal of material across numerous libraries and archives has now been replaced by segregation of data into a plethora of discrete and disparate online database resources. These are usually segregated according to media type (text, image, audio, video), date of publication, subject, language, and/or copyright holder. Yet typical musicological research cuts across these artificial divisions, meaning that musicologists are routinely forced to consult an extraordinarily heterogeneous body of online data repositories. In short, a significant amount of valuable research time is expended in establishing basic factual information, not because the data is unavailable, but because a lack of database integration requires extensive manual collation of discovered data. This problem of heterogeneity is exacerbated by the fact that search interfaces to data providers' content remain almost universally rooted in the now somewhat dated 'textbox-based' search paradigm. Not only does the current situation mean that users' research time is used inefficiently, but it also means that large, complex data queries are essentially intractable.

These barriers can be a major disadvantage at any stage of the research process. For example, a musicologist trying to mould an inchoate thought about Monteverdi's madrigals into a well-formed research question would need to execute the same keyword searches several times each because there are several relevant data sources. Similarly, because of the segregation of data into disparate, discreet databases and the limitations of currently deployed search interfaces, real-world multi-part queries such as "which scribes have created manuscripts of Monteverdi's works, and which other composers' works have they inscribed?" or "which singers have recorded the operas that Mozart composed during the 1780s, what other operatic roles have they taken, and where can I get hold of their recordings?" have to be broken down into their component parts, queried separately using multiple data sources, and finally collated, all of which can take hours or even days.

Recently, a number of academic publishers, including Oxford University Press (with Oxford Music Online

<http://www.oxfordmusiconline.com>) and Alexander Street Press (with Alexander Street Press Music Online <http://muco.alexanderstreet.com>), have recognised the benefits of integrating their musicological data sources [6, 7]. However, because their portals only provide access to their own data repositories, and because their interfaces rely on existing textbox-based search technology, their work only takes us partway towards overcoming the barriers to research highlighted above; there remains a pressing need for further integration of data sources and better interaction support for more diverse search paradigms.

2.2 "Intractable" Queries

The musicSpace team includes musicologists who specialise in four pilot research areas: Monteverdi recordings, Schubert's songs, nineteenth-century opera buffa, and twentieth-century electroacoustic music. At the start of the project we asked our musicologists for examples of queries that they considered intractable (or, more specifically, not readily tractable) using the current search interfaces of our data providers, such that they had largely given up on a particular line of enquiry, and which they hoped that musicSpace would be able to facilitate. The list of queries suggested included:

A. Which scribes have created manuscripts of a composer's works, and which other composers' works have they inscribed?
B. Which performers have recorded Monteverdi's madrigals, and what else did they record in the same years?
C. Which poets have had their poems set as songs by Schubert, which other song composers have also set them, and where can I get recordings of these settings?
D. Which singers have sung the role of Malatesta in *Don Pasquale*, and what else have they sung?
E. Which comic operas were composed in the nineteenth century and premiered in the twentieth?
F. Which electroacoustic works were published within five years of their premier?

It will be noted that all the above queries have multiple parts, and, therefore, if one were to use current search interfaces, one would have to break them down into their component queries and manually collate the results. There are several further obstacles to tractability. Queries B, C, D and F call (in particular) for several data sources to be consulted (for Queries B and D, for example, one would want to consult both the Naxos Music Library and the British Library Sound Archive catalogue), and so data source integration would clearly be beneficial in these cases. In addition, increased metadata granularity is a necessary prerequisite for the tractability of Queries A, C, D and F (for example, in Query A one would rely on metadata in RISM, yet although it is possible to use RISM's interface to search by "Person," it is not possible to further restrict this to "Composer" or "Scribe"). Finally, in addressing Queries C, E and F one would necessarily wish to consult the works lists in Grove Music Online. However, because these works lists are not marked up semantically, a system to generate relevant metadata from the raw data is needed (this particular issue is currently being addressed by musicSpace, and will be reported on at a later date).

3. EXPERIMENTAL SOLUTIONS: APPARENT INTEGRATION

There is at least one seemingly obvious solution to the above query dilemmas: enable integrated real-time querying over all the available metadata, and enable people to use that metadata to guide their queries. The associated issues for this solution also imply that all data that could be construed as useful, even if buried in the database records, is extracted in some way, and that, similarly, there is an interaction approach that will enable this metadata to be explored effectively to formulate the kinds of rich compound queries described above.

To this end, we have taken a dual approach to addressing this exploration problem: designing back-end services to integrate (and, where necessary, surface) available (meta)data for exploratory search; and providing a front-end interface to support rich exploratory search interaction. We discuss these components below.

3.1 Multi-Source Integration

Despite advances in the development of protocols for shareable metadata in the form of the Open Archives Initiative <http://www.openarchives.org> [8], federated search [9], and, more recently, the application of Semantic Web technologies to the domain of music [10, 11], only a very small number of musicSpace's data partners offer such systems for the harvesting of metadata. This is typically either because funds are presently unavailable to meet the costs of implementing such systems, or, in the case of some data providers, because metadata is considered to be as much of an intellectual property asset as data content itself. Hence our data partners' data sets are currently provided to us manually.

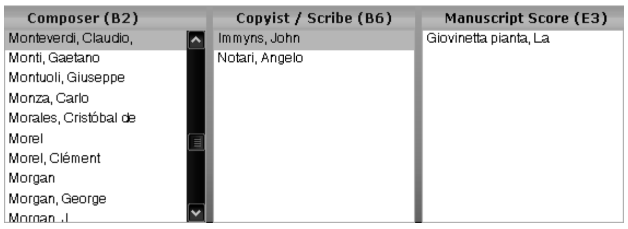
We have thus taken a purpose-driven approach to unifying the metadata from our data partners, which is supplied adhering to a number of different schemas and serialisations (MARCXML, MODS XML, custom MARC, and source-specific XML). In order to unify these sources for the purposes of cross-source exploration, we have created static mappings from the schemas used by each data provider to a two-level hierarchy based on metadata type. The upper level of the hierarchy includes, for example, "Person" and "Score," while the sub-level respectively adds granularity to "Composer" and "Manuscript Score" (among other possibilities). In some cases we were able to directly map a record field to our type hierarchy, while in other cases some light syntactic and/or semantic analysis was performed on the source data. For example, some sources denote a person with their name, followed by their role in that record, e.g. “J. S. Bach (composer).” In this case we extract the name and role as two individual related facts to allow us to associate “J. S. Bach” as “Composer” in the record, rather than simply “J. S. Bach (composer)” as “Person.” This pre-processing of the metadata adds granularity to the source data and allows richer filtering and exploration through the browsing interface. We developed a tool to map the imported data to an RDF representation of our type hierarchy. By using RDF for the integrated set of data, we can make use of the many benefits of Semantic Web technologies, one of which is the facility to create multiple files of RDF at different times and using different tools, assert them into a single graph of a knowledge base, and query all of the asserted files as a whole.

One of the challenges in aligning heterogeneous data sources is that of entity co-reference. It is rare that data providers share identifiers for entities, and as such, we have to perform co-reference mapping ourselves. For the musicological data we are aligning in musicSpace, a straightforward string matching system is appropriate to match entities across sources; we use Alignment API [12], which uses Wordnet. To ensure greater confidence in these matches, we have developed a semi-automated system that enables musicologists to check the mappings and inform the system of any changes that need correcting. Whenever a mapping is automatically performed, our system adds the mapping to a gazetteer, documenting the two strings that were matched along with a small amount of contextual metadata from both records to aid understanding. The gazetteer is then ordered by confidence, so that a musicologist – with reference to the Library of Congress Authorities website <http://authorities.loc.gov> – can check over the low-confidence mappings carefully, update the gazetteer (either to remove the mapping, alter it, or provide a replacement), and inform the co-reference software of the changes. By using this approach we can be sure that the data sources are aligned properly, and that any updates from our data partners will re-use the manually corrected gazetteers.

Because of the legacy issues that many of our data partners have to contend with, there are inevitably shortcomings and inconsistencies in their database structures, schemas, and records. But by using gazetteers in the string matching process, adding contextual metadata, and increasing granularity as records are imported, we are able to negate any such data quality issues. In addition, our approach means that we do not have to maintain copies of our data partners’ databases for ourselves; rather, we provide a user interface service that provides a single point of entry to our data partners’ repositories.

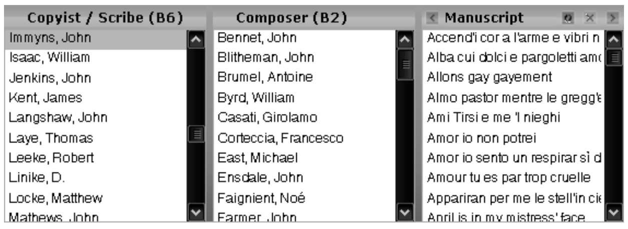
3.2 User Interface

Data sources integrated into musicSpace are explored via a customised version of the “mSpace” faceted browser [4, 5], which provides a scalable web-based faceted browsing interface for large-scale data sets and utilises the AJAX client-server query mechanism to improve response times. Faceted browsing is an alternative complementary search paradigm to keyword searching, the latter currently being the most commonly deployed form of large-scale data exploration. The faceted interface customisation used by musicSpace presents columns that list attributes from a number of facets of the data, such as “Date,” “Musical Work,” “Composer,” and “Genre,” allowing the user to make selections in these facets in order to filter down results. The interface is reactive, in that the lists of facets are updated every time a selection is made, so that subsequent choices are limited to those that would yield results.



Composer (B2)	Copyist / Scribe (B6)	Manuscript Score (E3)
Monteverdi, Claudio,	Immyns, John	Giovinetta pianta, La
Monti, Gaetano	Notari, Angelo	
Montuoli, Giuseppe		
Monza, Carlo		
Morales, Cristóbal de		
Morel		
Morel, Clément		
Morgan		
Morgan, George		
Morgan, J		

Figure 2. Scribes associated with the composer “Monteverdi, Claudio.”



Copyist / Scribe (B6)	Composer (B2)	Manuscript
Immyns, John	Bennet, John	Accend'i cor a l'arme e vibri n
Isaac, William	Blitheman, John	Alba cui dolci e pargoletti am
Jenkins, John	Brumel, Antoine	Allons gay gayement
Kent, James	Byrd, William	Almo pastor mentre le gregg'e
Langshaw, John	Casati, Girolamo	Ami Tirsi e me 'l nieghi
Laye, Thomas	Corteccia, Francesco	Amor io non potrei
Leeke, Robert	East, Michael	Amor io sento un respirar sì d
Linike, D.	Ensdale, John	Amour tu es par trop cruelle
Locke, Matthew	Faignient, Noé	Appariran per me le stell'in cie
Mathews, John	Farmer, John	April is in my mistress' face

Figure 3. Composers associated with the scribe “Immyns, John.”

The faceted and reactive nature of the interface enables complex queries to be addressed. Let us consider the query “which scribes have created manuscripts of Monteverdi’s works, and which other composers’ works have they inscribed?” In Figure 2, the musicSpace interface is showing three facets: “Composer,” “Copyist/Scribe,” and “Manuscript Score.” The selection “Monteverdi, Claudio” in “Composer” has been made, as well as “Immyns, John” in “Copyist/Scribe,” and the interface has filtered the results in “Manuscript Score” to a single record that matches these selections: “Giovinetta pianta, La.” Following from this interaction, in Figure 3 the user has dragged the column “Copyist/Scribe” leftwards, so that the selection “Immyns, John” now filters on the “Composer” column, as well as the “Manuscript Score” column, so that the user can see works by other composers that had John Immyns as the scribe.

3.3 Saving, Exporting, and Sharing Findings

Each interaction with the musicSpace interface generates a specific URL that, when re-entered into a web browser at a later stage, will return users to exactly that same point in the data exploration process. Thus users can pause and resume their research at any time by using the bookmarking feature common to all web browsers, and, moreover, they can save, share, and disseminate their findings with colleagues, students, and the wider internet by using Web2.0 services such as del.icio.us, Facebook and StumbleUpon, all of which can be accessed by clicking the appropriate icon in the musicSpace interface. Exporting of findings via email is also supported. In addition, musicSpace has the facility to allow users to access and export metadata as RDF (using the Music Ontology <http://musicontology.com> [11] as a data model), but licensing restrictions with our data partners currently prevent us from doing so for all data sets.

4. EVALUATION

Since the mSpace UI has been evaluated for exploratory search usability in a variety of contexts, our main focus in testing the musicSpace application is its impact on research: how well is it supporting the kinds of queries musicologists want it to enable? And, likewise, what new kinds of research questions, as yet unanticipated, may it enable? Towards answering these questions, we have recently completed an early pilot study. We describe our findings below. While these are early stage tests, our intention in outlining our findings here is to have knowledge of our approach and preliminary results available within the Music IR community in order to enhance engagement with the project.

4.1 First Phase

A version of the musicSpace interface was released internally to a team of six musicologists for an initial period of testing and evaluation on 29 April 2009, and their feedback was very encouraging. Although this initial release did not integrate our full spread of data sources, testers nevertheless reported significant improvements with search speed and ease:

- "All the information showed up very quickly, and it was easy to find material. It was really good to have different kinds of material in the same place."
- "[musicSpace offers] a speedier way to research crossed search pathways."
- "Excellent interface – very simple to understand."

Testers were also impressed with the way that musicSpace's faceted interface allowed for browsing around a subject and for instantaneous paradigmatic shifts in search focus:

- "I would recommend musicSpace for its ability to manipulate queries in order to get results that you wouldn't otherwise be able to get [without starting over]."
- "I liked the ability to explore around a topic once I'd identified something of interest."
- "The ability to switch columns around and add new columns was most useful."

Aside from these early hoped-for indications that musicSpace will provide a quicker and more flexible way to explore a variety of musicological data sources, testers also reported that increased search data granularity (as compared to that of our data partners' search interfaces) was a substantial benefit. For example, a number of testers were pleased by musicSpace's facility to browse by opera character:

- "[Without using musicSpace] it would not be at all easy to do a character search. You would have to use printed reference books like *Pipers Enzyklopädie des Musiktheaters* [13], but even this does not have an index of characters, so you'd have to look at the entry for each opera and draw up character lists by hand. You would also have to know what you were looking for before you started out!"
- "I used musicSpace to explore how many operas have a character named Alceste. This information simply isn't get-at-able using other search interfaces – you'd have to sort through the information on your own."

There was similar enthusiasm for musicSpace's ability to browse by scribe and the former owner of manuscripts.

4.2 Future Phases

Over the coming months there will be incremental releases of musicSpace, each expanding the data set, refining our data mappings, and polishing the UI. This process will culminate in a broader public release towards the end of 2009, which will enable us to assess its real-world efficacy as a research tool.

5. CONCLUSION

Early results from our testing of musicSpace's ability to enable rapid and effective exploratory search across heterogeneous musicological sources are promising. Our testers clearly appreciated the speed gains of integrating data sources; in fact the only recurring negative comments from testers during our initial period of evaluation concerned their desire to see still more data repositories integrated into musicSpace. In addition to data source integration, both increased data granularity and the flexibility of faceted browsing were found to be very beneficial. These three features enabled testers to explore data in a way that had not previously been possible, and a number of intractable queries were indeed made tractable.

In his keynote address to this conference in 2005, Nicholas Cook predicted that "working with larger data sets will open up new areas of musicology" [14]. But if

Cook's prediction is to be realised, then increasing the size and number of data sets that musicologists work with both demands and allows for better systems to integrate those data sets, and also for far more sophisticated systems for manipulating data. To this end, our research demonstrates a potentially powerful approach for helping musicologists to deal intelligently and productively with large and heterogeneous data sets. We believe that musicSpace will allow musicologists to find the information they need more easily, and to discover information that they did not think to look for. In so doing, it may also encourage additional speculative – but potentially fruitful – searches, thus enabling the discovery of new knowledge.

6. ACKNOWLEDGEMENTS

The musicSpace project is funded by the Arts and Humanities Research Council <http://www.ahrc.ac.uk>, the Engineering and Physical Sciences Research Council <http://www.epsrc.ac.uk>, and the Joint Information Systems Committee <http://www.jisc.ac.uk>. Our data partners include the British Library <http://www.bl.uk>, the British Library Sound Archive <http://www.bl.uk/nsa>, Cecilia <http://www.cecilia-uk.org>, Copac <http://copac.ac.uk>, Grove Music Online (OUP) <http://www.oxfordmusiconline.com>, Naxos Music Library <http://www.naxosmusiclibrary.com>, RILM <http://www.rilm.org>, and RISM UK and Ireland <http://www.rism.org.uk>.

7. REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila: "The Semantic Web," *Scientific American*, Vol. 284, No. 5, pp. 34–43, 2001.

[2] J. W. Dunn, D. Byrd, M. Notess, J. Riley, and R. Scherle: "Variations2: Retrieving and Using Music in an Academic Setting," *Communications of the ACM*, Vol. 49, No. 8, pp. 55–58, 2006.

[3] C. Landone, J. Harrop, and J. Reiss: "Enabling Access to Sound Archives through Integration, Enrichment and Retrieval: the EASAIER Project," *Proceedings of the Eighth International Conference on Music Information Retrieval*, pp. 159–160, 2007.

[4] mc schraefel, D. A. Smith, A. Owens, A. Russell, C. Harris, and M. L. Wilson: "The Evolving mSpace Platform: Leveraging the Semantic Web on the Trail of the Memex," presented at *Hypertext*, Salzburg, 6–9 September 2005.

[5] mc schraefel, M. L. Wilson, A. Russell, and D. A. Smith: "mSpace: Improving Information Access to Multimedia Domains with Multimodal Exploratory Search," *Communications of the ACM*, Vol. 49, No. 4, pp. 47–49, 2006.

[6] L. Macy: "Letter from the Editor," *Oxford Music Online*, March 2008 <http://www.oxfordmusiconline.com/public/page/letter_08>.

[7] A. Hall: "Alexander Street Press: New Developments," presented at the *Academic Music Librarians' Seminar*, Birmingham Conservatoire, 21 May 2009.

[8] C. Lagoze, and H. Van de Sompel: "The Open Archives Initiative: Building a Low-Barrier Interoperability Framework," *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, 2001, pp. 54–62.

[9] C. N. Cox, ed.: *Federated Search: Solution or Setback for Online Library Services*, Haworth Information Press, Binghamton NY, 2007.

[10] C. Lai, I. Fujinaga, D. Descheneau, M. Frishkopf, J. Riley, J. Hafner, and B. McMillan: "Metadata Infrastructure for Sound Recordings," *Proceedings of the Eighth International Conference on Music Information Retrieval*, pp. 157–158, 2007.

[11] Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson: "The Music Ontology," *Proceedings of the 8th International Conference on Music Information Retrieval*, 2007, pp. 417–422.

[12] J. Euzenat: "An API for Ontology Alignment," *Proceedings of 3rd International Semantic Web Conference*, pp 698–712, 2004.

[13] C. Dahlhaus, and S. Döhring, eds: *Pipers Enzyklopädie des Musiktheaters: Oper, Operette, Musical, Ballet*, 7 Vols, Piper, Munich, 1986–1997.

[14] N. Cook: "Towards the Complete Musicologist," *Proceedings of the Sixth International Conference on Music Information Retrieval*, 2005.

AN ECOSYSTEM FOR TRANSPARENT MUSIC SIMILARITY IN AN OPEN WORLD

Kurt Jacobson
Centre for Digital Music
Queen Mary
University of London, UK
kurtjx@gmail.com

Yves Raimond
BBC
London, UK
yves.raimond@bbc.co.uk

Mark Sandler
Centre for Digital Music
Queen Mary
University of London, UK

ABSTRACT

There exist many methods for deriving music similarity associations and additional variations are likely to be seen in the future. In this work we introduce the Similarity Ontology for describing associations between items. Using a combination of RDF/OWL and N3, our ontology allows for transparency and provenance tracking in a distributed and open system. We describe a similarity ecosystem where agents assert and aggregate similarity statements on the Web of Data allowing a client application to make queries for recommendation, playlisting, or other tasks. In this ecosystem any number of similarity derivation methods can exist side-by-side, specifying similarity relationships as well as the processes used to derive these statements. The data consumer can then select which similarity statements to trust based on knowledge of the similarity derivation processes or a list of trusted assertion agents.

1. INTRODUCTION

The process of music recommendation in a general sense involves drawing associations between music-related items - i.e. artist a is *similar* to artist b so recommend b if the user expresses interest in artist a. We believe that similarity is the underlying "currency" for recommendation. This realization drives our interest in developing a formal model for similarity.

Similarity is a difficult concept. The exact nature of similarity has been discussed extensively in cognition [26, 28], philosophy [22, 14], and computer science [27, 17]. In the field of music information retrieval we have been less concerned with the nature of similarity and more concerned with finding ways of calculating it [18, 20, 5]. This pragmatic approach has led to a wealth of methods for deriving music similarity statements from audio analysis and contextual metadata.

But if we want to develop a generalized model for music similarity, it becomes more complicated. As Wittgenstein puts it in his seminal work *Philosophical Investigations* "Some things share a complicated network of similarities overlapping and criss-crossing: sometimes overall similarities, sometimes similarities of detail." Music would definitely be such a thing. Discussing a pair of songs, we can have a dizzying array of similarity options: the audio could have timbral similarity, rhythmic similarity, or melodic similarity; the contexts of the songs could make them similar in terms of lyrical content, cultural meaning, or shared listenership; or an authoritative source such as a music critic or website could judge the songs to be similar without providing any additional justification. Further complicating matters, similarity is *subjective* - what one individual or agent considers similar another may not.

Because similarity can be so nebulous and contentious we purpose a model for expressing similarity that foregoes hierarchical classifications and instead focuses on provenance and transparency. Instead of focusing on how a particular similarity statement is related to another similarity statement, we focus on *who* made the similarity statement and *why*.

Our approach is based on the Resource Description Framework (RDF) [4, 9] and the Web Ontology Language [3]. While these technologies provide an impressive amount of expressiveness and form the foundation of the Semantic Web, we augment their expressiveness with N3 [7]. The facilities for quoting formulae provided by N3 allows us to use the N3-Tr framework [23] for defining similarity derivation workflows.

In Section 2 we develop our model in the form of a Web ontology, briefly discussing some of the supporting technologies and previous work. In Section 3 we describe our vision of a similarity ecosystem where a number of agents aggregate and publish similarity statements in the Web of Data while music applications query these statements for recommendation or playlist generation. In Section 4 we provide a cursory evaluation of our ontology. In Section 5 we review some related work and finally provide some conclusions and directions for future work in Section 6.

2. AN ONTOLOGY FOR SIMILARITY

Because of its decentralized nature, wide deployment base, and robust technological underpinnings we use the RDF/OWL framework [4, 3, 9] for defining our Similarity Ontology.

This allows us to use the concepts, practices, and resources of *Linked Data* [8]. In the Linked Data paradigm, every resource and concept is given a Unique Resource Identifier (URI). These URIs can be dereferenced using HTTP to provide additional information and links to other relevant URIs.

2.1 Previous Ontologies

RDF [4] allows us to express information in the form of *triples*: subject, predicate, object statements. Generally the subject will be an instance of a *class concept* while the predicate will be an instance of a *property*. The object will also be an instance of a class concept but not necessarily the same class as the subject. Classes and properties are defined in an ontology document using the Web Ontology Language (OWL) [3] or the RDF Schema (RDFS) [9] or a combination of both. These technologies together enable what is commonly referred to as the Semantic Web or Web of Data.

These concepts have been successfully applied to the domain of music with the Music Ontology [24, 23]. The Music Ontology allows us to express a wide variety of music-related information as structured data in a decentralized fashion. It has been adopted by the Linked Data community and is used extensively throughout the Web of Data as a means of describing tracks, artists, performances, and related data.

The Music Ontology provides a basic facility for dealing with music similarity. The `mo:similar_to` property allows one to assert a similarity relationship between two items. However, this property relation does not provide any further information - How was the similarity derived? Who derived it? How similar are the two items?

2.2 Association as a Concept

Instead of treating similarity or, to use a broader term, association as a *property*, we treat association as a *class concept*. This allows us to reify the association in order to provide additional information about it. We introduce the class `sim:Association` and a sub-class `sim:Similarity` as the key concepts in our ontology. A simple similarity example is presented in the following listing [1]:

```
:track01 a mo:Track .
:track02 a mo:Track .
:me a foaf:Person .
:mySimilarity a sim:Similarity ;
    sim:element :track01 ;
    sim:element :track02 ;
    sim:weight "0.90" ;
    foaf:maker :me .
```

We introduce the namespace `sim` to refer to our Similarity Ontology. First we define two tracks using the corresponding Music Ontology concept `mo:Track`. The identifiers of these tracks can give entry points to additional information in other data sets (i.e. linking to dbpedia.org [2] URIs or Musicbrainz [3] identifiers). We define `:mySimilarity` to actually make the similarity statement. The `sim:element` property is used to refer to the tracks involved in this similarity and the `foaf:maker` property refers to the agent which asserted this similarity. Also note we can assign a numerical weight value to the similarity using the `sim:weight` property.

Now we have a method for asserting a similarity statement and reifying that statement to some extent. However, in the above example we only know *who* is making the similarity statement, we do not know *how* or *why*.

2.3 Provenance and Transparency

We introduce the `sim:AssocationMethod` concept to identify the process used to derive a similarity statement. This enables some interesting functionality when consuming the associations data - a consumer application can elect to include only similarity statements that are tied to a particular `sim:AssocationMethod`. This is discussed further in section 3.1. For now let us consider the following N3 listing:

```
:timbreSimilarityStatement
  a sim:Similarity ;
  sim:element :track01 ;
  sim:element :track02 ;
  sim:weight "0.9" ;
  sim:method :timbreBasedSimilarity .

:timbreBasedSimilarity
  a sim:AssociationMethod ;
  foaf:maker :me ;
  sim:description :algorithm .

:algorithm = {
{ { ?signal1 mo:published_as ?track01 .
    ?signal1 sig:mfcc ?mfcc1 .
    ?mfcc1 sig:gaussian ?model1 }
      ctr:cc
  { ?signal2 mo:published_as ?track02 .
    ?signal2 sig:mfcc ?mfcc2 .
    ?mfcc2 sig:gaussian ?model2 } .
  (?model1 ?model2) sig:emd ?div .
  ?div math:lessThan 0.2 } =>
  { _:timbreSimilarityStatement
      a sim:Similarity ;
      sim:element ?track01 ;
      sim:element ?track02 }
}
```

Here `:timbreBasedSimilarity` is the entity that describes our process for deriving similarity statements. Note that this entity is only described by three triples - its class type, a property for the creator and the description.

N3 extends the semantics and syntax of RDF in a useful and intuitive way. It allows for the existence of RDF graphs (a set of triple statements) as quoted formulæ. We can then make statements about the entire RDF graph providing metadata about that graph. In this way N3 is similar to Named Graphs [10], the main difference being that N3 considers RDF graphs as literals (their identity is their value), whereas Named Graphs consider graphs as entities named by a web identifier.

[1] We use N3 [6] in all our code listings. Each block corresponds to a set of statements (subject, predicate, object) about one subject. Web identifiers are either between angle brackets or in a prefix:name notation (with the namespaces defined at the end of the paper). Universally quantified variables start with `?`. Existentially quantified variables start with `_:`. Curly brackets denote a literal resource corresponding to a particular RDF graph. The keyword `a` correspond to the identifier `rdf:type`. The keyword => correspond to the identifier `log:implies`.

[2] `http://dbpedia.org`

[3] `http://musicbrainz.org/`

In the above example, when we follow the `sim:descripti` property we see an RDF graph `:algorithm` denoted by the `{` and `}` characters. This RDF graph provides a disclosure of the algorithm used in the similarity derivation process. In this case, MFCCs are extracted and Gaussian mixture models are created concurrently for the two signals, and an earth mover's distance is calculated between models. Depending on that distance, we output a similarity statement. If more details are needed about a particular computational step, e.g. if we want to gather more information about the MFCC extraction step, we can look-up the corresponding web identifier, in this case `sig:mfcc`.

The algorithm is specified using the N3-Tr framework which uses transaction logic and N3 to describe signal processing workflows. Additional details on N3-Tr are available in [23].

Here, the N3-Tr formulæ describe the workflow supporting the similarity statement. We could forego the use of the `sim:AssociationMethod` concept and use the `log:supports` built-in predicate[4] in the N3 framework. However, as we will discuss in section 3.1, binding similarity workflows to the `sim:AssociationMethod` concept allows us to make simple, useful queries (i.e."show me all similarity derivation methods available in the system").

Finally, note that we bind the `foaf:maker` property to the association method rather than directly to the association itself. As in the above example we can make our association method transparent, or we can provide a minimum amount of information when dealing with a "black box" similarity derivation processes. In either case it is a matter of best practice to create an association method, even if we do not desire full transparency because this allows data consumers to make simple queries.

As indicated in Figure 2.3, our framework also supports the grounding of similarity statements directly through the property `sim:grounding`. This property associates a similarity statement with the instantiated N3-Tr formulæ which enabled its derivation. In the above example, we would link our timbre similarity statement directly to a specific workflow with references to the calculated values at each step.

3. A SIMILARITY ECOSYSTEM

The data model provided by the Similarity Ontology allows for lots of flexibility in specifying similarity statements. This flexibility is balanced by the built-in mechanisms for provenance tracking. By following the `method` property in a similarity statement we know *who* made the statement and *why*. When consuming similarity data, we select statements by deciding which agents and algorithms to trust. While it is entirely possible to make a similarity statement within this framework completely anonymously, such statements are likely to be ignored by data consumers. Instead the statements from trusted agents or transparent algorithmic processes are likely to be selected by data consumers. In a music recommendation application, this allows for more transparent recommendations - providing the end user with the source or process used to make the recommendation. Intuition as well as recommender system research suggest users are more likely to trust transparent recommendation processes [11].

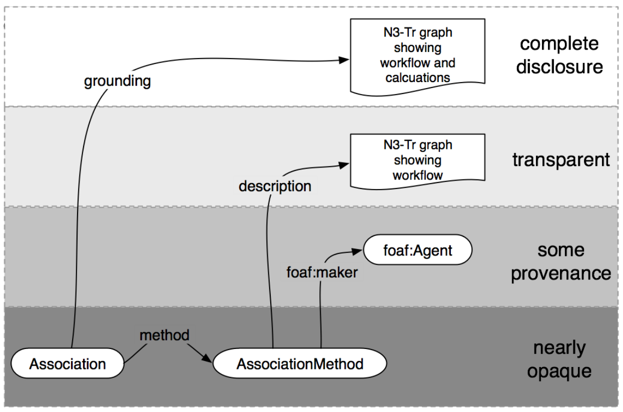


Figure 1. Using the Similarity Ontology. As additional properties are bound to our association and association method statements, we achieve greater transparency.

Beyond the specification of the Similarity Ontology, we envision a broader ecosystem where autonomous, semi-autonomous, and human agents operate in parallel, making similarity statements about music tracks and artists while providing provenance and justification for these statements. A simple diagram illustrating how this ecosystem might be structured is provided in Figure 2.

An enabled client music application publishes the end user's listening habits to the Web of Data. Similarity agents operate on the Web of Data and publish their own music similarity statements - perhaps consuming the listening habits of end users as well as other data. These statements refer to specific URIs for each track and artist. Similarly, the client music application links the content in the user's personal collection to URIs using methods such as those detailed in [25]. This avoids ambiguity - we can be sure that the similarity statements are referring to the specific resource in which we are interested. The similarity statements made by various agents are aggregated into one or more data stores for querying. The client music application, perhaps responding to a user request, can query the data store for similarity statements from trusted agents involving the target resource (i.e a track or artist). The query returns similarity information that can be used for content recommendations or playlist generation.

3.1 Similarity Queries

Queries in this similarity ecosystem would be made using the SPARQL query language [1]. The SPARQL specification is a W3C recommendation and the preferred method for querying RDF graphs. As mentioned before, the design of the Similarity Ontology allows for the construction of simple queries to retrieve similarity information. The following query retrieves artists similar to a target artist as

[4] see `http://www.w3.org/DesignIssues/N3Logic`

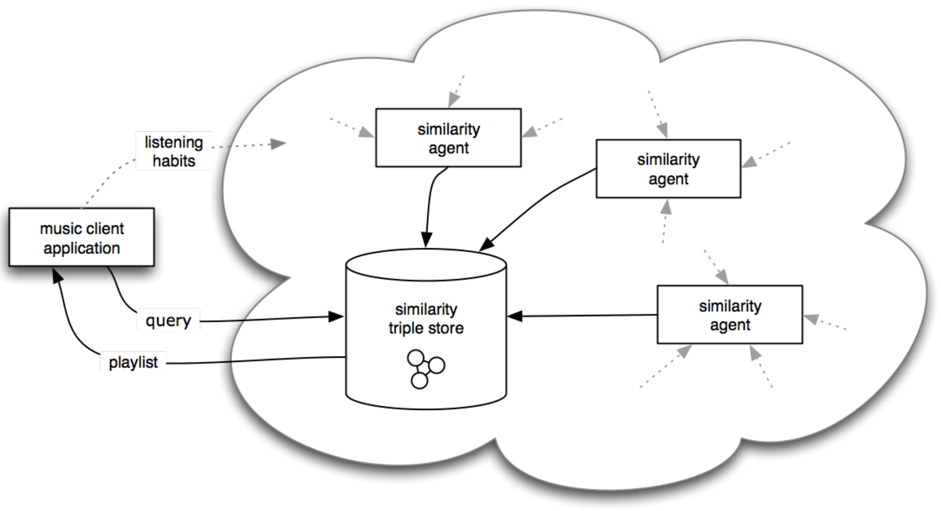


Figure 2. The music similarity ecosystem. Similarity agents operate on structured data to create similarity statements. Such statements are aggregated in a data store and queried by a client music application to provide recommendations, playlists, and other functionality.

stated by a specific trusted method:

```
SELECT ?artists WHERE {
  ?statement sim:method <http://trusted.method/uri> .
  ?statement sim:element <http://target.artist/uri> .
  ?statement sim:element ?artists . }
```

Notice we only have to include a triple pattern for our target resource, a triple pattern for our trusted agent, and a triple pattern to select the similar artists. Of course this is a very simple example and in real-world applications we include additional optional patterns and conjunctions for a more expressive query.

In an initialization step, an application could query available data sources to determine exactly what association methods and asserting agents are available. The application would use the following query:

```
SELECT DISTINCT ?method WHERE{
  ?method a sim:AssociationMethod . }
```

The application could then filter through the results and, perhaps with some input from the end-user, decide which similarity agents to trust.

3.2 Similarity and Recommendation

While we hold that similarity is the basis of recommendation, we also acknowledge that similarity and recommendation are not identical. By no means does the ecosystem proposed here solve the problems of recommender systems - rather it provides a new distributed cross-domain platform on which future recommender systems might be built.

While an item-to-item recommendation system fits quite naturally into this similarity ecosystem, we can also imagine a collaborative filtering-style user-item recommendation system. Each user in the system is treated as an `sim:AssocationMethod` instance. Each user's method makes a set of statements asserting that the tracks found in that user's personal collection are similar to each other. Then an additional `sim:AssocationMethod` instance is used to match users with each other based on the contents of their respective music libraries. Finally, for a given user, the recommendations for that user are an aggregation of the similarity statements derived from the association methods bound to the most similar users.

Also note that the similarity ecosystem fosters hybrid recommendation approaches. Because the similarity statements are made using common semantics and syntax, we can easily combine and compare these statements to derive recommendations or new similarity statements.

4. ONTOLOGY EVALUATION

While our Similarity Ontology is very flexible and potentially very expressive, there is one import limit to its expressiveness - there is no mechanism for expressing *dissimilarity*. This is an intentional design decision that follows from the open world assumption - we cannot know *all* instantiations of similarity, and what we consider dissimilar, another agent may consider similar.

As a cursory evaluation of our Similarity Ontology we present several real-world similarity scenarios and show how our ontology can accommodate these examples.

4.1 Directed Similarity

As often noted in psychology and cognition [28], similarity is not always symmetric. For example in the domain of music we may wish to express an influence relationship or we may simply have a similarity derivation algorithm that is non-symmetric. This leads to a *directed* similarity relationship. To accommodate such scenarios we introduce `sim:subject` and `sim:object` as sub-properties of the `sim:element` property. This allows us to specify a directed similarity statement where the subject is similar to the object, accepting that the reverse is not necessarily true.

4.2 Contextual Similarity

Because music is a complex construct deeply ingrained in culture and society, we often want to make music similarity statements that relate to the context of musical works rather than the content of the musical works themselves. Let us consider an example from popular rap music. In the mid to late 1980s a series of songs were released disputing the place of origin of the musical genre hip hop launching a multi-faceted feud that became colloquially referred to as *The Bridge Wars*[5]. By simply creating an association method that asserts similarities between artists and tracks related to this feud we can accommodate this scenario.

4.3 Personal Associations

The emotional affect of music can be highly personal. A set of associations between music artists or tracks might be unique for one particular individual. Consider the following statement, “When a first year student at college, I dated a girl who listened to Bob Marley and David Bowie” - while this association between David Bowie and Bob Marley might hold weight for the narrator, it is likely that few other individuals would share this association. However, the narrator, for any number of reasons, may wish to express this association anyway. This is entirely possible in our ontological framework. The narrator can simply create an `sim:AssociationMethod` that asserts similarity statements based on the musical taste of his ex-girlfriend.

5. RELATED WORK

Semantic Web technologies have been applied to music recommendation in previous works [12, 21] although, to the best knowledge of the authors, the present work is the first effort to develop a comprehensive framework for expressing music similarity on the Web of Data.

The Sim-Dl framework provides a basis for deriving similarities from semantic information within a description logic paradigm, although no formal syntax for expressing similarity results is provided [15]. Similarly, the iSPARQL framework extends SPARQL to include customized similarity functions [16] but fails to provide a formal method of expressing the resulting similarities.

Although the N3-Tr framework provides a clean and extensible syntax for describing similarity derivation workflows, alternative frameworks can be used as well. The Proof Markup Language provides a flexible means for justifying the results of a Semantic Web query [13].

The vast body of work on music similarity and music recommendation [18, 5, 20, 11] provides a set of templates for designing music similarity agents that might operate in our purposed ecosystem.

[5] `http://en.wikipedia.org/wiki/The_Bridge_Wars`

Knowledge management systems for music-related data such as Pachet’s work [19] and more specifically the ontology engineering of Raimond [24, 23] and Abdallah et. al [2] provide the basis for the similarity ecosystem. Without the Music Ontology framework for describing music metadata and the technology and infrastructure provided by the Linked Data community - including Muscibrainz URIs for songs and artists and data publishing guidelines - the Similarity Ontology would be unusable.

6. CONCLUSIONS AND FUTURE WORK

We have presented an ontological framework for describing similarity statements on the Web of Data. This ontology is extremely flexible and capable of expressing a similarity between any set of resources. This expressiveness is balanced by transparency and provenance, allowing the data consumer to decide what similarity statements to trust. We have shown hows this framework could exist as the foundation for a broader music similarity ecosystem where autonomous, semi-autonomous, and human agents publish a wealth of similarity statements which are combined, consumed, and re-used based on provenance, trust, and application appropriateness.

We have suggested how similarity algorithms can be made transparent. We have adopted the N3-Tr syntax for describing similarity derivation workflows. In future work we plan to extend this syntax and the supporting ontologies to better enable the publication of similarity derivation workflows. Furthermore we hope to develop a series of recommendations for best practice when publishing such workflows to maximize their usefulness and query-ability.

We also plan to adopt a method of digitally signing similarity statements in our ecosystem using terms available in the WOT RDF vocabulary[6]. This would allow agents to sign similarity statements using Public Key Cryptography to avoid “spam” similarity statements.

While our Similarity Ontology was designed with music similarity in mind, it is by no means limited to the domain of music. As we have shown, the framework is both flexible and extensible. We leave it to future work to explore how this framework might be applied in different domains and across domains.

7. NAMESPACES

The following namespaces are used throughout this work:

```
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix sim: <http://purl.org/ontology/similarity/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
@prefix log: <http://www.w3.org/2000/10/swap/log#>.
@prefix sig: <http://purl.org/ontology/signal/>.
@prefix ctr: <http://purl.org/ontology/ctr/>.
```

8. REFERENCES

[1] SPARQL query language for RDF, W3C recommendation, 2008.

[6] `http://xmlns.com/wot/0.1/`

[2] Samer Abdallah, Yves Raimond, and Mark Sandler. An ontology-based approach to information management for music analysis systems. In *120th Audio Engineering Society Convention*, 2006.

[3] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah McGuinness, Peter Patel-Schneijder, and Lynn Andrea Stein. OWL Web Ontology Language Reference. Recommendation, World Wide Web Consortium (W3C), February10 2004. See `http://www.w3.org/TR/owl-ref/`.

[4] D. Beckett. RDF/XML Syntax Specification (Revised). Recommendation, World Wide Web Consortium (W3C), 2004. Internet: `http://www.w3.org/TR/rdf-syntax/`.

[5] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. P. W. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music J.*, 28(2):63–76, 2004.

[6] Tim Berners-Lee. Notation 3, 1998. See `http://www.w3.org/DesignIssues/Notation3.html`.

[7] Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler. N3logic: A logical framework for the world wide web. *Theory and Practice of Logic Programming*, 2007.

[8] Chris Bizer, Richard Cyganiak, and Tom Heath. How to publish linked data on the web.

[9] Dan Brickley and Ramanatgan V. Guha. Rdf vocabulary description language 1.0: Rdf schema. W3c recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[10] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.

[11] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[12] Òscar Celma, Miquel Ramírez, and Perfecto Herrera-Boyer. Foafing the music: A music recommendation system based on rss feeds and user preferences. Proceedings of the 6th ISMIR, 2005.

[13] Paulo Pinheiro da Silva, Deborah L. McGuinness, and Richard Fikes. A proof markup language for semantic web services. *Inf. Syst.*, 31(4):381–395, 2006.

[14] Keith J. Holyoak. *The Cambridge Handbook of Thinking and Reasoning*. Cambridge University Press, Cambridge, UK, April 2005.

[15] Krzysztof Janowicz. Sim-dl: Towards a semantic similarity measurement theory for the description logic in geographic information retrieval. *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 1681–1692, 2006.

[16] Christoph Kiefer, Abraham Bernstein, and Markus Stocker. The fundamentals of isparql - a virtual triple approach for similarity-based semantic web tasks. In Karl Aberer and Key-Sun Choi, editors, *Proceedings of the 6th International Semantic Web Conference*, LNCS, pages 295–308, Berlin, 2007. Springer.

[17] Dekang Lin. An information-theoretic definition of similarity. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[18] B. Logan and A. Salomon. A music similarity function based on signal analysis. *Multimedia and Expo ICME*, pages 745–748, 2001.

[19] Francois Pachet. Knowledge management and musical metadata. *Encyclopedia of Knowledge Management*, 2005.

[20] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrival*. PhD thesis, Technischen Universität Wien, May 2006.

[21] Alexandre Passant and Yves Raimond. Combining social music and semantic web for music-related recommender systems. In *Semantic Web Workshop*, 2008.

[22] W. V. Quine. *Ontological relativity and other essays*. Columbia University Press, New York, NY, USA, 1969.

[23] Yves Raimond. *A distributed music information system*. PhD thesis, Queen Mary University of London, 2009.

[24] Yves Raimond, Samer Abdallah, Mark Sandler, and Frédérick Giasson. The music ontology. Proceedings of the 8th ISMIR, 2007.

[25] Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web, 2008.

[26] Roger N. Shepard. Geometrical approximations to the structure of musical pitch. *Physchological Review*, 89:305–333, 1982.

[27] Joshua B. Tenenbaum. Learning the structure of similarity. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge, MA, USA, 1996.

[28] Amos Tversky and Itamar Gati. Similarity, separability, and the triangle inequality. *Physchological Review*, 89:123–154, 1982.

INTERFACES FOR DOCUMENT REPRESENTATION IN DIGITAL MUSIC LIBRARIES

Andrew Hankinson[1] **Laurent Pugin**[2] **Ichiro Fujinaga**[1]
[1] Schulich School of Music of McGill University, Montréal (Quebec) Canada
[2] RISM Switzerland, Bern, Switzerland
{ahankinson,laurent,ich}@music.mcgill.ca

ABSTRACT

Musical documents, that is, documents whose primary content is printed music, introduce interesting design challenges for presentation in an online environment. Considerations for the unique properties of printed msic, as well as users' expected levels of comfort with these materials, present opportunities for developing a viewer specifically tailored to displaying musical documents. This paper outlines five design considerations for a music document viewer, drawing examples from existing digital music libraries. We then present our work towards incorporating these considerations in a new digital music library system currently under development.

1. INTRODUCTION

In 2008, the Swiss working group for the Répertoire International des Sources Musicales (RISM) project began work towards digitizing its national music collection. These digitized scores would be incorporated into an online catalogue of works, and would allow users of this system to view these items online. One crucial element for the success of this project was the implementation of software that presented these documents using musically consistent techniques.

The presentation of printed musical materials in an online environment poses interesting design challenges. Music and text documents have superficial similarities—they are written or printed on paper and bound in books—but they also differ significantly in their intended use, complexity of notation and stylistic considerations for presentation on the page.

When displaying music documents in an online environment these differences should be taken into account. Beyond putting the scanned content online, there needs to be consideration for how to show the material to users. As we will demonstrate in our literature review, the presentation of content can have a significant impact on a user's ability to navigate and comprehend the content itself. Next, we will propose five design considerations, formulated as requirements for implementation in a document viewer for a digital music library. Accompanying these design considerations we will show specific examples of how these have been implemented in existing digital library systems. Finally, we conclude with a brief discussion of our implementation of these design considerations, as well as commentary on possible directions for future work.

2. BACKGROUND

In his 1984 dissertation, Byrd [1] describes printed music—specifically, conventional music notation (CMN) —as a "modified coordinate system" that encapsulates semantic, syntactic, and graphic complexity occurring in four dimensions (pitch, time, loudness, and timbre). Accompanying the complexity of the music itself are practical considerations that play an integral role in the interpretation of the materials, such as page layouts, line justifications, and convenient page-turns. While these considerations are not part of the musical content, they are part of the total information content of the score. Put another way, while there is no one correct way to present the printed music, there are many wrong ways to present it that can lead to mis-interpretation of the music itself. The dimensionality and complexity of printed music, Byrd states, exceeds the complexity of printed text and is central to understanding the problems that exist with computerized analysis of these materials.

The delivery of information in online environments is an area of research that has received quite a bit of attention. In particular, Thong et al. [2] show that "[in] the context of digital libraries, it not only matters what we put on the screen, but how." They continue: "[the] way that information is arranged on the screen can influence the users' interaction with digital libraries beyond the effect of the information content."

Additional research has conclusively identified the affective relationship between the aesthetic perception of the materials and its effect on cognition and learning.

Kurosu and Kashimura [3] found that "[users] may be strongly affected by the aesthetic aspect of the interface even when they try to evaluate the interface in its functional aspects." Building on these findings, Tractinsky et al. [4] performed a study of automated teller machines and found "strong correlation between users' perception of an interface aesthetics and their perception of the usability of the entire system." They postulated that factors of aesthetics and usability can play a significant role in the overall satisfaction derived from an interface.

These studies' results are congruent with other work in the affective nature of human-computer interaction. When suggesting that "attractive things work better," Norman [5] (building on studies by Ashby et al. [6]) suggests that aesthetic interfaces can lead to a greater overall satisfaction in an interaction, which in turn can have significant effects on understanding the content. Increases in tension or anxiety, caused by unpleasant experiences with a system can negatively affect cognition of the material, leading not just to an unpleasant interaction, but also a decrease in the users' ability to understand the material itself.

While most usability research for digital libraries has focused specifically on textual materials, there has been work done on the evaluation of digital music library interfaces. Byrd and Crawford [7] touch on the topic of user interfaces for music information retrieval, simply stating that they are "hard." Byrd and Isaacson [8] address problems of music representation in a digital music library; however, they deal specifically with issues of notation layout, and not with interactions with digitized print materials.

The VARIATIONS project at Indiana University has conducted a number of usability studies on their system. Fuhman et al. [9] observed that non-musically trained users of their system took longer to complete musically-oriented tasks than musically-trained users, and gave a lower overall subjective rating to interfaces designed for displaying musical content. One possible explanation for this might be that musically-trained users have learned specific techniques for interacting with musical materials that are not shared by users who are unfamiliar with this content.

Finally, the SyncPlayer software [10] has received quite a bit of attention as a system that provides an easy-to-use interface for navigating score and audio representations of music. While this software presents an interesting interface for viewing and navigating complex scores, it was not included because it has not been used in a large-scale, public digital music library implementation.

3. DESIGN CONSIDERATIONS

As part of the design process for the document viewer, we identified five key considerations for designing an interface specifically for displaying printed music. These were formulated to encapsulate both the musical considerations of the documents, as well as some behavioural considerations of our target audience—musicologists and music researchers. For each consideration, we examined a number of existing systems used for displaying digital documents, musical or otherwise. By looking at these systems we were able to understand the current state of the art for displaying musical items, as well as discover interesting techniques to incorporate into our own implementation.

For a list of all the systems mentioned here, please refer to Table 1.

American Memory Project-Sheet: Music from the Civil War Era http://memory.loc.gov/ammem/cwmhtml/cwmhome.html
British Library "Turning the Pages" Project http://portico.bl.uk/onlinegallery/ttp/ttpbooks.html
Chopin Early Editions http://chopin.lib.uchicago.edu
Digital Image Archive of Medieval Music http://www.diamm.ac.uk
Google Books http://books.google.com
Inventions of Note Sheet Music Collection http://libraries.mit.edu/music/sheetmusic
Juilliard Manuscript Collection http://www.juilliardmanuscriptcollection.org
Lester S. Levy Sheet Music Collection http://levysheetmusic.mse.jhu.edu
Neue Mozart Ausgabe http://dme.mozarteum.at
Schubert Manusckripte http://www.univie.ac.at/wwtf/schubert
Sibley Music Library http://urresearch.rochester.edu/handle/1802/291
VARIATIONS Score Prototype http://www.dlib.indiana.edu/variations/scores
World Digital Library http://wdl.org

Table 1. Digital Music Libraries Examined

3.1 Preserve Document Integrity

One of the most common methods for presenting pages in a digital library is as a series of images on separate web pages, with navigation elements such as 'next' and 'previous' links, drop-down menus or hyperlinked page numbers as the primary means of navigating through the item. This method of document display suggests an 'image gallery' metaphor, rather than representing the

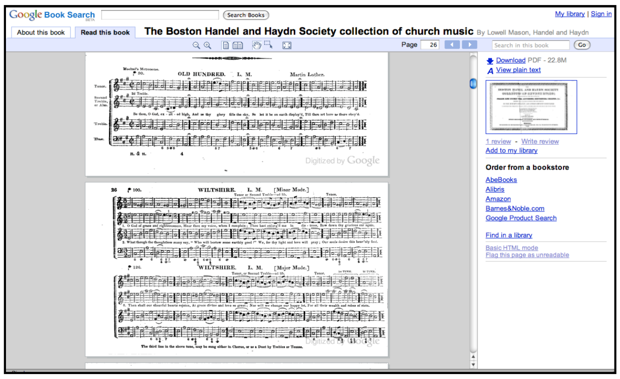


Figure 1. Google Books interface. The inner frame scrolls, with item metadata presented in the sidebar.

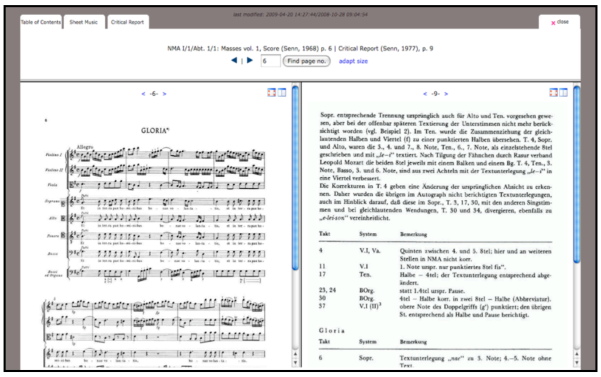


Figure 2. Side-by-side presentation of items in NMA. Musical material is presented on the left, and a critical report is on the right.

cohesive original document as a single entity. To preserve this cohesiveness, one of our design goals was to implement a display metaphor that preserved the original document integrity. Google Books, the Neue Mozart Ausgabe (NMA), and the VARIATIONS prototype viewer provide interesting examples of this functionality. These systems present the items as a single, scrollable entity embedded within a frame on the web page. This allows users to scroll very quickly through the item without having to click 'next' and 'back' links and wait for the page to reload.

A different technique was employed by the University of Illinois collection and the British Library's "Turning the Pages" project. These presented their documents using a book metaphor where users could use the mouse to 'turn' the pages. As a navigation system this was largely a novelty and presented some usability challenges for turning one page or many pages simultaneously. However, these systems excelled at presenting an accurate picture of the original page and book layout, an especially important consideration for musical materials.

3.2 Allow Side-by-side Comparison of Items

Musical documents can be divided into multiple physical items, with each item containing a portion of the complete musical work. Choir part books and orchestral instrument parts are common examples, but this can also extend to opera scores and libretti, early and later editions of a work, theory treatises and criticisms, adaptations, reductions, or various other modifications. It is not uncommon for scholars to need to consult multiple volumes for a single score.

Two systems, the Digital Image Archive of Medieval Music (DIAMM) and the NMA, had the facility for displaying multiple items, but neither of them allowed multiple musical items to be displayed simultaneously. DIAMM displayed corresponding scans from entries in a printed RISM catalogue that was digitized, while the NMA displayed scans from a published critical report on that piece of music (see Figure 2).

3.3 Provide Multiple Page Resolutions

When studying older manuscripts or printed works, the ability to view small details on a page, such as faint pencil markings or smudged note heads, can provide valuable information to the scholar. High-resolution images provides users with the ability to 'zoom in' on these markings, while lower resolution ones would allow them to move quickly through an entire document without having to navigate large pages.

Most of the systems examined provide more than one size of image. Typically, they would provide three page image sizes: a 'thumbnail' view for quick selection and browsing, a 'browser-safe' view for fitting in a browser, and a 'high resolution' view for downloading, printing, or further detailed study.

In two cases, DIAMM and the World Digital Library (WDL) provided methods for smoothly zooming in and out from the page images. In the case of DIAMM, they used an Adobe Flash-based viewer called "Zoomify," typically used for viewing high-resolution landscape photographs, while the WDL used a technology developed by Microsoft for their Photosynth viewer (see Figure 3).

3.4. Optimize Page Loading

Showing multiple high-resolution document pages presents significant challenges for network and browsing speeds. Furthermore, we know that our target user base often works in environments such as small libraries, monasteries, churches, or in rural locations, where bandwidth can be at a premium. To address these issues, one of our design goals was to only display the pages and areas of the page that the user was currently viewing. This would preclude the need to download an entire document of high-resolution page images if they

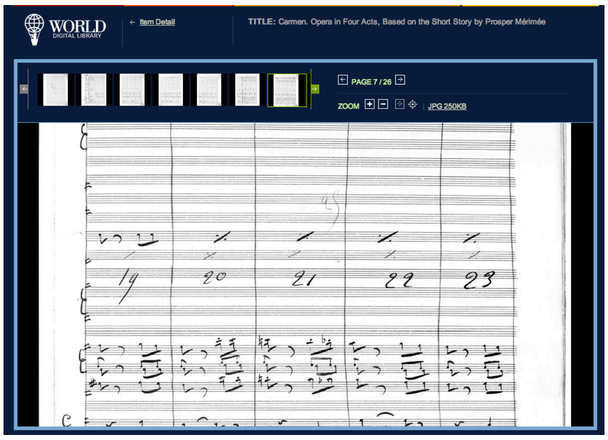

WORLD
DIGITAL LIBRARY
Item Detail
TITLE: Carmen. Opera in Four Acts, Based on the Short Story by Prosper Mérimée
PAGE 7 / 26
ZOOM
JPG 250KB

Figure 3. Zooming in on a manuscript in the World Digital Library

only wished to consult a single page.

As mentioned previously, Google Books uses an optimization technique that loads page images on demand. The Illinois Flip Book (beta) system seem to offer this as well, but at high zoom levels it required the user to download the whole high-resolution page image, slowing down the interaction.

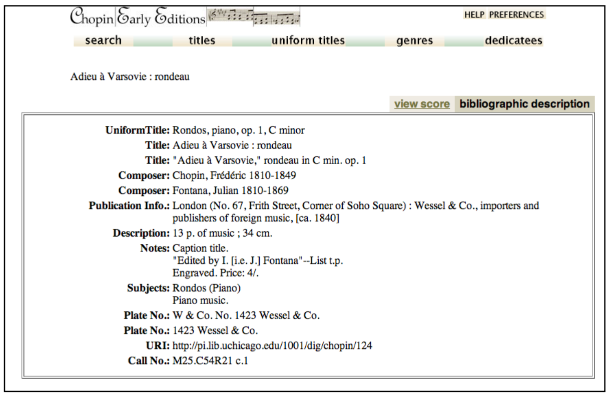
The viewing system for the Schubert Manuskripte library used a segmentation system to display single high-resolution scans of a single page. Each page image was broken into smaller image tiles that could be downloaded in parallel, theoretically speeding up the interaction. However, it seemed to use real-time image manipulation (e.g., re-sizing and rotating) on the server side, meaning that any speed optimization gained in parallel download was lost while the user waited for the server to recalculate the image.

3.5. Present Item and Metadata Simultaneously

The catalogue record of a document often contains more information than is immediately available in the item itself or can serve to correct erroneous or outdated information on the item. For example, some compositions have been commonly attributed to the wrong composer, or their catalogue of works may have updated numbers. Although this seems like a small interface consideration, many implementations we examined would open images in the current or new window, replacing or obscuring the metadata and causing users to constantly flip between two browser windows or use the 'back' and 'forward' browser buttons to switch between item and item record.

The reasons for this separation are varied. Some systems, e.g., Harvard, DIAMM, the University of Illinois, and Juilliard, used document presentation software separate from its catalogue to display the actual item. Other systems, such as the American Memory Project and the Levy Sheet Music Collection, separated the catalogue records and the navigation of the pages in the item on different web pages. Still others, such as the Sibley Music Library and the Inventions of Note collection, simply provided their items as PDFs to download.

The Chopin collection offered a "tab" for switching between the score and the bibliographic interface, (Figures 4 and 5) but switching between the two did not maintain the users' position in the score, reverting them to the view of the title page. Google Books and the VARIATIONS prototype feature a sidebar with some cataloguing information present, but the full catalogue entry was on a separate page.



Adieu à Varsovie : rondeau

UniformTitle:	Rondos, piano, op. 1, C minor
Title:	Adieu à Varsovie : rondeau
Title:	"Adieu à Varsovie," rondeau in C min. op. 1
Composer:	Chopin, Frédéric 1810-1849
Composer:	Fontana, Julian 1810-1869
Publication Info.:	London (No. 67, Frith Street, Corner of Soho Square) : Wessel & Co., importers and publishers of foreign music, [ca. 1840]
Description:	13 p. of music ; 34 cm.
Notes:	Caption title. "Edited by I. [i.e. J.] Fontana"--List t.p. Engraved. Price: 4/.
Subjects:	Rondos (Piano) Piano music.
Plate No.:	W & Co. No. 1423 Wessel & Co.
Plate No.:	1423 Wessel & Co.
URI:	http://pi.lib.uchicago.edu/1001/dig/chopin/124
Call No.:	M25.C54R21 c.1

Figure 4. Bibliographic Description Tab in the Chopin Early Editions.

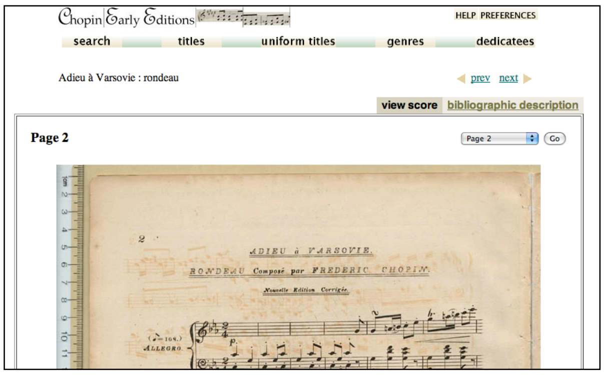


Figure 5. View Score tab in the Chopin Early Editions.

4. CURRENT WORK

While each design consideration we studied in our research can be found in several systems that we evaluated, our goal was to provide a system that would implement all of them. This viewer uses a number of technologies adopted from our examination of the existing solutions. Figure 6 shows a screenshot of our document viewer.

The unified document display methods found in the Google Books and VARIATIONS systems has been adopted. It has been enhanced to allow users to scroll both vertically and horizontally through an item, based on the page orientation of the item (vertical scrolling for items in portrait orientation, horizontal for items in landscape orientation).

For musical works with many physical items, panels in the viewer allow users to view one to four items simultaneously. These panels are synchronized so that the location in a score is maintained across all panels as the user scrolls through one panel. The synchronization is currently limited to movement or section indexing that has to be provided by hand by the cataloger.

In order to provide multiple page resolution while keeping page loading optimized at high and very high resolutions (600 dpi or higher), the system uses a tiling mechanism that separate the images into small tiles, enabling it to serve only the displayed part of the document. By restricting the download to only the tiles that are needed by the user, we avoid the need to download the entire high-resolution image to view only a specific portion of an image. When combined with the unified document approach, this means that users can very quickly scroll through a document and zoom in on a specific page or set of pages without having to download the entire item.

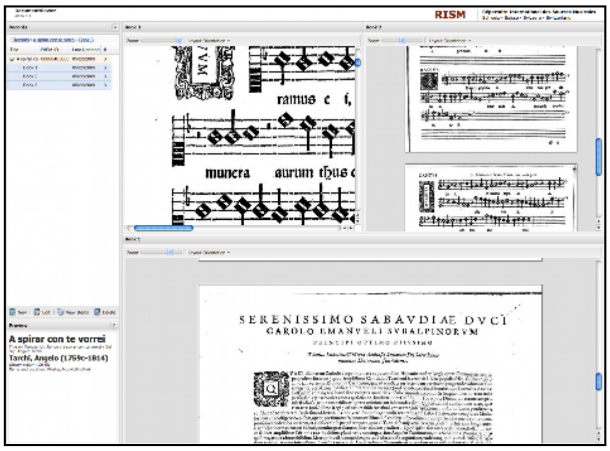

RISM
ramus e i,
munera aurum thus
SERENISSIMO SABAVDIAE DVCI
CAROLO EMANVELI SVBALPINORVM
A spirar con te vorrei
Tarchi, Angelo (1759c-1814)

Figure 6. The Swiss RISM digital music document viewer. Three separate documents are displayed in panels in the middle of the page and can be scrolled vertically or horizontally. Document metadata appears in the lower-left panel.

Finally, the simultaneous presentation of metadata was incorporated into the interface by employing a sidebar similar to the VARIATIONS prototype. This panel can be hidden and shown dynamically, allowing users to concentrate on viewing the item but giving them easy access to the full catalogue record without having to navigate to another page.

4.1. Technical aspects

Ruby-on-Rails and MySQL provide the data storage on the server side. The client interface uses the ExtJS Javascript Framework [11].

For multiple page resolutions and optimized page loading, the system uses the IIP Image Server [12] tiling system. The image server separates large, high-resolution images into separate 256×256 pixel tiles and serves them on-demand.

Javascript Object Notation (JSON) is used as a communication language between the database, tile server, and user interface. Client and server communication is performed asynchronously. From a user's perspective, this means that there are very few page refreshes and performance approaches that of a native application instead a website.

To create new documents in this system, the images representing the page images are placed in a ZIP file and uploaded through the interface. These are then unzipped on the server side and processed using the VIPS image processing software [13] to create a pyramid TIFF [14] file that contains a sequence of images at increasingly coarse resolutions, representing zoom levels for these images (see Figure 7).

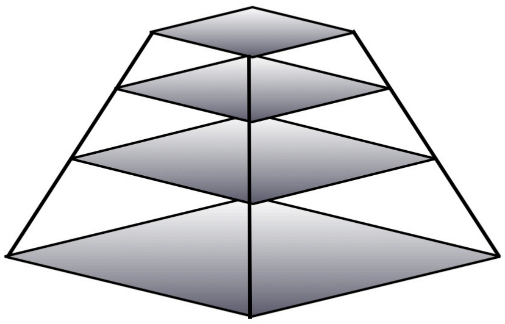


Figure 7. Pyramid TIFFs contain multiple resolutions of an image in a single file.

When a user requests a document through their web browser, the interface translates this into a request for the images and zoom level of the pages currently visible in the viewer. The images are then served to the client as separate tiles and re-assembled as a page image in the interface. This is repeated for each page so users can scroll through an entire document at very high resolutions without waiting for the whole document to download.

The software and display techniques presented here will be incorporated into a modular digital library system currently under development. Each component of this system uses open-source software, and we will be releasing this system under an open-source MIT license, freely available for implementation in existing digital library systems.

5. FUTURE WORK

The design considerations proposed here have not been verified by user testing. As part of our ongoing work on this software, we plan to study the impact of these design considerations on our target audience in real-world usage situations.

As part of the ongoing Swiss RISM project, the viewer interface will be integrated into the catalogue of Swiss musical works available online [15] as the documents are digitized.

6. CONCLUSION

This paper introduces our work towards a viewer for digital music documents, taking into account the unique properties of printed music and the expectations of users who use these systems. We also expect that having an architecture designed specifically for music documents will be of great benefit in the long run as it should facilitate the integration of other information research technologies specific to music, such as content-based synchronization or online optical music recognition.

7. ACKNOWLEDGEMENTS

This work was funded by the Swiss National Science Foundation and the Swiss Academy of Humanities and Social Sciences.

REFERENCES

[1] Byrd, D. 1984. Music notation by computer. PhD diss., Indiana University.

[2] Thong, J., W. Hong, and K.Y. Tam. 2002. Understanding user acceptance of digital libraries: What are the roles of interface characteristics, organizational context, and individual differences? *International Journal of Human-Computer Studies* 57 (3): 215–42.

[3] Kurosu, M., and K. Kashimura. 1995. Apparent usability vs. inherent usability: Experimental analysis on the determinants of the apparent usability. In *Proceedings of the Conference on Human Factors in Computing Systems,* 292–93.

[4] Tractinsky, N., A. Katz, and D. Ikar. 2002. What is beautiful is usable. *Interacting with Computers* 13 (2): 127–45.

[5] Norman, D. 2002. Emotion & design: Attractive things work better. *Interactions* 9 (4): 36–42.

[6] Ashby, F., A. Isen, and U. Turken. 1999. A neuropsychological theory of positive affect and its influence on cognition. *Psychological Review* 106 (3): 529–50.

[7] Byrd, D., and T. Crawford. 2002. Problems of music information retrieval in the real world. *Information Processing and Management* 38 (2): 249–72.

[8] Byrd, D., and E. Isaacson. 2003. Music representation in a digital music library. In *Proceedings of the Joint Conference on Digital Libraries,* 234–36.

[9] Fuhrman, M., D. Gauthier, and A. Dillon. 2001. *Usability test of Variations and DML prototypes.* http://variations2.indiana.edu/pdf/VariationsTest.pdf [Accessed 20 May 2009].

[10] Fremerey, C. 2006. SyncPlayer: A framework for content-based music navigation. Diplomarbeit, Univ. of Bonn.

[11] ExtJS Framework. http://extjs.com/ [Accessed 9 August 2009].

[12] Pillay, R., and D. Pitzalis. 2008. *IIP Image Server.* http://iipimage.sourceforge.net [Accessed 20 May 2009].

[13] Cupitt, J., and K. Martinez. 1996. VIPS: An image processing system for large images. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE)* 2663 (19): 19–28.

[14] Library of Congress. 2009. TIFF, Pyramid. http://www.digitalpreservation.gov/formats/fdd/fdd000237.shtml [Accessed 22 May 2009].

[15] RISM Switzerland Online Catalogue. http://www.rism-ch.org/ [Accessed 9 August 2009].

BODY MOVEMENT IN MUSIC INFORMATION RETRIEVAL

Rolf Inge Godøy
FourMs
Department of Musicology
University of Oslo
r.i.godoy@imv.uio.no

Alexander Refsum Jensenius
FourMs
Department of Musicology
University of Oslo
a.r.jensenius@imv.uio.no

ABSTRACT

We can see many and strong links between music and human body movement in musical performance, in dance, and in the variety of movements that people make in listening situations. There is evidence that sensations of human body movement are integral to music as such, and that sensations of movement are efficient carriers of information about style, genre, expression, and emotions. The challenge now in MIR is to develop means for the extraction and representation of movement-inducing cues from musical sound, as well as to develop possibilities for using body movement as input to search and navigation interfaces in MIR.

1. INTRODUCTION

There are strong links between music and body movement: Performers produce sound through movements, and listeners very often move to music, as can be seen in dance and innumerable everyday listening situations. The links between music and body movement have been discussed since antiquity, but it is mostly in the last decade that we have seen more systematic research efforts on this topic within fields such as music technology, music performance, and music cognition [1–3]. Despite this rapidly growing research in various music-related fields, the idea of body movement as an integral and ubiquitous part of both performance and perception of music seems so far not to have had many consequences for music analysis, music theory, and music information retrieval. Based on a quick survey of papers from recent ISMIR conferences as well as on the overview in [4], the papers that directly or indirectly are concerned with body movement seem limited to a few on query by humming and tapping, as well as some on beat tracking and tempo induction. Also, a cross-check on Google Scholar showed that out of 4670 hits on MIR, 3730 included "audio", 1990 "MIDI", while only 21 included "body movement". [1] It seems fair then to conclude that body movement has not been an important topic in MIR contexts.

Based on our own and various international colleagues' work of the past decade, we believe that body movement is not just something that incidentally co-occurs with music, but that body movement is integral to music as a phenomenon. We would go so far as to claim that our experience of music is based on the combination of sound and movement sensations, hence that music is a fundamentally embodied phenomenon [5, 6]. With such an understanding of music, it also becomes clear that sensations of music-related body movements are in fact highly salient features of music, and should be considered alongside various sonic features, e.g. pitch, melody, harmony, and timbre. Exploring music-related body movement then becomes an urgent task also in relation to MIR, and in this paper we shall try to give an overview of the kinds of body movement that could be of interest in MIR and how they can be studied. Finally, we shall present some suggestions for how body movements could be used in interfaces for the search and retrieval of music information.

2. MUSIC-RELATED MOVEMENT

It seems that listeners associate different kinds of body movement with the music they hear, or merely imagine. Here it can be useful to start by making the general distinction between *sound-producing* and *sound-accompanying* movements. Although this distinction may not always be so clear-cut, sound-producing movements are those that contribute to the production of musical sound, and sound-accompanying movements are those that are made in response to the sound being heard [3].

Sound-producing movements may further be divided into *excitatory* movements such as hitting, bowing, blowing, and *modulatory* movements such as those for making a vibrato or various timbral nuances. Associated with sound-producing movements we also have various types of *sound-facilitating*, *expressive*, and *communicative* movements, meaning movements that are not strictly speaking sound-producing but still play an important role in music performance. Sound-accompanying movements, on the other hand, are all kinds of movements that people may make to music such as in

[1] Search conducted 21 April 2009 using Google Scholar in English, and with a syntax of "Music Information Retrieval" + "...".

dancing, marching/walking, swaying, and gesticulating.

In practice, we may often see these different movement types occur together: it is possible to make movements that partly reflect the sound-production, partly are more independent of the sound-production, e.g. mimicking a solo drum passage with the hands at the same time as swaying the whole body to the meter of the music. We may also see performers making movements that are partly necessary for producing sound, and partly more theatrical for the benefit of the audience, e.g. lifting the hand high up before striking a chord on a guitar. This means that music-related movements may be multi-functional in that they serve several different purposes at the same time.

We believe that musical sound itself also conveys salient movement images that are related to listeners' sensations of effort (tense, relaxed, fast, slow, etc.) as well as to kinematics or geometry of musical instruments (register, up/down, position, posture, etc. in relation to instruments). Studies of so-called 'air-instrument' performance such as 'air guitar', 'air drums', and 'air piano' suggest that even listeners with little or no formal musical training are able to have images of sound-producing movements that reproduce both the effort and the kinematics of the imagined sound-production actions, i.e. they manage to follow the spatiotemporal unfolding of instrumental performance quite well as if they were actually playing the music themselves [7].

As for various kinds of sound-accompanying movement afforded by musical sound, a study of 'free dance' to music [2] shows that professional dancers tend to agree when it comes to the sensation of effort or energy in dance movements, although there are variations in the kinematics (geometry) of the movements [8, 9]. Furthermore, studies of 'sound-tracing' show that listeners with variable levels of musical training (ranging from none to professional level training) also seem to spontaneously associate various shapes with the musical sound that they hear [10]. In these studies, listeners were asked to draw on a digital tablet the shape they associated with a sound fragment immediately after they had heard the fragment. Figure 1 shows the sound-tracings of 9 participants to a sound taken from the contemporary western music repertoire. This sound consists of a high-pitched attack on a triangle, followed by a downward glissando on strings, and ending up with a drum roll [11]. The excerpt is rather unconventional with regards to melodic, harmonic, and timbral features, but as we can see from the images of the sound-tracings, there still seems to be some level of consensus between the nine listeners as to the movement shape that was afforded by the sound.

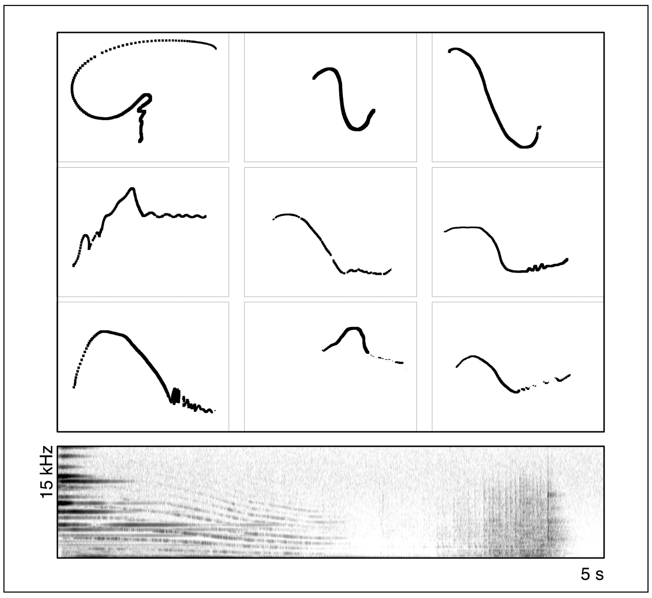


Figure 1. Sound-tracings by nine listeners of the sound fragment built up of an initial triangle attack, a downward glide in the strings and a final drum roll (spectrogram at the bottom) [11].

3. GLOBAL-LOCAL

It does not seem farfetched to suggest that listeners' music-related movements often match well the overall motion and emotion features of the musical sound, e.g. calm music tends to induce calm movements, agitated music tends to induce agitated movements, accentuated music tends to induce jerky movements, etc. The details of the movements may vary, however, something that may be seen both from qualitative annotations [8], as well as from quantitative data. An example of the latter may be seen in how the*quantity of motion* seems to correlate quite well with the dynamics of the waveform of the sound [7]. Similarly, *motiongrams* [3] are useful for displaying movement from video material. Figure 2 shows an example of how a motiongram of the hand movements of a pianist can be used together with the spectrogram of the resultant sound to study relationships between movement features and sonic features in a 20 seconds excerpt from the last movement of Beethoven's Tempest Sonata.

Visual representations such as motiongrams and spectrograms make it possible to move between global and more local perspectives, i.e. facilitates the correlation of music-related movement at different timescales with corresponding sonic features at different timescales. Here it could be useful to identify three different timescale levels when studying sound and movement in music:

Sub-chunk level: the level of perceiving continuous sound (pitch, timbre, and intensity) and movement (location, force, etc.).

Chunk level: sound fragments and actions that are perceived holistically and that may allow for the perception of rhythmical, textural, and melodic patterns, as well as tonal/modal and harmonic features, and importantly, also expressive features.

[2] The only instruction given was to make spontaneous movement to the musical excerpts upon first hearing.

[3] A motiongram is a visual representation of movement in a video, created by spatially reducing frame-differenced video images, see [9] for details

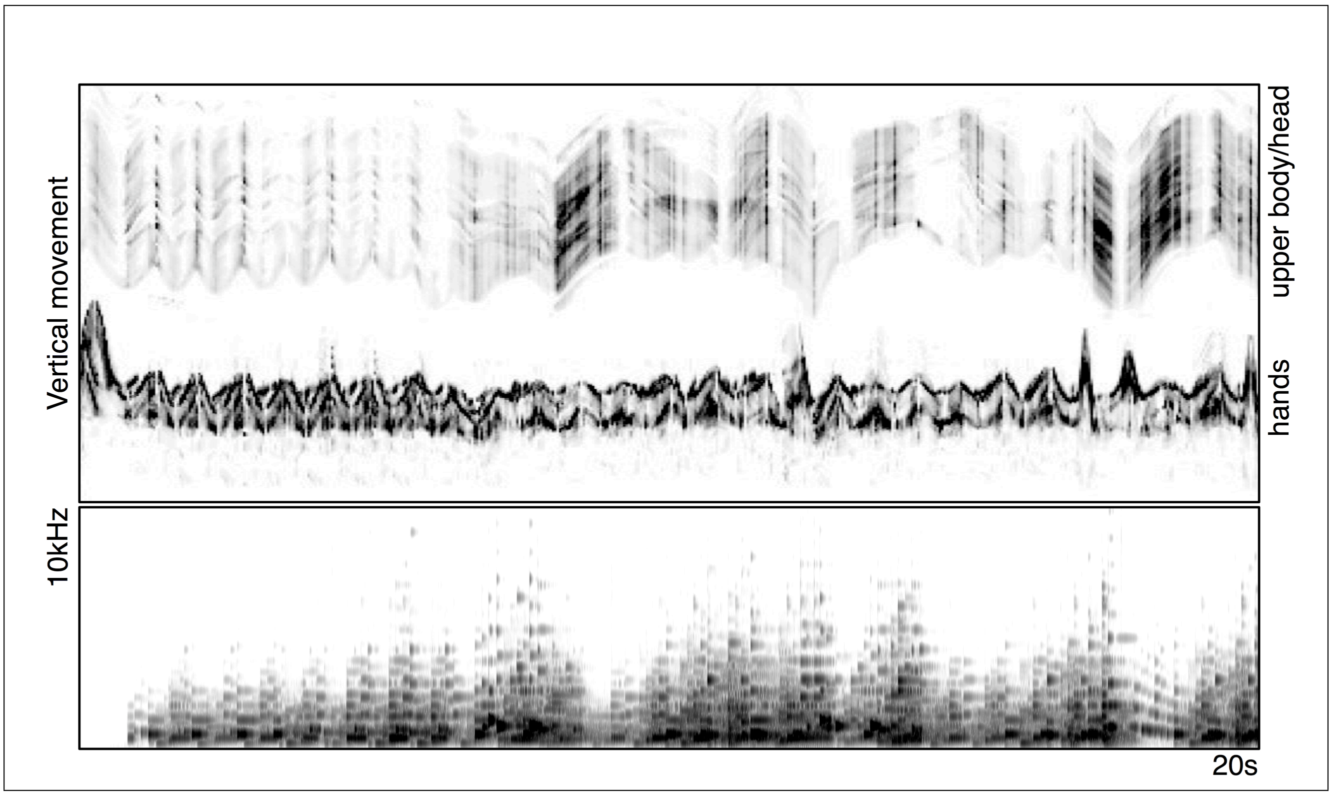


Figure 2. Motiongram of hand movement (top) and spectrogram (bottom) of the corresponding sound in a 20 seconds excerpt (first 30 measures) from the last movement of Beethoven's *Tempest Sonata* performed by François-René Duchable [12]. Notice the correlation between hand movements and the sound, as well as the sway in the upper body.

Supra-chunk level: several chunks are concatenated into larger-scale entities such as whole sections, tunes, movements, and even whole works.

We believe that the chunk-level, in the range of approximately 0.5 to 5 seconds, may be seen as the most important for identification of musical style, mode of performance, as well as emotive features. As suggested by Pierre Schaeffer's work on sonic objects several decades ago [13,14] and recently by work on more traditional western music [15], the chunk level seems to be more important than larger scale levels in music. Interestingly, and probably not accidentally, the temporal size of basic action units fits well with that of sonic objects, as well as with various other constraints on attention and memory, see [16] for a summary.

From what emerges of the sound-movement correspondences mentioned above, we think it is plausible to think of *gestural-sonic objects* in music [17]. This means multimodal units that combine sound and movement so that in addition to various sonic features we also have movement features such as proprioceptive, haptic, and visual images of trajectories and postures. This also means that there are movement-related schemata and constraints at work in gestural-sonic objects, i.e. various biomechanical and neurocognitive constraints such as limits to speed of movement, need for rests, etc., as well as the phenomena of *phase transition* and of *coarticulation*. Phase transitions mean that the speed of movement will lead to different groupings, e.g. speeding up will at some tempo threshold lead to fusion of pulses into a higher order pulse, slowing down will at some tempo threshold lead to fission of pulses into subdivision pulses. Coarticulation means that otherwise distinct sounds and movements will be hierarchically subsumed and contextually smeared so as to produce new emergent sensations, e.g. otherwise singular tone-events and movements fuse into superordinate phrases and movement shapes. Coarticulation seems to be one of the most important elements in the formation of chunks, and furthermore, concerns both the generation and the perception of musical sound [16].

Gestural-sonic images may be flexible, both with respect to resolution or acuity of detail, and with respect to generality by the principle of so-called *motor equivalence*. Motor equivalence means that motor images of singular actions may be generalized so as to encompass different versions of the action, allowing transfers and at the same time preserve basic cognitive schemata across variations. An example this is how the general category of 'hitting' is applicable to all percussion instrument actions, with or without mallets, as well as to all keyboard and struck string instruments.

4. TYPOMORPHOLOGY OF GESTURAL-SONIC OBJECTS

With chunk-level gestural-sonic objects as the basic local focus, we can differentiate various types as well as various features of such objects. Following the pioneering work of Pierre Schaeffer [13, 14], we can proceed in a top-down manner starting with depicting the global features of sonic objects and proceed on to successively finer differentiations of features. The main principle for Schaeffer was the subjective images of sonic objects, and where establishing correlations between these subjective images and the acoustic substrate of the sonic objects was seen as a long-term goal. It is also important to keep in mind that the ambition of Schaeffer was a universally applicable theory, equally valid for sonic objects in electroacoustic, instrumental, or vocal music, and applicable across different genres and musical cultures. Hence, such an approach could be seen as very much in accordance with a more open-ended, universal approach to MIR.

For a start, Schaeffer suggested three main classes of sounds based on their mode of production:

Impulsive: sounds that have a percussion like quality with a sudden onset followed by a decay, i.e. a discontinuous transfer of energy such as in hitting or kicking.

Sustained: a continuous transfer of energy so that the sound would be more or less stable throughout its duration such as in bowing, stroking, or blowing.

Iterative: sounds produced by a rapid series of impulses such as in a drum roll or in a tremolo.

It is the energy envelope of the sound that reflects the underlying assumed mode of sound-production, hence, that these sonic object types are transducers of movement information. This movement information can also be applied to pitch-related information with the following three main types:

Pitched: a more or less clearly perceptible and stable pitch throughout the duration of the sonic object.

Non-pitched: inharmonic or variably noise-dominated sounds with ambiguous or unclear pitch.

Variable: sensation of pitch that varies throughout the sonic objects, e.g. by glissando or vibrato.

Schaeffer combined these three pitch-related types with the three dynamic envelope types mentioned above into a 3 x 3 matrix of basic sonic objects in what he called the *typology*. The typology of sonic objects was a first and rough categorization to be followed by a more detailed depiction of features in what was called the *morphology* of the sonic objects. The morphology is basically concerned with the 'internal' features of the sonic objects such as its various pitch-related, dynamic, and/or timbral evolutions and fluctuations in the course of time. Two of the most prominent features of the morphology are the following:

Grain: fast fluctuations within the sound such as in the 'grainy' sound of a deep bassoon tone or in a flute flatterzunge.

Motion: slower fluctuations within the sound such as in slow ostinato or other textural movements. [4]

These features can be thought of as dimensions of sonic objects, and may also be further differentiated, e.g. the speed and amplitude of the grain fluctuations may be thought of as sub-dimensions, and variations in speed and amplitude may be thought of as further sub-dimensions to these dimensions. The exploration of thresholds for different feature values in relation to sound categories is then made possible, something that is useful for trying to determine categorical thresholds for salient features of sonic objects, hence for sonic features in general in a MIR context.

The typology and the morphology of sonic objects can be combined into an analytic system that for short is called the *typomorphology* of sonic objects. The general strategy here is then that of first attaching metaphorical labels to perceptually relevant (or salient) features of the musical sound, and then proceeding to differentiate various sub-features.

In summary, we believe that most (if not all) features of musical sound may be correlated to some kind of body movement. This is actually the main point of motor theory and embodied cognition, namely that we perceive by correlating whatever we hear (or see) to mental images of movement [6, 7].

5. SUGGESTIONS FOR IMPLEMENTATIONS

Given the abovementioned documentation of links between sound and body movement, the challenge now is to integrate our knowledge of such sound–movement links in audio analysis so that this can be useful in a MIR context.

Several of the features mentioned above can readily be found in audio using traditional analysis techniques. For example, the typological features can be correlated to the amplitude envelope of a sound signal and/or to the pitch contour or fluctuations in the spectral centroid. Details in the morphology, on the other hand, require more studies to be effectively implemented in a machine-based system. While it could be possible to implement this based on analysis of the sound alone, we believe that it may be worthwhile to also look at the movement of performers as well as listeners when they experience music.

As an example, consider the sensation of an undulating or even circular motion that we would assume many listeners would experience in the example illustrated with the motiongram in Figure 2. Although we may find considerable variation in the style of playing this piece, one source of such an undulating motion could be found in the sound-producing actions of the pianist. To an expert musician it might be natural or even obvious to predict from the score that pianists would tend to make this kind of undulating movements, yet it is an element that we believe could be captured and included in MIR as a feature of the music.

Figure 3 shows a graph of the movements of the wrists and elbows of a pianist performing the first 8 measures (with the upbeat figure) of the same piece as in Figure 2.

[4] 'Motion' is sometimes also rendered as 'gait' or 'allure' in English.

The graph is based on recordings with an infrared motion capture system and shows the markers' displacement along the keyboard (i.e. the horizontal plane). This is of course a crude simplification of the richness of the performance, yet we believe it does convey the salient feature of the undulating motion of this piece.

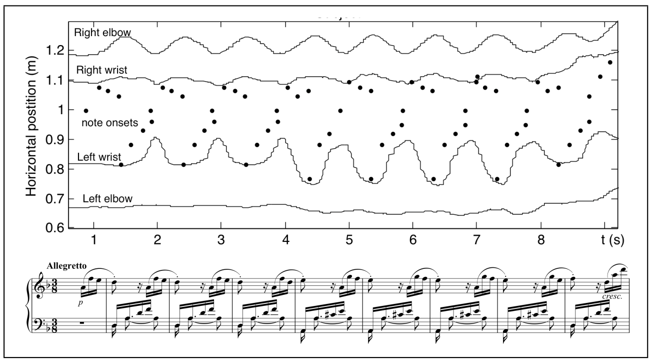


Figure 3. Trajectories of the wrists and elbows of a pianist performing the first 8 measures (and the upbeat measure) of the same Beethoven example as in Figure 2. The marked onset points are recorded from MIDI output from the digital piano used in the study.

Moving towards the analysis of body movement in a MIR context necessitates techniques to represent, store and navigate such movement data. We are here thinking about representations of data in many different forms, e.g.:

- Continuous data from various types of motion capture systems.
- Graphical representations of movement, both static and animated.
- Analyzed movement and gesture data in a structured and symbolic form.
- Various verbal movement metaphors.

Although there exist formats and standards that handle these types of data in other fields than music, we believe it is necessary to develop solutions that are specific to musical applications [18]. One of the most important parts here is to handle synchronisation between movement data, audio, video, MIDI, etc. We are not aware of any solutions that handle this issue in its full complexity, so for that reason we are currently developing the *Gesture Description Interchange Format* [5] (GDIF) as a system for streaming and storing motion capture data [19]. Equally important here is to work out a set of movement descriptors, and sound–movement descriptors, that are useful in a MIR context.

Also, considering that a substantial amount of music is readily available as audiovisual material (e.g. music videos of various kinds), this could be exploited if there were more readily available methods for analyzing both audio and video, and most importantly, for analyzing the *relationships* between features extracted from audio and video. This could then take into account the cross-modal interactions happening in our perception of audiovisual material, as documented in e.g. [20].

[5] http://www.gdif.org

Finally, including an embodied perspective in MIR research could also open for new applications of search and retrieval of music through body movement. Using various types of motion capture techniques, ranging from camera-based to sensor-based systems, users could explore a large music collection through body movement. While this could certainly be done in low-dimensional features spaces, we believe that systems that manage to connect complex body movements to complex sound features will open for new and exciting ways of exploring the multidimensionality of musical sound, e.g. as implemented in software for concatenative synthesis [21]. Considering the positive results of the studies of air-performance and sound-tracing as mentioned above, this is something that both novices and experts should be able to do without a too high learning threshold.

It could be useful to regard music-related body movement as a link between otherwise separate elements in western musical thought: the acoustic signal, symbolic notation, and higher level aesthetic and semiotic significations of music. This is because music-related body movement may encompass all these elements at once: On one side the continuous body movement relates to the continuous acoustic signal, with sound-producing movements incorporating the tone events of notational symbols, and with various types of expressive features in the movement touching on aesthetic and semiotic elements. On the other side, music-related body movement contain valuable information of the musical experience that is not present in the audio itself, but which is often available in video material accompanying the sound.

6. CONCLUSIONS

Although we still have a long way to go in exploring music-related body movement and its relationship to musical sound, it seems that we already have reasonable grounds for claiming that sensations of body movement are essential in musical experience. Actually, we would even claim that sensations of body movement are one of the most salient features of musical style and genre, and could for this reason alone be an important element in the development of MIR. When we rather optimistically believe that music-related body movement has great (and mostly untapped) potential for MIR, we are also acutely aware of great challenges here, challenges that may be summarized as follows:

- Development of signal processing methods for extracting movement-inducing cues from audio.
- Development of video processing methods for extracting features of music-related body movement.
- Development of taxonomies and formats for handling such multimodal features in MIR systems.

- Development of solutions for using body movement in searching, retrieval, and navigation in audio or audiovisual music files.

On the way to this, we need to continue working on what movement sensations listeners have to music, painstakingly building up our knowledge of subjective movement sensations and correlating these with lower-level signal-based features of musical sound.

7. REFERENCES

[1] M. M. Wanderley and M. Battier, eds., *Trends in Gestural Control of Music [CD-ROM]*. Paris: IRCAM – Centre Pompidou, 2000.

[2] A. Gritten and E. King, eds., *Music and Gesture*. Hampshire: Ashgate, 2006.

[3] R. I. Godøy and M. Leman, *Musical Gestures: Sound, Movement, and Meaning*. New York: Routledge, 2009 (in press).

[4] J. S. Downie, "The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research," *Acoust. Sci. & Tech*, vol. 29, no. 4, pp. 247–255, 2009.

[5] R. I. Godøy, "Motor-mimetic music cognition," *Leonardo*, vol. 36, pp. 317–319, August 2003.

[6] M. Leman, *Embodied Music Cognition and Mediation Technology*. Cambridge, MA: The MIT Press, 2007.

[7] R. I. Godøy, E. Haga, and A. R. Jensenius, "Playing 'air instruments': Mimicry of sound-producing gestures by novices and experts," in *Gesture in Human-Computer Interaction and Simulation, GW 2005* (S. Gibet, N. Courty, and J.-F. Kamp, eds.), vol. LNAI 3881, pp. 256–267, Berlin: Springer-Verlag, 2006.

[8] E. Haga, *Correspondences between music and body movement*. PhD thesis, University of Oslo, 2008.

[9] A. R. Jensenius, *Action–Sound : Developing Methods and Tools to Study Music-Related Body Movement*. PhD thesis, University of Oslo, 2007.

[10] R. I. Godøy, E. Haga, and A. R. Jensenius, "Exploring music-related gestures by sound-tracing - a preliminary study," in *Proceedings of the COST287-ConGAS 2nd International Symposium on Gesture Interfaces for Multimedia Systems* (K. Ng, ed.), (Leeds), pp. 27–33, 2006.

[11] P. Schaeffer, "Sound fragment from cd3, track 13, 20"-29" (no original source indicated) in [14]," in *Solfège de l'objet sonore*, Paris: (with sound examples by G. Reibel & B. Ferreyra), INA/GRM, 1998, first published in 1967.

[12] F.-R. Duchable, "Beethoven Concertos pour piano 1 and 3. A la decouverte des Concertos. François-René Duchable, piano, John Nelson, conductor, Ensemble Orchestral de Paris." [DVD] Harmonia Mundi, 2003.

[13] P. Schaeffer, *Traité des objets musicaux*. Paris: Editions du Seuil, 1966.

[14] P. Schaeffer, *Solfège de l'objet sonore*. Paris: (with sound examples by G. Reibel & B. Ferreyra), INA/GRM, 1998, first published in 1967.

[15] Z. Eitan and R. Granot, "Growing oranges on mozart's apple tree: 'inner form' and aesthetic judgment," *Music Perception*, vol. 25, no. 5, pp. 397–417, 2008.

[16] R. I. Godøy, "Reflections on chunking," in *Systematic and Comparative Musicology: Concepts, Methods, Findings. Hamburger Jahrbuch für Musikwissenschaft* (A. Schneider, ed.), vol. 24, pp. 117–132, Vienna: Peter Lang, 2008.

[17] R. I. Godøy, "Gestural-sonorous objects: embodied extensions of Schaeffer's conceptual apparatus," *Organised Sound*, vol. 11, no. 2, pp. 149–157, 2006.

[18] A. R. Jensenius, A. Camurri, N. Castagne, E. Maestre, J. Malloch, D. McGilvray, D. Schwarz, and M. Wright, "Panel: the need of formats for streaming and storing music-related movement and gesture data," in *Proceedings of the 2007 International Computer Music Conference*, (Copenhagen, Denmark), pp. 13–16, 2007.

[19] A. R. Jensenius, K. Nymoen, and R. I. Godøy, "A multilayered GDIF-based setup for studying coarticulation in the movements of musicians," in *Proceedings of the 2008 International Computer Music Conference*, (Belfast), pp. 743–746, 2008.

[20] B. Vines, C. Krumhansl, M. Wanderley, and D. Levitin, "Cross-modal interactions in the perception of musical performance," *Cognition*, vol. 101, pp. 80–113, 2005.

[21] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-time corpus-based concatenative synthesis with Catart," in *Proceedings of the 9th Int. Conference on Digital Audio Effects (DAFx-06)*, (Montreal), 2006.

WHO IS WHO IN THE END? RECOGNIZING PIANISTS BY THEIR FINAL RITARDANDI

Maarten Grachten
Dept. of Computational Perception
Johannes Kepler University, Linz, Austria
maarten.grachten@jku.at

Gerhard Widmer
Austrian Research Institute for
Artificial Intelligence, Vienna, Austria
Dept. of Computational Perception
Johannes Kepler University, Linz, Austria
gerhard.widmer@jku.at

ABSTRACT

The performance of music usually involves a great deal of interpretation by the musician. In classical music, final ritardandi are emblematic for the expressive aspect of music performance. In this paper we investigate to what degree individual performance style has an effect on the form of final ritardandi. To this end we look at interonset-interval deviations from a performance norm. We define a criterion for filtering out deviations that are likely to be due to measurement error. Using a machine-learning classifier, we evaluate an automatic pairwise pianist identification task as an initial assessment of the suitability of the filtered data for characterizing the individual playing style of pianists. The results indicate that in spite of an extremely reduced data representation, pianists can often be identified with accuracy significantly above baseline.

1. INTRODUCTION AND RELATED WORK

The performance of music usually involves a great deal of interpretation by the musician. This is particularly true of piano music from the romantic period, where performances are characterized by large fluctuations of tempo and dynamics. The expressive interpretation of the music by the musician is crucial for listeners to understand emotional and structural aspects of the music (such as voice and phrase structure) [1–3]. In addition to these functional aspects of expressive music performance, there is undeniably an aspect of personal style. Skilled musicians tend to develop an individual way of performing, by means of which they give the music a unique aesthetic quality (a notable example of this is the legendary pianist Glenn Gould). Although the main focus in music performance research has been on functional aspects of expression, some studies also deal with individual performance style. Through analysis of listeners ratings on performances, Repp characterized pianists in terms of factors that were mapped to adjective pairs [4]. In [5], a principal component analysis of timing curves revealed a small set of significant components that seem to represent performance strategies that performers combine in their performances. Furthermore, a machine learning approach to performer identification has been proposed by Stamatatos and Widmer [6], where performers are characterized by a set of features relating to score-related patterns in timing, dynamics and articulation. Saunders et al. [7] represent patterns in timing and dynamics jointly as strings of characters, and use string-kernel classifiers to identify performers.

It is generally acknowledged in music performance research that, although widely used, the mechanical performance (implying constant tempo throughout a piece or musical part) is not an adequate performance norm for studying expressive timing, as it is not the way we generally believe the music should sound. As an alternative, models of expressive timing could be used, as argued in [8]. However, only few models exist that model expressive timing in general [9, 10]. Because of the complexity and heterogeneity of expressive timing, most models only describe specific phenomena, such as the timing of grace notes [11], or the final ritardando [12, 13].

This paper addresses systematic differences in the performance of final ritardandi by different pianists. In a previous study [14] on the performance of final ritardandi, a kinetic model [13] was fitted to a set of performances. Although in some cases systematic differences were found between pianists, in general the model parameters (describing the curvature and depth of the ritardando) tend to reflect primarily aspects of the piece, rather than the individual style of the pianist. Given this result, a possible approach to study performer-specific timing in ritardandi would be by subtracting the fitted model from the modeled timing data and looking performer-specific patterns in the residuals. A problem with this approach is that the kinetic model is arguably too simple, since it models tempo as a function of score time only, and is ignorant of any structural aspects of the music, which also have an effect of the tempo curve [15]. As a result of this, residuals in the data with respect to the fitted model are likely to contain patterns related to piece-specific aspects like rhythmic grouping.

In this study, in order to minimize the amount of piece-specific information present in the residuals, we compute the average performance per piece and subtract it from each performance of that piece. In addition to this, we filter the residual data based on an estimation of its significance. This estimation is obtained from an analysis of data annotation divergences for a subset of the data. The resulting data contain the deviations from the common way of playing the ritardandi that are unlikely to be due to measurement errors.

Our long-term goal is to develop a thorough and sensible way of interpreting deviations of performance data with respect to some *performance norm*, be it either a model, or as in this study, a norm derived from the data. To obtain a first impression of the potential of characterizing artists by this method of analyzing the data, we defined a pairwise pianist identification task (as in [6]). Using a data set consisting of performances of ritardandi in Chopin's Nocturnes by a number of famous pianists, we show that pianists can be identified based on regularities in the way they deviate from the performance norm.

In section 2, we describe the acquisition and content of the data set. Section 3 documents the data processing procedure. Results of the pianist classification task are presented and discussed in section 4, and conclusions and future work in section 5.

2. DATA

The data used here consists in measurements of timing data of musical performances taken from commercial CD recordings of Chopin's Nocturnes. The contents of the data set are specified in table 1. We have chosen Chopin's Nocturnes since they exemplify classical piano music from the romantic period, a genre which is characterized by the prominent role of expressive interpretation in terms of tempo and dynamics. Furthermore, the music is part of a well-known repertoire, performed by many pianists, facilitating large scale studies.

Tempo in music is usually estimated from the interonset intervals of successive events. A problematic aspect of this is that when a musical passage contains few events, the obtained tempo information is sparse, and possibly unreliable, thus not very suitable for studying tempo. Therefore, through inspection of the score, we selected those Nocturnes whose final passages have a relatively high note density, and are more or less homogeneous in terms of rhythm. In two cases (Op. 9 nr. 3 and Op. 48 nr. 1), the final passage consists of two clearly separated parts, both of which are performed individually with a ritardando. These ritardandi are treated separately (see table 1). In one case (Op. 27 nr. 1), the best-suited passage is at the end of the first part, rather than at the end (so strictly speaking, it is not a *final* ritardando).

The data were obtained in a semi-automated manner, using a software tool [16] for automatic transcription of the audio recordings. From the transcriptions generated in this way, the segments corresponding to the final ritardandi were extracted and corrected manually by the authors, using *Sonic Visualizer*, a software tool for audio annotation and analysis [17].

3. METHOD

As mentioned in section 1, the expressive timing data is expected to have a strong component that is determined by piece-specific aspects like rhythmical structure and harmony. In order to focus on pianist-specific aspects of timing, it is helpful to remove this component. In this section, we first describe how the IOI data are represented. We then propose a filter on the data based on an estimate of the measurement error of IOI values. Finally, we describe a pianist identification task as an assessment of the suitability of the filtered data for characterizing the individual playing style of pianists.

3.1 Calculation of deviations from the performance norm

The performance norm used here is the average performance per piece. That is, for a piece k, Let M be the number of pianists, and N_k be the number of measured IOI times in piece k. We use $\mathbf{v}_{k,i}$ to denote the vector of the N_k IOI values of pianist i in piece k. Correspondingly, $\mathbf{u}_{k,i}$ is the IOI vector of pianist i for piece k, centered around zero ($\bar{\mathbf{v}}_{k,i}$ being the mean of all IOI's in $\mathbf{v}_{k,i}$):

$$\mathbf{u}_{k,i} = \mathbf{v}_{k,i} - \bar{\mathbf{v}}_{k,i} \tag{1}$$

The performance norm $\mathbf{a}_k$ for piece k is defined as the average over pianists per IOI value:

$$\mathbf{a}_k(j) = \frac{1}{M}\sum_{i=1}^{M} \mathbf{u}_{k,i}(j) \tag{2}$$

where $\mathbf{a}_k(j)$ is the j-th IOI value of the average performance of piece k.

Figure 1 shows the performance norms obtained in this way. Note that most performance norms show a two stage ritardando, in which a gradual slowing down is followed by a stronger decrease in tempo, a general trend that is also observed in [12]. The plots show furthermore that in addition to a global slowing down, finer grained timing structure is present in some pieces.

3.2 Estimation of measurement error

An inherent problem of empirical data analysis is the presence of measurement errors. As described above, the timing data from which the tempo curves are generated is obtained by measurement of beat times from audio files. The data is manually corrected, but even manually the exact time of some note onsets is hard to identify, especially when the pianist plays very softly while using the sustain pedal. Therefore, it is relevant to investigate to which degree different beat time annotations of the same performance differ from each other. This gives us an idea of the size of the measurement error, and allows us to distinguish significant deviations from the performance norm from the non-significant deviations.

Pianist	Year	Op.9 nr.3 rit1	Op.9 nr.3 rit2	Op.15 nr.1	Op.15 nr.2	Op.27 nr.1	Op.27 nr.2	Op.48 nr.1 rit1	Op.48 nr.1 rit2
Argerich	1965			X					
Arrau	1978	X	X	X	X	X	X	X	X
Ashkenazy	1985	X	X	X	X	X	X	X	X
Barenboim	1981	X	X	X	X	X	X	X	X
Biret	1991	X	X	X	X	X	X	X	X
Engerer	1993	X	X	X	X	X	X	X	X
Falvai	1997	X	X	X	X	X	X	X	X
Harasiewicz	1961	X	X	X	X	X	X	X	X
Hewitt	2003	X	X	X	X	X	X	X	X
Horowitz	1957			X		X			
Kissin	1993					X	X		
Kollar	2007	X	X	X	X		X	X	X
Leonskaja	1992	X	X	X	X	X	X	X	X
Maisenberg	1995			X					
Mertanen	2001	X	X	X	X	X	X		
Mertanen	2002							X	X
Mertanen	2003							X	X
Ohlsson	1979	X	X	X	X	X	X	X	X
Perahia	1994			X					
Pires	1996	X	X	X	X	X	X	X	X
Pollini	2005	X	X	X	X	X	X	X	X
Richter	1968			X					
Rubinstein	1937	X	X	X	X	X	X	X	X
Rubinstein	1965	X	X	X	X	X	X	X	X
Tsong	1978	X	X	X	X	X	X	X	X
Vasary	1966	X	X	X	X	X		X	X
Woodward	2006	X	X	X	X	X	X	X	X
d´Ascoli	2005	X	X	X	X	X	X	X	X

Table 1. Performances used in this study. The symbol "X" denotes the presence of the corresponding combination of pianist/piece in the data set. The additions "rit1" and "rit2" refer to two distinct ritardandi within the same piece

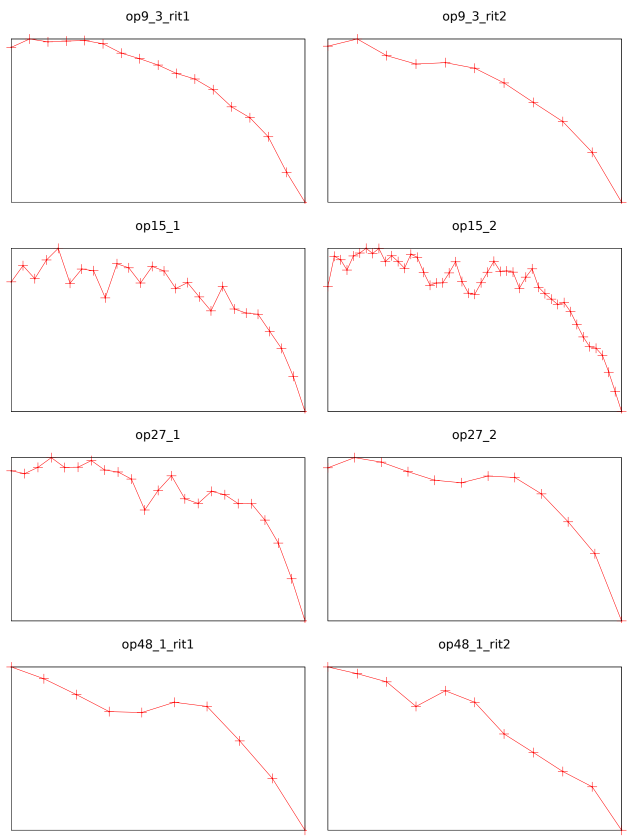


Figure 1. The average performance per ritardando. Both score time (horizontal axis) and tempo (vertical axis) are normalized

To this end, a subset of the data containing seven performances of various performers and different pieces has been annotated twice, by two different persons. [1] This set in total contains 304 time points to be measured. For each beat a pair of annotated beat times was available after annotation by both annotators, from which the absolute pairwise differences were calculated.

Figure 2 shows a scatter plot of absolute pairwise differences of measured IOI time versus the beat duration. [2] Note that beat durations have been calculated from note interonset times that were sometimes at a substantially faster pace than the beat. Hence, a beat duration of, say, 14 seconds does not imply that two measured points are actually 14 seconds apart. It can be observed from the plot that at slower tempos, there is more agreement between annotators about the onset times of notes. This is likely to be either because the slower parts tend to be played in a more articulate way, or simply because of the lower note density, which makes it easier to determine note onsets precisely.

The line in figure 2 shows the function that we use as a criterion to either accept or reject a particular IOI data point for further analysis. More specifically, the function specifies how far a data point must be away from the performance norm in order to be considered as a significant deviation. Conversely, we consider deviations of points closer to the norm too likely caused by measurement errors. The criterion is rather simple, and defines .2 seconds as an absolute minimum for deviations, with an increasing threshold for measurements at higher tempos (shorter beat durations), to accommodate for the increasing measurement differences observed in the data. The constants in the the function have been chosen manually, ensuring

[1] Because of the size of the data set, and the effort that manual correction implies, it was not feasible to annotate the complete data set multiple times

[2] by *beat* we mean score unit duration, rather than a perceived pulse

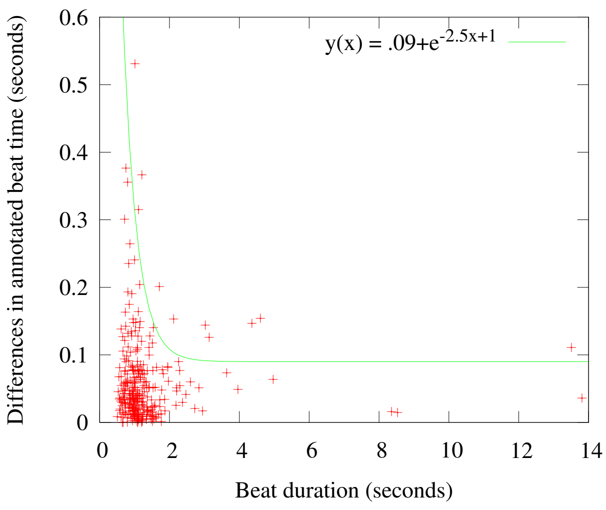


Figure 2. Scatter plot of absolute beat time annotation differences versus beat duration, between two annotators

that a substantial amount of the measurement deviations in the scatter plot ($> 95\%$) are excluded by the criterion. This approach can admittedly be improved. Ideally, taking into account the significance of deviations from the performance norm should be done by a weighting of data points that is inversely proportional to the likelihood of being due to measurement errors.

With the current criterion we filter the data by keeping only those data points that satisfy the inequality:

$$\mathbf{u}_{k,i}(j) > 0.09 + \exp\left[-2.5(\mathbf{a}_k(j) + \bar{\mathbf{v}}_{k,i}) + 1.0\right] \quad (3)$$

The set of data points after filtering is displayed for two pianists in figure 3. The left plot shows the significant deviations from the performance norm over all ritardandi performed by Falvai. The right plot shows those of Leonskaja. In order to compare the ritardandi from different pieces (with differing length and different number of measured IOI's), time has been normalized per piece. Note that a large part of Falvai's IOI deviations has been filtered out based on their size. This means that Falvai's ritardandi are are mostly in agreement with the performance norm. Interestingly, the endings of Falvai's ritardandi deviate in a very consistent way by being slightly faster than the norm until the last few notes, which tend to be delayed more than normal. Leonskaja's IOI deviations are more diverse and appear to be more piece dependent. A more in-depth investigation seems worthwhile here, but is beyond the scope of this article.

3.3 Evaluation of the data: automatic identification of pianists

In order to verify whether the residual timing data after subtracting the norm and filtering with the measurement error criterion in general carry information about the performing pianist, we have designed a small experiment. In this experiment we summarize the residual timing data by four attributes and apply a multilayer perceptron [18] (a standard machine learning algorithm, as available in the *Weka* toolbox for data mining and machine learning) to perform binary classification for all pairs of pianists in the data set. [3] The training instances (ritardandi of a particular piece performed by a particular pianist) containing varying numbers of IOI deviation values, each associated with a normalized score time value, describing where the IOI deviation occurs in the ritardando (0 denoting the beginning of the ritardando, and 1 the end). In order to use these data for automatic classification, they must be converted to data instances with a fixed number of attribute-value pairs. We choose an extremely simple approach, in which we represent a set of IOI deviation / score time pairs by the mean and standard deviation of the IOI values and the mean and standard deviations of the normalized time values. Thus, we effectively model the data by describing the size and location of the area where IOI deviation values tend to occur in the plots of figure 3.

4. RESULTS AND DISCUSSION

The pairwise pianist classification task is executed as follows: for each possible pair of pianists, the ritardandi of both pianists are pooled to form the data set for evaluating the classifier. The training set in most cases contains 16 instances, one for each of the eight pieces, for each of the two pianists. The pianists from whom less than 6 performances were contained in the data set were not included in the test. The data set was used to evaluate the multilayer perceptron using 10-fold cross-validation. This was done for all 171 combinations of 19 pianists. The results are compared to a baseline algorithm that predicts the mode of the target concept, the pianist, in the training data.

The classification results on the test data are summarized in tables 2 and 3. Table 2 shows the proportion of pairwise identification tasks where the multilayer perceptron classified above, at, and, below baseline classification, respectively. The top row presents the results for the condition where the IOI deviation data has been filtered using the measurement error criterion, as explained in subsection 3.2. The bottom row correspond to the condition where no such filtering was applied.

The measurement error filtering clearly leads to an improvement of classification accuracy. With filtering, the percentage of pianist identification tasks that are executed with an accuracy that is significantly ($\alpha = .05$) above baseline accuracy, is 32%. Although this percentage does not seem very high, it must be considered that the amount of information available to the classifier is very small. Firstly, the ritardandi are only short fragments of the complete performances. Secondly, the training sets within a 10-fold cross-validation never contain more than seven ritardandi of a single pianist. Lastly, the IOI deviation information available has been summarized very coarsely, by a mean and standard deviation of the values in the time and IOI dimension. This result implies that larger deviations from

[3] For some pianists less than six performances were available; Those pianists have not been included in the experiment.

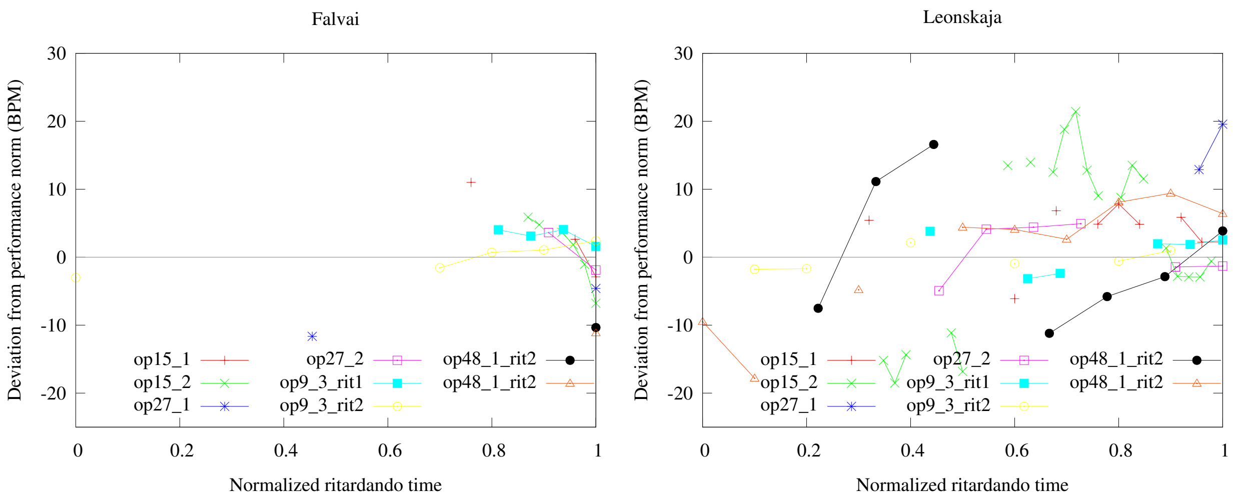


Figure 3. Deviations from the performance norm after applying the measurement error criterion; Left: Falvai; Right: Leonskaja

the performance norm by individual pianists are at least to some degree pianist specific, and not just piece specific.

We wish to emphasize that by no means we claim that the specific form of the measurement error criterion we proposed in subsection 3.2 is crucial for the success of of pianist identification. Other filtering criteria might work equally well or better. Note however that there is a trade off between avoiding the disturbing effect of measurement errors on the one hand, and a reduction of available data on the other. A more elegant approach to canceling the effect of measurement errors would be to use a weighting criterion rather than a filtering criterion.

Without filtering, accuracy is even significantly *below* the baseline in 19% of the cases. The fact that under this condition accuracy does not often surpass the baseline is not surprising, since the unfiltered data contains all available IOI deviation values, equally distributed over time. A consequence of this that mean and standard deviation of the normalized times associated to the IOI data are constant. This reduces the available information so much that it is unrealistic to expect above baseline accuracy. That the prediction accuracy is significantly below baseline is more surprising. Given that the performance norm is subtracted from the original timing data per piece, a strong interference of the piece with the pianist identification is not to be expected. A possible explanation for this result could be that there are multiple distinct performance strategies. Obviously, the average performance as a performance norm is not adequate for this situation, where multiple performance norms are present. If two pianists choose a similar strategy, their residual IOI values after subtracting the average performance may still be more similar to each other than to their own IOI values in a different piece.

Table 3 shows the average identification accuracy over all identification-tasks that involve a specific pianist. High accuracy could indicate that a pianist plays both consistently, and distinctively. By playing consistently we mean that particular IOI deviations tend to occur at the similar positions in the ritardando, as observed in the case of Falvai, in figure 3 (see also [19] for a discussion of performer consistency). Playing distinctively means that no other pianist has similar IOI deviations at similar positions. Conversely, a low identification accuracy could point to either a varied way of performing ritardandi of different pieces, or playing ritardandi in particular pieces in a way that is similar to the way (some) other pianists play them, or both.

Procedure	< baseline	baseline	> baseline
with filtering	0 (0%)	116 (68%)	55 (32%)
without filtering	33 (19%)	131 (76%)	7 (4%)

Table 2. Number of 10-fold cross-validated pairwise pianist classification tasks with results over, at, and below baseline results, respectively ($\alpha = .05$)

5. CONCLUSIONS AND FUTURE WORK

Ritardandi in musical performances are good examples of the expressive interpretation of the score by the pianist. We have investigated the possibility of automatically identifying pianists by the way they perform ritardandi. More specifically, we have reported an initial experiment in which we use IOI deviations from a performance norm (the average performance) to distinguish pairs of pianists. Furthermore, we have introduced a simple filtering criterion that is intended to remove parts of the data that are likely to be due to measurement errors. Although more sophisticated methods for dealing with measurement error can certainly be developed, the filtering method improved the accuracy of pianist identification substantially.

Continued work should include the development of a more gradual way to deal with the significance of IOI deviations, rather than an all-or-nothing filtering method. Also, better models of expressive timing and tempo are needed to serve as a performance norm. In this work we have employed the average performance as a substitute norm, but it

Pianist	avg. % correct
Leonskaja	65.31
Pollini	64.83
Vasary	63.50
Ohlsson	62.28
Mertanen	62.06
Barenboim	61.69
Falvai	57.42
Engerer	54.33
Hewitt	53.50
Woodward	53.47
Biret	51.47
Pires	51.03
Tsong	50.17
Harasiewicz	49.78
Kollar	49.33
d´Ascoli	48.06
Ashkenazy	47.69
Rubinstein	45.83
Arrau	43.53

Table 3. Average identification accuracy per pianist on test data

is obvious that a norm should be independent of the data.

6. ACKNOWLEDGMENTS

We wish to thank Werner Goebl and Bernhard Niedermayer for their help in the acquisition of the timing data from the audio recordings. This work is funded by the Austrian National Research Fund (FWF) under project number P19349-N15.

7. REFERENCES

[1] E. F. Clarke. Generative principles in music. In J.A. Sloboda, editor, *Generative Processes in Music: The Psychology of Performance, Improvisation, and Composition*. Oxford University Press, 1988.

[2] P. Juslin and J. Sloboda, editors. *Music and Emotion: Theory and Research*. Oxford University Press, 2001.

[3] C. Palmer. Music performance. *Annual Review of Psychology*, 48:115–138, 1997.

[4] B. H. Repp. Patterns of expressive timing in performances of a Beethoven minuet by nineteen famous pianists. *Journal of the Acoustical Society of America*, 88:622–641, 1990.

[5] B. H. Repp. Diversity and commonality in music performance - An analysis of timing microstructure in Schumann's "Träumerei". *Journal of the Acoustical Society of America*, 92(5):2546–2568, 1992.

[6] E. Stamatatos and G. Widmer. Automatic identification of music performers with learning ensembles. *Artificial Intelligence*, 165(1):37–56, 2005.

[7] C. Saunders, D. Hardoon, J. Shawe-Taylor, and G. Widmer. Using string kernels to identify famous performers from their playing style. *Intelligent Data Analysis*, 12(4):425–450, 2008.

[8] W. L. Windsor and E. F. Clarke. Expressive timing and dynamics in real and artificial musical performances: using an algorithm as an analytical tool. *Music Perception*, 15(2):127–152, 1997.

[9] N.P. Todd. A computational model of rubato. *Contemporary Music Review*, 3 (1), 1989.

[10] A. Friberg. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15 (2):56–71, 1991.

[11] R. Timmers, R.and Ashley, P. Desain, H. Honing, and L. Windsor. Timing of ornaments in the theme of Beethoven's Paisiello Variations: Empirical data and a model. *Music Perception*, 20(1):3–33, 2002.

[12] J. Sundberg and V. Verrillo. On the anatomy of the retard: A study of timing in music. *Journal of the Acoustical Society of America*, 68(3):772–779, 1980.

[13] A. Friberg and J. Sundberg. Does music performance allude to locomotion? a model of final ritardandi derived from measurements of stopping runners. *Journal of the Acoustical Society of America*, 105(3):1469–1484, 1999.

[14] M. Grachten and G. Widmer. The kinematic rubato model as a means of studying final ritards across pieces and pianists. In *Proceedings of the 6th Sound and Music Computing Conference*, 2009.

[15] H. Honing. Is there a perception-based alternative to kinematic models of tempo rubato? *Music Perception*, 23(1):79–85, 2005.

[16] B. Niedermayer. Non-negative matrix division for the automatic transcription of polyphonic music. In *Proceedings of the 9th International Conference on Music Information Retrieval*. ISMIR, 2008.

[17] C.L. Cannam, M. Sandler, and J.P. Bello. The sonic visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proceedings of the 7th International Conference on Music Information Retrieval*. ISMIR, 2006.

[18] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing*, volume 1. MIT Press, 1986.

[19] S. T. Madsen and G. Widmer. Exploring pianist performance styles with evolutionary string matching. *International Journal on Artificial Intelligence Tools*, 15(4):495–513, 2006. Special Issue on Artificial Intelligence in Music and Art.

AN ANALYSIS OF ISMIR PROCEEDINGS: PATTERNS OF AUTHORSHIP, TOPIC, AND CITATION

Jin Ha Lee
University of Washington
The Information School
jinhalee@uw.edu

M. Cameron Jones
University of Illinois at Urbana-Champaign
mjones2@illinois.edu

J. Stephen Downie
University of Illinois at Urbana-Champaign
jdownie@illinois.edu

ABSTRACT

This paper presents analyses of peer-reviewed papers and posters published in the past nine years of ISMIR proceedings: examining publication and authorship practices, topics and titles of research, as well as the citation patterns among the ISMIR proceedings. The main objective is to provide an overview of the progress made over the past nine years in the ISMIR community and to obtain some insights into where the community should be heading in the coming years. Overall, the ISMIR community has grown considerably over the past nine years, both in the number of papers and posters published each year, as well as the number of authors contributing. Furthermore, the amount of collaboration among authors, as reflected in co-authorship, has increased. Main areas of research are revealed by an analysis of most commonly used title terms. Also, major authors and research groups are identified by analyzing the co-authorship and citation patterns in ISMIR proceedings.

1. INTRODUCTION

This year, 2009, marks the tenth iteration of the International Symposium on Music Information Retrieval conference series (ISMIR). ISMIR was organized with the hope that the "resulting information interchange will enable scholars to move more quickly towards viable solutions to many problems" [1] in the field of Music Information Retrieval (MIR).

Futrelle & Downie [2] defined MIR as "a rapidly growing interdisciplinary research area encompassing computer science and information retrieval, musicology and music theory, audio engineering and digital signal processing, cognitive science, library science, publishing, and law. Its agenda, roughly, is to develop ways of managing collections of musical material for preservation, access, research, and other uses". Necessarily, MIR spans both audio and symbolic representations of music [3], but also includes musical metadata, usage data, and other extra-musical information [4], including user-studies and human-computer interaction studies of music systems. To date, most research in MIR has been content-based [5].

In 2000, MIR was still a fairly new field with a great deal of potential that was gaining the interest of researchers from many different domains. Although ISMIR started as a small-scale symposium, it has grown immensely over the past nine years as more people have recognized the importance of MIR research and have been drawn in to the field. The community has grown to the point of establishing the International Society for Music Information Retrieval, which will help orient, organize, and disseminate the community's future research.

We performed various informetric analyses on the ISMIR proceedings from 2000 to 2008 in order to discover how the patterns of publications have changed over the past nine years. Through these analyses, we hope to obtain insights into what the ISMIR community has and has not been able to accomplish and which directions it could be heading towards in the coming years.

In the following, we provide descriptive statistics showing the change in the number of publications and authorship patterns over the past nine years. We also provide the results of our analysis of the title terms, looking at the most commonly used single terms as well as bigrams. In addition, we performed analyses on the citation patterns among the publications and authors who have published in the ISMIR proceedings.

2. GROWTH OF THE ISMIR COMMUNITY

The first ISMIR conference had just 10 refereed papers and 16 posters representing 55 authors, with several other scholars presenting invited talks. To date, 881 authors have contributed peer-reviewed papers and posters to the ISMIR proceedings, not to mention the numerous participants in the annual Music Information Retrieval Evaluation eXchange (MIREX), conference workshops, demonstrations, tutorials, and invited talks. The rapid growth in participation has been paralleled by a similar increase in the number of papers and posters accepted to ISMIR. In total, over 700 peer-reviewed papers and posters have been published, comprising a substantial literature on a breadth of topics ranging from signal-processing techniques to user studies of MIR systems.

2.1 Change in the Number of Publications per Year

The number of publications in ISMIR proceedings has been steadily increasing over the past nine years. Exact numbers are presented in Table 1 along with the number of unique authors published in each year.

Year	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**
Papers	10	21	31	23	60	59	59	62	105
Posters	16	16	22	24	44	57	28	65	-
Total	**26**	**37**	**53**	**47**	**104**	**116**	**87**	**127**	**105**
Unique Authors	**55**	**74**	**113**	**108**	**213**	**232**	**185**	**267**	**262**

Table 1. The number of publications and unique authors per year.

We can better observe the changes in the proportion of papers and posters for each year, as well as the changes in the number of authors. The number of publications almost doubled in 2004, jumping from 47 in 2003 to 104. In 2008, there was a change in the submission format and all paper submissions were to have accompanying posters as well. Looking at the number of authors, we can see that there were two sharp increases in 2002 and 2004, and a major drop in 2006. However, the overall number of authors represented at the conference each year has generally grown over the past nine years.

Figure 1 shows the authorship trends within the ISMIR proceedings, tracking the proportion of papers with one, two, three, four, and five-or-more authors. As can be seen, the number of single-authored papers has decreased year-over-year. The number of papers with two co-authors peaked in 2002, and has steadily declined since. However, the number of papers with three authors has steadily increased year-over-year. The average number of co-authors on papers and posters published each year has increased over the past nine years, from an average of 2.27 authors per publication in 2000 to 2.93 authors per publication in 2008. Clearly ISMIR is becoming a much more collaborative community as the number of authors per paper increases, and the proportion of single, and double-authored papers diminishes in favor of papers with three or more authors.

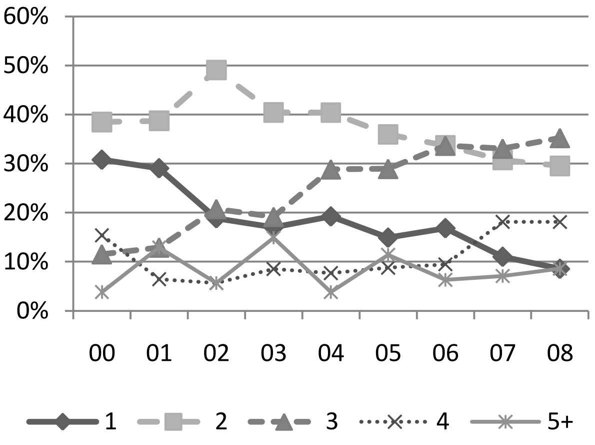


Figure 1. Co-authorship trends tracking the percentage of papers with 1, 2, 3, 4, and 5+ authors from 2000 to 2008.

2.2 Co-authorship Analysis

We performed an analysis to identify the patterns of co-authorship among all the authors who published in ISMIR proceedings and determine which authors appear as the central hubs in the co-authorship graph. Figure 2

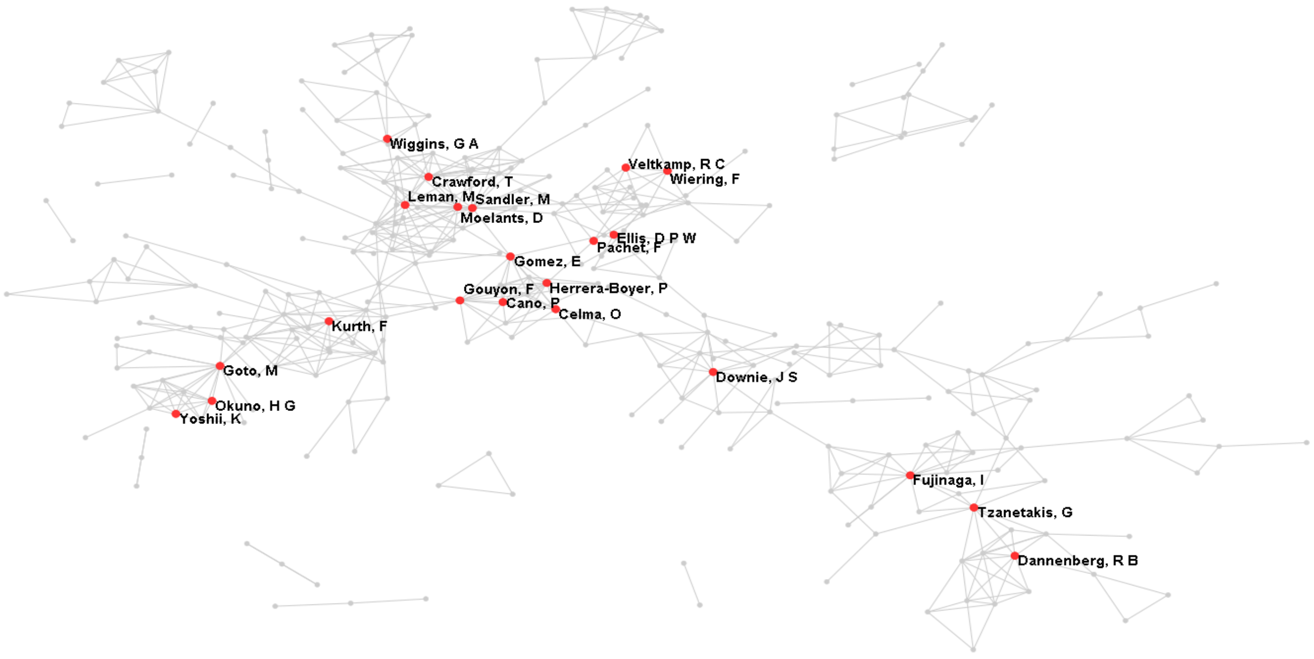


Figure 2. Co-authorship network among ISMIR authors who have published two or more articles. The 22 authors with the largest co-authorship networks have been highlighted.

was generated by using Pajek which is a social networking analysis and visualization tool [6]. In this figure, only the authors who published 2 or more papers/posters are included in order to simplify the network.

Several main clusters of people can be visually identified in the figure showing the close connections among some authors. The top 22 authors with the largest number of distinct co-authors (12+) are labeled in the figure. Often these authors represent an active research group such as National Institute of Advanced Industrial Science and Technology (AIST) headed by M. Goto in Japan, The International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) headed by J. S. Downie in Illinois, Distributed Digital Music Archives and Libraries headed by I. Fujinaga in Canada, the Center for Digital Music headed by Mark Sandler in London, and so on.

What is evident from these analyses is the growing role of research labs in the ISMIR community, and how they engender collaboration and increase participation in research. Many European labs and research groups are tightly interconnected, and are difficult to distinguish one from the other based on the co-authorship patterns. Furthermore, not all evidence of collaboration is represented in the co-authorship network; for example, the IMIRSEL lab appears relatively isolated, despite their central role in organizing MIREX. Large, intercontinental, multi-institutional grant projects, such as the Networked Environment for Music Analysis (NEMA) project [7], may start to change the shape of collaboration within the ISMIR community.

3. RESEARCH TOPICS IN ISMIR

The topics explored in the first ISMIR conference laid the foundation for the future growth and evolution of the field. While ISMIR has grown, it has remained true to the original vision laid out in the early conference programs. In this section, we present an analysis of terms extracted from the titles and abstracts of ISMIR papers. Only title and abstract terms were used as these represent concise summaries of the papers' content.

3.1 The Most Commonly Used Title Terms

In order to get an idea as to which research areas have been of interest over the past nine years, we analyzed the title terms of all peer-reviewed papers and posters in the ISMIR proceedings. All the terms from the titles of the papers and posters were extracted. The words were stemmed using a Perl-based implementation of the Porter stemming algorithm [8], and stop-words were removed using a combination of a standard list of common-usage English-language words, with the stop-word "music", as this term appears in almost all titles in the ISMIR proceedings. Table 2 shows the top terms that appeared in the publication titles for each year. New terms entering the top-ranked lists are highlighted in bold-face.

From the table, we can observe that the most often used title terms were relatively similar for each year; however, it is possible to identify certain trends. For instance, there was a strong interest in query by singing/humming systems in 2002 and 2003 shown by the title term query ("queri") appearing only in the lists of these two years. Research interest in musical genres increased in 2005 and 2006, and interest in music similarity research peaked in 2006. Interest in classification and modeling has been consistent over the past nine years. Additionally, the consistently high rank of the term "audio" suggests that ISMIR researchers have been focused primarily on audio rather than symbolic representations.

What is also evident from the title terms, is how closely the field has stuck to the original framing of MIR as represented in the 2000 ISMIR program. The core concepts have remained prevalent throughout the past dec-

2000	2001	2002	2003	2004	2005	2006	2007	2008
Retriev	Retriev	Retriev	Retriev	Audio	Audio	Audio	Audio	Audio
Inform	Inform	Audio	Automat	Retriev	Retriev	Similar	Retriev	Featur
Model	System	Inform	Model	Automat	Classif	Classif	Similar	Retriev
System	Audio	**Queri**	Similar	System	Featur	Model	Model	Model
Audio	**Approach**	System	**Database**	Classif	Inform	Genr	System	Analysi
Classif	Model	**Automat**	Audio	Polyphon	Model	Automat	**Recognit**	Automat
Polyphon	**Analysi**	Model	Inform	**Pattern**	Polyphon	Feature	Polyphon	**Song**
Segment	**Similar**	Polyphon	Queri	Inform	Extract	Approach	Featur	Inform
Instrument	**Match**	Similar	System	**Extract**	Similar	**Perform**	Analysi	Similar
Techniqu	**MIR**	Analysi	Classif	**Featur**	**Algorithm**	Retriev	Classif	**Chord**
Languag	**Spot**	**Content**		**Sound**	**Genr**	**Evalu**	Automat	Content
		Pattern		**Tempo**		**Key**	Approach	
		Voic					Evalu	
							Transcript	
							Algorithm	

Table 2. Top 10 ranked title terms of each year (w/ ties); new terms are highlighted in bold-face font.

ade, yet have accommodated expansion into new areas.

3.2 Title and Abstract Bi-grams

Single-term-concepts present a limited view of research concepts and topics, especially after subtle differences in terms are merged by stemming (e.g., 'using' and 'users' have the same stem, 'us', yet carry different connotations in usage). Furthermore, the limited text available in titles, only provides a glimpse at the complexity of concepts and ideas being researched and published. In order to get at more specific concepts which have taken the interest of ISMIR researchers, we extracted stemmed bi-grams (i.e., 2-word phrases) from the titles and abstracts of all papers and posters. Initially, we examined the bi-grams on a year-by-year basis, much as we did for single term concepts. However, as expected, the number of bi-grams exceeds the number of uni-grams, and the frequency with which any one bi-gram occurs is much lower. No meaningful or interesting patterns arose in the yearly analysis; however, when considered in aggregate, there is stronger evidence of dominant research topics within the field. Table 3 shows the top 20 most commonly used bi-grams in ISMIR proceedings over the last nine years.

Bi-gram (stemmed)	Count
inform_retriev	25
content_base	24
genr_classif	14
web_base	9
hidden_markov	9
queri_hum	9
polyphon_audio	8
real_time	7
system_base	7
optic_recognit	7
audio_featur	7
ground_truth	7
base_similar	6
featur_extract	6
playlist_gener	6
audio_fingerprint	6
sing_voic	6
retriev_system	6
automat_transcript	5
melod_similar	5
similar_measur	5
automat_genr	5

Table 3. Top 20 most commonly used bi-grams from titles and abstracts, reflecting the main research foci, methods, and approaches of the ISMIR community.

The most common bi-gram is "information retrieval", followed by "content based", "genre classification", and so on. Beyond these, we can see the prevalence of the web, and web-based systems, which has paralleled the emergence of "web 2.0" and greater access to music and music systems online within the commercial sector. Although the frequencies of occurrence of some individual concepts are low, overall we find the topics represented by the bi-gram analysis to be fairly representative of the major research interests in the field: such as "music similarity", "feature extraction", and so on.

4. CITATION PATTERNS

Moving beyond terms and bi-grams as representations of research interests, the papers themselves published in the ISMIR proceedings serve as representations of research topics and areas, and references to them serve as a way of highlighting the prevailing research interests of the community. Weinstock [9] outlines 15 motivations for why academics cite each other in scholarly writing including paying homage to pioneers, giving credit for related work, and so on. We examined the references lists of all peer-reviewed ISMIR papers and posters, and looked for references to other peer-reviewed ISMIR papers and posters. We did not consider references to demos, invited talks, tutorials, MIREX abstracts, or workshop papers. We also did not attempt to measure references to publications outside the ISMIR proceedings, nor did we attempt to gauge the number of citations to ISMIR papers from outside.

First, we shall outline and describe the general citation behavior of the ISMIR community. Figure 3 shows the frequency distribution of publications by the number of references to other ISMIR publications they contain. Most ISMIR papers and posters (nearly 50%) do not reference any other ISMIR publications. The average number of ISMIR references per paper/poster was 1.278 with the standard deviation of 2.05 and the maximum of 27.

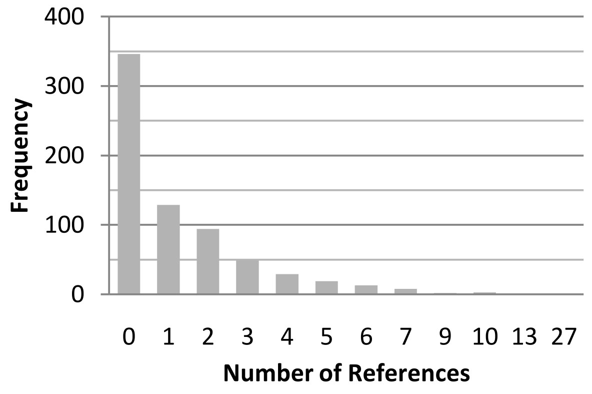


Figure 3. Number of ISMIR references in ISMIR papers

The reasons for low internal referencing within the ISMIR community may be due to the fact that some authors preferentially cite journals, books, and theses which are extensions of, or refinements of ideas initially published in ISMIR over the ISMIR publications. Other possible explanations include the fact that ISMIR proceedings are not indexed in commonly used digital library portals, such as the ACM Digital Library or CiteSeer, and are inconsistently indexed by Google Scholar. The fantastic resource on http://www.ismir.net/, which has been developed and maintained by Michael Fingerhut, contains a near-complete set of the full-text versions of all papers and posters published in the ISMIR proceedings; however, it lacks full-text search capabilities itself, and the site does not provide complete, standardized metadata records which may improve the visibility of ISMIR papers in search engines, and digital library portals.

Author/Title	# Refs
Goto, M., et al. (2002). *RWC Music Database: Popular, Classical and Jazz Music Databases*	21
Bello, J. & Pickens, J. (2005). *A Robust Mid-Level Representation for Harmonic Content in Music Signals*	13
Tzanetakis, G., Essl, G. & Cook, P. (2001). *Automatic Musical Genre Classification of Audio Signals*	13
Aucouturier, J. & Pachet, F. (2002). *Music Similarity Measures: What's the use?*	13
Sheh, A. & Ellis, D. (2003). *Chord segmentation and recognition using EM-trained hidden markov models*	12
Pampalk, E., Dixon, S. & Widmer, G. (2003). *Exploring music collections by browsing different views*	11
Paulus, J. & Kalpuri, A. (2002). *Measuring the similarity of Rhythmic Patterns*	11
Goto, M., et al. (2003). *RWC Music Database: Music genre database and musical instrument sound database*	10
Clausen, M., et al. (2000). *PROMS: A Web-based Tool for Searching in Polyphonic Music*	9
Ellis, D., et al. (2002). *The Quest for Ground Truth in Musical Artist Similarity*	8
Logan, B. (2000). *Mel Frequency Cepstral Coefficients for Music Modeling*	8
Birmingham, W., et al. (2001). *MUSART: Music Retrieval Via Aural Queries*	8
Logan, B. (2004). *Music Recommendation from Song Sets*	8
Abdallah, S. & Plumbley, M. (2004). *Polyphonic transcription by non-negative sparse coding of power spectra*	7
Foote, J., Cooper, M. & Nam, U. (2002). *Audio Retrieval by Rhythmic Similarity*	7
Mazzoni, D. & Dannenberg, R. (2001). *Melody Matching Directly from Audio*	7
Vinet, H., Herrera-Boyer, P. & Pachet, F. (2002). *The CUIDADO Project*	7
Soulez, F., Rodet, X. & Scharwz, D. (2003). *Improving polyphonic and poly-instrumental music to score alignment*	7
Whitman, B. & Ellis, D. (2004). *Automatic Record Reviews*	7
Whitman, B. & Smaragdis, P. (2002). *Combining Musical and Cultural Features for Intelligent Style Detection*	7

Table 4. Top cited papers and posters (excluding self-citation).

Working with the references we were able to extract, we filtered self-citations, which we defined as a reference to a paper in which an author of the citing paper is an author on the referenced paper. Table 4 shows the top cited papers and posters in the ISMIR proceedings, ranked by the number of references we were able to find to each.

Among the top cited papers and posters, there is a diversity of topics and publications, from which we may infer a range of motivations. The most cited publication in the ISMIR proceedings is Goto, et al.'s 2002 poster introducing the RWC database, garnering 21 references. Following Weinstock's taxonomy of citer motivation, the referencing of a data set is most like motivation three: identifying methodology, equipment, etc. The lack of standardized data sets with ground truth is a recurring problem in the MIR community and the RWC database has served as a valuable resource for MIR researchers, as it acts as a de facto standardized collection on which to build and evaluate systems. In fact, the presence of Goto, et al., 2003, and Ellis, et al., 2002 on this list reiterate the importance of standardized data sets with ground truth within MIR research.

There are several other methodological references, including references to Logan (2000), Tzanetakis, et al. (2001), Sheh & Ellis (2003), Goto, et al. (2003). There are also elements of "paying homage" in the references to several papers, especially the seminal work of Beth Logan, who introduced MFCCs to the MIR community.

Author	Ref. Count	Co-author Count	Paper/Poster Count
Goto, M	43	24	21
Ellis, D P W	41	12	12
Hashiguchi, H	34	5	3
Nishimura, T	34	5	3
Oka, R	34	5	3
Widmer, G	34	11	19
Dannenberg, R B	29	15	10
Logan, B	29	4	5
Whitman, B	28	6	5
Downie, J S	26	15	25
Pampalk, E	26	11	12
Tzanetakis, G	24	27	15
Birmingham, W P	23	11	7
Pachet, F	22	12	13
Dixon, S	22	9	9
Meek, C	22	10	5
Pickens, J	21	7	6
Pauws, S	19	6	8
Cook, P	19	7	6
Fujinaga, I	19	31	28

Table 5. Top 20 cited authors (excluding self references).

Without a more in-depth analysis of the individual contexts surrounding each citation, it is difficult to tease out the precise motivations for all the references. Regardless, the most referenced works comprise a diversity of topics and areas which span the breadth of research within MIR, including references to signal-processing algorithms and methods as well as techniques for handling symbolic representations of music. There are papers covering music transcription, and rhythm analysis, as well as high-level tasks such as genre-classification, search and recommendation algorithms, and approaches to understanding audio similarity.

Table 5 shows the top 20 cited authors excluding self references. The second column shows the count of coauthors each of these authors have in ISMIR proceedings and the third column shows the count of papers/posters each author published. The most heavily cited author was

Masataka Goto with 43 references by other ISMIR authors. Among these top-cited authors, we can see there are those who have many references, in part because they have published many papers (e.g., Goto; Widmer), and there are authors who are highly cited, but have only a few publications (e.g., Logan; Whitman). There is, however, no correlation (*r=0.021*) between reference count and paper count, indicating that the referencing of authors is not merely a product of their productivity within the community. It is worth noting that among the top-cited authors, there is a strong correlation between the number of co-authors an author has, and the number of papers he/she has written (*r=0.815*). This correlation is not that surprising given our findings from section 2 where we discussed the trend towards collaboration and co-authorship among ISMIR authors.

5. CONCLUSION

The ISMIR community has grown significantly, and through the contributions of nearly 900 researchers, the field of Music Information Retrieval has been well-defined and established. The community is a tightly-knit one, with a high-degree of collaboration and co-authorship, focused around a core set of research topics and areas.

The main insights of our analyses can be summarized as follows:

1) The ISMIR community is becoming more collaborative as shown by increasing co-authorship;
2) The role of research labs is growing in the ISMIR community as they promote collaboration and increased participation in research;
3) The focus of research has mainly been on audio so far as revealed by the most commonly used title and abstract terms;
4) The most cited works in the ISMIR proceedings comprise a variety of topics, but primarily point to datasets, techniques, and methods;

In their early ISMIR paper discussing the interdisciplinary communities and research issues, Futrelle and Downie [2] lists several key research areas in MIR. Among these, our analyses show that areas such as feature detection and classification/machine learning have been the major topics represented to date in the ISMIR proceedings, whereas topics such as user studies, metadata, work on symbolic representations, and epistemology/ontology have not been as well represented as others. Our advice for the sustained, future growth of the ISMIR community is to encourage greater activity in these areas, as they are relatively uncrowded, open topics of research in which great advances can be made.

We would like to continue our informetric analysis of MIR research, and there are several aspects that can be further analyzed to obtain a broader picture of MIR. One area in which we could improve our understanding of the domain, is to include external sources and references in our citation analysis, and track the number of ISMIR references found in other related journals and proceedings, references that are not from ISMIR proceedings and so on. Additionally, we explored several clustering analyses in researching this paper, and none provided immediately compelling results. We would like to continue to explore how papers, authors, and research topics cluster based on semantic similarity, co-authorship patterns, citation patterns, and bibliographic coupling.

6. ACKNOWLEDGEMENTS

We would like to acknowledge the work of Michael Fingerhut in maintaining the ISMIR proceedings database (http://ismir.net/), which was our primary resource for locating all of the publications, and was vital to verifying our data integrity; we are very thankful for his diligence. We also thank the Andrew W. Mellon Foundation for its financial support.

7. REFERENCES

[1] ISMIR 2000: International Symposium on Music Information Retrieval. Available at: http://ciir.cs.umass.edu/music2000/

[2] J. Futrelle, J. S. Downie: "Interdisciplinary Communities and Research Issues in Music Information Retrieval," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 215-221, 2002.

[3] D. Byrd, T. Crawford: "Problems of Music Information Retrieval in the Real World," *Information Processing and Management* 38, pp. 249-272, 2001.

[4] J. H. Lee: *Analysis of Information Features in Natural Language Queries for Music Information Retrieval: Use Patterns and Accuracy*. University of Illinois, Ph.D. thesis, 2008.

[5] N. Orio: "Music Retrieval," *Foundations and Trends in Information Retrieval* 1(1), pp. 1-90, 2006.

[6] V. Batagelj, A. Mrvar: *Pajek - Analysis and Visualization of Large Networks* in Jünger, M., Mutzel, P., (Eds.) Graph Drawing Software. Springer, Berlin, pp. 77-103, 2003.

[7] Networked Environment for Music Analysis (NEMA). Available at: http://nema.lis.uiuc.edu/

[8] M. Porter: An algorithm for suffix stripping in Sparck-Jones, K.; Willett, P. (Eds.): Readings in Information Retrieval. Morgan Kaufmann Publishers, pp. 313-316, 1997.

[9] M. Weinstock: "Citation Indexes," *Encyclopedia of Library and Information Science* 5. Marcel Dekker, New York, 1971.

THE ISMIR CLOUD: A DECADE OF ISMIR CONFERENCES AT YOUR FINGERTIPS

Maarten Grachten Markus Schedl Tim Pohle Gerhard Widmer
Department of Computational Perception
Johannes Kepler University
Linz, Austria

music@jku.at
http://www.cp.jku.at

ABSTRACT

In this paper, we analyze the proceedings of the past *International Symposia on Music Information Retrieval* (ISMIR). We extract meaningful term sets from the accepted submissions and apply term weighting and Web-based filtering techniques to distill information about the topics covered by the papers. This enables us to visualize and interpret the change of hot ISMIR topics in the course of time. Furthermore, the performed analysis allows for assessing the cumulative ISMIR proceedings by semantic content (rather than by literal text search). To illustrate this, we introduce two prototype applications that are publicly accessible online [1]. The first allows the user to search for ISMIR publications by selecting subsets of ISMIR topics. The second provides interactive visual access to the joint content of ISMIR publications in the form of a *tag cloud* – the *ISMIR Cloud*.

1. INTRODUCTION AND MOTIVATION

Music information retrieval and extraction has been a fast growing field of research during the past decade. Certainly the most important forum for this multidisciplinary field is the *International Symposium on Music Information Retrieval* (ISMIR) [11]. In 2009, ISMIR celebrates its 10th anniversary. Thus, we think it is time to look back and investigate which general topics and research problems were most important in MIR during the past decade. To this end, we analyzed the digital ISMIR proceedings [11] available online. Not only have we captured the principal topics reflected by previous, accepted ISMIR papers by means of text-based content extraction and analysis, but we have also investigated how these topics changed over time. Since MIR is a highly dynamic field of research, we gained interesting insights, which will be detailed in the following. We further visualized the corpus of ISMIR documents via clusters of topics described by sets of terms. To this end, we employed two approaches based on *Non-Negative Matrix Factorization* (NMF) and *Principal Components Analysis* (PCA). Two prototype applications are provided as a proof-of-concept. The first is a Web application for browsing the cumulative online ISMIR proceedings theme-wise. The second is an offline OpenGL application that visualizes the *ISMIR Cloud* in three dimensions, and allows for real-time interaction such as spatial navigation and text-based search for tags.

The remainder of the paper is organized as follows. Section 2. gives an overview of related work on text information extraction and retrieval, topic-based clustering, and visualization. Section 3. describes the features we extracted from the ISMIR corpus. In Section 4., we present our approaches to visualize and browse the papers. Finally, Section 5. draws conclusions and points out directions for future work.

2. RELATED WORK

Related work mainly falls into the two fields of *text mining*, more precisely, text-based information extraction/retrieval and *clustering and visualizing high-dimensional data*. In line with the dedication of this paper to the MIR community, we will focus on work carried out in the context of music information research.

In the context of MIR, extracting terms from texts, more precisely, from Web documents, in order to tag a music artist has first been addressed in [28], where Whitman and Lawrence extract different term sets (e.g., noun phrases and adjectives) from artist-related Web pages. Based on term occurrences, individual term profiles are created for each artist. The authors then use the overlap between the term profiles of two artists as an estimate for their similarity. A quite similar approach is presented in [13]. Knees et al. however do not use specific term sets, but create a term list directly from the retrieved Web pages. Subsequently, a term selection technique is applied to filter out less important terms. Hereafter, the TF·IDF measure, e.g., [31], is used to weight the remaining words and subsequently

[1] http://www.cp.jku.at/projects/ISMIR-cloud/

create a weighted term profile for each artist. Knees et al. propose their approach for artist-to-genre classification and similarity measurement.

A text-based music retrieval system that builds upon methods for term extraction from Web pages, term weighting, audio feature extraction, and similarity measurement is presented in [15]. In this paper, the authors relate audio features extracted from a given music collection with terms extracted from Web pages that contain part of the metadata present in the music collection. These terms are then weighted using an adapted version of the TF·IDF measure and joined with the audio features to build a feature vector for each track, which serves as a track descriptor. This approach allows for searching music collections via descriptive natural language terms, e.g., by issuing queries like "guitar riff" or "metal band with front woman". Other work related to MIR that makes use of text mining techniques includes [10], where a POS tagger is used to search *last.fm* [17] tags for adjectives that describe the mood of a song. In [3] the machine learning algorithm *AdaBoost* is used to learn relations between acoustic features and *last.fm* tags.

As for general work on text-based information extraction and retrieval, different methods for term selection and term weighting have been analyzed with respect to their performance in text categorization, cf. [2, 16, 30], in text-based retrieval, cf. [24], and in clustering, cf. [6]. A comprehensive evaluation of term weighting techniques and similarity measures for information retrieval purposes is presented in [31]. In their extensive evaluation of various formulations of TF, IDF, and similarity measures, Zobel and Moffat conclude that no single combination outperforms the others consistently. In fact, the performance of any combination was found to be highly dependent on the domain and query set it had been applied to. Text-based IE from the Web usually relies on identifying or learning specific patterns that contain the information to be extracted. Already in [7], the use of static rules to determine hyponyms in text corpora was proposed. [4] presents a system that complements generic text patterns with domain-specific rules found by pattern extraction via search engines and subsequent selection of high-quality extraction rules. [1] proposes an approach that solely relies on Google's page counts for specific patterns to determine instances of a given concept.

As for clustering and visualizing high-dimensional feature data in the context of MIR, in [21] *Non-Negative Matrix Factorization* (NMF) [18] was employed to determine clusters of concepts based on tags describing music artists, which were extracted from *last.fm*. Using NMF on features gained from a term weighting approach in order to cluster documents was already proposed in [29].

Another data projection and visualization technique is *Principal Components Analysis* (PCA) [8, 12]. PCA consists in a a linear projection of high-dimensional data onto a small set of orthogonal dimensions with minimal loss in variance. The relative distances between data points in the high-dimensional space are preserved as good as possible in the low-dimensional projection. Reducing the dimensionality of the feature space to two or three thus allows for the visualization of possible low-dimensional structure in the original high-dimensional data space.

A precedent of interactive visualization of scientific information flows (such as citation patterns across disciplines and journals, and temporal evolution of citation indices) is provided by [22, 27]. A notable difference of this approach is that the information being visualized is obtained from bibliometric data (journal citation reports) rather than data obtained through automatic content extraction from publications.

3. DATA ACQUISITION AND FEATURE EXTRACTION

Text-based information extraction and retrieval commonly relies on the *bag of words* model, which can be traced back at least to [19]. According to this model, a document is represented as an unordered set of its words, ignoring structure and grammar rules. Words can be generalized to terms, where a term may be a single word or a sequence of n words (*n-grams*), or correspond to some grammatical structure, e.g., a noun phrase. Using such a bag of words representation, each term t describing a particular document d is commonly assigned a weight $w_{t,d}$ that estimates the importance of t in d. Each document can then be described by a *feature vector* that aggregates the single term weights. When considering a whole corpus of documents, each document can be thought of as a representation of its feature vector in a *feature space* or *vector space* whose dimensions correspond to the particular term weights. This so-called *vector space model* is a fundamental model in information retrieval and was originally described in [25].

For the term weighting function $w_{t,d}$, in modern information retrieval, typically some variant of TF·IDF scores is used. The TF term gives more weight to terms that appear many times in a document, whereas the IDF term ensures that less weight is given to terms that appear in many documents. More details on term weighting via TF·IDF can be found in [32]. The TF·IDF function assigns a weight $w_{t,d}$ to a particular term t and document d. Calculating $w_{t,d}$ for all terms remaining after having performed term selection on the terms extracted from the corpus thus yields a representation of d as a term weight vector in the feature space.

Following these basic principles of text-based information retrieval, we performed feature extraction as follows. First, we retrieved the PDF files of the accepted ISMIR submissions from the online repository [11]. This yielded effectively 719 documents. Subsequently, we converted the PDF files to standard text files. To this end, the GNU/-Linux tools *pdftotext* from *xpdf-utils* and *iconv* from *libc6* were used. Minor problems encountered in the transcription process, such as occasional truncation of words, were addressed by prefix/suffix filtering as detailed later. Next, we employed the part-of-speech (POS) tagger *Geniatagger* [26] to extract all noun phrases from the corpus since we believe that these are most important to describe the content of ISMIR papers. As the output of the POS tagger

contained a lot of noise, we subsequently applied some ad-hoc filters: We discarded all terms containing non-alphabetic characters and retained only trigrams, bigrams, and unigrams. This yielded approximately 70,000 terms.
Since we aimed at emphasizing terms important to MIR, we performed term selection via Web-based filtering of terms that tend to be important in a general context. Such terms thus tend to be rather unimportant in the context of MIR and also not very discriminative for the corpus of ISMIR papers. For this purpose, we queried the Web search engine *exalead* [5] for the extracted n-grams as exact phrase and retrieved the returned page-count-values. We then discarded all terms whose TF in the ISMIR corpus was lower than their page-count-value. To alleviate the problem of truncated words after PDF-to-text-conversion, we removed any n-gram v that was a prefix/suffix of another n-gram w (of equal n) and whose minimal TF among the single words occurring in v was lower than the minimal TF among the single words occurring in w, assuming that truncated words typically have a low TF.
This approach finally yielded a term list of approximately 12,500 terms. We extracted, for each of these terms, its absolute TF count per ISMIR document and its global DF count in the corpus. Weighting each TF value with the (logarithmic) IDF obtained from the ISMIR corpus according to Formula 1 provided a TF·IDF vector representation of each ISMIR document. In Equation 1, n is the total number of documents in the corpus, $tf_{t,d}$ is the number of occurrences of term t in d, and df_t is the number of documents in the whole corpus in which t occurs at least once. Concatenating the TF·IDF representation of all documents yields a term-document matrix.

$$w_{t,d} = \begin{cases} tf_{t,d} \cdot \log \frac{n}{df_t} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

4. VISUALIZING ISMIR

Determining and illustrating the most important concepts tackled by ISMIR papers over time, we used Non-Negative Matrix Factorization (NMF) and Principal Components Analysis (PCA) as elaborated in the following.

4.1 Finding Concepts by Non-Negative Matrix Factorization

Topic detection on the TF·IDF vectors, calculated as described in the previous section, was performed as proposed, for example, in [9, 18, 21, 29]. For NMF calculation, the cost function is the square of the Euclidean distance, and update takes place by the standard multiplicative update rules. Initialization is done randomly. NMF aims to find an approximate decomposition (into matrices W and H) with non-negativity constraints, cf. Equation 2, where V is the $n \times m$ matrix of the 12,500-dimensional TF·IDF vectors and $m = 719$ documents.

$$V \approx WH \tag{2}$$

Matrix W is interpreted as containing the amount each of the n terms is associated with each of r concepts, and H as containing the amount each document is associated with each of the r concepts.

4.2 Changes of ISMIR Topics over Time

The association between concepts and documents that results from NMF over the term-document matrix, allows us to make an association between concepts and years (by summing the activation of concepts in the documents of each year). Figure 1 shows the evolution of $r = 22$ concepts over time, with the overall height representing the number of relevant publications in the year. The legend shows the three top-weighted terms for each concept. The concepts have been ordered vertically according to their growth/decline over time, the most growing concepts being on top. To this end, we performed linear regression over the development of concept weights during the considered time span.

Several interesting observations can be made. Firstly, note that the concepts seem to be of different categories. Whereas some concepts clearly represent topics (e.g., genre classification, onset detection, rhythm description, or fingerprinting), others seem to represent methods that can be used to solve different types of problems (such as matrix factorization or dynamic time-warping).

Secondly, even if the presence of most concepts is rather stable over the years, there are some notable changes over time. Some of the changes that the analysis reveals are not very surprising, such as the fact that semantic audio annotation performed via collaborative tagging (such as employed by last.fm) or Web mining, was virtually absent as a research topic in the first ISMIR conferences. By 2008, it has gained the largest share. Other changes are less obvious. For example, the share of query-by-humming/singing in ISMIR 2002 papers was considerably higher than it was in later years. Furthermore, genre classification seems to have boomed briefly at ISMIR 2005. This might be related to the MIREX 2005 genre classification contest.

4.3 A Web-Interface to Concept-based search of ISMIR publications

The concepts found by NMF can also be used to create an interface for searching ISMIR papers associated with particular concepts. The simplest form of such an interface is a selection screen that lets the user select one or more of the r concepts of interest. The user selection is transformed into a vector of length r, with all entries set to zero except these that correspond to selected concepts, which are set to one. This vector is used as a query vector, and compared to each of the documents' concept vectors by cosine similarity [23]. The outcome is then presented to the user as a list of suggested documents ranked according to their similarity to the query vector. More elaborate approaches would include search refinement (e.g., by relevance feedback), or query term expansion based on the

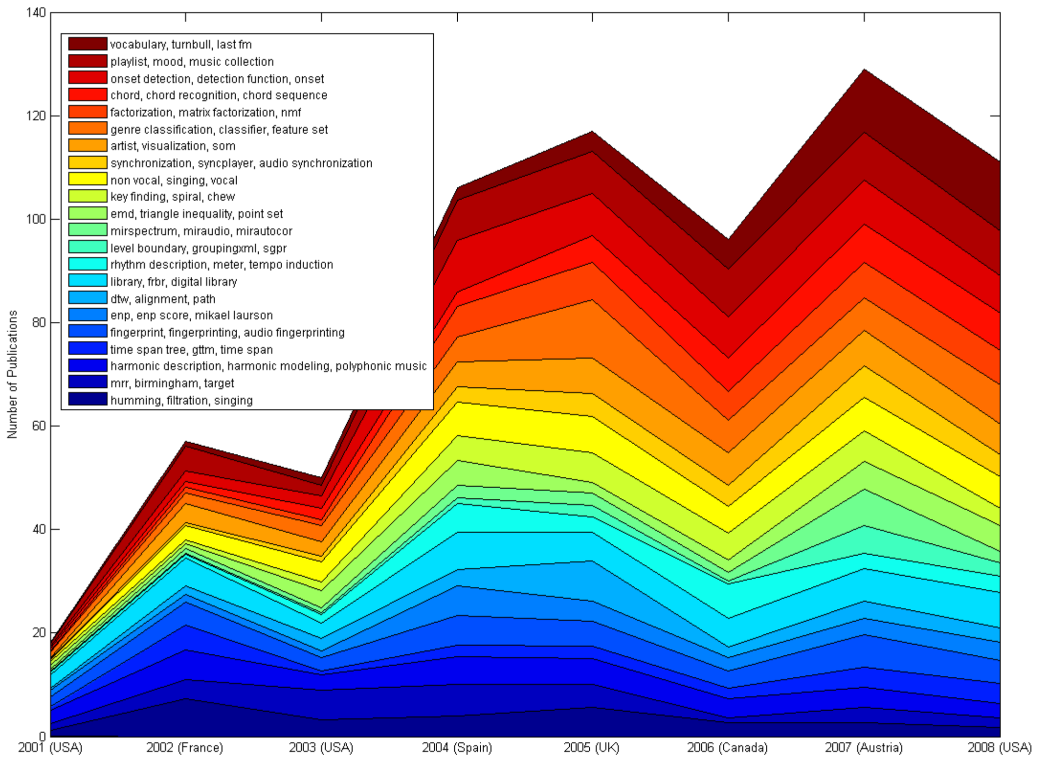


Figure 1. Evolution of the main ISMIR topics over the years.

concept vectors. We have implemented a small prototype application for document retrieval as a web service.

4.4 *ISMIRviewer*: Navigating the ISMIR Cloud

To visualize the semantic content of of the joint ISMIR publications, we pursue the idea of the *tag cloud*. In this subsection, we describe how we construct the tag cloud containing the terms extracted from the documents, and present the *ISMIRviewer*, an application for interactively navigating this tag cloud.

The term-document matrix that contains the IDF-weighted term frequency of each term in each of the 719 documents can be seen as specifying each term as a point in a 719-dimensional Euclidean space. In general, the more frequently two terms occur in the same subset of documents, the closer they will be in this space. This high-dimensional space, however, cannot be used directly for visualizing the relationships among terms. Moreover, if multiple documents contain the same terms with similar frequencies, there will be redundancies in the corresponding dimensions. In this case, the dimensionality of the feature space can be reduced without losing information about the distance between the terms and their relative location. PCA is a technique for such a dimensionality reduction, in which the data is projected on a set of orthogonal dimensions that have been rotated to maximize the variance along each dimension. The dimensions are ordered according to the data variance they hold. In this way, a subset of dimensions of any size can be chosen with maximal data variance. The principal components are obtained by computing the eigenvalues of the covariance matrix of the data, cf. [12].

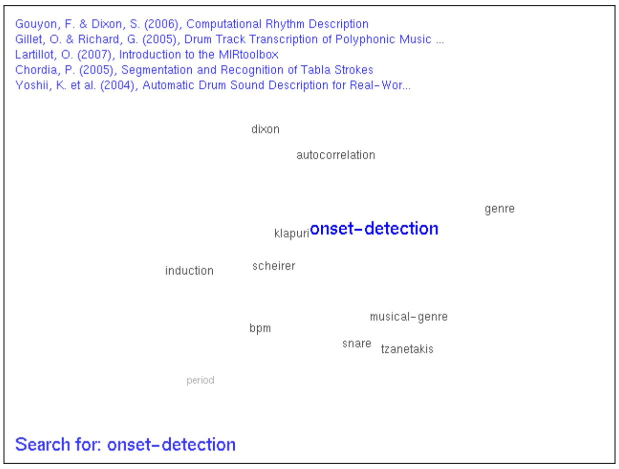
Since we aim at providing a spatial visualization, the number of dimensions is obviously limited to three. However, we found that projecting the data from 719 to three dimensions directly was not useful in this case as the first three principal components accounted only for 12% of the variance in the data (90% being reached when using 347 dimensions). When visualized, the terms are very condensed in space, where the variance is highly dominated by a few common terms like "music" and "audio". Using the logarithms of term frequencies alleviated this problem slightly, but not satisfyingly.

Instead, we have opted for a two-stage approach to data reduction. The first stage applies NMF, as described in subsection 4.1. It yields a small set of basis vectors, which are formally activation patterns over documents, and tend to represent musically meaningful concepts. Each term t has an activation value for each concept c, denoting how relevant t is to c. Experimentation with NMF using different numbers of concepts shows that an NMF reduction to twenty concepts include most recognizable subfields of MIR without introducing many unrecognizable concepts [2]. Given these twenty concepts, terms are filtered to include only the 100 most activated terms for each of the concepts.

As a second stage, we perform dimensionality reduction to three dimensions through PCA on the subset of terms and the activations over the twenty concepts that were obtained in the first stage. The resulting space is less densely populated and the terms it contains tend to be more MIR-relevant.

For interactive inspection of the constructed tag cloud,

[2] As judged informally by the authors


Gouyon, F. & Dixon, S. (2006), Computational Rhythm Description
Gillet, O. & Richard, G. (2005), Drum Track Transcription of Polyphonic Music ...
Lartillot, O. (2007), Introduction to the MIRtoolbox
Chordia, P. (2005), Segmentation and Recognition of Tabla Strokes
Yoshii, K. et al. (2004), Automatic Drum Sound Description for Real-Wor...

Search for: onset-detection

Figure 2. Screenshot of the ISMIRviewer showing the *onset-detection* neighborhood of the ISMIR Cloud.

we have developed the *ISMIRviewer*, that allows the user to freely rotate the space and zoom in on regions using the mouse. Furthermore, subsets of the cloud can be selected by text search. As the user types, the matching tags light up. For each matching tag, neighboring tags are displayed, while remote and non-matching tags are dimmed. For the given selection of tags, five publications are shown in the corner of the screen that have been determined to be the most relevant for that term. Instead of determining document relevance through TF·IDF, the term is mapped to the documents via the concepts found by NMF. This effectively realizes a document search by *query expansion*.

In this way, the user can search for MIR-related topics, methods, or author names, and obtain relevant publications. Figure 2 shows a screenshot of the application. The displayed terms are the result of searching for the term *onset-detection*. The neighborhood of the search term (small black font) contains related concepts, e.g., *period*, *bpm*, techniques used (e.g. *autocorrelation*), and authors who have published on onset detection, such as Klapuri and Dixon.

5. CONCLUSIONS

In this paper, we analyzed the proceedings of the past ISMIR conferences, extracted terms from the documents, and employed text and Web mining techniques to distill a set of n-grams we believe to be important to describe the field of music information retrieval. Using a TF·IDF weighting function, we described each document by means of its term weights. We then applied clustering techniques to reveal the most important concepts covered by the ISMIR papers. Furthermore, a year-wise analysis of the publications revealed interesting changes of topics addressed in ISMIR over the years. For example, in ISMIR 2002, query-by-humming was a major topic, that has received considerably less attention in later years. Furthermore, genre classification had a particularly large share in ISMIR 2005.

Moreover, we presented two prototype applications that provide access to the semantic content of the past ISMIR publications. The first one is a Web-based retrieval system to search the corpus of ISMIR proceedings via the concepts found by NMF. The second one, which we call *ISMIRViewer*, provides an interactive tag cloud visualization to reveal the relationships between MIR related terms. It employs a focus and context technique to show subsets of the tag cloud in response to user-entered text queries, and provides the ISMIR publications that are most relevant to the text queries.

The applications are presented as a proof-of-concept, their user-interfaces leave room for improvement. Further work to be done includes investigating other clustering techniques, e.g., *Aligned Self-Organizing Maps* [20] or *Music Description Maps* [14].

6. ACKNOWLEDGMENTS

This work was supported by the Austrian FWF under grants L511 and P19349.

7. REFERENCES

[1] P. Cimiano, S. Handschuh, and S. Staab. Towards the Self-Annotating Web. In *Proceedings of the 13th International Conference on World Wide Web (WWW 2004)*, pages 462–471, New York, NY, USA, 2004. ACM Press.

[2] F. Debole and F. Sebastiani. Supervised Term Weighting for Automated Text Categorization. In *Proceedings the 18th ACM Symposium on Applied Computing (SAC 2003)*, pages 784–788, Melbourne, FL, USA, March 9–12 2003. ACM.

[3] D. Eck, T. Bertin-Mahieux, and P. Lamere. Autotagging Music Using Supervised Machine Learning. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna, Austria, September 23–27 2007.

[4] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Methods for Domain-Independent Information Extraction from the Web: An Experimental Comparison. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2004)*, pages 391–398, San Jose, CA, USA, 2004.

[5] Exalead: Redefining information access for the enterprise and the web:. `http://www.exalead.com`, 2009. (access: April 2009).

[6] V. Fresno, R. Martínez, and S. Montalvo. Improving web page clustering through selecting appropiate term weighting functions. In *Proceedings of the 1st IEEE International Conference on Digital Information Management (ICDIM 2006)*, pages 511–518, Bangalore, India, December 6–8 2006.

[7] M. A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th*

Conference on Computational Linguistics – Vol. 2, pages 539–545, Nantes, France, August 1992.

[8] H. Hotelling. Analysis of a Complex of Statistical Variables Into Principal Components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.

[9] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[10] X. Hu, M. Bay, and J. S. Downie. Creating a Simplified Music Mood Classification Ground-Truth Set. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna, Austria, September 23–27 2007.

[11] International society for music information retrieval, ismir: Conferences, publications and related activities. `http://www.ismir.net`, 2009. (access: May 2009).

[12] I. T. Jolliffe. *Principial Component Analysis*. Springer, New York, NY, USA, 1986.

[13] P. Knees, E. Pampalk, and G. Widmer. Artist Classification with Web-based Data. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR 2004)*, pages 517–524, Barcelona, Spain, October 10–14 2004.

[14] P. Knees, T. Pohle, M. Schedl, and G. Widmer. Automatically Describing Music on a Map. In *Proceedings of 1st Workshop on Learning the Semantics of Audio Signals (LSAS 2006)*, Athens, Greece, December 6–8 2006.

[15] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A Music Search Engine Built upon Audio-based and Web-based Similarity Measures. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, Amsterdam, the Netherlands, July 23–27 2007.

[16] M. Lan, C.-L. Tan, H.-B. Low, and S.-Y. Sung. A Comprehensive Comparative Study on Term Weighting Schemes for Text Categorization with Support Vector Machines. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web (WWW 2005)*, pages 1032–1033, Chiba, Japan, May 10–14 2005. ACM Press.

[17] Last.fm - listen to internet radio and the largest music catalogue online. `http://last.fm`, 2008. (access: April 2009).

[18] D. D. Lee and H. S. Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401(6755):788–791, 1999.

[19] H. P. Luhn. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal*, pages 309–317, October 1957.

[20] E. Pampalk. Aligned Self-Organizing Maps. In *Proceedings of the Workshop on Self-Organizing Maps (WSOM 2003)*, pages 185–190, Kitakyushu, Japan, September 11–14 2003. Kyushu Institute of Technology.

[21] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Building an Interactive Next-Generation Artist Recommender Based on Automatically Derived High-Level Concepts. In *Proceedings of the 5th International Workshop on Content-Based Multimedia Indexing (CBMI'07)*, Bordeaux, France, June 25–27 2007.

[22] M. Rosvall and C. T. Bergstrom. Maps of information flow reveal community structure in complex networks. In *Proceedings of the National Academy of Sciences USA*, volume 105, pages 1118–1123, 2007.

[23] G. Salton. The Use of Citations as an Aid to Automatic Content Analysis. Technical Report ISR-2, Section III, Harvard Computation Laboratory, Cambridge, MA, USA, 1962.

[24] G. Salton and C. Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[25] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[26] Y. Tsuruoka and J. Tsujii. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of HLT/EMNLP*, pages 467–474, 2005.

[27] well-formed.eigenfactor.org : Visualizing information flow in science:. `http://well-formed.eigenfactor.org`, 2009. (access: May 2009).

[28] B. Whitman and S. Lawrence. Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of the 2002 International Computer Music Conference (ICMC 2002)*, pages 591–598, Göteborg, Sweden, September 16–21 2002.

[29] W. Xu, X. Liu, and Y. Gong. Document Clustering Based on Non-negative Matrix Factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 267–273, Toronto, Canada, July 28–August 1 2003. ACM Press.

[30] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, USA, 1997. Morgan Kaufman.

[31] J. Zobel and A. Moffat. Exploring the Similarity Space. *ACM SIGIR Forum*, 32(1):18–34, 1998.

[32] J. Zobel and A. Moffat. Inverted Files for Text Search Engines. *ACM Computing Surveys*, 38:1–56, 2006.

TOWARDS AUTOMATED EXTRACTION OF TEMPO PARAMETERS FROM EXPRESSIVE MUSIC RECORDINGS

Meinard Müller, Verena Konz, Andi Scharfstein
Saarland University and MPI Informatik
Saarbrücken, Germany
{meinard,vkonz,ascharfs}@mpi-inf.mpg.de

Sebastian Ewert, Michael Clausen
Bonn University, Computer Science
Bonn, Germany
{ewerts,clausen}@iai.uni-bonn.de

ABSTRACT

A performance of a piece of music heavily depends on the musician's or conductor's individual vision and personal interpretation of the given musical score. As basis for the analysis of artistic idiosyncrasies, one requires accurate annotations that reveal the exact timing and intensity of the various note events occurring in the performances. In the case of audio recordings, this annotation is often done manually, which is prohibitive in view of large music collections. In this paper, we present a fully automatic approach for extracting temporal information from a music recording using score-audio synchronization techniques. This information is given in the form of a tempo curve that reveals the relative tempo difference between an actual performance and some reference representation of the underlying musical piece. As shown by our experiments on harmony-based Western music, our approach allows for capturing the overall tempo flow and for certain classes of music even finer expressive tempo nuances.

1. INTRODUCTION

Musicians give a piece of music their personal touch by continuously varying tempo, dynamics, and articulation. Instead of playing mechanically they speed up at some places and slow down at others in order to shape a piece of music. Similarly, they continuously change the sound intensity and stress certain notes. The automated analysis of different interpretations, also referred to as *performance analysis*, has become an active research field [1–4]. Here, one goal is to find commonalities between different interpretations, which allow for the derivation of general performance rules. A kind of orthogonal goal is to capture what is characteristic for the style of a particular musician. Before one can analyze a specific performance, one requires the information about when and how the notes of the underlying piece of music are actually played. Therefore, as the first step of performance analysis, one has to annotate the performance by means of suitable attributes that make explicit the exact timing and intensity of the various note events. The extraction of such performance attributes constitutes a challenging problem, in particular for the case of audio recordings.

Many researchers manually annotate the audio material by marking salient data points in the audio stream. Using novel music analysis interfaces such as the Sonic Visualiser [5], experienced annotators can locate note onsets very accurately even in complex audio material [2, 3]. However, being very labor-intensive, such a manual process is prohibitive in view of large audio collections. Another way to generate accurate annotations is to use a computer-monitored *player piano*. Equipped with optical sensors and electromechanical devices, such pianos allow for recording the key movements along with the acoustic audio data, from which one directly obtains the desired note onset information [3, 4]. The advantage of this approach is that it produces precise annotations, where the symbolic note onsets perfectly align with the physical onset times. The obvious disadvantage is that special-purpose hardware is needed during the recording of the piece. In particular, conventional audio material taken from CD recordings cannot be annotated in this way. Therefore, the most preferable method is to automatically extract the necessary performance aspects directly from a given audio recording. Here, automated approaches such as *beat tracking* [6, 8] and *onset detection* [9] are used to estimate the precise timings of note events within the recording. Even though great research efforts have been directed towards such tasks, the results are still unsatisfactory, in particular for music with weak onsets and strongly varying beat patterns. In practice, semi-automatic approaches are often used, where one first roughly computes beat timings using beat tracking software, which are then adjusted manually to yield precise beat onsets.

In this paper, we present a novel approach towards extracting temporal performance attributes from music recordings in a fully automated fashion. We exploit the fact that for many pieces there exists a kind of "neutral" representation in the form of a musical score (or MIDI file) that explicitly provides the musical onset and pitch information of all occurring note events. Using music synchronization techniques, we temporally align these note events with their corresponding physical occurrences in the music recording. As our main contribution, we describe various algorithms for deriving tempo curves from these align-

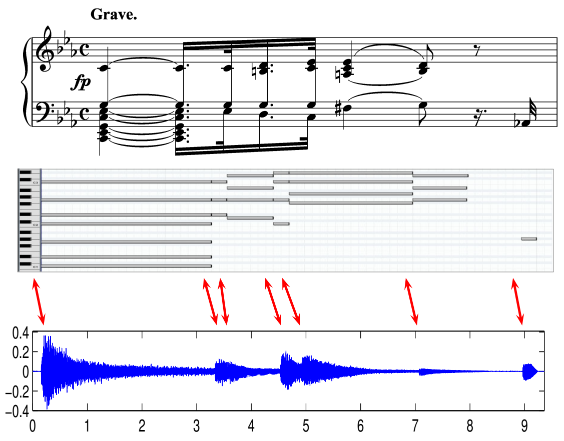


Figure 1. First measure of Beethoven's Pathétique Sonata Op. 13. The MIDI-audio alignment is indicated by the arrows.

ments which reveal the relative tempo differences between the actual performance and the neutral reference representation. We have evaluated the quality of the automatically extracted tempo curves on harmony-based Western music of various genres. Besides a manual inspection of a representative selection of real music performances, we have also conducted a quantitative evaluation on synthetic audio material generated from randomly warped MIDI files. Our experiments indicate that our automated methods yield accurate estimations of the overall tempo flow and, for certain classes of music such as piano music, of even finer expressive tempo nuances.

The remainder of this paper is organized as follows. After reviewing some basics on music synchronization (Sect. 2), we introduce various algorithms for extracting tempo curves from expressive music recordings (Sect. 3). Our experiments are described in Sect. 4, and prospects on future work are sketched in Sect. 5. Further related work is discussed in the respective sections.

2. MUSIC SYNCHRONIZATION

The largest part of Western music is based on the equal-tempered scale and can be represented in the form of musical scores, which contain high-level note information such as onset time, pitch, and duration. In the following, we assume that a score is given in the form of a “neutral” MIDI file, where the notes are played with a constant tempo in a purely mechanical way. We refer to this MIDI file as *reference representation* of the underlying piece of music. On the other hand, we assume that the performance to be analyzed is given in the form of an audio recording. In a first step, we use conventional *music synchronization* techniques to temporally align the note events with their corresponding physical occurrences in the audio recording [10, 11]. The synchronization result can be regarded as an automated annotation of the audio recording with the note events given by the MIDI file, see Fig. 1.

Most synchronization algorithms rely on some variant of dynamic time warping (DTW) and can be summarized as follows. First, the MIDI file and the audio recording

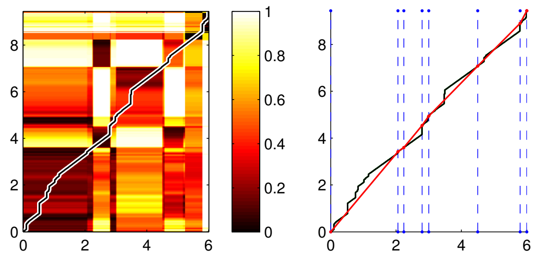


Figure 2. **Left:** Cost matrix and cost-minimizing alignment path for the Beethoven example shown in Fig. 1. The reference representation (MIDI) corresponds to the horizontal and the performance (audio) to the vertical axis. **Right:** Original (black) and onset-rectified alignment path (red). The MIDI note onset positions are indicated by the blue vertical lines.

to be aligned are converted into feature sequences, say $X := (x_1, x_2, \ldots, x_N)$ and $Y := (y_1, y_2, \ldots, y_M)$, respectively. Then, an $N \times M$ cost matrix C is built up by evaluating a local cost measure c for each pair of features, i. e., $C((n,m)) = c(x_n, y_m)$ for $n \in [1 : N] := \{1, 2, \ldots, N\}$ and $m \in [1 : M]$. Each tuple $p = (n,m)$ is called a *cell* of the matrix. A (global) *alignment path* is a sequence $(p_1, \ldots, p_L)$ of length L with $p_\ell \in [1 : N] \times [1 : M]$ for $\ell \in [1 : L]$ satisfying $p_1 = (1,1)$, $p_L = (N,M)$ and $p_{\ell+1} - p_\ell \in \Sigma$ for $\ell \in [1 : L-1]$. Here, $\Sigma = \{(1,0),(0,1),(1,1)\}$ denotes the set of admissible step sizes. The *cost* of a path $(p_1, \ldots, p_L)$ is defined as $\sum_{\ell=1}^{L} C(p_\ell)$. A cost-minimizing alignment path, which constitutes the final synchronization result, can be computed via dynamic programming from C, see Fig. 2. For a detailed account on DTW and music synchronization we refer to [11].

Based on this general strategy, we employ a synchronization algorithm based on high-resolution audio features as described in [12]. This approach, which combines the high temporal accuracy of onset features with the robustness of chroma features, generally yields robust music alignments of high temporal accuracy. In the following, we use a feature resolution of 50 Hz with each feature vector corresponding to 20 milliseconds of MIDI or audio. For details, we refer to [12].

3. COMPUTATION OF TEMPO CURVES

The feeling of pulse and rhythm is one of the central components of music and closely relates to what one generally refers to as tempo. In order to define some notion of tempo, one requires a proper reference to measure against. For example, Western music is often structured in terms of measures and beats, which allows for organizing and sectioning musical events over time. Based on a fixed time signature, one can then define the tempo as the number of beats per minute (BPM). Obviously, this definition requires a regular and steady musical beat or pulse over a certain period in time. Also, the very process of measurement is not as well-defined as one may think. Which musical entities (e. g., note onsets) characterize a pulse? How precisely can these entities be measured before getting drowned in noise? How many pulses or beats are needed to obtain a meaningful tempo estimation? With these questions, we want to indicate that the notion of tempo is far from being well-defined. Different representations of timing and tempo are presented in [7].

In this paper, we assume that we have a reference representation of a piece of music in the form of a MIDI file generated from a score using a fixed global tempo (measured in BPM). Assuming that the time signature of the piece is known, one can recover measure and beat positions from MIDI time positions. Given a specific performance in the form of an audio recording, we first compute a MIDI-audio alignment path as described in Sect. 2. From this path we derive a *tempo curve* that describes for each time position within the MIDI reference (given in seconds or measures) the tempo of the performance (given as a multiplicative factor of the reference tempo or in BPM). Fig. 4 and Fig. 5 show some tempo curves for various performances.

Intuitively, the value of the tempo curve at a certain reference position corresponds to the slope of the alignment path at that position. However, due to discretization and alignment errors, one needs numerically robust procedures to extract the tempo information by using average values over suitable time windows. In the following, we describe three different approaches for computing tempo curves using a fixed window size (Sect. 3.1), an adaptive window size (Sect. 3.2), and a combined approach (Sect. 3.3).

3.1 Fixed Window Size

Recall from Sect. 2 that the alignment path $p = (p_1, \ldots, p_L)$ between the MIDI reference and the performance is computed on the basis of the feature sequences $X = (x_1, \ldots, x_N)$ and $Y = (y_1, \ldots, y_M)$. Note that one can recover beat and measure positions from the indices $n \in [1 : N]$ of the reference feature sequence, since the MIDI representation has constant tempo and the feature rate is assumed to be constant.

To compute the tempo of the performance at a specific reference position $n \in [1 : N]$, we basically proceed as follows. First, we choose a neighborhood of n given by indices n_1 and n_2 with $n_1 \leq n \leq n_2$. Using the alignment path, we compute the indices m_1 and m_2 aligned with n_1 and n_2, respectively. Then, the tempo at n is defined as quotient $\frac{n_2 - n_1 + 1}{m_2 - m_1 + 1}$. The main parameter to be chosen in this procedure is the size of the neighborhood. Furthermore, there are some technical details to be dealt with. Firstly, the boundary cases at the beginning and end of the reference need special care. To avoid boundary problems, we extend the alignment path p to the left and right by setting $p_\ell := (\ell, \ell)$ for $\ell < 1$ and $p_\ell := (N+\ell-L, M+\ell-L)$ for $\ell > L$. Secondly, the indices m_1 and m_2 are in general not uniquely determined. Generally, an alignment path p may assign more than one index $m \in [1 : M]$ to a given index $n \in [1 : N]$. To enforce uniqueness, we chose the minimal index over all possible indices. More precisely, we define a function $\varphi_p : \mathbb{Z} \to [1 : M]$ by setting

$$\varphi_p(n) := \min\{m \in [1 : M] \mid \exists \ell \in \mathbb{Z} : p_\ell = (n, m)\}.$$

We now give the technical details of the sketched procedure for the case that the neighborhoods are of a fixed window (FW) size $w \in \mathbb{N}$. The resulting tempo curve is denoted by $\tau_w^{\mathrm{FW}} : [1 : N] \to \mathbb{R}_{\geq 0}$. For a given alignment path p and an index $n \in [1 : N]$, we define

$$n_1 := n - \left\lfloor \tfrac{w-1}{2} \right\rfloor \quad \text{and} \quad n_2 := n + \left\lceil \tfrac{w-1}{2} \right\rceil. \qquad (1)$$

Then $w = n_2 - n_1 + 1$ and the tempo at reference position n is defined by

$$\tau_w^{\mathrm{FW}}(n) = \frac{w}{\varphi_p(n_2) - \varphi_p(n_1) + 1}. \qquad (2)$$

The tempo curve τ_w^{FW} crucially depends on the window size w. Using a small window allows for capturing sudden tempo changes. However, in this case the tempo curve becomes sensible to inaccuracies in the alignment path and synchronization errors. In contrast, using a larger window smooths out possible inaccuracies, while limiting the ability to accurately pick up local phenomena. This effect is also illustrated by Fig. 3 (a), where the performance is synthesized from a temporally warped MIDI reference. We continue this discussion in Sect. 4.

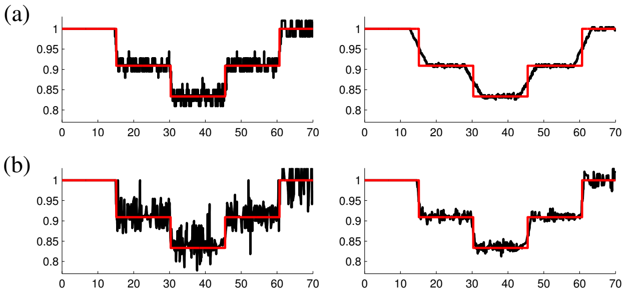


Figure 3. Ground truth tempo curve (step function) and various computed tempo curves. **(a)** τ_w^{FW} using a fixed window size with small w (left) and large w (right). **(b)** τ_v^{AW} using an adaptive window size with small v (left) and large v (right).

3.2 Adaptive Window Size

Using a window of fixed size does not account for specific musical properties of the piece of music. We now introduce an approach using an adaptive window size, which is based on the assumption that note onsets are the main source for inducing tempo information. Intuitively, in passages where notes are played in quick succession one may obtain an accurate tempo estimation even when using only a small time window. In contrast, in passages where only few notes are played one needs a much larger window to obtain a meaningful tempo estimation.

We now formalize this idea. We assume that the note onsets of the MIDI reference are given in terms of feature indices. Furthermore, for notes with the same onset position we only list one of these indices. Let $O = \{o_1, \ldots, o_K\} \subseteq [1 : N]$ be the set of onset positions with $1 \leq o_1 < o_2 < \ldots < o_K \leq N$. The distance between two neighboring onset positions is referred to as inter onset interval (IOI). Now, when computing the tempo curve at position $n \in [1 : N]$, the neighborhood of n is specified not in terms of a fixed number w of feature indices but in terms of a fixed number $v \in \mathbb{N}$ of IOIs. This defines an onset-dependent adaptive window (AW). More precisely, let $\tau_v^{\mathrm{AW}} : [1 : N] \to \mathbb{R}_{\geq 0}$ denote the tempo function to be computed. To avoid boundary problems, we extended the set O to the left and right by setting $o_k := o_1 + k - 1$ for $k < 1$ and $o_k := o_K + k - K$ for $k > K$. First, we compute τ_v^{AW} for all indices n that correspond to onset positions. To this end, let $n = o_k$. Then we define

$$k_1 := k - \left\lfloor \tfrac{v-1}{2} \right\rfloor \quad \text{and} \quad k_2 := k + \left\lceil \tfrac{v-1}{2} \right\rceil.$$

Setting $n_1 := o_{k_1}$ and $n_2 := o_{k_2}$, the tempo at reference position $n = o_k$ is defined as

$$\tau_v^{\mathrm{AW}}(n) := \frac{n_2 - n_1 + 1}{\varphi_p(n_2) - \varphi_p(n_1) + 1}. \tag{3}$$

Note that, opposed to (2), the window size $n_2 - n_1 + 1$ is no longer fixed but depends on the sizes of the neighboring IOIs around the position $n = o_k$. Finally, $\tau_v^{\mathrm{AW}}(n)$ is defined by a simple linear interpolation for the remaining indices $n \in [1 : N] \setminus O$. Similar to the case of a fixed window size, the tempo curve τ_v^{AW} crucially depends on the number v of IOIs, see Fig. 3 (b). The properties of the various tempo curves are discussed in detail in Sect. 4.

3.3 Combined Strategy

So far, we have introduced two different approaches using on the one hand a fixed window size and on the other hand an onset-dependent adaptive window size for computing average slopes of the alignment path. Combining ideas from both approaches, we now present a third strategy, where we first rectify the alignment path using onset information and then apply the FW-approach on the rectified path for computing the tempo curve. As in Sect. 3.2, let $O = \{o_1, \ldots, o_K\} \subseteq [1 : N]$ be the set of onsets. By possibly extending this set, we may assume that $o_1 = 1$ and $o_K = N$. Now, within each IOI given by two neighboring onsets $n_1 := o_k$ and $n_2 := o_{k+1}$, $k \in [1 : K-1]$, we modify the alignment path p as follows. Let $\ell_1, \ell_2 \in [1 : L]$ be the indices with $p_{\ell_1} = (n_1, \varphi_p(n_1))$ and $p_{\ell_2} = (n_2, \varphi_p(n_2))$, respectively. While keeping the cells p_{ℓ_1} and p_{ℓ_2}, we replace the cells $p_{\ell_1} + 1, \ldots, p_{\ell_2} - 1$ by cells obtained from a suitably sampled linear function having the slope $\frac{n_2 - n_1 + 1}{\varphi_p(n_2) - \varphi_p(n_1) + 1}$. Here, in the sampling, we ensure that the step size condition given by Σ is fulfilled, see Sect. 2. The resulting rectification is illustrated by Fig. 2 (right). Using the rectified alignment path, we then compute the tempo curve using a fixed window size $w \in \mathbb{N}$ as described in Sect. 3.1. The resulting tempo curve is denoted by τ_w^{FWR}. This third approach, as our experiments show, generally yields more robust and accurate tempo estimations than the other two approaches.

4. EXPERIMENTS

In this section, we first discuss some representative examples and then report on a systematic evaluation based on temporally warped music. In the following, we specify the window size w in terms of seconds instead of samples. For example, by writing $w \propto 3$ seconds, we mean that $w \in \mathbb{N}$ is a window size with respect to the feature rate corresponding to 3 seconds of the underlying audio.

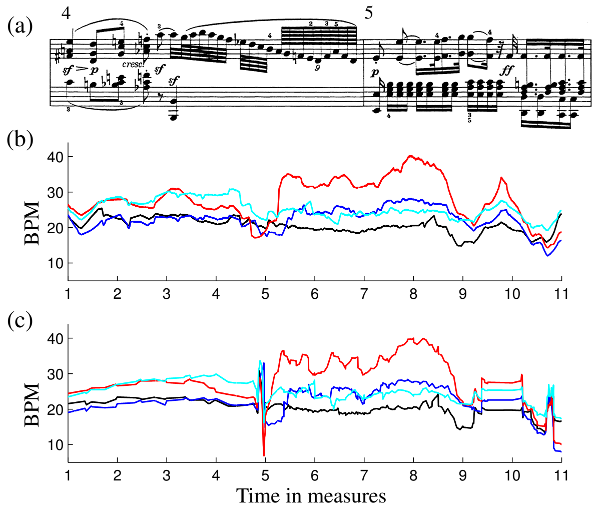


Figure 4. Tempo curves of four different interpretations played by different pianists of the first ten measures (slow introductory theme marked *Grave*) of Beethoven's Pathétique Sonata Op. 13. **(a)** Score of measures 4 and 5. **(b)** Tempo curves τ_w^{FWR} for $w \propto$ 3 seconds. **(c)** Tempo curves τ_v^{AW} for $v = 10$ IOIs.

In our first example, we consider Beethoven's Pathétique Sonata Op. 13. The first ten measures correspond to the slow introductory theme marked *Grave*. For these measure, Fig. 4 (b) shows the tempo curves τ_w^{FWR} for four different performances using the combined strategy with a window size $w \propto 3$ seconds. From these curves, one can read off global and local tempo characteristics. For example, the curves reveal the various tempi chosen by the pianists, ranging from roughly 20 to 30 BPM. One of the pianists (red curve) significantly speeds up after measure 5, whereas the other pianists use a more balanced tempo throughout the introduction. It is striking that all four pianists significantly slow down in measure 8, then accelerate in measure 9, before slowing down again in measure 10. Musically, the last slow-down corresponds to the fermata at the end of measure 10, which concludes the *Grave*. Similarly, the curves indicate a ritardando in all four performances towards the end of measure 4. In this passages, there is a run of 64th notes with a closing nonuplet, see Fig. 4 (a). Using a fixed window size, the ritardando effect is smoothed out to a large extent, see Fig. 4 (b). However, having many consecutive note onsets within a short passage, the ritardando becomes much more visible when using tempo curves with an onset-dependent adaptive window size. This is illustrated by Fig. 4 (c), which shows the four tempo curves τ_v^{AW} with $v = 10$ IOIs.

As a second example, we consider the Schubert Lied *Der Lindenbaum* (D. 911 No. 5). The first seven measures (piano introduction) are shown in Fig. 5 (a). Using the combined strategy with a window size $w \propto 3$ seconds, we computed tempo curves for 13 different interpretations, see Fig. 5 (b). As shown by the curves, all interpretations exhibit an accelerando in the first few measures followed

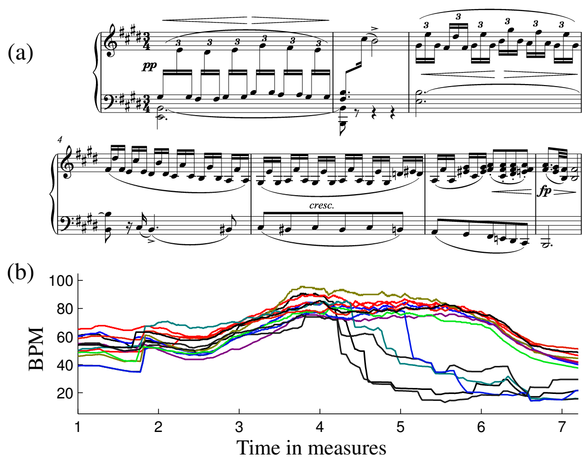


Figure 5. Tempo curves of 13 different performances of the beginning of the Schubert song *Der Lindenbaum*. **(a)** Score of measures 1 to 7. **(b)** Tempo curves τ_w^{FWR} for $w \propto 3$ seconds.

by a ritardando towards the end of the introduction. Interestingly, some of the pianists start with the ritardando in measure 4 already, whereas most of the other pianists play a less pronounced ritardando in measure 6. These examples indicate that our automatically extracted tempo curves are accurate enough for revealing interesting performance characteristics.

In view of a more quantitative evaluation, we computed tempo curves using different approaches and parameters on a corpus of harmony-based Western music of various genres. To allow for a reproduction of our experiments, we used pieces from the RWC music database [13]. In the following, we consider 15 representative pieces, which are listed in Table 1. These pieces include five classical piano pieces, five classical pieces of various instrumentations (full orchestra, strings, flute, voice) as well as five jazz pieces and pop songs. To automatically determine the accuracy of our tempo extraction procedures, we temporally modified MIDI files for each of the 15 pieces. To this end, we generated continuous piecewise linear tempo curves τ^{GT}, referred to as *ground-truth tempo curves*. These curves have a constant slope on segments of roughly 10 seconds of duration, where the slopes are randomly generated either using a value $v \in [1 : 2]$ (corresponding to an accelerando) or using a value $v \in [1/2 : 1]$ (corresponding to a ritardando). These values cover a range of tempo changes of $\pm 100\%$ of the reference tempo. Intuitively, the ground-truth tempo curves simulate on each segment a gradual transition between two tempi to mimic ritardandi and accelerandi. For an example, we refer to Fig. 6. We then temporally warped each of the original MIDI files with respect to a ground-truth tempo curve τ^{GT} and generated from the modified MIDI file an audio version using a high-quality synthesizer. Finally, we computed tempo curves using the original MIDI files as reference and the warped audio versions as performances.

To determine the accuracy of a computed tempo curve τ, we compared it with the corresponding ground-truth tempo curve τ^{GT}. Here, the idea is to measure deviations by *scale* rather than by *absolute value*. Therefore,

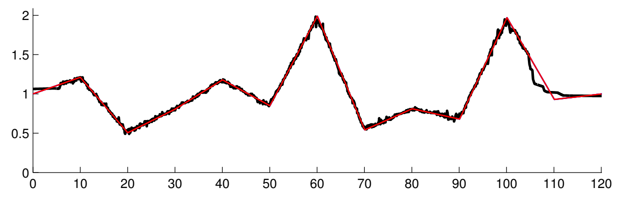


Figure 6. Piecewise linear ground-truth tempo curve (red) and computed tempo curves (black).

RWC ID (Comp./Int., Instr.)	**FW** μ	**FW** σ	**AW** μ	**AW** σ	**FWR** μ	**FWR** σ
C025 (Bach, piano)	3.29	7.30	2.60	5.05	1.59	2.86
C028 (Beethoven, piano)	3.24	6.98	6.36	21.14	2.66	6.72
C031 (Chopin, piano)	3.32	7.72	2.77	4.76	1.75	3.42
C032 (Chopin, piano)	2.54	4.17	3.05	4.67	1.56	2.34
C029 (Schumann, piano)	4.52	8.86	4.18	5.97	2.44	5.13
C003 (Beethoven, orchestra)	4.20	5.39	10.58	22.97	3.56	4.79
C015 (Borodin, strings)	2.44	2.85	4.68	9.85	2.25	2.71
C022 (Brahms, orchestra)	1.70	1.95	2.41	2.96	1.31	1.66
C044 (Rimski-K., flute/piano)	1.62	2.59	2.47	4.27	1.61	2.58
C048 (Schubert, voice/piano)	2.61	3.27	3.95	7.76	2.07	2.98
J001 (Nakamura, piano)	1.44	1.87	1.44	2.43	1.03	1.59
J038 (HH Band, big band)	2.24	2.96	3.20	5.41	1.91	2.74
J041 (Umitsuki, sax/bass/perc.)	1.88	2.40	3.75	4.69	1.72	2.34
P031 (Nagayama, electronic)	2.01	2.42	8.35	14.89	1.94	2.39
P093 (Burke, voice/guitar)	2.50	3.26	6.21	14.74	2.34	3.13
Average over all	2.64	4.27	4.40	8.77	1.98	3.16

Table 1. Tempo curve evaluation using the approaches FW and FWR (with $w \propto 4$ seconds) and AW (with $v = 10$ IOIs). The table shows for each of the 15 pieces the mean error μ and standard deviation σ (given in percent) of the computed tempo curves and the ground truth tempo curve. For generating the ground-truth tempo curves, MIDI segments of 10 seconds were used.

as distance function, we use the average multiplicative difference and standard deviation (both measured in percent) of τ and τ^{GT}. More precisely, we define

$$\mu(\tau, \tau^{\mathrm{GT}}) = 100 \cdot \frac{1}{N} \cdot \sum_{n=1}^{N} \left(2^{|\log_2(\tau(n)/\tau^{\mathrm{GT}}(n))|} - 1 \right).$$

Similarly, we define the standard deviation $\sigma(\tau, \tau^{\mathrm{GT}})$. For example, one obtains $\mu(\tau, \tau^{\mathrm{GT}}) = 100\%$ in the case $\tau = 2 \cdot \tau^{\mathrm{GT}}$ (double tempo) and in the case $\tau = \frac{1}{2} \cdot \tau^{\mathrm{GT}}$ (half tempo). Similarly, a computed tempo of 110 BPM or 90.9 BPM would imply a mean error of $\mu = 10\%$ assuming a ground-truth tempo of 100 BPM.

In a first experiment, we computed the curves τ_w^{FW} and τ_w^{FWR} with $w \propto 4$ seconds as well as τ_v^{AW} with $v = 10$ IOIs for each of the 15 pieces. Table 1 shows the mean error μ and standard deviation σ between the computed tempo curves and the ground truth tempo curves. For example, for the Schubert song *Der Lindenbaum* with identifier **C048**, the mean error between the computed tempo curve τ_w^{FW} and the ground-truth tempo τ^{GT} amounts to 2.61%. This error decreases to 2.07% when using the FWR-approach based on the rectified alignment path. Looking at the average mean error over all pieces, one can notice that the error amounts to 2.64% for the FW-approach, 4.40% for the AW-approach, and 1.98% for the FWR-approach. For example, assuming a tempo of 100 BPM, the last number implies a mean difference of less than 2 BPM between the computed tempo and the actual tempo.

In general, the FWR-approach yields the best tempo es-

w [sec]	FW		FWR		v [IOI]	AW	
	μ	σ	μ	σ		μ	σ
1	10.62	49.88	5.58	12.47	2	14.50	31.00
2	5.37	14.21	3.58	6.16	4	9.54	23.44
3	4.39	6.90	3.42	5.34	6	7.34	17.34
4	4.62	6.52	3.99	5.74	8	6.18	12.99
5	5.48	7.08	5.06	6.63	10	5.65	10.66
6	6.79	8.02	6.52	7.74	12	5.46	9.48
7	8.40	9.19	8.22	9.00	16	5.54	8.20
8	10.15	10.51	10.03	10.38	20	5.98	8.09

Table 2. Tempo curve evaluation using the approaches FW, AW, and FWR with various window sizes w (given in seconds) and v (given in IOIs). The table shows the average values over all 15 pieces, see Table 1. For generating the ground-truth tempo curves, MIDI segments of 5 seconds were used.

timation, whereas the AW-approach often produces poorer results. Even though the onset information is of crucial importance for estimating local tempo nuances, the AW-approach relies on accurate alignment paths that correctly align the note onsets. Synchronization approaches as described in [12] can produce highly accurate alignments in the case of music with pronounced note attacks. For example, this is the case for piano music. In contrast, such information is often missing in string or general orchestral music. This is the reason why the purely onset-based AW-strategy yields a relatively poor tempo estimation with a mean error of 10.58% for Beethoven's Fifth Symphony (identifier **C003**). On the other hand, using a fixed window size without relying on onset information, local alignment errors cancel each other out, which results in better tempo estimations. E. g., the error drops to 3.56% for Beethoven's Fifth Symphony when using the FWR-approach.

Finally, we investigated the dependency of the accuracy of the tempo estimation on the window size. We generated strongly fluctuating ground-truth tempo curves using MIDI segments of only 5 seconds length (instead of 10 seconds as in the last experiment). For the corresponding synthesized audio files, we computed tempo curves for various window sizes. The mean errors averaged over all 15 pieces are shown in Table 2. The numbers show that the mean error is minimized when using medium-sized windows. E. g., in the FWR-approach, the smallest error of 3.42% is attained for a window size of $w \propto 3$ seconds. Actually, the window size constitutes a trade-off between robustness and temporal resolution. On the one hand, using a larger window, possible alignment errors cancel each other out, thus resulting in a gain of robustness. On the other hand, sudden tempo changes and fine agogic nuances can be recovered more accurately when using a smaller window.

5. CONCLUSIONS

In this paper, we have introduced automated methods for extracting tempo curves from expressive music recordings by comparing the performances with neutral reference representations. In particular when using a combined strategy that incorporates note onset information, we obtain accurate and robust estimations of the overall tempo progression. Here, the window size constitutes a delicate trade-off between susceptibility to alignment errors and sensibility towards timing nuances of the performance. In practice, it becomes a difficult problem to determine whether a given change in the tempo curve is due to an alignment error or whether it is the result of an actual tempo change in the performance. Here, one idea for future work is to use tempo curves as a means for revealing problematic passages in the music representations where synchronization errors may have occurred with high probability. Furthermore, it is of crucial importance to further improve the temporal accuracy of synchronization strategies. This constitutes a challenging research problem in particular for music with less pronounced onset information, smooth note transitions, and rhythmic fluctuation.

Acknowledgements: The first three authors are supported by the Cluster of Excellence on *Multimodal Computing and Interaction* at Saarland University. The last author is funded by the German Research Foundation (DFG CL 64/6-1).

6. REFERENCES

[1] J. Langner and W. Goebl, "Visualizing expressive performance in tempo-loudness space," *Computer Music Journal*, vol. 27(4), pp. 69–83, 2003.

[2] C. S. Sapp, "Comparative analysis of multiple musical performances," in *ISMIR Proceedings*, pp. 497–500, 2007.

[3] G. Widmer, "Machine discoveries: A few simple, robust local expression principles," *Journal of New Music Research*, vol. 31(1), pp. 37–50, 2002.

[4] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic, "In search of the Horowitz factor," *AI Magazine*, vol. 24(3), pp. 111–130, 2003.

[5] Sonic Visualiser. Retrieved 19.03.2009, `http://www.sonicvisualiser.org/`.

[6] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, pp. 39–58, 2001.

[7] H. Honing, "From Time to Time: The Representation of Timing and Tempo," *Computer Music Journal*, vol. 25(3), pp. 50–61, 2001.

[8] E. D. Scheirer, "Tempo and beat analysis of acoustical musical signals," *Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.

[9] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A Tutorial on Onset Detection in Music Signals," *IEEE Trans. on Speech and Audio Proc.*, vol. 13, no. 5, pp. 1035–1047, 2005.

[10] N. Hu, R. Dannenberg, and G. Tzanetakis, "Polyphonic audio matching and alignment for music retrieval," in *Proc. IEEE WASPAA, New Paltz, NY*, October 2003.

[11] M. Müller, *Information Retrieval for Music and Motion*. Springer, 2007.

[12] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Taipei, Taiwan), 2009.

[13] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases," in *ISMIR*, 2002.

PROBABILISTIC SEGMENTATION AND LABELING OF ETHNOMUSICOLOGICAL FIELD RECORDINGS

Matija Marolt
Faculty of Computer and Information Science
University of Ljubljana, Slovenia
matija.marolt@fri.uni-lj.si

ABSTRACT

The paper presents a method for segmentation and labeling of ethnomusicological field recordings. Field recordings are integral documents of folk music performances and typically contain interviews with performers intertwined with actual performances. As these are live recordings of amateur folk musicians, they may contain interruptions, false starts, environmental noises or other interfering factors. Our goal was to design a robust algorithm that would approximate manual segmentation of field recordings. First, short audio fragments are classified into one of the following categories: speech, solo singing, choir singing, instrumental or bell chiming performance. Then, a set of candidate segment boundaries is obtained by observing how the energy of the signal and its content change, and finally the recording is segmented with a probabilistic model that maximizes the posterior probability of segments given a set of candidate segment boundaries with their probabilities and prior knowledge of lengths of segments belonging to different categories. Evaluation of the algorithm on a set of field recordings from the Ehtnomuse archive is presented.

1. INTRODUCTION

Ethnomusicological field recordings are recordings made “in the field”, capturing music in its natural habitat. Starting in the early 20th century and continuing to the present day, ethnomusicologists and folklorists have travelled and made recordings in various parts of the world primarily to preserve folk music, but also to make it available for further researches, such as studies of acculturation and change in music, comparative studies of music cultures and studies of the music making process and its effect through performance. Segmentation of field recordings into meaningful units, such as speech, sung or instrumental parts is one of the first tasks researchers face when a recording is first being studied. It is also a prerequisite for further automatic processing, such as extraction of keywords, melodies and other semantic descriptors.

Segmentation of audio recordings has been extensively explored for applications such as speech recognition (removal of non-speech parts, speaker change detection), segmentation in broadcast news or broadcast monitoring. Typically, the distinction is made between speech, music and silence regions. Approaches to segmentation include either first classifying short periods of the signal into desired classes using some set of features and then making the segmentation [1-3], or first finding change points in features and forming segments and later classifying the segments [4-6]. Authors use a variety of features, classifiers and distances depending on the nature of signals to be segmented. More recently, Ajmera [7] performed classification and segmentation jointly by using a combination of standard hidden Markov models and multilayer perceptrons for speech/music discrimination of broadcast news. Pikrakis et al. [8] used a three step approach: first they identified regions in the signal which are very likely to contain speech or music with a region growing algorithm. Then, they segmented the remaining short (few seconds long) regions with a maximum likelihood model that maximized the probability of class labels given frame-level features and segment length limits. A Bayesian network was used to estimate the posterior probability of a music/speech class label given a set of features. Finally, a boundary correction algorithm was applied to improve the found boundaries. Their use of a probabilistic model is somewhat similar to the proposed segmentation method, but as we describe further on, we use a maximum likelihood approach to segment an entire field recording by first labeling signal fragments, then finding candidate boundaries, and finally maximizing the probability of segmentation considering probabilities of boundaries and segment lengths given their class.

The algorithm presented in this paper was designed to robustly label and segment ethnomusicological field recordings into consistent units, such as speech, sung and instrumental parts. Resulting segmentations should be comparable to manual segmentations researchers make when studying recordings. Field recordings are documents of entire recording sessions and typically contain interviews with performers intertwined with actual performances. As these are live recordings of amateur folk musicians, they usually contain lots of “noise” and interruptions, such as silence when performers momentarily forget parts of songs, false starts and restarts, laughter, dancing noises, interruptions by other persons, dogs barking or cars driving by. Performances may also change character; singing may become reciting, a second voice may join or drop out of a performance etc.

The described nature of field recordings calls for a robust segmentation algorithm that would not over-segment a recording at each interruption – for example; we are not interested in each boundary separating speech and sung parts, as only some of them are actual segment boundaries. We would also like to distinguish between several different classes of segments and would like to take some prior knowledge of the classes into account. And last, we are not interested in millisecond-exact segment boundaries or exact labeling of each small recording fragment; sometimes placing a boundary between two performances is a very soft decision and accuracy of a few seconds is good enough. Taking these points into account, we propose a three step algorithm for segmentation. First, a standard classification algorithm is used to classify short audio segments into a set of predefined classes. Then, a set of candidate segment boundaries is obtained by observing how the energy and class distribution change, and finally the recording is segmented with a probabilistic model that maximizes the posterior probability of segments given a set of candidate segment boundaries with their probabilities and prior knowledge of lengths of segments belonging to different classes.

2. CLASSIFICATION

Classification of short field recording fragments into a set of predefined categories represents the first part of our segmentation algorithm. We base our work on field recordings from the EthnoMuse digital archive [9]. The archive contains folk song, music and dance collections of the Institute of Ethnomusicology, Scientific Research Centre of Slovene Academy of Sciences and Arts. Audio recordings represent the largest part of the archive and comprise recordings of folk songs and melodies, with the oldest on wax cylinders from 1914 and around 30.000 field recordings on magnetic tape and digital media dating from 1955 onwards. Only parts of the archive are digitally annotated. Field recordings are typically around an hour long and contain interviews with performers intertwined with performances. The latter include singing (solo or group), reciting, instrumental pieces (a large variety of instruments is used, depending on the region), as well as bell chiming, which is a Slovenian folk tradition of playing rhythmic patterns on church bells. The quality of recordings varies a lot and depends on their age, equipment used, location (inside, outside) and type of event (arranged recording session or recording of a public event).

We identified five categories into which field recording fragments are to be classified: speech, solo singing, choir singing (any performance with two or more voices belongs to this class), instrumental (including instrumental with singing) and bell chiming. We then evaluated a set of features often used for speech/music discrimination and timbre recognition to find the ones most suitable for classification into these categories. The following nine features were selected:

- the quotient of RMS energy variance over the squared mean of RMS energy. RMS energy r is defined as:
$$r = \sqrt{\frac{1}{W}\sum_{i=0}^{W-1} x_i^2}\ , \tag{1}$$
where x represents the time domain signal and W the window size. The feature describes the amount of signal energy fluctuations and is typically larger for speech than for other types of signals;
- mean spectral entropy, as defined by Pikrakis [10]. The entropy represents the instability of signal energy calculated over a number of spectral sub-bands and is typically low for bell chiming recordings, somewhat higher for music, and high for other signal types. It is calculated as:
$$H = -\sum_{i=0}^{L-1} \frac{X_i}{\sum_{j=0}^{L-1} X_j} \log_2 \frac{X_i}{\sum_{j=0}^{L-1} X_j}\ , \tag{2}$$
where L represents the number of spectral sub-bands and X_i the energy of the i-th sub-band (see [10] for more details);
- variance of spectral entropy deltas. Deltas are calculated as a linear trend over five consecutive windows;
- variances of the first three MFCC coefficients (omitting the zero-*th*). MFCC coefficients describe the shape of the signal spectrum and are thus very appropriate for our classification task;
- variances of deltas of the first three MFCC coefficients (omitting the zero-*th*). Deltas are calculated as a linear trend over five consecutive windows.

To train and test a classifier, we manually labeled 1760 3 second long field recording fragments from the EthnoMuse digital archive. All features were calculated on signals windowed with a 46ms Hamming window with 23ms overlap. Feature means and variances were calculated over 3 second periods, thus taking approx. 130 feature values into account. A multinomial logistic regression classifier [11] was chosen for classification, because it's simple and gives good results. Furthermore, its output can be regarded as a probability distribution over all classes. We trained the classifier to classify each fragment into one of the five previously described classes. 2/3 of the labeled fragments were used for training and 1/3 for testing. Table 1 shows the average confusion matrix of our classifier for 10 training/test runs. The overall accuracy is at 78% of correctly classified instances.

Most of the errors made by the classification algorithm are easy to explain. The confusion of speech and solo singing segments is understandable, if we take into account that singers are not professional musicians, they are often old persons and their singing close to reciting or very monotonous. Confusion between solo and choir singing occurs in choir segments sung in unison, as well as duet singing, while instrumental and bell chiming segments are correctly classified in most cases with confusion mostly arising between the two classes.

	classified as				
	speech	solo	choir	instr.	bell ch.
speech	79%	14%	4%	3%	0%
solo singing	13%	61%	24%	1%	1%
choir singing	2%	10%	82%	3%	3%
instrumental	1%	3%	3%	82%	11%
bell chiming	0%	0%	2%	7%	91%

Table 1. Confusion matrix of the classification algorithm.

3. SEGMENTATION

To segment a recording, we first find a set of candidate segment boundaries and calculate the probability of splitting the recording at each boundary. Segmentation is then performed by maximizing the joint probability of all segments, taking prior knowledge of segment lengths of different signal classes into account.

3.1 Finding and Evaluating Candidate Boundaries

We consider two criteria for boundary placement: a criterion based on change in signal energy, such as when performances are separated by regions of silence, and a criterion based on change in signal content, such as when speech is followed by singing. To observe changes in energy, we calculate RMS energy *e* of the audio signal; changes in signal content are detected by calculating the symmetric Kullback-Leibler (KL) divergence *d* [12] between probabilities of signal classes as calculated by the logistic classifier described in section 2. We find a set of candidate segment boundaries $\mathfrak{B}$ by low-pass filtering both measures to obtain their filtered versions e^f and d^f and finding all candidate boundary regions (b_l, b_r) that satisfy:

$$\mathfrak{B} = \left\{ (b_l, b_r) \mid \forall t \in [b_l, b_r] : \begin{bmatrix} e_t < \max\left(e_t^f E_1, E_0\right) \text{ or} \\ d_t > \max\left(d_t^f + D_1, D_0\right) \end{bmatrix} \right\}, \quad (3)$$

where E_0 and E_1 are the global and relative thresholds that determine the selection of energy-based candidate boundary regions and D_0 and D_1 the global and relative thresholds that determine the selection of divergence-based candidate boundary regions (see also Figure 1 for illustration).

Thus, the set of all candidate segment boundaries contains regions of the signal where its energy falls below, or the amount of change in signal content rises above an adaptive threshold. This is illustrated in Figure 1, which displays a 13 minute long field recording excerpt. The overall RMS energy *e* (in dB) is displayed on top, the symmetric KL divergence *d* below. Both adaptive thresholds are indicated with a dotted line; regions where the curves fall below (energy) or raise above (KL divergence) the threshold represent candidate segment boundaries. True segment boundaries are indicated in the middle. As shown, the candidate boundary regions correspond well with true boundaries. Many segments are clearly separated by regions of silence, as the energy plot shows. On the other hand, KL divergence is high where signal content changes, such as between speech and instrumental or sung parts.

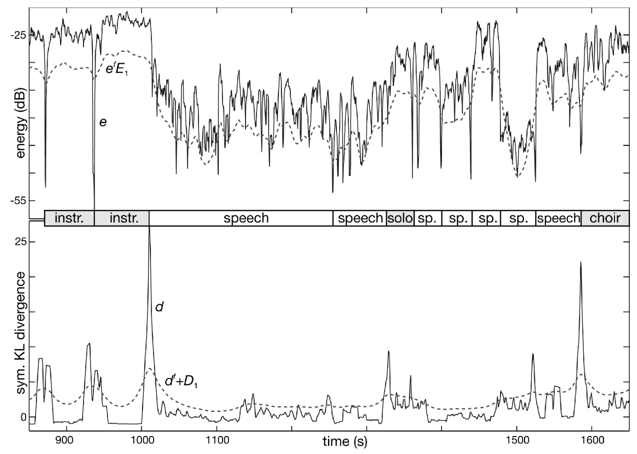


Figure 1. Finding candidate boundaries.

Selecting all of the candidate boundary regions as true boundaries and splitting a recording accordingly is not the best idea; for example energy fluctuates a lot in speech parts (as can be seen in Figure 1) and these parts would consequently be over-segmented. One could attempt to find the best values for relative thresholds D_1 and E_1, but as we show, we can do better by treating the boundary selection process as a classification task. For this purpose, we trained two logistic regression classifiers (one for energy, one for KL divergence) to predict the probability of splitting the segment at a candidate boundary.

The following features were found to be useful for energy-based boundary classification: the amount of signal energy below the energy threshold (s_e) and the maximum difference in signal content to the left and right of the boundary region (m_c). They are calculated as:

$$\begin{aligned} s_e &= \sqrt{\sum_{t=b_l}^{b_r} \max(e_t^f E_1, E_0) - e_t} \\ m_c &= \frac{1}{N+1} \max_c \left| \sum_{t=b_l-N}^{b_l} P(c_t = c) - \sum_{t=b_r}^{b_r+N} P(c_t = c) \right| \end{aligned}, \quad (4)$$

where $P(c_t = c)$ denotes the probability that the signal at time *t* belongs to class *c*, as calculated by the classification algorithm presented in section 2 and *N* the number of frames taken into account to the left or right of the boundary region. The most useful features for the KL divergence-based classifier were found to be the amount of divergence above the threshold (s_d) and the total amount of divergence within the boundary region (t_d):

$$s_d = \log\left(1+\sum_{t=b_l}^{b_r} d_t - \max(d_t^f + D_1, D_0)\right)$$
$$t_d = \log\left(1+\sum_{t=b_l}^{b_r} d_t\right) \quad . \tag{5}$$

Both classifiers were trained and tested on a set of 30 field recordings from the Ethnomuse archive, which were manually segmented and labeled, containing a total of 840 segments. The classifiers were trained to predict whether a found candidate boundary represents a true segment boundary or not. RMS energy e_t was calculated as the average RMS energy within a 3s window around t and a step size of 0.5s. Symmetric KL divergence d_t was calculated between 10 second long segments to the left and right of t with the same step size. Such large window sizes were chosen primarily to make the algorithm more robust to "noise" in performances, such as false starts, performers forgetting songs, interruptions etc. To obtain the smoothed vectors e^f and d^f, we zero-phase filtered e and d with a first order low-pass Butterworth filter with cutoff frequency of 0.01π. The values of other parameters were experimentally obtained and set to: E_1=0.2, E_0=10^{-6}, D_1=0.1, D_0=3 and N=9. Using these parameters, we extracted approximately 2400 candidate boundary regions from the field recordings and used two thirds of this set to train each classifier to predict whether a candidate boundary is a true segment boundary or not. We evaluated the performance of the two classifiers on the remaining third of the dataset and compared it to an alternative of using an optimal fixed threshold for candidate selection. Table 2 displays average precision and recall scores on the test set for 10 training/test runs. Compared to choosing a fixed threshold for boundary selection, logistic classifiers improve the accuracy of selection. An additional advantage is that their output can be regarded as the probability of splitting the recording at a candidate boundary; a fact exploited by our segmentation algorithm described in section 3.2.

criterion	select. method	precision	recall
energy	best fixed threshold	0.71	0.57
	logistic classifier	0.7	0.67
KL	best fixed threshold	0.77	0.71
divergence	logistic classifier	0.79	0.78

Table 2. Selection of boundary candidates.

3.2 Segmentation algorithm

We perform segmentation by following the logic of Bayesian modeling and infer the most probable segmentation by maximizing:

$$P(seg \mid data) \propto P(data \mid seg)P(seg) \tag{6}$$

To obtain a generative segmentation model, we define segmentation as a sequence of segments S_{i1}, S_{i2}, ..., S_{iN}, $0<i1<i2< \ldots <iN$, where S_{i1} starts at time 0 and ends at candidate boundary B_{i1}, S_{i2} starts at candidate boundary B_{i1} and ends at B_{i2}, S_{i3} starts at B_{i2} and ends at B_{i3} and so on. We treat each candidate boundary $B_t \in \mathfrak{B}$ as a discrete random variable with two outcomes: either the candidate boundary represents an actual boundary and splits the recording into two segments, or not. The probability mass function for the variable is defined by outputs of the energy (P_e) and KL divergence (P_{kl}) classifiers, as described in section 3.1:

$$P(B_t = true) = \max\left(P_e(B_t), P_{kl}(B_t)\right) \tag{7}$$

In our model, the probability of each segment is only dependent on location of the previous segment, so we can express the joint probability of all segments as:

$$P(S_{i1})P(S_{i2} \mid S_{i1})P(S_{i3} \mid S_{i2}) \cdot \ldots \cdot P(S_{iN} \mid S_{iN-1}) \, . \tag{8}$$

To calculate the probability of segment S_i given S_j, we must consider all candidate boundaries within the segment, as well as its duration. If the segment is to start at time j and end at i, values of all candidate boundary variables within the segment must be *false*, while the value of candidate boundary variable at time i must be *true*. Segmentation is further constrained by our previous knowledge of typical lengths of segments given their class, leading to the following formulation:

$$P(S_i \mid S_j) = P(D_i \mid S_i, S_j)P(B_i = true)\prod_{j<k<i} P(B_k = false) \, . \tag{9}$$

Equation (8) then becomes:

$$\prod_{(i,j)\in\mathfrak{S}} \left(P(B_i = true)P(D_i \mid S_i, S_j)\right) \times \prod_{j\notin\mathfrak{S}} P(B_j = false) \, , \tag{10}$$

where $\mathfrak{S}$ is the set of all segment indices and (i,j) a pair of consecutive indices from this set.

Probability of segment duration given its boundaries is dependent on the class of the segment, as calculated by the classifier presented in section 2. By analyzing durations of segments in our collection of field recordings, we estimated the means and standard deviations for all segment classes (μ_c, σ_c); for example the duration of speech segments varies a lot and ranges from several seconds to over ten minutes, while the average length of choir singing segments is around three minutes and their standard deviation below two minutes. By additionally enforcing minimal segment duration D_{min}, we obtain the following expression:

$$P(D_i \mid S_i, S_j) = \begin{cases} \sum_c P(C_i = c \mid S_i, S_j)\, G(i-j, \mu_c, \sigma_c) \\ 0, \; i-j < D_{min} \end{cases} \, , \tag{11}$$

where $P(C_i=c|S_i, S_j)$ represents the probability that segment S_i belongs to class c and is calculated as the average probability of classification of frames within the segment into class c. G is the unscaled Gaussian function.

To find the sequence of segments that maximizes Equation (10) and thus provides an optimal solution, we resort to dynamic programming that leads us to a simple and efficient solution. For each segment S_i ending at the candidate boundary B_i we can calculate the most probable segmentation that ends with this boundary $d(S_i)$ by the following rules:

$$d(S_i) = \begin{cases} 0.5 & i = 0 \\ P(B_i = true) \max\limits_{0<=j<i} \left(d(S_j)\, c(i,j) \right) & i > 0 \end{cases} , \quad (12)$$
$$c(i,j) = P(D_i \mid S_i, S_j) \prod_{j<k<i} P(B_k = false)$$

where S_0 represents the segment boundary at time 0; S_0 is a boundary if a performance starts at time 0, or not if there is silence or noise present, so we give it a probability of 0.5.

In our implementation, we minimize the negative log-likelihood of segmentation, so all products become summations. When the function $d(S_i)$ is calculated for all candidate boundaries, the most likely segmentation can be recovered by tracking back the calculation and retrieving optimal boundary indices.

After segmentation is calculated, segments can be labeled by finding the class c that maximizes $P(C_i=c|S_i, S_j)$; as mentioned before, the latter and is calculated as the average probability of classification of frames within the segment S_i into class c.

3.3 Evaluation

As with boundary selection, we evaluated our segmentation algorithm on a set of 30 field recordings from the Ethnomuse archive, which were manually segmented and labeled, containing a total of 840 segments. Because of its specific nature, it is difficult to directly compare the algorithm to other segmentation approaches. We therefore provide a comparison of the proposed method to a simple thresholding algorithm, where segments are formed by thresholding either the energy, KL divergence or the maximal energy/KL divergence candidate boundary probabilities. Results are given in Table 3. Average precision and recall scores of true vs. estimated segment boundaries for all 30 recordings for the three thresholding and the proposed probabilistic method are shown.

	average precision	average recall
thresholding $P_e(B_t)$	0.61	0.61
thresholding $P_d(B_t)$	0.65	0.64
thresholding $\max(P_e(B_t),P_d(B_t))$	0.73	0.78
proposed algorithm	0.78	0.81

Table 3. Comparison of segmentation algorithms.

The probabilistic algorithm is quite robust and improves segmentation accuracy over the more naive thresholding approaches. Most of the false positives occur in speech sections containing very long regions of silence that for example occur when people reflect on past events (consequently causing large drops in energy), or in solo singing performances that are interleaved with reciting or spoken statements, such as “this is repeated three times and we start dancing in a circle so and so ...” (causing high KL divergence). False negatives occur when performances follow each other without significant changes, for example several songs sung in a row almost without interruptions, or when the start or end of a segment is missed, because it interleaves with speech, so that the boundary is placed either too soon or too late in a recording.

To evaluate the influence of the choice of relative and global thresholds (see eq. (3)) on segmentation, we evaluated the algorithm’s performance by varying values of the four thresholds individually, with other parameters fixed. The resulting precision/recall curves are given in Figure 2.

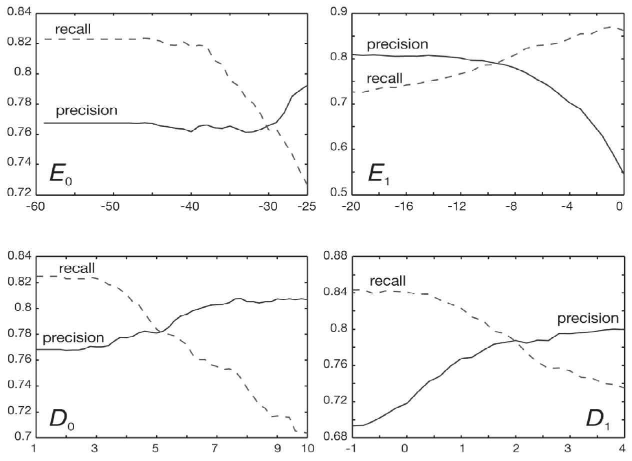


Figure 2. Precision/recall curves obtained by varying the four thresholds that influence candidate boundary region selection: E_0 and E_1 for energy (both are shown in dB), D_0 and D_1 for KL divergence curves.

We can observe that precision is only marginally affected by both global thresholds (E_0 and D_0) – raising them will result in a smaller number of boundaries found, thus decreasing recall, while precision will not increase by much, as the false positives seem to be almost equally spread between weak (low global threshold) and strong (high global threshold) candidate boundary regions. On the other hand, precision is more strongly affected by relative threshold selection (E_1 and D_1); small relative threshold values will result in many false positives, as any significant drop in energy or rise in the KL divergence curve will result in a new boundary candidate. Higher values increase precision and decrease recall, as expected.

The accuracy of classification of correctly found segments into one of the five classes is 86%; errors are similar to the ones described in section 2.

4. CONCLUSION

The proposed algorithm for segmentation and labeling of ethnomusicological field recordings provides a good starting point for further development of automatic methods for analysis of such recordings. Its accuracy is good enough for practical use and the algorithm has already been integrated into the tools of the Ethnomuse archive and is available to its users. For further improvements, we need to start looking into the inner structure of each segment, which may help us to improve the found boundaries. We also plan to explore hierarchical segment classification to classify instrumental segments into typical ensemble types, speech and singing segments into male and female etc.

Acknowledgments. This work was supported in part by the Slovenian Government-Founded R&D project EthnoCatalogue: creating semantic descriptions of Slovene folk song and music.

5. REFERENCES

[1] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997, pp. 1331-1334 vol.2.

[2] L. Lie*, et al.*, "Content-based audio segmentation using support vector machines," in *IEEE International Conference on Multimedia and Expo,* 2001, pp. 749-752.

[3] G. Williams and D. P. W. Ellis, "Speech/music Discrimination Based On Posterior Probability Features," in *Eurospeech'99*, Budapest, Hungary, 1999, pp. II-687-690.

[4] G. Tzanetakis and P. Cook, "Multifeature audio segmentation for browsing and annotation," in *Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop on*, 1999, pp. 103-106.

[5] C. Panagiotakis and G. Tziritas, "A speech/music discriminator based on RMS and zero-crossings," *Multimedia, IEEE Transactions on,* vol. 7, pp. 155-166, 2005.

[6] M. Cettolo*, et al.*, "Evaluation of BIC-based algorithms for audio segmentation," *Computer Speech & Language,* vol. 19, pp. 147-170, 2005.

[7] J. Ajmera, "Robust Audio Segmentation," Ph.D., Faculte des sciences et techniques de l'ingenieur, Ecole Polytechnique Federale de Lausanne, Lausanne, 2004.

[8] A. Pikrakis*, et al.*, "A Speech/Music Discriminator of Radio Recordings Based on Dynamic Programming and Bayesian Networks," *Multimedia, IEEE Transactions on,* vol. 10, pp. 846-857, 2008.

[9] M. Marolt*, et al.*, "Ethnomuse: Archiving Folk Music and Dance Culture," in *Eurocon 2009*, St. Petersburg, Russia, 2009.

[10] A. Pikrakis*, et al.*, "A computationally efficient speech/music discriminator for radio recordings," in *ISMIR 2006, 7th International Conference on Music Information Retrieval*, Victoria, Canada, 2006.

[11] I. H. Witten and E. Frank, *Data Mining*. San Francisco, USA: Morgan Kaufmann, 2005.

[12] W. D. Penny "Kullback-Liebler divergences of normal, gamma, Dirichlet and Wishart densities," *Technical report*, Wellcome Department of Cognitive Neurology, London, UK, 2001.

A MUSIC CLASSIFICATION METHOD BASED ON TIMBRAL FEATURES

Thibault Langlois
Faculdade de Ciências da Universidade de Lisboa
tl@di.fc.ul.pt

Gonçalo Marques
Instituto Superior de Engenharia de Lisboa
gmarques@isel.pt

ABSTRACT

This paper describes a method for music classification based solely on the audio contents of the music signal. More specifically, the audio signal is converted into a compact symbolic representation that retains timbral characteristics and accounts for the temporal structure of a music piece. Models that capture the temporal dependencies observed in the symbolic sequences of a set of music pieces are built using a statistical language modeling approach. The proposed method is evaluated on two classification tasks (Music Genre classification and Artist Identification) using publicly available datasets. Finally, a distance measure between music pieces is derived from the method and examples of playlists generated using this distance are given. The proposed method is compared with two alternative approaches which include the use of Hidden Markov Models and a classification scheme that ignores the temporal structure of the sequences of symbols. In both cases the proposed approach outperforms the alternatives.

1. INTRODUCTION

Techniques for managing audio music databases are essential to deal with the rapid growth of digital music distribution and the increasing size of personal music collections. The Music Information Retrieval (MIR) community is well aware that most of the tasks pertaining to audio database management are based on similarity measures between songs [1–4]. A measure of similarity can be used for organizing, browsing, visualizing large music collections. It is a valuable tool for tasks such as mood, genre or artist classification that also can be used in intelligent music recommendation and playlist generation systems.

The approaches found in the literature can roughly be divided in two categories: methods based on metadata and methods based on the analysis of the audio content of the songs. The methods based on metadata have the disadvantage of relying on manual annotation of the music contents which is an expensive and error prone process. Furthermore, these methods limit the range of songs that can be analyzed since they rely on textual information which may not exist. The other approach is based solely on the audio contents of music signals. This is a challenging task mainly due to the fact that there is no clear definition of similarity. Indeed, the notion of similarity as perceived by humans is hard to pinpoint and depends on a series of factors, some dependent on historical and cultural context, others related to perceptual characteristics of sound such as tempo, rhythm or voice qualities.

Various content-based methods for music similarity have been proposed in recent years. Most of them divide the audio signal in short overlapping frames (generally 10-100ms with 50% overlap), and extract a set of features usually related to the spectral representation of the frame. This approach converts each song into a sequence of feature vectors, with a rich dynamic structure. Nevertheless, most of the similarity estimation methods ignore the temporal contents of the music signal. The distribution of the features from one song or a group of songs are modeled, for instance, with the k-means algorithm [3], or with a Gaussian mixture model [1, 5, 6]. To measure similarity, models are compared in a number of ways, such as the Earth-Mover's distance [3], Monte-Carlo sampling [1], or nearest neighbor search. Additionally, some information about the time-dependencies of the audio signal can be incorporated through some statistics of the features over long temporal windows (usually a few seconds), like in [4–8].

In this work we propose computing a measure of similarity between songs based solely on timbral characteristics. We are aware that relying only on timbre to define a music similarity measure is controversial. Human perception of music similarity relies on a much more complex process, albeit timbre plays an important role in it. As pointed out by J.-J. Aucouturier and J. Pachet [1], methods that aim at describing a timbral quality of whole song will tend to find similar pieces that have similar timbres but belong to very different genres of music. For instance, pieces like a *Schumann* sonata or a *Bill Evans* tune will have a high degree of similarity due to their common romantic piano sounds [1]. Following our approach by modeling time dependencies between timbre-based feature vectors, we expect to include some rhythmic aspects in the models. As we will see in section 3.3, this approach leads to playlists with more variety while conserving the same overall mood.

We use a single type of low-level features: the Mel Frequency Cepstral Coefficients (MFCC). The MFCC vectors are commonly used in audio analysis and are described as timbral features because they model the short-time spectral characteristics of the signal onto a psychoacoustic frequency scale. On their own, the MFCC vectors do not explicitly capture the temporal aspects of the music, and therefore are often associated with the "bag of frames" classifiers. In this type of classifiers, songs with the same MFCC frames in different order would be yield the same results. It is our contention that the order of MFCC frames is indeed important and that this information can be used to estimate a similarity measure between songs. We use a language model approach to achieve this result. The most related works include Soltau *et al.* [9], Chen *et al.* [10], and Li and Sleep [11].

In Soltau *et al.* [9], each music is converted into a sequence of distinct music events. Statistics like unigram, bigram, trigram counts are concatenated to form a feature vector that is fed into a neural network for classification. In Chen et al. [10] a text categorization technique is proposed to perform musical genre classification. They build a HMM from the MFCC coefficients using the whole database. The set of symbols is represented by the states of the HMM. Music symbols are tokenized by computing 1 and 2-grams. The set of tokens is reduced using Latent Semantic Indexing. In Li and Sleep, a support vector machine is used as a classifier. The feature are based on n-grams of varying length obtained by a modified version of the Lempel-Ziv algorithm.

This paper is organized as follows: In section 2. we describe our method for music similarity estimation. In section 3. we report and analyze the results of the algorithm on various task and datasets. We also compare performance of our approach to other types of techniques. We close with some final conclusions and future work.

2. PROPOSED APPROACH

The proposed approach is divided into several steps. First, the music signals are converted into a sequence of MFCC vectors [1]. Then, the vectors are quantized using a hierarchical clustering approach. The resulting clusters can be interpreted as codewords in a dictionary. Every song is converted into a sequence of dictionary codewords. Probabilistic models are then built based on codeword transitions of the training data for each music category, and for classification, the model that best fits a given sequence is chosen. The details of each stage are described in the following sections. In the last section we consider building models based on a single music piece, and describe an approach that allows us to define a distance between two music pieces.

2.1 Two-Stage Clustering

The objective of the first step of our algorithm is to identify, for each song, a set of the most representative frames. For each track, the distribution of MFCC vectors is estimated with a gaussian mixture model (GMM) with five gaussians and full covariance matrix (Λ_i):

$$\mathrm{pdf}(f) = \sum_{i=1}^{N} w_i G_i(f) \tag{1}$$

with:

$$G_i(f) = \frac{1}{\sqrt{(2\pi)^d |\Lambda_i|}} \exp\left(-\frac{1}{2}(f-\mu_i)\Lambda_i^{-1}(f-\mu_i)^\top\right) \tag{2}$$

where μ_i represent the Gaussian's mean and f an MFCC frame. We did not perform exhaustive tests in order to chose the optimal value for the number of Gaussians (N) but realized some tests on a reduced number of tracks and decided to use $N = 5$. At this step, the use of GMM is similar to Aucouturier's work [12] were some hints are given about the optimal value of N. The parameters are estimated using the Expectation-Maximization (EM) algorithm. The probabilistic models of the songs are used to select a subset of the most likely MFCC frames in the song. For each track a, $\mathcal{F}_a$, is the set of k_1 frames that maximize the likelihood of the mixture.

Contrasting with Aucouturier's approach, we do not use the GMM as the representation of tracks in the database. This leads to an increased memory requirement during the training phase that is later reduced as we will see in the next section.

The second step consists in finding the most representative timbre vectors in the set of all music pieces. At this stage, the dataset correspond to the frames extracted from each song: $\mathcal{F} = \bigcup_j^{N_m} \mathcal{F}_j$ and the objective is to deduce k_2 vectors that represent this dataset. This is achieved using the k-means algorithm. As an alternative, a GMM trained on the set $\mathcal{F}$ was also used. But thanks to the robustness, scalability and computational effectiveness of the k-means algorithm, better results were obtained using this simpler approach. More precisely, the EM algorithm is sensible to parameters like the number of gaussians and the dimension and the number of data points, and can result in ill-conditioned solutions. That was verified in numerous cases, and we managed to train GMMs with only a reduced number of kernels that was too small for our objectives.

The output of this two-stage clustering procedure is a set of k_2 twelve-dimensional centroids that represent the timbres found in a set of music pieces. The value of the k_1 parameter must be chosen in order to balance between precision [2], computing and space resources. One of the advantages of dividing into two steps is scalability. Indeed, the first stage has to be done only once and, as we will see in section 3. can be used to compute various kinds of models.

2.2 Language Model Estimation

The set of k_2 vectors obtained during the previous step is used to form a "dictionary" that allow us to transform a track into a sequence of symbols. For each MFCC frame f a symbol s corresponding to the nearest centroid c_i is assigned:

$$s = \operatorname*{argmin}_{i=1..k_2} d(f, c_i)$$

[1] Twelve Mel Frequency Cepstral Coefficients are calculated for each frame, all audio files were sampled at 22050Hz, mono and each frame has a duration of 93ms with 50% overlap

[2] We expect that higher values of k_1 parameter will lead to a more accurate description of the set of timbres present in a song.

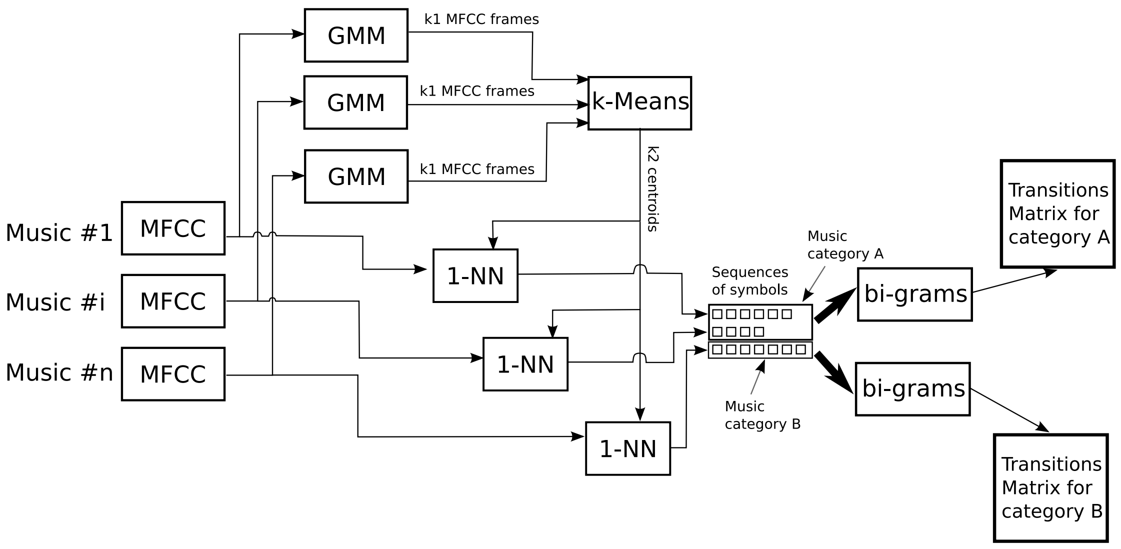


Figure 1. System structure for the language modeling approach. The music signals are converted into a sequence of MFCC vectors, and a two-stage clustering is performed on all the training sequences. Then all the MFFCs are vector quantized resulting in a sequences of symbols. The sequences are divided by category, and the bigrams probabilities are estimated.

where $d()$ is the Euclidian distance. Once tracks are transformed into sequences of symbols, a language modeling approach is used to build classifiers. A Markov Model is built for each category by computing the transition probabilities (bigrams) for each set of sequences. The result is a probability transition matrix for each category containing, for each pair of symbols (s_i, s_j), the probability $P(s_j|s_i)$ of symbol s_i to be followed by the symbol s_j.

This matrix cannot be used like this because it contains many zero-frequency transitions. Many solutions to this problem have been studied by the Natural Language Processing community. Collectively known as "smoothing" the solution consist in assigning a small probability mass to each unseen event in the training set. In the context of this work we experimented several approaches such as the Expected Likelihood Estimator and the Good-Turing estimator [13]. Neither of these approaches are suitable for our case, because the size of our vocabularies is much smaller than those commonly used in Natural Language Processing. We used a technique inspired by the "add one" strategy that consists in adding one to the counts of events. After some tests, we concluded that adding a small constant $\epsilon = 1.0e-5$ to each zero probability transition allowed us to solve the smoothing problem without adding to much bias toward unseen events.

Once a set of models is built, we are ready to classify new tracks into one of the categories. A new track is first transformed into a sequence of symbols (as explained above). Given a model M, the probability that it would generate the sequence $S = s_1, s_2, ...s_n$ is:

$$P_M(s_{i=1..n}) = P_M(s_1)\prod_{i=2}^{n} P_M(s_i|s_{i-1}) \tag{3}$$

which is better calculated as

$$\begin{aligned} S_M(s_{i=1..n}) &= \log(P_M(s_{i=1..n})) \\ &= \log(P_M(s_1)) + \sum_{i=2}^{n} \log(P_M(s_i|s_{i-1})) \end{aligned} \tag{4}$$

This score is computed for each model M and the class corresponding to the model that maximize the score values is assigned to the sequence of symbols. One of the benefits of our method is that once the models are computed, there is no need to have access to the audio files and MFCC features since only the sequences of symbols are used. With vocabulary size between 200 and 300 symbols the space needed to keep this symbolic representation is roughly one byte/frame or 1200 bytes/minute.

2.3 Distance Between Music Pieces

Given a database of music tracks, a vocabulary is build following the steps described in section 2.1. Then, instead of creating a model for each "class" or "genre" a model is built for each track (i.e. a probability transtion matrix). Let $S_a(b)$ be the score of music b given the model of music a (see section 2.2). We can define a distance between music a and music b by:

$$d(a,b) = S_a(a) + S_b(b) - S_a(b) - S_b(a) \tag{5}$$

This distance is symmetric but it is not a metric distance since $d(a,b) = 0 \Rightarrow a = b$ is not verified. It is a difficult task to evaluate a distance between music pieces since there is no "ground truth". One can examine the neighborhood of a song and verify to what extend the songs found nearby show similarities. In our case, the expected similarities should be relative to timbral characteristics since we are using features that represent the timbre. A common application of distances measures over music pieces is to generate playlists. The user selects a song he likes (the

	C	E	J	M	R	W	%acc.	pre.	rec.
Classical	304	2	0	0	0	14	95.0	0.95	0.95
Electronic	1	96	0	0	10	7	84.2	0.74	0.84
JazzBlues	0	2	16	0	6	2	61.5	1.00	0.62
MetalPunk	0	1	0	24	18	2	53.3	0.89	0.53
RockPop	1	13	0	3	78	7	77.5	0.63	0.77
World	17	15	0	0	12	78	63.9	0.72	0.64

Table 1. Confusion matrix, accuracy, precision and recall for each class of the ISMIR 2004 dataset.

seed song) and the system returns a list of similar songs from the database.

3. EXPERIMENTAL RESULTS AND ANALYSIS

3.1 Genre Classification task

We used the ISMIR 2004 genre classification dataset which is composed of six musical genres with a total of 729 songs for training and 729 songs for test [3] . The method described in sections 2.1 and 2.2 was used to classify this dataset. Table 1 shows the confusion matrix on the test set, classification rate, precision and recall for each class, obtained using parameters $k_1 = 200$ and $k_2 = 300$. The overall accuracy is 81.85% if we weight percentages with the prior probability of each class. These results compare favorably with those obtained with other approaches (see for example [5], 78.78% and [14], 81.71%). As can be seen in the following table, the method is not too sensible to its parameters (k_1 and k_2).

k_1	k_2	accuracy	k_1	k_2	accuracy
100	25	74.90%	200	200	81.07%
200	50	77.37%	200	300	81.89%
100	50	79.70%	200	400	81.48%
100	100	80.93%	300	300	81.76%
100	200	81.34%	300	400	81.07%
100	300	81.76%	300	1000	80.52%

3.2 Artist Identification task

One of our objectives with this task is to assess the performance of our method when models are based on smaller datasets. Indeed, contrasting with genre classification, in the case of Artist Identification, a model is build for each artist. We evaluated our method using two datasets: `artist20` [4] that contains 1412 tracks from 20 artists. Each artist is represented by 6 albums. The second dataset focus on Jazz music and is based on authors' collection. It contains 543 tracks from 17 artists (we will call this dataset `Jazz17`). This dataset is smaller than `artist20` but the interest here is to see if our system is able to distinguish songs that belong to a single genre. The abreviations used for the names of the 17 artists are: DK: Diana Krall, SV: Sarah Vaughan, DE: Duke Ellington, TM: Thelonious Monk, CB: Chet Baker, MD: Miles Davis, CJ: Clifford Jordan, NS: Nina Simone, JC: John Coltrane, FS: Frank Sinatra, LY: Lester Young, OP: Oscar Peterson, EF: Ella Fitzgerald, AD: Anita O'Day, BH: Billie Holliday, AT: Art Tatum and NJ: Norah Jones.

Regarding the `Jazz17` dataset, the results are shown in the following table. For two sets of parameter values (k_1 and k_2) the training and test was repeated ten times and the two last columns show the average accuracy and the corresponding standard deviation observed on the test set.

k_1	k_2	mean	std. dev.
100	100	73.49%	1.75
200	200	74.25%	2.25

Because of the reduced number of albums per artist, 50% of each artist's songs were randomly selected and for training while the other half was used for test. Table 2 contains a confusion matrix obtained with `Jazz17`. As can be seen in the confusion matrix, number of misclassifications occur between songs with strong vocals and are thus understandable.

The results obtained with the `artist20` dataset are shown in the following table. We used two different setups. For rows 1 and 2, 50% of an artist's songs are randomly selected and used for training while the other half is used for testing. In rows 3 and 4 we used the strategy suggested in [15]. For each artist an album is randomly selected for test and the other five albums are used for training.

	k_1	k_2	mean	std. dev.
1	100	100	57.40%	0.74
2	200	200	59.14%	1.49
3	100	200	45.28%	7.27
4	200	200	48.98%	7.96

The results shown in rows 3 and 4 are worse than those obtained by Dan Ellis [15] since his approach leads to 54% accuracy using MFCC features and 57% using MFCC and chroma features.

As we can see, choosing the training and testing sets randomly leads to significantly better results than keeping one album for test. This is due to the "album effect" [16]. These results show that despite the name of the task, it is clear that, at least in our case, the problem solved is not the Artist Identification problem. Indeed, our method aims at classifying songs using models based on timbre. Different albums of the same artist may have very different styles, use different kinds of instruments, sound effects and recording conditions. If a sample of each artist's style is found in the training set, it is more likely that the classifier will recognize a song with similar timbre. If every songs of an album are in the test set, then the accuracy will depend on how close are the mixtures of timbres of this album from those of the training set. This is confirmed by the standard deviation observed with both approaches. When trying to avoid the "album effect" we observe a large variation of performance due to the variation of the datasets. In one of our tests we reached an accuracy of 62.3% but this was due to a favorable combination of albums in the training and test sets.

[3] The distribution of songs along the six genres is: classical: 320; electronic: 115 jazzblues: 26; metalpunk: 45; rockpop: 101; world: 122 for the training and the test set.This data set was used for the Genre Classification contest organized in the context of the International Symposium on Music Information Retrieval - ISMIR 2004 (http://ismir2004.ismir.net).

[4] This dataset is available upon request, see: http://labrosa.ee.-columbia.edu/projects/artistid/ .

Notwithstanding these observations the results are interesting. In particular with the `Jazz17` dataset, we can see that the timbre-based classification is quite accurate even with music pieces that belong to the same genre.

3.3 Similarity Estimation task

The good results obtained for the classification of large sets of tracks (Genre classification) and more specific sets (Artist Identification) led us to consider building models based on a single track. In this section some examples of playlists generated using our distance are shown and discussed. From our Jazz music set (see section 3.2), we picked some well-known songs and generated a playlist of 20 most similar songs.

In the first example, the seed song is “Come Away With Me” by Norah Jones. The playlist, shown in table 3, is composed of songs where vocals are the dominant timbre. It is interesting to note that with one exception, the artists that appear in this list are all women. The timbre of Chet Baker’s voice is rather high and in sometimes may be confused with a women’s voice. However, John Coltrane’s “Village Blues” appears as an intruder in this list.

	Dist.	Artist	Song
0	0	N. Jones	Come Away with Me
1	4093	N. Jones	Come Away with Me (other version)
2	10774	D. Krall	Cry Me a River
3	11345	N. Jones	Feelin’ the Same Way
4	12212	D. Krall	Guess I’ll Hang My Tears Out To Dry
5	12333	J. Coltrane	Village blues
6	13015	D. Krall	Every Time We Say Goodbye
7	13201	D. Krall	The Night we Called it a Day
8	13210	N. Jones	Don’t Know Why
9	13401	D. Krall	I Remember You
10	13458	D. Krall	Walk On By
11	13758	D. Krall	I’ve Grown Accustomed To Your Face
12	13852	S. Vaughan	Prelude to a Kiss
13	13915	D. Krall	Too Marvelous For Words
14	13969	D. Krall	The Boy from Ipanema
15	14099	N. Jones	Lonestar
16	14114	C. Baker	My Funny Valentine
17	14405	D. Krall	The Look of Love
18	14674	N. Jones	Lonestar (other version)
19	15039	D. Krall	Este Seu Olhar

Table 3. Playlist generated from “Come Away With Me”

The playlist generated starting with the seed song “Blue Train” by John Coltrane (Table 4) is characterized by Saxophone solos and trumpet. Excluding the songs from the same album, the songs found in the playlist are performed by Miles Davis, Dizzy Gillespie whose trumpets are assimilated with saxophone and Ella Fitzgerald and Frank Sinatra who are accompanied by a strong set of copper instruments.

3.4 Other Approaches

3.4.1 *Using unigrams and bigrams*

Our classification method is based on models of bigram probabilities whereas most of previous approaches rely on the classification of frame-based feature vectors or on estimates of statistical moments of those features computed on wider temporal windows. In order to quantify the benefit of taking into account transition probabilities an hybrid approach was implemented. With this approach, the classification of a sequence depends on a linear combination of unigrams and bigrams. If we consider only unigrams, the score of a sequence os symbols $s_{i=1..n}$ is:

$$S'_M(s_{i=1..n}) = \log\left(P_M(s_{i=1..n})\right) = \sum_{i=1}^{n} \log\left(P_M(s_i)\right)$$

	Dist.	Artist	Song
0	0	J. Coltrane	Blue Train
1	11367	J. Coltrane	Moment’s Notice
2	14422	J. Coltrane	Lazy Bird
3	17344	J. Coltrane	Locomotion
4	23418	E. Fitzgerald	It Ain’t Necessarily So
5	25006	E. Fitzgerald	I Got Plenty o’ Nuttin’
6	25818	F. Sinatra	I’ve Got You Under My Skin
7	27054	M. Davis	So What
8	27510	M. Davis	Freddie Freeloader
9	28230	E. Fitzgerald	Woman is a Sometime Thing
10	28598	S. Vaughan	Jim
11	28756	F. Sinatra	Pennies From Heaven
12	29204	D. Gillespie	November Afternoon
13	30299	M. Davis	Bess oh Where’s my Bess
14	31796	F. Sinatra	The Way You Look Tonight
15	31971	E. Fitzgerald	There’s a Boat Dat’s Leavin’ Soon for NY
16	32129	E. Fitzgerald	Dream A Little Dream of Me
17	32232	J. Coltrane	I’m Old Fashioned
18	32505	E. Fitzgerald	Basin’ Street Blues
19	34045	M. Davis	All Blues

Table 4. Playlist generated from “Blue Train”

Using the score computed for bigrams (see equation 4), a linear combination can be writtem as:

$$S''_M(s_{i=1..n}) = \alpha S'_M(s_{i=1..n}) + (1-\alpha) S_M(s_{i=1..n}) \quad (6)$$

where $\alpha \in [0,1]$. This approach was experimented on the ISMIR 2004 dataset. The results are shown in the following table:

α	1.0	0.5	0.0
accuracy	71.88%	77.64%	81.89%

When $\alpha = 1$, only unigrams are taken into account whereas $\alpha = 0$ reverts to the case where only bigrams are considered. As we can see in this table, the introduction of unigrams in the classification process in not beneficial. A closer look at unigram probabilities give an explaination to these observations. The following table show, for each class, the number of clusters were the class is most represented, the average probability (and standard deviation) of observing the class M given a symbol s, $(P(M|s_i))$.

	Cl.	El.	JB	MP	RP	Wo.
#C	73	68	4	9	16	30
$P(M\|s)$	0.599	0.503	0.578	0.423	0.409	0.471
std.dev.	0.194	0.162	0.180	0.063	0.103	0.139

One can see that for three classes this average probability is below 0.5 i.e. most symbols represents a mixture of timbres. This explains why unigram probabilities are not a good indicator of the class.

3.4.2 *Hidden Markov Models*

We implemented another technique commonly used to model time-varying processes, the Hidden Markov Models (HMMs). These models were tested on the genre classification task with the ISMIR 2004 genre dataset. The same (discrete) sequences used to train the language models were also used

	DK	SV	DE	TM	CB	MD	CJ	NS	JC	FS	LY	OP	EF	AD	BH	AT	NJ
DK	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
SV	0	9	0	0	1	0	0	0	0	0	0	1	1	5	0	0	0
DE	0	0	5	0	0	1	0	0	0	0	0	0	0	1	0	0	0
TM	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0
CB	0	2	0	0	20	1	1	1	0	2	0	0	0	0	0	0	0
MD	0	2	0	0	1	14	1	0	0	1	0	0	0	0	0	0	0
CJ	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0
NS	0	1	0	0	1	0	0	9	0	0	1	0	1	0	0	0	0
JC	0	0	0	0	1	2	0	0	2	0	0	0	1	0	0	0	0
FS	0	0	0	0	0	0	0	0	0	20	0	0	3	0	0	0	0
LY	0	1	0	0	4	0	0	0	0	0	11	2	1	1	1	0	0
OP	0	1	0	0	0	0	0	1	0	0	1	11	0	0	0	0	0
EF	0	4	0	0	0	0	0	0	0	0	2	0	9	2	0	0	0
AD	0	0	0	0	1	0	0	0	0	1	0	0	3	15	0	0	0
BH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0
AT	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	18	0
NJ	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16

Table 2. Confusion matrix obtained with the `Jazz17` dataset.

in the HMM's training. For classification, we calculated the probabilities of a given sequence with the HMM's trained for different genres, and assigned the music to the genre with the highest probability.

We used left-right models with 2, 3 and 4 delays, and a fully connected model. We also tested these models with 10 and 20 hidden states. The results, shown in the following table, indicate that the performance of the HMMs is worse than our method. Nevertheless, it should be noted that in our approach, we need a significant number of states (between 100 and 400) in order to achieve reasonable accuracy in timbre modeling. To train an HMM with such a number of hidden states would require a huge amount of data in order for the model to converge.

HMM	LR-2	LR-3	LR-4	FC
10 states	68.3%	69.3%	68.7%	69.1%
20 states	69.1%	69.8%	69.5%	69.5%

4. CONCLUSION AND FUTURE WORK

We described a method[5] for the classification of music signals that consists in a two-stage clustering of MFCC frames followed by a vector quantization and a classification scheme based on language modeling. We verified that the method was suitable for problems with different scales: Genre Classification, Artist Identification and computing of a distance between music pieces. The distance measure, used on a set of songs belonging to a single genre (Jazz), allowed us to derive consistent playlists. The proposed approach was compared with an HMM-based approach and a method that involves a linear combination of unigrams and bigram. On-going work include testing approaches based on compression techniques for symbolic strings.

5. REFERENCES

[1] J.-J. Aucouturier and F. Pachet, "Music similarity measures: What's the use?" in *ISMIR*, France, October 2002.

[2] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman, "A large-scale evaluation of acoustic and subjective music similarity measures," *Computer Music Journal*, vol. 28, no. 2, pp. 63–76, 2004.

[3] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *ICME*, 2001.

[4] K. West and P. Lamere, "A model-based approach to constructing music similarity functions," *Journal on Advances in Signal Processing*, 2007.

[5] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *ISMIR*, 2005.

[6] G. Tzanetakis and P. Cook, "Musical genre classification of audio singals," *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[7] T. Lidy and A. Rauber, "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification," in *ISMIR*, 2005, pp. 34–41.

[8] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, "Aggregate features and AdaBoost for music classification," *Machine Learning*, vol. 65, no. 2-3, pp. 473–484, 2006.

[9] H. Soltau, T. Schultz, M. Westphal, and A. Waibel, "Recognition of music types," in *ICASSP*, 1998.

[10] K. Chen, S. Gao, Y. Zhu, and Q. Sun, "Music genres classification using text categorization method," in *MMSP*, 2006, pp. 221–224.

[11] M. Li and R. Sleep, "A robust approach to sequence classification," in *ICTAI*, 2005.

[12] J.-J. Aucouturier, F. Pachet, and M. Sandler, "The way it sounds: Timbre models for analysis and retrieval of polyphonic music signals," *IEEE Transactions of Multimedia*, no. 6, pp. 1028 – 1035, 2005.

[13] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. MIT Press, 2002.

[14] P. Annesi, R. Basili, R. Gitto, A. Moschitti, and R. Petitti, "Audio feature engineering for automatic music genre classification," in *RIAO*, Pittsburgh, 2007.

[15] D. Ellis, "Classifying music audio with timbral and chroma features," in *ISMIR*, 2007.

[16] Y. Kim, D. Williamson, and S. Pilli, "Towards understanding and quantifying the "album effect" in artist identification," in *ISMIR*, 2006.

[5] This work was partially supported by FCT, through the Multi-annual Funding Programme.

A PERIODICITY-BASED THEORY FOR HARMONY PERCEPTION AND SCALES

Frieder Stolzenburg
Hochschule Harz, Automation & Computer Sciences Department, 38855 Wernigerode, GERMANY
fstolzenburg@hs-harz.de

ABSTRACT

Empirical results demonstrate, that human subjects rate harmonies, e.g. major and minor triads, differently with respect to their sonority. These judgements of listeners have a strong psychophysical basis. Therefore, harmony perception often is explained by the notions of dissonance and tension, computing the consonance of one or two intervals. In this paper, a theory on harmony perception based on the notion of periodicity is introduced. Mathematically, periodicity is derivable from the frequency ratios of the tones in the chord with respect to its lowest tone. The used ratios can be computed by continued fraction expansion and are psychophysically motivated by the just noticeable differences in pitch perception. The theoretical results presented here correlate well to experimental results and also explain the origin of complex chords and common musical scales.

1. INTRODUCTION

1.1 Motivation

Music perception and composition seem to be influenced not only by convention or culture, manifested by musical styles or composers, but also by the psychophysics of tone perception [1–3]. Thus, in order to better understand the process of musical creativity and information retrieval, the following questions should be addressed:

- What are underlying (psychophysical) principles of music perception?
- How can the perceived sonority of chords and scales, in particular of western music, be explained?

Therefore, in the rest of this section (Sect. 1), we will introduce basic musical notions and results. After that, we will briefly review existing psychophysical theories on harmony perception (Sect. 2), which are often based on the notions dissonance and tension, taking harmonic overtone spectra into account. In contrast to this, the approach presented here (Sect. 3) is simply based on the periodicity of chords. Applying this theory to common musical chords and also scales (Sect. 4), shows a very good correlation to empirical results, that e.g. most subjects prefer major to minor chords. Finally, we will highlight the psychophysical basis of the proposed approach, by reviewing some recent results from neuro-science on periodicity detection of the brain, and end up with conclusions (Sect. 5).

1.2 Basic Musical Notions

Before we are able to address the problem of harmony perception, we should clarify the terminology we use. For this, we follow the lines of [2]. The basic entity we have to deal with is a *tone*: A *pure* tone is a tone with a sinusoidal waveform. It has a specific pitch, corresponding to its perceived *frequency* f, usually measured in Hertz (Hz), i.e. periods per second. In practice, pure tones almost never appear. The tones produced by real instruments like strings, tubes, or the human voice have harmonic or other *overtones*. The frequencies of harmonic overtones are integer multiples of a fundamental frequency f. For the frequency of the n-th overtone ($n \geq 1$), it holds $f_n = n \cdot f$, i.e. $f_1 = f$. The amplitudes of the overtones define the *spectrum* of a tone or sound and account for its loudness and specific timbre.

A *harmony* in an abstract sense can be identified by a set of tones forming an interval, chord, or scale. Two tones define an *interval*, which is the distance between two pitch categories. The most prominent interval is the *octave*, corresponding to a frequency ratio of $2/1$. Since the same names are assigned to notes an octave apart, they are assumed to be *octave equivalent*. An octave is usually divided into 12 semitones in western music, corresponding to a frequency ratio of $\sqrt[12]{2}$ in equal temperament (cf. Sect. 3.3). Thus, intervals may also be defined by the number of semitones between two tones. A *chord* is a complex musical sound comprising three or more simultaneous tones, while a *scale* is a set of musical notes, whose corresponding tones usually sound consecutively. Both can be identified by the numbers of semitones in the harmony.

A *triad* is a chord consisting of three tones. Classical triads are built from major and minor thirds, i.e., the distance between successive pairs of tones are 3 or 4 semitones. For example, the major triad consists of the semitones $\{0,4,7\}$, which is the *root position* of this chord. An *inversion* of a chord is obtained by transposing the currently lowest tone by an octave. Fig. 1 (a) shows the three inversions of the E major chord, including the root position. Fig. 1 (b)–(e) shows all triads that can be build from thirds including their inversion, always with e' as lowest tone. Fig. 1 (f) shows the suspended chord, built from perfect fourths (5 semitones). Its last inversion, consisting of the semitones $\{0,5,10\}$, reveals this.

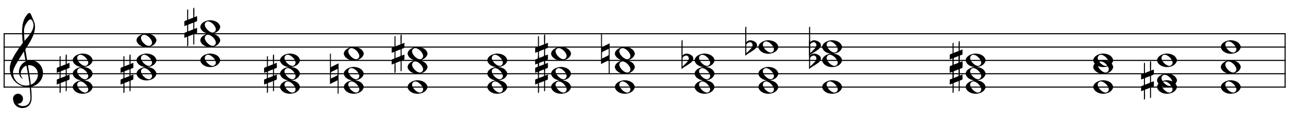


Figure 1. Triads and their inversions.

2. THEORIES ON HARMONY PERCEPTION

Chord classes lead to different musical modes. The major chord is often associated with emotional terms like *happy*, *strong*, or *bright*, and, in contrast to this, the minor chord with terms like *sad*, *weak*, or *dark*. Empirical results (see e.g. [4]) reveal a preference ordering on the perceived sonority of the triads as follows: major $\prec$ minor $\prec$ diminished $\prec$ augmented. Since all these triads are built from thirds, thirds do not provide an explanation of this preference ordering on its own. Therefore, let us now review existing theories on harmony perception, discussing some of their merits and drawbacks.

2.1 Explanation by Overtones

Overtones can explain the origin of the major triad and hence its high perceived sonority. The major triad appears early in the sequence, namely overtones 4, 5, 6 (root position) and —even earlier— 3, 4, 5 (second inversion). But it is well-known, that overtones fail to explain the origin of the minor chord.

2.2 Dissonance and Tension

Since the origin of harmony and scales cannot be explained well by overtones, newer explanations base upon the notions of dissonance [2,5] and tension [6]. In general, *dissonance* is the opposite to consonance, meaning how well tones sound together. Although this approach correlates better to the empirical results on harmony perception, it does not explain the low perceived sonority of the diminished or the augmented triad, which are built from two minor or major thirds, respectively. Therefore, [6] adopts the argument from psychology that neighboring intervals of equivalent size are instable and produce a sense of tonal tension, that is resolved by pitch changes leading to unequal intervals. Since lowering any tone in an augmented triad by one semitone leads to a major triad and raising to a minor triad, [6] assumes sound symbolism, where the major triad is associated with social strength and the minor triad with social weakness. But on the contrary, a minor triad becomes a major triad by raising the third. In addition, it is unclear whether suspended triads, built from two perfect fourths, also have a low perceived sonority. Finally, most of the empirical experiments on harmony perception present only single chords to the tested subjects. This means, there is actually no pitch movement at all.

3. A PERIODICITY-BASED THEORY

The approaches discussed so far more or less take the frequency spectrum of a sound as their starting point. Obviously, analyzing the frequency spectrum is closely related to analyzing the time domain (periodicity). Fourier transformation allows to translate between both mathematically. However, subjective pitch detection, i.e., the capability of our auditory system to identify the repetition rate (periodicity) of a complex tone sensation, only works for the lower but musically important frequency range up to about 1.500 Hz [3]. In consequence, a *missing fundamental* tone can be assigned to each interval. The tone with the respective frequency, called *virtual pitch* of the interval, is not present as an original tone component. It has nothing to do with (first-order) beats and is perceived not directly in the ear, but in the brain.

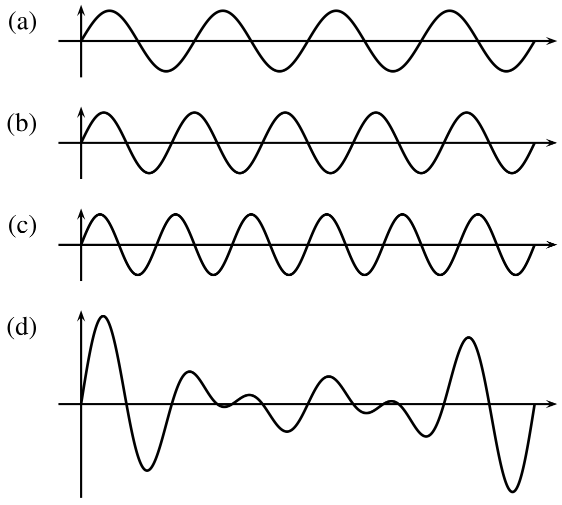


Figure 2. Sinusoids of the major triad.

3.1 Periodicity Pitch of Chords

For intervals, i.e. two tones, the concept of virtual pitch has been studied many times in the literature (see [3] and references therein). The idea in this paper now is to transfer this concept to chords by considering *relative periodicity*, i.e. the period length of complex sinusoids relative to the period length of the frequency of the lowest tone component (cf. [7, Sect. 7.1]). For example, the *A* major triad in just intonation consists of three tones with (absolute) frequencies $f_1 = 440\,\text{Hz}$, $f_2 = 550\,\text{Hz}$, and $f_3 = 660\,\text{Hz}$. The respective frequency ratios wrt. the lowest tone (a') are $F_1 = 1/1$, $F_2 = 5/4$ (third), and $F_3 = 3/2$ (fifth), corresponding to the semitones $\{0,4,7\}$. Fig. 2 (a)–(c) show the sinusoids for the three pure tone components and Fig. 2 (d) their superposition, i.e. the graph of the function $\sin(\omega_1 t) + \sin(\omega_2 t) + \sin(\omega_3 t)$, where $\omega_i = 2\pi f_i$ are the respective angular frequencies, and t is the time.

As one can see, the period length of the chord is (only) four times the period length of the lowest tone for this example. In the following, we call this ratio h. It depends on the frequency ratios $\{a_1/b_1,\ldots,a_k/b_k\}$ of the given chord. We assume, that each frequency ratio F_i is a fraction a_i/b_i (in its lowest terms), because otherwise no finite period length can be found in general, and it holds $F_i \approx f_i/f_1$ for $1 \leq i \leq k$. This means, all frequencies are relativized to the lowest frequency f_1, and $F_1 = 1$. The value of h then can be computed as $\mathsf{lcm}(b_1,\ldots,b_k)$, i.e., it is the *least common multiple* (lcm) of the denominators of the frequency ratios. This can be seen as follows: Since the relative period length of the lowest tone $T_1 = 1/F_1$ is 1, we have to find the smallest integer number that is an integer multiple of all relative period lengths $T_i = 1/F_i = b_i/a_i$ for $1 < i \leq k$. Since after a_i periods of the i-th tone, we arrive at the integer b_i, h can be computed as the least common multiple of all b_i.

3.2 A Hypothesis on Harmony Perception

We now set up the following hypothesis on harmony perception: The perceived sonority of a chord, called *harmonicity* in this context, decreases with the value of h. For the major triad in root position we have $h = 4$ (see above), which is quite low. Therefore, its predicted sonority is high. This correlates well to the empirical results, in general better than the approaches discussed in the previous section (Sect. 2), as we will see later on (in Sect. 4). In addition, the periodicity-based theory presented here is computationally simple, because it needs no assumptions on parameters, such as harmonic overtone spectra. Neither complex summation nor computing local extrema is required. Only the frequency ratios of the tone components in the chord are needed as input parameters. But we still have to answer the question, which frequency ratios should be used in the computation of h. Since this is done in a special way here, we present this now in more detail.

3.3 Tuning and Frequency Ratios

The frequencies for the k-th semitone in *equal temperament* with twelve tones per octave can be computed as $f_k = \sqrt[12]{2}^k \cdot f_1$, where f_1 is the frequency of the lowest tone. The respective frequency ratios are shown in Tab. 1 (a). The values grow exponentially and not linearly, following the *Weber-Fechner law* in psychophysics, which says that, if the physical magnitude of stimuli grows exponentially, then the perceived intensity grows only linearly. In equal temperament, all keys sound equal. This is essential for playing in different keys on one instrument and for modulation, i.e. changing from one key to another within one piece of music. Since this seems to be universal, at least in western music, we will adopt the equal temperament as *reference system* for other tunings.

The frequency ratios in equal temperament are irrational numbers (except for the ground tone and its octaves), but for periodicity detection they must be fractions, as mentioned above. Let us thus consider other tunings with rational frequency ratios. The oldest tuning with this property is probably the *Pythagorean tuning*, shown in Tab. 1 (b). Here, frequency relationships of all intervals have the form $3^m/2^n$ for some integers m and n, i.e., they are based on fifths, strictly speaking, a stack of perfect fifths (frequency ratio 3/2), applying octave equivalence. However, although huge numbers appear in the numerators and denominators of the fractions in Pythagorean tuning, the relative errors compared to equal temperament (shown in brackets in Tab. 1) grow up to more than 1%.

In fact, the Pythagorean tuning does not follow results of psychophysics, namely that human subjects can distinguish frequency differences for pure tone components only up to a certain resolution, namely 0.5% under optimal conditions. For the musically important low frequency range, especially the tones in (accompanying) chords, this so-called *just noticeable difference* is worse, namely only below about 1% [3]. Therefore, we should look for tunings, where the relative error is approximately 1%. In addition, the frequency ratios should be simple integer ratios, i.e. fractions with small numerators and denominators. In order to achieve the latter, we can look in the harmonic overtone sequence,when a tone of the chromatic scale appears for the first time, applying again octave equivalence. The result of this procedure, which we will call *overtonal tuning*, leads to frequency ratios of the form $m/2^n$ for some integers m and n as shown in Tab. 1 (c). However, as one can see, the relative error compared to equal temperament again is sometimes high.

In the literature (see e.g. [5] and references therein), other historical and modern tunings are listed, e.g. *Kirnberger III*, see Tab.1 (d). However, they are also only partially useful in this context, because they do not take into account the fact on just noticeable differences explicitly. In principle, this also holds for the adaptive tunings in [5], where simple integer ratios are used and scales are allowed to vary. An adaptive tuning can be viewed as a generalized dynamic *just intonation*, which fits well to musical practice, because the frequencies for one and the same pitch category may vary significantly during the performance of a piece of music. Trained musicians try to intonate e.g. a perfect fifth with the frequency ratio 3/2, and listeners are hardly able to distinguish this frequency ratio from others that are close to the value in equal temperament, namely $\sqrt[12]{2}^7 \approx 1.498$. In consequence, also the *rational tuning*, which we introduce now, primarily should not be considered as a tuning, but more as the basis for intonation and perception of intervals. We will use the frequency ratios of the rational tuning, shown in Tab. 1 (e), in our analyses of harmonicity. They are fractions with smallest possible denominator, such that the relative error wrt. equal temperament is just below 1%. They can be computed by means of *Farey sequences*, i.e. ordered sequences of completely reduced fractions between 0 and 1 which have denominators less than or equal to some (small) n, or by continued fraction expansion.

3.4 Continued Fraction Expansion

In mathematics, a (regular) *continued fraction* is an expression as shown in Fig. 3 (a), where the c_i are integer numbers that must be positive for $i > 0$. For a given rational or

interval	k	(a) equal temperament	(b) Pythagorean		(c) overtonal		(d) Kirnberger III		(e) rational	
prime, unison	0	1.000	1/1	(0.00%)	1/1	(0.00%)	1/1	(0.00%)	1/1	(0.00%)
minor second	1	1.059	$3^7/2^{11}$	(0.79%)	17/16	(0.29%)	25/24	(–1.68%)	16/15	(0.68%)
major second	2	1.122	9/8	(0.23%)	9/8	(0.23%)	9/8	(0.23%)	9/8	(0.23%)
minor third	3	1.189	$3^9/2^{14}$	(1.02%)	19/16	(–0.14%)	6/5	(0.91%)	6/5	(0.91%)
major third	4	1.260	81/64	(0.45%)	5/4	(–0.79%)	5/4	(–0.79%)	5/4	(–0.79%)
perfect fourth	5	1.335	$3^{11}/2^{17}$	(1.25%)	21/16	(–1.67%)	4/3	(–0.11%)	4/3	(–0.11%)
tritone	6	1.414	$3^6/2^9$	(0.68%)	23/16	(1.65%)	45/32	(–0.56%)	17/12	(0.17%)
perfect fifth	7	1.498	3/2	(0.11%)	3/2	(0.11%)	3/2	(0.11%)	3/2	(0.11%)
minor sixth	8	1.587	$3^8/2^{12}$	(0.91%)	25/16	(–1.57%)	25/16	(–1.57%)	8/5	(0.79%)
major sixth	9	1.682	27/16	(0.34%)	27/16	(0.34%)	5/3	(–0.90%)	5/3	(–0.90%)
minor seventh	10	1.782	$3^{10}/2^{15}$	(1.14%)	7/4	(–1.78%)	16/9	(–0.23%)	16/9	(–0.23%)
major seventh	11	1.888	243/128	(0.57%)	15/8	(–0.68%)	15/8	(–0.68%)	15/8	(–0.68%)
octave	12	2.000	2/1	(0.00%)	2/1	(0.00%)	2/1	(0.00%)	2/1	(0.00%)

Table 1. Table of relative frequencies for different tunings.

$$\text{(a)}\quad x \approx c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{c_3 + \cfrac{1}{\ddots}}}}$$

$$\text{(b)}\quad \begin{array}{ll} c_0 = \lfloor x \rfloor & c_n = \lfloor 1/x_{n-1} \rfloor \\ x_0 = x - c_0 & x_n = 1/x_{n-1} - c_n \end{array}$$

$$\text{(c)}\quad \begin{array}{lll} a_{-1} = 1 & a_0 = c_0 & a_{n+1} = a_{n-1} + c_{n+1} a_n \\ b_{-1} = 0 & b_0 = 1 & b_{n+1} = b_{n-1} + c_{n+1} b_n \end{array}$$

Figure 3. Continued fractions and Euclidean algorithm.

real number x, the values c_i can be computed recursively by the (extended) *Euclidean algorithm*, stated in Fig. 3 (b), where the floor function $\lfloor x \rfloor$ is used, which yields the largest integer less than or equal to x. The sequence of the c_i induces a sequence of fractions a_i/b_i, called convergents or fraction expansion of x, which can be computed by the equations in Fig. 3 (c). Continued fractions obey many interesting properties (see [8]), for instance:

- Any finite continued fraction represents a rational number.
- Every convergent a_i/b_i of a continued fraction is in its lowest terms, i.e. , a_i and b_i have no common divisors.
- Each convergent is nearer to x than the preceding convergent and also than any other fraction whose denominator is less than that of the convergent.

The most important property in this context is the last one, because it provides a procedure for computing the frequency ratios of the rational tuning as follows. For the k-th semitone, we consider the fraction expansion of $x = \sqrt[12]{2}^k$, i.e. the frequency ratio in equal temperament, until the relative error of the convergent $y = a_n/b_n$ wrt. x, i.e. the term $|y/x - 1|$, is less than 1%.

Continued fractions may help us explain the origin of the chromatic twelve-tone scale. For this, we look for a tuning in equal temperament with n tones per octave, such that the perfect fifth in just intonation (frequency ratio $3/2$) is approximated as good as possible. Thus, we develop a fraction m/n with $2^{m/n} \approx 3/2$, where m is the number of the semitone representing the fifth. Hence, we have to approximate $x = \log_2(3/2) \approx 0.585$. In this case, the sequence of convergents is 0/1, 1/1, 1/2, 3/5, 7/12, 24/41, 31/53, ..., showing $m/n = 7/12$ as desired, because semitone $m = 7$ gives the perfect fifth in the chromatic scale with $n = 12$ tones per octave.

4. APPLICATION OF THE THEORY

4.1 Comparison of Different Approaches

Let us now apply the periodicity-based theory to common musical chords and correlate the obtained results with empirical results. Tab. 2 shows the perceived and computed relative sonority of basic chord classes (cf. Fig. 1). Tab. 2 (a) shows the ranking for the perceived sonority according to empirical experiments reported in [4], which have been repeated by many others with similar results. Unfortunately, [4] does not consider the suspended triad. Therefore, it is not ranked in the table. Tab. 2 (b) provides the ranking for *complex tonalness* [2], whose numerical values are shown in brackets. The model according to [2] builds on earlier work [9]. However, especially the dissonance of the augmented triad is not reflected in this model by its calculated tonalness: It appears on rank 2, right after the major triad in root position. Therefore, [2] argues, that this has cultural rather than sensory origin. Tab. 2 (c) shows the ranking wrt. *instability* [6]. The notion of tension used in this model produces the desired low sonority of the diminished and the augmented triad (cf. Sect. 2.2). The correlation with the empirical results is good, but can still be improved, e.g., the minor triad in root position (rank 2) scores better than the inversions of the major triad (ranks 4 and 5), which is not as desired.

Tab. 2 (d)–(e) shows the ranking wrt. the harmonicity values h. As one can see, there is almost a one-to-

chord class		(a) empirical [4]	(b) tonality [2]		(c) instability [6]		(d) harmonicity		(e) harmonicity*	
major	$\{0,4,7\}$	1	1	(0.48)	1	(0.624)	2	(4)	2	(4.0)
	$\{0,3,8\}$	2	6	(0.38)	5	(0.814)	3	(5)	3	(5.0)
	$\{0,5,9\}$	3	3	(0.43)	4	(0.780)	1	(3)	1	(3.0)
suspended	$\{0,5,7\}$				8	(1.175)	4	(6)	4	(6.0)
	$\{0,2,7\}$				11	(1.219)	5	(8)	5	(8.0)
	$\{0,5,10\}$				9	(1.191)	6	(9)	6	(9.0)
minor	$\{0,3,7\}$	4	4	(0.42)	2	(0.744)	7	(10)	7	(10.0)
	$\{0,4,9\}$	5	7	(0.38)	3	(0.756)	8-9	(12)	8	(12.0)
	$\{0,5,8\}$	6	10	(0.32)	6	(0.838)	10-11	(15)	9	(15.0)
diminished	$\{0,3,6\}$	7	9	(0.35)	12	(1.431)	13	(60)	13	(26.0)
	$\{0,3,9\}$	8	5	(0.40)	7	(1.114)	10-11	(15)	10	(16.6)
	$\{0,6,9\}$	9	8	(0.37)	10	(1.196)	8-9	(12)	12	(19.9)
augmented	$\{0,4,8\}$	10	2	(0.44)	13	(1.998)	12	(20)	11	(19.7)

Table 2. Ranking relative sonorities of common triads.

one correspondence with the empirical results. The numbers in brackets are the respective harmonicity values h and h^*, where the latter are averaged over all inversions. For this, we compute the harmonicity of the given chord (cf. Sect. 3), e.g. the first inversion of the diminished triad $\{0,3,9\}$, that is $h_0 = \mathsf{lcm}(1,5,3) = 15$. In addition, we adopt each tone as reference tone, not only the lowest tone. Thus, we consider also the chords with the semitones $\{-3,0,6\}$ and $\{-9,-6,0\}$. For semitones associated with a negative number n, we take the frequency ratio of semitone $12-n$ according to Tab. 1 (e) and halve it, i.e., we do not apply octave equivalence here. Therefore, we get the frequency ratios $\{5/6,1/1,17/12\}$ and $\{3/5,17/24,1/1\}$ with harmonicity values $h_1 = 12$ and $h_2 = 120$, respectively. Since periodicity of chords is related to the lowest tone, we multiply the h values by the lowest frequency ratio in the chord, obtaining $h'_0 = 15$, $h'_1 = 5/6 \cdot 12 = 10$, and $h'_2 = 3/5 \cdot 120 = 72$. We then average the virtual chord frequencies f_1/h, where h appears in the denominator. Hence, we calculate the harmonic average of all harmonicity values h'_0, h'_1, and h'_2, which yields $h^* \approx 16.6$.

Tab. 2 (a) and (e) differ only in two respects: First, the most consonant chord according to harmonicity (rank 1) is the second inversion of the major triad with semitones $\{0,5,9\}$ and not the root position. Its calculated harmonicity is $h = 3$, which however coincides with the fact, that the second inversion appears before the root position in the harmonic overtone sequence (cf. Sect. 2.1). Second, the augmented triad appears late as expected (rank 11 of 13), but the root position and the second inversion of the diminished triad appear still later. However, the continued fraction expansion for the tritone (semitone 6, frequency ratio $\sqrt{2}$), occurring in both triads, yields first $7/5$, which is only slightly mistuned. This would lead to a significantly lower h value of the two chords – as desired. Thus, in summary, the periodicity-based approach on harmony perception fits best to empirical results.

4.2 Overtones and Periodicity

Harmonic overtone spectra are irrelevant for determining relative periodicities. The period length of such complex waveforms is identical with that of its fundamental tone. We obtain $h = 1$, since the frequencies of harmonic overtones are integer multiples of the fundamental frequency, hence all frequency ratios $\{1/1,2/1,3/1,\dots\}$ have 1 as denominator. Therefore, harmonicity is independent from concrete amplitudes and phase shifts of the sinusoids of the pure tone components. This seems plausible, because harmony perception only partially depends on loudness and timbre of the sound. It should not matter much, whether a chord is played e.g. on guitar, piano, or pipe organ. Of course, this argument only holds for tones with harmonic overtone spectra. If we have inharmonic overtones in a complex tone such as in *gamelan music* (cf. [5]), then it holds $h > 1$ for the harmonicity value of a single tone, i.e., we have an inherently increased *harmonic complexity* (cf. [2]).

4.3 From Chords to Scales

The harmonicity value h can be determined for harmonies, consisting of far more than three tones, without any computational problems. Thus, let us apply the formulae from Sect. 3 to general chords and scales. Fig. 4 (a)–(b) shows harmonies with 5 tones, that have low h values. The pentachord Emaj$7/9$ with $h = 8$, classically built from a stack of thirds, is standard in jazz music. Alternatively, it may be understood as superposition of the major triads E and B, which are in a tonic-dominant relationship according to classical harmony theory. Fig. 4 (b) shows the pentatonic scale ($h = 24$), which could alternatively be viewed as the standard jazz chord $E6/9$. All harmonies shown in Fig. 4 have low, i.e. good harmonicity values h, ranking among the top 5% in their tone multiplicity category. This also holds for the diatonic scale (7 tones, $h = 24$) and the blues scale (8 tones, $h = 24$) in Fig. 4 (c)–(d). Furthermore, according to their h^* value, all church modes, i.e. the diatonic scale and its inversions, rank among the top 11 of 462 possible scales with 7 tones. Therefore, the periodicity-based theory can contribute significantly to the discussion about the origin of scales of western music. There are other mathematical explanations for the origin of scales, e.g. by group theory [10], ignoring however the sensory psychophysical

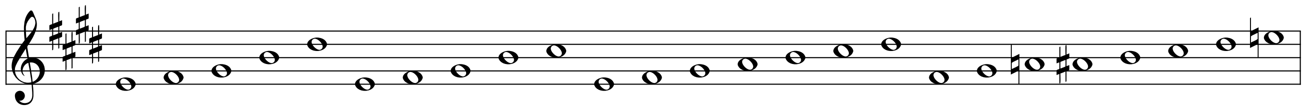


Figure 4. Harmonies (scales) with more than three tones.

basis for the musical importance of the perfect fifth.

5. CONCLUSIONS

As we have seen in this paper, harmony perception can be explained well by considering relative periodicities of chords, that can be computed from the frequency ratios of the intervals in the chord. The approach shows a good correlation to empirical studies on perceived sonority. Even the origin of scales can be described with this approach. It is mathematically simple, employing Farey sequences or the Euclidean algorithm for computing continued fractions. The approach has a strong psychophysically basis. It takes into account that human pitch perception is limited by a just noticeable difference of about 1% and assumes that virtual pitch of chords (chord periodicity) can be detected. The latter is indeed possible, as results from neuroscience prove, which we briefly review now.

5.1 Periodicity and Neuro-Science

From a spectral point of view, sounds are combinations of a fundamental frequency and certain overtones. Spectral analysis is performed in the cochlea. When a pure tone is detected, waves travel along the basilar membrane, which the cochlea houses, reaching a maximum amplitude at a point depending on the frequency of the tone [1–3]. Thus, the ear works as a spectral analyzer. This function of the ear is used in the explanations of harmony perception, based on overtones or dissonance (Sect. 2).

Periodicity-based explanations use missing fundamental tones, i.e. tones that are physically not present and hence cannot perceived by the ear directly. It has been well-known for years that periodicity can be detected in the brain. For example, two pure tones forming a mistuned octave cause so-called *second-order beats*, although no exact octave is present [3]. Recently, neuro-science found the mechanism for being able to perceive periodicity. As a result of a combined frequency-time analysis, i.e. some kind of *auto-correlation* by comb-filtering, pitch and timbre are mapped temporally and also spatially and orthogonally to each other in the auditory midbrain and auditory cortex [1] (see also [11]). [12] reviews neuro-physiological evidence for interspike interval-based representations for pitch and timbre in the auditory nerve and cochlear nucleus. Timings of discharges in auditory nerve fibers reflect the time structure of acoustic waveforms, such that the interspike intervals (i.e. the period lengths) that are produced convey information concerning stimulus periodicities, that are still present in short-term memory [1].

5.2 Summary and Open Questions

From the good correlation of the periodicity-based theory with the empirical results presented here, one may conclude, that there is a strong psychophysical basis for harmony perception and the origin of musical scales. As underlying principle for this, periodicity detection turns out to be more important than spectral analysis, although cultural and other aspects certainly must not be neglected. The question, how different harmonies cause different emotions or subjective effects like happiness or sadness is not yet answered by this, of course.

6. REFERENCES

[1] Gerald Langner. Die zeitliche Verarbeitung periodischer Signale im Hörsystem: Neuronale Präsentation von Tonhöhe, Klang und Harmonizität. *Zeitschrift für Audiologie*, 46(1):8–21, 2007.

[2] Richard Parncutt. *Harmony: A Psychoacoustical Approach*. Springer, Berlin, Heidelberg, New York, 1989.

[3] Juan G. Roederer. *The Physics and Psychophysics of Music: An Introduction*. Springer, Berlin, Heidelberg, New York, 4th edition, 2008.

[4] L. A. Roberts. Consonant judgments of musical chords by musicians and untrained listeners. *Acustica*, 62:163–171, 1986.

[5] William A. Sethares. *Tuning, Timbre, Spectrum, Scale*. Springer, London, 2nd edition, 2005.

[6] Norman D. Cook and Takashi X. Fujisawa. The psychophysics of harmony perception: Harmony is a three-tone phenomenon. *Empirical Musicology Review*, 1(2), 2006.

[7] James Beament. *How we hear music: The relationship between music and the hearing mechanism*. The Boydell Press, Woodbridge, UK, 2001.

[8] Carl D. Olds. *Continued Fractions*, volume 9 of *New Mathematical Library*. L.W. Singer Company, 1963.

[9] Ernst Terhardt, Gerhard Stoll, and Manfred Seewann. Algorithm for extraction of pitch and pitch salience from complex tonal signals. *Journal of the Acoustical Society of America*, 71(3):679–688, 1982.

[10] Gerald J. Balzano. The group-theoretic description of twelve-fold and microtonal pitch systems. *Computer Music Journal*, 4(4):66–84, 1980.

[11] Ray Meddis and Michael J. Hewitt. Virtual pitch and phase sensivity of a computer model of the auditory periphery: I. Pitch identification, II. Phase sensivity. *Journal of the Acoustical Society of America*, 89(6):2866–2894, 1991.

[12] Peter A. Cariani. Temporal coding of periodicity pitch in the auditory system: An overview. *Neural Plasticity*, 6(4):147–172, 1999.

AUTOMATIC GENERATION OF MUSICAL INSTRUMENT DETECTOR BY USING EVOLUTIONARY LEARNING METHOD

Yoshiyuki Kobayashi
SONY Corporation, Japan
Yoshiyuki.Kobayashi@jp.sony.com

ABSTRACT

This paper presents a novel way of generating information extractors that obtain high-level information from recorded music such as the presence of a certain musical instrument. Our information extractor is comprised of a feature set and a discrimination or regression formula. We introduce a scheme to generate the entire information extractor given only a large amount of labeled dataset. For example, data could be waveform, and label could be the presence of musical instruments in them. We propose a very flexible description of features that allows various kinds of data other than waveform. Our proposal also includes a modified evolutionary learning method to optimize the feature set. We applied our scheme to automatically generate musical instrument detectors for mixed-down music in stereo. The experiment showed that our scheme could find a suitable set of features for the objective and could generate good detectors.

1. INTRODUCTION

Musical information extraction technology has been extensively studied for various kinds of applications. Generally speaking, it extracts some features from input data, and then applies discriminant or regression analysis to estimate an objective variable from the features. There are some popular feature sets like MFCC (Mel-frequency cepstrum coefficient) [1] and features defined in Mpeg-7 standard [2], along with many other proposed features designed by heuristics. Popular discriminant analyses, which estimate objective variable from given feature set, include SVM, AdaBoost, GMM, HMM and so on. For example, Soo-Chang Pei et al. introduced instrumentation analysis and identification method with MFCC, Mpeg-7 features, and SVM [3]. T.Kitahara et al. introduced instrument identification method which can estimate the note-by-note presence probability of musical instruments by using linear discriminant analysis and some features other than MFCC or Mpeg-7 [4]. In these studies, feature sets are designed by human.

Meanwhile, there are some studies on Feature Generation [5]. Typically, a feature is obtained with a feature extractor composed of some basic functions. Genetic programming (GP) is used to design a feature that gives optimum objective variable. However, only a single feature could be designed, rather than an effective set of features for multivariate analysis. As a result the generated extractor is not accurate enough compared to popular methods with discriminant and multi-dimensional feature set designed by human. Also the description of feature is specialized to waveforms. As such, we could not apply this method to other kinds of data such as log-frequency spectrum.

It would appear that we can realize more accurate information extractor if we could automatically generate a set of effective features specialized for the objective. The work presented here is an approach to automatically generate an information extractor from dataset. The resulting extractor includes a set of effective features to estimate the objective variable. It also supports various types of data as input. First, we introduce the structure of the information extractor that our proposal generates. Next, the modified evolutionary learning method to optimize the feature set is presented. And finally as an application of this approach, we introduce our experiment of designing musical instrument detectors.

2. STRUCTURE OF INFORMATION EXTRACTOR

Figure 1 shows the structure of information extractor.

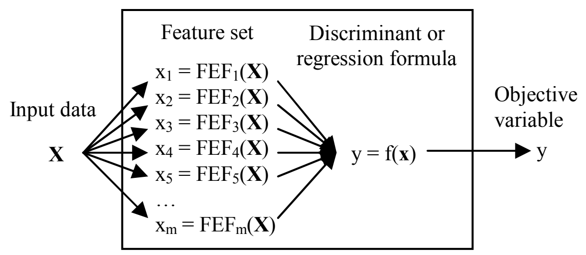


Figure 1. Structure of information extractor. **X** represents input data itself such as waveform. FEF represents a feature extraction function which extracts a single feature

from the input data. x_j represents the feature extracted by FEF_j, and **x** represents the feature vector consisting of x_j. f represents discriminant or regression formula which estimates the objective variable y based on the feature vector **x**.

First, the information extractor calculates multiple features from input data in accordance with the feature extraction functions (FEFs). The discriminant or regression formula estimates the objective variable from the extracted features. This structure itself is the same as the traditional information extractors. The difference is that our approach optimizes the entire information extractor, i.e. not only the discrimination or regression formula, but also the feature set.

2.1 Structure of input data

In our scheme, input data is expressed as a multi-dimensional matrix. For example, we can express stereo waveform as a two-dimensional matrix with channel and time dimensions (Figure 2). In this example, each element in two-dimensional matrix contains amplitude of the waveform in the channel at the time.

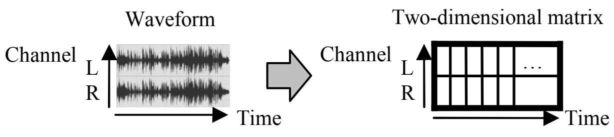


Figure 2. Example of input data of waveform.

Also we can express an image in RGB representation as a three-dimensional matrix with color, X, and Y axes (Figure 3). In this example, each element in three-dimensional matrix contains the brightness in RGB color space at the coordinate.

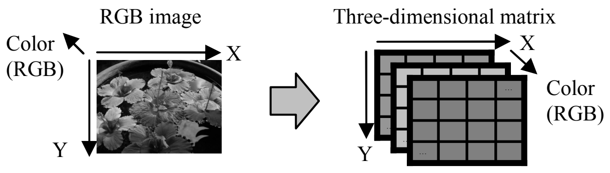


Figure 3. Example of input data of RGB image.

To express video data in this fashion, we would use four-dimensional matrix obtained just by adding one more dimension for time to the matrix for image. With this matrix based representation, we can flexibly handle various kinds of data as input data.

2.2 Description method for FEF

To support wide variety of input data and features, we propose a very flexible description of FEF. In our approach, FEF is formed as a cascade of basic functions (BFs) like a short computer program to reduce the input data matrix to a scalar. We prepared 51 BFs listed in Table 1.

Normalize	DiagonalDifferential	MStdDev
NormalizeAvg	Integrate	MaxIndex
NormalizeEach	LPF_1	Max
NormalizeEachAvg	HPF_1	Min
StandardizeEach	DCCut	MaximumNum
Abs	Order	MinimumNum
Sign	Window_Hanning	ZCR
Add	Window_Gauss	TCR
Multiply	MovingAverage	ZCP
InverseSign	Extract1	TCP
Sin	Cut	Difference
Cos	LogAxis	XDifference
Tan	LogAxisOctH	Histogram_1D
ASin	Mean	Histogram01
ACos	RMS	Histogram_2D
ATan	StdDev	Histogram2D01
Differential	MMean	DownSampling_To

Table1. List of basic functions.

The list includes four arithmetic operations, exponent functions, trigonometric functions, normalization algorithms, statistical functions, digital filters, etc. Figure 4 shows an example of FEF. And Figure 5 shows the calculation of the example FEF.

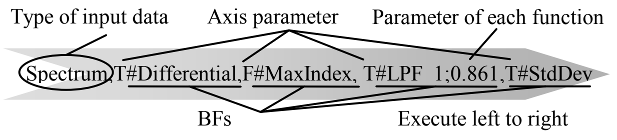

Type of input data
Axis parameter
Parameter of each function
Spectrum,T#Differential,F#MaxIndex, T#LPF_1;0.861,T#StdDev
BFs
Execute left to right

Figure 4. Example of FEF.

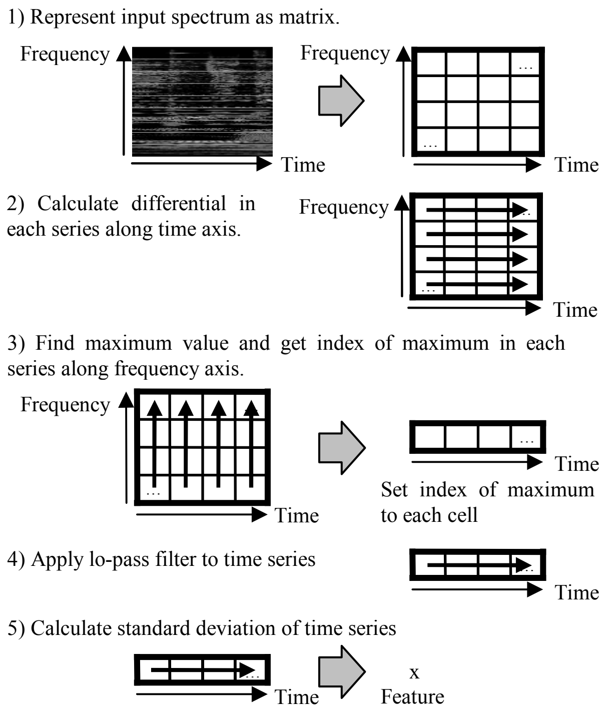

1) Represent input spectrum as matrix.
Frequency
Time
Frequency
Time
2) Calculate differential in each series along time axis.
Frequency
Time
3) Find maximum value and get index of maximum in each series along frequency axis.
Frequency
Time
Time
Set index of maximum to each cell
4) Apply lo-pass filter to time series
Time
5) Calculate standard deviation of time series
Time
x
Feature

Figure 5. Calculation of the example FEF.

First, FEF represents the input spectrum as two-dimensional matrix with time and frequency axes, then it calculates differential along time axis, finds maximal value and gets the position of maximal value along frequency axis, applies lo-pass filter along time axis, and calculates standard deviation along time axis. With this formula, it extracts a single feature from input data of two-dimensional matrix. F and T before # represent frequency and time axes, and these are the axis parameters representing the axis along which the given matrix is processed. As Figure 4 shows, it executes several processes to the matrix of input data by following the FEF from left to right. The number of dimensions of the matrix was reduced in the course of processing, and eventually, a single value is extracted from input data. Some BFs have parameters. There are two kinds of parameter, one is axis parameter that represents which axis to process, and the other is the specific parameter for each BF such as the coefficient of lo-pass filter.

2.3 Discriminant or regression formula

We use linear discriminant or regression analysis with feature selection to estimate the objective variable from the feature set as below.

$$y = f(\mathbf{x}) = \Sigma_j b_j x_j + b_0 \qquad (1)$$

b_j represents linear combination coefficients, and b_0 represents intercept coefficient. We use linear procedure here because we can easily calculate contribution ratio which we later use to optimize the information extractor as a whole. Also it would appear that we can obtain a measure of accuracy without non-linear procedure because FEF can express various non-linear conversions.

3. MODIFIED EVOLUTIONARY LEARNING METHOD

Information extractor is optimized over training dataset which is a list of input data with label information. Table 2 shows an example of dataset. The label can be 0 or 1 for two-class discriminant analysis, or a numeric value for regression analysis.

Input data	1.wav	2.wav	3.wav	4.wav	5.wav	…
Vocal presence	0	0	1	0	1	

Table 2. Example of dataset to generate a vocal presence detector which accepts a segment of waveform and estimates the presence of vocal in the waveform. 0 signifies no vocal present in the waveform, and 1 signifies vocal present.

As previously described, each FEF in the information extractor has immense flexibility, so we used evolutionary learning method to search for a good feature set from the infinite set of possibilities. One generation of our evolutionary learning method executes the following steps.

1. Feature set generation
2. Feature extraction
3. Linear discriminant or regression analysis with feature selection
4. Calculation of contribution ratio of each feature

These steps are repeated until the learning is stopped by a user.

3.1 Feature set generation

In the first generation, the method synthesizes the feature set which is a list of fixed number of FEFs by combining BFs randomly. To generate the FEF, first, it chooses a BF randomly from the prepared BFs. If the chosen BF has parameters, they are set also randomly. Then this process is repeated to append more BFs until the matrix of input data is reduced to a single value by the FEF.

In the second and later generations, the method generates a new feature set based on the feature set from the previous generation by evolutionary learning process. It uses the contribution ratio of each feature calculated in the fourth step of the previous generation as the evaluation of that feature. Figure 6 shows the schematic of feature set generation in the second and later generations. First, it selects features in the order of contribution ratio and adds them to the feature set of next generation unmodified until cumulative contribution ratio becomes 99%. Next, it generates some features by randomly selecting from highly contributing features and mutating them by inserting, deleting BFs or modifying parameters. Finally, it generates remaining features randomly as done in the first generation. Figure 7 shows an example of the mutation of FEF.

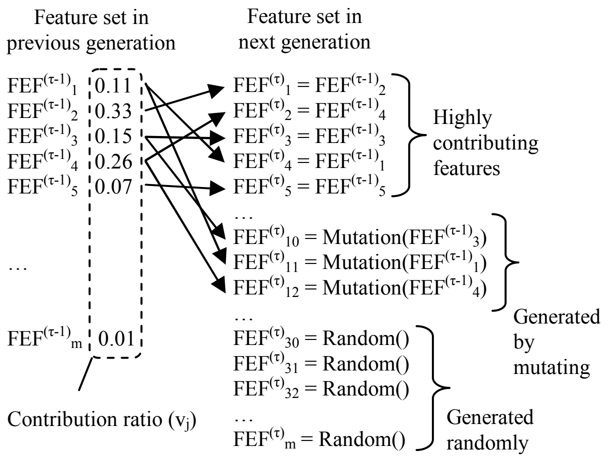


Figure 6. Example of feature set generation. τ represents generation in evolutionary learning process. Feature set in next generation contains highly contributing features in the previous generation, features generated by mutating the highly contributing features in the previous generation, and those randomly generated. All features in the first generation are generated randomly.

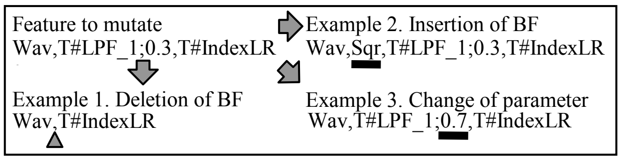

Feature to mutate
Wav,T#LPF_1;0.3,T#IndexLR
Example 1. Deletion of BF
Wav,T#IndexLR
Example 2. Insertion of BF
Wav,Sqr,T#LPF_1;0.3,T#IndexLR
Example 3. Change of parameter
Wav,T#LPF_1;0.7,T#IndexLR

Figure 7. Example of mutation of feature. A feature is mutated by inserting, deleting BFs or modifying parameters randomly.

3.2 Feature extraction

In this step, FEF_j extracts feature $x^{(i)}_j$ from input data with index i. At this point, we have dataset with its features.

3.3 Linear discriminant or regression analysis with feature selection

In this step, the method estimates parameters of discriminant or regression formula (**b**) in equation 1 with the dataset and the features calculated in step 2. Because some features are generated randomly, there are many meaningless or redundant ones in the generated feature set, particularly in the first generation. Feature selection is very important in keeping only the effective features to realize maximum generalization accuracy. It is also important for the calculation of fair contribution ratio of features from discriminant or regression formula. For the feature selection, we used local-search to search for a good combination of features from information criteria perspective. More precisely, first, it prepares parameter $u_j = \{1, 0\}$ which indicates whether the j-th feature is selected or not, and sets all bits to 0 at the beginning. Then, it tries inverting a single bit among u_j's one by one starting from the first one, estimates parameters **b** with the currently selected features by using least squares method, and calculates AIC [6] by comparing the estimated objective variable and the label in the dataset.

$$\mathrm{AIC} = n * \log(\mathrm{PMSE}) + 2 * (k+1) \qquad (2)$$

n represents the number of the input data in the dataset, PMSE represents the prediction mean square error, and k represents the number of the features selected in **u**. Among the possible m bit inversion positions, the one at which the AIC improved the most is selected and executed, and the local-search is continued. In case of no improvement, it finishes the local-search with the selected features and the computed **b** as the optimum with respect to AIC.

3.4 Calculation of contribution ratio of each feature

Contribution ratio of each feature is calculated by the following formula.

$$v_j = b_j / \mathrm{StDev}(\mathbf{x}_j) * \mathrm{StDev}(\mathbf{t}) * \mathrm{Correl}(\mathbf{x}_j, \mathbf{t}) \qquad (3)$$

v_j represents the contribution ratio of the feature with index j. **t** represents objective variable which is the label in the dataset. StDev($\mathbf{x}_j$) represents the standard deviation of the feature with index j in the dataset. StDev(**t**) represents the standard deviation of the objective variable in dataset. And Correl($\mathbf{x}_j$, **t**) represents the coefficient of correlation between $\mathbf{x}_j$ and **t**. If x_j is not selected in step 3, v_j becomes zero. If there are multiple objectives, we can just use mean contribution ratio from each formula for each objective. With step 1, highly contributing features will survive and prosper, and poorly contributing features will die. With iteration of steps 1 through 4, the feature set will improve with respect to the objective compared to the previous generation. While traditional GP methods can optimize only a single feature, our approach can optimize multiple features simultaneously to achieve better generalization accuracy. Moreover because we use contribution ratio to select features, we maintain the variety of features in the later generations, which alleviates the local optimum problem.

4. APPLICATION TO MUSICAL INSTRUMENT DETECTION

We used our scheme to automatically generate musical instrument detectors for mixed sound.

4.1 Dataset

We prepared about 100 commercially available music files which are sampled at 44.1 kHz in stereo. They cover variety of genres such as pops, rock, jazz, world, and so on, and various kinds of musical instruments appear in these music files. We labeled each 1-second interval according to the presence of 10 kinds of musical instruments which are vocal, harmonize, piano, clean guitar, distortion guitar, distortion guitar solo, strings, brass, bass and drums with true (1), false (0) or unclear (no label). If there is audible sound of the instrument in an interval, we labeled it 1, otherwise 0, and if we feel it is very difficult to determine the presence of the musical instrument from only 1-second of waveform even for human ear, we put no label. We decided that it was not necessary to label the whole music file because there are repetitions in music, so there are about 40% of unlabeled sections. Finally, we got 21,272 segments of 1-second waveform in total. Table 3 shows the number of correctly labeled segments for each musical instrument.

	Vocal	Harmonize	Piano	Brass	Strings
TRUE	3505	1655	3184	946	1810
FALSE	7884	12455	8748	12083	10643

	C. guitar	D. guitar	D.G. solo	Bass	Drums
TRUE	2684	1706	354	5675	5062
FALSE	10185	14836	16208	5078	4414

Table 3. Number of segments of waveform with correct label information.

Segments contain 3.2/10 musical instruments on average and 7.5/10 musical instruments at maximum if we treat non-labeled instrument as 0.5. And we shuffled these segments without keeping reference to the songs from which they were taken. We used the half for training, and the other half for testing.

4.2 Input data

Our scheme can handle waveform directly. However, we found that we can achieve better accuracy by applying suitable pre-processing that emphasizes the characteristics of the input data for the objective. So, we converted the waveforms into three kinds of input data whose names are "12TonesM", "12TonesF" and "12TonesB". Each data is two-dimensional matrix with dimensions of time and musical pitch. The difference among these three data will be shown later. Original waveform is converted to these matrices with the following steps.

4.2.1 Simplified sound source separation

We applied simplified form of the sound source separation algorithm described in [7] to obtain foreground and background sounds from the original stereo sound. Figure 8 shows the signal flow diagram of this sound source separation.

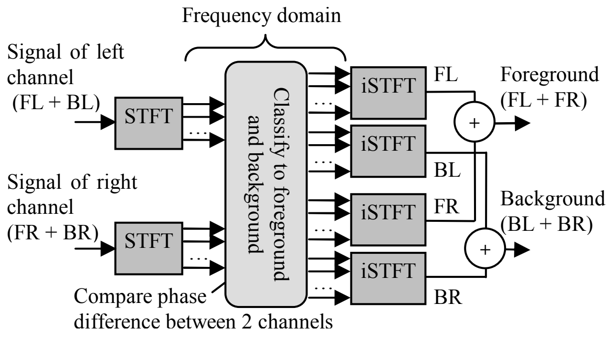

Frequency domain
Signal of left channel (FL + BL)
STFT
Signal of right channel (FR + BR)
STFT
Classify to foreground and background
Compare phase difference between 2 channels
iSTFT
iSTFT
iSTFT
iSTFT
FL
BL
FR
BR
Foreground (FL + FR)
Background (BL + BR)

Figure 8. Signal flow diagram of the simplified sound source separation. FL, BL, FR and BR represent foreground-left, background-left, foreground-right and background-right, respectively.

Each channel is analyzed with short-time Fourier transform with rectangle window of 16k samples and overlap of 8k samples. This very long frame size is needed to maintain the quality of separated sound. Then the phase difference between stereo channels in each frequency is calculated. If there is a difference greater than 0.2 PI, the frequency component is labeled as background. Otherwise, it is labeled as foreground. Then, for each channel, two waveforms for foreground and background are synthesized with inverse short-time Fourier transform with triangle window. This results in four channels of waveforms. Then, the left and right foreground channels are mixed, and the same is done for the background channels. As a result, two waveforms of foreground and background sounds are obtained. With this sound source separation, monaurally recorded sounds such as vocal, bass, snare and kick drums will appear in the foreground channel. On the other hand, sound recorded in stereo like strings or brass section will appear in the background channel.

4.2.2 Wavelet transform

We applied wavelet transformation to convert single waveform into two-dimensional matrix with time and musical pitch dimensions. We used band-pass filter which passes only a single semi-tone, as the mother wavelet. The original waveform was decomposed into 108 sub-bands corresponding to 12 semi-tones over 9 octaves. Then the logarithm of energy in each 7.8ms in each semi-tone is calculated. Figure 9 and Figure 10 show the schematic diagram and an example result of this process.

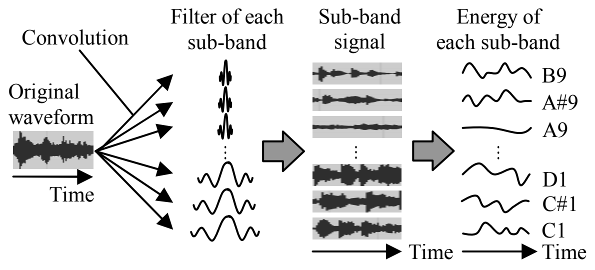


Figure 9. Schematic diagram of wavelet transform. It separates original waveform into 108 sub-bands, and calculates energy in 7.8ms in each band.

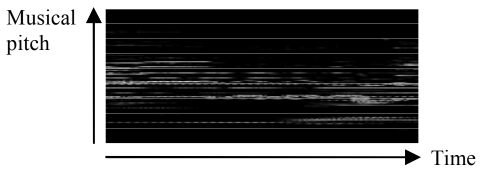


Figure 10. Example of result of wavelet transform. Brightness represents energy in each time and each musical-pitch.

We used the result of this process from foreground sound as "12TonesF", result from background sound as "12TonesB", and average of foreground and background as "12TonesM".

4.3 Result of learning

With our scheme and dataset, we generated musical instrument detection algorithms for mixed sound. Number of features is 1,000, and 165 generations were used in our evolutionary learning method. Figure 11 shows the learning curve. For comparison, it also shows the result for extractors with single feature. They are optimized with GP by selecting 3% of features most correlated with the label information in each generation.

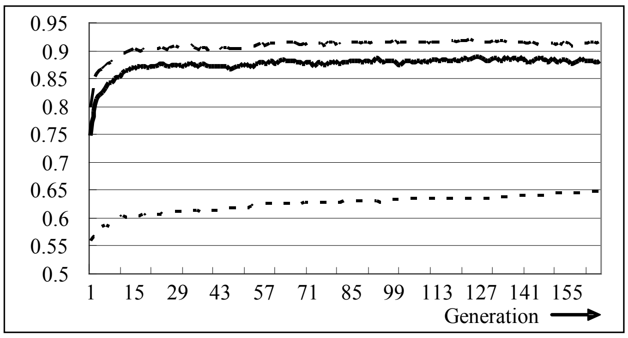


Figure 11. Learning curve. Dashed line represents the F-measure on training dataset averaged over all musical instrument detectors, and solid line represents the F-measure on testing dataset. Dotted line represents the F-measure of the detector with single feature optimized with GP on testing dataset.

As the learning curve shows, in the first generation, our detector realized average F-measures of 0.75 on testing dataset with features selected from 1,000 randomly generated features of various sorts. In the final generation, it realized 0.88 with the feature set optimized with our scheme. There is very clear advantage over the result of extractor with single feature optimized with GP. And Table 4 shows the F-measures for each musical instrument in the final generation on testing set.

Vocal	0.912	Brass	0.751	D. Guitar	0.956	Drums	0.991
Harmonize	0.852	Strings	0.794	D.G.Sol	0.762		
Piano	0.92	C. Guitar	0.897	Bass	0.987	Avg.	0.882

Table 4. F-measures of each musical instrument detector in the final generation on testing set.

```
12TonesF,T#Differential,T#StdDev,F#Window_Hanning,F#Mean
12TonesF,Add;0.223798,T#Differential,F#Window_Gauss;0.210
475;0.532910,T#StdDev,F#Mean
12TonesB,T#StdDev,ACos,F#Difference;0.633319
12TonesF,T#MaximumNum,F#Difference;0.689421
12TonesB,T#StdDev,F#Difference;-0.623785
12TonesB,T#Mean,F#Difference;-0.703600
```

Table 5. Part of highly-contributing features found in final generation.

Finally, table 5 shows some examples of generated FEF. The first feature in table 5 takes log-frequency spectrum of foreground as input, calculates differential in each series along time axis, calculates standard deviation in each series along time axis, processes Hanning window to frequency series and calculates average from frequency series. "Difference" function in table 5 splits the input in two at the boundary specified by the parameter, computes the sums for the two parts, and outputs the difference of the sums. It is not easy to understand what is going on in these generated features explicitly. However, it looks like it found variety of features, not only ones like MFCC and Mpeg-7 but also unique features with alien concept.

5. CONCLUSION

We presented a novel method to automatically design a information extractors. We introduced a very flexible description of features which supports various kinds of data types, and a modified evolutionary learning method to optimize multiple features given a partially labeled dataset. The method generated complete musical instrument detectors for mixed sound with various undiscovered and specialized features. The detectors realized either equal or superior performance compared to other methods even though the feature set is designed automatically given only the dataset without human intervention. Now we are applying the method to build various kinds of detection or recognition algorithms such as beat detection, attribute estimation, melody line estimation and more, not just for music recognition but for image recognition. We would like to report these results in the future.

ACKNOWLEDGEMENTS

The author would like to thank Francois Pachet, Pierre Roy, Gabriele Barbieri, Hideki Shimomura, Tatsuki Kashitani and Kei Tateno.

6. REFERENCES

[1] Beth Logan. Mel Frequency Cepstral Coefficients for music modelling. In International Symposium on Music Information Retrieval, 2000.

[2] H.G. Kim, N. Moreau and T. Sikora. MPEG-7 Audio and Beyond. Audio Content Indexing and Retrieval. John Wiley & Sons Ltd. 2005.

[3] Soo-Chang Pei, Nien-Teh Hsu. Instrumentation analysis and identification of polyphonic music using beat-synchronous feature integration and fuzzy clustering. Pages 169 – 172, ICASSP 2009.

[4] T.Kitahara, et al. Instrument Identification in Polyphonic Music: Feature Weighting to Minimize Influence of Sound Overlaps. EURASIP Journal on Advances in Signal Processing, Volume 2007.

[5] Pachet, F. and Roy, P. Analytical Features: A Knowledge-Based Approach to Audio Feature Generation. Eurasip Journal on Audio Speech and Music Processing, February 2009.

[6] H. Akaike. Information theory and an extension of the maximum likelihood principle. 2nd International Symposium on Information Theory, pages 267-281, 1973.

[7] Dae-young Jang, et al. Center channel separation based on spatial analysis. 11th International Conference on Digital Audio Effects, 2008.

RHYTHMIC SIMILARITY IN TRADITIONAL TURKISH MUSIC

Andre Holzapfel
Institute of Computer Science, FORTH and
University of Crete
hannover@csd.uoc.gr

Yannis Stylianou
Institute of Computer Science, FORTH and
University of Crete
yannis@csd.uoc.gr

ABSTRACT

In this paper, the problem of automatically assigning a piece of traditional Turkish music into a class of rhythm referred to as *usul* is addressed. For this, an approach for rhythmic similarity measurement based on scale transforms has been evaluated on a set of MIDI data. Because this task is related to time signature estimation, the accuracy of the proposed method is evaluated and compared with a state of the art time signature estimation approach. The results indicate that the proposed method can be successfully applied to audio signals of Turkish music and that it captures relevant properties of the individual *usul*.

1. INTRODUCTION

Traditional music of Turkey has a big community of listeners, and the music is strongly related to the music of neighboring regions. For example, in Greece and Arabian countries music melodies of traditional music are often based on similar modal systems as in Turkey. Concerning rhythm, there is a correspondence in classes of rhythm found in Arabic music (*iqa'*) and in Turkey (*usul*), and dances encountered in Turkey have influenced rhythms played in Greek *Rembetiko* music. Thus, automatic retrieval of this information not only enables a better understanding of an important cultural heritage but may also be of major commercial interest. Methods for this type of retrieval can be assigned to the branch of computational ethnomusicology, as introduced in [20]. Only recently, first research results on the classification of Turkish music into melodic classes were presented [7]. The retrieval of rhythmic information from traditional Turkish music has not been addressed yet. In this paper, classification of samples of Turkish music into rhythmic classes is proposed. These classes are referred to as *usul* [16]. A data set containing samples of songs composed in six different *usul* has been compiled to conduct experiments. As it will be shown in the later Sections, in the context of this data set the classification into a specific rhythmic class is related to the recognition of the time signature in Western music.

In [18], an approach was presented to estimate the time signature of a piece of music based on symbolic descriptions (MIDI). This approach uses autocorrelation coefficients (ACF) derived from the annotated onsets. In [21], a time signature estimation system for audio signals was proposed and evaluated on a set of percussive music. The system estimates the tatum [2] of the signal using inter-onset intervals (IOI) and in parallel, ACF are computed from the amplitude envelope of the signal. Beat and bar length are chosen from the peaks of the ACF, taking into account the estimated tatum. In [8], the determination of musical meter was reduced to a classification into either binary or ternary meter. Beat indexes are extracted in a semi-automatic way and then ACF on a chosen set of features are used to decide on the meter type. Using audio signals, the general problem of rhythmic similarity was addressed previously in [10] [1] in the context of traditional music, in both cases by applying Dynamic Time Warping techniques. In [4], rhythmic patterns were computed from samples of Western ballroom dances.
In [14] a system was proposed for the automatic estimation of the musical meter, i.e., the estimation of the position of tatum, beat and bars in the signal. The estimation of bar positions in $\frac{3}{4}$ time signatures is mentioned to be error-prone. Compound time signatures such as $\frac{9}{8}$ are not mentioned and to the best of our knowledge no reliable method has been presented to estimate the meter in such signals.
On the other hand, compound or complex time signatures are commonly encountered in traditional music of Turkey. The time signatures can take various forms, as it will be detailed in Section 2. Furthermore, the goal of the approach presented in this paper is not only the correct estimation of a time signature, but a description of the rhythmic properties of a class, because *usul* cannot be only distinguished by time signature in all cases. In [11], audio samples of traditional dances were compared: ACF were computed from onset strength signals (OSS) and these ACF were transformed into the scale domain by using the scale transform [3]. This results in descriptors that do not vary due to tempo changes. Thus, the scale transform magnitudes (STM) can be used to compare the rhythmic content of audio using simple point to point distance measures without the need of meter estimation. The approach in [11] was shown to be superior to the DTW based approach presented in [10]. In this paper, it will be combined with the approach presented in [18] and applied to a set of MIDI data. MIDI data was chosen as a first step to approach the problem of automat-

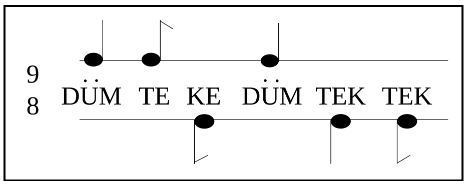


Figure 1. Symbolic description of the *usul Aksak*

ic rhythm description in Turkish music. This approach can easily be applied to audio signals by replacing the type of OSS, has no need of meter estimation and is robust to large tempo deviations.

Section 2 introduces the basic concepts of the analyzed type of music and describes the data set. Section 3 introduces the descriptors based on scale transform and proposes a method of comparison. Section 4 gives experimental results and Section 6 concludes the paper.

2. DATA SET

Compositions in Turkish traditional music follow certain schemes regarding their melodic and rhythmic content. Melodies are characterized by a modal system referred to as *makam*, and it defines a melodic texture consisting of specific tonal segments, progressions, directionality, temporal stops, tonal centers and cadences [13]. The rhythmic schemes encountered in traditional Turkish music are referred to as *usul*. An *usul* is a rhythmic pattern of certain length that defines a sequence of strong and weak intonations. An example is shown in Figure 1: the *usul Aksak* has a length of nine beats. The notes on the upper line labelled *düm* have the strongest intonation while the notes on the low line denote weak intonations. The note durations in the sequence shown in Figure 1 can be described as the string xoxxxoxox, where x symbolizes the start of a note and o metric unit without note [19]. Note that this representation is a further simplification of the one shown in Figure 1, because no differentiation of the intonation strength is contained. However these representations can be used for estimating the similarity between rhythms of same lengths by computing a chronotonic distance, as detailed in [19].

Unlike in [19], the length of the *usul* varies. According to H. Sadeddin Arel (1880-1955), the *usul* can be divided into minor and major *usul*. Minor *usul* have a length of up to 15 time units, while the major *usul* have up to 124 time units. As denoted in [16], minor usul are related to small musical forms, while larger musical forms employ the major usul in most cases. Musical forms that are usually composed in major usul are, e.g., *Presrev* and *Beŝte*. Two examples of small musical forms are *Sarkı* and *Türkü*. The latter are folk songs of unknown composers, while the former are short songs based usually on four lines of text with known composer. Both forms have in common that a song follows a certain minor *usul* and a certain *makam*, and both forms are vocal music. The most popular songs in Turkish music are composed in these forms. Because of that, along with a system for the recognition of the *makam* as presented in [7], an approach for the recognition of the *usul* represents an essential element in automatic retrieval of information from this music. Apart from that, the relation between melody and *usul* has not been investigated and an automatic approach like the one presented here can give valuable insight into the relation between melody and *usul*.

The data set used in this paper consists of Turkish songs in the forms of *Sarkı* and *Türkü*. They are following six different types of rhythmic schemes having lengths from 3 up to 10: *Aksak* ($\frac{9}{8}$), *Curcuna* ($\frac{10}{8}$), *Düyek* ($\frac{8}{8}$), *Semai* ($\frac{3}{4}$), *Sofyan* ($\frac{4}{4}$), and *Türk Aksaği* ($\frac{5}{8}$). The software *mus2okur* [13] has been used to obtain a data set consisting of 288 songs distributed along the six classes as shown in the second line of Table 1. Each sample consists of a MIDI description of the song melody, in most cases also a MIDI voice with a percussive accompaniment is contained. This percussive accompaniment has been left out, in order to be able to focus on the rhythmic properties of the melody. Due to the character of this music, there exists no chord accompaniment.

As all *usul* in the data set have different length, the recognition of the *usul* can be reduced to a recognition of its length. This is closely related to the task of time signature recognition and motivates the experimental setup described in the following Sections. The lower two lines in Table 1 depict the mean values of the tempi in *bpm* (beats per minute) and the standard deviation of the tempi, respectively. It is apparent that there are large overlaps between the tempo distributions of the *usul*. Thus, a system for *usul* length estimation for a given audio signal has to be robust to the tempo deviations and overlaps.

Table 1. Data set: number of songs, mean and standard deviation of tempi in *bpm*

CLASS	AKS	CUR	DUY	SEM	SOF	TUR
N_{Songs}	64	57	47	22	60	38
MEAN	98.5	98.3	70.7	131.9	81.3	73.1
STD	27.9	13.5	12.6	26.3	16.7	22.3

3. TIME SIGNATURE ESTIMATION

3.1 Rhythm Description

3.1.1 Tempo-invariant ACF

In order to describe and compare the content of the samples, an autocorrelation based method as presented in [18] has been combined with a method used for estimating rhythmic similarity presented in [11]. The onset times are read from the MIDI files and each onset is assigned a weight. In [18], different methods to set the weights were evaluated, and in this paper the three most successfull weighting schemes have been applied: the weight of an onset can either be related to the note duration as proposed in [15], to characteristics of the melody [17], or all onsets are assigned the same weight. The best weighting scheme will be

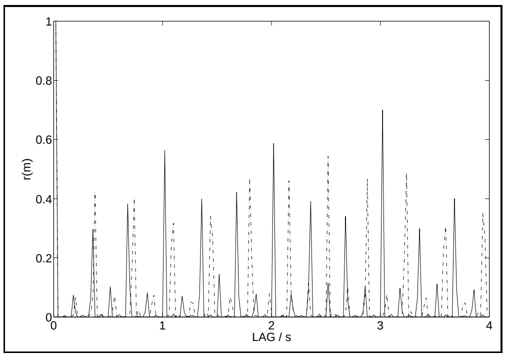


Figure 2. Autocorrelations r_u derived from two samples of *usul aksak*

determined in Section 4. In the method presented in [18], an onset strength signal (OSS) is generated at a sampling frequency related to the eighth note of the piece. This OSS has an impulse of height according to the assigned weight at the positions related to the onset time. From an OSS $o(n)$ an ACF $r(m)$ can be derived

$$r(m) = \frac{\sum_n o(n)o(n-m)}{\sum_n o(n)^2} \tag{1}$$

Note that the autocorrelations are not affected by tempo differences, when the OSS are computed at a sampling frequency that changes with the tempo (eighth note). Because of this, changing the tempo will result in constant ACF, which will be denoted as r_c.

3.1.2 Tempo-variant ACF

As mentioned in [18], beat tracking is a necessary step when applying the above described approach to audio. It is necessary to correctly estimate all metric levels in order to determine the eighth note pulse of the piece. When dealing with compound rhythms of different type as they are contained in the data set and commonly encountered in the music of Turkey and the whole eastern Mediterranean, no method has been presented yet to perform this task. For that reason, the MIDI data contained in the data set as described in Section 2 is used to compute OSS using a constant sampling frequency of $f_s = 50Hz$. From the OSS autocorrelations are derived. For two pieces having the same time signature but different tempi, their autocorrelations will differ by an unknown scaling factor, as can be seen in Figure 2. This is particularly critical for the type of music examined in this paper due to the large tempo deviations as detailed in Section 2. In order to overcome this scaling problem, typically the beat tracking would be necessary in order to estimate the tempo difference between the pieces. However, in this paper the usage of the method introduced in [11] is proposed to avoid the intractable problem of beat tracking in the presence of complex and compound time signatures. Due to the unknown scaling factor depicted in Figure 2, a simple point-to-point distance measure cannot be applied when comparing these autocorrelations, which due to the unknown scaling will be denoted as r_u. In order to solve this problem, a scale transform has been applied

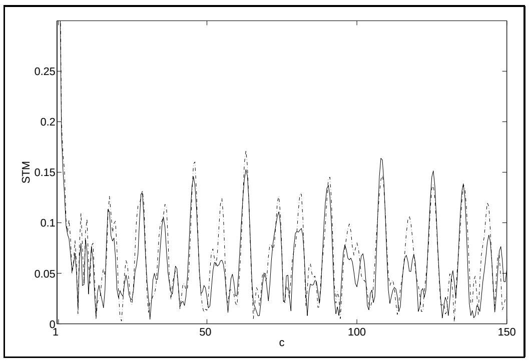


Figure 3. Two STM derived from the two *aksak* examples shown in Figure 2

to the autocorrelation sequence $r_u(t)$:

$$R(c) = \frac{1}{2\pi} \int_0^\infty r_u(t) e^{(-jc-1/2)\ln t} dt \tag{2}$$

The scale transform has the property that for a signal $r_u(t)$ and its time scaled version $\sqrt{a} r_u(at)$, with $a > 0$ being the scaling factor, the two computed scale transform magnitudes will be the same. This can be seen in Figure 3, where the two scaled autocorrelations from Figure 2 have been transformed to scale space. Due to the scale invariance property they are aligned and can be directly compared.

Thus, in this paper OSS will be computed from the MIDI files using a constant sampling frequency of $f_s = 50Hz$. Then, scale transform magnitudes (STM) are computed from the autocorrelations r_u using the discrete scale transform algorithm proposed in [22]. This results in a STM vector that describes the rhythmic content of the signal, the scale resolution was found to be of minor importance and has been set to $\Delta c = 0.5$. The accuracy in the task of time signature recognition when using either scaling free autocorrelations r_c or the STM derived from r_u will be compared. The results will indicate if by using a scale transform, the unsolved problem of meter estimation in complex time signatures can be avoided and the *usul* length could be determined by using this method.

3.2 Rhythm Dissimilarity

In order to determine the time signature of a piece the following approach will be applied: All pairwise dissimilarities between songs are computed using either the scale-free ACF r_c or the STM vectors, by using a cosine distance as proposed in [6] [9]. This results in dissimilarity matrices, having values close to zero whenever two pieces are found to be similar regarding their rhythmic content. In order to determine the accuracy of the proposed rhythmic similarity measure, the accuracies of a modified k-Nearest Neighbor (kNN) classification will be determined. For this, each single song will be used as a query for that a classification into one of the available classes is desired. This classification is performed by applying the modified kNN to the dissimilarity matrix. As shown in [10], a locally weighted kNN was found to improve accuracies on similar data, and

DURATION	MELODY	FLAT
80.2%	68.1%	72.9%

Table 2. Time signature recognition accuracies when using scale free r_c representation

therefore it has been used in the experiments. It assigns a weight $w_i = 1 - (d_i/d_{k+1})$ to the i-th training sample, where d_{k+1} is the distance of the $k+1$-nearest neighbor to the test sample. Thus, training samples more far away from the test sample contribute less to its classification.
An *usul* can be expressed in a simplified way as a string, as for example the string xoxxxoxox for *Aksak*. In Section 4, for some *usul* their string representations will be used to estimate their similarity using a method proposed in [19]: From the string representations chronotonic chains can be computed, by breaking down the rhythm into its smallest time unit on the *x-axis* and assigning to each element a height on the *y-axis* according to the beat-to-beat interval. This results in the chronotonic chain $[211221]$ in case of *Aksak*. As proposed in [19], in order to compare two such chronotonic chains, then a discrete form of the Kolmogorov Variational Distance (DKVD) can be applied. Given two chronotonic chains g and f of same length L, this distance can be computed as

$$K = \sum_{i=1}^{L} |f[i] - g[i]| \tag{3}$$

and is equal to the $1 - norm$ distance between the chains. Thus, by depicting an *usul* pair as two strings of same length, their rhythmic similarity can be estimated. In this paper, this method will be applied to pairs of *usul* for that samples frequently were confused in the time signature recognition.

4. EXPERIMENTS

4.1 Scale-free ACF

Three different weighting schemes have been evaluated in the experiments: the duration accent as proposed in [15], the melodic accent [17], and the flat accent (i.e., using the same accent weight for all onsets). Using the r_c autocorrelations computed using these three accents in the classification approach as described in Section 3.2, resulted in the best accuracies for the duration accent, as documented in Table 2. This contradicts with the findings in [18], where the melodic and flat accents were found to be preferable. Furthermore, using a selected range of autocorrelation coefficients could not further improve results on this data set, while in [18] using the coefficients of longer lags and leaving out the coefficients of short lags was found superior. This must be assigned to the differences between the data sets.

In Table 3 the confusion matrix for the best classification in Table 2 is shown. The biggest confusion happens between the $\frac{8}{8}$ time signature *usul* and the $\frac{4}{4}$ *usul* (*Düyek* and *Sofyan*, respectively). The pieces in the $\frac{8}{8}$-*usul* could be equivalently annotated in a $\frac{8}{4}$ time signature by changing their degree, referred to as *mertebe*, to four. The second biggest confusion happens between *Curcuna* and *Türk Aksaği*. The time signatures are related by a factor of two as well ($\frac{10}{8}$ and $\frac{5}{8}$). These types of errors have been denoted as typical as well in [18]. Still, the confusion between between *Düyek* and *Sofyan* is larger. This can be attributed to the different degree of similarity of the *usul*, which can be estimated using the approach proposed in [19]: In Table 4, the symbolic descriptions for the two confused *usul*-pairs are depicted as vectors of same length. From these descriptions the chronotonic chains have been derived that are depicted in Table 4. Note that *Sofyan* would be typically denoted as $[211]$ as its smallest beat-to-beat interval is a fourth note. In order to get chains of equal length, the eighth note has been chosen as smallest unit. Computing the Kolmogorov Variational Distances between the chronotonic chains, and normalizing by the length of the vectors it can be seen that the *usul Düyek* and *Sofyan* are more similar than the other pair. This is reflected in the higher confusion in Table 3. Thus, it can be concluded that the applied autocorrelation method is not only suitable for determining time signatures, but can as well capture rhythmic similarities contained in the piece.

		Predicted					
		9/8	10/8	8/8	3/4	4/4	5/8
Notated	9/8	62	0	1	0	1	0
	10/8	0	50	0	0	1	6
	8/8	1	4	24	0	18	0
	3/4	0	0	0	20	2	0
	4/4	2	0	12	0	46	0
	5/8	0	9	0	0	0	29

Table 3. Confusion matrix for r_c using duration accent

Symbolic Description			
Düyek:	xxoxxoxo	*Curcuna*:	xoxxoxoxox
Sofyan:	xoooxoxo	*Türk Aksaği*:	xoooxoooxo
Chronotonic Chains			
Düyek:	12212222	*Curcuna*:	2212222221
Sofyan:	44442222	*Türk Aksaği*:	4444444422
Normalized DKVD betw. Chronotonic Chains			
10/8=1.25		18/10=1.8	

Table 4. Computing chronotonic distances between confused *usul*

4.2 Scale Transform Magnitudes

The results presented in Section 4.1 have been obtained using the known note values that have been read from the MIDI files. As discussed above, when audio signals have to be examined instead of MIDI, this knowledge can only be obtained by means of beat tracking, which is an unsolved

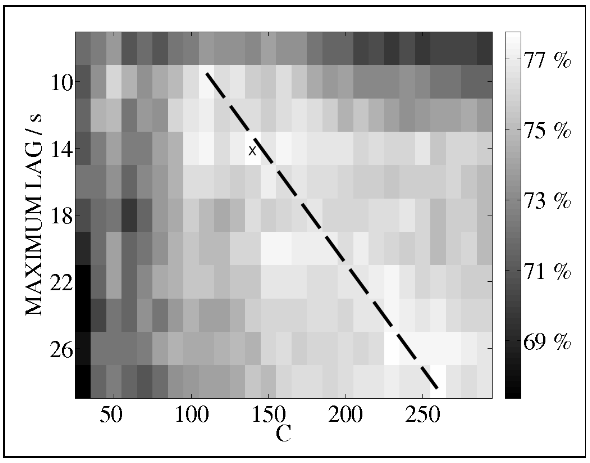


Figure 4. Result of the parameter grid search using the STM descriptors

		Predicted					
		9/8	10/8	8/8	3/4	4/4	5/8
Notated	9/8	51	3	3	1	3	3
	10/8	0	52	2	0	0	3
	8/8	1	1	30	2	11	2
	3/4	3	0	3	15	1	0
	4/4	0	2	8	1	48	1
	5/8	2	4	3	0	1	28

Table 5. Confusion matrix for STM at $C = 140$ and maximum lag of $14s$

task for the time signatures obtained in the data set. Thus, the STM represent a solution to avoid beat tracking, and in this Section the influence of its application on the resulting accuracies will be documented.

The parameters to set when using the STM are the maximum lag considered in the autocorrelation r_u and the number of scale coefficients C that is to be used when computing the cosine distance.

The influence of these parameters has been evaluated in a grid search. The resulting accuracies are depicted in Figure 4. It can be seen that by increasing the maximum lag size and the maximum scale coefficient the accuracies are improved until a level of about 77% is reached. The highest accuracy achieved at some points on the dotted line in Figure 4 is 77.8%, for example at $C = 140$ and at a maximum lag of $14s$ (marked in Figure 4). Choosing a point with small maximum lag leads to faster computation of the scale transform, and choosing a small value of C means a more compact STM description.

The related confusion matrix is shown in Table 5 and comparing it with the confusion matrix shown in Table 3 reveals very similar structure. The decrease in accuracy seems to be caused by some misclassification that cannot be justified by a similarity of the *usul*, as for example the $\frac{9}{8}$-time signature, which for the STM descriptor is randomly misclassified. Thus it appears that transforming autocorrelations to scale domain in the proposed way introduces some noise to the rhythm descriptors. However, the performance is only 2.4% lower than for using the scale-free autocorrelations (77.8% instead of 80.2%). Hence, by including scale transform the currently infeasible step of beat tracking in this kind of meters is avoided and time signature estimation is made feasible, when presented with arbitrary types of music signals having a compound or complex meter.

5. FUTURE WORK: TOWARDS AUDIO SIGNALS

As mentioned in [18], in order for the above described approach to work on audio instead of MIDI three algorithmic steps have to be added: onset detection, pitch estimation and beat tracking. The first step appears to be necessary, because the onset locations are not known as it is the case for MIDI. The pitch estimation is necessary only when the weights in the OSS are desired to be influenced by the pitch properties of the melody. On audio data, this can be approached using a fundamental frequency based OSS as proposed in [12], otherwise this step can be left out and an OSS as described in [5] can be used instead. The most error-prone step when dealing with audio is the beat tracking: it is necessary to correctly estimate all metric levels in order to determine the eighth note pulse of the piece, when the method as described in Section 3.1.1 is desired to be applied. Fortunately, the results using the STM as described in Section 3.1.2 avoids this step of beat tracking. Thus, time signatures and rhythmic properties can be captured by computing an OSS from an audio signal, and computing ACF and STM as described above. In order to evaluate the accuracy of the approach on audio data, a set of audio recordings similar to the MIDI data set will have to be compiled.

6. CONCLUSIONS

In this paper the application of scale transform for the recognition of time signatures is proposed. Using a data set of MIDI data with high class intern tempo deviations it is shown that this method achieves almost the same accuracy as a method that assumes that the metric levels of the piece are known. Thus, this method can be applied to the time signature recognition of audio signals by estimating an OSS suitable for the character of the signal and then computing the STM descriptors as proposed. This represents a significant achievement because the estimation of the metric levels in music signals having compound or complex meters is not a solved problem. The proposed approach is computationally simple because the scale transform can be performed using FFT algorithms. Furthermore, the proposed descriptors seem to capture a reasonable amount of information about the rhythmic properties of the *usul*, as could be seen in the relation between symbolic similarity and the confusion. As the rhythmic properties of Turkish music have never been studied using computational methods, this indicates an interesting direction for future studies. Next steps of these studies have to be the usage of audio signals and the examination of *usul* of same length.

7. REFERENCES

[1] Iasonas Antonopoulos, Angelos Pikrakis, Sergios Theodoridis, Olmo Cornelis, Dirk Moelants, and Marc Leman. Music retrieval by rhythmic similarity applied on greek and african traditional music. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, Vienna, Austria, 2007.

[2] Jeffrey A. Bilmes. *Timing is of the Essence*. PhD thesis, Master Thesis, Massachusetts Institute Of Technology, 1993.

[3] L. Cohen. The scale representation. *IEEE Transactions on Signal Processing*, 41(12):3275–3292, 1993.

[4] Simon Dixon, Fabien Gouyon, and Gerhard Widmer. Towards characterisation of music via rhythmic patterns. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, 2004.

[5] Daniel P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[6] Jonathan Foote, Matthew D. Cooper, and Unjung Nam. Audio retrieval by rhythmic similarity. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, pages 265–266, 2002.

[7] Ali C. Gedik and Baris Bozkurt. Automatic classification of turkish traditional art music recordings by ariel theory. In *Proc. of CIM08, 4th Conference on Interdisciplinary Musicology*, Thessaloniki, Greece, 2008.

[8] Fabian Gouyon and Perfecto Herrera. Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors. In *114th Convention of the Audio Engineering Society*, 2003.

[9] Andre Holzapfel and Yannis Stylianou. Musical genre classification using non-negative matrix factorization based features. *IEEE Transactions on Audio, Speech and Language Procesing*, 16(2):424–434, 2008.

[10] Andre Holzapfel and Yannis Stylianou. Rhythmic similarity of music based on dynamic periodicity warping. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP*, pages 2217–2220, 2008.

[11] Andre Holzapfel and Yannis Stylianou. A scale transform based method for rhythmic similarity of music. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP*, pages 317–320, 2009.

[12] Andre Holzapfel, Yannis Stylianou, Ali C. Gedik, and Barı¸s Bozkurt. Three dimensions of pitched instrument onset detection. *Accepted for publication in IEEE Trans. on Audio, Speech and Language Procesing*, 2009.

[13] M. K. Karaosmanoğlu, Süleyman Metin Yılmaz, Ömer Tören, Secgi Ceran, Utku Uzmen, Gülhan Cihan, and Emre Başaran. *Mus2okur*. Data-Soft Ltd., http://www.musiki.org/, Turkey, 2008.

[14] A. P. Klapuri, A. J. Eronen, and J. T. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Acoustics Speech and Signal Processing*, 14(1):342–355, 2006.

[15] R. Parncutt. A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception*, 11(4):409–464, 1994.

[16] Tolga Bektaş. Relationships between prosodic and musical meters in the beste form of classical turkish music. *Asian Music*, 36(1), Winter/Spring 2005.

[17] M. T. Thomassen. Melodic accent: Experiments and a tentative model. *Journal of the Acoustical Society of America*, 71:1596–1605, 1982.

[18] Petri Toiviainen and Tuomas Eerola. Autocorrelation in meter induction: The role of accent structure. *Journal of the Acoustical Society of America*, 119(2):1164–1170, 2006.

[19] Godfried T. Toussaint. A comparison of rhythmic similarity measures. In *Proc. of ISMIR - International Conference on Music Information Retrieval*, 2004.

[20] George Tzanetakis, Ajay Kapur, Andrew Schloss, and Matthew Wright. Computational ethnomusicology. *Journal of interdisciplinary music studies*, 1(2):1–24, 2007.

[21] Christian Uhle and Juergen Herre. Estimation of tempo, micro time and time signature from percussive music. In *Proc. of the Int. Conference on Digital Audio Effects (DAFx)*, 2003.

[22] W.J. Williams and E.J. Zalubas. Helicopter transmission fault detection via time-frequency, scale and spectral methods. *Mechanical systems and signal processing*, 14(4):545–559, July 2000.

PITCHED INSTRUMENT ONSET DETECTION BASED ON AUDITORY SPECTRA

Emmanouil Benetos, André Holzapfel, and Yannis Stylianou
Institute of Computer Science, FORTH, Greece,
and Multimedia Informatics Lab, Computer Science Department, University of Crete, Greece
{benetos,hannover,yannis}@csd.uoc.gr

ABSTRACT

In this paper, a novel method for onset detection of music signals using auditory spectra is proposed. The auditory spectrogram provides a time-frequency representation that employs a sound processing model resembling the human auditory system. Recent work on onset detection employs DFT-based features, such as the spectral flux and group delay function. The spectral flux and group delay are introduced in the auditory framework and an onset detection algorithm is proposed. Experiments are conducted on a dataset covering 11 pitched instrument types, consisting of 1829 onsets in total. Results indicate the superiority of the auditory representations over the DFT-based ones, with the auditory spectral flux exhibiting an onset detection improvement by 2% in terms of F-measure when compared to the DFT-based feature.

1. INTRODUCTION

The detection of the starting time of each musical note plays an important role in the analysis of music signals. This process is referred to as musical instrument onset detection and it is an essential step for music transcription applications, as well as for music signal compression, beat tracking, and music information retrieval. The goal of an onset detection system is the accurate estimation of note onset times, regardless of the instrument type or performance style. Several approaches for pitched instrument onset detection have been proposed in the literature, however they are mostly limited to a small number of instrument classes.

In [1], an onset detection system combining both energy and phase information was proposed. The employed dataset contained pitched nonpercussive, pitched percussive, nonpitched percussive, and complex sounds. Reported results indicated an improvement over energy and phase-based approaches. An improved version of the system in [1] was proposed in [4], tested on the same dataset. In [3], a system for onset detection employing a constant-Q pitch detector was proposed, tested on the pitched nonpercussive sounds also employed in [1]. It is also suggested in [3] that a detector based on a computational auditory model might improve onset detection performance. Gainza et al. employed FIR comb filters on a frame by frame basis combining the inharmonicity properties with the energy increases of the signal onset [5]. Results report an improvement offer energy-based and phase-based approaches. Finally in [6], the group delay function was proposed for onset detection in a beat tracking application. Multiband analysis was performed on two datasets, the first from the MIREX 2006 beat tracking task and the second containing samples of traditional Cretan music.

In this paper, a novel approach for onset detection is proposed by employing auditory spectrograms instead of DFT-derived spectrograms for the computation of onsets detection features. The auditory spectra, based on the model presented in [11], are designed to mimic the functions of the human auditory system. In the auditory domain, the group delay and spectral flux features are introduced, and an onset detection system is proposed. Comparative experiments on onset detection were performed using the same features in the DFT domain. The dataset used for experimentation contains a wide variety of pitched instrument types, not limited to western instruments, containing 1829 onsets in total. Results indicate that the auditory features outperform DFT-based features for onset detection, with the auditory spectral flux reaching an F-measure of 75.9%.

The outline of the paper is as follows. Section 2 is devoted to the DFT-based features and system for onset detection. In Section 3, the auditory model and features are presented, along with the proposed onset detection system. The employed dataset, the methods used for evaluation and the experimental results are discussed in Section 4. Conclusions are drawn and future directions are indicated in Section 5.

2. DFT-BASED ONSET DETECTION

2.1 Group Delay

As described in [6], phase information can be used for onset detection by considering the group delay $\tau(\omega)$, which for a given signal $x[n]$ with a phase spectrum $\Phi(\omega)$ is defined as the derivative of phase over frequency:

$$\tau(\omega) = -\frac{d\phi(\omega)}{d\omega} \tag{1}$$

The average of the group delay is determined by the distance between the center of the analysis window and the position of an impulse within the window, even when the impulse has been filtered by a causal and stable filter. As the onset of a musical instrument might be modelled by an impulse sent into a causal and stable system, in [6] the average group delay is used as an onset detection function: using a large overlap, an analysis window is shifted over the signal and for each window position the average group delay is computed. The obtained sequence of average group delays is referred to as phase slope function. In Figure 1, an example of a phase slope function is depicted by the dashed line which has positive zero crossings at the position of impulses in the signal. In order to avoid error problems when unwrapping phase in the group delay computation, the slope of the phase function can be computed as [10]:

$$\tau(\omega) = \frac{X_R(\omega)Y_R(\omega) + X_I(\omega)Y_I(\omega)}{|X(\omega)|^2} \tag{2}$$

where

$$\begin{aligned} X(\omega) &= X_R(\omega) + jX_I(\omega) \\ Y(\omega) &= Y_R(\omega) + jY_I(\omega) \end{aligned}$$

are the Fourier Transforms of $x[n]$ and $nx[n]$, respectively. The phase slope is then computed as the negative of the average of the group delay function. In this paper, the implementation of the phase slope onset detector as presented in [7] has been used, which includes a multiband processing of the complex DFT spectra and band-wise zero-crossing selection for increased accuracy. The resulting group delay onset detection signal, computed from the band-wise zero-crossing selection, contains peaks located at the time instants of the detected onsets.

2.2 Spectral Flux

Spectral flux (SF) is based on the detection of sudden positive energy changes in the signal which indicate attack parts of new notes. The accuracy of onset detection using SF and its computational simplicity were presented in [2, 4]. SF is computed as:

$$SF(k) = \sum_{\omega} HW(|X(\omega,k)| - |X(\omega,k-1)|) \tag{3}$$

where $HW(x) = \frac{x+|x|}{2}$ is the half wave rectifier function, and $X(\omega, k)$ is the STFT of the signal with $5.6ms$ hop size and a window length h of $46ms$. For the experiments in this paper, the L1-norm SF is used as shown in (3), since it was shown in [4] that it outperforms the L2-norm.

2.3 DFT-based Onset Detection System

Onsets are detected by selecting the zero crossings of the phase slope and the local maxima of the spectral ux detection signals. The onset detection method has been motivated by the processing steps proposed in [1]: first, the detection signals are smoothed using a Hanning window of length 51ms, which was found to be crucial for improving onset detection results. Afterwards, the signals are normalized using z-score. In [7], the application of an adaptive threshold has been shown to improve accuracy for SF, while it was found be impaired in case of PS. For that reason, an adaptive threshold is applied to SF only. It is computed by applying a moving median lter of length 97ms which is subtracted from the SF detection signals. Finally, a peak selection algorithm is performed in order to produce the detected onsets, by selecting peaks that are separated by a minimum peak distance of 40ms.

3. AUDITORY SPECTRUM-BASED ONSET DETECTION

In this Section the auditory model is presented, followed by the definition of the group delay function and spectral flux in the auditory spectrum domain. Finally, an onset detection system using auditory spectra is proposed.

3.1 Auditory Model

The auditory model was first introduced in [13] and formalized in [11]. It is inspired by physiological, psychoacoustical and computational studies in the human primary auditory cortex. The model consists of two stages, a spectral estimation model (designed to mimic the cochlea in the auditory system) and spectral analysis model (which mimics the primary auditory cortex). The spectral estimation model produces the so-called auditory spectrogram.

The auditory spectrum produces a time-frequency representation of the signal on a logarithmically scaled frequency axis, referred as the tonotopic axis. The auditory spectrogram consists of 128 log-frequency bins and can be approximated as:

$$X_A[n,l] = \max(\partial_l g(\partial_n x[n] *_n h[n,l]), 0), \tag{4}$$

where $x[n]$ is the original signal and $h[n, l]$ is a minimum-phase seed bandpass filter where $h[n, l] = \alpha h[\alpha n, l_0]$, with scaling factor $\alpha = 2^{l-l_0}$ and $l = 1, \ldots, 129$. The convolution of $x[n]$ with $h[n, l]$ is an application of a constant-Q filter-bank wavelet transform. ∂_i stands for differentiation over i, and $g(m) = \frac{1}{1+e^{-m}} - \frac{1}{2}$ is a sigmoid-like function, which is used to model the hair cell response in the human auditory system. It should be noted that in (4) two operations are not mentioned for simplicity purposes, they are however employed for the auditory spectra computation. The first consists of a temporal smoothing operation which filters out responses beyond 4 kHz and the second consists of a temporal integration of $X_A[n, l]$, which is followed by subsampling.

3.2 Auditory Group Delay

According to (2), and by noting that $X_A[n, l]$ has no imaginary values like the DFT-based group delay, the proposed function for computing the group delay in the auditory

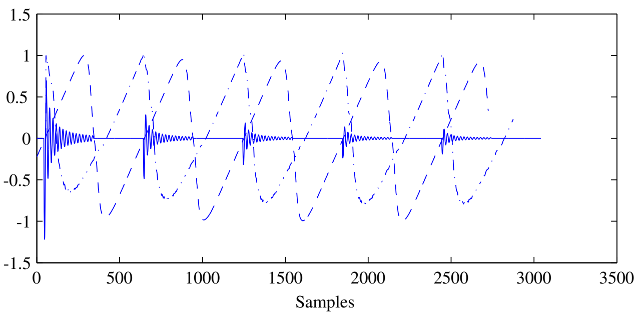


Figure 1. A sequence of impulses with linearly time varying amplitudes, the associated DFT-based group delay function (dashed line), and the associated auditory spectrum-based group delay function (dashed-dotted line).

spectrum is defined as:

$$AUD\_GRD[n,l] = \frac{Y_A[n,l]}{X_A[n,l]}, \quad (5)$$

where $Y_A[n,l]$ is the auditory spectrum of $nx[n]$. Due to the differentiation factor ∂_n in (4), onsets are detected by determining the positions of positive peaks rather than positive zero-crossings. In Figure 1, the auditory spectrum-based group delay that is obtained when shifting an analysis window over a sample signal is depicted as a dashed-dotted line. Note that the term group delay was preferred for the detection function instead of auditory phase slope, because no average value has been computed for neighboring bands as is the case for the DFT-based phase slope.

The processing steps for the computation of the onset detection signal, based on the auditory spectrum group delay function, can be seen in Figure 2. The auditory spectrum was computed using the NSL toolbox [9]. For the computation of the auditory spectrum the window length is set to 0.1s, with 4.5ms hop size and the resulted spectrogram is computed for a bandwidth of 76-3242 Hz. In processing block 2, the auditory group delay function is computed from auditory spectrograms $X_A[n,l]$ and $Y_A[n,l]$ using (5). For our analysis, tonotopic bands $b = 10, \ldots, 39$ of the auditory spectrogram were utilized, thus ignoring bands containing high-frequency noise, as well as bands ranging from 76-104 Hz which are not crucial for onset detection purposes, because these frequencies are below the F0 range of the investigated instruments. In processing block 3 of Figure 2, each band is smoothed in time using a 3rd degree Savitzky-Golay filter with window size equal to 12 samples [12]. The Savitzky-Golay filter uses local polynomial regression and is considered superior compared to FIR filters or moving average filters, preserving the local maxima of the signal while rejecting noise. In processing block 4, for each group delay band, peak picking is performed in order to select candidate onsets. For each band, an onset detection signal is constructed containing either the value zero when no peak has been detected, or the amplitude of the detected peak. In each band b, a threshold for peak detection is determined separately by the mean value of the half-wave rectified group delay function for the particular band. Finally, all band-wise detection signals are summed, creating a single onset detection signal based on the auditory group delay.

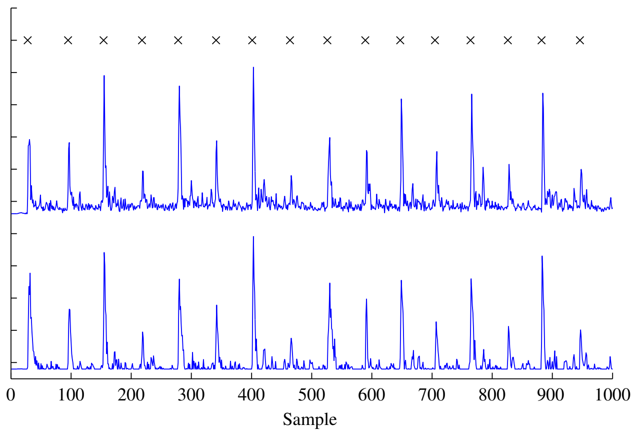


Figure 3. The spectral flux onset strength signals of a tanbur recording. The lower-placed signal depicts the auditory spectrum-derived spectral flux, while the higher-placed signal depicts the DFT-based spectral flux. The 'x' marker corresponds to the annotated onset time.

3.3 Auditory Spectral Flux

The spectral difference in the auditory domain is defined in a similar manner to the group delay. The spectral flux in the auditory spectrum is defined using the L1 norm:

$$AUD\_SF[n] = \sum_l HW(X_A[n,l] - X_A[n-1,l]). \quad (6)$$

For the auditory spectral flux, the original signal is resampled to 8kHz and the spectral flux is computed with a step size of 8*ms*. It should be noted that no band-wise smoothing or band selection was performed on the auditory spectral flux, since it was found to degrade onset detection performance. In Figure 3, the auditory spectrum-based and DFT-based spectral flux onset strength signals of a tanbur (plucked string instrument) recording are depicted. The annotated onset times can also be seen, as well as a false detection for the DFT-based spectral flux at sample 790.

3.4 Auditory Spectrum-based Onset Detection System

Onsets from the auditory group delay and spectral flux detection signals are detected using roughly the same approach as for the DFT representations, by selecting the local maxima of the signals. First, each detection function is normalized using z-score standardization. Afterwards, a moving median filter of length 0.2s is computed as an adaptive threshold, which is a robust method for detecting impulses in audio signals [8]. The adaptive threshold is then subtracted from the detection signals. Finally, peak picking is performed, by selecting peaks that are higher than threshold δ and are separated by a minimum peak distance of 40ms.

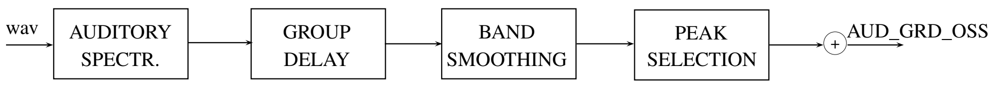


Figure 2. Block diagram of the computation of the auditory spectrum-based group delay.

Instrument	No. of onsets	No. of files
Cello	150	5
Clarinet	149	5
Guitar	174	5
Kemençe	186	5
Ney	147	7
Ud	211	5
Piano	195	5
Saxophone	148	5
Tanbur	156	5
Trumpet	140	5
Violin	173	5
Total	**1829**	**57**

Table 1. Onset dataset details.

4. EXPERIMENTS

4.1 Dataset

In our experiments, the dataset introduced in [7] was employed. It consists of 57 recordings of pitched instruments, including 11 instrument types, as seen in Table 1. The various instrument types can be organized into three classes: pitched-percussive instruments (guitar, ud, piano, and tanbur), wind instruments (clarinet, ney, saxophone, and trumpet), and bowed string instruments (cello, kemençe, and violin). It should be noted that the set is not limited to western instruments, but also contains middle-eastern instrument samples. In total, the recordings contain 1829 annotated onsets, while each instrument type contains roughly the same number of onsets. All recordings are monophonic, sampled at 44.1kHz.

4.2 Evaluation Methods

For evaluating the results of the proposed onset detection systems, the recall (R), precision (P), and F-measure (F) are employed as figures of merit. Let N_{tp} stand for the number of correctly detected onsets, N_{fp} the number of false positives, and N_{fn} the number of missed onsets. P and R are defined as:

$$P = \frac{N_{tp}}{N_{tp} + N_{fp}}, \quad R = \frac{N_{tp}}{N_{tp} + N_{FN}} \tag{7}$$

while the F-measure is computed from P and R:

$$F = \frac{2PR}{P + R} \tag{8}$$

It should be noted that P, R, and F are utilized for evaluation in the MIREX onset detection contests. An onset is correctly matched if it is detected within 50ms of the ground truth onset time. By varying parameter δ in small steps, P/R-curves can be created by placing R values on the horizontal axis and P values on the vertical one. The P/R-curve which is closer to the upper right corner of the diagram is considered to be the best detector with regards to F.

Feature	GRD	SF	AUD_GRD	AUD_SF
F-measure	73.7%	73.9%	73.8%	75.9%

Table 2. F-measures for the various onset detection features.

4.3 Results

The performance of the various onset detection features is shown in P/R-curves in Figure 4. In Figure 3(a) the performance of the complete dataset as described in Table 1 is shown. Regarding the optimum F-measure, the DFT-based group delay and spectral flux along with the auditory group delay seem to perform almost equally good, but they are surpassed by the auditory spectral flux. The best F-measures on the complete dataset can be seen in Table 2, where it can be seen that the auditory spectral flux outperforms the other three features by about 2% in terms of F-measure. The auditory group delay performs marginally better than its DFT-based counterpart, achieving high precision rates. In general, the auditory-based features outperform their respective DFT-based features.

As far as the individual instrument types are concerned, the auditory group delay exhibits very high precision rates for the set of string instruments in Figure 3(b), making it useful for beat tracking tasks. However, the auditory group delay is vastly outperformed by the remaining three features when pitched percussive instruments are employed in Figure 3(c), with the DFT-based spectral flux achieving very high precision and recall rates. The DFT-based spectral flux slightly outperforms the auditory spectrum-based spectral flux for pitched percussive instruments, which can be attributed to the limited frequency range of the auditory spectrum, since percussive onsets are detected in high frequency bands [2]. It should be noted that all features report high rates for pitched percussive instruments compared to string and wind instruments. Finally, the set of wind instruments in Figure 3(d) shows lower precision rates compared to the other sets. The auditory features achieve roughly the same best F-measure, outperforming the DFT-based features.

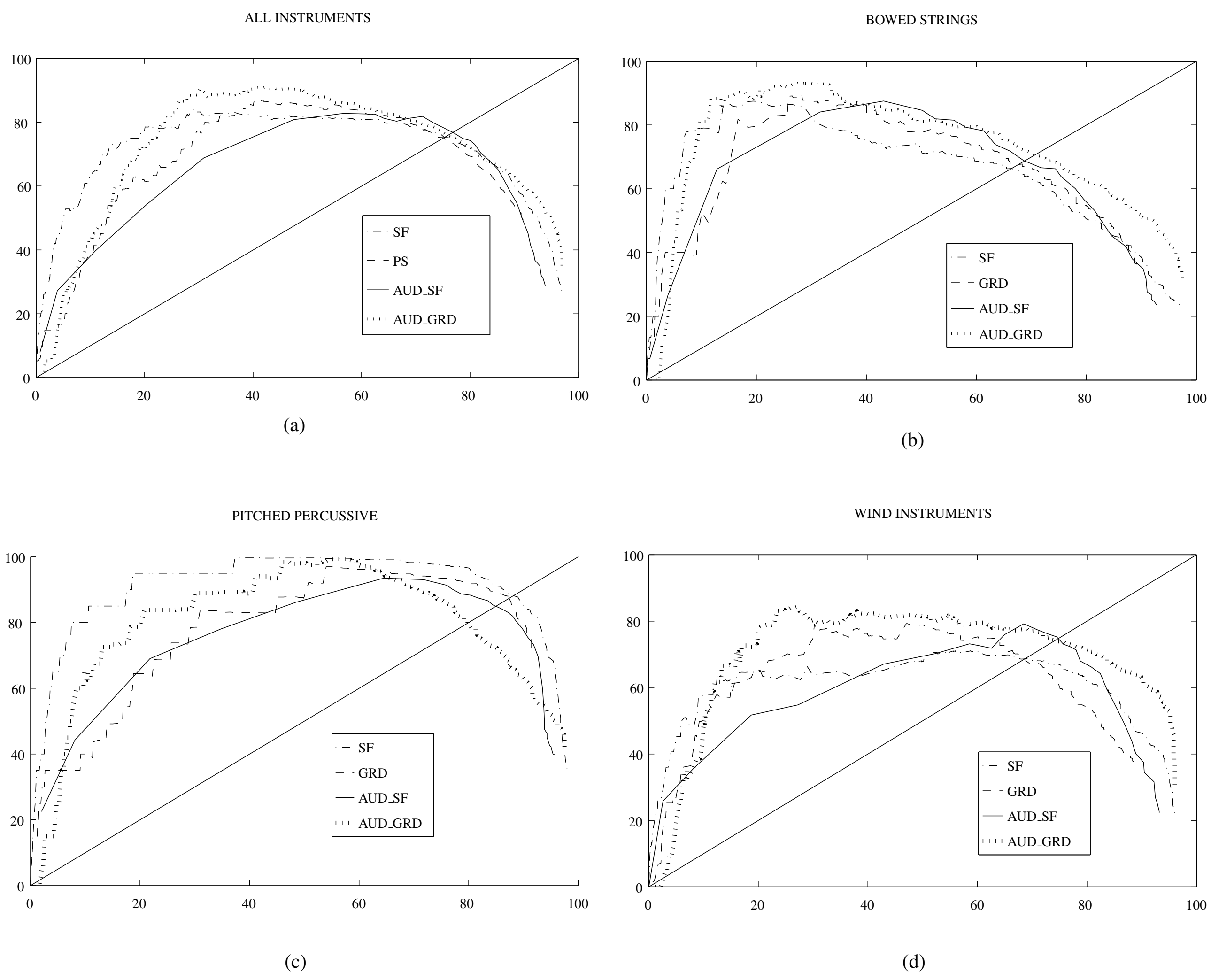


Figure 4. Performance curves of the various onset detection features. Recall and Precision values are plotted on the horizontal and vertical axis, respectively.

5. CONCLUSIONS

In this paper a new approach for onset detection using auditory spectra was proposed. The group delay function and spectral flux in the auditory domain were introduced as features for onset detection, and a system was proposed. The onset detection performance of the auditory spectral flux was found to be superior compared to the DFT-based feature, reaching an F-measure of 75.9% compared to 73.9% of the DFT-based spectral flux. While the performance of the auditory spectral flux for pitched percussive instruments was inferior compared to DFT-based features, it is relatively superior when string and wind instruments are tested.

In the future, a fusion of the onset detection features in the auditory domain will be performed, in an attempt to maximize onset detection performance. The system could also consider onsets produced by non-pitched percussive instruments, which can be easily detected using energy descriptors. In addition, the creation of an onset detection system which is dependent of the instrument family can lead to improved results. Finally, the aforementioned techniques can be developed for usage in polyphonic recordings.

6. REFERENCES

[1] J. P. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Proc. Letters*, Vol. 11, No. 6, pp. 553-556, June 2004.

[2] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection of music signals," *IEEE Trans. Speech and Audio Proc.*, Vol. 13, No. 5, pp. 1035-1047, Sep. 2005.

[3] N. Collins, "Using a pitch detector for onset detection," in Proc. *6th Int. Conf. Music Information Retrieval*, pp. 100-106, September 2005.

[4] S. Dixon, "Onset detection revisited," in Proc. *9th Int. Conf. Digital Audio Effects*, pp. 133-137, 2006.

[5] M. Gainza, E. Coyle, and B. Lawlor, "Onset detection

using comb filters," in Proc. *IEEE Workshop Applications of Signal Processing to Audio and Acoustics*, pp. 263-266, 2005.

[6] A. Holzapfel and Y. Stylianou, "Beat tracking using group delay based onset detection," in Proc. *9th Int. Conf. Music Information Retrieval*, Sep. 2008.

[7] A. Holzapfel, Y. Stylianou, A. C. Gedik, and B. Bozkurt "Three dimensions of pitched instrument onset detection," *IEEE Trans. Audio, Language, and Speech Processing*, accepted for publication.

[8] I. Kauppinen, "Methods for detecting impulsive noise in speech and audio signals," in Proc. *14th Int. Conf. Digital Signal Proc.*, Vol. 2, pp. 967-970, July 2002.

[9] T. Chi and S. A. Shamma, "NSL Matlab Toolbox," `http://www.isr.umd.edu/Labs/NSL/Software.htm`, Neural Systems Lab., Univ. Maryland.

[10] A. V. Oppenheim, R. W Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, Prentice Hall, 1998.

[11] P. Ru, "Multiscale multirate spectro-temporal auditory model," *PhD Thesis, Univ. Maryland College Park*, 2001.

[12] A. Savitzky, and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, Vol. 36, No. 8, pp. 1627-1639, July 1964.

[13] X. Yang, K. Wang, and S. A. Shamma, "Auditory representations of acoustic signals," *IEEE Trans. Information Theory*, Vol. 38, No. 2, pp. 824-839, March 1992.

SHADES OF MUSIC: LETTING USERS DISCOVER SUB-SONG SIMILARITIES

Dominikus Baur, Tim Langer, Andreas Butz
Media Informatics Group
University of Munich (LMU), Munich, Germany
{dominikus.baur,andreas.butz}@ifi.lmu.de, tim.langer@campus.lmu.de

ABSTRACT

Many interesting pieces of music violate established structures or rules of their genre on purpose. These songs can be very atypical in their interior structure and their different parts might actually allude to entirely different other songs or genres. We present a query-by-example-based user interface that shows songs related to the one currently playing. This relation is not based on overall similarity, but on the similarity between the part currently playing and parts of other songs in the collection along different dimensions (pitch, timbre, bars, beats, loudness). The similarity is initially computed automatically, but can be corrected by the user. Once a sufficient number of corrections has been made, we expect the similarity measure to reach an even higher precision. Our system thereby allows users to discover hidden similarities on the level of song sections instead of whole songs.

1. INTRODUCTION

All music is based on repetition on different levels: From the lowest level of sounds in different frequencies to the highest, cultural aspects of genres and trends, every song is contained in an intricate network of repeating segments. One of the best known of these patterns is the verse-chorus form [1] that has been defining for the last half century of popular music and implies inherent repetitive structures, possibly to increase recognition. Nevertheless, certain parts such as the intro, outro or especially the bridge can stand in complete contrast to the rest of the song, sometimes forming a mini-song of their own (and sometimes even digressing along this path and never returning to their origin).

Music recommendation and visualization often relies on an abstract idea of ”similarity” between songs, which is actually a measure for repetition. It is mostly generated by collaborative filtering or content-based measures, but this similarity normally works on the level of whole songs, with a set of related songs based on their averaged closeness. While some systems access songs on a lower level to extract segments, they do so to find the most representative part of the song to, again, do an overall comparison.

In this way, parts of songs with a high inner diversity (as in the bridge parts mentioned above) simply disappear in the similarity measure: While the overall impression of song A might be very similar to song B regarding content and sound, its bridge might be an allusion to a third song C and its outro even closer to another song D, neither of which is reflected in a generalized, one-dimensional similarity value. Query-by-example/humming systems, in contrast, have to rely on these deeper structures within a song, as they mostly have to work with incomplete input. Still, their main use is not to reveal hidden connections between parts of songs but to retrieve the one song that the user has in mind - multiple songs are only displayed because of inaccuracies in retrieval.

In this paper we present our web-based system *Shades of Music* that provides users with an interface to retrieve and discover connections between songs at the level of parts or sections. The user can listen to songs and see which other songs are similar to the currently playing section and in which of their parts. To stay with the example from above: For most of song A, sections from song B are shown as the most similar ones, but during A's bridge song C and during A's outro, song D appear. A similarity between these sections is initially calculated using the web service Echo Nest[2], but our system then encourages users to give feedback and improve its classification. In the rest of this paper we present related work in the areas of music user interfaces, then describe our system, the way users can give feedback, and the underlying calculations.

2. RELATED WORK

Query-by-example is an active field of research that aims for retrieving an item with only insufficient information. As the input mostly represents a part of the full item, extracting segments and being able to compare them is an important first step. Older QBE systems for audio mostly worked with symbolic MIDI-files[3], but more recent systems evaluate the actual audio signals. Various attributes, such as note sequences[3], melody[4] (e.g., with Query-by-humming), or beat[5] are used. Since these systems always try to retrieve one specific item, the segmentation is used to create a ranked list of possible candidates. More creative approaches to QBE such as [6] are trying to let

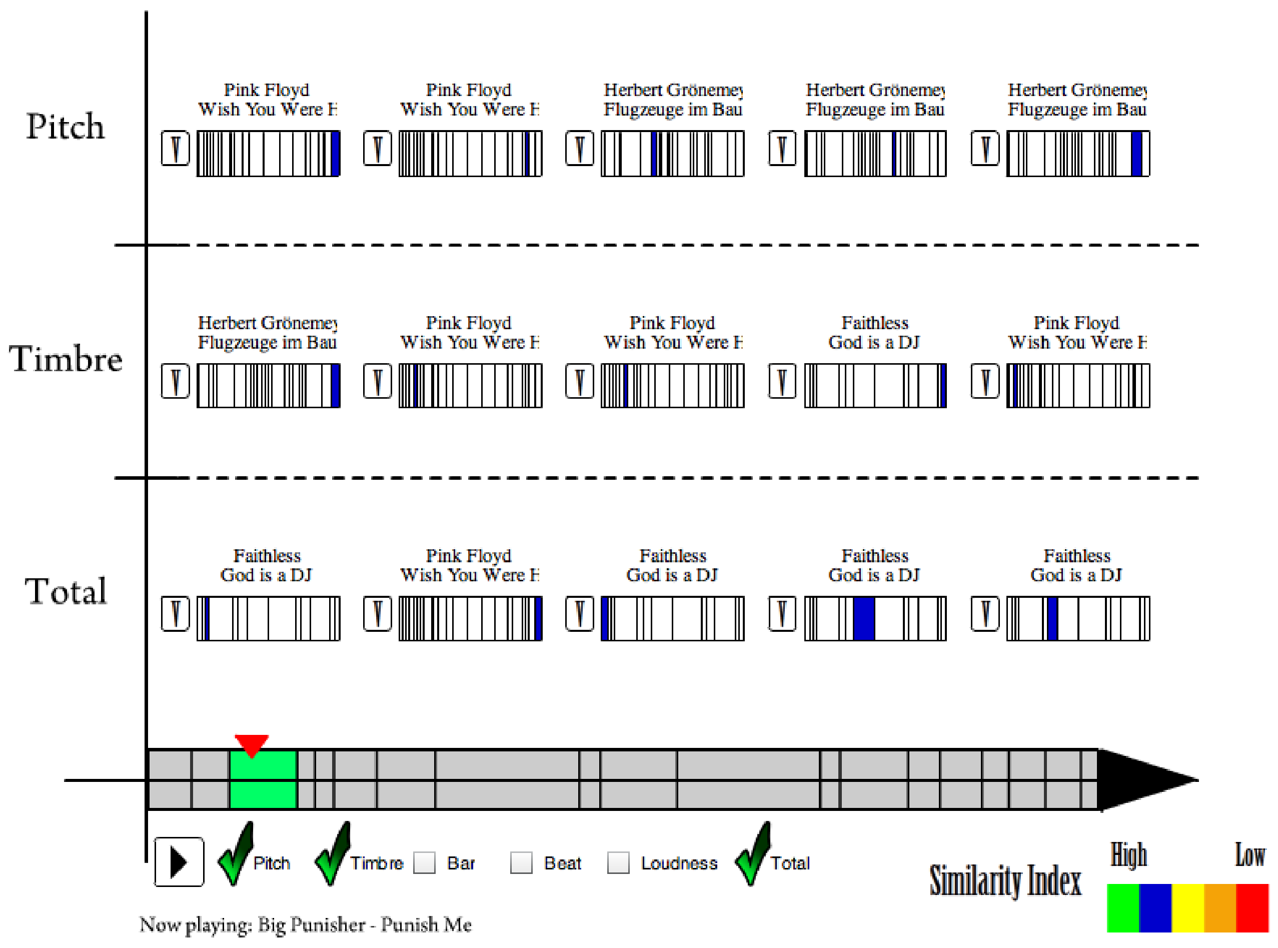

Pitch
Pink Floyd Wish You Were H
Pink Floyd Wish You Were H
Herbert Grönemey Flugzeuge im Bau
Herbert Grönemey Flugzeuge im Bau
Herbert Grönemey Flugzeuge im Bau
Timbre
Herbert Grönemey Flugzeuge im Bau
Pink Floyd Wish You Were H
Pink Floyd Wish You Were H
Faithless God is a DJ
Pink Floyd Wish You Were H
Total
Faithless God is a DJ
Pink Floyd Wish You Were H
Faithless God is a DJ
Faithless God is a DJ
Faithless God is a DJ
Pitch
Timbre
Bar
Beat
Loudness
Total
Now playing: Big Punisher - Punish Me
Similarity Index
High
Low

Figure 1. Shades of Music: Listening to a song and finding related songs based on different attributes

the user ”sketch” aspects of a song in various ways as an input to the system. Applications for representing larger music collections follow two courses: One approach is to visualize the collection in a global way, for example using the popular self-organizing maps (Islands of Music [7], but also [8] and [9]) or force-directed layouts[10]. Another way is to display related items based on one currently active item (in principle QBE) as in Musicream[11] or the Expressive Music Jukebox[12].

To make up for the shortcomings in automatic content-based similarity analysis and allow for personalization, user feedback is incorporated in various systems. Recommender systems[13], for example based on ratings [14] or implicit data such as listening histories in the online community Last.fm[15] offer the user suggestions for novel music. Connections between song parts are central, for example, for the music website Who Sampled? [16] whose community adds samples and their origin to the database.

3. SHADES OF MUSIC

Shades of Music is a (prototype of a) web-based service that allows users to listen to songs and find related sections. Based on the currently playing song, related sections of other songs are displayed. Echo Nest does an initial similarity classification, but as the interface collects user feedback this similarity measure becomes more accurate. We implemented Shades of Music with the Ruby on Rails framework on the server side and a browser application based on Adobe Flash on the client side.

3.1 The User Interface

Initially, a list of all songs in her or his collection is displayed. Additional songs can easily be uploaded from the computer or retrieved from online sources. After choosing a song, the application starts to play this song and displays the main interface (see figure 1). A horizontal bar at the bottom represents the current song and its sections. A play head and additional color highlighting show the currently playing section of the song. With the check boxes below the bar, the user can choose the criteria based on which related sections are displayed. Pitch, timbre, bar, beat and loudness plus a cumulative total value are available. For each selected attribute, an additional line of songs is displayed (”Pitch”, ”Timbre” and ”Total” in figure 1). Their order reflects the similarity: The most similar song section appears on the left followed by less similar ones to the right. For each of these sections, the complete song is displayed including artist and title. Each of these songs is again divided into its subsections with unrelated sections transparent and related sections with a five-step color coding that shows similarity for the current attribute from low to high. Once familiar with this visualization, the user can see at first glance that, for example, the intro of another song is similar to the active section. It is important to note here that the same song might appear not only once but several times: Once among each of the different attributes but also within the list for one attribute if more than one section of the song corresponds to the current section (see ”Faithless - God is a DJ” in figure 2). The lists contain only the five most similar songs along that particular dimension. Since the sections of a song are often very similar to other sections of the same song, related sections from the current song are not displayed.

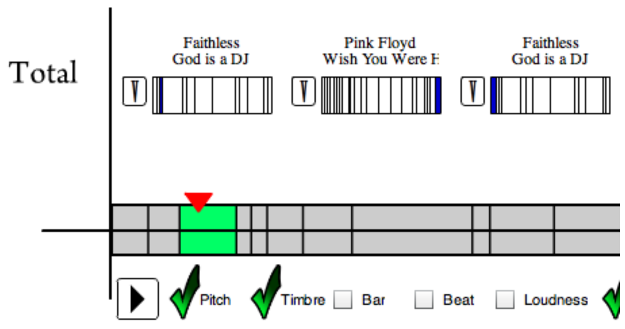


Figure 2. Detail of figure 1: The same song might be represented by several sections

Shades of Music can be used as a web-based radio: If one song is over, the system automatically picks the overall most similar song and starts playing that (which makes the first chosen song the seed song of the playlist[17]). With our similarity metric (see below) being symmetrical, this would lead to two similar songs playing in an endless loop (as the one most similar to the first would in turn have the first song as its top candidate). Therefore, the system only plays each song once. The user can of course also use the system to actively navigate her or his collection: Upon double-clicking one of the suggested songs, the system starts playing it.

3.2 Segmentation

Separating music into relevant subsections is a topic of active research. Methods learned from extracting representative audio thumbnails [1] can also be used to analyse the structure of an audio source [18]. Echo Nest is a web service that provides among others such an analysis for audio data. Besides retrieving meta-data for songs and values such as their current popularity (based on mentioning on webpages), it also performs segmentation and analysis of songs. Details can be found in [19].

One useful feature in our case is the automatic division into longer *sections* of several seconds length (e.g., verse or chorus) and very short *segments* that form short stable elements of a song. For each of these segments, Echo Nest returns a value for variations in loudness plus a chroma vector for pitch and another twelve-dimensional vector for timbre. The pitch chroma vector reflects the relative distribution of the acoustic content along the twelve semitones, while the timbre vector tries to capture the spectral surface of sound in an Echo Nest specific format with weights for twelve basis functions [2].

Additionally, positions of beats and bars for the whole song can also be retrieved. To calculate the similarity between sections of songs, we use the following procedure: First, the positions of segments and sections are retrieved. The longer sections form the basis for the comparison and are displayed in the interface as separate areas. Two sections' beat- or bar-wise similarity is determined by counting the number of beats and bars within a section and comparing these numbers.
For all segments within one section, values for changes in loudness, pitch and timbre are available for a more sophisticated comparison. Variations in loudness can be very easily compared by calculating their difference in decibels. To compare the pitch and timbre vectors, the positions of the vectors within one section are averaged and the resulting vectors compared using the euclidean distance between them.

The final comparison value for two sections is formed by normalizing all five values (beats, bars, loudness, pitch, timbre) and calculating the average difference, which leads to a final similarity between 0 and 1.

This very simple algorithm provides an initial comparison that is sufficient for our purposes, as the given values can be adjusted by the users anyway. Adding weights to the different features could also improve the classification, but since this would need more fine-tuning, at the moment all attributes have the same influence on the final result.

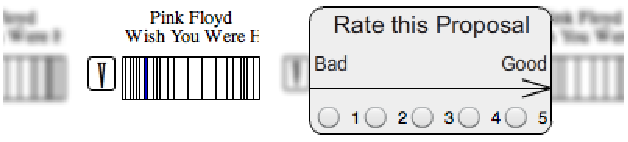


Figure 3. User feedback for one suggestion of the system

3.3 User Feedback

Automatically extracted similarity naturally has its limits. Although the hypothesized glass ceiling [20] for content-based extraction might be circumventable [21], some inherent problems will remain: Especially the issue of personalization is crucial. One user's idea of similarity might completely differ from another's who has a different taste in music or a more sophisticated sense for it. Thus, we are convinced that a metric based on automation is only a first step. For a final classification, user input has to be incorporated into the interface and its underlying algorithms. Last.fm [15] is a prominent example of a robustly classified music library based on user feedback.

3.3.1 Ratings for the automatic suggestions

In its current version, Shades of Music provides a very straightforward mechanism with which the user can correct the suggestions of the system. In the general play view, each song has a button to the left of its icon that makes a small window pop up (see figure 3). Here, the user can rate the suggestion made by the system on a scale from 1 to 5. As songs can appear more than once in the list of suggestions (for example, if the current song section corresponds to the repeated chorus of the other song) and even in several lists (for example for beat and pitch), the user can also criticize certain suggestions while promoting others. This means that the feedback is very specific and doesn't simply rate the computed similarity, but actually the aspect on which it was based.

If the user makes the effort to actually rate a suggestion, this overrides the respective computed value. The user is unable to see the actual internal similarity values and is shown the most similar sections only, so a negative rating always results in a reduction of the calculated similarity (and possibly a removal of the rated section from the list). Therefore, a vote replaces (for the user who made it) the initial similarity calculated by the system for the two sections concerned. The rating of a specific aspect is interpreted as a similarity of 0.0 (1), 0.25 (2), 0.5 (3), 0.75 (4) or 1.0 (5) and stored in the database. If the user votes on the total cumulative value, the rating is used as a factor for all the other attributes, so that their average corresponds with the rating value.

3.3.2 Incorporation of multiple users and feedback

From an initial five-dimensional metric of similarity between sections, the additional user feedback leads for a number of users to a higher-dimensional similarity. In the simplest case, only one user accesses the system and uploads songs from her or his own collection. The system calculates similarity values for existing sections and the user rates these suggestions as replacements for the automatically extracted similarity. In the end, the system reaches an optimal suggestion for this theoretical single user (of course with the overhead of rating millions of section combinations).

As Shades of Music is a web-based system, it is inherently targeting multiple users who all upload their own songs. This is used to reduce the analytical overhead by using meta-data to identify identical songs within separate collections. For these songs, existing classification data can be used. To counter erroneous meta-data, audio thumbnails could also be used for identification as the data is extracted anyway. Previous ratings by other users work as a refinement of the system-generated similarity: All ratings for an attribute of a pair of sections are again converted to a similarity value and, together with the system-generated one, averaged to reach a final value. In this way, we are able to improve suggestions even for new users (as long as they upload existing songs which were already rated by other users). Once a user starts rating suggestions within her or his own collection, these ratings are of course again directly applied (see 3.3.1).

4. SUMMARY AND FUTURE WORK

We presented Shades of Music, a web-based system that lets users discover connections between parts of songs within their music collection. For an exemplary song, a number of similar song sections are displayed, regarding the five attributes beats, bars, loudness, pitch and timbre and an average total. The user can give a rating for a suggestion and thus improve the system's results for himself and others. Informal first feedback showed great potential for the application as especially users with large song collections were curious what connections might be discovered. As a user study for our system should show the merits of the underlying idea and not, for example, the usability of the interface, we plan to open the system for multiple users over a longer period of time and collect our observations. In this way, we will also be able to investigate the value of the integrated rating system.

Extensive testing showed that our prototype also has some shortcomings. First of all, we used a rather simple and not state-of-the-art algorithm for calculating the similarity between sections. When improving this, we would also address the lack of scalability caused by the pair-wise comparison of sections, for example by indexing [22]. With our initial test set of ten songs and an average number of twenty extracted sections, adding one song already leads to a total of 20.000 comparisons (4.000 for each of the five attributes).
The user interface can also be improved in several ways: The representation of the current song as grey section blocks does not help in understanding its structure. Labels with 'verse' or 'chorus' might help, but automation to do that is probably not feasible. Heuristics, such as "repeated sections are a chorus" will most likely be insufficient. Interface elements for labelling could be included to let users do that (and maybe also add the lyrics to the song for additional orientation).

The ways in which users are able to give feedback could also be expanded: Adjustment of section borders or suggestion of new songs (or sections) are only two ideas. Based on our algorithm of averaging all users' votes and the Echo Nest value for novel users we also face the problem of changing suggestions if new votes arrive. To avoid confusing our users with ever-changing suggestions, it might help to only initially use this method and don't update the results every time the interface is launched. Finally, with the generated database of related song sections, additional projects are also feasible: Novel visualizations for a global music collection as a network of interconnected song sections could prove interesting just as clustering the user community ("neighbors" in Last.fm) based on their votes.

5. ACKNOWLEDGMENTS

This work was funded by the University of Munich and the state of Bavaria. We would like to thank the members of the echonest.com forum for valuable feedback and support.

6. REFERENCES

[1] M.A. Bartsch, and G.H. Wakefield: "To catch a chorus: Using chroma-based representations for audio thumbnailing," *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pp. 15–18, 2001.

[2] The Echo Nest, 2009-05-11, http://the.echonest.com

[3] Y.H. Tseng: "Content-based retrieval for music collections," *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 176–182, 1999.

[4] W.H. Tsai, H.M. Yu, and H.M. Wang: "A query-by-example technique for retrieving cover versions of popular songs with similar melodies," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 183–190, 2005.

[5] A. Kapur, M. Benning, and G. Tzanetakis: "Query-by-beat-boxing: Music retrieval for the DJ," *Proceedings of the International Conference on Music Information Retrieval*, pp. 170–177, 2004.

[6] G. Tzanetakis, A. Ermolinskyi, and P. Cook: "Beyond the query-by-example paradigm: New query interfaces for music information retrieval," *Proceedings of the 2002 International Computer Music Conference*, pp. 177–183, 2002.

[7] E. Pampalk, A. Rauber, and D. Merkl: "Content-based Organization and Visualization of Music Archives," *Proceedings of the tenth ACM international conference on Multimedia*, pp. 570–579, 2002.

[8] F. Morchen, A. Ultsch, M. Nocker, and C. Stamm: "Databonic visualization of music collections according to perceptual distance," *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.

[9] P. Knees, M. Schedl, T. Pohle, and G. Widmer: "An innovative three-dimensional user interface for exploring music collections enriched with meta-information from the web," *Proceedings of the ACM Multimedia*, pp. 17–24, 2006.

[10] R. van Gulik, F. Vignoli, and H. van de Wetering: "Mapping music in the palm of your hand, explore and discover your collection," *Proceedings of the 5th ISMIR Conference*, 2004.

[11] M. Goto, and T. Goto: "Musicream: New music playback interface for streaming, sticking, sorting, and recalling musical pieces," *Proceedings of the 6th International Conference on Music Information Retrieval*, pp. 404–411, 2005.

[12] F. Vignoli, and S. Pauws: "A music retrieval system based on user-driven similarity and its evaluation," *Proceedings of 6th ISMIR Conference*, pp. 272–279, 2005.

[13] G. Adomavicius, and A. Tuzhilin: "Toward the next generation of recommender systems: A survey of state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 6, pp. 734–749, 2005.

[14] K. Hoashi, K. Matsumoto, and N. Inoue: "Personalization of user profiles for content-based music retrieval based on relevance feedback," *Proceedings of the eleventh ACM internation conference on Multimedia*, pp. 110–119, 2003.

[15] Last.fm, 2009-05-11, http://www.last.fm

[16] Who Sampled?, 2009-05-17, http://www.whosampled.com

[17] E. Pampalk, T. Pohle, and G. Widmer: "Dynamic playlist generation based on skipping behavior," *Proceedings of the 6th ISMIR Conference*, pp. 634–637, 2005.

[18] J. Paulus, and A. Klapuri: "Music structure analysis by finding repeated parts," *Proceedings of the 1st ACM Workshop on Audio and music computing multimedia*, pp. 59–68, 2006.

[19] T. Jehan: *Creating Music by Listening*, PhD Thesis in Media Arts and Sciences. MIT, 2005.

[20] F. Pachet, and J.J. Aucouturier: "Improving timbre similarity: How high is the sky?," *Journal of negative results in speech and audio sciences*, Vol. 1, No. 1, 2004.

[21] T. Lidy, A. Rauber, A. Pertusa, and J. M. Inesta: "Improving Genre Classification By Combination of Audio and Symbolic Descriptors Using a Transcription System," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 61–66, 2007.

[22] R. Cai, C. Zhang, L. Zhang, and W.-Y. Ma: "Scalable Music Recommendation by Search," *Proceedings of the 15th international conference on Multimedia*, pp. 1065–1074, 2007.

A COMPARISON OF SCORE-LEVEL FUSION RULES FOR ONSET DETECTION IN MUSIC SIGNALS

Norberto Degara-Quintela, Antonio Pena
Department of Signal Theory and Communications
University of Vigo, Spain
{ndegara, apena}@gts.tsc.uvigo.es

Soledad Torres-Guijarro
Laboratorio Oficial de Metroloxía
de Galicia (LOMG), Spain
storres@lomg.net

ABSTRACT

Finding automatically the starting time of audio events is a difficult process. A promising approach for onset detection lies in the combination of multiple algorithms. The goal of this paper is to compare score-level fusion rules that combine signal processing algorithms in a problem of automatic detection of onsets. Previous approaches usually combine detection functions by adding these functions in the time domain. The combination methods explored in this work fuse, at score-level, the peak score information (peak time and onset probability) in order to obtain a better estimate of the probability of having an onset given the probability estimates of multiple experts. Three state-of-the-art spectral-based onset detection functions are used: a spectral flux detection function, a weighted phase deviation function, and a complex domain detection function. Both untrained and trained fusion rules will be compared using a standard data set of music excerpts.

1. INTRODUCTION

The automatic detection of onsets is essential in many applications, including a number of important music information retrieval (MIR) tasks. Onset detection is useful in the analysis of the temporal structure of music as, for example, beat tracking and tempo induction, but it is also important in other relevant tasks such as melody, bass-line and chord extraction.

Finding automatically the starting time of audio events is a difficult process and many onset detection methods exist [1–3]. However, the performance of current detection methods is highly dependent on the nature of the signal as shown in [1]. The reason is that onset detection techniques assume an implicit nature or probability model for the signal to be analyzed. Actually, several well known algorithms can be described in terms of an implicit probability model of the signal [4].

For this reason, it is not expected that a single method will perform accurately for strongly nonstationary signals and audio signals are intrinsically variable in nature. Instead of designing a very complex algorithm, a promising development lies in the combination of multiple methods [5]. In fact, this is most likely the way human perception seems to work [6], using different processing principles for the same purpose so when one of them fails perhaps another succeeds.

Methods that combine time-domain onset detection functions to provide with a more accurate detection have been proposed. However, most of the existing combination schemes use ad-hoc approaches that, for example, choose a particular detection function between two different functions based on the type of onset [7] or a quality measure [8].

Recently, onset detection systems based on machine learning algorithms have been developed. In [9] two Gaussian Mixture Models are used to merge multiple audio features, but the combination of the individual detection functions is still done by a linear weighted sum of the time domain functions. Other approaches merge the detection functions using a time-delay neural network [10, 11].

The integration of tools and information is one of the significant challenges for the field of MIR as discussed in [12] and fusion methods can potentially be used for this purpose. Fusion is an important research area that studies the combination of multiple sources of knowledge to obtain more reliable information [13, 14].

This paper emphasizes the use of information fusion methods to gather the efforts of MIR community which develops multiple signal processing algorithms for the same purpose. In particular, we compare the use of untrained and trained fusion rules to combine, at score-level, the peak information obtained from three spectral-based onset detection functions. Scores represent the estimated time instant and the probability of having an onset at that instant. Hence, our multiple-expert approach aims to calculate a better estimate of that probability given the probability estimates (scores) of multiple experts, which is radically different to adding time-detection functions as previous approaches do. This study is the first work, to our knowledge, that focuses just on the combination of techniques by introducing score-level fusion for onset detection, opening a novel direction to address the problem of combining detection algorithms.

Section 2 introduces the fusion approach to onset detection, describing the structure of the system and the detection functions extracted. Section 3 describes the dataset

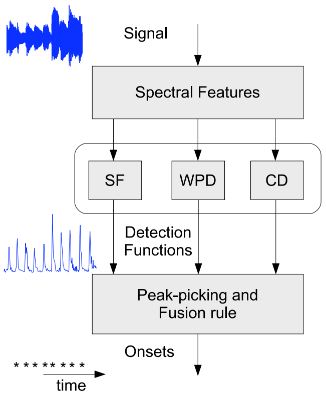


Figure 1. The Multiple-expert paradigm. The system fuses the peak information extracted from three detection functions: the spectral flux measure (SF), the weighted phase deviation (WPD) and the complex domain method (CD).

and the evaluation measures used in the present work. Results are presented in Section 4. And finally, Section 5 contains the conclusions and some ideas for future work.

2. FUSION FOR ONSET DETECTION

Fusion is an important and widely studied area that focuses on the issue of how to combine information to achieve an improved performance. This multiple expert paradigm is based on the combination of various diagnosis to exploit the expertise of the different experts. Score-level fusion combines the different opinions (probability estimates) of the experts to obtain a better estimate of the appropriate a posteriori probability.

Figure 1 shows the multiple expert fusion system that combines the peak information obtained from three state-of-the-art onset detection algorithms. First, the spectrum of the audio signal is calculated using the Short Time Fourier Transform (STFT). Then, three experts derive the detection functions using features extracted from the STFT. Finally, the system combines the peak information obtained from the detection functions using a fusion rule.

2.1 Onset Detection Functions

The detection functions used for fusion in this work are the following spectral-based reduction methods: the spectral flux measure (SF), the weighted phase deviation (WPD) and the complex domain method (CD) described in [2].

All these methods are based on a STFT scheme that applies a Hamming window $h(n)$. Given an audio signal $x(n)$ sampled at $f_s = 44.1$ kHz, the kth frequency bin of the nth spectrum frame $X(n,k)$ is given by:

$$X(n,k) = \sum_{m=-\frac{N}{2}}^{m=\frac{N}{2}-1} x(nh+m)h(m)\exp^{-\frac{j2\pi km}{N}} \quad (1)$$

In our experiments, the window size in samples is $N = 2048$ (46 ms) and the hop size $h = 441$ (10 ms).

The spectral flux (SF) measures the distance between successive short-time Fourier spectra:

$$SF(n) = \sum_{m=-\frac{N}{2}}^{m=\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|) \quad (2)$$

where $H(x) = \frac{x+|x|}{2}$ is a half-wave rectifier. This function is used to emphasize onsets rather than offsets since the sum is restricted to those frequencies where the spectral difference is positive and an increase of energy exists.

In order to add phase information in this system of multiple experts, the weighted phase deviation reduction method has also been considered. The rate of change of phase is an estimation of the instantaneous frequency and abrupt changes in the instantaneous frequency may suggest a potential onset. The weighted phase deviation (WPD) reduction method takes the mean of the absolute value of the instantaneous frequency difference weighted by the magnitude of the spectra:

$$WPD(n) = \frac{1}{N}\sum_{m=-\frac{N}{2}}^{m=\frac{N}{2}-1} |X(n,k)||\varphi''(n,k)| \quad (3)$$

where $\varphi''(n,k)$ is the second derivative of the 2π-unwrapped phase of the Fourier spectra $X(n,k)$.

Finally, the complex domain detection function considers jointly both magnitude and phase to search for transients on the signal. The spectral component $X(n,k)$ can be predicted from the previous frame spectra magnitude and phase change:

$$\widehat{X}(n,k) = |X(n-1,k)|e^{\varphi(n-1,k)+\varphi'(n-1,k)} \quad (4)$$

The complex domain (CD) detection function is defined as the sum of the absolute deviations from the predicted spectral values $\widehat{X}(n,k)$,

$$CD(n) = \sum_{m=-\frac{N}{2}}^{m=\frac{N}{2}-1} |X(n,k) - \widehat{X}(n,k)| \quad (5)$$

Normalization is a key step in fusion, therefore each of the detection functions is normalized to have a mean 0 and standard deviation of 1.

2.2 The Multiple-expert Architecture

In this approach, where multiple algorithms are combined to accomplish the same goal and can potentially interact to adapt its behavior, the architecture is very important. In this sense, blackboard modeling, an approach taken from artificial intelligent systems, has been successfully applied to other relevant applications such as computational auditory scene analysis [15] and polyphonic music transcription [16]. In a blackboard model, experts communicate using a common database what allows to pursue multiple lines of analysis at the same time and to adapt the strategies to a particular problem context.

The multiple-expert approach described in this paper has been developed within a blackboard-agent framework. Although the number of experts used in this paper is small, the blackboard-agent framework will probably be useful when combining many more experts, by implementing top-down processing where results coming from fusion are fed back into experts to improve individual results.

2.3 Peak Selection

Peaks are selected from the onset detection functions by peak-picking the local maxima. We apply the peak-picking algorithm used in [2] to obtain the peak-score information used for fusion: the peak time and the estimated probability of having an onset at that time.

A peak at time $t = \frac{nh}{f_s}$ (where n is the current sample, h the hop length and f_s the sampling frequency) is chosen as a relevant peak if the peak is a local maximum and the detection function is larger than a threshold above the local mean of the detection function $f(n)$, this is:

$$f(n) \geq f(m) \text{ for } m \text{ such that } n - w \leq m \leq n + w \quad (6)$$

$$f(n) - \frac{\sum_{m=n-lw}^{n+w} f(m)}{lw + w + 1} \geq \delta \quad (7)$$

where w is the size of the window used to find local maxima, l is a weighting factor to calculate the mean over a larger range before the peak (emphasizing onsets rather than offsets) and δ is the threshold.

Peak scores are normalized by subtracting the calculated local mean to the peak value of the detection function as given in equation (7).

The values of the peak-picking parameters have a large impact on the results. Hence, we follow the approach chosen in [1] and [2] selecting the parameters that maximize the F-measure, a performance measure defined in Section 3.

2.4 Fusion

Onsets in the original signal are related to peaks in the detection functions, therefore the normalized peak scores and times pairs are selected by using the mean-filter peak-picking algorithm described above. Peak scores and time stamps from the three experts are grouped in time frames of 50 ms and 50% overlap. If a given expert proposes several peaks within the merging frame, the peak with the highest score is selected.

Let $F(l) = \{f_{sf} f_{pd} f_{cd}\}$ and $T(l) = \{t_{sf} t_{pd} t_{cd}\}$ be, respectively, the peak scores and time stamps for each expert in the grouping time frame l. The proposed system fuses this peak information using the rules described below and classifies the frame as an onset or non-onset frame. The parameters of the fusion algorithms are chosen so as to maximize the performance of the fusion system.

Voting is perhaps one of the oldest strategies for decision making. The voting mechanism counts the number of expert scores that are higher than a given threshold and a consensus pattern is applied. A grouping frame can be classified as an onset-frame if any, the majority or all (unanimity) the experts exceed the threshold.

The sum rule simply adds the normalized expert scores in the grouping frame to obtain a better estimate of the a posteriori onset probability. A frame is labeled as an onset-frame if the resulting sum score exceeds a threshold.

Trained fusion strategies are also explored in this paper. In particular, we evaluate the performance of a K-Nearest Neighbor (K-NN) rule and a Support Vector Machine (SVM) with RBF kernel using cross-validation. The parameters of the RBF kernel are selected using a grid-search technique.

Grouping peak information in overlapping time frames generates doubled detections, therefore the output of the fusion rule is post processed to remove doubled onsets.

3. DATASET AND EVALUATION METHODOLOGY

The evaluation of the proposed fusion approaches is performed using the annotated dataset used in [1]. The dataset is composed of excerpts of different musical styles classified into the following categories: pitched non-percussive (PN), pitched percussive (PP), non-pitched percussive (NP) and complex mixtures (CM). This allows to test the algorithms on different classes of audio signals. There is a total of 1060 onsets.

The majority of the literature reporting results on onset detection shows a lack of proper statistical evaluation. Few works report standard deviations to give an idea of the variability of the results and most of them rely on mean performances only. Fortunately, a proper statistical hypothesis testing methodology has been adopted in MIREX 2008.

Hence, we decided to segment the original signals into homogeneous folds to evaluate the accuracy of our system using K-fold cross-validation. Cross-validation allows the statistical evaluation of the performance measures, enabling the estimation of confidence intervals [17]. We used different cross-validation files for each category, with no overlap between folds. The number of folds were 14 (CM), 12 (NP), 12 (PN) and 14 (PP).

For the evaluation and comparison of onset detection algorithms three measures are usually considered: precision (P), recall (R) and F-measure (F). These evaluation measures are defined as:

$$P = \frac{n_{cd}}{n_{cd} + n_{fp}} \quad (8)$$

$$R = \frac{n_{cd}}{n_{cd} + n_{fn}} \quad (9)$$

$$F = \frac{2PR}{P + R} \quad (10)$$

where n_{cd} is the number of correctly detected onsets, n_{fp} is the number of false positives (detection of an onset when no ground truth onset exists) and n_{fn} is the number of false negatives (missed detections). Due to the reliability of hand-labeled annotations, a time tolerance of 50 ms is usually assumed. This means that an onset is considered to be correctly matched if the detected onset is within

50 ms of the ground truth onset time. In addition, we do not penalized merged onsets since we do not try to identify individual notes.

As discussed in Section 2.3, peak-picking and fusion rule parameters are chosen so as to maximize the F-measure, which assigns the same significance to false positives and false negatives.

4. RESULTS AND DISCUSSION

Table 1 compares the results of the best individual experts and the proposed fusion rules on the different datasets. We choose the best expert for comparison because fusion always performed better than the worst expert in our experiments. In addition, we want to show that fusion can obtain even better results than the best expert and that fusion performance is not limited by the worst expert.

Total performance do not show enough information to compare different approaches and a proper statistical analysis is essential to fully understand how the different methods perform. Hence, Table 1 shows mean values and the 95% confidence interval for the F-measure using cross-validation.

As it can be seen in this table, fusion rules are able to achieve better performance than the best of the experts. For the PN and PP datasets, the relative increase in F-measure is important considering that the performance of the best of the experts is already high. Hence, the accuracy of the fusion algorithms is not limited by the worst of the experts and fusion achieves an improvement in performance by exploiting consensus diagnosis of the three experts.

For the NP and CM cases, the increase in performance given by the fusion rules is not significant. In fact, the performance is limited by the number of false negatives because there is a number of onsets that are not detected for any of the the experts. To exploit the benefits of fusion, experts should be as diverse as possible meaning that onset detection functions should be accurate and should not make coincident errors.

It is noteworthy that fusion has reduced the F-measure deviation in the PN and PP datasets but is still large for the NP and CM datasets. A large deviation means that fusion obtains good results for some of the folds but the performance is very low for other folds. In this sense, the performance could potentially be increased if we were able to identify the quality of the detection functions and apply different fusion strategies based on this quality measure.

We turn now to discuss the different approaches for fusion. Simple fusion rules obtain better results than trained fusion rules. The size of the test sets is small and both the K-NN and SVM approaches suffer from overfitting. In addition, the SVM achieves better performance than the K-NN except for the CM case. Finally, the SVM achieves very good results for the PP case, probably because the number of samples required to learn the task of identifying PP onsets is low.

We followed the statistical evaluation methodology proposed in [17] and we assumed a t-distribution for the sample mean estimator of the F-measure (the number of folds for cross-validation was less than 30). However, performance depends on various factors such as the set size, composition and the choice of the samples. Another interesting accuracy measure would be the Weighted Error Rate (WER), widely used in biometrics. In this case, a specific method for the calculation of confidence intervals for the total WER, not the mean, is already defined in [18]. This method reduces the performance dependency of these factors. The WER, a error measure widely used in biometrics, is defined as:

$$WER(R) = \frac{f_n + Rf_p}{1 + R} \tag{11}$$

where f_n and f_p are the false negatives and positives rates. The parameter R allows to balance the significance of the false positives and false negatives in the error measure which could be of interest in some applications and useful to compare algorithms at different operating points. Therefore, the WER can be an appropriate measure for the statistical evaluation of music information retrieval experiments.

5. CONCLUSIONS AND FUTURE WORK

The originality of this contribution is the introduction of score-level fusion strategies for onset detection, looking at the problem of combining onset information as a multiple-expert fusion problem. Our approach aims to calculate a better estimate of that probability given the probability estimates of multiple experts, which is radically different to adding time-detection functions as previous approaches do. This study is the first work, to our knowledge, that focuses just on the combination of techniques by introducing score-level fusion for onset detection, opening a novel direction to address the problem of combining detection algorithms.

This paper compares untrained and trained fusion rules on four sets of different music styles. Results show how information fusion rules can lead to a higher performance when combining multiple signal processing algorithms designed for onset detection. However, the increase in performance seems to be not important if experts are not diverse. Simple fusion rules show better performance than trained rules due to, probably, the small number of samples available for training.

In addition, a performance measure widely used in biometrics has been proposed. The Weighted Error Rate allows to balance the significance of the false positives and false negatives in the error measure and a specific method for the calculation of the confidence intervals of the total error rate is already defined.

In future work, we will include more experts to exploit diversity in the information fusion process. In addition, the cross-validation analysis showed a high deviation of the F-measure for complex signals. This means that the performance of the experts is highly dependent on the conditions of the signal. To face this problem, we will explore quality-based information fusion which basically weights scores according to the quality of the expert's detection functions.

	PN data			PP data			NP data			CM data		
	P	R	**F**	P	R	**F**	P	R	**F**	P	R	**F**
B.E.	93.8	98.1	**95.7** ± **5.1**	97.4	98.5	**97.8** ± **1.7**	99.5	94.5	**96.7** ± **5.5**	89.4	89.6	**88.8** ± **6.7**
Vot.	99.1	95.6	**97.3** ± **2.4**	98.4	98.8	**98.6** ± **1.0**	96.9	96.7	**96.7** ± **5.7**	91.0	88.5	**89.2** ± **7.5**
Sum	99.1	95.6	**97.3** ± **2.4**	99.8	98.6	**99.2** ± **0.9**	98.0	94.6	**96.2** ± **5.5**	93.9	85.4	**88.9** ± **7.0**
KNN	91.4	96.4	**93.5** ± **4.3**	95.7	98.0	**96.7** ± **1.5**	94.2	92.6	**93.2** ± **8.0**	88.2	82.9	**84.3** ± **10**.4
SVM	92.2	98.1	**94.7** ± **5.6**	99.5	98.5	**99.0** ± **1.0**	96.8	95.6	**96.2** ± **6.3**	84.0	84.8	**83.5** ± **8.9**

Table 1. Performance comparison of the score-fusion rules and the best individual expert (B.E.), showing precision (P), recall (R) and F-measure (F), for the different data sets. The table shows the mean and 95% confidence interval for the F-measure using K-fold cross-validation.

We will also intend to use a larger dataset to avoid overfiting in trained fusion rules. Finally, we will consider information fusion in other relevant problems such as beat tracking and tempo induction.

6. ACKNOWLEDGMENTS

Thanks to Juan Pablo Bello for kindly providing the dataset. This work has been partially supported by the Spanish MEC, ref. TEC2006-13883-C04-02, under the project AnClaS3 "Sound source separation for acoustic measurements".

7. REFERENCES

[1] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *Speech and Audio Processing, IEEE Transactions on*, vol. 13, no. 5, pp. 1035–1047, 2005.

[2] S. Dixon, "Onset detection revisited," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 133–137.

[3] N. Collins, "A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions," in *In AES Convention 118*, no. 6363, 2005.

[4] S. Abdallah and M. Plumbley, "Probability as metadata: Event detection in music using ICA as a conditional density model," in *ica03*, Nara, Japan, Apr. 2003, pp. 233–238.

[5] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 5, pp. 1832–1844, 2006.

[6] A. Bregman, "Psychological data and computational asa," in *Computational auditory scene analysis*, D. Rosenthal and H. Okuno, Eds. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1998.

[7] R. Zhou, M. Mattavelli, and G. Zoia, "Music onset detection based on resonator time frequency image," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 1685–1695, 2008.

[8] Y. S. Wan-Chi Lee and C.-C. J. Kuo, "Musical onset detection with linear prediction and joint features," in *2007 MIREX contest results*, 2007.

[9] C.-C. Toh, B. Zhang, and Y. Wang, "Multiple-Feature Fusion Based Onset Detection for Solo Singing Voice," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, USA, September 2008.

[10] A. Lacoste and D. Eck, "A supervised classification algorithm for note onset detection," *EURASIP J. Appl. Signal Process.*, vol. 2007, no. 1, pp. 153–153, 2007.

[11] N. Degara-Quintela, A. Pena, M. Sobreira-Seoane, and S. Torres-Guijarro, "A mixture-of-experts approach for note onset detection," in *126th AES Convention*, Munich, Germany, May 2009.

[12] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[13] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, July 2004.

[14] J. Kittler, "Combining classifiers: A theoretical framework," *Pattern Analysis & Applications*, vol. 1, no. 1, pp. 18–27, March 1998.

[15] D. P. Ellis, "Prediction-driven computational auditory scene analysis," Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, 1996, i.

[16] J. P. Bello, "Towards the automated analysis of simple polyphonic music: A knowledge-based approach," Ph.D. dissertation, King's College London - Queen Mary, University of London, 2003, i.

[17] A. Flexer, "Statistical evaluation of music information retrieval experiments," *Journal of New Music Research*, vol. 35, no. 2, pp. 113–120, June 2006.

[18] N. Poh and S. Bengio, "Estimating the confidence interval of expected performance curve in biometric authentication using joint bootstrap," Tech. Rep., 2006.

LYRIC-BASED SONG EMOTION DETECTION WITH AFFECTIVE LEXICON AND FUZZY CLUSTERING METHOD

Yajie Hu, Xiaoou Chen and Deshun Yang
Peking University
Institute of Computer Science & Technology
{huyajie,chenxiaoou,yangdeshun}@icst.pku.edu.cn

ABSTRACT

A method is proposed for detecting the emotions of Chinese song lyrics based on an affective lexicon. The lexicon is composed of words translated from ANEW and words selected by other means. For each lyric sentence, emotion units, each based on an emotion word in the lexicon, are found out, and the influences of modifiers and tenses on emotion units are taken into consideration. The emotion of a sentence is calculated from its emotion units. To figure out the prominent emotions of a lyric, a fuzzy clustering method is used to group the lyric's sentences according to their emotions. The emotion of a cluster is worked out from that of its sentences considering the individual weight of each sentence. Clusters are weighted according to the weights and confidences of their sentences and singing speeds of sentences are considered as the adjustment of the weights of clusters. Finally, the emotion of the cluster with the highest weight is selected from the prominent emotions as the main emotion of the lyric. The performance of our approach is evaluated through an experiment of emotion classification of 500 Chinese song lyrics.

1. INTRODUCTION

In order to organize and search large song collections by emotions, we need automatic methods for detecting the emotions of songs. Especially, they should work in small devices such as iPod and PDA. At present, much, if not most, research work on song emotion detection was concentrated on the audio signals of songs. For example, a number of algorithms [2,7,9] that classify songs from their acoustic properties were developed.

The lyric of a song, which will be heard and understood by listeners, plays an important part in determining the emotion of the song. Therefore, detecting the emotions of the lyric effectively contributes to detecting the emotions of the song. However, there is now comparatively less research done on methods for detecting the emotions of songs based on lyrics. There has been indeed a very large literature already out there on emotion analysis or opinion analysis of text. But, nearly all of them [1,3,6] use a one-dimensional model of emotions, such as positive-negative, which is not fine enough to represent lyric emotions which need more dimensions. Lyrics are much smaller in size than other kinds of text, such as Weblogs and reviews, and this makes it hard to detect lyrics' emotions. Being more challenging, lyrics are often abstract and in lyrics, emotions are expressed implicitly.

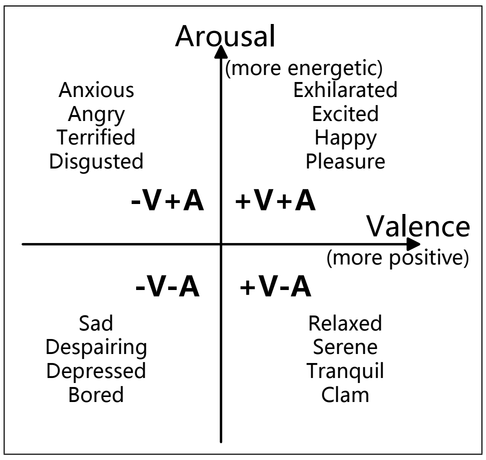


Figure 1. Russell's model of mood

We propose an approach to detecting the emotions of lyrics based on an affective lexicon. The lexicon is originated from a translated version of ANEW and then extended. According to the lexicon, emotion units(EUs) [13] of a sentence are extracted and the emotion of the sentence is calculated from those EUs.

A lyric generally consists of several sentences and those sentences usually expresses more than one emotions. In order to figure out all the prominent emotions of a lyric, we use a fuzzy clustering method on the sentences of the lyric. The method is robust enough to sustain the noises induced in previous processing steps.

In our approach, Russell's model of mood [11] is adopted, as shown in Figure 1, in which emotions are represented by two dimensions, *valence* and *arousal*. The lyric files we use are in `LRC` format [1] which have time tags in them and we got the `LRC` files from the Web. The framework of our approach is illustrated in Figure 2. It consists of three

[1] `http://en.wikipedia.org/wiki/LRC_(file_format)`

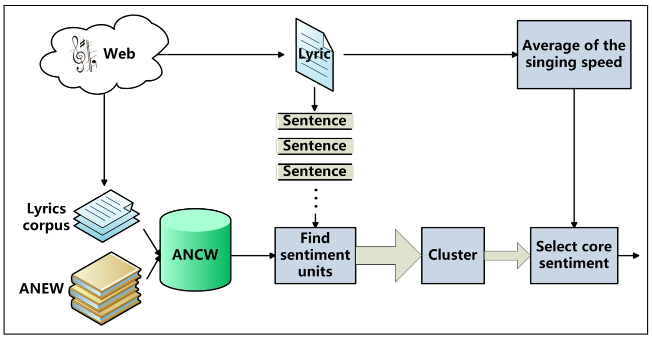


Figure 2. The framework of the proposed approach

main steps: (i) building the affective lexicon (ANCW); (ii) detecting the emotion of a sentence; (iii) integrating the emotions of all sentences.

The rest of this paper is organized as follows. In Section 2, the method for building an affective lexicon is presented. Section 3 describes the method for detecting the emotions of sentences. The approach to integrating the emotions of sentences is described in Section 4. Experiments and discussion are presented in Section 5. Finally, we conclude our work in Section 6.

2. BUILDING THE AFFECTIVE LEXICON

2.1 Translating the Words in ANEW

For analyzing the emotion of Chinese song lyrics, an affective lexicon called ANCW (Affective Norms for Chinese Words) is built from the Bradley's ANEW [4]. The ANEW list was constructed during psycholinguistic experiments and contains 1,031 words of all four open classes. As described in it, humans assigned scores to each word according to dimensions such as *pleasure*, *arousal*, and *dominance*. The emotional words in ANEW were translated into Chinese and these constitute the basis of ANCW. 10 people took part in the translation work. Each of them was asked to translate all the words in ANEW into Chinese words that he/she thought to be unambiguous and used often in lyrics. The Chinese word that was chosen by the largest number of translators for an ANEW word was picked and added into ANCW. A word may have more than one part of speech(POS), namely performs different functions in different context, and each may have a different emotion. Therefore, the part of speech of an ANCW word must be indicated. The words, the emotions of which in English culture are different from that in Chinese culture, are simply excluded from ANCW. To see if ANCW is consistent with ANEW, we use Meyers's method [10] to extend ANCW based on a corpus of People's Daily and the extended ANCW includes 18819 words. Meyers extends ANEW to a word list including 73157 words. The distributions of the emotion classes of the words in the extended ANCW is illustrated in Figure 3. We find that the emotion class distribution of the words in the extended ANCW is similar to the distribution of the words in the extended ANEW. This proves that ANCW is consistent with ANEW and is reasonable.

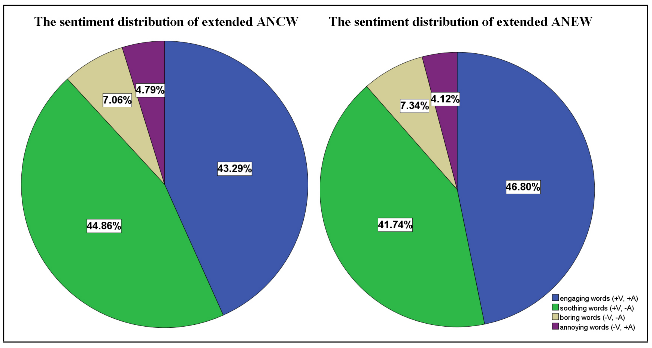


Figure 3. Distributions of the words in the extended ANEW and ANCW

Table 1. The origins of the words in ANCW

Origin	Translated from ANEW	Synonyms	Added by lyrics corpus
# of words	985	2995	71

2.2 Extending ANEW

However, the words translated from ANEW are not sufficient for the purpose of detecting emotions of lyrics so it is necessary to extend ANCW. We extend ANCW in two ways. In one way, with each word in ANCW as a seed, we find out all of its synonyms in TONG YI CI CI LIN [2]. Then, only synonyms with the same part of speech as that of their seed are added to ANCW. In the other way, we extract all constructions of apposition and coordination in a corpus of lyrics(containing 18000 Chinese lyrics) by an off-the-shelf natural language processing tool [8]. If either word in such a construction is in ANCW, its counterpart is added to ANCW. The origins of the words in ANCW is shown in Table 1 and valence-arousal distribution of the words in ANCW is illustrated in Figure 4. To indicate whether a word in ANCW is a translated word from ANEW or a later added word, we attach an *origin* property to each word. Therefore, terms in the affect lexicon have the following form: $< word, origin, POS, valence, arousal >$

3. DETECTING THE EMOTION OF A SENTENCE

First, word segmentation, POS annotation and NE recognition are performed for lyrics, with the help of the NLP tool. After stop words removed, the remaining words of a sentence are examined to see if they appear in ANCW, and each of the words that do appear in ANCW constitutes an EU. If there is an adverb that modifies or negates an emotion word, it is included in the corresponding EU as a modifier. We recognize the modifiers of EUs by using the NLP tool. The emotion of an EU is determined as follows:

$$v_u = v_{Word(u)} \cdot m_{Modifier(u),v} \tag{1}$$

$$a_u = a_{Word(u)} \cdot m_{Modifier(u),a} \tag{2}$$

[2] The lexicon of synonyms is manually built and includes 77,343 terms

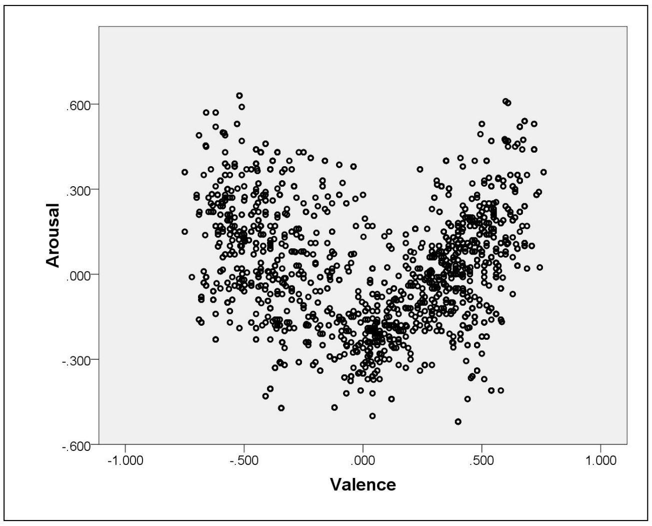


Figure 4. Valence-arousal distribution of the words in ANCW

Where v_u and a_u denote the valence and arousal value of EU u respectively, $v_{Word(u)}$ and $a_{Word(u)}$ denote the valence and arousal value of the EU's emotion word respectively, $m_{Modifier(u),v}$ and $m_{Modifier(u),a}$ denote modifying factors to represent the effect of the EU's modifier on the EU's valence and arousal respectively. $v_{Word(u)}$ and $a_{Word(u)}$, the valence and arousal value of the emotion word are obtained through looking up in ANCW. Sentences that have not any emotion unit are discarded.

We have collected 276 individual modifier words, which cover all the occurrences in the Chinese lyric corpus we use, and a table of modifiers has been set up. According to the polarities and degrees to which modifiers influence the emotions of EUs, we assign each modifier a modifying factor on valence and a modifying factor on arousal. The values of the modifying factors are in the range of $[-1.5, 1.5]$. For a negative modifier adverb, $m_{Modifier(u),v}$ is set to a value in $[-1.5, 0]$ and for a positive modifier adverb, it is set to a value in $[0, 1.5]$.

Tenses influence the emotions of sentences. Some sentences literally depict a happy life or tell romantic stories in one's memory but, actually, the lyric implies the feeling of missing the happiness or romances of past days. Similarly, the sentence with future tense sometimes gives the sense of expectation. Therefore, when we calculate the emotions of sentences, the influence of particular tenses are considered. We use Cheng's method [5] to recognize tenses of sentences and sentences are classified into three categories namely, *past*, *current* and *future*, according to their tenses. A sentence may have more than one EUs. Because the EUs of a sentence always have similar or even identical emotions, they can be unified into one in a simple way, as follows:

$$v_s = \frac{\sum\limits_{u \in U_s} v_u}{|U_s|} \cdot f_{Tense(s),v} \tag{3}$$

Table 2. Adjustment of w_u and r_u of unit u

	Increase when	Decrease when
w_u	u is after adversative words; u is after progressive words; u is in title.	u is before adversative words; u is before progressive words.
r_u	None.	The emotion word's origin is extended; The sentence is adjusted by tense.

$$a_s = \frac{\sum\limits_{u \in U_s} a_u}{|U_s|} \cdot f_{Tense(s),a} \tag{4}$$

where v_s and a_s denote the valence and arousal of sentence s respectively, U_s denotes the set of EUs of the sentence, v_u and a_u denote the valence and arousal of EU $u(u \in U_s)$ respectively, and $f_{Tense(s),v}$ and $f_{Tense(s),a}$ are modifying factors to represent the effect of the tense of the sentence on valence and arousal respectively. The values of the modifying factors representing the effects of tenses on emotions are in the range of $[-1.0, 1.0]$.

There are cases where two sentences(clauses) joined by an adversative or progressive word form an adversative or progressive relation. The following are two examples:
Adversative relation:

```
You are carefree
But I am at a loss what to do
```

Progressive relation:

```
Not only I miss you
But also I love you
```

Adversative and progressive relations in lyrics tend to remarkably affect the strength of involved EUs in determining the emotions of lyrics. Specifically, an emotion unit following an adversative word in a lyric influences the emotion of the lyric more significantly than a unit before an adversative word does. For example, the EUs in the sentence before `but` is given less weight, while the EUs of the sentence after the adversative word is given more weight. Similarly, in a progressive relation, the emotion unit after the progressive word is thought to be more important. So, a *weight* property is introduced for an EU to represent its strength of influence on lyric emotions. The initial value of weight of an EU is set to 1. A *confidence* property is also attached to an EU. If the emotion word of an EU is a later added word in ANCW, its confidence will be decreased. Also, if the emotion of a sentence is adjusted due to a particular tense, the confidence of its EUs will be decreased. The initial value of confidence of an EU is set to 0. The details of how to adjust the values of the weight and confidence of an EU are shown in Table 2. Accordingly, properties *weight* and *confidence* are also introduced for a sentence, which are calculated from that of its EUs in a simple way as follows:

$$w_s = \sum_{u \in U_s} w_u \tag{5}$$

$$r_s = \sum_{u \in U_s} r_u \tag{6}$$

where w_s and r_s denote the weight and confidence of sentence s respectively, and w_u and r_u denote the weight and confidence of EU u respectively. w_s and r_s are used to determine the main emotion of a lyric in the following processing.

4. INTEGRATING THE EMOTIONS OF ALL SENTENCES

4.1 Challenges

1. Reduce the effect of errors in sentence emotions on the result of the emotions of lyrics.
2. Recognize all the emotions of a lyric on the condition that the lyric has more than one emotion.
3. Select one emotion as the main emotion, if needed, or give a probability to each of the emotions.

4.2 Methodology

In recent years, spectral clustering based on graph partition theories decomposes a document corpus into a number of disjoint clusters which are optimal in terms of some predefined criteria functions. If the sentences of a lyric are considered as documents and the lyric is regarded as the document set, the document clustering technology can conquer the above three challenges. We define an emotion vector space model, where each sentence of a lyric is considered as a node with two dimensions that represent the valence and arousal of an emotion respectively. We choose Wu's fuzzy clustering method [12] because it can cluster the sentences without the need to specify the number of clusters, which meets our demands. Wu's fuzzy clustering method includes three steps: building a fuzzy similarity matrix, generating a maximal tree using Prim algorithm and cutting tree's edges whose weight is lower than a given threshold.

A song usually repeat some sentences. Sometimes the repeated sentences are placed in one line, with each sentence having its own time tag. In other cases, each repeated sentence occupies one line and the line has one time tag. If the repeated sentences are placed in more than one lines, these sentences are bound to form a cluster in the later clustering processing. If the emotions of those repeated sentences were not recognized correctly, subsequent processing will be ruined definitely. Hence, before sentences are clustered, lyrics should be compressed so as to place the iterative sentences in one line, with each sentence having its own time tag.

Having examined hundreds of lyrics, we find that sentences in a lyric always fall into several groups. The sentences of a group have similar emotions which can be unified to a prominent emotion of the lyric. Therefore, the isolated sentences are mostly noises and will be removed.

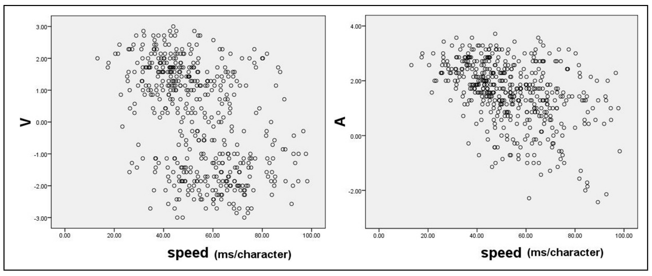


Figure 5. Distribution of speed, V and A

There are a dozen of means to measure the similarity between two nodes in vector space. After experiment those means, we select the following means to measure the similarity of the sentences' emotions i, j.

$$Sim_{ij} = 1 - \sigma(|v_i - v_j| + |a_i - a_j|) \tag{7}$$

where v_i,v_j,a_i, and a_j denote the valence and arousal of sentences i and j respectively, and σ is set to 0.3.

The center of a survived cluster is calculated as the weighted mean of emotions of all members of the cluster. The weighted mean is defined as follows:

$$v_c = \frac{\sum\limits_{s \in S_c} v_s \cdot w_s}{|S_c|} \tag{8}$$

$$a_c = \frac{\sum\limits_{s \in S_c} a_s \cdot w_s}{|S_c|} \tag{9}$$

where S_c denotes the set of sentences in cluster c, v_c and a_c denote the valence and arousal respectively of cluster c, and v_s, a_s and w_s denote the valence, arousal and weight respectively of sentence $s(s \in S_c)$.

The weight of cluster c is calculated as follows:

$$w_c = \sum_{s \in S_c} \frac{(\alpha \cdot w_s + \beta \cdot Loop(s))}{-\gamma \cdot r_s + 1} \tag{10}$$

where $Loop(s)$ denotes the number of times sentence $s(s \in S_c)$ repeats, α, β and γ are set to 2, 1, 1, respectively. These constant parameters are adjusted through experimentation and the set of values resulting in the highest F-measure was chosen.

Lyrics we got have time tags and we use these tags to compute the singing speed of sentences in lyrics, which is defined in milliseconds per word. Although, singing speed is not the only determinant of the emotions of lyrics, there is correlation between the singing speed of a song and its emotions, as shown in Figured 5. Hence, we use singing speeds of sentences to re-weight each clustering center. Having analyzed the singing speeds and emotions of the songs in the corpus, we think that Gaussian Model is suitable for expressing the degrees to which different singing speeds influence emotions. The re-weighting is considered as follows:

Table 3. The distribution of the songs corpus

Class	+V,+A	+V,-A	-V,-A	-V,+A
# of lyrics	264	8	174	54

$$w'_c = w_c + \frac{M}{\sqrt{2\pi}\sigma} e^{-\frac{(Speed(c)-\mu_v)^2}{2\sigma^2}} + \frac{M}{\sqrt{2\pi}\sigma} e^{-\frac{(Speed(c)-\mu_a)^2}{2\sigma^2}} \quad (11)$$

$$M = \max(w_c \,|c \in Lyric) \quad (12)$$

where the μ_v and μ_a are the offset of v and a, respectively. The meaning of σ is the variance of the speed of lyrics. $Lyric$ is the set of emotion clusters of a lyric. $Speed(c)$ is the average speed of sentences in cluster c. Finally, the clustering center with the highest weight is considered the main emotion. If there is a need for the possibility of several emotions, the possibility is computed as follows:

$$p\left(c\right) = \frac{w'_c}{\sum\limits_{c \in Lyric} w'_c} \quad (13)$$

5. EXPERIMENTS

Our ultimate goal is to compute the valence and arousal value of lyrics, not to do classification. We do classification for broad classes for the purpose of evaluating our emotion detecting method and comparing the performance of our method with that of other classification methods proposed in the literatures, many of which were for the same broad classes.

5.1 Data Sets

To evaluate the performance of our approach, we collected 981 Chinese songs from the classified catalogue according to emotion in `www.koook.com`. These songs are uploaded by netizens and their genres include pop, rock & roll and rap. These songs were labeled by 7 people whose ages are from 23 to 48. Two of them are professors and five are postgraduate students, all native Chinese. Each judge was asked to give only one label to a song. The songs that are labeled by at least 6 judges to the same class are remained. We use these songs' lyrics as the corpus. The distribution of the corpus in four classes is shown in Table 3. Although the number of songs in +V-A class is small, it is not surprising. This phenomenon conforms to the distribution in reality.

5.2 Results

To demonstrate how our approach improves the emotion classification of lyrics in comparison to existing methods, we implemented a emotion classification method based on lyrics with emotion lexicon: Lyricator [10]. Lyricator uses ANEW to extend the emotion lexicon by natural language corpus with a co-occurrence method. Using the extended emotion lexicon, Lyricator computes the emotion of each sentence of a lyric and the sentence emotion is the mean of emotion values of the emotion words contained in the sentence. The emotion of a lyric is weighted mean of values of the emotions of sentences. The weight is defined as the loop of sentences in the lyric.

Table 4. Evaluation results of Lyricator and our work

Class		Lyricator	Our work
+V+A	Precision	0.5707	**0.7098**
	Recall	**0.7956**	0.6856
	F-measure	0.6646	**0.6975**
+V-A	Precision	0.0089	**0.0545**
	Recall	0.1250	**0.7500**
	F-measure	0.0167	**0.1017**
-V+A	Precision	**0.6875**	0.6552
	Recall	0.0632	**0.3276**
	F-measure	0.1158	**0.4368**
-V-A	Precision	0.0000	**0.3125**
	Recall	0.0000	**0.2778**
	F-measure	0.0000	**0.2941**

To process Chinese lyrics, we translate the lexicon used in Lyricator and implement Lyricator's method. What's more, the parameters are adjusted to gain its best performance. Under the same test corpus that has been mentioned above, we compare Lyricator with our system. Table 4 shows the evaluation results between Lyricator and our work in the same songs corpus. The precision for a class is the number of lyrics correctly labeled the class divided by the total number of lyrics labeled as belonging to the class. The Recall is defined as the number of true positive divided by the total number of lyrics that actually belong to the positive class. The small number of lyrics in +V-A leads to the low precision for this class. Because we have used the wealth of NLP factors and fuzzy clustering method, our method's performance is better than the previous work.

5.3 Discussion

An analysis of the recognition results reveals the following findings:

1. Errors made by the NLP tool are especially salient because lyrics are very different from ordinary texts in word selection and arrangement. It is challenging for the NLP tool to do word segmentation, POS and NE recognition well. For example,
   `Hope desperation and helpless to fly away`
   the NLP tool considered terms "desperation" and "helpless" as verbs while they are actually norms. Without word lemmatization, recognizing POS of words in Chinese is much harder than in English. What's more, it will lead to errors in subsequent processing.

2. Some errors were due to complex and unusual sentence structures, which make it hard for our rather simple method to recognize emotion units correctly.

For example, the subject of a sentence is usually omitted due to the limitation of length of lyrics.

3. It seems that lyrics usually don't express much about *arousal* dimension of emotion. Experimental results show confusion rate between +A and -A is higher than that between +V and-V, suggesting that lyrics don't express much about *arousal* dimension.

4. The emotions of some lyrics were not explicitly expressed, and therefore deduced by human listeners based on his or her knowledge and imagination.

The following sentences come from a typical lyric, the emotions of which are not recognized correctly:
Do you love me? Maybe you love me.
Hanging your head, you are in silence.
Those sentences form the chorus of CherryBoom's *Do You Love Me* and they express intensive emotions. Although it is easy for human listeners to tell the emotions, it is quite difficult for a computer to detect the emotions only literally from the words of the lyric.

6. CONCLUSION

In this paper, we propose an approach to detecting emotions of songs based on lyrics. The approach analyzes the emotion of lyrics with an emotion lexicon, called ANCW. In order to obtain the emotion of a lyric from that of its sentences, we applied a fuzzy clustering technique which can reduce the effect of errors introduced in the process of analyzing emotions of sentences. Finally, we use the mean singing speed of sentences to re-weight the emotion results of clusters. The experimental result is encouraging.

Although this paper handles Chinese lyrics, we also implement an English version of emotion analysis system using English lexicon because our method is not specifically designed for Chinese environment. What's more, the method is unsupervised and training is not needed. Consequently, it takes about two seconds[3] to process a lyric and is apt to apply in small devices.

7. REFERENCES

[1] N. Archak, A. Ghose, and P. Ipeirotis. Show me the money! deriving the pricing power of product features by mining consumer reviews. *In Proceedings of The ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2007.

[2] D. Bainbridge, S. J. Cunningham, and J. S. Downie. Analysis of queries to a wizard-of-oz mir system: Challenging assumptions about what people really want. *In Proceedings of The 4th International Conference on Music Information Retrieval*, pages 221–222, 2003.

[3] M. Bansal, C. Cardie, and L. Lee. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. *In Proceedings of The International Conference on Computational Linguistics*, 2008.

[4] M. M. Bradley and P. J. Lang. Affective norms for english words (anew): Stimuli, instruction manual and affective ratings. Technical report, The Center for Research in Psychophysiology, University of Florida, 1999.

[5] J. Cheng, X. Dai, J. Chen, and Q. Wang. Processing of tense and aspect in chinese-english machine translation. *Application Research of Computer*, Vol 3:79–80, 2004.

[6] P. Chesley, B. Vincent, L. Xu, and R. Srihari. Using verbs and adjectives to automatically classify blog sentiment. *In AAAI Symposium on Computational Approaches to Analysing Weblogs*, page 27C29, 2006.

[7] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A music search engine built upon audio-based and web-based similarity measures. *In Proceedings of The 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 23–27, 2007.

[8] J. Lang, T. Liu, H. Zhang, and S. Li. Ltp: Language technology platform. *The 3rd Student Workshop of Computational Linguistic*, pages 64–68, 2006.

[9] B. Logan, D. P.W. Ellis, and A. Berenzweig. Toward evaluation techniques for music similarity. *In Proceedings of The 4th International Conference on Music Information Retrieval*, pages 81–85, 2003.

[10] Owen C. Meyers. A mood-based music classification and exploration system. Master's thesis, Massachusetts Institute of Technology, 2007.

[11] Russell and James A. A circumplex model of affect. *Journal of Personality and Social Psychology*, Vol 39(6):1161–1178, 1980.

[12] Z. Shi. *Knowledge Discovery*. Tsinghua University Press, 2002.

[13] Y. Xia, L. Wang, K. Wong, and M. Xu. Sentiment vector space model for lyric-based song sentiment classification. *In Proceedings of ACL-08: HLT, Short Papers*, pages 133–136, 2008.

[3] CPU: 400MHz; Memory: 128MB; OS: Windows Mobile 6.0

BROWSING MUSIC RECOMMENDATION NETWORKS

Klaus Seyerlehner
Dept. of Computational Perception
Johannes Kepler University
Linz, Austria
klaus.seyerlehner@jku.at

Peter Knees
Dept. of Computational Perception
Johannes Kepler University
Linz, Austria
peter.knees@jku.at

Dominik Schnitzer
Austrian Research Institute for AI
Vienna, Austria
dominik.schnitzer@jku.at

Gerhard Widmer
Austrian Research Institute for AI
Vienna, Austria
gerhard.widmer@jku.at

ABSTRACT

Many music portals offer the possibility to explore music collections via browsing automatically generated music recommendations. In this paper we argue that such music recommender systems can be transformed into an equivalent recommendation graph. We then analyze the recommendation graph of a real-world content-based music recommender systems to find out if users can really explore the underlying song database by following those recommendations. We find that some songs are not recommended at all and are consequently not reachable via browsing. We then take a first attempt to modify a recommendation network in such a way that the resulting network is better suited to explore the respective music space.

1. INTRODUCTION

Now that millions of songs are available for purchase and download on modern music platforms, developing concepts that help customers to navigate and explore the underlying song database becomes more and more important. A straight forward solution that is used in many commercial settings to assist users in finding songs in a database is to simply present lists of recommendations. Users are then able to explore a collection by moving from recommendation to recommendation. Exploring a music collection via such a sequence of recommendations is called *browsing*. We believe that browsing will be a key feature of modern music portals and consequently it is important to view recommendation not just in terms of individual recommendation queries only, but also as a continuous process. To analyze recommender systems with respect to their ability to support users to browse throughout a music collection, we can view a music recommender as a recommendation network. Recent research work on analyzing music recommendation networks [1, 2] indicates that many songs in such a network stay hidden in the so-called Long Tail [3,4]. One reason why songs stay hidden in the Long Tail is that it is hard to navigate through the network to reach those unknown songs. Thus, it seems to be an essential property of such a recommendation network that each song can be reached via browsing the recommendations. The goal of this paper is to analyze music recommendation networks with respect to their *browsability*.

The rest of this paper is organized as follows: In section 2 we start with formally defining the general recommendation scenario. In section 3 we show that under some restrictions any recommender system can be transformed into an equivalent *recommendation graph*. We then define properties for a recommendation graph that make such a graph useful for browsing the underlying music database and introduce the notion of a *browsing graph*. In section 4 an analysis of a recommendation graph of a real world content-based music recommender system illustrates the limitations of a simple recommender system with respect to the reachability of database items. We then propose in section 4.2 an algorithm which effectively modifies a recommendation graph to overcome these reachability limitations. Finally, we give an outlook on the application of the proposed method and some future work.

2. RECOMMENDATION SCENARIO

Although many different music recommender systems have been proposed so far, the fundamental principle is basically the same. Independent of the actual recommendation approach we can give a **formal model of a recommendation scenario** for item-based recommendation:

Given a set of database items U of size N and a specific item $o \in U$ that a user is currently focusing on, a *recommendation* is a subset of items $R \subset U$ related to o, where the size of the subset R is far smaller than the total number of items in the database. This very simple recommendation scenario can be extended by generating a recommendation not only based on the current item o but additionally specifying a user profile $p \in P$, where P is a set of all user profiles stored in the recommender system. We call a tuple $q = (o, p)$ a *recommendation query* and the item set $R(q)$ returned by the recommender system the *result set* or *recommendation*.

Actual recommender systems then differ in the way the recommendations are generated in this scenario. With respect to music recommender systems, there seem to exist five general recommendation approaches: collaborative filtering approaches, content-based approaches, web-mining based approaches, expert-based approaches and hybrid approaches.

Our investigations in the next sections are in general independent of the recommendation approach. The only requirement is that the recommender system under investigation returns, for any query $q(o, p)$, an ordered set of recommended items of a given length k such that the recommended items are ordered according to a measure of relatedness.

3. RECOMMENDATION GRAPHS

An intuitive way of exploring a music catalog is to pick an arbitrary item out of the database and than navigate throughout the database moving from recommendation to recommendation. One important requirement of such a browsing system is reachability. Reachability essentially ensures that a user will be able to access all songs in the collection by means of exploration and will not be limited to a small subset by the recommender system. To be able to show that reachability is ensured for a specific recommendation algorithm, we have to establish a formal model of the browsing process.

Based on our definition of a recommendation scenario (see section 2), *browsing* can be seen as an extension to recommendation from a single query to a consecutive sequence of queries $s = (q_1, q_{,2}, ..., q_N)$. Two consecutive queries $q_i = (o_i, p)$ and $q_{i+1} = (o_{i+1}, p)$ within such a browsing sequence are related by the fact that the item o_{i+1} of the next recommendation query q_{i+1} is an element of the result set of the previous recommendation query q_i. Consequently, a sequence s of recommendation queries of length N is a valid browsing sequence in the case that the following property is fulfilled:

$$\forall i < N : o_{i+1} \in R(q_i) \tag{1}$$

To guarantee this essential reachability property for a recommender system we have to show that starting from a arbitrary but fixed database item, all other database items can be reached by a finite sequence of recommendation queries. Formally, **reachability** starting from an arbitrary but fixed item o_1 holds if:

$$\begin{gathered}\forall o \in U : \exists i \in \mathbb{N} : \forall j < i : \\ o_{j+1} \in R(q_j) \wedge q_{j+1} = (o_{j+1}, p) \wedge o \in R(q_i)\end{gathered} \tag{2}$$

Before we can start drawing any conclusions about reachability, we have to make some additional assumptions about the recommender system. The reason is that for dynamic recommenders, e.g., based on collaborative filtering, where the recommendations may change as a result of system use, it is impossible to prove reachability, since we cannot make any assumption about future recommendations. Therefore we have to assume a *static* recommender system where the recommendation will not change over time. It is important to note that this is not in principle a loss in generality; it just implies that if there are any changes in the recommender system then we also have to prove reachability for this new recommendation state.

Furthermore we constrain our analysis to systems where the recommendation result is independent of the user profile. This implies that all users get the same recommendations for one and the same query item. Once more this is not in principle a loss in generality as we could handle such systems by proving reachability for each user separately. In practice, however, analyzing recommender systems that generate personalized recommendations seems to be impossible due to the potential enormous computational costs.

Given these restrictions, we can now transform every possible recommender system into a *recommendation network* or *recommendation graph*. A recommendation graph is a directed graph $G = (V, E)$, where each vertex in the graph corresponds to a database item. For each item o in the database the corresponding vertex in the graph has a directed edge to all the items in the result set $R(q)$ of the recommendation query $q = (o, p)$. (Note that based on our assumptions $R(q)$ does not depend on p, an optionally given user profile.) To prove reachability for such a recommendation graph we can for instance apply the *depth first search* algorithm for each vertex in the graph separately.

While this is not a very practical or fast method to prove reachability, in most cases it is quite trivial to disprove reachability either by showing that the recommendation graph is not connected, or by identifying a single *source*. A source is a vertex v which has no incoming edges, i.e., has an indegree of zero ($\deg^-(v) = 0$). This implies that there is a song in the database that does not occur in the result set of any possible recommendation query and is consequently not reachable at all. Sources are especially problematic with respect to browsing: not only are they not reachable if one starts from some specific song in the database, but they are not reachable from *any* other song in the database. In section 4 we show, based on empirical analysis of a real world music recommender system, that in contrast to what one would expect it is rather likely that there are many sources in a simple recommendation graph. Identifying sources in a graph is a fast operation and can be done in $O(n)$.

Proving and disproving reachability is of course an important analysis, however in the likely case that we are able to disprove reachability, what can we do about it? How can we find a recommendation algorithm that guarantees reachability? To put it another way, can we modify a recommendation graph in such a way that the recommendation graph guarantees reachability? In section 4 we will show that it is quite likely that a recommendation graph does not fulfill the reachability property. We then propose an algorithm that transforms a *recommendation graph* into a *browsing graph*, a recommendation graph that besides reachability has some other properties that we are going to introduce in the next section.

3.1 Further Requirements and Constraints

Up to now we have only considered reachability as an important property of a recommendation graph. But we can derive additional constraints for the recommendation graph by analyzing user requirements of browsing systems.

The first requirement that jumps to the eye is that the result set should be relatively small — first of all, because the display space for recommendations is in general limited on output devices, and secondly, because too large a result set would confuse the user and make for a very unfocused search. Thus it is a natural constraint that the size of the result set should not exceed a maximum number of recommendations $k_{\max}$. For the corresponding recommendation graph this implies that the outdegree of all vertices is less or equal to $k_{\max}$. We call this property **maximum outdegree property**.

$$\forall v \in V : \deg^{+}(v) \leq k_{\max} \tag{3}$$

The second constraint is that if item B is a recommendation for item A then item A should also be a recommendation of item B. This corresponds not only to humans' intuition that similarity relations are symmetric, but also allows to easily go back each recommendation step. The **symmetry property** as defined in (4) implies that the browsing graph is an undirected graph.

$$\begin{aligned} &\forall e_1 = (v_1, u_1) \in E : \\ &\exists e_2 = (v_2, u_2) \in E : \\ &\quad v_1 = u_2 \wedge u_1 = v_2 \end{aligned} \tag{4}$$

Finally, we extend our notion of reachability. Reachability just ensures that starting from an arbitrary vertex there is at least a single path to each other vertex. This could make it rather difficult to find this path. Therefore we require each vertex to have a minimum number of incoming edges. For the browsing graph this implies that each vertex has a minimum indegree $k_{\min}$ and means that each item is reachable by recommendations from at least $k_{\min}$ other items. This property is called **minimum indegree property**.

$$\forall v \in V : \deg^{-}(v) \geq k_{\min} \tag{5}$$

As a result from this requirement analysis we claim that a recommendation graph is better suited for browsing a music archive if these four properties are ensured. We then call such a graph no longer a *recommendation graph*, but a *browsing graph* instead.

In the next section we illustrate the limitations of a simple recommendation graph based on a real world content-based music recommender system and show that in most cases such a recommendation graph is not adequate for browsing. We then introduce a heuristic algorithm that can transform a recommendation graph into a browsing graph.

4. BROWSING GRAPHS

4.1 An Empirical Study

In this section we will show that properties like reachability are essential and cannot be neglected when designing a recommender or browsing system. To do so we analyze a real world content-based music recommender system attached to the music portal. The FM4 Soundpark [1] is an internet platform of the Austrian public radio station FM4. This internet platform allows artists to present their music free of any cost in the WWW. All interested parties can download this music free of any charge. At the moment this music collection contains about 10000 songs and is steadily growing. In our experiments we were allowed to use a subset of 7665 songs out of the whole collection.

The recommender system attached to the FM4 Soundpark music portal is based on a standard similarity measure for music audio files. Each song is modeled as a distribution of local spectral features, namely Mel Frequency Cepstrum Coefficients (MFCCs). MFCCs are a compact representation of the spectral envelope of a short audio frame and are one of the most widespread features used in the Music Information Retrieval (MIR) community. A single multivariate Gaussian distribution is used to model the distribution of MFCCs of a song. Recommendations can then be generated by comparing these distributions. This is commonly done by computing the Kullback-Leibler (KL) divergence [5] or relative entropy between the distributions of two songs. For more details on the feature extraction process and the generation of music recommendations we refer to [6–8]. Using the MIR system of the FM4 Soundpark we were able to generate lists of recommended songs of a given length k, ordered according to the similarity to the query song, exactly as required by our general scenario (see section 2).

Assuming a fixed sized result set of k recommendations for each query, we systematically created all recommendation graphs for $k = 1 \ldots 100$, where we denote k as the degree of the recommendation graph. For each of these graphs we computed the indegree for all vertices and counted the number of sources in each graph. Figure 1 shows that for small result sets the number of sources is extremely high. For example, in the recommendation graph of degree 5 there are 2661 sources, which implies that 34.72% of all the songs in the music collection are not reachable at all within this graph. By increasing the result set size the number of sources decreases, but even for a quite large result set of size 20 we still have approximately 1320 sources. Consequently still 17.22% of the songs in the collection cannot be reached. From figure 2 we can see how the number of sources scales with the collection size. To simulate different collection sizes songs were randomly removed from the collection. Figure 2 illustrates that the problem gets worse for increasing collection sizes. In fact the analysis of the recommendation graph that corresponds to the online version of the FM4 Soundpark — there are only three recommendations per song — revealed

[1] `http://fm4.orf.at/soundpark/main`

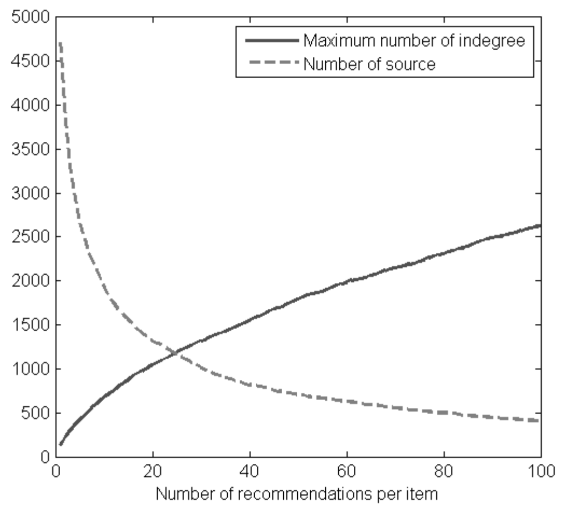


Figure 1. *For small result sets, the number of sources is extremely large and decreases with an increasing number of recommendations per query, whereas the maximum indegree over all vertices in each graph increases. For a result set size of 100, there is one song that appears in the recommendation list of 2628 other songs, or in 34.29% of all recommendation lists.*

that only 56,79% of all songs are reachable by recommendations, the remaining 43,21% of the songs are sources and are never recommended.

In addition to the number of sources, we also computed the maximum indegree over all vertices in each graph, visible in figure 1. Obviously, while some songs are not reachable at all, some others are directly reachable from very many songs. However it is of course quite implausible that a single song is similar to several hundred other songs. Songs that have a very high indegree, but do not share any perceptual similarity with the referring songs are called *hub*-songs according to [9]. In our case the hub problem seems to be related to the content-based audio similarity measure itself. Interestingly, hubs naturally appear in social networks (including collaboration networks) as well [10]. Regardless of the reasons for hubs and sources, both essentially reduce the usability of music recommender systems to explore the music spaces. In the following we propose a heuristic algorithm that transforms a recommendation graph into a browsing graph that fulfills the properties introduced in section 3.

4.2 Constructing a Browsing Graph

The main idea behind our approach is to transform a recommendation graph into a browsing graph, simply by replacing all directed edges by undirected edges and then iteratively (and heuristically) removing edges from the resulting graph such that the *maximum outdegree* and the *minimum indegree* property are satisfied for all vertices. The symmetry property is automatically ensured because the graph is undirected. Furthermore, reachability is guaranteed if the resulting graph is connected.

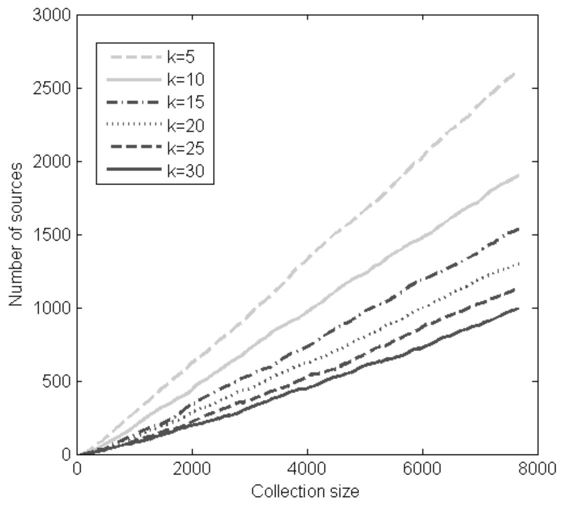


Figure 2. *The number of sources in a recommendation graph scales with the number of items in a database. Furthermore the number of sources depends on the number of recommendations for each query. This is illustrated for fixed result set sizes of* $k = 5, 10, 15, 20, 25, 30$.

The proposed algorithm has three important parameters. There is the minimum indegree $k_{\min}$ and the maximum outdegree $k_{\max}$, which directly result from the required properties. It is easy to see that in combination with the symmetry property this implies that each vertex in the final browsing graph will have to have an edge degree between $k_{\min}$ and $k_{\max}$. The proposed algorithm starts from the directed version of the recommendation graph. One could of course start the algorithm from a recommendation graph with outdegree $k_{\max}$, but since we want to give our algorithm additional flexibility during the process of removing edges, it is required that the original recommendation graph has an outdegree of at least k_{start} for all vertices. This simply means that for each item we can generate at least k_{start} recommendations and is in line with the requirement on recommender systems in section 2. The three parameters are related to each other as stated in (6).

$$k_{\min} < k_{\max} < k_{\text{start}} \tag{6}$$

The only thing left to do is to remove edges till each vertex has a degree in between $k_{\min}$ and $k_{\max}$. This should be done in such a way that each vertex tries to remove its 'weakest' links (i.e., those with the lowest degree of relatedness), since the recommendations should be as good as possible. This can be done as follows:

1. Put all vertices into a priority queue q, where all vertices are sorted according to their degree $\deg(v)$; break ties among same-degree nodes randomly;

2. Pop the vertex with the highest degree from the queue.

3. If this vertex already has a degree smaller than or

equal to $k_{\max}$, then all vertices in the queue have a degree smaller or equal to $k_{\max}$. We are done.

4. As the current vertex has too many edges, remove an edge that connects this vertex to another vertex having a degree greater than $k_{\min}$. Choose the edge to remove according to the indegree of the neighboring vertices. Remove the edge connecting to the vertex with the highest indegree and if there are several vertices of the same indegree remove the vertex with the weakest (lowest similarity) edge. If this vertex is not connected to any other vertex having a degree greater than $k_{\min}$, then we are not able to ensure the maximum indegree property for this node. Stop in this case.

5. Since we have removed an edge, the indegrees of the two vertices connected by the edge have changed. Remove them from the queue and reinsert them such that the queue is up to date.

6. Go back to step 2.

Of course it is true that this algorithm might find a solution where individual vertices have an edge degree higher than $k_{\max}$, violating the maximum outdegree property. This can be due to the fact that for given constraints there simply does not exist any solution. In such a case weakening the constraint till enough solutions to the problem exist can help. If there are enough solutions, simply rerunning the algorithm might help. Vertices of the same edge count are inserted into the priority queue in random order. Therefore the algorithm might find other solutions. However our experiments indicate that it is quite easy to find a valid solution. Furthermore, the proposed algorithm does not guarantee that the resulting graph is connected, but in all our conducted experiments the resulting browsing graph turned out to be connected.

4.2.1 Time Complexity

One major advantage of this algorithm is that it is of time complexity $O(n \log(n))$. At most $n(k_{\text{start}} - k_{\min})$ edges have to be removed. Therefore we have to perform a maximum of $3n(k_{\text{start}} - k_{min})$ removal or insertion operations on the sorted priority queue. Sorting and removing elements from a priority queue can be done in $O(\log(n))$, e.g., by using a balanced red-black tree. Therefore removing all the additional edges from the graph can be done in $O(n \log(n))$. The initial insertion operation of all elements in the priority queue is also of complexity $O(n \log(n))$. Thus, the overall complexity of this algorithm is $O(n \log(n))$.

4.2.2 Validation of the Transformation Algorithm

To validate the proposed algorithm we analyzed the result after the transformation of the FM4 Soundpark into a browsing graph. The parameters used to transform the graph were $k_{\min} = 4$, $k_{\max} = 7$ and $k_{\text{start}} = 9$. As we do not have yet statistics of the usage before and after the transformation, we follow the standard procedure in MIR research and evaluate transformation algorithm in an indirect way, via a music genre analysis. For all query songs q we count the number of songs in the result set $R(q)$ that have the same genre as the query song and compute the overall percentage relative to the number of recommended songs. That way we measure the accuracy of the recommendations independent of the number of the recommendations. The accuracy of the recommendations using result sets of length $k = 5$ was 35.39%, for $k = 6$ was 34.86% and for $k = 7$ was 34.32%. After the transformation using the above parameters the accuracy was 35.63% with an average degree of 5.918 per vertex. This preliminary result indicates that there is only a marginal change in recommendation quality, however a more detailed empirical study will be done in future. Furthermore to evaluate how

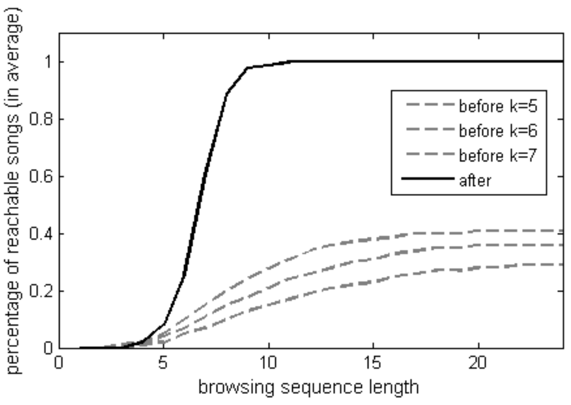


Figure 3. *The average percentage of songs that can be reached by browsing sequences of different length. Before the transformation (for $k = 5, 6, 7$) and after the tranformation.*

the reachability of songs has changed we investigated how many songs can be reached in average by a recommendation sequence of length l. To do so we computed for each song the number of songs that can be reached by such a sequence. This can be done by traversing the recommendation graph using the *breadth-first search (BFS)* algorithm up to a maximum depth of l. We then take the average over all songs to get a quality indicator for the whole network. As one can see from figure 3 after the transformation more songs can be reached when browsing the resulting graph than before.

5. APPLICATION AND FUTURE WORK

Based on the graph-theoretic studies performed on the FM4 Soundpark Recommender, we are now investigating ways of turning the Soundpark into a Browsing Graph. Given the purpose of the system – to make new music artists known to a wide public – reachability of as many artists (or works) as possible would be a prime feature. This is not quite straightforward and will involve some interesting research questions. Several aspects have to be addressed:

Recommendation quality: Clearly, the quality of the recommendations changes as a recommendation graph (which is based on content-based similarity relations) is transformed into a browsing graph (which sacrifices certain recommendation links in order to satisfy the browsing constraints). Whether or not that unduly degrades the quality of the recommendation service can only be studied empirically. We will address this issue by means of a large-scale user study, which is yet to be designed (see below).

Incremental updates: The FM4 music database grows on a daily basis. Every day, dozens of new songs, mostly by new artists, are added to the database and integrated into the recommender system in nightly batch update sessions. Thus, the browsing graph transformation will also have to be run at regular intervals. As an alternative, we will look into the possibility of incremental update algorithms for browsing graphs.

Time-varying recommendations: A specific aspect of the growing database is that the system's recommendations may change from day to day. That is, if the user selects the same seed song on two consecutive days, she may get different recommendations of songs that are supposedly 'similar'. This may be a problem in certain applications, but perhaps not in the case of the Soundpark. Soundpark users have been taught to regard the recommendation service as a means to explore the Soundpark and find new things that they would not otherwise find. From the user feedback we currently have, we can conclude that many of the users are quite open-minded about occasional 'strange' recommendations, regarding them as 'interesting' or 'funny' ideas by the computer, rather than annoying mistakes. Thus, they might find time-varying recommendations (if they ever notice them) to be enriching rather than irritating.

Modifications to the Soundpark recommender system will be accompanied with a large scale *user study*. We have access to two kinds of user feedback: the browsing sessions themselves (click data) as logged by the Soundpark server, and an on-line user forum, where users discuss their impressions of the system (among other things). Questions to be studied include, e.g., whether improved reachability conditions really increase the number of artists that are listened to by users; whether and how one can quantify differences in recommendation quality between recommendation and browsing graphs; and general aspects of user browsing behaviour that may help in designing better recommenders in the future (for instance: how long is a typical browsing sequence? do users follow more than one recommendation in a given recommendation list? etc.).

In this way, the FM4 Soundpark may then become one of the first real-world music recommendation system that is (a) purely content-based, that is, based on musical similarity as estimated by the system itself, and (b) specifically designed to maximize the percentage of music items that can be found via similarity-based browsing.

6. CONCLUSIONS

In this paper we have shown that designing music recommender systems is not as straight forward as it seems. Especially reachability is an important property if a music recommendation system should also allow users to explore a music archive via browsing. A bad system design might have the consequence that a portion of all songs in the database cannot be discovered as they are not accessible at all. To overcome these limitations we took a first attempt to modify the graph representation of a recommender system in such a way that browsing the resulting recommendation network is more convenient. We believe that improving the accessibility of songs in a music archives can significantly increase the usability of music services and might even help to alleviate the long tail phenomenon by ensuring the accessibility of *'niche'* products.

7. ACKNOWLEDGEMENTS

This research was supported by the Austrian Research Fund (FWF) under grant L511-N15, and by the Austrian Research Promotion Agency (FFG) - project number 815474.

8. REFERENCES

[1] O. Celma and P. Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. In *2nd Workshop on Large-Scale Recommender Systems (ACM KDD)*, Las Vegas, USA, 2008.

[2] O. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *2008 ACM Conference on Recommender Systems*, Lausanne, Switzerland, 2008.

[3] C. Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, July 2006.

[4] E. Brynjolfsson, Y. J. Hu, and M.l D. Smith. From niches to riches: Anatomy of the long tail. *Sloan Management Review*, 47(4), 2006.

[5] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 1951.

[6] M. Levy and M. Sandler. Lightweight measures for timbral similarity of musical audio. In *Proc. of the 1st ACM Workshop on Audio and Music Computing Multimedia*, Santa Barbara, USA, 2006.

[7] M. Mandel and D. Ellis. Song-level features and svms for music classification. In *Proc. of the 6th Int. Conf. on Music Information Retrieval*, September 2005.

[8] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. In *Proc. Int. Sym. on Music Information Retrieval*, 2008.

[9] J.-J. Aucouturier and F. Pachet. A scale-free distribution of false positives for a large class of audio similarity measures. *Pattern Recogn.*, 41(1):272–284, 2008.

[10] A.-L. Barabsi R. Albert. Statistical mechanics of complex networks. *Review of Modern Physics*, 2002.

FULL-AUTOMATIC DJ MIXING SYSTEM WITH OPTIMAL TEMPO ADJUSTMENT BASED ON MEASUREMENT FUNCTION OF USER DISCOMFORT

Hiromi Ishizaki
KDDI R&D Laboratories Inc.
ishizaki@kddilabs.jp

Keiichiro Hoashi
KDDI R&D Laboratories Inc.
hoashi@kddilabs.jp

Yasuhiro Takishima
KDDI R&D Laboratories Inc.
takisima@kddilabs.jp

ABSTRACT

This paper proposes an automatic DJ mixing method that can automate the processes of real world DJs and describes a prototype for a fully automatic DJ mix-like playing system. Our goal is to achieve a fully automatic DJ mixing system that can preserve overall user comfort level during DJ mixing.

In this paper, we assume that the difference between the original and adjusted songs is the main cause of user discomfort in the mixed song. In order to preserve user comfort, we define the measurement function of user discomfort based on the results of a subjective experiment. Furthermore, this paper proposes a unique tempo adjustment technique called “optimal tempo adjustment”, which is robust for any combination of tempi of songs to be mixed. In the subjective experiment, the proposed method obtained higher averages of user ratings on three evaluation items compared to the conventional method. These results indicate that our system is able to preserve user comfort.

1. INTRODUCTION

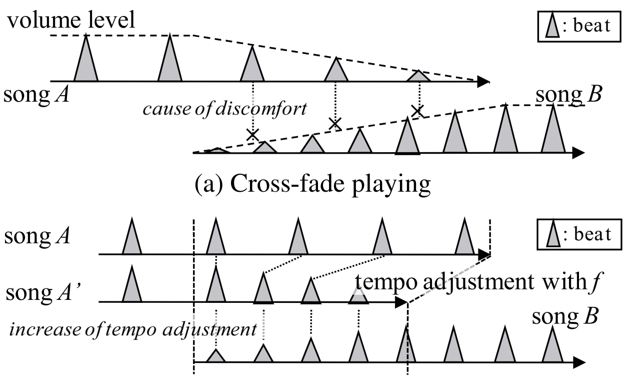
Due to the development of various audio compression methods, many online music distribution services have provided the opportunity for users to listen to songs from huge music collections. Furthermore, the increasing popularity of portable music players has enabled users to carry around thousands of songs. However, the variety of methods for the common user to enjoy listening to the songs in their collection is basically limited to “shuffle” play, which simply plays songs in the collection (and/or playlists) in random order. In order to extract a set of songs that match user preferences from large-scaled music collections , there are many useful techniques such as [1–3]. These techniques can provide users a set of songs as playlists, from which users select and play songs. In order to provide users new experience, it is important to play the songs in an entertaining way. For instance, Basu proposed a method which can blend two songs smoothly to create different aspects of the songs [4].

In the real world, DJs (disk jockey), *i.e.*, people who select and play music in clubs and discos, are able to maintain the excitement of the audience by continuously playing songs with the utilization of various DJ techniques: selections of songs, beat adjustment, *etc.*. One fundamental DJ technique is to gradually switch from one song to the other, while adjusting the beats of the songs. This technique enables the DJ to switch songs smoothly without disturbing the listener. A similar method should be effective in providing an entertaining music experience for common music listeners. However, such music playing requires skilled techniques and/or specialized equipment, which are both difficult for casual users to utilize.

In this research, we propose an automatic DJ mixing method that can automate real world DJ processes and describe a prototype for a fully automatic system. The objective of this research is to develop an automatic music playing system that can play a variety of different songs consecutively in an entertaining way without causing the users any discomfort. Specifically, we define the measurement function of user discomfort based on the results of a subjective experiment. Furthermore, we propose an optimal tempo adjustment technique that is robust for any combinations of the tempi of songs to be mixed.

2. CONVENTIONAL PLAYING METHOD

As mentioned in the previous section, DJs effectively utilize the cross-fade playing (*CFP*) technique to maintain the entertain level of the music they play. Naive *CFP*, *i.e.*, cross-fading two songs without any tempo/beat adjustment, is a simple and effective approach in avoiding silence between songs, and can be easily implemented in any music playing application. This method is effective in avoiding silence between songs, which may be distracting to listeners who prefer that the music play continuously. However, especially in situations where the tempi of the two songs to be cross-faded are significantly different (Figure 1-(a)), naive *CFP* may result in a negative listening experience, since the beats of the two songs occur asynchronously. Therefore, it is necessary for DJs to conduct *CFP* while adjusting the tempo and beat of one song to the other. The adjustment of tempo can be done by simple signal expansion (in cases where the song is to be played slower than the original) or contraction [5].



(b) naive DJ mixing

Figure 1. Conceptual illustrations of cross-fade playing and naive DJ mixing.

3. PROBLEMS

However, there are two problems in realizing such DJ techniques automatically.

One problem is the degradation in the acoustic quality of music, which may occur in the tempo adjustment process, especially in conditions where the tempi of the two target songs are significantly different (Figure 1-(b)). Such quality degradation may cause discomfort for listeners. Furthermore, the double or half tempo error is common for any existing automatic tempo extraction algorithm, as mentioned in [9]. Although a highly accurate tempo and beat extraction method is obviously essential for the implementation of a fully automatic DJ mix playing system, it is unrealistic to expect any system to achieve 100% accurate beat extraction. If the fully automatic DJ mix playing system adjusts the tempo based on tempo extraction results with double/half errors, the resulting factors of tempo adjustment will be two times the actual requirement. It is obvious that such excessive tempo adjustment is a cause of acoustic quality degradation, and ultimately, discomfort for music listeners. Furthermore, in the cases of adjustment of the song/songs that result in double/half tempo errors, strong beats and weak beats are adjusted to each other, which causes user discomfort.

The other problem is that there is no previous work on the effective measure of tempo adjustment to preserve the comfort level of users. It is not clear that users feel discomfort with regard to the degree of tempo adjustment or the manner in which the tempo was adjusted for the songs to be mixed. Actually, it is essential to define some kind of measure in order to achieve the fully automatic DJ mixing system. Additionally, it is important to investigate the threshold and the applicable range of tempo adjustment for songs to be mixed in order to achieve a comfortable DJ mixing system.

4. DEFINITION OF MEASUREMENT FUNCTION

In this section, we conducted a subjective experiment to define the measurement function of user discomfort. The objective of this experiment is to define the measurement function of user discomfort to determine the level of user discomfort given the tempo adjustment ratio.

In this experiment, we assume that the difference between the original and adjusted songs is the main cause of user discomfort. We investigate the correlation between user discomfort and tempo adjustment factors with actual tempo adjusted songs using time-scaling algorithms. Details of this experiment are presented as follows.

4.1 Experimental method

The methodology of this experiment, namely, details on the method of generating the sample audio and the subjective measure, are explained. In this experiment, we generate the actual songs for which the tempo will change. Subjects listen to these songs and input the time when they feel discomfort.

The experimental data set consists of 18 popular songs selected from the RWC music database [11]. For each of the selected songs, tempo changes are applied to the song excerpts. The adjusted tempo is obtained by multiplication of the original tempo of song and the factor of tempo adjustment f, $f > 1$ means the speedup factor and $f < 1$ means the slowdown factor. The speedup and slowdown factors for tempo changes are set from 1.00 to 2.00 and 1.00 to 0.30, respectively. For each experiment, the song is played in its original tempo for the first 15 seconds. After this initial period, the tempo of the song is repetitively increased (in the case of speedup) or decreased (in the case of slowdown) by a scale of 0.05, for every three seconds, until the tempo change factor reaches its maximal/minimal value. This range is decided empirically enough to investigate the correlation.

In the tempo adjustment, we have changed the time scale of the songs, while maintaining the original pitch. As tools of tempo adjustment, we use the two time-scaling algorithms: the audio processing library *SoundTouch Library* [1] and the *SOLA* [10] time-scaling algorithm. *SoundTouch* is a high quality means to change tempo, *SOLA* is a low quality means. A total of 72 excerpts are generated for this experiment (44.1 kHz, 16-bit, WAV).

In this experiment, the 96 subjects are divided into two groups. Each group listens to half of the excerpts (36 excerpts per group). In the listening task, the subject is to submit the time when they feel discomfort to the tempo change of the song. The submission results are accumulated to analyze the effects of tempo change factors.

4.2 Results

Table 1 shows the averages of tempo adjustment factors that subjects feel discomfort to the song associated with each time-scaling algorithm. In this table, there are differences between speedup and slowdown factors where the subjects feel discomfort. These results show that the subjects are more sensitive to effect of slowdown as opposed to speedup. Furthermore, the averages of tempo adjustment factors for *SoundTouch* and *SOLA* are approximately

[1] SoundTouch Library: http://www.surina.net/soundtouch/

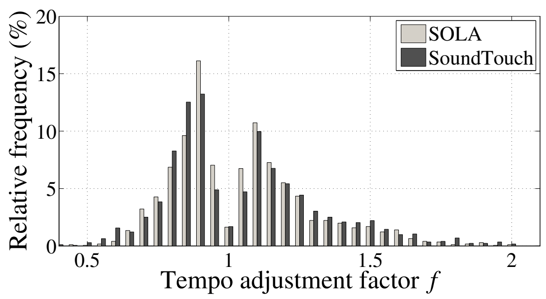


Figure 2. Histogram of user discomfort to factors of tempo adjustment.

Table 1. Averages of tempo adjustment factors

method	speedup	slowdown
SoundTouch	1.227	0.852
SOLA	1.226	0.852

equal to each other. These results indicate that user discomfort depends on tempo adjustment factors rather than the method.

Figure 2 shows the histogram of user discomfort and factors of tempo adjustment with each time-scaling algorithm. In this figure, the factors at the peaks of each algorithm are 1.10 (speedup) and 0.90 (slowdown). The percentages of subjects that feel discomfort inside these factors of each algorithm are 15.42% (*SOLA*) and 11.31% (*SoundTouch*). In the area near the original tempo, there are differences between the algorithms. *SoundTouch* is better able to preserve the comfort level of subjects under the condition in which the factor satisfies $0.90 < f < 1.10$ than *SOLA*.

4.3 Definition from the result

In order to define the measurement function based on the results in the previous section, we assume that the difference between the original and adjusted songs is the main cause of user discomfort. On the basis of this assumption and previous results, we define the level of discomfort (L_{dc}) expressed by the following equation:

$$L_{dc}(f) = \begin{cases} a(f-1) & f > 1 \\ 0 & f = 1 \\ b(1/f-1) & f < 1 \end{cases} \quad (1)$$

In Eq.(1), parameters a and b are to be weighted because the level of user discomfort is different between the adjustment from the speedup factor and from the slowdown factor as described in the previous section. Hence we extract the weighted parameters a and b as $a = 0.765$ and $b = 1.000$, these are extracted to make the score computed by speedup and slowdown factors equal when the factors are given as those written in Table 1. These weighted parameters are assumed to be effective in preserving the users' level of comfort in the song-to-song (*StS*) transition of DJ mixing. For example, Eq.(1) is able to decide which factor is appropriate (speedup or slowdown) in the DJ mixing. Additionally, we extract the stricter and average applicable ranges from the factors at the peaks (mentioned in Section 4.2) and the averages shown in Table 1. Specifically, we extract the $0.90 < f < 1.10$ as the stricter applicable range and $0.852 < f < 1.227$ as the average applicable range.

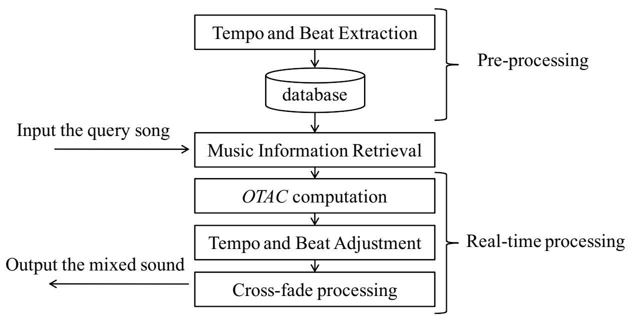


Figure 3. An overview of prototype of fully automatic DJ mixing system.

5. SYSTEM

In this section, we describe the prototype of the fully automatic DJ mixing system, which can solve the problems of tempo/beat adjustment, described in Section 3. By applying the score of the measurement function, which is computed based on the tempi of the target songs, our system is designed to be able to preserve the overall level of user comfort during the transition between songs.

Fig. 3 shows the overview of the prototype for the fully automatic DJ mixing system. This system mainly consists of five processes: tempo and beat extraction, music information retrieval (*MIR*), optimal tempo adjustment coefficients computation, tempo and beat adjustment, and cross-fade playing. In this system, we propose a unique tempo and beat adjustment method, which is able to deal with double or half tempo errors in the tempo and beat extraction technique: optimal tempo adjustment is able to compute the optimal factors of tempo adjustment to minimize the amount of tempo adjustment by dealing with tempo octave relationships. Details of the main processes of the system are described as follows.

5.1 Tempo and beat extraction

In this section, we describe the method of automating the DJ processes: tempo and beat extraction. As concerns the tempo and beat extraction process, there are many research efforts in tempo and beat extraction techniques, such as [6–8]. Although these techniques have the common problem of double/half error, there are practical mean to extract the tempo and beat automatically. Such methods can be useful to automate the tempo and beat extraction in DJ mixing processes. In our proposal, we apply *BeatRoot*[2] as the method of extracting the beat in the pre-process to the database.

[2] http://www.elec.qmul.ac.uk/people/simond/beatroot/

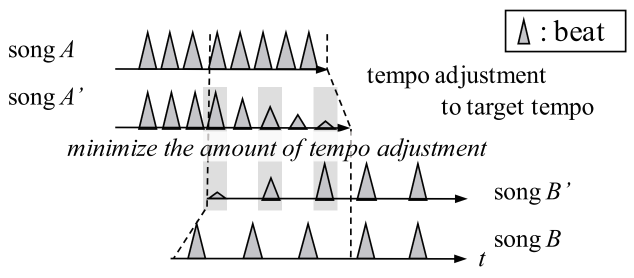


Figure 4. Conceptual image of dual tempo adjustment

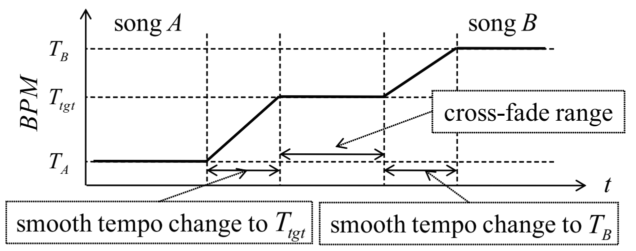

song A
song B
T_B
T_{tgt}
T_A
BPM
cross-fade range
t
smooth tempo change to T_{tgt}
smooth tempo change to T_B

Figure 5. Shifts of tempi of target songs in *StS* transition.

5.2 Music information retrieval

In this section, we describe the method of automating the DJ processes of selecting the songs to be mixed. As mentioned in Section 1, there are many research efforts in music information retrieval/recommendation. Although these are not specialized to DJ mix playing, these have achieved highly accurate retrieval/recommendations. Hence these are practical ways of substituting and selecting the song manually. In this system, we apply the content-based MIR technique [2], which can retrieve songs from the database by means of content-based similarity to the users' query.

5.3 Proposed DJ mixing

5.3.1 Optimal tempo adjustment coefficient computation

In order to automatically generate a smooth *StS* transition, we propose a unique tempo adjustment technique. Our proposal computes the optimal tempo adjustment coefficients, hereafter described as *OTAC*, which expresses the factors of tempo adjustment for the songs to be consecutively played, thus is capable of automatically generating smooth *StS* transitions for any given combination of songs. Namely, two *OTAC*s are computed and optimized for each song in the combination. As previously mentioned, the naive tempo adjustment approach may result in user discomfort, especially under conditions where the tempo of song *A* (T_A) and song *B* (T_B) are significantly different, which causes the tempo adjustment factor to be extremely high.

In order to solve this problem, the proposed method considers the individual position of beats in the two songs to compute the *OTAC*s, which will hereafter be denoted as f_{opt}. Figure 4 shows the conceptual image of proposed DJ mixing. We focus on the position of beats in the two songs, and it is clear that the beats of the two songs can match the smaller factors of tempo adjustment compared to naive DJ mixing. The proposed method computes *OTAC*s by utilizing the double/half characteristics to reduce the score for user discomfort.

The following describes the computational procedure for *OTAC*s, which expresses the factors of optimal tempo adjustment of the two target songs. In this procedure, we reduce the amount of tempo adjustment and user discomfort in a *StS* transition by dual tempo adjustment, for example, song *A* with a 5% speedup factor and song *B* with a 5% slowdown factor, instead of song *A* with a 10% speedup factor and song *B* untouched. In the following explanation, song *A* is defined as the target song to compute *OTAC*s.

First, a candidate set of adjusted T_A is computed using the following Equation:

$$T'_A = 2^C \times T_A \qquad (2)$$

where $C = \{-2, -1, 0, 1, 2\}$. From the set of T'_A, we select the result which is closest to T_B. This is equivalent to determining $C_{opt} = \text{argmin}(|T'_A - T_B|)$.

Next, parameter b_{opt} is computed with the following Equation:

$$b_{opt} = 2^{C_{opt}} \times T_A \qquad (3)$$

In Eq.(3), multiple values of b_{opt} can be computed in certain combinations of T_A and T_B. In such cases, the value b_{opt}, which results in a smaller $|C_{opt}|$, is selected. For example, given tempo combination as $(T_A, T_B) = (50, 75)$, possible solutions of Eq.(3) are $b_{opt} = 50, 100$. In this case, $b_{opt} = 50$ is selected as the final parameter.

The target tempo T_{tgt}, which the adjustment of the tempi of songs *A* and *B* will match, is computed with the following equation:

$$T_{tgt} = \frac{(a-b)T_{low} + \sqrt{(a-b)^2 T_{low}^2 + 4abT_{high}T_{low}}}{2a} \qquad (4)$$

where T_{high} denotes the tempo of the song with a higher tempo, and T_{low} denotes the lower in b_{opt} and T_B. T_{tgt} is designed to divided the score based on Eq.(1) equally between the two songs, *i.e.*, T_{tgt} is computed in order to satisfy that the L_{dc} of speedup and slowdown is equal. Figure 5 shows the shifts in the tempi of target songs in the transition, which is the case where the tempo of song *A* is lower than song *B*. These shifts are optimized for reducing the score of user discomfort based on Eq.(1).

Finally, the *OTAC*s f_{optA}, f_{optB} are computed based on b_{opt}.

$$f_{optA} = \frac{T_{tgt}}{b_{opt}}, \quad f_{optB} = \frac{T_{tgt}}{T_B} \qquad (5)$$

The proposed method is capable of computing the factors of optimal tempo adjustment for any combination of two songs. For instance, where the tempi of songs *A* and *B* are 60 and 120 BPM, the result of the computed *OTAC*s is $f_{optA} = f_{optB} = 1$, which is equal to the ideal rate for preserving the overall acoustic quality of the DJ mix result. It is also notable that the proposed method is capable of applying the DJ mix regardless of the existence of double/half tempo estimation errors, since the effect of such errors is disregarded during the *OTAC* computational procedure.

5.3.2 Beat adjustment and cross-fade playing

Next, we explain the procedure to generate the *StS* transition of the mixed sound. This procedure is necessary to reduce the discomfortness of the mixed sound, which assume to occur when the strong beats of a song are adjusted to the weak beat of the other song during the cross-fade range. In this procedure, we utilize the power of the beats in the cross-fade sections, to avoid the mismatching of strong and weak beats in the two songs to be mixed.

In order to generate the *StS* transition that matches the strong beats precisely, our method computes the score for the cross-correlation of the beats of target songs within the range of the cross-fade. When the powers of beats within the range of the cross-fade of songs *A*, *B* are described as Pow_A and Pow_B. The following describes the power of n-th beat as $Pow_A(n)$ and $Pow_B(n)$. The score between the songs *A*, *B* is described as Equation (6):

$$score(\tau) = \frac{\sum_{k=1}^{\tau}(Pow_A(N_A - k + 1)Pow_B(k))}{\tau} \quad (6)$$

where τ denotes the number of beats within the range of the cross-fade and N_A denotes the number of beats of song *A* as the former song in the mixed sound. Specifically, the beats of song *A* are matched to the beats of song *B* when $\tau_{max} = \mathrm{argmax}_\tau(score(\tau))$ is satisfied. Pows are computed by the power located near the beat ($\pm 50ms$). The powers of the spectrogram are computed by the FFT of the audio signal low-pass filtered (20th order FIR, cutoff freq. 1500Hz). Finally, cross-fade is applied to the overlapped range based on the highest score computed by τ_{max}.

6. EXPERIMENT

In this section, we will describe the experiment to subjectively evaluate our system and the proposed DJ mixing method. The objective of this experiment is to evaluate the effectiveness of the proposal.

In order to conduct this evaluation, two sets of DJ mixed sounds are generated; one by naive DJ mixing, and the other by the proposed method. The experiment is evaluated in a subjective manner. Namely, subjects of the experiment are to listen to the mixed sounds and provide preference ratings for each sample. Details of the experiment are described as follows.

6.1 Data

Experimental data consist of 1434 songs, which are collected from *Jamendo* [3] , a web site which distributes music licensed by Creative Commons. The source audio used for the experiments is extracted from the songs in the data collection. The length of each source is 30 seconds including the chorus. Note that, for all source audio, meta-information, such as the position of each beat, and tempo (BPM) are applied by *BeatRoot*.

[3] http://www.jamendo.com/

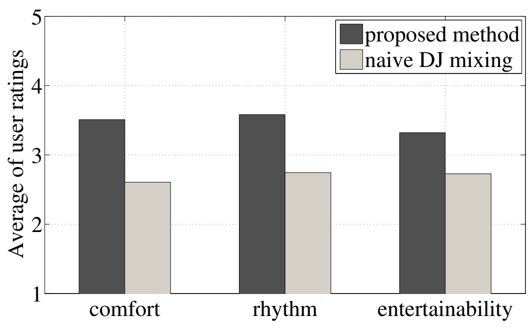


Figure 6. Average of user ratings in proposal and naive DJ mixing.

6.2 Experimental method

6.2.1 DJ mixed sound generation

The mixed sound files are generated by applying one of the previously described methods using five selected source audio extracted by MIR system [2] as the target songs. In total, six mixed sounds are generated by naive DJ mixing and the proposal, respectively. For the methods that utilize tempo adjustment, we have added interval periods to gradually change the tempi from/to the original to/from the target tempo, as shown in Fig.5. This interval period, which is fixed as 5 seconds for all mixed songs, is inserted in order to avoid abrupt changes in tempo, which is obviously uncomfortable. The period in which *CFP* is conducted begins immediately after the 5 second interval. For tempo adjustment, we use the *SoundTouch Library*.

6.2.2 Subjects and evaluation measures

A total of 27 subjects participated in the experiment. Each subject listened to all of the generated DJ mixed sounds and were asked to provide subjective ratings in five ranks for all sounds. In total, 165 ratings were collected on naive DJ mixing and the proposed method, respectively. Evaluation measures consist of the following three items: "*comfort*": the level of listener comfort during *StS* transition (1: discomfort – 5: comfort), "*rhythm*": the smoothness of the rhythm through the sound (1: bad – 5: good), and "*entertainability*": the overall preference rating (1: bad – 5: good).

6.3 Results

Average of user ratings in proposed method and naive DJ mixing are shown in Figure 6. It is clear from this figure that the proposed method was given a higher rating for all evaluation items compared to the conventional method, proving the overall effectiveness of the proposed method. According to the result of paired t-test, there are statistically-significant differences ($p < 0.001$).

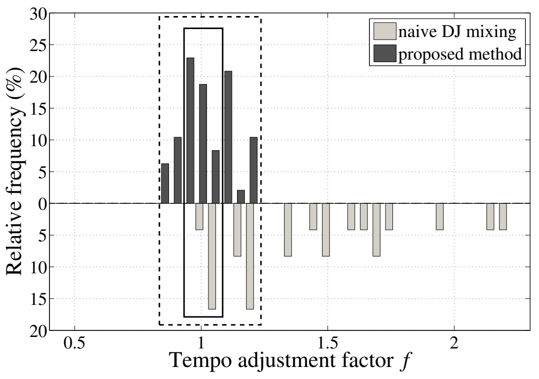


Figure 7. Histograms of the relative frequency of factors in *StS* transitions of proposal and naive DJ mixing.

Figure 7 shows the histograms of the relative frequency of factors in each of the *StS* transitions in each mixed sound. In this figure, stricter and average applicable ranges described in Section 4.2 are plotted as solid and dashed lines.

It is clear from this figure that the proposed method can keep factors near the original tempo compared to naive DJ mixing in a transition. The proposed method is able to deal with the difference in tempi between the former and latter songs. Furthermore, it is notable that the proposed method can almost satisfy the stricter applicable range and perfectly satisfy the average applicable range. Specifically, the percentage of factors inside the stricter range of the proposed method is 50.00% and inside the average range is 100.00%.

For further analysis, we investigated the averages of user ratings for each mixed sound. There were some cases that although L_{dc} of the proposed method were lower than naive DJ mixing, the score of user ratings was lower than naive DJ mixing. These cases tended to be adjusted strong beats and weak beats. In this case, user ratings of the proposed method about the evaluation item *RH* is lower than that of naive DJ mixing, which is able to adjusted appropriately. Furthermore, the correlation between *CF* and *RH* has a strong positive-correlation to each other. These results indicate that appropriate beat adjustment is one of the important factors. Generation of a smooth *StS* transition in the aspect of *RH* is essential to achieving a high quality DJ mixing method.

7. CONCLUSIONS

In this paper, we proposed an automatic DJ mixing method with optimal tempo adjustment with a function to measure user discomfort, described a prototype for a fully automatic DJ mixing system. The measurement function is defined by a subjective experiment, and our proposed method is designed to optimize the score of the function. In order to generate a smooth song-to-song transition, this paper proposes an optimal tempo adjustment based on the computation of optimal tempo adjustment coefficient. Furthermore, the proposed DJ mixing method is designed to preserve user comfort. The proposed DJ mixing is capable of generating a smooth song-to-song transition for any given combination of songs that includes double or half tempo errors. The advantages of the proposed method were proved by comparing the subjective evaluations of the samples generated by the proposed and conventional methods.

However, it is also obvious that tempo is just one of many elements in music that affect user preferences. For example, some combinations of source songs were unacceptable to subjects in the experiments, regardless of the DJ mixing method implemented to generate the sample audio. Therefore, we plan to further pursue research to develop a way to effectively apply the measurement function and a fully automatic music playing method, including the extraction and utilization of features other than tempo and beat position.

8. REFERENCES

[1] S. Pauws and B. Eggen: "PATS: Realization and user evaluation of an automatic playlist generator," *Proc. ISMIR 2002*, pp. 222-230, 2002.

[2] K. Hoashi, *et al.* : "Personalization of User Profiles For Content-based Music Retrieval Based on Relevance Feedback," *Proc. ACM Multimedia 2003*, pp.110-119, 2003.

[3] K. Yoshii, *et al.* : " Improving Efficiency and Scalability of Model-based Music Recommender System Based on Incremental Training," *Proc. of ISMIR* , pp.89-94, Vienna, Sep. 2007.

[4] S. Basu: "Mixing with Mozart," *Proc. of ICMC 2004*

[5] A. Inoue, *et al.* : "Playback and Distribution Methods for Digital Audio Players" *IPSJ SIG Notes 2006(9)* pp.133-138 (*in japanese*)

[6] M. Alonso, *et al.* : " Tempo and beat estimation of musical signals," *Proc. ISMIR 2004*, pp.158-163, 2004.

[7] S. Dixon: "Automatic extraction of tempo and beat from expressive performances," *J.New Music Res.*, Vol.30, No.1, pp.39-58, 2001.

[8] E. Scheirer: "Tempo and beat analysis of acoustic musical signals," *J. Acoust. Soc. Amer.*, Vol.103, No.1, pp.588-601, 1998.

[9] F. Gouyon, *et al.* : "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. Audio, Speech, and Lang. Process.*, IEEE Transactions on. Sept., pp.1832-1844, 2006.

[10] S. Roucos and A. M. Wilgus: "High quality time-scale modification for speech," *IEEE ICASSP*, pp.493-496, 1985.

[11] M. Goto, *et al.* : "RWC Music Database: Popular, Classical, and Jazz Music Databases," *Proc. of ISMIR 2002*, pp.287-288, October 2002.

FINGERING WATERMARKING IN SYMBOLIC DIGITAL SCORES

David Gross-Amblard
Le2i-CNRS Lab.
Université de Bourgogne
France
first.last@u-bourgogne.fr

Philippe Rigaux
Lamsade-CNRS Lab.
Université de Dauphine
Paris IX, France
first.last@dauphine.fr

Lylia Abrouk   Nadine Cullot
Le2i-CNRS Lab.
Université de Bourgogne
France
first.last@u-bourgogne.fr

ABSTRACT

We propose a new watermarking method that hides the writer's identity into symbolic musical scores featuring fingering annotations. These annotations constitute a valuable part of the symbolic representation, yet they can be slightly modified without altering the quality of the musical information. The method applies a controlled distortion of the existing fingerings so that unauthorized copies can be identified. The proposed watermarking method is robust against attacks like random fingering alterations and score cropping, and its detection does not require the original fingering, but only the suspect one. The method is general and applies to various fingering contexts and instruments.

Keywords. Watermarking, fingering

1. INTRODUCTION

In this work we consider symbolic musical scores that contain *fingering annotations*. Such fingerings ease the score interpretation for the novice player, and can guide the professional player. Producing high quality fingerings is a complex and costly task for the score writer. Up to now, it mainly remains an hand-made task, although several automatic fingering methods have been proposed recently [1–3].

The score writer's investment is threaten by the development of musical scores in digital form. Any buyer of such scores can obtain a perfect copy of the files and resell illegal copies. Watermarking is a known tool to protect the intellectual property of digital content, and it can be envisioned for musical scores as well. This would enable the distribution and sharing of score files marked by the copyright of their owner(s), just like score sheets are nowadays, but with the numerous advantages associated with the digital format.

Several methods have been proposed to hide the owner's identity into score images, by changing pixels [4], staff thickness [5] or symbols shape [6, 7]. These approaches are well fitted for protecting score images, but are not relevant for data exchange in a symbolic format like MusicXML [8]. Given the high cost of producing a symbolic digital score, writers may demand a robust mechanism to embed their copyright mark in the music symbolic representation. This copyright mark must be preserved throughout the operations that can be applied to the digital representation (e.g., transposition). It should not depend on side aspects such as graphical output details (e.g., the thickness of staff lines) which can easily be replaced or even eliminated without harm, as they are not part of the symbolic representation. Finally, the watermark should not alter the music content. In order to satisfy these requirements, our approach consists in watermarking the existing scores annotations. In the present paper we apply this idea to fingering annotations. Up to our knowledge, this is the first work on watermarking the music semantics itself.

The key idea of the method, given a musical score and a hand-made high quality fingering, is to choose several short secret fragments of the score. Given a score fragment, we replace the existing fingering with another fingering, chosen secretly among several computer-made fingerings of comparable quality. All secret choices are made using a cryptographic pseudo-random number generator, seeded by a summary of the musical structure and with a secret key known only by the legitimate owner. The resulting fingering will be published with the musical score. Finally, given a suspect score, the correspondence of the suspect fingering with our secret choices on our secret fragments acts as the proof of ownership. Our method applies to any fingering scenario, as soon as a quality metric of fingerings is available along with an automatic fingering method for small fragments (such as in piano or guitar music for example).

It should be clear that we protect the combination of the score and its fingering, and not the score itself. We also suppose that the attacker cannot afford to alter the score significantly, as this would result in an unsellable score (nevertheless we moderate this assertion in Section 3).

Outline. The paper is organized as follows. In Section 2 we introduce our general model for fingering and watermarking. Section 3 presents our watermarking and detection algorithms. Section 4 discusses several issues on the robustness of the watermark against natural score manipulation or malevolent attacks. Experiments assessing our method are presented in Section 5. Section 6 briefly covers the related work and Section 7 concludes.

2. FINGERING AND WATERMARKING

2.1 Fingering

The method proposed in this paper applies to any fingering context, but for the sake of simplicity we will focus on right-hand piano fingering for melodic inputs. Given a score in symbolic notation, we abstract it as a sequence $s = (n_1, \dots, n_N)$ of N consecutive notes. A fingering $f(n_i)$ for a note n_i is an integer in $\{1, 2, \dots, 5\}$, where number 1 to 5 represents a right-hand finger, respecting the usual conventions. For example, $f(A) = 2$ means that note A will be played by the forefinger.

The watermarking method uses an estimate of the quality of a fingering, that is related to the player inner feelings. We suppose the existence of a cost function $cost(f, s)$ that provides the cost of fingering f for the score s: the higher the cost output by this function, the lower the quality of the provided fingering (such functions exist for several instruments like piano [1]). We will explicit such a function in the experiments of Section 5, but our method applies to any cost function. We also often use the cost of a fragment w of the score s, that we denote $cost(f, w, s)$.

The first staff of Figure 1 presents an original score fragment with fingering annotations built by the score writer. Fingering annotations appear above the score. Annotations below the score are presented here only for the purpose of explanation, but are not published by the score writer. They show the cumulative cost of playing the score with the corresponding fingering (for example, playing the whole score costs 50 according to the chosen cost function).

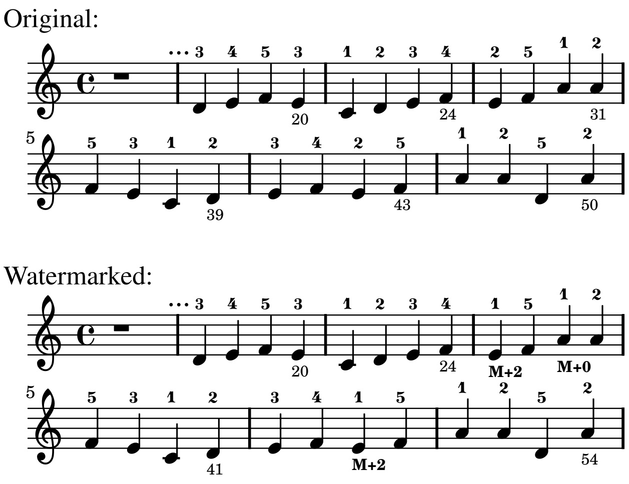


Figure 1. Different fingerings of the same score, with cumulative costs

2.2 Watermarking protocols

A watermarking protocol is a pair of algorithms $(\mathcal{W}, \mathcal{D})$, where $\mathcal{W}$ and $\mathcal{D}$ are respectively the marker and detector algorithms (see Figure 2). Given an original score s and a high quality fingering f, the score writer will watermark it by obtaining a specific fingering $f_M = \mathcal{W}(s, f, \mathcal{K})$, depending on a secret numerical key $\mathcal{K}$. The watermarked score (s, f_M) is sold to users. If a suspect copy (s^*, f^*) is discovered, the detector $\mathcal{D}$ applied on (s^*, f^*) using the secret key $\mathcal{K}$ should output *guilty* if f^* was obtained from f_M, and *not guilty* if f^* is a fingering obtained independently from f_M. A watermarking protocol is said to be *blind* if the original fingering is not needed at detection time, which may be useful as writer's fingerings may not be accessible easily or archived properly. The suspect fingering may have been also attacked/distorted before reselling, in order to erase the watermark. A watermarking protocol is said to be *robust* if it can still detect reasonably altered fingerings. Finally, respecting usual conventions, marker and detector algorithms are public, and their security relies only on the secret key.

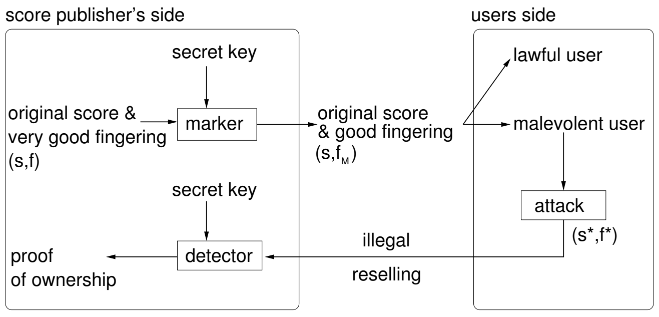


Figure 2. Protecting score and fingering by watermarking

3. FINGERING WATERMARKING

3.1 Watermarking algorithm

Algorithm 1 gives the pseudo-code of the marker. This algorithm scans a given score s by considering only a window of k consecutive notes (line 4 and 5). For each window, we first decide if it constitutes a good candidate for watermarking (line 6 and 7). This choice is secret and is based on the window content, the secret key $\mathcal{K}$ and a watermarking period γ known only by the score writer (this will be explained in the next section).

If a given window w is considered for watermarking, we focus on its first note n_i. We try to replace the original fingering $f(n_i)$ for this note by another one, $f'(n_i)$, also chosen secretly between the 5 possible fingerings for our piano example (line 9).

We compare the cost of this new fingering $cost(f', w, s)$ on window w with the cost of the original fingering $cost(f, w, s)$ on w (line 10). If the new cost exceeds the previous one by a limit ε, we cancel this modification (line 12). If the new fingering has a reasonable cost, we keep it for publication. Parameter ε, chosen by the score writer, controls the allowed amount of alteration that results from the watermarking process, and guarantees to produce fingerings with a good quality.

The second staff on Figure 1 demonstrates the process. For example, the 9th note (E) is considered for watermark-

Algorithm 1: Watermarking

```
Input: a score $s$ of $N$ notes $n_1, \ldots, n_N$, a high
       quality fingering $f$ for $s$, a secret key $\mathcal{K}$, a
       window size $k$, a quality threshold $\varepsilon$, a
       period $\gamma$.
Output: a watermarked fingering $f'$.
1  begin
2    // copy $f$ to $f'$
3    $f' := f$
4    for $i = 1$ to $N - k + 1$ do
5      $w = n_i.n_{i+1} \ldots n_{i+k-1}$ // reference window
6      seed PRNG $G$ with $signature(w).\mathcal{K}$
7      if ($G.nextInt() \mod \gamma = 0$) then
8        // try to watermark the first note
9        $f'(n_i) := G.nextInt() \mod 5$
10       if ($|cost(f', w, s) - cost(f, w, s)| > \varepsilon$) then
11         // revert changes
12         $f'(n_i) := f(n_i)$
13       end
14     end
15   end
16   return $f'$
17 end
```

ing. Its original fingering (finger 2) has been replaced by a new fingering (finger 1). This yields an extra cost of 2, which is considered reasonable for this example. The overall watermarking process yields a total extra cost of 4 on the score fingering.

3.2 Randomness

We now explain how random choices are made. Given a window w, we compute its musical signature based on its core music content (*signature()* function, line 6). The signature is independent from annotations and ornaments that are pointless for our algorithm. It is robust against naïve transposition attacks as it transposes the score into a common key (but of course, fingering costs are computed according to the original score). It is also invariant against score rewriting replacing a note or group of notes by an equivalent encoding (for example, replacing a half note by two tied quarters). In this paper, the signature is the concatenation of transposed note pitches, where consecutive equal pitches are suppressed. For example, the signature of `ABAABC` is `ABABC` (seen as a number), and time is not taken into account.

We concatenate this signature with the secret key $\mathcal{K}$ (a number), known only by the score writer. Then (line 6), we seed a cryptographic pseudo-random number generator (PRNG) with this number (as in [9]). This generator is used for all subsequent choices and has interesting properties. First, if it is seeded with the same value, the produced numbers are deterministic. Hence, if we know the secret key, we will be able to reproduce the pseudo-random choices made at watermarking time. Second, if the secret key is unknown, the generator outputs look completely random and can not be reproduced. Hence an attacker, unaware of the secret key, is fighting against randomness.

3.3 Detection algorithm

Algorithm 2: Detection

```
Input: a suspect score $s$ of $N$ notes $n_1, \ldots, n_N$ with
       its fingering $f^*$, a secret key $\mathcal{K}$, a window size
       $k$, a quality threshold $\varepsilon$, a period $\gamma$, a security
       parameter $\delta$.
Output: $guilty$ or $not\ guilty$.
1  begin
2    // copy $f^*$ to $f'$
3    $f' := f^*$
4    $total := 0$, $match := 0$
5    for $i := 1$ to $N - k + 1$ do
6      $w = n_i.n_{i+1} \ldots n_{i+k-1}$ // reference window
7      seed PRNG $G$ with $signature(w).\mathcal{K}$
8      if ($G.nextInt() \mod \gamma = 0$) then
9        // check this window
10       // compute awaited value
11       $f'(n_i) := G.nextInt() \mod 5$
12       if ($|cost(f', w, s) - cost(f^*, w, s)| \leq \varepsilon$) then
13         // probably watermarked position
14         $total$++
15         if ($f'(n_i) = f^*(n_i)$) then
16           $match$++
17         end
18       end
19       else
20         $f'(n_i) := f^*(n_i)$ // revert changes
21       end
22     end
23   end
24   if ($match/total > \frac{1}{5} + threshold(N, \delta)$) then
25     return $guilty$
26   else
27     return $not\ guilty$
28   end
29 end
```

The detection algorithm (see Algorithm 2 for the pseudocode) proceeds like the marker algorithm. Using the same window size, watermarking period and secret key used at watermarking time, we seed the generator with each window signature and the secret key (line 7). Hence, the same random choices made at watermarking time are reproduced. Thus we can locate exactly those windows selected at watermarking time (line 8). Then, *since the detector does not have the watermarked fingering for comparison* (blind detector), we have to assess that this position has really been used for watermarking. For that, we replace the fingering of the first note by the awaited one, using the random generator (line 11). We then compute the cost of this fingering. If it exceeds the error limit ε, we discard this window and restore the initial fingering (line 20). If error limit is respected, this position is probably a watermark (line 14).

We then compare the awaited fingering with the found one (line 15). For the whole score, we maintain the ratio of the number of matching fingerings with the number of windows considered for detection. If this ratio exceeds a given threshold (line 24), we consider the score as suspect (the threshold value is discussed below).

4. DISCUSSION

In this section we discuss several classical issues related to watermarking algorithms.

Impact on quality. Since the PRNG outputs random numbers with uniform distribution, the probability for a window w to be considered for watermarking is $1/\gamma$. The impact of watermarking this window can not be higher than ε. Hence, for a N notes score, the mean overall alteration is at most $\varepsilon\lfloor N-k\rfloor/\gamma$.

Window size. As the window size k increases, the amount of randomness injected into the random generator extends. If we consider reasonable scores whose notes span 2 octaves, there is up to 14^k potential fingerings for k consecutive notes. We chose $k=5$ in our experiments, leading to half-a-million distinct window signatures.

False positives probability and threshold function. A false-positive detection occurs when the detector considers a random score as guilty. Clearly, this probability must be negligible. Let δ be this acceptable probability, say $\delta = 10^{-10}$. Let us consider a random score. The probability of a given window to be selected by the detector is $1/\gamma$. For piano fingering, the probability of a fingering to correspond – by chance – to the watermarked one is 1/5 (as there is 5 different possible fingerings). Hence the average number of total matches on a random score is $\lfloor N-k\rfloor/5\gamma$. By the Hoeffding bound [10], the probability that the detector ratio $\frac{match}{total}$ on a random score deviates from the previous average is such that

$$P[|\frac{match}{total}-\frac{1}{5}| > threshold(N,\delta)] < e^{-2\frac{N}{\gamma}threshold(N,\delta)^2}.$$

Hence, choosing $threshold(N,\delta) = \sqrt{\frac{\gamma}{N}\ln\frac{1}{\delta}}$ guarantees a false positive rate smaller than δ. For example, on a score of 10,000 notes with a watermarking period $\gamma = 10$ and $\delta = 10^{-10}$, the recommended threshold is 0.22.

Available bandwidth. Robustness and significance are proportional to the amount of watermark bits that can be hidden. In popular guitar pieces (e.g., guitar scores and tablatures for beginners), a significant number of watermark positions are available. But music for expert players may contain only a few fingering annotations. If this number is not sufficient to reach the security limit, or if the musical corpus is made of small pieces only, a natural extension is to consider the watermarking of an entire piece collection (collected in a CD for example). The watermark is spread on the collection, and since the detection method uses only a finite-size sliding window, the order of pieces within the collection is pointless at detection time. The method is also robust enough to recover the watermark on a subset/superset of scores.

Attacks. An attacker suspecting the occurrence of a watermark may try to evade detection by several means. First, the attacker can add easy-to-correct errors in the fingering. To be successful, the attacker will have to add such errors all along the piece, in order to erase sufficient watermark positions. Hence the overall fingering is full of errors. Second, the attacker can leave the fingering unchanged, but add errors on the score itself, in order to break synchronization with the fingering. If errors are simply equivalent notes rewritings, the signature method will probably recover the correct ones. If the error is large, it will break one watermark position. Again, errors must span the whole score to be efficient, which is unreasonable (due to lack of space, we omit the mathematical proof of these statements. They are similar to the false-positive analysis).

Another approach for the attacker is to refinger the score. A complete rewriting represents a significant amount of work, so why would this attacker bother buying a fingered score in the first place ? On the contrary, a small refingering acts as a random attack, as the attacker has no idea where to perform this fingering.

Finally, the malevolent user can attack the score structure. Brute-force transposition is not sufficient, as we normalize the score in a specific key for detection. A first technique is to resell only subscores (excerpts). This can occur even for a normal buyer using the score. However, as long as a significant fraction of the piece is present, the watermark can be detected (this fraction is typically 30% in the database watermarking literature [9]). If less than 1/3 of the piece is stolen, the loss of property is harmless. If an attacker mixes a watermarked collection with a huge number of unwatermarked pieces, the argument is similar.

A last technique is to fold or unfold the score according to repetition symbols. This attack can be counterfeited by discarding repeated parts in the $signature()$ function, both for watermarking and detection.

5. EXPERIMENTS

5.1 Data, cost function, parameters

Our experiments are based on 50 Chopin piano pieces from the KernScores repository [11], for a total of around 10,000 notes. Original fingerings were found with a Dijkstra algorithm using a fingering cost function close to [1] and [2] (our method supposes hand-made high quality fingerings, but this approach is sufficient to measure the watermarking impact on quality). These models encompass the cost of playing a note with a given hand position (vertical cost $cost_v(f,n)$), and the cost of the transition between one hand position to the next one (horizontal cost $cost_h(f_i, n_i \rightarrow f_{i+1}, n_{i+1})$). These costs are constant values that agree with the human hand physical possibilities (the precise definition of these costs in not relevant for the present paper, we refer the reader to [2] for in-depth explanations.) The $cost(f, n)$ of a fingering f is the sum of its horizontal and vertical costs, i.e.,

$$cost(f, n) = \sum_{i=1}^{N} cost_v(f_i, n_i) + cost_h(f_i, n_i \to f_{i+1}, n_{i+1}).$$

We used window size $k = 5$, error tolerance $\varepsilon = 10$ and detection threshold 0.8 (vertical and horizontal costs for one note or transition span between 0 and +14).

5.2 Experiments

Figure 3 shows the impact of the watermarking method for various values of watermarking period γ. Clearly, a period smaller than 5 yields a huge distortion, and greater values tend toward a constant error with respect to the original fingering. Figure 4 and 5 study the impact of a random attack that tries to erase the watermark as follows: a note fingering is chosen with probability $1/\gamma_a$, and changed into a random fingering up to a cost impact of 10. Figure 4 shows the attack impact on the watermarked fingering quality for various values of γ_a. It appears that the attack impact is larger than the watermark impact on the fingering cost: choosing $\gamma_a < 5$ leads to fingerings with poor (unsellable) quality. Figure 5 shows the attack impact on the detector ratio. Choosing a detection threshold of 0.8 guarantees that all suspect fingerings are correctly detected, expect for those with attack γ_a smaller than 6. Hence, Figure 4 and 5 argue that any attack tricking the detector also destroys the fingering quality. Finally, Figure 6 shows that using a random secret key does not yield false positive detection (the correct key is presented at index 50).

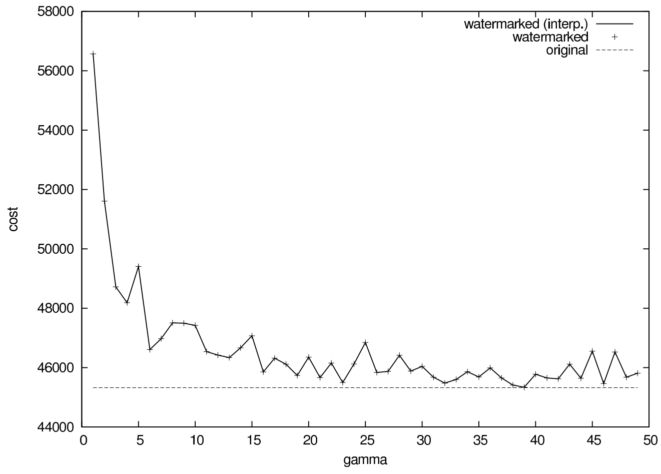


Figure 3. Impact of watermarking on fingering cost

6. RELATED WORK

Hiding information (for various purposes) in musical scores is an old story. A study of music score watermarking was performed during the WEDELMUSIC project. A good survey [12] recalls these approaches. In the visual domain,

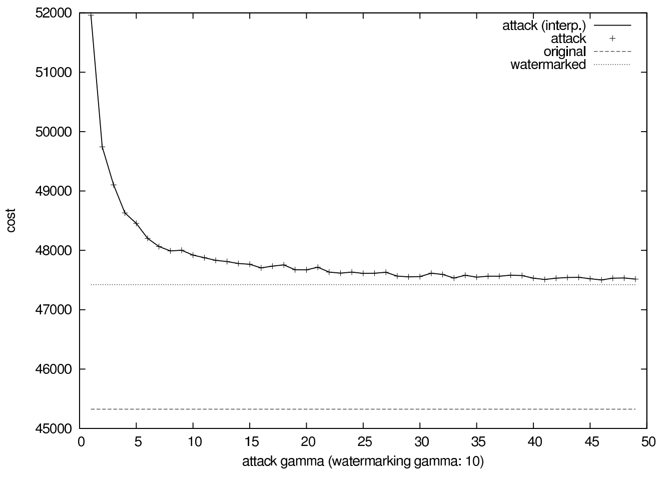


Figure 4. Impact of attack on fingering cost

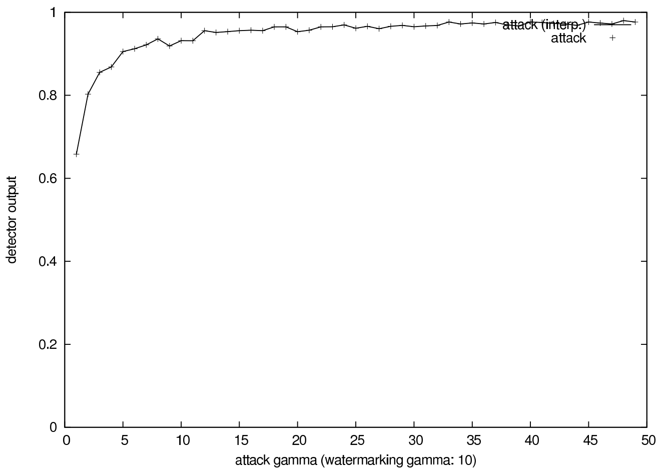


Figure 5. Impact of attack on detector's output

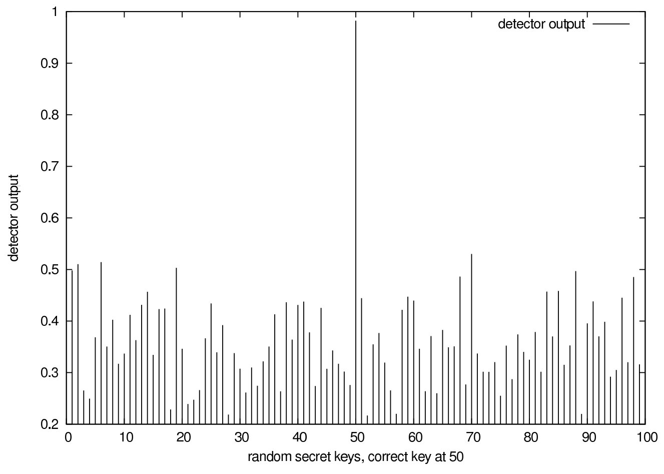


Figure 6. Detector output for random secret keys (correct one at 50)

classical but adapted image watermarking techniques can be applied on the image of a musical score. The watermark can be hidden by altering grayscales, or the binary representation of images, or the pixels themselves. In the musical notation (but still into the score image), one can alter the staff thickness, the vertical or horizontal distance between notes or groups of notes, notes orientation, thickness [5] or shape [6, 7]. Little is known on information hiding into the music semantics, where our work stands.

Our method shares some similarities with database watermarking methods: watermarking of relational databases of numerical values [9], numerical data streams [13] and XML streams [14]. All these methods use the same PRNG technique, and [13, 14] also use a finite window to scan a numerical or textual stream. The main difference is that our method has to control a non-local cost on data and may require rollbacks.

7. CONCLUSION

On-line distribution of musical scores is a promising area. Among other advantages, it could offer instant access to music collections, a wide diffusion of rare musical pieces, and computer-based services to browse, recommend, search and analyze music. However, producing music scores is a costly process and the protection of score writers against illegal copies is a prerequisite for on-line collection to emerge. In the present paper, we propose a watermarking algorithm based on the idea that the owner signature should be based on the musical content (which can hardly be modified) and hidden in a valuable annotation of this content – namely, fingerings. We propose a simple algorithm and show that it results in an effective protection. Although currently limited to fingerings, we believe that our approach can be extended to music annotations in general, for instance lyrics in vocal music. We are currently investigating this larger context.

8. ACKNOWLEDGEMENT

This work is partially supported by the French ANR Content and Interaction *Neuma* Project [15].

9. REFERENCES

[1] Melanie Hart, Robert Bosch, and Elbert Tsai. Finding optimal piano fingerings. *The UMAP Journal*, 2(21):167–177, 2000.

[2] Alia Al Kasimi, Eric Nichols, and Christopher Raphael. A simple algorithm for automatic generation of polyphonic piano fingerings. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 355–356, 2007.

[3] Yuichiro Yonebayashi, Hirokazu Kameoka, and Shigeki Sagayama. Automatic decision of piano fingering based on a hidden Markov model. In Manuela M. Veloso, editor, *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2915–2921, 2007.

[4] Wolfgang Funk and Martin Schmucker. High capacity information hiding in music scores. In Paolo Nesi, editor, *First International Conference on WEB Delivering of Music, proceedings of Wedelmusic 2001*, number 5020 in IEEE Computer Society Technical Committee on Computer Generated Music, pages 12–19, Los Alamitos, California, USA, 2001. IEEE Computer Society.

[5] Martin Schmucker. Capacity improvement of a blind symbolic music score watermarking technique. In Wah Ping Wong, editor, *Security and Watermarking of Multimedia Contents IV.*, number 4675 in SPIE Proceedings, pages 206–213, Washington, 2002.

[6] Martin Schmucker and Hongning Yan. Music score watermarking by clef modifications. In Edward J. Delp, editor, *Security and Watermarking of Multimedia Contents V.*, number 5020 in SPIE Proceedings, pages 403–412, Bellingham, 2003.

[7] Martin Schmucker, Christoph Busch, and Anoop Pant. Digital watermarking for the protection of music scores. In Wah Ping Wong, editor, *Security and Watermarking of Multimedia Contents III*, number 4314 in SPIE Proceedings, pages 85–95, Washington, 2001.

[8] Recordare. MusicXML document type definition. `http://www.recordare.com/xml.html`.

[9] Rakesh Agrawal, Peter J. Haas, and Jerry Kiernan. Watermarking relational data: framework, algorithms and analysis. *VLDB J.*, 12(2):157–169, 2003.

[10] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[11] Craig Stuart Sapp. Online database of scores in the Humdrum file format. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 664–665, 2005.

[12] M. Monsignori, Paolo Nesi, and Marius B. Spinu. Watermarking music sheets. In *PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, pages 646–653, London, UK, 2001. Springer-Verlag.

[13] Radu Sion, Mikhail J. Atallah, and Sunil Prabhakar. Rights protection for discrete numeric streams. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 18(5):699–714, 2006.

[14] Julien Lafaye and David Gross-Amblard. XML streams watermarking. In *IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC)*, 2006.

[15] Neuma project: network-enabled and user-friendly music analysis tools, 2008. `http://neuma.irpmf-cnrs.fr`.

IMPROVING MUSICAL CONCEPT DETECTION BY ORDINAL REGRESSION AND CONTEXT FUSION

Yi-Hsuan Yang, Yu-Ching Lin, Ann Lee, Homer Chen
National Taiwan University
affige@gmail.com, vagante@gmail.com, an918tw@yahoo.com.tw, homer@cc.ee.ntu.edu.tw

ABSTRACT

To facilitate information retrieval of large-scale music databases, the detection of musical concepts, or auto-tagging, has been an active research topic. This paper concerns the use of concept correlations to improve musical concept detection. We propose to formulate concept detection as an ordinal regression problem to explicitly take advantage of the ordinal relationship between concepts and avoid the data imbalance problem of conventional multi-label classification methods. To further improve the detection accuracy, we propose to leverage the co-occurrence patterns of concepts for context fusion and employ concept selection to remove irrelevant or noisy concepts. Evaluation on the cal500 dataset shows that we are able to improve the detection accuracy of 174 concepts from 0.2513 to 0.2924.

1. INTRODUCTION

Music plays an important role in human's history, even more so in the digital age. Never before has such a large collection of music been created and accessed daily by people. Bridging the semantic gap–the chasm between raw data (signals) and high-level semantics (meanings)–is essential for exploiting the growing music content. Toward this goal, recent research has focused on building detectors for detecting musical *concepts* such as genre, emotion, and instrumentation using a pre-defined lexicon and a sufficient number of annotated examples [1–8]. Once trained, these detectors can be used to semantically tag and index music content in a fully automatic fashion. A user can then query music by *semantic description* [2], such as "find me a song that is brit poppy and alternative, features male vocal, and has a nice distorted electric guitar solo."

Early attempts to musical concept detection formulated the problem as a multi-label binary classification problem and trained detector independently for each concept [1–3]. The training data is annotated by human subjects and the relationship between ground truth and audio features is learnt by machine. Subsequent efforts went one step forward and utilized the correlation between concepts (either

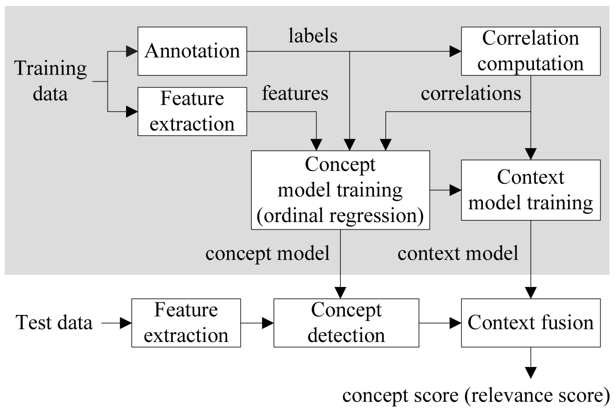


Figure 1. A schematic diagram of the proposed musical concept detection system.

positive or negative) to improve concept detection. Duan *et al.* proposed a collective annotation scheme that trains additional models for the pairs of concepts that have strong correlations [4]. Bertin-Mahieux *et al.* studied a second-stage learning and a correlation reweighting scheme to boost the result of concept detection [5]. Aucouturier *et al.* [6] used decision tree to refine the result of individual detectors. Chen *et al.* built anti-models to exploit the negative correlations of concepts [7]. Modeling concept correlation has been shown effective for improving musical concept detection.

It is, however, noted that most existing works focus on the refinement of the individual detectors by training additional models rather than focus on the direct incorporation of concept correlation in training the individual detectors. Evidently, there are different levels of correlation between concepts, by which we can divide the training data into more than two categories; some of the training pieces should be more relevant to a target concept than other pieces. Consider the following toy example. We are training a concept detector of "happy" based on three training pieces a, b and c, which are annotated with "happy," "tender" and "sad" respectively. Conventional approaches formulate the problem as a *flat* binary classification, using a as positive example and b, c as negative examples. However, since "happy" is semantically closer to "tender," there should be an ordinal scale among them, $\mathsf{a} \succ \mathsf{b} \succ \mathsf{c}$, where $\succ$ denotes a relevance relationship. Such ordinal information is neglected by treating b and c the same.

In this paper, we propose to formulate concept detection as an *ordinal regression* problem [9–11] and train a concept model to estimate the *relevance score* of a song with respect to a target concept. A higher relevance score represents a higher probability of the song being annotated with the concept. The advantage of this approach is two-fold. First, we can make better use of the training data (whose collection process is fairly time-consuming and labor-intensive) by explicitly leveraging the ordinal relationship between concepts. Second, conventional classification algorithms are hampered by the so-called data imbalance problem: the performance of a classifier degrades significantly when the number of training data is not uniformly distributed across classes. For example, when 95% of the training data is negative, a classifier can achieve a 95% accuracy by simply classifying everything as negative, which is highly undesirable. This problem is usually observed for infrequent concepts such as "genre-swing" and "instrument-organ." Ordinal regression is free of this problem because the objective function of learning is not minimizing classification errors and because the training pieces that are annotated with semantically close concepts can still be leveraged in learning.

The second contribution of the paper is the investigation of context fusion and concept selection to improve the detection result. The basic idea of context fusion is to leverage the co-occurrence patterns between target semantic and peripherally related concepts to improve the result of an initial model. It has been successfully applied to improve visual concept detection and image search [12–14]. Because of the assumption that the result is presented in an ordered form, context fusion can be combined with ordinal regression in an elegant way. We also study a concept selection method to remove irrelevant concepts to improve context fusion. The number of selected concepts is target concept-dependent. For a concept that lacks strongly correlated concepts, context fusion is not applied.

A schematic diagram of the overall system is shown in Fig. 1. In the train phase, the annotations, features, and concept correlations are utilized to train the individual concept detectors by ordinal regression. We then exploit the contextual patterns among concepts to train a context detector for each concept. The concepts utilized in context fusion are selected based on concept correlations. In the test phase, we extract the features of the test data and then apply concept detection and context fusion in cascade to generate the detection result.

The paper is organized as follows. In Section 2 we describe the corpus adopted in this work and the concept correlations therein. The correlations are then used in Section 3 for ordinal regression and in Section 4 for context fusion. We report the experimental results in Section 5. Section 6 concludes the paper.

2. CORPUS AND CONCEPT CORRELATION

We use the Computer Audition Lab 500-Song (cal500) data set [2] in this study for it is publicly available. [1] The collection is made of 502 recent Western songs by 502 different artists chosen to cover a large amount of acoustic variation. 66 paid students were recruited to annotate the songs with a fixed vocabulary of 135 musical concepts, with each song annotated by at least three respondents. A song is annotated with a concept if there is at least 80% agreement between the respondents. The concept lexicon spans six semantic categories: 29 instruments, 22 vocal characteristics, 36 genres, 18 emotions, 15 acoustic qualities, and 15 usage terms. The concepts of emotions and acoustic qualities are further broken down into bipolar ones (e.g., "emotion-happy" and "emotion-NOT happy"), resulting in a total of 174 concepts [2].

[1] Available at: http://cosmal.ucsd.edu/cal/projects/AnnRet

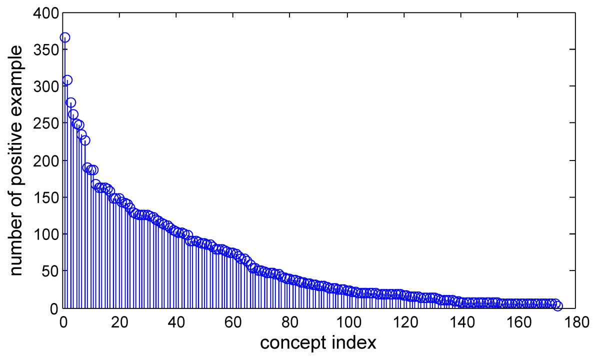


Figure 2. Concept frequency distribution of a subset (413 songs) of cal500 [2]. Note the concepts have been sorted by the number of positive examples.

We collect the audio files of 413 songs of cal500 and analyze the frequency of each concept. As Fig. 2 shows, rare concepts form a long tail in the concept frequency distribution. While frequent concepts (e.g., "song-recorded" and "instrument-male lead vocals") have more than 300 positive examples, 37 concepts have less than 10 positive examples. A preliminary evaluation also shows the detection accuracy of the infrequent concepts are particularly low. Because of this data imbalance problem, we also use a subset of concepts that have more than 50 positive examples in this study. The resulting lexicon, which consists of 69 concepts, is denoted as cal500-lite hereafter.

Given a concept lexicon $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ and N annotated examples $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, we can measure the pairwise correlation ρ_{mn} between two concepts c_m and c_n from the annotations A, which are represented by a $|\mathcal{C}| \times N$ binary matrix, with $A_{mi} = 1$ indicating that d_i is annotated with c_m. We compute ρ_{mn} by the Pearson's correlation coefficient of A_m and A_n,

$$\rho_{mn} = \frac{E((A_m - \mu_{A_m})(A_n - \mu_{A_n}))}{\sigma_{A_m}\sigma_{A_n}}. \tag{1}$$

$\rho_{mn} \in [-1, 1]$ and $\rho_{mn} > 0$ iff there is a positive correlation. Interestingly, we find the correlation values generally follow a Laplacian-like distribution: the number of concept pairs decreases exponentially along with the absolute values of correlation. See Fig. 3.

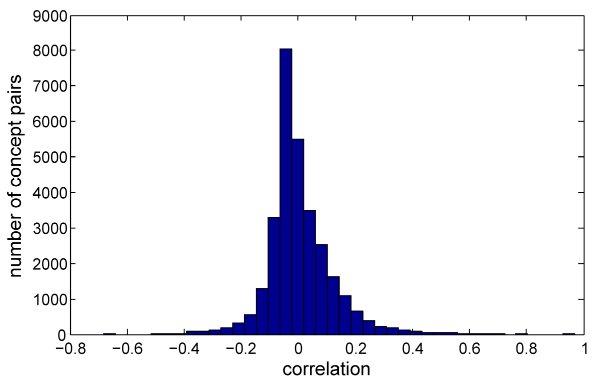


Figure 3. Distribution of the correlation values of cal500.

3. ORDINAL REGRESSION

3.1 Brief Review

Unlike classification, ordinal regression defines a number of classes that exhibits an *ordinal scale* among them. For example, the preference of a song can be categorized to "very dislike," "dislike," "neutral," "like," and "very like." The outcome space can be denoted as $\mathcal{Y} = \{r_1, \ldots, r_K\}$, with ordinal classes $r_K \succ_{\mathcal{Y}} r_{K-1} \succ_{\mathcal{Y}} \ldots \succ_{\mathcal{Y}} r_1$, where K is the number of classes. A closely related problem is ranking, which presents ordered results to a user in response to a query. A common example is the ranking of search results from the search engine (e.g., Google). Both ordinal regression and ranking assign each object a *relevance score*, by which the object is ranked. The difference is ordinal regression needs a further step that determines the class membership of each object with respect to the discrete ordinal classes.

In the seminal work of Herbrich *et al.*, the ordinal classes were modeled by intervals on the real line [9]. A discriminative function $f : \mathcal{X} \mapsto \mathbb{R}$ was trained to predict the relevance score $\hat{y_i} = f(\mathbf{x}_i) = (\mathbf{w} \cdot \mathbf{x}_i)$, where $\mathbf{x}_i$ is a feature vector of an object and $\mathbf{w}$ is a vector of weights. However, because the outcome space $\mathcal{Y}$ is discrete, Herbrich *et al.* determined the rank boundary $\theta(r_k)$ between classes r_k and r_{k+1} on the real line according to the following heuristics,

$$\theta(r_k) = \frac{1}{2}(f(\mathbf{x}_1) + f(\mathbf{x}_2)), \tag{2}$$

$$(\mathbf{x}_1, \mathbf{x}_2) = \operatorname*{arg\,min}_{(\mathbf{x}_i, \mathbf{x}_j) \in \Theta(k)} [f(\mathbf{x}_i) - f(\mathbf{x}_j)], \tag{3}$$

where $\Theta(k)$ is the set of object pairs $(\mathbf{x}_i, \mathbf{x}_j)$ with $y_i = r_k$, $y_j = r_{k+1}$, and $(\hat{y_i} - \hat{y_j})(y_i - y_j) \geq 0$. In other words, the optimal threshold $\theta(r_k)$ for rank r_k lies in the middle of the estimates of the closest objects of rank r_k and r_{k+1} that can be correctly ranked by $f(\cdot)$. After the estimation of the boundaries $\theta(r_k)$ a new object is assigned to an ordinal class according to the following equation,

$$g(\mathbf{x}_i) = r_k \Leftrightarrow f(\mathbf{x}_i) \in [\theta(r_{k-1}), \theta(r_k)]. \tag{4}$$

To learn $f(\cdot)$, Herbrich *et al.* viewed the problem as the classification of object pairs into two categories (correctly ranked and incorrectly ranked) and trained a support vector machine (SVM) to minimize the classification error $\sum_{i,j}^{N} (\hat{y_i} - \hat{y_j})(y_i - y_j)$. Though this algorithm, generally called rankSVM, offers advantages, it is time-consuming as the operation on every possible pair is $O(N^2)$.

Alternatively, we employ the listNet [11] algorithm in this work. It uses score lists directly as learning instances and minimizes the listwise loss between the ground truth ranking list and the estimated one. In this way, the optimization is performed directly on the list and the computation complexity is reduced to $O(N)$. More specifically, to define a listwise loss function, the top-one probability is employed to transform a list of relevance scores into a probability distribution. The top one probability $P(y_i)$ of the ith object, defined as follows, represents the probability of the object being ranked on the top,

$$P(y_i) = \frac{\Phi(y_i)}{\sum_{i=1}^{N} \Phi(y_i)} = \frac{\exp(y_i)}{\sum_{i=1}^{N} \exp(y_i)}, \tag{5}$$

where $\Phi(\cdot)$ is an increasing and strictly positive function such as the exponential function. Modeling the list of scores as a probabilistic distribution, a metric such as the cross entropy can be used to measure the distance (listwise loss) between the ground truth list and the estimated one,

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{N} P(y_i) \log(P(f(\mathbf{x}_i))), \tag{6}$$

where $\mathbf{y} = \{y_i\}_{i=1}^{N}$ and $\hat{\mathbf{y}} = \{f(\mathbf{x}_i)\}_{i=1}^{N}$. The algorithm then learns the weighting vector $\mathbf{w}$ by updating it at a learning rate η by gradient descent,

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \times \Delta\mathbf{w}, \tag{7}$$

$$\Delta\mathbf{w} = \frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{w}} = \sum_{i=1}^{N} (P(f(\mathbf{x}_i)) - P(y_i))\mathbf{x}_i. \tag{8}$$

It has been shown that listNet is more efficient and effective than rankSVM for a variety of ordinal regression and ranking problems, such as image/video search [13].

3.2 Concept Model Training by Ordinal Regression

Given the ground truth A_m of concept c_m and the feature representation of $\mathcal{D}$, typically a binary classifier $b_m(\cdot)$ is trained by treating $\mathcal{D}_m^{+} = \{d_i | A_{mi} = 1\}$ as positive examples and $\mathcal{D}_m^{-} = \{d_i | A_{mi} = 0\}$ as negative examples. However, such a dichotomy of the training data loses many valuable information embedded in A_m, as we have illustrated in Section 1. We can in fact divide the training data to multiple ($K \geq 2$) ordinal classes according to concept correlations and then employ listNet to train a concept model $f_m(\cdot)$ for each concept c_m,

$$\hat{y}_{mi} = f_m(\mathbf{x}_i) = (\mathbf{w} \cdot \mathbf{x}_i). \tag{9}$$

Two such implementations are employed in this work, $K = 2$ and $K = 4$. The first one simply dichotomizes $\mathcal{D}$ as the binary classification setting. That is, $\mathcal{D}_m^{r_2} = \mathcal{D}_m^{+}$ and $\mathcal{D}_m^{r_1} = \mathcal{D}_m^{-}$. We then set $y_{mi} = r_k$ if $d_i \in \mathcal{D}_m^{r_k}$.

In this way, the concept correlations are not explicitly utilized, but thanks to the ordinal regression algorithm (which minimizes a listwise loss instead of clarification error) the data imbalance problem is avoided. The second implementation uses $K = 4$ and partitions $\mathcal{D}$ to four classes according to the following rules, which are listed in descending order of precedence,

- $\mathcal{D}_m^{r_4} = \mathcal{D}_m^+$
- $\mathcal{D}_m^{r_1} = \{d_i | A_{ni} = 1, \rho_{mn} \leq l, d_i \in \mathcal{D}_m^-\}$
- $\mathcal{D}_m^{r_3} = \{d_i | A_{ni} = 1, \rho_{mn} \geq u, d_i \in \mathcal{D}_m^- \setminus \mathcal{D}_m^{r_1}\}$
- $\mathcal{D}_m^{r_2} = \mathcal{D}_m^- \setminus \bigcup\{\mathcal{D}_m^{r_1}, \mathcal{D}_m^{r_3}\}$

In other words, $\mathcal{D}_m^{r_1}$ consists of songs that are annotated with any of the concepts $\mathcal{C}_m^{r_1}$ that are strongly negatively correlated with c_m, and $\mathcal{D}_m^{r_3}$ consists of songs that are annotated with any of the concepts $\mathcal{C}_m^{r_3}$ that are strongly positively correlated with c_m. In this work, we set $l = \mu_\rho - \sigma_\rho$, $u = \mu_\rho + \sigma_\rho$, where $\mu_\rho \simeq 0.01$ and $\sigma_\rho \simeq 0.11$ are the mean and the standard deviation of all the correlation values of the concept corpus (see Fig. 3).

Table 2 shows some highly correlated concepts for four different target concepts. It can be found that most of the correlated concepts are intuitively correct.

4. CONTEXT FUSION

The nature of concept detection makes it possible to discover co-occurrence patterns through mining ground truth annotations and utilize the patterns to improve concept detection. For example, if a song has the concepts "song-high energy" and "song-heavy beat," it is very likely that it also has the concept "song-fast tempo." If the relevance score of "song-fast tempo" is somehow detected low (maybe the detector is less reliable), we can modify the result by increasing the value. We refer to such a model that learns the co-occurrence patterns as the context model.

Following the idea of discriminative model fusion (DMF) [12, 13], we train a context model for each concept based on the output of the concept models. For each song, the $|\mathcal{C}|$ concept models are employed to predict the relevance score of song d_i with respect to each concept; this results in a $|\mathcal{C}|$-dimensional *model vector* $\mathbf{v}_i = \{\hat{y}_{ni}\}_{n=1}^{|\mathcal{C}|} = \{f_1(\mathbf{x}_i), \cdots, f_{|\mathcal{C}|}(\mathbf{x}_i)\}$. We use the model vectors to train the context model $\tilde{f}_m(\cdot)$ for each concept c_m by minimizing the loss between $\{y_{mi}\}_{i=1}^N$ and $\{\tilde{f}_m(\mathbf{v}_i)\}_{i=1}^N$ using listNet. We then replace $\hat{y}_{mi}$ with $\tilde{f}_m(\mathbf{v}_i)$. That is,

$$\hat{y}_{mi} \leftarrow \tilde{f}_m(\mathbf{v}_i) = (\tilde{\mathbf{w}} \cdot \mathbf{v}_i) = \sum_{n=1}^{|\mathcal{C}|} \tilde{w}_n f_n(\mathbf{x}_i). \quad (10)$$

Therefore, $\tilde{f}_m(\mathbf{v}_i)$ can be regarded as the weighted combination of the relevance scores of d_i with respect to other concepts. Intuitively, the absolute value of $\tilde{w}_n$ would be large if c_n is highly correlated with c_m. A total of $|\mathcal{C}|$ context models are trained.

```
TRAINING PHASE
INPUT: training data $\mathcal{D}$, $A$, $\{\mathbf{x}_i\}_{i=1}^N$, parameters $K, \theta$
  compute correlations $\{\rho_{mn}\}_{m,n}^{|\mathcal{C}|}$ by Eq. 1.
  for $m = 1$ to $|\mathcal{C}|$
    partition $\mathcal{D}$ to $K$ classes by $A_m$ and $\{\rho_{mn}\}_{n=1}^{|\mathcal{C}|}$
    set $y_{mi} = r_k$ if $d_i \in \mathcal{D}_m^{r_k}$
    train $f_m(\cdot)$ by minimizing $L(\{y_{mi}\}, \{f_m(\mathbf{x}_i)\})$
  end
  for $m = 1$ to $|\mathcal{C}|$
    construct $\mathbf{v}_{mi} = \{f_n(\mathbf{x}_i)\}_{n:\text{abs}(\rho_{mn}) \geq \theta}$
    train $\tilde{f}_m(\cdot)$ by minimizing $L(\{y_{mi}\}, \{\tilde{f}_m(\mathbf{v}_i)\})$
  end
OUTPUT: concept and context models $\{f_m, \tilde{f}_m\}_{m=1}^{|\mathcal{C}|}$
TEST PHASE
INPUT: test data $\{\mathbf{x}_z\}$
  for $m = 1$ to $|\mathcal{C}|$
    predict $\hat{y}'_{mz} = f_m(\mathbf{x}_z)$
  end
  for $m = 1$ to $|\mathcal{C}|$
    construct $\mathbf{v}_{mz} = \{\hat{y}'_{mz}\}_{n:\text{abs}(\rho_{mn}) \geq \theta}$
    predict $\hat{y}_{mz} = \tilde{f}_m(\mathbf{v}_{mz})$
  end
OUTPUT: concept scores $\{\hat{y}_{mz}\}$ (one can get binary
result with the boundary $\theta(r_{K-1})$; see Eqs. 2–4)
```

Table 1. Pseudo codes of the concept detection framework.

We also study concept selection by removing concepts whose absolute correlation values to the target concept are below a threshold θ. That is,

$$\mathbf{v}_{mi} = \{f_n(\mathbf{x}_i)\}_{n:\text{abs}(\rho_{mn}) \geq \theta}, \quad (11)$$

where abs$(\cdot)$ is an operator that takes the absolute value. Intuitively, the number of selected concepts $|\mathbf{v}_{mi}|$ decreases as θ is set larger and the actual number of $|\mathbf{v}_{mi}|$ depends on c_m but not on d_i. When $\theta = 0$, no concept selection is performed and all the concepts are utilized; when $\theta = 1$, no context fusion is conducted. For a concept that does not have strongly correlated concepts, $|\mathbf{v}_{mi}|$ would equal zero and we do not apply context fusion to it.

The algorithmic descriptions of the proposed concept detection framework is shown in Table 1.

5. EXPERIMENTAL RESULT

5.1 Experiment Setup

For fair comparison, each songs is converted to a standard format (22,050 Hz sampling frequency, 16 bits precision and mono channel) and represented by a 30-second segment starting from the initial 30th second of the song, a common practice in music classification.

For feature representation of a song we use the computer program MA toolbox [15] to extract Mel-frequency cepstral coefficients (MFCC), one of the most popular feature representation for audio signal processing. It is computed by taking the cosine transform of the short-term log

target concept $c_m = \mathcal{C}_m^{r_4}$	strongly positively correlated concepts $\mathcal{C}_m^{r_3}$	strongly negatively correlated concepts $\mathcal{C}_m^{r_1}$
emotion: angry/agressive	emotion: exciting/thrilling, powerful/strong genre: metal/hard rock, hip hop/rap, punk instrument: drum machine, electric guitar (distorted) song: fast tempo, heavy beat, high energy;	emotion: calming, laid-back, happy, loving, positive, tender instrument: piano song: positive feelings, texture acoustic
emotion: sad	emotion: calming/soothing, emotional/passionate instrument: female lead vocals song: quality, texture acoustic usage: going to sleep, intensive listening	emotion: arousing, carefree, happy instrument: drum set, male lead vocals song: high energy, positive feelings usage: cleaning the house
genre: jazz	emotion: calming, laid back, pleasant, tender, touching genre: bebop, contemporary R&B, cool jazz, swing instrument: piano, saxophone, trombone, trumpet	emotion: not calming, not loving genre: rock instrument: male lead vocals
song: very danceable	emotion: arousing, carefree, exciting, happy, light genre: dance pop, funk, swing, hip-hop/rap, pop, R&B usage: at a party, exercising	emotion: calming, laid-back, sad, tender genre: alternative, soft rock, rock

Table 2. Using the rules described in Section 3.2, we can obtain the highly correlated concepts for a target concept and partition the training data to four classes. This table shows some (only partial) highly correlated concepts for four concepts.

power spectrum expressed on a nonlinear perceptual-related mel-frequency scale. We use the default 23ms frame size with half overlapping to compute a bag of 20-dimensional MFCC vectors and then collapse the sequence of feature vectors into a single feature vector by taking the mean and standard deviation. As prior works [2], we also take the first-order derivatives of the MFCC vectors to capture temporal information, resulting in a 80-dimensional feature vector $\mathbf{x}_i$ for each song.

We randomly hold out 100 songs as the test set and use the remaining 313 songs for training. The evaluation process is repeated 100 times to compute the average accuracy, which is measured by average precision (AP), the approximation of the area under the recall/precision curve [10]. Let $\hat{\mathbf{p}}_m = \{\text{rank}(\hat{y}_{mi})\}_{i=1}^N$, where rank$(\hat{y}_{mi})$ is the ranking order of d_i in $\mathcal{D}$ according to $\hat{y}_{mi}$, we have

$$AP(\hat{\mathbf{p}}_m, A_m) = \frac{1}{rel} \sum_{j:A_{mj}=1} Prec@j, \qquad (12)$$

where $rel = |i : A_{mi} = 1|$ is the number of relevant objects (true positives) of concept c_m, and $Prec@j$ is the percentage of relevant objects in the top j objects in predicted ranking $\hat{\mathbf{p}}_m$. AP equals 1 when all the relevant objects are ranked at top. Since AP only shows the performance of a concept, we evaluate the performance in terms of mean average precision (MAP), the mean of APs for all concepts.

5.2 Evaluate Ordinal Regression

We first compare the performance of ordinal regression and multi-label classification. We use listNet for ordinal regression and SVM for multi-label classification.[2] Table 3 shows the MAP of different learning algorithms. It can be found that listNet(K=2) significantly outperforms SVM (p-value<0.01) for both the cal500 and cal500-lite lexicons, showing the effectiveness of ordinal regression. Setting $K = 4$ and leveraging concept correlation to the training process further improves the accuracy. The relative gain of listNet(K=4) over SVM is +10.89% and +9.35% for cal500 and cal500-lite, respectively.

	SVM	listNet(K=2)	listNet(K=4)
cal500	**0.2513**	0.2769	**0.2787**
cal500-lite	0.4323	0.4687	0.4727

Table 3. Evaluation of ordinal regression.

	SVM	listNet(K=4)
the 40 most freq. cpts	0.5113	0.5523 (+8.02%)
the medium freq. cpts	0.2182	0.2460 (+12.74%)
the 40 least freq. cpts	0.0690	0.0818 (+18.47%)
average	0.2513	0.2787 (+10.89%)

Table 4. The accuracy of concept detection for the cal500 concepts of different concept frequencies.

To investigate the detection accuracy of ordinal regression for concepts of different frequencies, we break down the cal500 concept lexicon to three groups: the 40 most frequent ones, the 40 least frequent ones, and the others. Table 4 shows the MAP of the concept groups of SVM and listNet(K=4). The correlations between concept frequency, accuracy of concept detection, and the relative performance gain of listNet(K=4) over SVM are salient. The detection accuracy is generally higher for frequent concepts, while the relative performance gain of listNet(K=4) is generally higher for rare concepts. This implies that the data imbalance problem is mitigated by listNet.

5.3 Evaluate Context Fusion and Concept Selection

We then evaluate the performance of context fusion (using DMF) with and without concept selection. We use listNet to train both the concept models and context models and vary the value of the concept selection threshold θ. Results shown in Table 5 lead to the following obser-

[2] We use SVM for its superior performance in classification problems. We implement it based on the LIBSVM library [16]. The parameters are tuned by a cross validation procedure to achieve better result: for SVM, we set the cost parameter C to 1000 and the gamma γ in the RBF kernel to 0.01; for listNet, we set the learning step η to 0.05.

	cal500	cal500-lite
listNet(K=4)	0.2787	0.4727
listNet(K=4)+DMF(θ=0)	0.2829	0.4873
listNet(K=4)+DMF(θ=0.1)	0.2911	**0.4882**
listNet(K=4)+DMF(θ=0.2)	**0.2924**	0.4854
listNet(K=4)+DMF(θ=0.3)	0.2856	0.4824
listNet(K=4)+DMF(θ=0.5)	0.2784	0.4754

Table 5. Evaluation of context fusion with different values of threshold θ (smaller θ selects more concepts).

vations. First, with mild concept selection, context fusion greatly improves concept detection. The MAP reaches 0.2924 (+4.92%) for cal500 and 0.4882 (+3.28%) for cal500-lite. This degree of performance gain is similar to that of applying context fusion to visual concept detection [13]. Second, without concept selection (θ=0), the performance of context fusion for cal500-lite is similar to the optimal one 0.4882, which may result from the fact that the detection accuracy of cal500-lite is generally high and thus directly leveraging all concepts is effective. On the contrary, due to the rather inconsistent accuracy, the detection of cal500 calls for concept selection to remove irrelevant concepts. Finally, setting θ too large removes most of the concepts and degrades accuracy. A mild value of θ exhibits the best result.

Table 6 shows the MAP of different semantic categories with and without context fusion. It can be found that context fusion with concept selection consistently improves all the semantic categories, especially for “emotion,” “genre,” and “usage.” In particular, because the detection accuracy of “genre” and “usage” are relatively low, concept selection is prerequisite for context fusion to be effective. In addition, due to the lack of strongly correlated concepts, context fusion does not improve the category “vocal.” Another interesting observation is the selected concepts often belong to “emotion,” “song,” or the same semantic category as the target concept. This evaluation demonstrates the importance of context fusion and concept selection.

6. CONCLUSION

In this paper, we have presented a novel framework of utilizing concept correlations to improve musical concept detection. A concept model is trained by an ordinal regression algorithm, which effectively utilizes the ordinal relationships among concepts and avoids the data imbalance problem of the commonly-used classification methods. A context model is then trained to improve the detection result by leveraging the co-occurrence patterns among concepts. We also employ a concept selection method to keep irrelevant concepts from being used in context fusion. Experimental results show that ordinal regression outperforms the conventional multi-label classification method by a great margin; a +10.89% relative gain in mean average precision is achieved. With mild concept selection, context fusion further improves the detection accuracy to 0.2924 for the 174 musical concepts of cal500.

	listNet	+DMF(θ=0)	+DMF(θ=0.2)
emotion	0.4272	0.4369 (+2%)	0.4522 (**+6%**)
genre	0.1731	0.1769 (+2%)	0.1890 (**+9%**)
instrument	0.2321	0.2345 (+1%)	0.2383 (+3%)
song	0.4233	0.4302 (+2%)	0.4345 (+3%)
usage	0.1753	0.1750 (0%)	0.1952 (**+11%**)
vocal	0.1981	0.1998 (+1%)	0.1997 (+1%)
average	0.2787	0.2829 (+2%)	0.2924 (+5%)

Table 6. The accuracy of concept detection for the cal500 concepts of different semantic categories.

7. ACKNOWLEDGEMENT

This work was supported by the National Science Council of Taiwan under contract NSC 97-2221-E-002-111-MY3.

8. REFERENCES

[1] B. Whitman and R. Rifkin: “Musical query-by-description as a multicalss leanring problem,” *MMSP*, pp. 153–156, 2002.

[2] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet: “Semantic annotation and retrieval of music and sound effects,” *IEEE Trans. Audio, Speech and Language Processing*, Vol. 16, No. 2, pp. 467–476, 2008.

[3] M. I. Mandel and D. P. W. Ellis: “Multiple-instance learning for music information retrieval,” *ISMIR*, 2008.

[4] Z.-Y. Duan, L. Lu, and C.-S. Zhang: “Collective annotation of music from multiple semantic categories,” *ISMIR*, pp. 237–242, 2008.

[5] T. Bertin-Mahieux et al: “Autotagger: A model for predicting social tags from acoustic features on large music databases,” *J. New Music Research*, Vol. 37, No. 2, pp. 115–135, 2008.

[6] J.-J. Aucouturier, F. Pachet, P. Roy, and A. Beurivé: “Signal+context=better classification,” *ISMIR*, 2007.

[7] Z.-S. Chen, J.-M. Zen, and J.-S. Jang: “Music annotation and retrieval system using anti-models,” *AES Convention*, 2008.

[8] E. Law et al, “Tagatune: a game for music and sound annotation,” *ISMIR*, pp. 361–364, 2007.

[9] R. Herbrich et al: “Support vector learning for ordinal regression,” *ICANN*, pp. 97–102, 1999.

[10] Y. Yue et al: “A support vector method for optimizing average precision,” *SIGIR*, pp. 271–278, 2007.

[11] F. Xia et al: “Listwise approach to learning to rank: Theory and algorithm,” *ICML*, pp. 1192–1199, 2008.

[12] J. Smith, M. Naphade, and A. Natsev: “Multimedia semantic indexing using model vectors,” *ICME*, pp. 445–448, 2003.

[13] Y.-H. Yang et al: “Online reranking via ordinal informative concepts for context fusion in concept detection and video search,” *IEEE Trans. Circuits and Sys. for Video Tech.*, 2009.

[14] M. Naphade et al: “Large-scale concept ontology for multimedia,” *IEEE Multimedia Magazine*, Vol. 13, No. 3, pp. 86–91, 2006.

[15] E. Pampalk: “A Matlab toolbox to compute music similarity from audio,” *ISMIR*, 2004. http://www.ofai.at/ elias.pampalk/ma/.

[16] C.-C. Chang and C.-J. Lin: “LIBSVM: a library for support vector machines,” 2001.

TEMPLATE-BASED CHORD RECOGNITION : INFLUENCE OF THE CHORD TYPES

Laurent Oudre[1], **Yves Grenier**[1], **Cédric Févotte**[2]
[1]Institut TELECOM ; TELECOM ParisTech ; CNRS LTCI
[2]CNRS LTCI ; TELECOM ParisTech
37-39 rue Dareau, 75014 Paris, France
{oudre,grenier,fevotte}@telecom-paristech.fr

ABSTRACT

This paper describes a fast and efficient template-based chord recognition method. We introduce three chord models taking into account one or more harmonics for the notes of the chord. The use of pre-determined chord models enables to consider several types of chords (major, minor, dominant seventh, minor seventh, augmented, diminished...). After extracting a chromagram from the signal, the detected chord over a frame is the one minimizing a measure of fit between the chromagram frame and the chord templates. Several popular measures in the probability and signal processing field are considered for our task. In order to take into account the time persistence, we perform a post-processing filtering over the recognition criteria. The transcription tool is evaluated on the 13 Beatles albums with different chord types and compared to state-of-the-art chord recognition methods. We particularly focus on the influence of the chord types considered over the performances of the system. Experimental results show that our method outperforms the state-of-the-art and more importantly is less computationally demanding than the other evaluated systems.

1. INTRODUCTION

Chord transcription is a compact representation of the harmonic content and structure of a song. Automatic chord transcription finds many applications in the field of Musical Information Retrieval such as song identification, query by similarity or structure analysis.

The features used for chord recognition may differ from a method to another but are in most cases variants of the 12-dimensional *Pitch Class Profiles* [1]. Every component represents the spectral energy of a semi-tone on the chromatic scale regardless of the octave. The succession of these chroma vectors over time is called *chromagram* : the chord recognition task consists in outputting a chord label for every chromagram frame.

The first chord recognition systems consider many chord types. The method proposed by Fujishima [1] considers 27 chord types. The transcription is done either by minimizing the Euclidean distance between *Pitch Class Profiles* and 12-dimensional chord templates constituted by 1's (for the chromas present in the chord) and 0's (for the other chromas) or by maximizing a weighted dot product. Sheh & Ellis [2] use a Hidden Markov Model composed of 147 hidden states each representing a chord (7 types of chords and 21 root notes). All the HMM parameters are learned by a semi-supervised training with an EM algorithm.

These two methods have been improved upon by reducing the number of chord types considered. Fujishima's system is improved in [3] by reducing the number of chords types from 27 to 4 (major, minor, augmented, diminished) and by calculating a more elaborate chromagram including notably a tuning algorithm. Chord transcription is then realized by retaining the chord with larger dot product between the chord templates and the chromagram frames. Sheh & Ellis method is modified in [4] : the number of hidden states is reduced from 147 to 24 by only considering major and minor chords for the 12 semi-tones root notes. Musical knowledge is introduced into the model by initializing the HMMs parameters with values inspired by musical and cognitive theory. Since then, almost all the chord transcription methods [5], [6], [7], [8], [9], only consider major and minor chords.

Our chord recognition system is based on the intuitive idea that for a given 12-dimensional chroma vector, the amplitudes of the chromas present in the chord should be larger than the ones of the non-played chromas. By introducing chord templates for different chord types and roots, the chord present on a frame should therefore be the one whose template is the *closest* to the chroma vector according to a specific measure of fit.

The paper is organized as follows. Section 2 gives a description of our recognition system. Section 3 describes the corpus and the protocol of evaluation. Section 4 presents the results of our system, a study on the influence of the chord types considered, a comparison with the state-of-the-art and an analysis of the frequent errors. Finally the main conclusions of this work are summarized in Section 5.

2. DESCRIPTION OF THE SYSTEM

2.1 General idea

Given N successive chroma vectors $\{\mathbf{c}_n\}_n$, K chord templates $\{\mathbf{p}_k\}_k$ and a measure of fit D, we define :

$$d_{k,n} = D\left(h_{k,n}\,\mathbf{c}_n; \mathbf{p}_k\right). \tag{1}$$

$h_{k,n}$ is a scale parameter whose role is to fit the chroma vector $\mathbf{c}_n$ with the chord template $\mathbf{p}_k$ according to the measure of fit used. In practice, $h_{k,n}$ is calculated such as :

$$h_{k,n} = \operatorname*{argmin}_h D\left(h\,\mathbf{c}_n; \mathbf{p}_k\right). \tag{2}$$

The detected chord $\hat{k}_n$ for frame n is then the one minimizing the set $\{d_{k,n}\}_k$:

$$\hat{k}_n = \operatorname*{argmin}_k \{d_{k,n}\}. \tag{3}$$

In our system, the chroma vectors are calculated from the music signal with the same method as Bello & Pickens [4]. The frame length is set to 753 ms and the hop size is set to 93 ms. We use the code kindly provided by these authors.

We have omitted for sake of conciseness the expressions of $d_{k,n}$ and $h_{k,n}$ which are easily obtained by canceling the gradient of (1) wrt $h_{k,n}$.

2.2 Chord models

The intuitive chord model is a simple binary mask constituted of 1's for the chromas present in the chord and 0's for the other chromas [1], [3].

Yet, the information contained in a chromagram captures not only the intensity of every note but a blend of intensities for the harmonics of every note. Like Gomez [10] and Papadopoulos [5], we assume an exponentially decreasing spectral profile for the amplitudes of the partials. An amplitude of 0.6^{i-1} is added for the i^{th} harmonic of every note in the chord.

In our system three chord models are defined, corresponding to 1, 4 or 6 harmonics. Examples for C major and C minor chords are displayed on Figure 1.

From these three chord models we can build chord templates for all types of chords (major, minor, dominant seventh, diminished, augmented,...). By convention in our system, the chord templates are normalized so that the sum of the amplitudes is 1.

2.3 Measures of fit

We consider for our recognition task several measures of fit, popular in the field of signal processing : the **Euclidean distance** (later referred as *EUC*), the **Itakura-Saito divergence** [11] and the **Kullback-Leibler divergence** [12].

Since the Itakura-Saito and Kullback-Leibler divergence are not symmetrical, they can be calculated in two ways. $D\left(h_{k,n}\,\mathbf{c}_n|\mathbf{p}_k\right)$ will respectively define *IS1* and *KL1*, while $D\left(\mathbf{p}_k|h_{k,n}\,\mathbf{c}_n\right)$ will define *IS2* and *KL2*.

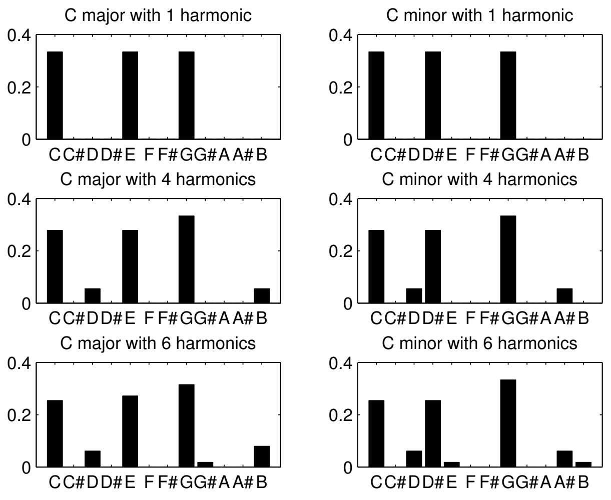


Figure 1. Chord templates for C major / C minor with 1, 4 or 6 harmonics.

2.4 Filtering methods

In order to take into account the time-persistence, we introduce some post processing filtering methods which work upstream on the calculated measures and not on the sequence of detected chords.

The new criterion $\tilde{d}_{k,n}$ is based on L successive values $\{d_{k,n'}\}_{n-\frac{L-1}{2}\leq n'\leq n+\frac{L-1}{2}}$ previously calculated. In our system two types of filtering are used.

The **low-pass filtering** takes the mean of the L values. It tends to smooth the output chord sequence and to reflect the long-term trend in the chord change.

The **median filtering** takes the median of the L values. It has been widely used in image processing and is particularly efficient to correct random errors.

In every case, the detected chord $\hat{k}_n$ on frame n is the one that minimizes the set of values $\left\{\tilde{d}_{k,n}\right\}_k$:

$$\hat{k}_n = \operatorname*{argmin}_k \left\{\tilde{d}_{k,n}\right\} \tag{4}$$

3. EVALUATION

3.1 Corpus

The evaluation database used in this paper is made of the 13 Beatles albums (180 songs, PCM 44100 Hz, 16 bits, mono). The chord annotations for these 13 Beatles albums are kindly provided by Harte and Sander [13].

In these annotation files, 17 types of chords and one 'no chord' label (N) corresponding to silences or untuned material are present.

The most common chord types in the corpus are major (63.89% of the total duration), minor (16.19%), dominant seventh (7.17%) and 'no chord' states (4.50%). Figure 2 shows the repartition of the chord types among the 13 albums of the Beatles. We can see that the number of major, minor and dominant seventh chords varies much with the album. Yet, the last six albums clearly contain more chord

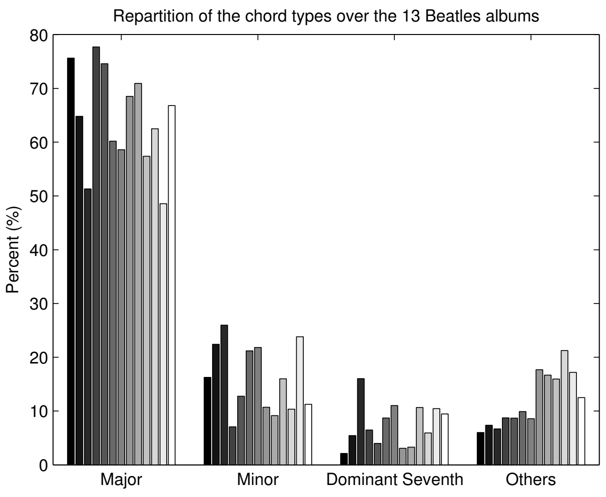


Figure 2. Repartition of the chord types as percentage of the total duration for the 13 Beatles albums.

types (other than major, minor and dominant seventh) than the first seven ones.

3.2 Protocol of evaluation

The evaluation method used in this paper corresponds to the one used in MIREX 08 for the Audio Chord Detection task. [1]

As the evaluation method only takes into account major and minor chords, the 17 types of chords present in the annotation files are first mapped into major and minor types following the rules used in MIREX 08 :

- major : maj, dim, aug, maj7, 7, dim7, hdim7, maj6, 9, maj9, sus4, sus2
- minor : min, min7, minmaj7, min6, min9

For the systems detecting more chord types (dominant seventh, diminished, etc.), once the chords have been detected with their appropriate models, they are then mapped to the major and minor following the same rules than for the annotation files.

A score is calculated for each song as the ratio between the lengths of the correctly analyzed chords and the total length of the song. The final *Average Overlap Score (AOS)* is then obtained by averaging the scores of all the 180 songs. An example of calculation of an Overlap Score is presented on Figure 3.

4. RESULTS

The five previously described measures of fit (*EUC*, *IS1*, *IS2*, *KL1* and *KL2*), three chord models (1, 4 or 6 harmonics) and two filtering methods (low-pass and median) with neighborhood sizes from $L = 1$ to $L = 25$ are tested. For every method we only present the results for the optimal parameters (measure of fit, chord models, filtering method and neighborhood size).

[1] http://www.music-ir.org/mirex/2008/

4.1 Results with major/minor chord types

Considering only major and minor chords (like most of the chord recognition methods of the actual state-of-art), we obtain a *Average Overlap Score* of 0.70 over the 13 Beatles albums. The optimal parameters are the Kullback-Leibler divergence *KL2*, the single harmonic chord model and the median filtering with a neighborhood size of $L = 17$.

4.2 Introduction of other chord types

The simplicity of our method allows to easily introduce chord templates for chord types other than major and minor : we study here the influence of the chord types considered over the performances of our system. The choice of these chord types is guided by the statistics on the corpus previously presented : we introduce in priority the most common chords types of the corpus.

4.2.1 Dominant seventh and minor seventh chords

In the Beatles corpus, the two most common chord types other than major and minor are dominant seventh *(7)* and minor seventh *(min7)* chords. The results for major, minor, dominant seventh and minor seventh chords are presented in Table 1. The score displayed in a case is the best *Average Overlap Score* obtained by considering the chord types of the corresponding row and column.

	min	min7	min & min7
maj	0.70	0.64	0.69
7	0.69	0.63	0.65
maj & 7	0.71	0.66	0.69

Table 1. Average Overlap Scores with major, minor, dominant seventh and minor seventh chords.

The best results are obtained by detecting major, minor and dominant seventh chords, with the Kullback-Leibler divergence *KL2*, the single harmonic chord model and the median filtering with $L = 17$ giving a recognition rate of 71%. Only the introduction of dominant seventh chords, which are very common in the Beatles corpus, enhances the results. The introduction of minor seventh chords, which are less common, degrades the results. Indeed, the structure of minor seventh chords (for example *Cmin7*) leads to confusion between the actual minor chord and the relative major chord (*E*♭ in our example).

4.2.2 Augmented and diminished chords

Augmented and diminished chords have been considered in many template-based chord recognition systems [1], [3]. Interestingly, while the augmented and diminished chords are very rare in the Beatles corpus (respectively 0.62% and 0.38% of the total length), the introduction of chord templates for augmented and diminished chords does not degrade the results. We obtain a recognition rate of 69% by considering major, minor, augmented and diminished chords and of 71% by taking into account major, minor, dominant seventh, augmented and diminished chords.

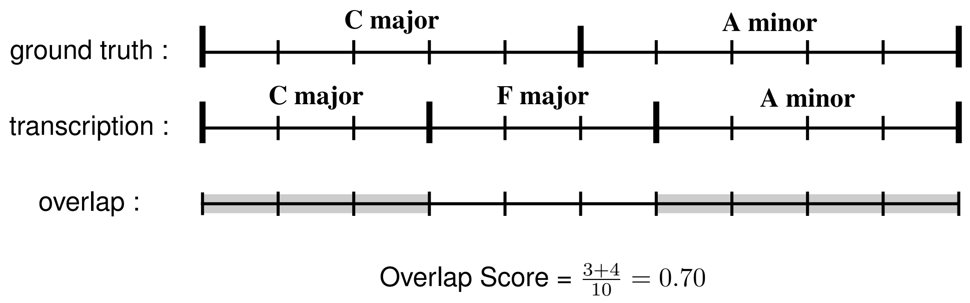


Figure 3. Example of calculation of an Overlap Score.

4.2.3 Other chord types

The introduction of other chord types (ninth, major seventh, sus4, etc.) does not improve the results. This can be explained either by the structures of the chords which can lead to confusions with other chord types or by the low number of chords of these types in the Beatles corpus. Indeed, the introduction of a model for a new chord type gives a better detection for chords of this type but also leads to new errors such as false detections. Therefore only frequent chords types should be introduced, ensuring that the enhancement caused by the better recognition of these chord types is larger than the degradation of the results caused by the false detections.

4.3 Influence of the album

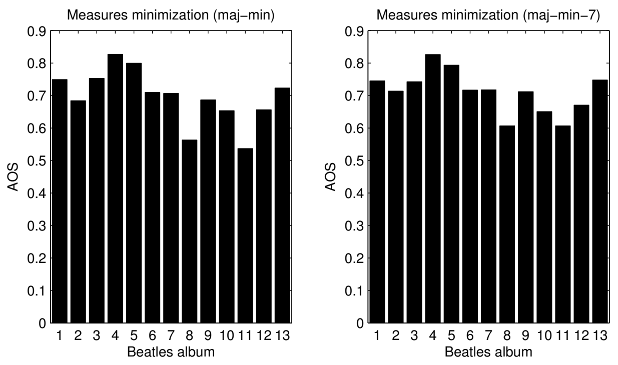


Figure 4. Average Overlap Scores for the 13 Beatles albums (in chronological order) for the major/minor and the major/minor/dominant seventh methods.

We can see on Figure 4 that results are better for the first seven albums : this can be explained by the low number of chords other than major, minor and dominant seventh on these albums (see Figure 2). Surprisingly the introduction of dominant seventh chords tend to improve results not necessarily on albums containing many dominant seventh chords (for example album number 3) but on albums containing many chords other than major, minor and dominant seventh (for example albums number 8 & 11).

4.4 State-of-the-art

Our method is now compared to the following methods that entered MIREX 08.

Bello & Pickens [4] use 24-states HMM with musically inspired initializations, Gaussian observation probability distributions and EM-training for the initial state distribution and the state transition matrix.

Ryynänen & Klapuri [6] use 24-states HMM with observation probability distributions computed by comparing low and high-register profiles with some trained chord profiles. EM-training is used for the initial state distribution and the state transition matrix.

Khadkevich & Omologo [7] use 24 HMMs : one for every chord. The observation probability distributions are Gaussian mixtures and all the parameters are trained through EM.

Pauwels, Verewyck & Martens [8] use a probabilistic framework derived from Lerdahl's tonal distance metric for the joint tasks of chords and key recognition.

These methods have been tested with their original implementations on the same Beatles corpus than before and evaluated with the same protocol (AOS). Results of this comparison with the state-of-the-art are presented on Table 2.

	AOS	Time
Our method (Maj-Min-7)	0.71	796s
Bello & Pickens	0.70	1619s
Our method (Maj-Min)	0.70	790s
Ryynänen & Klapuri	0.69	1080s
Khadkevich & Omologo	0.64	1668s
Pauwels, Varewyck & Martens	0.62	12402s

Table 2. Comparison with the state-of-the-art.

First of all it is noticeable that all the methods give rather close results : there is only a 9% difference between the methods giving the best and worse results. Our method gives the best results, but more importantly with a very low computational time. It is indeed twice as fast as the best state-of-the-art method (Bello and Pickens).

4.5 Analysis of the errors

In most chord transcription systems, the errors are often caused by the structural similarity (common notes) and the harmonic proximity between the real chord and the wrongly detected chord.

Two chords are likely to be mistaken one for another when they *look alike*, that is to say, when they share notes (especially in template-based systems). Given a major or minor chord, there are 3 chords which have 2 notes in common with this chord : the parallel minor/major, the relative minor/major (or submediant) and the mediant chord.

Besides the structural similarity, errors can also be caused by the harmonic proximity between the original and the detected chord. Figure 5 pictures the doubly nested circle of fifths which represents the major chords (capital letters), the minor chords (lower-case letters) and their harmonic relationships. The distance linking two chords on this doubly nested circle of fifths is an indication of their harmonic proximity.

Given a major or minor chord, the 4 closest chords on this circle are the relative (submediant), mediant, subdominant and dominant. One can notice that these 4 chords are also structurally close to the original chord, since they share 1 or 2 notes with it.

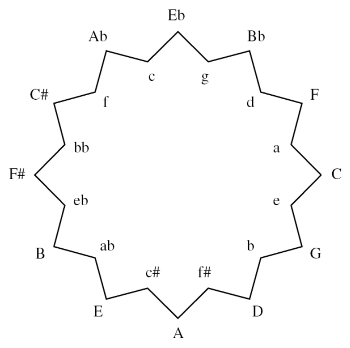


Figure 5. Doubly nested circle of fifths [4].

We have therefore brought out 5 potential sources of errors among the 23 possible ones (i.e., the 23 other wrong candidates for one reference chord). Examples of these potential sources of errors for C major and C minor chords are displayed on Figure 6.

Reference chord	**C**	**Cm**
parallel	Cm	C
relative (submediant)	Am	A♭
mediant	Em	E♭
subdominant	F	Fm
dominant	G	Gm

Figure 6. Particular relationships between chords and potential sources of errors : examples for C major and C minor chords.

Figure 7 displays the repartition of these error types as a percentage of the total number of errors for every evaluated method. Errors due to the bad detection of the 'no chord' states are represented with the 'no chord' label.

The main sources of errors correspond to the situations previously described and to the errors caused by silences ('no chord'). Actually, in most methods, the 5 types of errors previously considered (over the 23 possible ones) represent approximately 60% of the errors.

The introduction of the dominant seventh chords clearly reduces the proportion of the errors due to relative (submediant) and mediant (-9%). Another noteworthy result is that the methods by Ryynänen & Klapuri, Bello & Pickens and our major/minor method approximately have the same error repartition despite the different structures of the methods, which proves that the semantic of the errors is inherent to the task. Pauwels, Varewyck & Martens' system is mostly penalized by the wrong detection of the 'no chord' states, when Khadkevich & Omologo's method produces a wider range of errors.

5. CONCLUSION

Our system offers a novel perspective about chord detection. The joint use of popular measures and filtering methods distinguishes from the predominant HMM-based approaches. The introduction of chord templates allows to easily consider many chord types instead of only major and minor chords. Since our method is only based on the chromagram no information about style, rhythm or instruments is required and thank to the fact that no training or database is needed, the computation time can be kept really low.

6. ACKNOWLEDGMENT

The authors would like to thank J. Bello, M. Khadkevich, J. Pauwels, M. Ryynänen for making their code available. We also wish to thank C. Harte for his very useful annotation files.

This work was realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

7. REFERENCES

[1] T. Fujishima. Realtime chord recognition of musical sound: a system using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 464–467, Beijing, China, 1999.

[2] A. Sheh and D.P.W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 185–191, Baltimore, MD, 2003.

[3] C.A. Harte and M.B. Sandler. Automatic chord identification using a quantised chromagram. In *Proceedings of the Audio Engineering Society*, Barcelona, Spain, 2005.

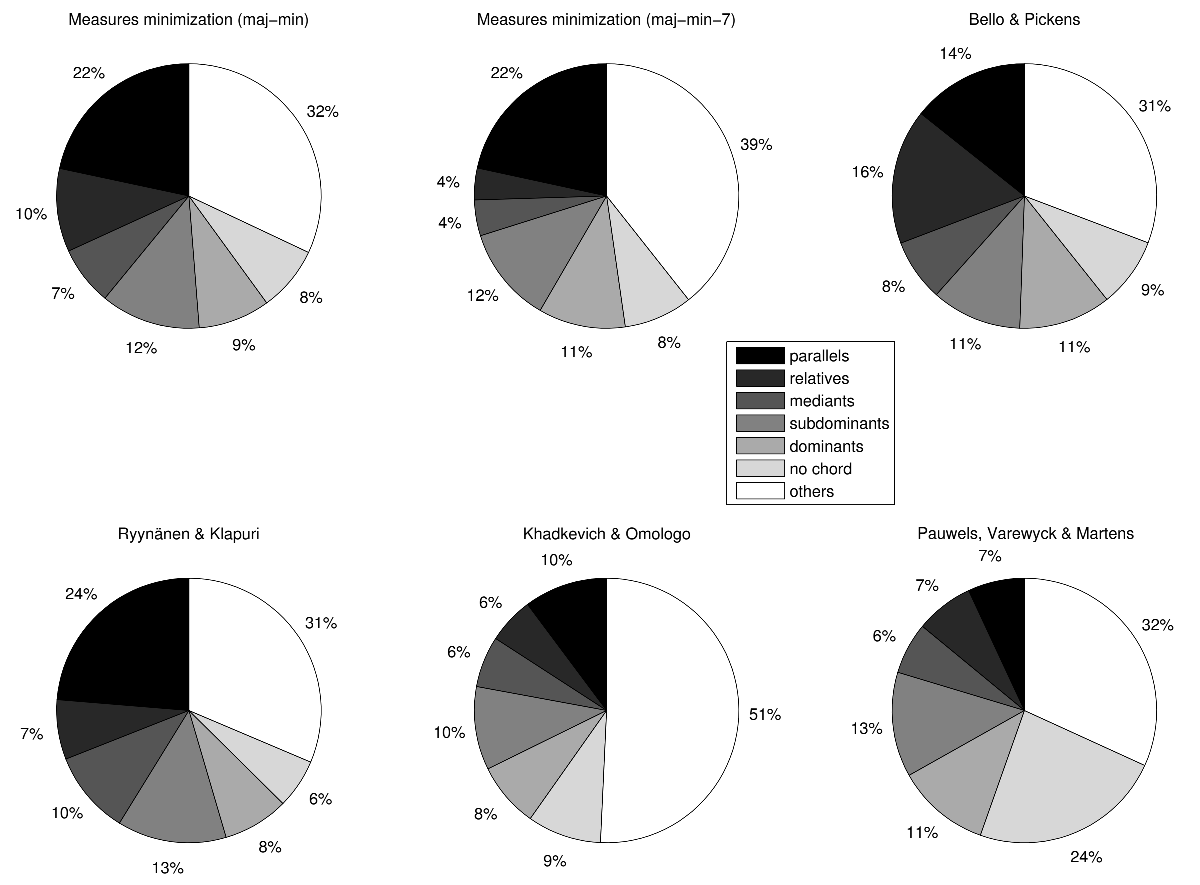


Figure 7. Repartition of the errors as a percentage of the total number of errors.

[4] J.P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 304–311, London, UK, 2005.

[5] H. Papadopoulos and G. Peeters. Large-scale study of chord estimation algorithms based on chroma representation and HMM. In *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, pages 53–60, Bordeaux, France, 2007.

[6] M.P. Ryynänen and A.P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[7] M. Khadkevich and M. Omologo. Mirex audio chord detection. Abstract of the Music Information Retrieval Evaluation Exchange, 2008.

[8] J. Pauwels, M. Varewyck, and J-P. Martens. Audio chord extraction using a probabilistic model. Abstract of the Music Information Retrieval Evaluation Exchange, 2008.

[9] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):291–301, 2008.

[10] E. Gómez. Tonal description of polyphonic audio for music content processing. In *Proceedings of the INFORMS Computing Society Conference*, volume 18, pages 294–304, Annapolis, MD, 2006.

[11] F. Itakura and S. Saito. Analysis synthesis telephony based on the maximum likelihood method. In *Proceedings of the International Congress on Acoustics*, pages 17–20, Tokyo, Japan, 1968.

[12] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[13] C. Harte, M. Sandler, S. Abdallah, and E. Gomez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 66–71, London, UK, 2005.

TAG-AWARE SPECTRAL CLUSTERING OF MUSIC ITEMS

Ioannis Karydis[i] **Alexandros Nanopoulos**[ii] **Hans-Henning Gabriel**[iii] **Myra Spiliopoulou**[iii]
[i]Department of Informatics, Ionian University, Greece
[ii]Institute of Informatics, Hildesheim University, Germany
[iii]Faculty of Computer Science, Otto-von-Guericke-University Magdeburg, Germany
karydis@ionio.gr, nanopoulos@ismll.de, {hgabriel, myra}@iti.cs.uni-magdeburg.de

ABSTRACT

Social tagging is an increasingly popular phenomenon with substantial impact on Music Information Retrieval (MIR). Tags express the personal perspectives of the user on the music items (such as songs, artists, or albums) they tagged. These personal perspectives should be taken into account in MIR tasks that assess the similarity between music items. In this paper, we propose an novel approach for clustering music items represented in social tagging systems. Its characteristic is that it determines similarity between items by preserving the 3-way relationships among the inherent dimensions of the data, i.e., users, items, and tags. Conversely to existing approaches that use reductions to 2-way relationships (between items-users or items-tags), this characteristic allows the proposed algorithm to consider the personal perspectives of tags and to improve the clustering quality. Due to the complexity of social tagging data, we focus on spectral clustering that has been proven effective in addressing complex data. However, existing spectral clustering algorithms work with 2-way relationships. To overcome this problem, we develop a novel data-modeling scheme and a tag-aware spectral clustering procedure that uses tensors (high-dimensional arrays) to store the multigraph structures that capture the personalised aspects of similarity. Experimental results with data from Last.fm indicate the superiority of the proposed method in terms of clustering quality over conventional spectral clustering approaches that consider only 2-way relationships.

1. INTRODUCTION

Music Information Retrieval (MIR) is highly interdisciplinary a field that, due to the nature of music, requires an increased amount of contextual information for most of its processes [1]. One popular method that supplies this contextual information is the practice of *social-tagging*. Social tags are shared, free-text keywords that web users can assign to music items, such as artists, albums, songs, playlists, genres, etc. The popularity of music tagging rests with the easy and effective organisation it produces, in contrast to the obscure and ambiguous hierarchical classification (in terms of genre, mood, etc). Social tagging assists the retrieval of items and social expression of taste [2]. Therefore, tags over music items reflect conveniently the personalised opinion of users for these items.

Social tagging attracts increasing attention and MIR systems like Last.fm [3] and MyStrands [4] contain a body of collected data in which data mining is challenging and promising. One of the most essential data mining tasks is the clustering of music data to assist their organisation, the creation of playlists and the model-based music recommendation. However, several existing MIR approaches consider clustering of data based solely on features extracted directly from the audio. In contrast, the proposed approach is based on user-generated content, in the form of tags, in order to include contextual information that would be otherwise non-extractable from the content of items.

Data from social-tagging systems have 3 inherent dimensions: the users, the music items, and the tags. Moreover, they contain 3-way relationships of the form items–users–tags between these dimensions. Thus, there is a clear difference between just knowing that a tag has been applied to an item regardless by which users, and knowing the specific users that applied this tag to the item. The reason is that in the latter case the tag expresses the personalised perspective of the specific users on the item. Clustering of music items with existing algorithms requires the suppression of the 3 dimensions and the reduction of their 3-way relationships into 2-way of the form items–users or items–tags. This is because most existing clustering algorithms model the data in 2-dimensional arrays whose rows correspond to items and columns to features. Thus, clustering can be performed over items-users or items-tags arrays, but not without breaking the original 3-way relationships between items-users-tags. However, such approach may incur loss of valuable information contained in the 3-way relationships.

To address the complexity of data from social tagging systems, we focus on the popular family of spectral clustering algorithms. This type of clustering algorithms work on a similarity graph that connects every item to its k nearest-

THE SECOND AUTHOR GRATEFULLY ACKNOWLEDGES THE PARTIAL CO-FUNDING OF HIS WORK THROUGH THE EUROPEAN COMMISSION FP7 PROJECT MYMEDIA (WWW.MYMEDIAPROJECT.ORG) UNDER THE GRANT AGREEMENT NO. 215006.

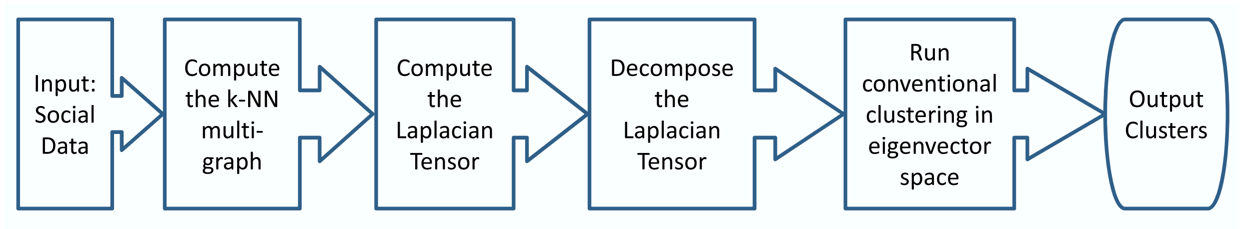

Input: Social Data
Compute the k-NN multi-graph
Compute the Laplacian Tensor
Decompose the Laplacian Tensor
Run conventional clustering in eigenvector space
Output Clusters

Figure 1. The steps followed by the proposed approach.

neighbors (k-NN) and map each item to a feature space defined by eigenvectors of the similarity graph. Spectral clustering algorithms have been proven effective in addressing complex data [5]. However, existing spectral clustering algorithms cannot be used directly for data from social tagging systems without suppressing the 3 dimensions in order to consider only either items-users or items-tags relationships. The reason is that existing spectral clustering algorithms form the k-NN similarity graph based on the *single value* of similarity between each pair of items.

To overcome the problems of existing approaches and avoid breaking the original 3-way relationships existing in social-tagged data, we propose the extension of spectral clustering in order to become tag-aware and directly handle all present dimensions. Our technical contributions towards this objective are the following: (i) We provide the insight that *multiple similarity values* between each pair of items should be used to account for the fact that when all 3 dimensions are considered, then similarity between two items depends both on the users who tagged them and the tags they assigned, a fact that leads to several similarity values between them. (ii) To support multiple similarity values, we extend the modeling based on k-NN similarity graphs by using k-NN similarity multigraphs, which allow the existence of multiple edges between two nodes. (iii) We extend existing spectral clustering algorithms to consider the k-NN similarity multigraphs by extracting information about eigenvectors from *tensors* (i.e., multidimensional arrays). (iv) We perform experiments with real data crawled from Last.fm and compare the proposed method against conventional spectral clustering that suppresses the original data and consider only 2-way relationships (either items-users or items-tags) in terms of quality of the final clustering.

The rest of this article is organised as follows. Section 2 reviews related work. Section 3 presents an overview of the proposed approach, whereas Section 4 describes the proposed data modeling and Section 5 the proposed clustering algorithm. Experimental results are detailed in Section 6. Finally, Section 7 concludes the article.

2. RELATED WORK

Clustering tagged music data, as well as their visualisation, has also been the focus of the research of Lehwark et al. [6]. In the interest of discovering new music based on the semantic organisation provided by tags on music data, they propose the use of the Emergent-Self-Organising-Map (ESOM) for the clustering of tagged data. Additionally, they also utilise U-Map in order to provide a visually appealing user interface and an intuitive way of exploring new content. Differently from this approach, we apply spectral clustering in contrast to ESOM while our focus is on multiple pairwise similarities in contrast to visualisation of the produced clusters.

Levy et al. [7], investigate the performance of models for varying latent dimensions examining the alteration of low-dimensional semantic representations discriminative capability in searching music collections. This approach is different than the one presented in our work, as we focus on multiple pairwise similarities on the music data for the purpose of clustering the music items, in contrast to [7] where different models are tested in order to uncover emergent semantics from social tags for music.

The clustering of music data has received extensive attention from the MIR community. Most research aims in genre classification (readers are suggested [8] for a detailed survey of the area) as the classification emerging is based on objective similarity measures from the data, thus avoiding the constraints possed by fixed taxonomies, which may be difficult to define as well as suffer from ambiguities and inconsistencies. Using a set of extracted features from the content of the music data, and a similarity measure for the comparison of the data, clustering algorithms organise music data in clusters of similar objects.

Symeonidis et al. [9] proposed dimensionality reduction using higher order SVD for the purposes of personalised music recommendation. That is, given a user and a tag, their purpose is to predict how likely is the user to label a specific music item with this tag. However, conversely from [9] we use tensor factorisation for extracting spectral information and performing spectral clustering, not for predicting recommendations.

3. OVERVIEW OF PROPOSED APPROACH

This section outlines the proposed approach. The steps that will be described in the following are depicted (for reference) in Figure 1.

Existing (non tag-aware) spectral clustering algorithms [5] first compute the k-NN similarity graph, which connects every item with its k-NN. Next, the Laplacian graph of the k-NN similarity graph is used instead, because of the benefits it offers, i.e., it is always positive-semidefinite (allowing its eigenvector decomposition) and the number of times 0 appears as its eigenvalue is the number of connected components in the k-NN similarity graph. Due to these convenient properties, if c clusters are required to be found, spectral clustering algorithms proceed by computing the c eigenvectors that correspond to the c smallest eigenvalues, and represent each original item as a c-dimensional vector whose coordinates are the corresponding values within the c eigenvectors. With this representation, they cluster the c-dimensional vectors using simple algorithms, like k-means or hierarchical agglomerative.

As described in Introduction, differently from conventional spectral clustering algorithms, our proposed approach considers multiple similarity values between each pair of items. In particular, let U be the set of all users. For a given tag t, let $U_1 \subseteq U$ be the set of users that tagged an item i_1 with t, whereas $U_2 \subseteq U$ be the set of users that tagged an item i_2 with t too. We can define a similarity value between i_1 and i_2 as follows. We form two vectors v_1 and v_2, both with $|U|$ elements that are set to 1 at positions that correspond to the users contained in U_1 and U_2, respectively, whereas all rest positions are set to 0. Therefore, the similarity between i_1 and i_2 is given by the cosine measure between the two vectors v_1 and v_2. Since the above process can be repeated for all tags, the result is several similarity values between each pair of items i_1 and i_2. The set of all multiple similarity values are tag-aware and reflect the personalised aspect of similarity perceived by the users (e.g., two users may tag the same item but using entirely different tags).

To account for the various similarity values between each pair of items, we extend (Section 4) the k-NN similarity graph to a k-NN multidigraph that is the union of multiple simple k-NN graphs, one for each distinct tag. The adjacency matrix of a k-NN multidigraph forms a tensor, i.e., a multidimensional array. In order to attain the aforementioned advantages of the Laplacian graph, we propose a method (Section 5.1) to extent towards the construction of the Laplacian multidigraph, whose adjacency matrix is again represented as a tensor. To map each item to a feature space comprised from spectral information extracted from the Laplacian tensor, we describe (Section 5.2) how to use tensor factorisation that extends SVD to multidimensional arrays. Finally, based on the computed features, we describe (Section 5.3) how the clustering is performed. To help comprehension, we use the data from the following example.

Example 1 (Data representation). We assume 3 users that assign tags to 4 music items (henceforth 'items' for simplicity) from a tag-set with 3 tags. Each assignment comprises a triple of the form (user, item, tag). The 9 triples of the example are given in Table 1, whereas we additionally denote (in the first column) the ID of the triple. The corresponding view of the data as tripartite graph is depicted in Figure 2. In this figure, the numbered edges correspond to the triple IDs in Figure 2a. For instance, the first triple (ID = 1) is: Alice tagged Elvis as Classic. In Figure 2 this corresponds to the path consisting of all edges labelled as 1. To avoid cluttering the figure, parallel edges (i.e., edges between the same two nodes) with different labels are depicted as one with different labels separated by comma. In this example, we assume that Elvis and Beatles form one cluster, whereas Mozart and Bach form a second cluster. This follows by observing in Figure 2 that, although users tag items from both clusters, they assign different tags to the first cluster than the second. Therefore, the relationships between users-items alone are not able to determine a clustering structure among the items. In contrast, when considering the multi-way relationships between users-items-tags, we are able to better detect the clustering of items. Although this simple example highlights the advantage of preserving the multi-way relationships compared to considering only item-user relationships, our experimental results show the advantages compared to the consideration of only item-tag relationships, as well. □

ID	User	Item	Tag
1	Alice	Elvis	Classic
2	Bob	Beatles	Classic
3	Bob	Elvis	Classic
4	Bob	Mozart	Symphonic
5	Joe	Mozart	Symphonic
6	Joe	Bach	Symphonic
7	Alice	Mozart	Orchestral
8	Joe	Mozart	Orchestral
9	Joe	Bach	Orchestral

Table 1. Example of input data

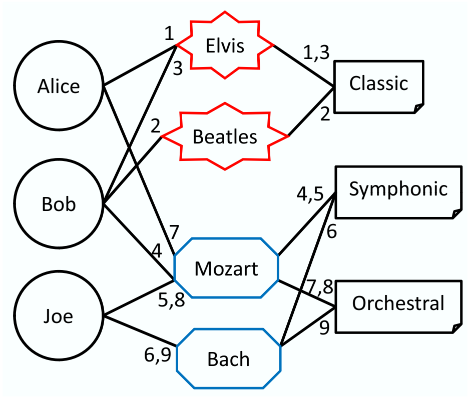


Figure 2. Illustration of the tripartite graph.

4. DATA MODELLING

In this section, we describe the modelling of multiple similarity values with a k-nearest-neighbor multidigraph. A multidigraph is a directed graph permitted to have multiple directed edges (henceforth, simply called edges), i.e., edges with the same source and target nodes.

A tripartite graph (like in the example of Figure 2b) can be partitioned according to the tags. For each tag t, we get the corresponding underlying subgraph B_t, by keeping users and items that participate in triples with this tag.

Each bipartite subgraph is represented with its adjacency matrix B_t ($1 \leq t \leq |T|$), whose size is $|I| \times |U|$; that is, its rows correspond to items and its columns to users. (Henceforth, wherever there is no ambiguity, we use interchangeably the same symbol for a graph and its adjacency matrix.) Each element $B_t(i, u)$ is equal to 1, if there is an edge between the item i and user u, or 0 otherwise. Therefore, from each adjacency matrix B_t we can compute for every pair of items i, j ($1 \leq i, j \leq |I|$), a similarity measure according to the values in their corresponding rows $B_t(i, :)$ and $B_t(j, :)$. Following the widely used approach for 2 dimensional matrixes (like document-term in information retrieval or user-item in CF), we consider the cosine similarity measure between every pair of items.

Having defined a similarity measure, from each subgraph B_t ($1 \leq t \leq |T|$), we can compute the corresponding k-nearest neighbor (k-NN) graph, N_t, which is a labelled and directed graph (digraph). The nodes of each N_t correspond to the items. There is an directed edge between items i and j ($1 \leq i, j \leq |I|$), if j is among the k nearest neighbors of i. Each edge is labelled with the corresponding similarity value.

Considering all k-NN digraphs together, we form the k-NN labelled multidigraph, $\mathcal{N}$, that summarises all multiple similarities. The nodes of $\mathcal{N}$ correspond to the items. The labelled edges of $\mathcal{N}$ is a multiset resulting from the union of the labelled edges of all N_t for $1 \leq t \leq |T|$.

Example 2 (k-NN multidigraph). For the data in Figure 2, the resulting k-NN multidigraph $\mathcal{N}$, for $k = 1$, is depicted in Figure 3a. The multiple edges between the nodes of $\mathcal{N}$ denote the different similarities between the items, according to the different tags. To assist notation, we assume that T_1 denotes the first tag, i.e., Classic, T_2 the second, i.e., Symphonic, and T_3 the third, i.e., Orchestral. In Figure 3a, the edges representing similarities according to tag T_i ($1 \leq i \leq 3$) are annotated with T_i and then follows the corresponding similarity value.[1] Notice that $\mathcal{N}$ correctly captures the clustering structure: edges exist only between items of the same cluster, i.e., between Elvis and Beatles for the first cluster and between Mozart and Bach for the second. Conversely, in Figure 3b, which depicts the k-NN digraph when only user-item relationships are considered, the separation of clusters is not clear. □

5. THE PROPOSED CLUSTERING ALGORITHM

5.1 Constructing the Laplacian Tensor

For each k-NN digraph N_t ($1 \leq t \leq |T|$) of $\mathcal{N}$, compute D_t as a diagonal matrix the diagonal elements of which are defined as follows:

[1] In this small example, to avoid numerical problems, we assign similarity equal to 0 when at least one item has no edge at all in the corresponding bipartite graphs. Moreover, to avoid cluttering the graph, only the non-zero similarities are depicted.

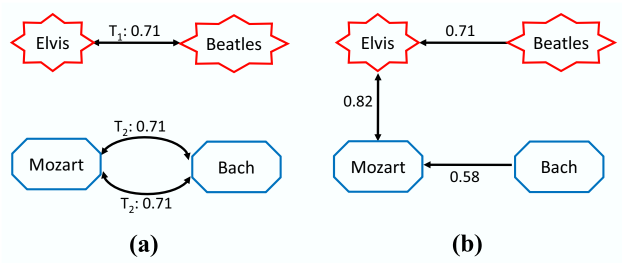


Figure 3. The k-NN multidigraph for the running example.

$$D_t(i, i) = \sum_{j=1}^{|I|} N_t(i, j) \tag{1}$$

The Laplacian matrix, L_t, of each N_t is computed as follows [10]:

$$L_t = \mathbb{I} - D_t^{-1/2} N_t D_t^{-1/2} \tag{2}$$

where $\mathbb{I}$ is the identity matrix.

The Laplacian tensor of $\mathcal{N}$ is therefore defined as $\mathcal{L} \in \mathbb{R}^{|I| \times |I| \times |T|}$, whose elements are given as follows:

$$\mathcal{L}(i, j, t) = L_t(i, j) \tag{3}$$

Thus, each matrix L_t, for $1 \leq t \leq |T|$, comprises a frontal slice in $\mathcal{L}$.

The Laplacian tensor $\mathcal{L}$ has 3 *modes*: the first mode corresponds to the items, the second mode to the neighboring items, and the third mode to the tags. To perform spectral clustering, we are interested in extracting the spectrum of $\mathcal{L}$ for the first mode. This procedure is explained in the section to follow.

5.2 Factorising the Laplacian Tensor

In this subsection, we summarise the factorisation of the Laplacian tensor using Tucker decomposition [11], which is the high-order analogue of the Singular Value Decomposition (SVD) for tensors. The factorisation of the Laplacian tensor will produce the required spectrum of its first (corresponding to items) mode.

First, we define the n-mode product $\mathcal{T} \times_n M$ between a general N-order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ and a matrix $M \in \mathbb{R}^{J_n \times I_n}$. The result is an $(I_1 \times I_2 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots \times I_N)$-tensor, whose entries are defined as follows (elements are denoted through their subscript indexes):

$$(\mathcal{T} \times_n M)_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} \mathcal{T}_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} M_{j_n i_n} \tag{4}$$

Since $\mathcal{L}$ is a 3-order tensor, we henceforth focus only on 1-mode, 2-mode and 3-mode products.

The Tucker decomposition of the 3-order tensor $\mathcal{L}$ can be written as follows [12]:

$$\mathcal{L} \approx \mathcal{C} \times_1 P_1 \times_2 P_2 \times_3 P_3 \tag{5}$$

The $P_1 \in \mathbb{R}^{|I|\times|I|}, P_2 \in \mathbb{R}^{|I|\times|I|}, P_3 \in \mathbb{R}^{|T|\times|T|}$ are called the mode-1 (items), mode-2 (neighboring items), and mode-3 (tags) projection matrixes, respectively. The 3 projection matrixes contain the orthonormal vectors for each mode, called the mode-1, mode-2 and mode-3 singular vectors, respectively. $\mathcal{C}$ is called the core tensor and has the property of all orthogonality. Nevertheless, unlike SVD for matrixes, $\mathcal{C}$ is not diagonal. Recently, several algorithms have bee proposed to efficiently compute the components of the Tucker decomposition. Due to lack of space, more details about the algorithms and their complexity can be found in a recent survey on tensor factorisation [11].

Having already performed the Tucker decomposition of the Laplacian tensor $\mathcal{L}$, we are interested in the mode-1 singular vectors that are stored in P_1. A frequently followed approach in spectral clustering, when c clusters are required, is to select the c eigenvectors associated to the c smallest eigenvalues [5]. Similarly, we select the c mode-1 singular vectors in P_1 associated to the smallest singular values in the core tensor $\mathcal{C}$.

5.3 Performing the Final Clustering

To find c clusters of items using the c mode-1 singular vectors that were computed and selected during the factorisation of the Laplacian tensor, we apply the following steps: (1) Normalise the c selected mode-1 singular vectors to have norm equal to 1. (2) Form a matrix $X \in \mathbb{R}^{|I|\times k}$, whose columns are the normalised c selected mode-1 singular vectors. (3) Associate each item i to a point x_i whose coordinates are the contents of the i-th row of X. (4) Choose a distance metric for the $(x_i)_{i=1,\ldots,|I|}$ points. (5) Cluster the points $(x_i)_{i=1,\ldots,|I|}$ into c clusters using a conventional clustering algorithm, according to the chosen distance metric. (6) Assign each item to the cluster of its associated point.

Due to the properties of the Laplacian tensor, in practice, the points in X can be easily clustered (Step 5) using simple conventional algorithms, like the K-Means or the hierarchical agglomerative algorithms. In the sequel we consider hierarchical agglomerative algorithms for this purpose based on Euclidean distance (Step 4).

Therefore, the proposed approach can better detect the clustering as it fully exploits all users-items-tags relationships. This is verified with the experimental results in the following section.

6. PERFORMANCE EVALUATION

6.1 Experimental setting

In our experiments we tested the proposed method, denoted as Tag-aware Spectral Clustering (TSC). For comparison purposes we tested two baseline Spectral Clustering methods, denoted as SC(U) and SC(T), that apply spectral clustering on a 2-dimensional item-user and item-tag matrix, respectively. In the former matrix an element is set to 1 when the corresponding item has been tagged at least once by the corresponding user (otherwise set to 0), whereas in the second matrix, when the corresponding item has been assigned at least once the corresponding tag (otherwise set to 0). All methods have been implemented in Matlab using the same components. Tensor factorisation was computed using the Tensor toolbox [2].

We used a real data set crawled from Last.fm (June 2008) by using Last.fm web services. The music items correspond to song titles. There are 64,025 triplets in the form user–tag–song. These triplets correspond 732 users, 2,527 tags and 991 songs.

Social-tagging data present problems like tag polysemy and sparsity. To address them, we applied the widely used technique of Latent Semantic Indexing (LSI) and reduced the number of dimensions in the modes of users and tags, by maintaining a percentage of them. This reduction was performed by modelling the original triples as a 3-mode tensor and applying Tucker decomposition [11]. The item mode is left unchanged, whereas the number of maintained users and tags after this process is expressed as a percentage (default value 30%) of the original number of users and tags (for simplicity we use the same percentage for both). Both SC(U) and SC(T) also utilise this technique by maintaining the same percentage for users or tags.

To form the k-NN similarity graphs and multidigraphs, we used the cosine distance, which is commonly applied for 0-1 sparse data like in our case. We tested several values of k and found that all examined methods are not sensitive in this parameter (default value $k = 10$). For the fifth step of the spectral clustering algorithm, we examined the Unweighted Pair Group Method with Arithmetic mean (UPGMA) hierarchical agglomerative clustering algorithm over the Euclidean distance (in the spectral feature space). Following the approach of conventional spectral clustering algorithms [5], we considered the number of clusters as a user-defined parameter. The quality of the final clustering result is measured with the popular Silhouette coefficient (the higher the better) that expresses both the coherency within clusters and the separation between clusters. For an item that is mapped to a vector x in the spectral feature space and is assigned to cluster C, its silhouette coefficient $s(x)$ is calculated as follows: a_x is the mean distance of x from all other vectors in C, whereas b_x is the minimum mean distance from vectors in all other clusters except C. Then, $s(x) = (b_x - a_x)/\max(a_x, b_x)$. The overall silhouette coefficient is the mean of all $s(x)$ for each x. [3]

6.2 Experimental results

We experimentally compare TSC against SC(U) and SC(T). The mean Silhouette coefficients for varying number of clusters is depicted in Figure 4. Due to its ability to consider 3-way relationships, TSC clearly outperforms the two baseline methods, which suppress the 3-way relationships

[2] http://csmr.ca.sandia.gov/∼tgkolda/TensorToolbox/

[3] For all compared method the silhouette coefficients are computed based on the Euclidean distance in the resulting feature space.

into 2-way, thus loosing information that is valuable for the clustering.

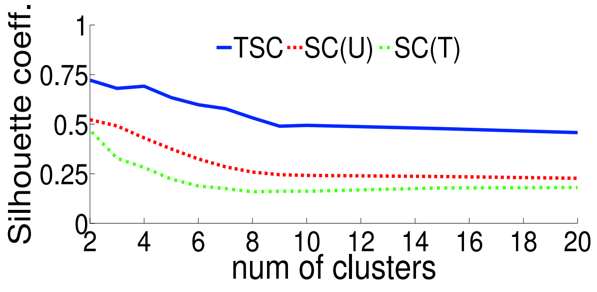


Figure 4. Results for varying number of clusters.

We also tested the sensitivity of the result against the percentage of maintained users/tags after the application of LSI (described in Section 6.1). Figure 5 depicts the resulting Silhouette coefficients for varying values of this percentage (the number of clusters is set to 10). When the percentage of maintained users/tags is severely low, the quality of TSC is reduced, as the resulting information is not adequate to capture the clustering structure. When the percentage is high, quality is again reduced, as the problems in the original data (polysemy, sparsity, noise) cannot be addressed. Therefore, in accordance to most applications of LSI, the best performance is attained with percentages that are in between the two extremes. In all cases, TSC compares favorably against TS(U) and TS(T).

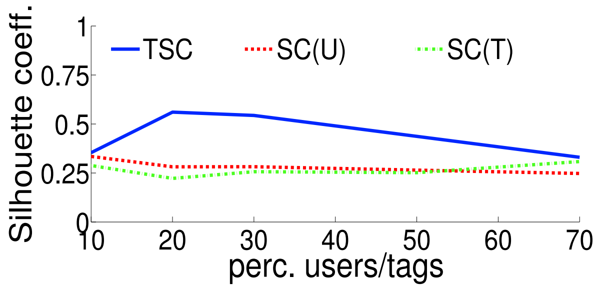


Figure 5. Results for varying perc. maintained users/tags.

7. CONCLUSIONS

We proposed a novel, tag-aware clustering algorithm for music data from social tagging systems. The advantage of the proposed algorithm over conventional clustering algorithms is that it preserves all 3 dimensions in the data and the 3-way relationships among them. The 3-way relationships of the form items–users–tags between these dimensions offers a clear advantage between just knowing that a tag has been applied to an item regardless by which users, and knowing the specific users that applied this tag to the item. To attain its advantages, the proposed algorithm uses tensors to store the underlying data model represented with multigraph structures, and extracts spectral features from them using tensor factorisation. Experimental results with real data showed that the proposed method yields clustering with better quality compared to conventional spectral clustering methods that suppress the dimensions and consider only 2-way relationships.

8. REFERENCES

[1] D. Byrd. Organization and searching of musical information, course syllabus, 2008.

[2] A. Morgan and M. Naaman. Why we tag: motivations for annotation in mobile and online media. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 971–980, 2007.

[3] Last.fm. Listen to free music with internet radio and the largest music catalogue online, 2009.

[4] MyStrands. Social recommendation and discovery, 2009.

[5] U. von Luxburg. A tutorial on spectral clustering. Technical report, (No. TR-149) Max Planck Institute for Biological Cybernetics, 2006.

[6] P. Lehwark, S. Risi, and A. Ultsch. *Data Analysis, Machine Learning and Applications*, chapter Visualization and Clustering of Tagged Music Data, pages 673–680. Springer Berlin Heidelberg, 2008.

[7] M. Levy and M. Sandler. Learning latent semantic models for music from social tags. *Journal of New Music Research*, 37(2):137–150, 2008.

[8] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: A survey. *IEEE Signal Processing Magazine*, 23(2):133–141, 2006.

[9] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos. Ternary semantic analysis of social tags for personalized music recommendation. In *Proc. International Conference on Music Information Retrieval (ISMIR08)*, 2008.

[10] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS'01)*, pages 849–856, 2001.

[11] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3), September 2009 (to appear).

[12] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

MULTIPLE F0 ESTIMATION IN THE TRANSFORM DOMAIN

Christopher A. Santoro[+*] **Corey I. Cheng**[*#]

[+]LSB Audio
Tampa, FL 33610
chris@lsbaudio.com

[*]University of Miami
Music Engineering Technology
Frost School of Music
Coral Gables, FL 33124

[#]University of Miami
Department of Electrical and Computer Engineering
coreyc@miami.edu

ABSTRACT

A novel algorithm is proposed to estimate the fundamental frequencies present in polyphonic acoustic mixtures expressed in a transform domain. As an example, the algorithm operates on Modified Discrete Cosine Transform (MDCT) coefficients in order to demonstrate the utility of the method in commercially available perceptual audio codecs which use the MDCT. An auditory model is developed along with several optimizations that deal with the constraints of processing in the transform-domain, including an interpolation method, a transform-domain half-wave rectification model, tonal component estimation, and sparse convolution. Test results are separated by instrument and analyzed in detail. The proposed algorithm is shown to perform comparably to state of the art time-domain methods.

1. INTRODUCTION

Perceptually coded formats such as mp3 and AAC have become the dominant storage and distribution format for commercial digital music. These formats are popular because they greatly reduce bandwidth and memory requirements related to transmission and storage. As a result of the successes of these formats, portable media players are becoming increasingly important platforms for the analysis and synthesis of digital media. These devices have limited processing power and battery life, and therefore require analysis and synthesis algorithms with minimal computational complexity where possible.

One emerging family of algorithms that is finding increased applicability in music information processing is multiple fundamental (F0) estimation. Loosely speaking, the purpose of multiple F0 estimation is to estimate the perceived pitches of multiple harmonic series, such as those created by the human voice or various musical instruments, when sounding concurrently. Multiple F0 estimation finds widespread use as a front-end to various pitch tracking and source separation algorithms.

State of the art analysis algorithms are typically designed to begin their operations on uncompressed PCM audio signals in the time-domain. Because music files on portable devices are stored in a perceptually coded format, they must first be decoded before the algorithms can begin their analysis. For example, many perceptual audio codecs spend considerable resources in using the well-known Inverse Modified Discrete Cosine Transform (IMDCT) to synthesize a time-domain signal during the decoding process. Therefore, it would be especially advantageous to avoid this expensive decoding process where possible, and operate directly on the native MDCT representation used in a perceptually coded file.

This work adapts a state of the art multiple F0 estimation algorithm to operate directly on a transform-domain representation used in modern audio codecs as a starting point for this research. Very few authors have researched transform-domain processing. Previous works include beat detection [1], music/speech classification [2], and sinusoidal analysis [3]. A primary reason for the limited body of work related to F0 estimation algorithms is the limited frequency resolution used in transform-based audio coders. When working in the transform domain, we are stuck with whatever frame size the coder uses. A secondary difficulty with transform domain processing is that some time-domain processes do not easily lend themselves to operation in the transform-domain. In this work several novel modifications to the auditory model are proposed that successfully mitigate both of these limitations. A third difficulty with transform domain processing is that processing in some transform domains, such as the MDCT domain, can be problematic because of some of the aliasing properties of the transforms [10].

We propose an algorithm for multiple F0 estimation in the transform domain that adapts the work of Klapuri [4] to function in the MDCT domain. Like [4], the proposed algorithm uses a model of the human auditory system along with iterative estimation and cancellation to estimate component F0s, as the auditory model described by Klapuri consists of an auditory filterbank followed by half-wave rectification and low pass filtering. However, in this work, each of these processes is adapted to operate in the transform-domain. We also incorporate modifications to the iterative estimation portion of the algorithm by Paz [6] in order to improve performance.

2. OVERVIEW OF REFERENCE ALGORITHM

The design of the proposed algorithm is based on the work of Klapuri [4]. The basic framework of Klapuri's

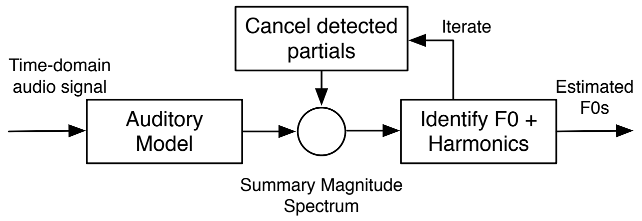


Figure 1. Overview of reference multiple F0 estimation method

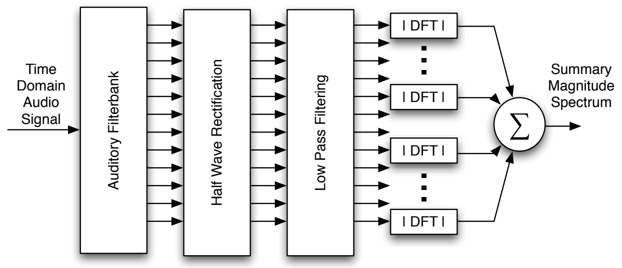


Figure 2. Detail of auditory model used in reference method

method is shown in Figure 1. A key feature of the algorithm is the auditory model, shown in Figure 2. The auditory model used by Klapuri consists of a 70 channel gammatone filterbank design, using the computationally efficient implementation by Slaney [11], followed by half-wave rectification and low-pass filtering. Finally, a Discrete Fourier Transform (DFT) is taken in each channel, and the spectra in each channel are summed to create a summary magnitude spectrum.

To identify F0s, the *salience* of each candidate F0 is calculated using (1), where U_{SMS} is the summary magnitude spectrum, $K_{\tau,m}$ is a region where each partial is expected to be based on the period (τ) of the F0 and the harmonic number (m), and $\omega(\tau,m)$ is an exponentially decreasing weighting function dependent on F0 and harmonic number.

$$s(\tau) = \sum_{m=1}^{M} \omega(\tau,m) \max_{k \in K_{\tau,m}} U_{sms}(k) \qquad (1)$$

The salience can be interpreted as a measure of the perceptual strength of each candidate F0. On each iteration, the F0 with the highest salience is chosen. Its partials are then identified and partially subtracted from the mixture according to an exponentially decreasing weighting scheme. This process is repeated until a known number of F0s have been estimated. In Klapuri's work the polyphony considered to be known *a priori* in most cases.

3. PROPOSED ALGORITHM

Our proposed algorithm comprises three main stages: interpolation, a low complexity transform-domain auditory model plus iterative estimation and subtraction. The algorithm makes changes mainly to the auditory model and adds transform domain interpolation to the front end of the salience calculation in an attempt to deal with the limited frequency resolution of the transform-domain representation used in audio codecs.

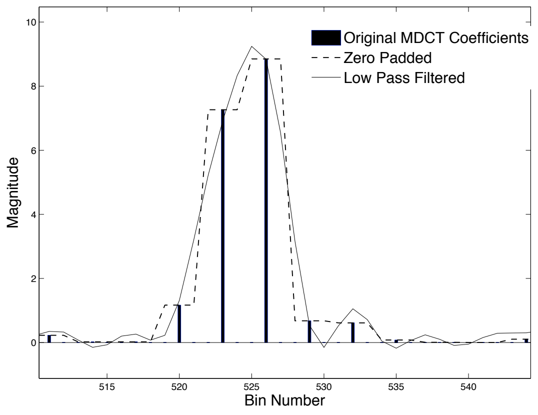


Figure 3. Example of interpolation process. In this figure the peak has been shifted to the left after interpolation based on the distribution of energy around the old peak.

3.1. Interpolation of MDCT coefficients

As stated previously, one of the fundamental limitations of transform-domain processing is that we are stuck with whatever frame size the codec uses. Codecs like AAC use large frame sizes of 2048 samples for tonal content, and smaller frame sizes of 256 samples for transients [9]. A frame size of 2048 samples at 44.1 kHz yields a frequency resolution of roughly 21.5 Hz. Since F0s are more closely spaced at lower frequencies in contemporary western scales, this means that a peak in one MDCT bin can span as many as six notes. In fact, F0s are not spaced more than 21 Hz apart until the 4th octave (E4 or 330 Hz). If nothing is done to address this, this means that any peak corresponding to an F0 below 330 Hz could be one of many F0s. This assumes an equal tempered scale. The details of other tuning systems will be different.

To solve this problem, we use a simple interpolation method in the transform-domain. While this method does not increase the real frequency resolution of the transform-domain representation, it does have the effect of shifting peaks to a more accurate location corresponding to the true F0, making estimates of frequency when peak picking more accurate. The implementation of this method consists of zero padding/upsampling, then performing a zero order hold and low pass filtering. An upsampling factor of 3 was used here, but any odd factor may be used.

The low pass filtering was implemented as a simple FIR filter, which takes the form of a convolution with a sinc function. We found a filter length of 24 samples to be adequate for an upsampling factor of 3. An example of the result of the interpolation process is shown in Figure 3.

What is left as a result of the interpolation process can no longer be called a realistic MDCT spectrum, but this is of no consequence to the proposed algorithm. What we do have at this point is a reasonable estimation of the magnitude spectrum of our signal, and a better estimation of true peak locations due to the interpolation process. It is important to remark that this is a computationally expensive process, which could probably be replaced by a less intensive interpolation method. However, this method was chosen here for its simplicity and good results.

3.2. Transform-Domain Auditory Model

After interpolation, the spectrum is passed onto a transform-domain auditory model, where we implement a modified version of the Unitary model of human pitch perception, proposed by Meddis [7]. In this section, we describe our modifications and improvements to the model's four following steps:

1. The stimulus is passed through a filterbank of band-pass filters, which simulate the action of the basilar membrane.
2. Each sub-band signal is compressed, half wave rectified, and low-pass filtered to obtain the time domain amplitude envelope.
3. Periodicity estimation is carried out on each sub-band.
4. Periodicity information from each sub-band is combined across channels.

Step 1 of the unitary model is said to mimic the frequency selectivity of the inner ear. Typically a gammatone filterbank implementation by Slaney is used, although the number of channels necessary to achieve good results is debated. Depending on the application, previous works using auditory models use as few as 2 channels [8] and as many as 70 [4]. For this reason, we implemented filterbanks with 8, 16, 32, 64, and 70 channels to explore the effect on performance of the algorithm. If the number of channels can be reduced, then the computational complexity of the algorithm can be reduced significantly.

Step 2 of the unitary model processes information contained in the time domain amplitude envelope of the stimulus signal. Many musicians know the information we are looking for here as *beating*. Beating occurs when two sinusoids that have slightly different frequencies cancel and/or reinforce each other periodically. The fundamental period of the beating corresponds to the difference in frequency between the two sinusoids. Thus, this process (half wave rectification and low pass filtering) can be considered as a way of analyzing the intervals between harmonics, which corresponds to the F0 of a harmonic sound. This type of information is called *spectral interval information*.

Steps 3 and 4 are merely ways to extract the periodicity of the time-domain amplitude envelope, which is reinforced in step 2. Typically an autocorrelation function (time-domain) or a Discrete Fourier Transform (frequency-domain) is used in each channel. Given these processes, it is easy to see why we do not want a large number of channels if it is not necessary.

Some of these steps lend themselves easily to a transform domain implementation, while others prove more difficult. For example, step 1 of the auditory model is trivial in the transform domain. The auditory filterbank can be implemented easily by a matrix multiply if we store the magnitude response of each channel in an $N \times nC$ matrix, where N is the number of coefficients in our upsampled MDCT spectrum and nC is the number of channels in our filterbank. The magnitude response of each channel of the auditory filterbank can be obtained easily by first obtaining a standard time-domain design, and processing each channel with an impulse. Taking the magnitude DFT of the result yields the magnitude response of each channel. Since the filterbank is static, it can be calculated once, and the magnitude response can simply be stored in memory.

Step 2 of the auditory model is the most difficult to adapt for the transform domain. It is not obvious how half-wave rectification can be translated into the transform domain. However, Klapuri [5] points out that half wave rectification can be modeled as a convolution operation. The mathematical details of that argument are beyond the scope of this paper, but the interested reader is encouraged to check the source for an in depth analysis. Instead, we simply note that since the goal of this step of the model is to reinforce spectral interval information, that a convolution operation is an intuitive method for extracting that information.

There are two difficulties with the convolution of spectra to extract spectral interval information. One is that spectra have a DC offset due to the fact that all magnitude coefficients are positive. This causes a triangular shaped buildup around DC that obscures peaks indicating prominent spectral intervals. The other is that the process is prohibitively expensive computationally, especially since this is a process that must be performed on each channel individually. A standard way to attack the first problem is to subtract the mean from the signal. Not only does this not work in this case because the DC offset is caused by a small region of disproportionately large peaks, but it also does not address the second problem. We propose here a method for solving both of these problems based on tonal component estimation.

Since we are looking for intervals between peaks, we begin the process by finding the locations of the peaks in the subband spectrum. This is called tonal component estimation (TCE). First, the derivative of each subband is taken. Next, the derivative is used to analyze the slope of each peak. Using a sliding window of 15 coefficients, local maxima are found by identifying locations in which the derivative transitions from a positive value to a negative value, and the difference between the two is greater than some threshold. The mean of each subband signal was found to be a good threshold, though this value may be changed to adjust sensitivity. Once we have identified the locations of tonal components, we replace each peak

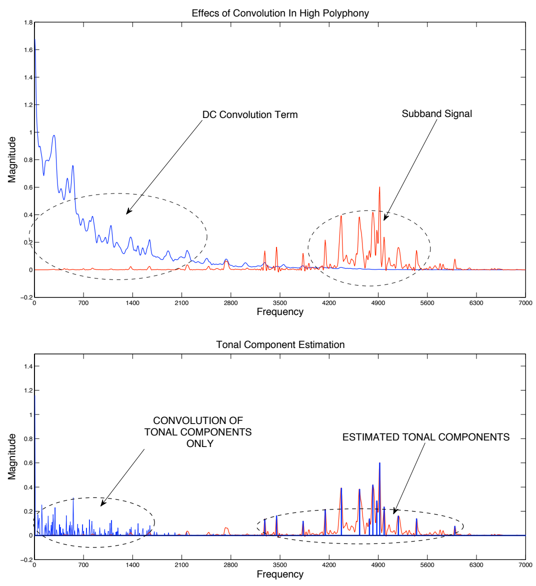


Figure 4. Comparison of TCE and sparse convolution to traditional convolution of MDCT spectra.

with a single spike that has the same amplitude as the peak.

Using this process, each subband signal is reduced to only the most pertinent information (i.e., the locations and magnitudes of harmonics). This transforms the signal into a sparse vector, since most of the elements in each subband signal are now zeros. Next, a sparse convolution may be used to extract spectral interval information. This process is shown in Figure 4 and compared to a standard convolution. Not only does it solve the problem of DC buildup, but it also reduces the computational complexity drastically.

The result of this process is a vector of extracted spectral interval information, V_c. By combining this with the original subband spectrum, we can obtain both spectral interval and spectral location information. Therefore, we calculate a weighted combination of the original spectrum (X_c), using (2), where α is a simple parameter which can be used to adjust the importance of spectral interval information. Y_c is the resulting signal in each channel after the half wave rectification process.

$$Y_c = (1\text{-}\alpha)\, X_c + \alpha\, V_c \tag{2}$$

Step 3 of the auditory model is performed by a DFT in the reference method. Here, we are already in the frequency domain, so this step can be skipped, yielding a large computational savings. Step 4 is also trivial and is computed by summing across channels to create a summary magnitude spectrum, U_{SMS}. This is shown in (3). In this step, channels that have peaks in the same location in their V_c components (meaning their spectral interval information is in agreement) reinforce each other to accentuate (or in some cases, reproduce) the peaks corresponding to the F0s in the mixture.

$$U_{sms}(k) = \sum_{c=1}^{nC} Y_c(k) \quad \textit{for all } k \tag{3}$$

3.3. Iterative Estimation and Subtraction

Once the summary magnitude spectrum has been calculated, the algorithm performs an iterative estimation and subtraction process largely similar to that in the reference algorithm. On each iteration, the salience is calculated for all fundamental candidate periods τ as described in (1). The candidate period with the maximum salience is chosen to be an F0. Next, we attempt to identify the peaks that contributed to the salience of the currently estimated F0. An adaptive scheme is used to capture peaks as well as their side lobes, which was developed by Paz [6] and was found to improve performance significantly. First local maxima are identified within each region defined by $K_{\tau,m}$. Next, the boundaries are expanded until they lie at adjacent local minima. Once this is completed, a detected spectrum U_{D} is formed consisting of the partials of the estimated fundamental. These partials are then weighted by the same weighting function that was used to calculate the salience. This allows us to remove some of the energy in each partial, but not all of it. This is critical for cases in which multiple sounds have partials that overlap.

Finally, a residual spectrum U_{R} is formed by subtracting U_{D} from U_{SMS}. The process of calculating the salience and estimating the partials of F0s is repeated on U_{R} a number of times that is equal to the polyphony, which is known *a priori*. The estimated F0s are then quantized to the nearest frequency value corresponding to a valid note on the equal tempered scale, with A4 corresponding to 440Hz.

4. RESULTS

The proposed algorithm was tested in a similar manner to the reference algorithm. Polyphonic mixtures of 2, 4, and 6 notes were created from four different types of instruments: Saxophone, Flute, Violin, and Cello. Sample recordings of individual notes were used from the University of Iowa[1] database. For each polyphony and instrument, 100 mixtures were created by lining up the onsets of notes and mixing them at equal RMS levels. Each individual file was then encoded using the LAME mp3 encoder[2] at 128 kbps. The results presented in all tests for the reference algorithm are based on a careful implementation based on the information given in the papers published by the author.

4.1. Decoder Model

To modify an actual decoder to return just MDCT coefficients (prior to taking the IMDCT and performing overlap and add) would have taken considerable time and effort. Instead, we constructed a simplified decoder

[1] http://theremin.music.uiowa.edu/MIS.html
[2] http://lame.sourceforge.net/

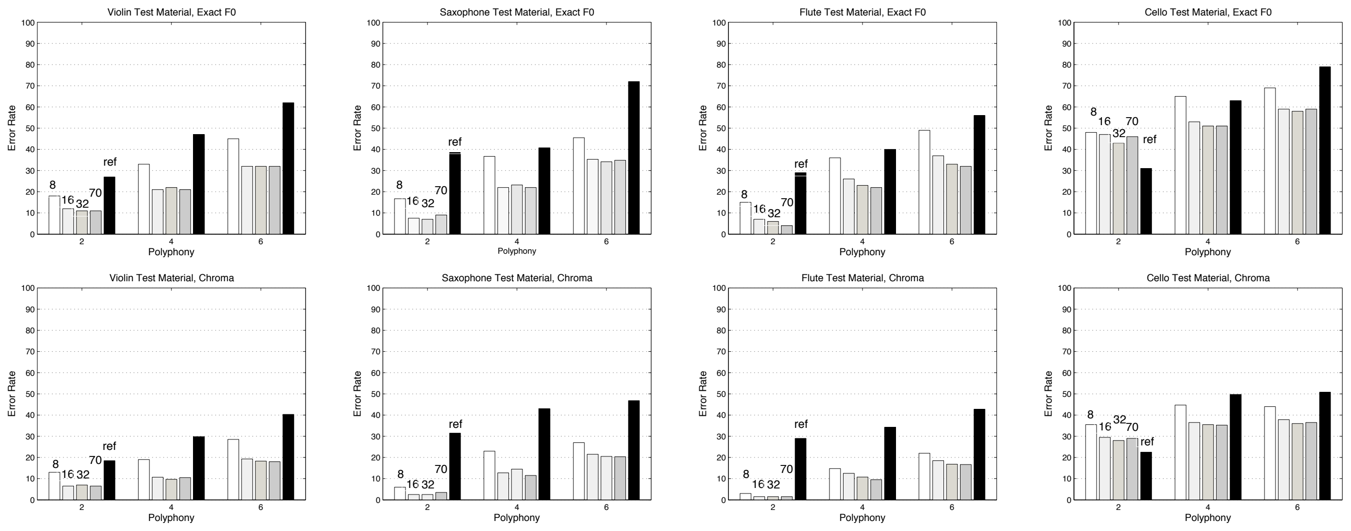


Figure 5. Multiple F0 estimation error rates for several musical instruments and several polyphonies. The reference method is in black. The proposed algorithm is tested for a Unitary model of hearing having 8, 16, 32, and 70 frequency bands.

model that fully decodes the mp3 file and then reverses the last two steps by windowing and then taking the MDCT.

While a real partial decoder would be best, we consider this decoder model to be a sufficient first attempt at multiple F0 estimation in the transform-domain. To implement the MDCT, we used a "fast MDCT" which utilizes an FFT with two rotations to perform an MDCT. In order to have a consistent basis for comparison with the reference algorithm, we used a frame size of 46ms, which corresponds to 2048 time-domain samples. This is also the largest frame size used in AAC [9].

4.2. F0 Estimation Results

The results of the F0 estimation tests for each instrument and filterbank design are shown in the top of Figure 5.

Error rate is calculated in the same manner as in the reference. The most important result of the F0 estimation results is that performance is roughly equal for filterbank designs with as few as 16 channels. The algorithm performed significantly worse when using less than 16 channels. Interestingly, a 16 channel filterbank design of the range of 60 Hz to 2.2 kHz roughly corresponds to a 1/3 octave filterbank design (which would have 19 channels in this case). This is a common psychoacoustically motivated design for equalizers in stereo systems. This result is important, as it demonstrates that we can drastically reduce the complexity of our filterbank while paying a minimal performance penalty.

Additionally, the results show that the algorithm performs well, outperforming the reference algorithm in most cases. The error rates published here are slightly higher than previously published for the reference using a 46ms window. This could be due to implementation inaccuracies or a discrepancy in test material.

4.3. Chroma Estimation Results

In some applications, the exact octave that a note is from may not be as important as the chroma of the note. That is, in a mixture that contains the notes C3, E4, and G3, an estimation of C, E, and G may be sufficient. To investigate the proposed algorithm's perfomance in chroma estimation, the F0 estimation tests were re-run, but this time octave errors were not counted as errors. The results in the bottom half of Figure 5 show that the error rates dropped drastically for all instruments except for Cello. Error rates for each filterbank design with more than 8 channels were less than 5% for low polyphonies. This shows that a majority of the errors from the F0 estimation tests were octave errors.

4.4. Discussion

While the proposed algorithm's performance was impressive on each task, the question still remains as to why the Cello performed so poorly, while the other instruments performed well. One would think that the inharmonicity of stringed instruments as well as the low frequency range of the cello played a part. An analysis of the distribution of samples for each instrument was conducted and this revealed that indeed the cello had a distribution that occupied a significantly lower range than the other instruments. This is likely to have played a larger role than inharmonicity, since the algorithm had no problem dealing with the violin samples.

This reveals a primary weakness of the proposed algorithm. It does not seem to deal well with lower F0s. This is most likely due to inadequate frequency resolution for F0s below the 3rd octave. A higher upsampling rate in the interpolation stage may mitigate this somewhat, but this would increase computational complexity.

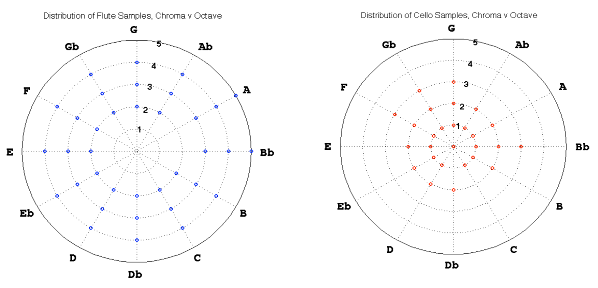


Figure 6. Polar comparison of flute (left) and cello (right) sample distributions, by chroma (angle) vs. octave (radius).

5. CONCLUSIONS AND FUTURE WORK

In conclusion, we have shown here that it may be possible to perform multiple F0 estimation entirely in the transform-domain. We have adapted a state of the art algorithm to work in the transform-domain, which includes a model of human pitch perception. We have shown that upsampling and interpolation of MDCT coefficients is a viable strategy for mitigating the inadequate frequency resolution of frame sizes native to perceptual audio coders. However, we have also found that F0s in the lower octaves still remain a problem due to limited frequency resolution.

For the purposes of comparing this algorithm against other multiple F0 estimation algorithms, it would be useful to use a MIREX database [12] for future test material. This would provide more reliable grounds on which to make comparisons. The test material used here was intended to be as close as possible to that used in the reference method. Furthermore, while a large effort was made to accurately implement the reference algorithm, mistakes will always be made because limited publication space inevitably causes some details to be left out.

Future work should also include a more detailed decoder model, as well as further experimentation with different upsampling factors and filterbank designs. The MDCT spectra used for this investigation, while fine for a starting point on this research, are certainly not an exact representation of what we would see coming from an actual decoder. Strategies will need to be developed to deal with the limitations of more realistic representations of MDCT coefficients in perceptually coded files. While this work does not address these tedious details, it lays the groundwork for an evolution in that direction.

6. REFERENCES

[1] E. Ravelli, G. Richard, and L. Daudet: "Fast MIR in a sparse transform domain," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 527-532, 2008.

[2] G. Tzanetakis and P. Cook: ”Sound analysis using MPEG compressed audio,” *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 761-764, 2000.

[3] S. Merdjani and L. Daudet: “Direct estimation of frequency from MDCT-encoded files,” *Proceedings of the 6th International Conference on Digital Audio Effects,* 2003.

[4] A. Klapuri: “A perceptually motivated multiple-F0 estimation method,” *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005.

[5] A. Klapuri: “Signal processing methods for the automatic transcription of music,” *PhD Thesis*, Tampere University of Technology, 2004.

[6] L. Paz: “Multiple-F0 estimation using auditory models,” *M. Sci. Thesis,* University of Miami, 2008.

[7] R. Meddis: “Virtual pitch and phase sensitivity of a computer model of the auditory periphery. 1. Pitch idenficitaion,” *Journal of the Acoustical Society of America*, vol. 86, no. 6, pp. 2886-2882, 1991.

[8] T. Tolonen and M. Karjalainen: “A computationally efficient multipitch analysis model,” *IEEE Trans. On Speech and Audio Processing,* vol. 8, no. 6, 2000.

[9] M. Bosi, et al.: “ISO/IEC Advanced Audio Coding,” *Journal of the Audio Engineering Society*, vol. 45, no. 10, pp. 789-811, 1997.

[10] C. Cheng: “Method for Estimating Magnitude and Phase in the MDCT Domain,” *Audio Engineering Society (AES) 116th Convention*, Berlin, Germany, 2004.

[11] M. Slaney: “An Efficient Implementation of the Patterson Holdsworth Auditory Filter Bank,” Perception Group, Apple Computer, Tech. Rep. 35, 1993.

[12] J. Downie: “The Music Information Retrieval Evaluation Exchange: A window into music information retrieval research,” *Acoustical Science and Technology*, vol. 29, no. 4, pp. 247-255, 2008.

MOTIVE IDENTIFICATION IN 22 FOLKSONG CORPORA USING DYNAMIC TIME WARPING AND SELF ORGANIZING MAPS

Zoltán Juhász
Research Institute for Technical Physics
and Materials Science.
P.O.B 49. Budapest H-1525.

juhasz@mfa.kfki.

ABSTRACT

A system for automatic motive identification of large folksong corpora is described in this article. The method is based on a dynamic time warping algorithm determining inherent repeating elements of the melodies and a self-organizing map that learns the most typical motive contours. Using this system, the typical motive collections of 22 cultures in Eurasia have been determined, and another great common self organising map has been trained by the unified collection of the national/areal motive collections. The analysis of the overlaps of the national-areal excitations on the common map allowed us to draw a graph of connections, which shows two main distinct groups, according to the geographical distribution.

1. INTRODUCTION

In order to study interethnic and historical relations, Bartók and Kodály compared different layers of Hungarian folk music to those of other nations living in the neighborhood of Hungarians. Later, they extended the study to Anatolian, Mari and Chuvash folk music [1-2]. These exciting results raise the question, whether it is possible to describe a whole and clear system of musical contacts in Eurasia by a systematic comparison of a sufficient number of national or regional cultures.

A further question, raised by the classical results mentioned above, refers just to the method of the analysis. The aim of these classical works was to find parallelism of entire melody structures. The similarity of whole melody contours seems to be really a sufficient condition to find genetic musical relations [1-3]. However, the question rises: do less rigorous requirements also exist? Instead of comparing the complete melody structures, our aim was to find and analyse the smallest independent melodic units. It is well known that folksongs can usually be divided into certain phrases on the basis of musical and textual regularities. In a previous work, we have shown some results comparing individual phrases, as well as whole melodies of 6 European cultures [4].

The idea of a motive identification algorithm can be derived from the recognition that phrases are not necessarily the smallest intelligible units in folk music. We want to find the most frequently appearing motive types in a well defined melody corpus, with the assumption that each motive type may have several variants. However, the repetition inside a melody can also be considered as an indication of a motive. Therefore, we suppose two possible detections of the motives. In addition to the "culture-defined" motive identification, based on the frequent appearance in different songs, we also suppose the existence of the "melody-defined" identification which is based on the repetitions inside the melodies.

The central problem of algorithmic melody pattern identification is the musical relevance of the results [5]. The most frequently applied melody segmentation techniques can be divided into two main groups. In the first group, segmenting is based on pre-defined and data-independent rules [6-8]. Using such rules, the so-called Local Boundary Detection Model (LBDM) determines a boundary strength value between each couple of notes, and determines the segment boundaries at the maximal strength values [6,9]. Due to the requirement of pre-defined rules, such methods are not available for the sake of a learning system. The second group of segmenting techniques is based on a learning process to determine the regularities of a given melody corpus. Such regularities can be characterised by the frequencies or conditional probabilities of the motives [10-12]. The so called Markov technique operating with conditional probabilities has already been applied to folk songs [13-14]. A further data-based self learning method for segmenting a large corpus of folksongs has been also described, which determines the conditional entropy of the motives and defines an average entropy increment value for a given segmentation [15]. A method based on knowledge representation has been elaborated for identifying recurrent melody parts in large folksong corpora [16].

The learning unit of the system described in this paper is a self organising map (SOM), trained by the contour functions of the motives [17-18]. The motive identification in a given melody is accomplished in two steps. Firstly we determine the repeating elements of the melody by an algorithm based on dynamic time warping (DTW). After that, the remaining melody parts are analysed using a self organizing map, which learns and identifies the most frequently appearing patterns as "culture-defined" motives.

Our current possibilities allowed us to set up 22 folksong corpora, each of them consisting of 600-2400 melodies, representing Hungarian, Slovak, Moravian, Chinese, Mongolian, Kyrgyz, Mari-Chuvash-Tatar, Karachay-Balkar, Anatolian Turkish, Azeri, Sicilian, Spanish, Rumanian, Bulgarian, Polish - Cassubian, Finnish, Norwegian, German, Luxembourgian-Lotharingian, French, Dutch and Irish-Scottish-English musical traditions. In order to make an unbiased and general analysis, these nearly 40 000 melodies were transposed to the common final tone G automatically in the analysis.

2. DETERMINATION OF MOTIVES DEFINED BY REPETITION WITHIN MELODIES

To search for essentially identical, but not completely uniform motives inside melodies, we developed an algorithm based on dynamic time warping technique [17]. The operation of the algorithm is illustrated in Figures 1 and 2.

In the first step, the contour vectors $\underline{v}$ of the melodies are generated in the way demonstrated in Figure 1. The time duration of the *k*th melody is divided into small units according to the rhythmic value of 1/16, and the pitch values belonging to these subsequent small time intervals are stored in a multidimensional vector $\underline{v}_k$.

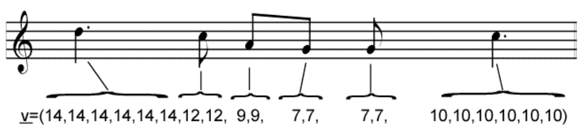

v=(14,14,14,14,14,14,12,12, 9,9, 7,7, 7,7, 10,10,10,10,10,10)

Figure 1. Generation of the contour vector $\underline{v}$.

The original aim of a DTW process is to determine a non-negative scalar number characterising the difference of two vectors. In order to calculate this DTW-distance between melody contours $\underline{v}$ and $\underline{u}$, the matrix $\underline{\mathbf{P}}$ is generated containing the deviations of the *n*th and *m*th pitch samples of the vectors $\underline{v}$ and $\underline{u}$:

$$\delta_{n,m} = \left|v_n - u_m\right| \quad n = 1...D_v, m = 1...D_u, \tag{1}$$

where D_v and D_u are the dimensions of $\underline{v}$ and $\underline{u}$ respectively. Figure 2 shows an example of the above calculation for the contour vectors demonstrated by the diagrams on the left side and the bottom of the matrix.

The zero elements of the matrix $\underline{\mathbf{P}}$ marked by bold italic characters indicate local warping curves assigning similar parts of the two vectors to each other. Our algorithm is based right on this recognition: instead of determining the total DTW distance of the vectors, we search for such local warping paths in matrix $\underline{\mathbf{P}}$. To do this, the partial time warping distances $\Delta_{n,m}$ are calculated, according to the dynamic time warping process:

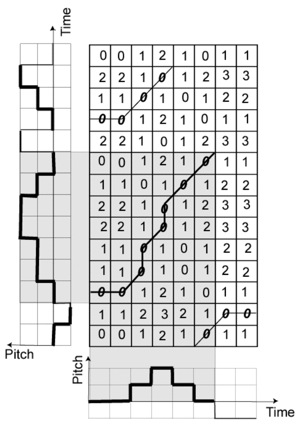


Figure 2. Generation of the partial deviation matrix $\underline{\mathbf{P}}$, and the path of 0 elements indicating the relation between corresponding motives.

$$\Delta_{n,m} = \min\begin{pmatrix}\Delta_{n-1,m} + \delta_{n,m}\\ \Delta_{n-1,m-1} + \delta_{n,m}\\ \Delta_{n,m-1} + \delta_{n,m}\end{pmatrix} \rightarrow \begin{pmatrix} v_{n,m} = 1\\ v_{n,m} = 2\\ v_{n,m} = 3\end{pmatrix} \qquad n = 2..D_v, m = 2...D_u$$
(2)

The original DTW algorithm produces the final distance at the end of the above recursive calculation as $\Delta_{D_v D_u}$. The local warping paths can be determined using the $D_v * D_u$ dimensional matrix $\underline{\mathbf{V}}$. Since the elements of the matrix $\underline{\Delta}$ cannot be defined for negative indices, the algorithm starts with the values of $(n = 2, m = 2, v_{1,1} = 2)$, and the initial values of $\underline{\Delta}$ are $\Delta_{1,m} = \delta_{1,m}$ and $\Delta_{n,1} = \delta_{n,1}$.

The overall similarity of the vectors can be characterised by the summed length of the similar sub-sequences compared to the sum of the total length of the vectors $D_v + D_u$. Thus, our technique can characterise the similarity of two different contour vectors by a scalar number ranging between 0 and 1. This similarity measure ignores the order of the motives, in contrast to the original DTW and the Euclidean distances. Therefore it is able to detect the relationship even if the successions of the characteristic melody parts are different in the compared melodies.

Example 1 shows two couples of melodies arising from different cultures, with a significant amount of similar parts found by the above described method. For instance, the first, second and fourth phrases of the Hungarian song in the first example are practically identical to the second and fourth phrases of the corresponding Appalachian melody, and the third phrase of the Appalachian song appears as a dominant part of the corresponding Hungarian phrase, too. Due to these local correspondences, the melodies are found to be similar, in spite of the difference between the domed, as well as descending character of the two melodies.

The above technique can be applied also to identical vectors (i.e. $\underline{v} = \underline{u}$). In such cases, the trivial result that the whole melody is identical to itself is indicated by the zero elements of the diagonal, but the partial warping paths marked by zero matrix elements indicate the similar subsequences. Therefore, our technique is also able to find similar parts within one given melody (see Figure 3).

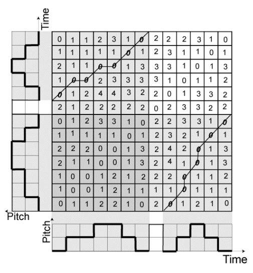
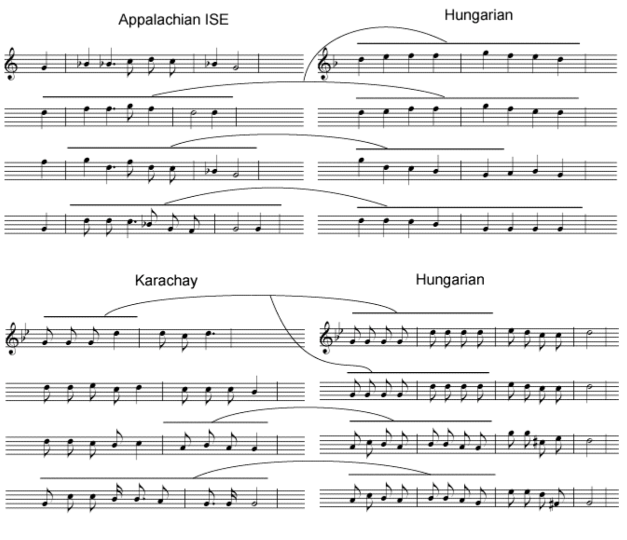


Figure 3. Application of the DTW technique to search for repeated parts in a melody.

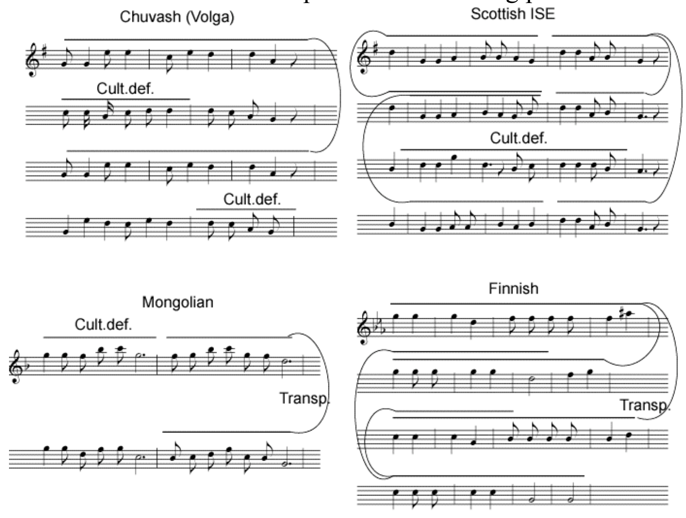
The technique can be generalized for not exactly identical pitch values, too, by the extension of the search for paths of small elements in matrix $\underline{\mathrm{P}}$. Some results of the method are shown in Example 2.



Example 1. Common motives of related melodies arising from different cultures.

3. THE COMPLEX MOTIVE IDENTIFICATION ALGORITHM

In addition to the melody-based motive identification, we also need a technique for the culture-defined identification which was defined as the determination of those melody parts which frequently appear in a whole national/areal database. While the melody-based technique needs the analysis of one given melody, the culture-based identification requires a self learning process



Example 2. Melody-defined and culture-defined motives in 4 folksongs.

analysing the whole database simultaneously. In order to solve this problem, i.e. to identify the most frequent melody parts automatically, we developed a system based on a self organising map, as it is shown in Figure 4.

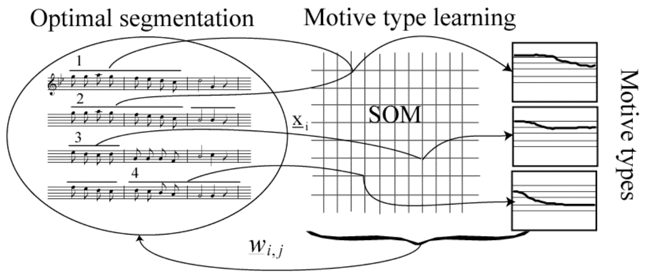


Figure 4. The complex motive identification system.

The input to the algorithm is a melody selected randomly from the database. At the beginning of the process, the D dimensional motive type contour vectors assigned to the lattice points of the SOM, $\underline{w}_{i,j}$ are filled by random numbers. The choice of $D = 32$ proved to be sufficient for our database.

The processing is done by the following steps:

1. In the first step, all melody-defined motives of a melody are determined, using the melody-based identification algorithm.

2. All possible motives of the remaining parts of the melody are determined. The time duration of each possible motive is divided into D parts, and the pitch values belonging to the subsequent time intervals are stored in a vector $\underline{x}$ of dimensionality D. This operation has been discussed in reference to melody contour generation (see Figure 1), but it is worth mentioning here an important

difference: When generating motive contour vectors, the vector dimension D is a pre-defined constant, while it is variable for melody vector generation, because the sampling time unit is pre-defined in this latter case.

3. The optimal motives are identified on the basis of the current estimates of the most typical motive types assigned to the lattice points of the SOM. Let $\underline{x}_k$ denote the contour vector belonging to the kth possible motive, and $\underline{w}_{i,j}$ the current estimate of the motive contour type belonging to the lattice point with the coordinates (i, j). The motive contour vector $\underline{x}_k$ is assigned to the most similar motive contour type vector of the SOM:

$$d_k = \min_{(i,j)}(\delta_k(i,j)) \quad ,$$
(3)

where the similarity measure $\delta_k(i, j)$ is the Euclidean distance between the $\underline{x}_k$ and $\underline{w}_{i,j}$.

$$\delta_k(i,j) = \sqrt{(\underline{x}_k - \underline{w}_{i,j})^T(\underline{x}_k - \underline{w}_{i,j})} \qquad (4)$$

Finally, the culture-based motives are defined using the following constraints:

- The distance of the motive and the corresponding motive type must be less than a critical value.
- The culture-defined motives are defined as the longest melody parts satisfying the above requirement.
- The culture-defined motives should not overlap with melody-defined motives. Melody-defined motives have priority.

4. The SOM is trained with the resulting set of culture-defined and melody-defined motives, using the well known algorithm. Each $\underline{x}_k$ vector determines a "winner" motive type contour on the SOM according to Equation 3, and the winner vectors $\underline{w}_{m,n}$ are modified towards the corresponding motive contour (denoting a winner position by (m,n) on the SOM). The motive type vectors located in the surroundings of a winner are also modified, while the radius defining the surroundings decreases during the training steps [17].

The input data vectors are usually invariable during the training process of self organising maps. In our system, however, they are variable, because the optimal culture-based motive identification depends on the current state of the motive type vectors $\underline{w}_{i,j}$ (see Equations 3 and 4). Since $\underline{w}_{i,j}$ are modified during the learning process, the optimal segmentation itself depends on the current state of the SOM. In other words, there exists a feedback between the segmentation and the learning algorithm, thus, our system converges to an optimal training- and feature vector set in parallel. The results of many independent training processes verified that all of the characteristic motive contour types have been learned consistently and independently of the starting conditions of the SOM-s.

4. ANALYSYS OF THE CULTURAL CONTACTS AMONG 22 CORPORA

Let suppose that we can create a whole collection of motive contour types, containing all the significant contours that appear in any of the 22 cultures. It is obvious that the national/areal sets of motive types can be considered as different subsets of this great common collection, therefore the study of musical connection between different cultures can be determined by the analysis of the intersections of these subsets.

Being in possession of the size of the great common motive contour type collection (N), the sizes of its two national/areal subsets (A and B), as well as the size of their intersection (X), the measure of the relationship between these cultures can be expressed by a probability as follows.

As a first step we compute the probability of the event that a random choice of two subsets with sizes A and B from the set of size N results in an intersection of size x, as

$$p(x) = \binom{N}{x}\frac{\binom{N-x}{A-x}\binom{N-A}{B-x}}{\binom{N}{A}\binom{N}{B}} \quad .$$
(5)

Using this probability density function, the probability of the event that the size of the intersection is less than X, is expressed as

$$P(X) = \sum_{x=1}^{X-1} p(x)$$
(6).

A high value of this probability indicates that the number of common contour types in the two corpora is much higher than the expected value in case of random correlations. Consequently the similarity, manifested by such high intersection of two corpora, cannot be a product of occasional coincidences of independent musical evolutions. It can be stated in such cases of similarity that the common musical characteristics implicate a historical or present, immediate or intermediate cultural interaction, that is, the established relationship is necessarily deterministic.

To construct the above mentioned sets, we first had to deduce the characteristic motive contour type collections for eash of the 22, by training 22 SOM-s of size 20*20 lattice points separately. After determining the 22 national/areal motive contour type collections, a new large self organizing map of size 30*30 was trained by the united set of them, in order to determine the set of all possible motive contour types appearing anywhere in the 22 cultures.

This common SOM allows us to classify all motive types of a given national/areal collection on it using Equations 3 and 4. We call this process "excitation of the common map by a culture". The values *A, B* and *X* can be determined for any selected two cultures by counting up the lattice points excited in the great common SOM. With these quantities, the calculation of the probability $P(X)$ can be carried out using Equations 5 and 6, knowing that *N* is equal to the total number of the common contour types. It is worth mentioning here that this calculation avoids the problems arising from the different sizes of the corpora, since the expected intersection decreases with decreasing subset sizes *A* and *B*.

The graph of the system of closest relationships is summarized in Figure 5, where a connection line indicates a high probability ($P(x) > 0.999$) of deterministic contact between the nodes of musical cultures. The Figure shows two main sub-graphs containing an "Eastern" - Mongolian, Chinese, Volga, Hungarian, Slovak, Moravian, Spanish, Kyrgyz, Romanian, Bulgarian, Azeri, Sicilian, Turkish and Karachay-Balkar, as well as a "Western" - Finnish, Norwegian, Irish-Scottish-English, French, German, Dutch, Luxembourgish and Cassubian – group of nodes. There are some interconnections between these two large sets due to the close connections of the Hungarian – Slovak – Finnish – (Irish-Scottish-English), and the Moravian - Norwegian corpora. Besides these close contacts of the Carpathian Basin to the Scandinavian and Irish-Scottish-English cultures, the Irish-Scottish-English and Norwegian corpora have certain further Eastern contacts to the Volga-region and Kyrgyzstan. Anyhow, the connection of the two main subsystems indicates a special role of the above mentioned cultures inside their main groups and also in the whole system.

The structure of the graph indicates certain smaller groups inside the great "Eastern" system. The majority of the motives belonging to the large pattern excited by the Mongolian, Chinese and Volga group on the common SOM move in the highest regions of the melodies – they start or end at the octave or higher notes (See the Mongolian motive contour type in Figure 4). The visible overlaps of the patterns of the Hungarian, Slovak, Karachay-Balkar, Turkish and Sicilian excitations with the above mentioned triad are based mainly on the above mentioned motives in the highest region of the melodies.

The patterns of the Irish-Scottish-English, Finnish and Norwegian excitations also indicate an important role of such motives, resulting in the deterministic contacts of these cultures to the Carpathian Basin and the Volga-region. However, this "Eastern" part of the common motive type map empties in the further Western patterns.

The French and Dutch contour examples show that the most common Western motive types move in the lowest

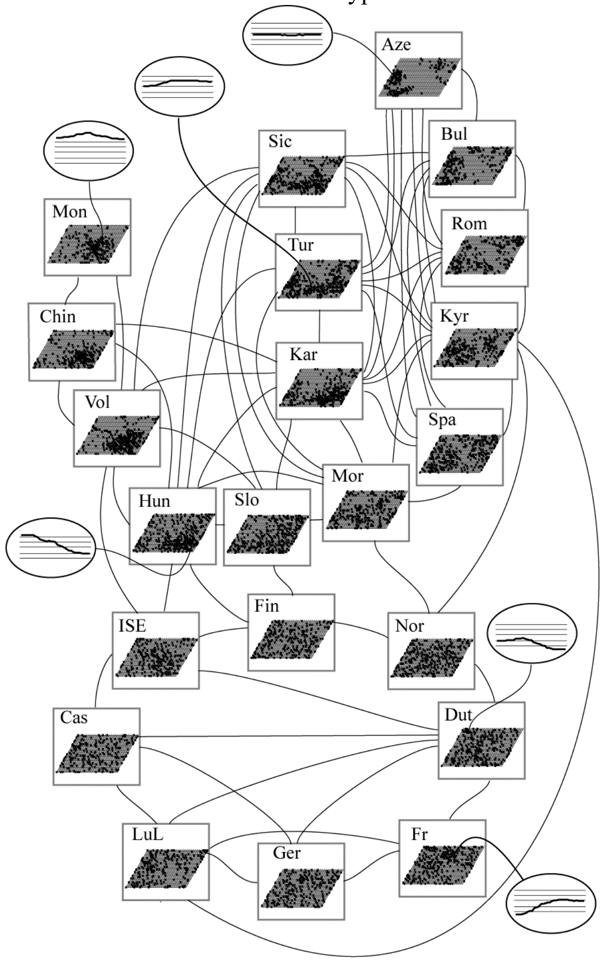


Figure 5. The graph of deterministic relations of 22 musical cultures in Eurasia.

ranges of the melodies, starting or ending at a fourth or fifth below the ending note.

The cloud of the high motives also disappears gradually along the branch of the Spanish – Kyrgyz – Romanian – Bulgarian – Azeri excitations, while the pattern on the left side of the motive type map becomes more and more emphasized. The Azeri motive example illustrates that the motive types belonging to this part of the map are of low ambit, ranging between the fourth, third or the second. The Sicilian, Turkish and Karachay-Balkar excitations show that these cultures also frequently apply such motive types, (beneath the above mentioned group of motives in high), indicating deterministic cultural contacts between the two branches. However, the group of these low-ambit motives practically misses in the Mongolian-Chinese-Volga branch, and it is also rather rare in the Hungarian, Slovak and Moravian melodies. Therefore, these cultures have no direct connections to the Spanish-Kyrgyz-Romanian-Bulgarian-Azeri branch.

SUMMARY

The very clear connections between the patterns of the different national/regional excitations on the common motive type map allowed us to analyze the musical structures of different cultures as different manifestations of a common motive set, and led to the conclusion that the main contacts between the cultures can be explained by the dominance/lack of a few motive types. This analysis clarified that "Eastern" cultures prefer motives in high regions of the melody, generally moving between the octave and the fifth as well as fourth, while the "Western" melodies prefer motives connecting the tonic to a fifth or a fourth beyond the tonic. The combined analysis of the contact probabilities and the overlaps of the national/areal patterns indicated several distinguishable branches among the Eastern cultures. The Mongolian-Chinese-Volga branch highly prefers motives in high, while the Sicilian-Turkish-Karachay branch evaluates a balance between these high motives and those of an explicitly low ambit. The close contacts of Hungarian, Slovak and Moravian cultures to these two distinguishable branches are based mainly on the high motive types. At the same time, the high motive types gradually disappear in the Spanish-Kyrgyz-Romanian-Bulgarian-Azeri branch, while the dominance of motives of low ambit connects them to the Sicilian-Turkish-Karachay branch.

Not forgetting the simplifications made during the application of our technique, we can state that the motive analysis allowed us to draw a rather perspicuous picture of the cross-cultural connections of different folksong cultures. We hope that these results may demonstrate the feasibility of an extended research of "musical linguistics", and suggest an efficient and quantitative tool for "melody mining", using artificial intelligence and other mathematical tools.

Acknowledgement. The Author is grateful to Ewa Dahlig-Turek, Damien Sagrillo, Louis Grijp, Hans-Hinrich Thedens, János Sipos and Gergely Agócs for their altruistic help to extend the database with Cassubian, Luxembourgian, Lotharingian, Dutch, Norwegian, Anatolian, Kyrgyz, Mongolian, Azeri and Karachay-Balkar collections.

References

[1] B. Bartók: "On Collecting Folk Songs in Turkey" Tempo, *New Ser., No. 13, Bartok Number* (Autumn, 1949), pp. 15-19+38

[2] Z. Kodály.: "Folk Music of Hungary". Budapest, Corvina.

[3] D. Huron: "The melodic arch in Western folksongs". *Computing in Musicology,* Vol. 10, pp. 3-23.

[4] Z. Juhász: "A systematic comparison of different European folk music traditions using self-organising maps". *Journal of New Music Research* 2006, Vol. 35, No. 2, pp 95-112.

[5] O. Lartillot and P. Toiviainen. (2007). "Motivic matching strategies for automated pattern extraction", *Musicae Scientiae*, Discussion Forum 4A, pp. 281-314.

[6] E. Cambouropoulos : « Musical Parallelism and Melodic Segmentation", *Proceedings XII Colloquium on Musical Informatics*, Gorizia, Italy

[7] D. Halperin: "A Segmentation Algorithm and its Application to Medieval Monophonic Music". *Musikometrika* 2 (1990)

[8] J. Singer: "Creating a nested melodic representation: competition and cooperation among bottom-up and top-down Gestalt principles". ISMIR 2004

[9] E. Cambouropoulos: "A Formal Theory for the Discovery of Local Boundaries in a Melodic Surface". *Proceedings of the Troisiémes Journées d'Informatique Musicale (JIM-96),* Caen, France

[10] E. Charniak: "Tree-bank Grammars", *Pproceedings AAAI-96* Menlo Park, Ca

[11] E. Charniak: "A Maximum-Entropy-Inspired Parser". *Proceedings ANLP-NAACL'2000,* Seattle, Washington

[12] S. Seneff: "TINA: A Natural Language System for Spoken Language Applications". *Computational Linguistics 18(1), 61-86.*

[13] R. Bod: "Memory-Based Models of Melodic Analysis: Challenging the Gestalt Principles". *Journal of New Music Research* 31 (2002) 27-37.

[14] R. Bod: "Probabilistic Grammars for Music Proceedings*" BNAIC 2001*, Amsterdam

[15] Z. Juhász: "Segmentation of Hungarian Folk Songs Using an Entropy-Based Learning System". *Journal of New Music Research* 33 (2004) No 1, (pp 5-15).

[16] D. Conklin, Ch. Anagnostopoulou: "Segmental Pattern Discovery in Music", *Informs Journal on Computing*,
Vol. 18, No. 3, Summer 2006, pp. 285-293
DOI: 10.1287/ijoc.1040.0122

[17] T. Kohonen: „Self-organising Maps". Berlin:Springer-Verlag

[18] P. Toiviainen: "Symbolic AI Versus Connectionism in Music Research". *In E. Mirinda (Ed.), Readings in Music and Artificial Intelligence.* Amsterdam: Harwood Academic Publishers (2000).

IMPROVING RHYTHMIC SIMILARITY COMPUTATION BY BEAT HISTOGRAM TRANSFORMATIONS

Matthias Gruhne
Bach Technology AS
ghe@bachtechnology.com

Christian Dittmar
Fraunhofer IDMT
dmr@idmt.fhg.de

Daniel Gaertner
Fraunhofer IDMT
gtr@idmt.fhg.de

ABSTRACT

Rhythmic descriptors are often utilized for semantic music classification, such as genre recognition or tempo detection. Several algorithms dealing with the extraction of rhythmic information from music signals were proposed in literature. Most of them derive a so-called beat histogram by auto-correlating a representation of the temporal envelope of the music signal. To circumvent the problem of tempo dependency, post-processing via higher-order statistics has been reported. Tests concluded, that these statistics are still tempo dependent to a certain extent. This paper describes a method, which transforms the original auto-correlated envelope into a tempo-independent rhythmic feature vector by multiplying the lag-axis with a stretch factor. This factor is computed with a new correlation technique which works in the logarithmic domain. The proposed method is evaluated for rhythmic similarity, consisting of two tasks: One test with manually created rhythms as proof of concept and another test using a large real-world music archive.

1. INTRODUCTION

During the last years the need of new search and retrieval methods for digital music increased significantly due to the almost unlimited amount of digital music on users hard disks and in online stores. An important pre-requisite for these search methods is the semantic classification, which requires suitable low- and mid-level features. The major goal of many researchers is the computation of mid-level representations from audio signals, which are destined to capture the rhythmic gist from the music. A huge amount of work has been done in this field so far by developing techniques like beat histogram, inter-onset-interval histogram or rhythmic mid-level features, e.g., [1], [2], [3], [4], [5]. In general, the beat histogram technique very often used as feature basis for semantic classification. This histogram is computed by taking the audio spectrum envelope signal, which is differentiated and half/full-wave rectified. As a final step an auto-correlation function is applied, which estimates the periodicities within the modified envelope. The resulting feature vector is only limited usable for pattern recognition. Two similar rhythms are easily comparable with the beat histogram as feature, if their tempi are equal. A different tempo leads to a compression or expansion of the lag-axis, as depicted in Figure 1. This modification has a disadvantageous effect when performing a comparison of beat histograms via Euclidean distance measure. This issue has been raised by Foote [6]. A number of approaches tried to come up with solutions for that challenge. Paulus [7] presented a method, which could be considered reasonable for comparing beat histogram vectors containing different tempi by applying a dynamic time warping technique. A similar approach has been also proposed by Holzapfel [8]. These techniques require specialized classifiers and the beat histogram cannot be used as feature in conjunction with other low-level features. In order to solve that problem, Tzanetakis [1], Gouyon [2], and Burred [3] computed descriptive statistics, such as mean, variance, and kurtosis on the beat histogram. These statistics were used as feature vector for classification. To a certain degree, these are also tempo-dependent. This paper suggests a new post-processing method which performs a transformation of the beat histogram into the logarithmic lag domain. The transformation into the logarithmic domain has not been described for rhythm features, but for harmonic and chroma features in [9] and [10]. This transformation transfers the multiplicative factor of the tempo changes into an additive offset. Hence, the transformed rhythmic feature vector contains a tempo independent part located on the right-hand side of the vector. An approach for detection of this tempo independent rhythmic information is presented. A number of different features were extracted and evaluated for the task of rhythmic similarity.

The remainder of this paper will be organized as follows: Section 2 introduces the proposed algorithm, Section 3 describes the evaluation and discusses the results. Section 4 concludes and indicates further directions in this area.

2. PROPOSED APPROACH

In this work, the beat histogram is extracted from MPEG-7 AudioSpectrumEnvelope (ASE) features [11]. Different variants of the basic feature extraction algorithm have been reported in literature. Tzanetakis' [1] work was based on a wavelet transform, Scheirer [12] used a filter bank. Nevertheless, both authors extracted an envelope signal from non-linearly spaced frequency bands, as is the case with ASE. In the proposed implementation, the different ASE bands are smoothed in time. Subsequently, the bands are weighted by enhancing the lower and higher frequency bands and decreasing the center frequencies. All bands are accumulated, differentiated in time, and full-wave-rectified. This results into a so-called detection function, containing the the most salient rhythmic information of the music signal. The detection function is subdivided into snippets of N successive frames. The auto-correlation inside such a frame yields the beat histogram, also called rhythmic mid-level feature, beat spectrum, etc.

The beat histogram may be used in a different number of applications, such as beat tracking or tempo detection. As already mentioned in Chapter 1, this vector should not be directly utilized for classification. If two similar rhythms are played in different rhythms and there beat histograms are compared, the vectors would look similar, but one would be a more stretched or compressed (in terms of the lag-axis) version from the other. Hence, a direct comparison of these vectors using common distance measures (e.g., Euclidean distance) results in large distances. Thus, it is state of the art to compute descriptive statistics from the beat histogram and use these measures as features for classification. Unfortunately, these statistics are also prone to tempo changes.

In order to create a tempo independent beat histogram, Foote [6] proposed to stretch or compress the original vector based on the tempo of the rhythm. The compression of the beat histogram can be considered as multiplication of a time-stretching factor f with the argument τ of the underlying pattern signal $c(\tau')$. This pattern signal can be the mentioned auto-correlation signal. The observed feature vector can therefore be described with $c(\tau) = c(\tau' * f)$. In order to obtain the tempo invariant beat histogram $c(\tau')$, the stretch factor f needs to be known, but its automatic computation might be unreliable. One option for solving this issue is to use a logarithm function. By applying the logarithm on an arbitrary function, multiplicative terms are transformed to additive terms. Transferring this theorem to the lag-axis of the beat histogram $c(\tau)$ leads to the equation (1):

$$c(\log(\tau' * f)) = c(\log(f) + \log(\tau')) \tag{1}$$

For the logarithmic processing step, a new argument is estimated by (2):

$$\tau_{log} = \frac{\log(\tau) * max(\tau)}{\log(max(\tau))} \tag{2}$$

Resampling the original beat histogram $c(\tau)$ in such a way, that the values in τ are available on places of τ_{log} results in a new beat histogram feature with logarithmized lag-axis (Figure 2 d).

Since τ_{log} consists of non-integer values, the practical implementation of this variable requires an interpolation. For this task, a bicubic interpolation method as described in [13] has been applied.

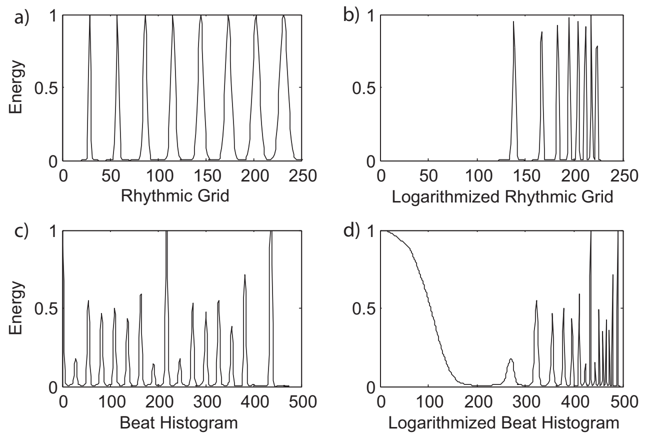


Figure 2. This figures shows an example beat histogram (c) and a rhythmic grid (a) and their logarithmic counterparts (d,a, respectively).

Figure 2 c,d shows an example beat histogram and its transformation into the log-lag domain.

By inspecting a large number of such logarithmized vectors it can be observed, that all vectors consist of a large decaying slope towards a first local minimum, whose absolute position depends on the tempo of the music. That slope represents the first maximum lobe of the auto-correlation function. Due to the fact, that a time-varying signal is always most similar to itself for small lags, the first lobe is always the highest and does not carry any significant rhythmic information. However, the successive minimum appears to be the point from where on the logarithmized beat histogram shows similar tempo-independent characteristics if the rhythm is similar. These characteristics are similar, but they are moved further right or further left, depending on the tempo. The goal is to find the starting point of these tempo-independent characteristics and to use the tempo-independent excerpt of the feature vector for classification. In the original beat histogram the first local minimum (or maximum) could be used as starting point for stretching or compressing the vector in order to receive a tempo-independent version. Unfortunately, this procedure is only applicable on a minority of rhythms, since often the first local minimum is misleading and the stretched vector results in octave errors. In the log-lag domain the result would be similar, if only the first minimum is used. The proposal in this publication is to find the point more reliably by taking the evolution of the vector into account. Therefore, the authors use an artificial rhythmic grid featuring eight successive Gaussian pulses as depicted in Figure 2 a. The Gaussian pulses are computed as described in the following Matlab code snippet (Code 1) with the blocksize $blksize$ as functional parameter and tmp_acf as result vector.

This rhythmic grid is transformed into the logarithmic domain with the same method as described above. In order to find the tempo-independent characteristics of the logarithmized beat histogram, both vectors, the logarithmized rhythmic grid and the logarithmized beat histogram are

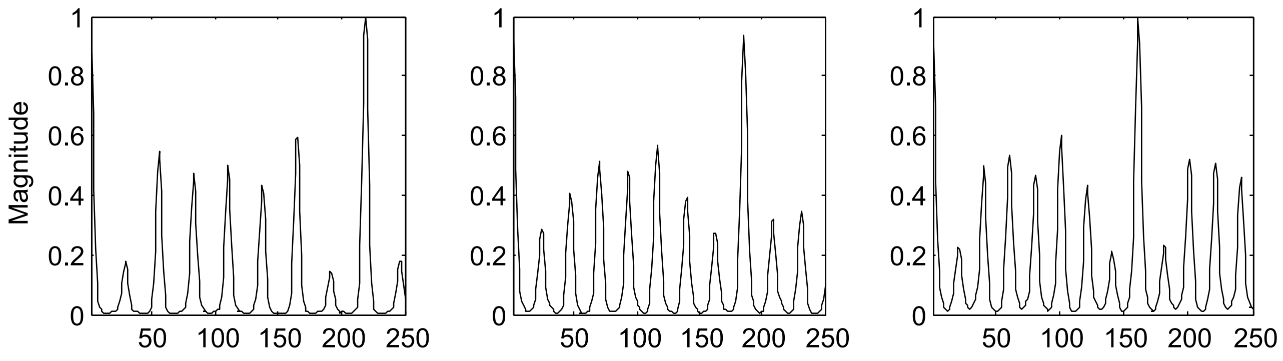


Figure 1. These figures depict a beat histogram excerpt for the same rhythm with tempos of 90 Bpm (left), 110 Bpm (middle), 130 Bpm (right).

Code 1 Example Matlab code for the creation of Gaussian pulses

```
mu = [29:29:blksize]; sd = 2;
tmp_acf = zeros(1,blksize); lobe=[];
for k = 1:length(mu)
t_exp=-0.5*(((1:blksize)-mu(k))/sd).^2;
lobe(k,:) = exp(t_exp)/(sd*sqrt(2*pi));
lobe(k,:) = lobe(k,:)/max(lobe(k,:));
tmp_acf = tmp_acf + lobe(k,:);
end
```

cross-correlated. Best results could be achieved by only evaluating only the first slope (histogram points 200-300 in 2 d). The maximum of the correlation function equals the point in the vector, where the tempo-independent characteristic starts. A faster tempo results in a shift of the tempo-independent part to the left, and thus additional peaks appearing at the right border. In order to process almost identical beat histograms, regardless of the tempo, the length of the tempo independent characteristics has to be suitably restricted. This tempo independent vector could be theoretically used as feature vector for rhythmic similarity. Due to the interpolation for the logarithmic processing, small variations lead sometimes to a small movement either to the right or to the left side of the axis. These small variations affect the rhythmic similarity negatively. In order to reduce this effect, statistical measures as proposed by the other authors have been applied in the tests for this paper. The following statistics as described by Tzanetakis [1], Gouyon [4], and Burred [3] were computed from the tempo independent vector. All statistics from these authors were appended and formed the final feature vector for the experiments:

- Tzanetakis: Relative amplitude (divided by the sum of amplitudes) of the first, and second histogram peak; ration of the amplitude of the second peak divided by the amplitude of the first peak; period of the first, second peak in bpm; overall sum of the histogram
- Gouyon: Mean of magnitude distribution; geometric mean of magnitude distribution; total energy; centroid; flatness; skewness; high-frequency content
- Burred: Mean; standard deviation; mean of the derivative; standard deviation of the derivative; skewness; kurtosis and entropy.

Since some statistics from Gouyon and Burred partly overlapped the final feature size consisted of 18 dimensions. For the practical implementation, excerpts of 500 ASE frames were chosen, which corresponds to 5 seconds in music, given a low-level hop-size of 10 milliseconds. This size constitutes a trade-off between the length of at least two repeating patterns and the ability to track abrupt tempo changes sometimes encountered in real-world music. A correlation size of 5 seconds has been also used in previous approaches (e.g., [14]). Since the test songs contain more than five seconds of audio content, one of such a feature vector is computed every 0.5 seconds. In order to compute the Gaussian pulses, a default standard deviation of 2 has been chosen and and only eight successive pulses were used in the evaluation. Another standard deviation could also be chosen, which increases/decreases the width of the pulses.

For the tests in this paper, the following 4 feature vectors were created:

- Statistics of original beat histogram: The beat histogram has been extracted as described in this paper. Based on that histogram, a feature vector containing all statistics by Tzanetakis [1], Gouyon [4], and Burred [3] as described above was extracted.
- Statistics of logarithmized beat histogram: The statistics by Tzanetakis, Gouyon, and Burred were computed from the logarithmized beat histogram technique as described above.
- Statistics of beat histogram with stretch factor: Based on the logarithmized beat histogram, a point has been estimated, where the tempo-independent rhythmic characteristic begins. This point has been transformed into the non-logarithmic domain and a stretch factor (as proposed by Foote) has been computed. The original beat histogram has been stretched by the

stretch factor and the statistics from Tzanetakis, Gouyon, and Burred were computed from that vector.

- Beat histogram with stretch factor: The original beat histogram has been stretched as suggested by Foote with the stretch factor derived from the logarithmic post-processing.

3. EVALUATION

3.1 Evaluation Procedure

In order to test the logarithmic post-processing of the beat histogram, two different evaluation strategies were implemented. The first test evaluated a number of manually created rhythms in order to prove the theoretic improvement of the results. The second test evaluates rhythmic similarity based on beat histograms with a large real-world music set.

3.1.1 Tests based on manually created rhythms

The first test scenario examined the tempo dependence of the described feature sets based on different rhythms. A number of 18 different base rhythms were established, which can be divided into 9 rhythm genres, e.g., electro, drum'n'base or hip hop. The rhythms were played without any additional instruments in order to test the tempo dependence of only the base rhythms. Each of these rhythms was played in six different tempo variations ranging from 90 Bpm to 190 Bpm in 20 Bpm steps. Each base rhythm was repeated a number of times, whereby the duration of one single rhythm pattern was less than 5 seconds. A total of 108 rhythms were collected and the low-level ASE features as well all four versions of the described mid-level features were extracted. Since the window length of the described mid-level features consisted of 5 seconds, the base rhythm of every rhythm class is contained in every frame of the feature matrix. Therefore, an arbitrary frame from the feature matrix can be chosen for comparison. In the evaluation for this paper, the second consecutive vector was used as mid-level feature. Prior to the classification, a mean and a variance normalization step over all data was applied. A simple k-nearest neighbor classifier with Euclidean distance was set up using the features and the rhythm class information as ground-truth. k for the k-nearest neighbor classifier has been chosen to be one. Subsequently, all features were consecutively used as query to the classifier, whereby it has been ensured, that the query item was not contained in the reference set. The evaluation method returned the distance and the closest class to each of the 108 rhythms. The average accuracy has been estimated per class. The minimum, maximum and average of the overall test set has been estimated by using the class-dependent accuracy. Based on the results of this simple classifier a base-line assumption can be made about the accuracy of the tempo independent rhythmic classification. One might raise concerns that the comparison of base rhythms is not very practice relevant, since popular music contains additional polyphonic properties in the signal, which may interfere with the beat histogram. In order to prevent this "distortion" it has been shown, e.g. in [15], that drum transcription algorithms as preprocessing steps have a positive effect on beat histogram.

3.1.2 Tests based on a large test set

To evaluate the performance on real world data instead of the rather artificial data, a diverse set of 753 songs from 60 different genres and sub-genres was compiled. Rhythmic similarity measures are hard to evaluate by using real world data. An option for testing rhythmic similarity measures can be based on the assumption, that songs from the same genre have similar rhythms, while songs from different genres have different rhythms. But similar rhythms might be also available across genres and the results would not directly predicate rhythmic similarity. To cope with that, another approach was chosen. A rhythm similarity ground truth was manually created for the used dataset. First, for each song, a representative rhythm pattern was annotated by hand, then a similarity matrix from all pairs of rhythms was calculated.

Representative rhythm pattern: For each song, one representative rhythm pattern was manually annotated. Five different classes of rhythmical events were differentiated: base drum, snare drum, hi hats, further percussive events, and non-percussive events. A quantization could be freely chosen, but in general, events have been quantized onto 1/16 bar length in case of a 4/4 bar and 1/12 bar length in case of a 3/4 bar. Similarity between patterns: The distance between two characteristic patterns was calculated by performing the following steps. First, both of the patterns have been stretched onto the same length. Then, all the simultaneous occurrences of an event of a certain class in both patterns were summed up. Finally, the resulting value was normalized by the length of the pattern. For each of the mentioned percussion classes, the 753x753 distance matrix was computed. Afterwards, the mean distance matrix was estimated by equally weighting all distances of the distance matrices from each percussion class.

Also, for each song in the database the features described above were extracted whereby the mean value for all feature frames of a song was calculated. Using Euclidean distance, the 5 closest songs to each song excluding the query itself were determined. The list of the 5 closest songs to the query song C are denoted L_C. Incorporating both the ground-truth rhythm similarity matrix and the list of the 5 closest songs for each of the 753 queries, the different feature sets were compared using the following procedure: For each query song C, a list T_C of all the other songs, was generated. This list was sorted in ascending order of the distances derived from the manually annotated rhythm patterns. Then, for each song c in L_C the number of songs in T_C have been counted, which were closer to C than c. By averaging over these numbers, a value r is calculated. This value describes the mean number of songs in T_C that are closer to the query song than the retrieved songs. In order to obtain a statement about the accuracy of the system in such a way, that higher numbers refer to better results, a score has been computed by $S_i = |S - 1|$.

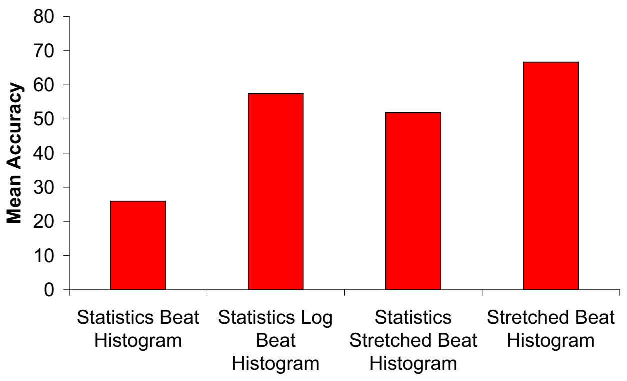


Figure 3. Average accuracy for rhythmic classification of the first test based on different feature vectors in percent.

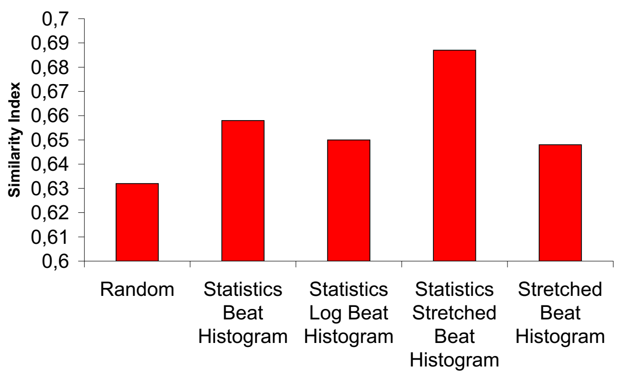


Figure 4. Similarity index (second test) expressing the rhythmic similarity for different feature vectors.

	Mean	Min	Max
Stat. Original Beat Hist.	25.93	0.00	83.33
Stat. Logarithm. Beat Hist.	57.41	0.00	100.00
Stat. Stretched Beat Hist.	51.85	16.67	66.67
Stretched Beat Hist.	66.67	33.33	100.00

Table 1. Accuracy measures (first test) for rhythm classification based on different feature vectors in percent.

This score is referred to the term similarity index. For significance purposes a random score has been established by generating a random result list for each of the 753 songs. This result list has been evaluated in a similar procedure as the described mid-level features.

Other rhythmic similarity measures were described in literature by Hofman-Engl [16] and Toussiant [17]. These measures are established when it comes to the comparison of actual rhythmic descriptions. In this paper features based on rhythms are to be compared. Therefore, these methodologies could not be applied.

3.2 Results and Discussion

3.2.1 Test based on manually created rhythms

The following table (Table 3.2.1) shows the results for the first test containing the manually created rhythms. This table shows minimum, maximum and mean accuracy. In order to get a quick overview about the results in general, the mean is also plotted in Figure 3.

The state of the art methodology by computing statistics over the beat histogram achieves an average accuracy of approx. 26%. This is based on the fact, that the statistic measures are by far not tempo independent. Better results could be obtained by the logarithmic post-processing step. The statistics computed on the logarithmized beat histogram and over the stretched beat histogram performed reasonably well with 57.4% and 51.9%, respectively. The best results could be obtained by the stretched beat histogram with the stretch factor computed from the logarithmized beat histogram. This methodology leads to an average accuracy of 66.7%. An intuitive guess would be, that identical rhythms in different tempos should always return an accuracy of 100%. In practice, the results look different due to windowing effects. The minimum accuracy of the algorithms ranges from 0% to 33.3%. This is based on the fact, that the separability between some of the 18 base rhythms is strongly restricted. The highest accuracy is obtained by the stretched beat histogram also in case of the minimum. This might imply that postprocessed beat histogram performs better as feature than the statistics over postprocessed beat histograms. A similar statement can be also made by evaluating the maxima of the four feature vectors. These tests prove, that the tempo independent version of the beat histogram (stretched beat histogram) outperforms the statistics over the beat histogram.

3.2.2 Test based on a large real-world music set

The following figure (Figure 4) shows the accuracy for the test with real-world music. Additionally, these numbers are depicted in Table 3.2.2. The similarities between manually annotated base rhythms and the beat histogram features are expressed by a similarity index. The higher the index is, the better is the similarity between the manually annotated rhythms and the automatically extracted rhythms. The figure shows, that a random generation of similarities results with a similarity index of 0.632. Most of the observed feature vectors obtained a similarity index around 0.65, including the statistics over the beat histogram, the statistics over the logarithmized beat histogram and the stretched version of the beat histogram. The statistics computed from the stretched beat histogram outperform all other results by a similarity index of 0.03.

The first test, which was based on the manually created rhythms, showed the best results on the stretched beat histogram. In this second test, these results cannot be validated in every case. This may be based on the fact that the point in the logarithmic domain, which separates the tempo dependent and tempo independent parts is inaccurate in a few cases. These inaccuracies have influence on the stretched beat histogram and may result in octave errors, which affect the rhythmic similarity. However, computing the descriptive statistics over the resulting vectors improves the results. These statistics seem to neglect the slight deviations significantly. This test on real world data might be not optimal, since rhythms in real songs might

Feature Name	Similarity Index
Random	0.632
Stat. Original Beat Hist.	0.658
Stat. Logarithm. Beat Hist.	0.650
Stat. Stretched Beat Hist.	0.687
Stretched Beat Hist.	0.648

Table 2. Similarity index of the second test expressing the rhythmic similarity for different feature vectors.

change and the evaluation was performed on one representative rhythm of the song. But this methodology gives a rough indication of the performance of the logarithmic processing.

4. CONCLUSIONS AND FUTURE WORK

The rhythmic information from music is captured by the commonly used beat histogram. This paper presented a post-processing technique for the beat histogram, which is based on logarithmic re-sampling of the lag axis and cross-correlation with an artificial rhythmic grid. This technique seems to improve the applicability of the beat histogram technique as feature for music information retrieval tasks. The practical tests on a large music archive were based on a mean feature vector per song. In order to be more accurate, future tests should perform a rhythmic segmentation and analyze the segments individually. The logarithmic processing methodology as described in this paper may be also beneficial for beat tracking and tempo detection. Future tests will provide an evaluation, if the tempo estimation results can be improved when using the proposed algorithm.

5. ACKNOWLEDGMENT

This work has been partly supported by the PHAROS Integrated Project (IST-2005-2.6.3), funded under the EC IST 6th Framework Program. Additionally, this project has been funded by the MetaMoses project (nr. 183217) from the Norwegian research council.

6. REFERENCES

[1] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech, Audio, and Language Processing*, 10(5):293–302, 2002.

[2] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1), 2005.

[3] J. Burred and A. Lerch. A hierarchical approach to automatic musical genre classification. In *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, 2003.

[4] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Proceedings of the 25th AES International Conference*, 2004.

[5] S. Dixon, F. Gouyon, and G. Widmer. Towards characterisation of music via rhythmic patterns. In *Proceedings of the 25th AES International Conference*, 2004.

[6] J. Foote and S. Uchihashi. The beat spectrum: A new approach to rhythm analysis. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, 2001.

[7] J. Paulus and A. Klapuri. Measuring the similarity of rhythmic patterns. In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR)*, 2002.

[8] A. Holzapfel and Y. Stylianou. A scale transform based method for rhythmic similarity of music. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009.

[9] S. Saito, H. Kameoka, T. Nishimoto, and S. Sagayama. Specmurt analysis of multi-pitch music signals with adaptive estimation of common harmonic structure. In *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.

[10] J. Jensen, M. Christensen, D.P.W. Ellis, and S. Jensen. A tempo-insensitive distance measure for cover song identification based on chroma features. In *Proceedings of the IEEE International Conference on Audio, Acoustics, and Signal Processing (ICASSP)*, 2008.

[11] M. Casey. Mpeg-7 sound recognition. *IEEE Transaction on Circuits and Systems Video Technology, special issue on MPEG-7*, 11:737–747, 2001.

[12] E. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

[13] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[14] S. Dixon, E. Pampalk, and G. Widmer. Classification of dance music by periodicity patterns. In *Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR)*, 2003.

[15] M. Gruhne and C. Dittmar. Improving rhythmic pattern features based on logarithmic preprocessing. In *Proceedings of the 126th Audio Engineering Society (AES) Convention*, 2009.

[16] L. Hofmann-Engl. Rhythmic similarity: A theoretical and empirical approach. In *Proceedings of the Seventh International Conference on Music Perception and Cognition*, 2002.

[17] G.T. Toussaint. A comparison of rhythmic similarity measures. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004.

USING SOURCE SEPARATION TO IMPROVE TEMPO DETECTION

Parag Chordia
Dept. of Music, GTCMT
Georgia Institute of Technology

Alex Rae
Dept. of Music, GTCMT
Georgia Tech Center for Music Technology

ABSTRACT

We describe a novel tempo estimation method based on decomposing musical audio into sources using principal latent component analysis (PLCA). The approach is motivated by the observation that in rhythmically complex music, some layers may be more rhythmically regular than the overall mix, thus facilitating tempo detection. Each excerpt was analyzed using PLCA and the resulting components were each tempo tracked using a standard autocorrelation-based algorithm. We describe several techniques for aggregating or choosing among the multiple estimates that result from this process to extract a global tempo estimate. The system was evaluated on the MIREX 2006 training database as well as a newly constructed database of rhythmically complex electronic music consisting of 27 examples (IDM DB). For these databases the algorithms improved accuracy by 10% (60% vs 50%) and 22.3% (48.2% vs. 25.9%) respectively. These preliminary results suggest that for some types of music, source-separation may lead to better tempo detection.

1. BACKGROUND AND MOTIVATION

A working definition of tempo is the rate of the underlying rhythmic pulse of music determined by a human listener tapping along to the music, typically expressed in beats per minute (BPM). This may differ from a notated tempo, and different listeners, or the same listener at different times, often entrain to different metrical levels, so that some tapping rates may be half or double as fast as others. Further, in some types of music, the most natural way to tap along is asymmetric (e.g. tapping on the accented first and third beat in a fast group of five beats). For our purposes, these complexities are important to acknowledge at the outset as they set natural bounds on performance and suggest appropriate ways of judging accuracy.

Tempo estimation is a fundamental MIR task and underlies almost all rhythmic descriptions of music. However, state-of-the-art tempo detection is still highly variable in its accuracy, working well on most simple cases, but often performing poorly or not at all on rhythmically complex music [1]. The current work is motivated by two observations: 1) rhythmically complex music may be constructed out of components or layers (e.g. musical parts or sources) that are rhythmically simpler than the mix and thus easier to track; 2) in many types of music, humans track the beat or the tempo by hearing out a particular instrument or part. For example, in many types of rhythmically complex electronic music, a "click track" is present in the mix. More generally, in many musical genres a particular part plays a time-keeping function: for example, in standard jazz the walking bass line is the time keeper, in Indian music the tabla, in Afro-Cuban music the clave. Being able to hear out these time-keeping parts makes tempo tracking easier for humans.

2. RELATED WORK

The starting point of the current work is tempo detection that looks for periodicities in the signal by taking the autocorrelation of the detection function (ACF). A good review of current algorithms can be found in McKinney et al. [2] as well as specific descriptions of autocorrelation-based approach in Ellis [3] and Davies and Plumbley [4]. Recent work has explored the extension of this basic approach to tempo detection in a variety of ways. Wright et al. [5] describe a system that searches for the rhythmic pattern of the clave in Afro-Cuban music and show that such an approach out-performs techniques more reliant on isochronous events such as the Ellis and Dixon [6] algorithms. In their work, a matched filter is used to extract the clave from the mix. In this paper, we attempt to generalize the idea of finding the time-keeper in the mix in a way that is less reliant on domain-specific knowledge. Seyerlehner et al. [7] cast tempo estimation as a nearest neighbor problem, representing instances using a smoothed autocorrelation function (ACF). This approach suggests the idea of using not just the peak of the ACF, but including other features to improve tempo detection. Xiao et al, [8] demonstrate that using timbral features in addition to ACF-based features can reduce double/half tempo errors and indicates that even very crude uses of timbre can improve tempo estimation accuracy. Earlier work on tempo detection has also sought to improve accuracy by processing information in particular frequency sub-bands [9, 10]. In some cases, this is akin to a crude source separation, for example, separating the bass drum from the rest of a song.

Probabilistic latent component analysis (PLCA), a technique for source-separation described in Section 3.2, has been used for unmixing as well as transcription [11, 12].

The closely related technique of non-negative matrix factorization (NMF) has been used to improve drum detection [13, 14]. In these works, tracks were separated into sources that were then grouped into either tonal or percussive layers based on features of the components. This is relevant to the current work because it demonstrates the idea of using source-separation as a pre-processing step to improve performance on a standard MIR task. Additionally, features of the components are used to classify them into different groups, a technique used in this work to judge how strong a pulse different components have.

3. METHOD

3.1 Overview

As stated, the technique described here builds on ACF-based tempo detection. First, the track is separated into components using a single-channel source separation method (PLCA). Next, the tempo of each component is estimated on the separated audio. The component tempo estimates, along with the windowed ACF that was used to calculate the component tempo, are then used to find a global tempo estimate for the excerpt. We discuss several attempts to solve the problem of finding the best tempo estimate from the components. Two basic strategies were employed: selecting the tempo of a component with the highest estimated rhythmic clarity, and clustering component tempo estimates and weighting each cluster by the rhythmic clarity of each element in the cluster. Figure 1 shows a block diagram of the system.

3.2 Source Separation

Blind source separation attempts to recover constituent elements from a signal without any specific *a priori* knowledge of their characteristics. For audio, this corresponds to "unmixing," the reconstruction of a clean signal of each of a number of sounds that have been mixed together. Faithful reconstruction of component elements has a wide array of potential applications; in the current work we are less interested in mimicking the timbre of the original sources than in capturing rhythmic characteristics that may be less evident in a full mix.

We approach this task using the non-shift-invariant version of Probabilistic Latent Component Analysis (PLCA) [11, 12]. The input to the PLCA is a spectrogram, computed using a 1024 sample Hann window with a hopsize of 256 samples and then normalized to be a valid probability distribution. Latent variables representing the components are estimated using expectation maximization, and the output consists of a magnitude spectrum and relative contribution over time for each component; the number of desired components must be specified by the user.

After some experimentation, we set the number of components to be extracted to eight. A more systematic evaluation of the optimal number components remains for future work. The corresponding timbral and temporal profiles were used to synthesize audio for each component using phase information from the original audio.

3.3 Tempo Estimation

The tempo was estimated for each component using the Ellis algorithm [3]. The algorithm constructs a detection function based on a 40-channel db-magnitude mel spectrogram. First the signal is downsampled to 8 kHz, mixed to mono and divided into 32 ms frames with a 4 ms hopsize. The first-order difference is taken for each channel, and the sum of positive values across all channels is the value of the detection function for that frame (spectral flux). The autocorrelation of the detection function is calculated and then windowed to bias it towards tempos close to 120 BPM. The windowing effectively excludes tempos falling outside an acceptable range, and at the same time mimics the natural preference of humans to tap at rates between 90-120 BPM [1]. The tempo estimate is simply the lag time corresponding the peak value of the windowed ACF, converted to BPM. Any peaks before the first zero-crossing of the ACF are disallowed to prevent spurious peaks near zero lag. A small modification was made to the Ellis algorithm so that the top ten tempo candidates were returned rather than a single best tempo estimate, defined as the BPMs corresponding to the ten highest peaks in the windowed ACF. These additional tempo estimates were used in the clustering method described below. For all other techniques, only the best estimate for each component was used.

Each component was tempo tracked in this way, resulting in ten candidate tempos for each component. This meant that for a given track there were 80 tempo candidates (8 components $\times$ 10 estimates). The ACF value associated with each candidate and the entire windowed ACF were also stored, and these were used to help select the best global tempo estimate from the candidates.

3.4 Tempo Selection

Below we describe several approaches to selecting a single tempo estimate from the candidates.

3.4.1 Pulse-clarity

Inspired by the idea that certain components might accurately represent a relatively isochronous part of the track, the first approach focused on finding the best component from which to estimate the global tempo. That is, we attempted to find the component with the clearest pulse, and then choose the highest ranked tempo estimate for that one component as the global tempo estimate.

Lartillot et al. [15] showed that several features of the ACF are correlated with human judgments of pulse-clarity. Intuitively, the idea is that a relatively isochronous part with clear onsets will lead to an ACF that has well-defined and relatively large peaks. Following Lartillot et al., we calculated the following features on the ACF: maximum, minimum, and kurtosis. Additionally we added entropy and sparseness [16] as features, with sparseness defined as:

$$\text{sparseness(x)} = \frac{\sqrt{n} - (\sum |x_i|)/\sqrt{\sum x_i^2}}{\sqrt{n} - 1} \quad (1)$$

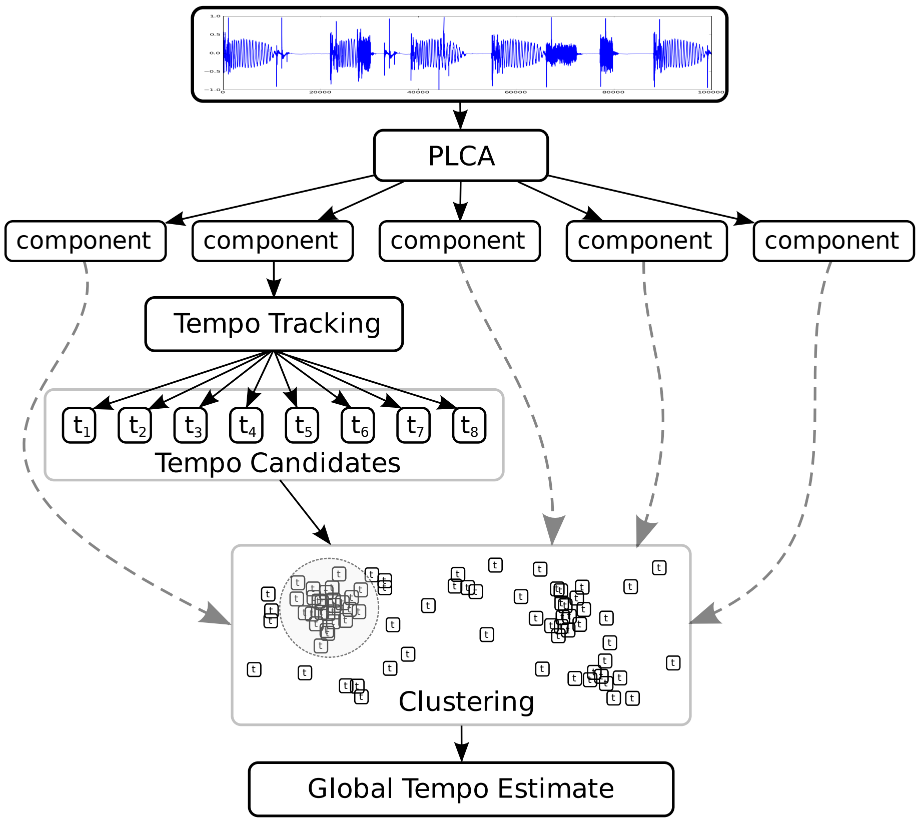


Figure 1. Block diagram of tempo estimation algorithm

Before calculating these features the ACF was normalized so that higher amplitude components would not dominate. For each component the ACF values were divided by the sum of the absolute value of all ACF values. The max and min were simply the maximum and minimum ACF values after normalization, and we expected that larger absolute values would correspond with greater pulse-clarity. Kurtosis was used to measure the peakiness of the ACF, i.e. how well-defined the ACF peaks were. Entropy and sparseness also assessed peakiness.

Each feature was evaluated separately; Table 2 summarizes the performance of each feature (evaluation criteria are discussed in Section 4.2). It can be seen that the most obvious feature, the maximum ACF value, outperformed the other measures on the IDM09 data, while they were about equal on the MIREX06 data.

In order to make better use of pulse-clarity features, an attempt was made to apply them to a more systematic supervised machine-learning framework. For this, we trained a multivariate Gaussian classifier using a ten-fold cross validation scheme. In addition to the ACF features we defined a new set of features based on the ratios of the candidate tempo estimates for each component. These features were based on the idea that we would expect to see harmonically related peaks in the ACF of rhythmically clear components. The ratio between every possible pairing of the ten candidate tempos was computed, leading to $\binom{10}{2}$, i.e. forty-five ratios per component. We then computed a histogram of these values in the range .45 to 2.05 with a bin width of .1, leading to 23 features. The targets were binary, representing whether the component estimate matched the ground truth. The tempo of an excerpt was calculated by choosing the tempo associated with the component that had the highest posterior probability, i.e. the greatest likelihood of its tempo matching the ground truth given the ACF features. This approach worked well for the MIREX06 data but less so for the IDM09 data (Table 2). At this point it is difficult to say whether the IDM09 performance was due to an insufficiently large training database to accurately learn the multivariate distribution or if more discriminative features must be found.

3.4.2 *Clustering*

Another method was attempted for determining the global tempo, based on the idea of taking a vote among the candidate tempos, possibly weighted by the corresponding normalized ACF values. The basic intuition was that the true tempo should appear more frequently than spurious estimates among the candidate tempos. To implement this, we first partitioned the candidate tempos into clusters using a hierarchical cluster tree. However, a simpler approach that did not attempt an exclusive partitioning performed better. In the latter approach, the candidate tempos for all components and their associated normalized ACF values were merged into a single matrix. For each tempo candidate, a score was determined by summing the ACF values for that tempo as well as for any tempos that were half or double, within a 5% tolerance. Of course such a method will often lead to ties, which we resolved by choosing the tempo closest to 120 BPM. Because we chose to measure accuracy accounting for half and double matches (see Section 4.2), this was not a major issue. The tempo candidate with the highest score was chosen as the global tempo estimate. To see if ACF weighting was important we also performed experiments ignoring ACF values and assigning scores by simply counting the number of elements in each cluster. However, ACF weighting consistently improved performance and was chosen as the default. Additionally, we experimented with multiplying each ACF value by the pulse-clarity estimate of the component based on the heuristics described above. This did not affect results and was therefore not included in the final version.

4. EVALUATION

4.1 Databases

Evaluation was performed on two databases. The main database consisted of twenty-seven 30-second excerpts chosen from the IDM/glitch genre of electronic music (IDM09), with an emphasis on tracks that we thought were rhythmically complex and layered. For each excerpt, two independent manual annotations were made. [1] For all excerpts the human annotators agreed, with the exception of a few half/double conflicts. In those cases, we randomly selected a single estimate. It should be noted that our accuracy measure allowed for half/double errors.

Additionally, the twenty publicly available MIREX06 training excerpts were used [2]. These consisted of a mix of genres and tempo ranges, and included annotation of two tempos representing the two highest peaks in a distribution of tempos calculated from listeners' tapping times. For our experiments we simply selected the ground truth tempo that was more commonly assigned.

4.2 Accuracy measure

We defined a match to be whenever the estimated tempo matched the annotated tempo, or double or half the annotated tempo, within a five percent tolerance window. Evaluation of tempo detection algorithms is somewhat dependent on the end-goal. We might reasonably hope that the tempo detection algorithm would correspond to judgments of human listeners. However, although there may be a fair degree of reliability between judgments for simple rhythms, there can be substantial disagreement about the appropriate metrical level or even the tempo for more rhythmically complex music. Moreover, more experienced listeners often tap at a lower metrical level (i.e. slower tempo) than novice listeners and in some cases novice listeners tap irregularly and are unable to clearly sense the tempo. Although this may be trivially true for music with no clear rhythm, this can also occur for music where there is a high degree of reliability for experienced listeners. For retrieval tasks, such as selecting tracks with similar tempos, it might be more appropriate to consider a match only when the metrical level of the main ground truth annotation is matched. On the other hand, for transcription or synchronization tasks it is appropriate to consider matches at different metrical levels. Because our emphasis here was on IDM, a genre that often contains metrical level ambiguity, we decided that this latter definition of accuracy made the most sense.

[1] The IDM09 database and the tempo annotations will be made publicly available online.

	Baseline (Ellis)	**Clustering**	**Change**
MIREX06	0.50	0.60	0.10
IDM09	0.26	0.48	0.22
combined	0.36	0.53	0.17

Table 1. The primary results are summarized here for each of the databases as well as for the combined set. The baseline is the Ellis algorithm run on the unseparated excerpts. Clustering refers to choosing the global estimate according to the procedure described in Section 3.4.2

4.3 Results

To get a sense of the upper-bound of performance for each track we checked to see if the true tempo was the primary tempo estimate for any of the components, and also whether the true tempo was present in any of the candidate tempos. Since subsequent steps attempt to filter these values, our pulse-clarity based technique can do no better than this first value, and the clustering method can do no better than the latter. The primary component tempo was correct for 70.4% of excerpts from IDM09 and 75% of MIREX06. A match was found in a candidate tempo of one of the components 96.3% and 85% of the time for IDM09 and MIREX06 respectively. Of course it should be noted while that we would expect this percentage to increase as the number of candidate tempos per component increases, the number of false positives will also tend to increase. Nevertheless these data suggest a high performance ceiling.

Table 1 summarizes the main the results, while Table 2 provides a more complete view of the performance of the different algorithms described in the paper. The first column in both tables is the baseline performance, given by running the Ellis algorithm on the unseparated excerpt using the definition of accuracy described above. Baseline accuracy for the MIREX06 data was 50% and 25.9% for IDM09. The substantially lower baseline accuracy for IDM09 reflects the rhythmic complexity of these excerpts. It can be seen that for the MIREX06, IDM09, and combined databases that the clustering algorithms improved accuracy by 10% (60% vs 50%) , 22.3% (48.2% vs. 25.9%) and 17% (53.2% vs. 36.2%), respectively. From the detailed results table we can see that the ML-based approach achieved a 10 percentage-point improvement on MIREX06 (60% vs 50%), and a 3.7 percentage-point increase on IDM09.

Clustering using multiple candidates per component, as well as the ML-based approach, had an accuracy of 60% for MIREX06, a 10% improvement on the baseline. In the case of IDM09 there was a substantial improvement of 22.3% (48.2% vs. 25.9%). However in this case the ML-based approach was only marginally better than baseline.

	Baseline	**Pulse**					**Clustering**	**ML**	
	Ellis	min	max	entropy	kurtosis	sparseness		all features	pulse-only
MIREX06	0.50	0.35	0.40	0.40	0.40	0.40	0.60	0.60	0.60
IDM09	0.26	0.15	0.33	0.26	0.26	0.30	0.48	0.27	0.26
combined	0.36	0.23	0.36	0.32	0.32	0.34	0.53	0.43	0.40

Table 2. Detailed accuracy results for each of the pulse-clarity measures described in Section 3.4.1 as well as for the machine learning algorithm also described in Section 3.4.1. For the ML algorithm results are shown for all features, as well as with only the original pulse heuristic features.

For both data sets using pulse-clarity alone did not improve results, with the exception of the max ACF (33.3% vs. 25.9%) and sparseness (29.6% vs 25.9%) features for IDM09.

5. DISCUSSION AND CONCLUSION

From these data it seems that the clustering-based approach is the superior method, particularly when compared to using a single component as the basis for the global estimate. It is possible, however, that this is simply an artifact of an inaccurate source separation step. Auditioning components reveals that many components are not true sources at all but parts of sources or several sources; source separation is still a delicate art. Nevertheless, many components do clearly correspond to parts and at times one can clearly hear time-keeping parts popping out. This noisiness probably accounts for the fact that the clustering approach, which retains more information about possible periodicities by retaining multiple tempo estimates for each component, is more robust. Although the current work did not bear out the ML-based approach, we believe that systematic incorporation of multidimensional rhythmic information will play an important part in future component-based tempo detection algorithms.

We have shown that for these data, using source separation in conjunction with clustering can substantially improve results, particularly for rhythmically complex and layered material. We have also explored a variety of techniques for implementing the core idea of using source decomposition to improve tempo estimation. In particular, we developed techniques for tempo estimation based on pulse clarity scoring of components and clustering of component tempo estimates. As source separation techniques improve, it should be possible to more closely mimic the rhythmic perception of humans, which in many cases is based on recognizing distinct parts that have a time-keeping function.

We expect that the approach described here will be most useful for layered, rhythmically complex music that tends to have simpler sub-parts. For simpler music, on the other hand, the less dramatic results are unlikely to justify the computational cost of source separation. We expect that this method will fail for music where the rhythm is emergent, i.e. only becomes apparent when several layers are played simultaneously.

6. FUTURE WORK

There are many possible extensions to this work. Thus far we have done little work to tune the source separation step. For example, what is the optimum number of components? It is likely that eventually this should be set adaptively based on the characteristics of the piece and the likely number of sources. These, however, remain unsolved problems, though the recent surge in research on single-channel source separation using PLCA and NMF is likely to dramatically improve our unmixing capabilities. Additionally, we intend to pursue the ML-based approach. In the long-run, it is likely that some combination of features can be found that will determine more reliably whether a component tempo estimate is the correct global estimate. And, as always, only with the expansion of the tempo database, and additional benchmarking against multiple systems, will we truly be able to assess the strengths and weaknesses of the techniques presented here.

7. REFERENCES

[1] Martin F. McKinney and Dirk Moelants? Ambiguity in tempo perception: What draws listeners to different metrical levels? *Music Perception*, 24(2):155–166, 2006.

[2] M. F. Mckinney, D. Moelants, M. E. P. Davies, and A. Klapuri. Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1):1–16, 2007.

[3] Daniel P. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[4] M. E. P. Davies and M. D. Plumbley. Beat tracking with a two state model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2005.

[5] Matthew Wright, Andrew Schloss, and George Tzanetakis. Analyzing afro-cuban rhythm using rotation-aware clave template. In *Proceedings of International Conference on Music Information Retrieval*, 2008.

[6] Simon Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 26(1):39–50, 2007.

[7] K. Seyerlehner, G. Widmer, and D. Schnitzer. From rhythm patterns to perceived tempo. In *Proceedings of International Conference on Music Information Retrieval*, 2007.

[8] Linxing Xiao, Aibo Tian, Wen Li, and Jie Zhou. Using a statistic model to capture the association between timbre and perceived tempo. In *Proceedings of International Conference on Music Information Retrieval*, 2008.

[9] Eric D. Scheirer. Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

[10] Masataka Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

[11] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. Supervised and semi-supervised separation of sounds from single-channel mixtures. In *Proceedings of the 7th International Conference on Independent Component Analysis and Signal Separation*, London, UK, September 07.

[12] Paris Smaragdis and Bhiksha Raj. Shift-invariant probabilistic latent component analysis. Technical report, 2007.

[13] Christian Uhle, Christian Dittmar, and Thoma Sporer. Extraction of drum tracks from polyphonic audio using independent subspace analysis. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, 2003.

[14] Marko Heln and Tuomas Virtanen. Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In *In: Proc. EUSIPCO2005. (2005*, 2005.

[15] Olivier Lartillot, Tuomas Eerola, Petri Toiviainen, and Jose Fornari. Multi-feature modeling of pulse clarity: Design, validation, and optimization. In *Proceedings of International Conference on Music Information Retrieval*, 2008.

[16] Patrik Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

A MID-LEVEL REPRESENTATION FOR CAPTURING DOMINANT TEMPO AND PULSE INFORMATION IN MUSIC RECORDINGS

Peter Grosche and Meinard Müller
Saarland University and MPI Informatik, Saarbrücken, Germany
{pgrosche,meinard}@mpi-inf.mpg.de

ABSTRACT

Automated beat tracking and tempo estimation from music recordings become challenging tasks in the case of non-percussive music with soft note onsets and time-varying tempo. In this paper, we introduce a novel mid-level representation which captures predominant local pulse information. To this end, we first derive a tempogram by performing a local spectral analysis on a previously extracted, possibly very noisy onset representation. From this, we derive for each time position the predominant tempo as well as a sinusoidal kernel that best explains the local periodic nature of the onset representation. Then, our main idea is to accumulate the local kernels over time yielding a single function that reveals the predominant local pulse (PLP). We show that this function constitutes a robust mid-level representation from which one can derive musically meaningful tempo and beat information for non-percussive music even in the presence of significant tempo fluctuations. Furthermore, our representation allows for incorporating prior knowledge on the expected tempo range to exhibit information on different pulse levels.

1. INTRODUCTION

The automated extraction of tempo and beat information from audio recordings has been a central task in music information retrieval. To accomplish this task, most approaches proceed in two steps. In the first step, positions of note onsets in the music signal are estimated. Here, one typically relies on the fact that note onsets often go along with a sudden change of the signal's energy and spectrum, which particularly holds for instruments such as the piano, guitar, or percussive instruments. This property allows for deriving so-called *novelty curves*, the peaks of which yield good indicators for note onset candidates [1, 15]. In the second step, the novelty curves are analyzed with respect to reoccurring or quasiperiodic patterns. Here, generally spoken, one can roughly distinguish between three different methods. The autocorrelation method allows for detecting periodic self-similarities by comparing a novelty curve with time-shifted copies [5, 12]. Another widely used method is based on a bank of comb filter resonators, where a novelty curve is compared with templates consisting of equally spaced spikes or pulses representing various frequencies and phases [10, 14]. Similarly, one can use a short-time Fourier transform to derive a time-frequency representation of the novelty curve [12]. Here, the novelty curve is compared with templates consisting of sinusoidal kernels each representing a specific frequency. Each of the methods reveals periodicity properties of the underlying novelty curve, from which one can estimate the tempo or beat structure. The intensities of the estimated periodicity, tempo, or beat properties typically change over time and are often visualized by means of spectrogram-like representations referred to as *tempogram* [3], *rhythmogram* [9], or *beat spectrogram* [6].

Relying on previously extracted note onset indicators, tempo and beat tracking tasks become much harder for non-percussive music, where one often has to deal with soft onsets or blurred note transitions. This results in rather noisy novelty curves, exhibiting many spurious peaks. As a consequence, more refined methods have to be used for computing the novelty curves, e. g., by analyzing the signal's spectral content, pitch, or phase [1, 8, 15]. Even more challenging becomes the detection of locally periodic patterns in the case that the music recording reveals significant tempo changes, which typically occur in expressive performances of classical music as a result of ritardandi, accelerandi, fermatas, and so on [4]. Finally, the extraction problem is complicated by the fact that the notions of tempo and beat are ill-defined and highly subjective due to the complex hierarchical structure of rhythm [2]. For example, there are various levels that are presumed to contribute to the human perception of tempo and beat. Most of the previous work focuses on determining musical pulses on the *tactus* (the foot tapping rate or beat [10]) or *measure* level, but only few approaches exist for analyzing the signal on the finer tatum level [13]. Here, a *tatum* or *temporal atom* refers to the fastest repetition rate of musically meaningful accents occurring in the signal.

In this paper, we introduce a novel mid-level representation that unfolds predominant local pulse (PLP) information from music signals even for non-percussive music with soft note onsets and changing tempo. Avoiding the explicit determination of note onsets, we derive a tempogram by performing a local spectral analysis on a possibly very noisy novelty curve. From this, we estimate for each time position a sinusoidal kernel that best explains the local periodic nature of the novelty curve. Since there may be a number of outliers among these kernels, one usually obtains unstable information when looking at these kernels in a one-by-one fashion. Our idea is to accumulate all these kernels over time to obtain a mid-level representation, which we refer to as *predominant local pulse* (PLP) curve. As it turns out, PLP curves are robust to outliers and reveal musically meaningful periodicity information even in the case of poor onset information. Note that it is not the objective of our mid-level representation to directly reveal musically meaningful high-level information such as tempo, beat level, or exact onset positions. Instead, our representation constitutes a flexible tool for revealing locally *predominant* information, which may then be used for tasks such as beat tracking, tempo and meter estimation, or music synchronization [10, 11, 14]. In particular, our representation allows for incorporating prior knowledge, e. g., on the expected tempo range, to exhibit information on different pulse levels. In the following sections, we give various examples to illustrate our concept.

The remainder of this paper is organized as follows. In Sect. 2, we review the concept of novelty curves while introducing a variant used in the subsequent sections. Sect. 3 constitutes the main contribution of this paper, where we introduce the tempogram and the PLP mid-level representation. Examples and experiments are described in Sect. 4 and prospects of future work are sketched in Sect. 5.

2. NOVELTY CURVE

Combining various ideas from [1, 10, 15], we now exemplarily describe an approach for computing novelty curves that indicate note onset candidates. Note that the particular design of the novelty curve is not in the focus of this paper. Our mid-level representation as introduced in Sect. 3 is designed to work even for noisy novelty curves with a poor pulse structure. Naturally, the overall result may be improved by employing more refined novelty curves as suggested in [15]. Given a music recording, a short-time Fourier transform is used to obtain a spectrogram $X = (X(k,t))_{k,t}$ with $k \in [1:K] := \{1,2,\dots,K\}$ and $t \in [1:T]$. Here, K denotes the number of Fourier coefficients, T denotes the number of frames, and $X(k,t)$ denotes the k^{th} Fourier coefficient for time frame t. In our implementation, each time parameter t corresponds to 23 milliseconds of the audio. Next, we apply a logarithm to the magnitude spectrogram $|X|$ of the signal yielding $Y := \log(1 + C \cdot |X|)$ for a suitable constant $C > 1$, see [10]. Such a compression step not only accounts for the logarithmic sensation of sound intensity but also allows for adjusting the dynamic range of the signal to enhance the clarity of weaker transients, especially in the high-frequency regions. In our experiments, we use the value $C = 1000$. To obtain a novelty curve, we basically compute the discrete derivative of the compressed spectrum Y. More precisely, we sum up only positive intensity changes to emphasize onsets while discarding offsets to obtain the novelty function $\Delta : [1 : T-1] \to \mathbb{R}$:

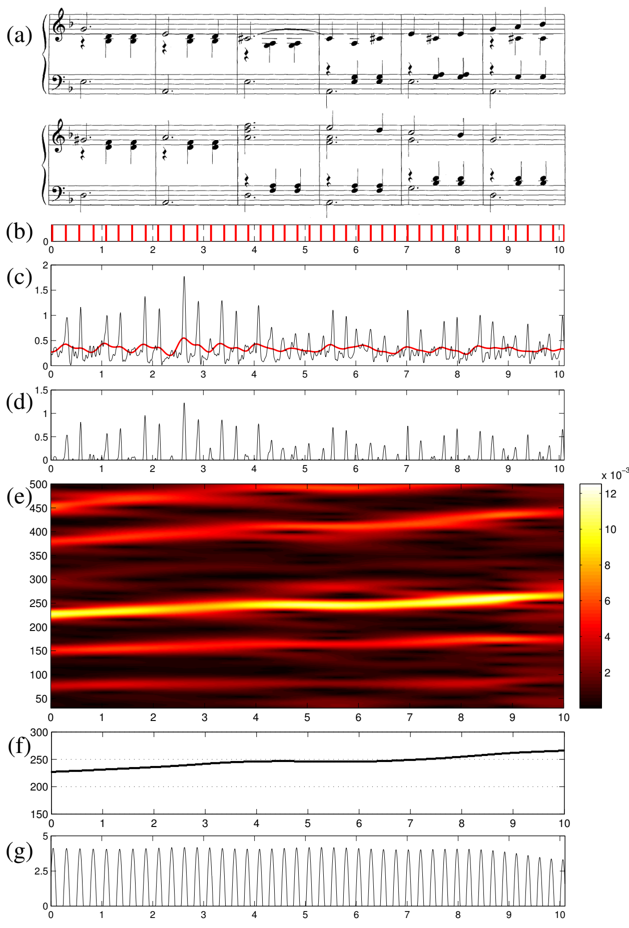


Figure 1: Excerpt of Shostakovich's second Waltz from Jazz Suite No. 2. The audio recording is a temporally warped orchestral version conducted by Yablonsky with a linear tempo increase (216 − 265 BPM). **(a)** Piano-reduced score of measures 13 − 24. **(b)** Ground truth onsets. **(c)** Novelty curve Δ with local mean. **(d)** Novelty curve $\bar{\Delta}$. **(e)** Magnitude tempogram $|\mathcal{T}|$ for KS = 4 sec. **(f)** Estimated tempo τ_t. **(g)** PLP curve Γ.

$$\Delta(t) := \textstyle\sum_{k=1}^{K} |Y(k,t+1) - Y(k,t)|_{\geq 0} \qquad (1)$$

for $t \in [1 : T-1]$, where $|x|_{\geq 0} := x$ for a non-negative real number x and $|x|_{\geq 0} := 0$ for a negative real number x. Fig. 1c shows the resulting curve for a music recording of an excerpt of Shostakovich's second Waltz from the Jazz Suite No. 2. To obtain our final novelty function $\bar{\Delta}$, we subtract the local average and only keep the positive part (half-wave rectification), see Fig. 1d. In our implementation, we actually use a higher-order smoothed differentiator. Furthermore, we process the spectrum in a bandwise fashion [14] using 5 bands. The resulting 5 novelty curves are weighted and summed up to yield the final novelty function. For details, we refer to the quoted literature.

3. TEMPOGRAM AND PLP CURVE

We now analyze the novelty curve with respect to local periodic patterns. Note that the novelty curve as introduced above typically reveals the note onset candidates in form of impulse-like spikes. Due to extraction errors and local tempo variations, the spikes may be noisy and irregularly spaced over time. Dealing with spiky novelty curves, autocorrelation methods [5] as well as comb filter techniques [14] encounter difficulties in capturing the quasiperiodic information. This is due to the fact that spiky structures are hard to identify by means of spiky analysis functions in the presence of irregularities. In such cases, smoothly spread analysis functions such as sinusoids are much better suited to detect locally distorted quasiperiodic patterns. Therefore, similar to [12], we use a short-time Fourier transform to analyze the novelty curves. More precisely, let $\bar{\Delta}$ be the novelty curve as described in Sect. 2. To avoid boundary problems, we assume that $\bar{\Delta}$ is defined on $\mathbb{Z}$ by setting $\bar{\Delta}(t) := 0$ for $t \in \mathbb{Z} \setminus [1 : T-1]$. Furthermore, we fix a window function $W : \mathbb{Z} \to \mathbb{R}$ centered at $t = 0$ with support $[-N : N]$. In our experiments, we use a Hann window of size $2N + 1$. Then, for a frequency parameter $\omega \in \mathbb{R}_{\geq 0}$, the complex Fourier coefficient $\mathcal{F}(t, \omega)$ is defined by

$$\mathcal{F}(t,\omega) = \textstyle\sum_{n\in\mathbb{Z}} \bar{\Delta}(n) \cdot W(n-t) \cdot \mathrm{e}^{-2\pi i\omega n} \quad . \tag{2}$$

Note that the frequency ω corresponds to the period $1/\omega$. In the context of beat tracking, we rather think of tempo measured in beats per minutes (BPM) than of frequency measured in Hertz (Hz). Therefore, we use a tempo parameter τ satisfying the equation $\tau = 60 \cdot \omega$.

Similar to a spectrogram, we define a *tempogram* which can be seen as a two-dimensional *time-pulse representation* indicating the strength of the local pulse over time. Here, intuitively, a *pulse* can be thought of a periodic sequence of accents, spikes or impulses. We specify the periodicity of a pulse in terms of a tempo value (in BPM). The semantic level of a pulse is not specified and may refer to the tatum, the tactus, or measure level. Now, let $\Theta \subset \mathbb{R}_{>0}$ be a finite set of tempo parameters. In our experiments, we mostly use the set $\Theta = [30 : 500]$, covering the (integer) musical tempi between 30 and 500 BPM. Here, the bounds are motivated by the assumption that only events showing a temporal separation between 120 milliseconds and 2 seconds contribute to the perception of rhythm [2]. Then, the tempogram is a function $\mathcal{T} : [1 : T] \times \Theta \to \mathbb{C}$ defined by

$$\mathcal{T}(t,\tau) = \mathcal{F}(t,\tau/60). \tag{3}$$

For an example, we refer to Fig. 1e, which shows the magnitude tempogram $|\mathcal{T}|$ for our Shostakovich example. Note that the complex-valued tempogram contains magnitude as well as phase information. We now make use of both, the magnitudes and the phases given by $\mathcal{T}$, to derive a mid-level representation that captures the *predominant local pulse* (PLP) of accents in the underlying music signal. Here, the term *predominant pulse* refers to the pulse that is most noticeable in the novelty curve in terms of intensity. Furthermore, our representation is *local* in the sense that it yields the predominant pulse for each time position, thus making local tempo information explicit, see also Fig. 1f. Also, the semantic level of the pulse may change over time, see Fig. 4a. This will be discussed in detail in Sect. 4.

To compute our mid-level representation, we determine for each time position $t \in [1 : T]$ the tempo parameter

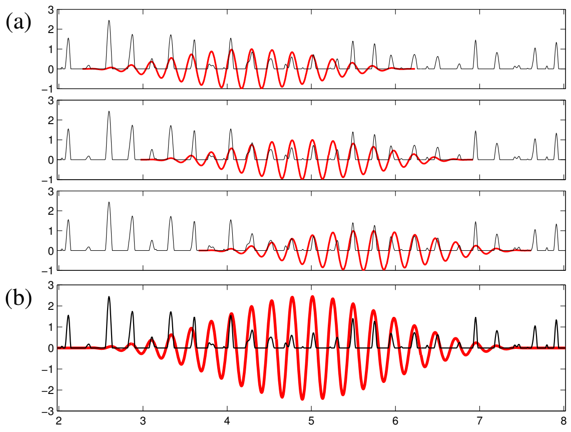


Figure 2: (a) Optimal sinusoidal kernel κ_t for various time parameters t using a kernel size of 4 seconds for the novelty curve shown in Fig. 1d. **(b)** Accumulation of all kernels. From this, the PLP curve Γ (see Fig. 1f) is obtained by half-wave rectification.

$\tau_t \in \Theta$ that maximizes the magnitude of $\mathcal{T}(t, \tau)$:

$$\tau_t := \mathrm{argmax}_{\tau\in\Theta} |\mathcal{T}(t,\tau)|. \tag{4}$$

The corresponding phase φ_t is defined by [11]:

$$\varphi_t := \frac{1}{2\pi} \arccos\left(\frac{\mathrm{Re}(\mathcal{T}(t,\tau_t))}{|\mathcal{T}(t,\tau_t)|}\right). \tag{5}$$

Using τ_t and φ_t, the optimal sinusoidal kernel $\kappa_t : \mathbb{Z} \to \mathbb{R}$ for $t \in [1 : T]$ is defined as the windowed sinusoid

$$\kappa_t(n) := W(n-t)\cos(2\pi(\tau_t/60 \cdot n - \varphi_t)) \tag{6}$$

for $n \in \mathbb{Z}$. Fig. 2a shows various optimal sinusoidal kernels for our Shostakovich example. Intuitively, the sinusoid κ_t best explains the local periodic nature of the novelty curve at time position t with respect to the set Θ. The period $60/\tau_t$ corresponds to the predominant periodicity of the novelty curve and the phase information φ_t takes care of accurately aligning the maxima of κ_t and the peaks of the novelty curve. The properties of the kernels κ_t depend not only on the quality of the novelty curve, but also on the window size $2N+1$ of W and the set of frequencies Θ. Increasing the parameter N yields more robust estimates for τ_t at the cost of temporal flexibility. In our experiments, we chose a window length of 4 to 12 seconds. In the following, this duration is referred to as *kernel size* (KS).

The estimation of optimal sinusoidal kernels for novelty curves with a strongly corrupted pulse structure is still problematic. This particularly holds in the case of small kernel sizes. To make the periodicity estimation more robust, our idea is to accumulate these kernels over all time positions to form a single function instead of looking at the kernels in a one-by-one fashion. More precisely, we define a function $\Gamma : [1 : T] \to \mathbb{R}_{\geq 0}$ as follows:

$$\Gamma(n) = \textstyle\sum_{t\in[1:T]} |\kappa_t(n)|_{\geq 0} \tag{7}$$

for $n \in [1 : T]$, see Fig. 2b. The resulting function is our mid-level representation referred to as *PLP curve*. Fig. 1g shows the PLP curve for our Shostakovich example. As it turns out, such PLP curves are robust to outliers and reveal musically meaningful periodicity information even when starting with relatively poor onset information.

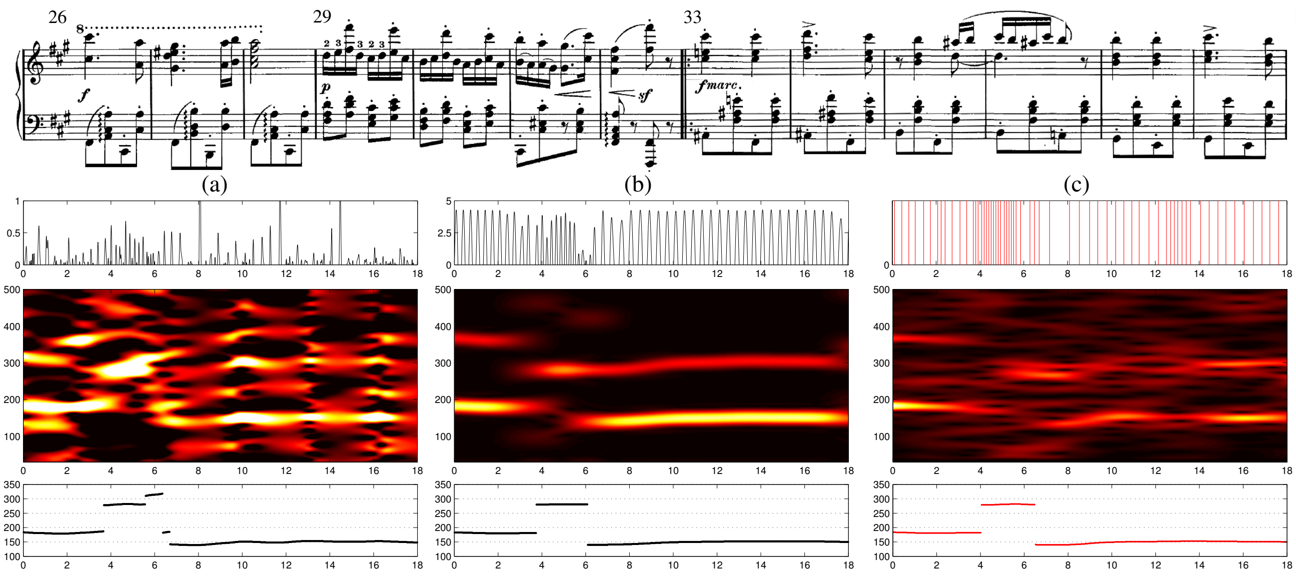


Figure 3: Excerpt of an orchestral version conducted by Ormandy of Brahms's Hungarian Dance No. 5. The score shows measures 26 to 38 in a piano reduced version. **(a)** Novelty curve $\bar{\Delta}$, tempogram derived from $\bar{\Delta}$, and estimated tempo. **(b)** PLP curve Γ, tempogram derived from Γ, and estimated tempo. **(c)** Ground-truth pulses, tempogram derived from these pulses, and estimated tempo. KS $= 4$ sec.

4. DISCUSSION AND EXPERIMENTS

In this section, we discuss various properties of our PLP concept and sketch a number of application scenarios by means of some representative real-world examples. We then give a quantitative evaluation on strongly distorted audio material to indicate the potential of PLP curves for accurately capturing local tempo information.

First, we continue the discussion of our Shostakovich example. Fig. 1a shows a piano-reduced score of the measures $13 - 24$. The audio recording (an orchestral version conducted by Yablonsky) has been temporally warped to possess a linearly increasing tempo starting with 216 BPM and ending at 265 BPM at the quarter note level. Firstly, note that the quarter note level has been identified to be the predominant pulse throughout the excerpt, see Fig. 1e. Based on this pulse level, the tempo has been correctly identified as indicated by Fig. 1f. Secondly, first beats in the 3/4 Waltz are played by non-percussive instruments leading to relatively soft and blurred onsets, whereas the second and third beats are played by percussive instruments. This results in some hardly visible peaks in the novelty curve shown in Fig. 1d. However, the beats on the quarter note level are perfectly disclosed by the PLP curve Γ shown in Fig. 1d. In this sense, a PLP curve can be regarded as a periodicity enhancement of the original novelty curve, indicating musically meaningful pulse onset positions. Here, the musical motivation is that the periodic structure of musical events plays a crucial role in the sensation of note changes. In particular, weak note onsets may only be perceptible within a rhythmic context.

As a second example, we consider Brahm's Hungarian Dance No. 5. Fig. 3 shows a piano reduced version of measures $26 - 38$, whereas the audio recording is an orchestral version conducted by Ormandi. This excerpt is very challenging because of several abrupt changes in tempo. Additionally, the novelty curve is rather noisy because of many weak note onsets played by strings. Fig. 3a shows the extracted novelty curve, the tempogram, and the extracted tempo. Despite of poor note onset information, the tempogram correctly captures the predominant eighth note pulse and the tempo for most time positions. A manual inspection reveals that the excerpt starts with a tempo of 180 BPM (measures $26 - 28$, seconds $0 - 4$), then abruptly changes to 280 BPM (measures $29 - 32$, seconds $4 - 6$), and continues with 150 BPM (measures $33 - 38$, seconds $6 - 18$). Due to the corrupted novelty curve and the rather diffuse tempogram, the extraction of the predominant sinusoidal kernels is problematic. However, accumulating all these kernels smooths out many of the extraction errors. The peaks of the resulting PLP curve Γ (Fig. 3b) correctly indicate the musically relevant eighth note pulse positions in the novelty curve. At this point, we emphasize that all of the sinusoidal kernels have the same unit amplitude independent of the onset strengths. Actually, the amplitude of Γ indicates the confidence in the periodicity estimation. Consistent kernel estimations produce constructive interferences in the accumulation resulting in high values of Γ. Contrary, outliers or inconsistencies in the kernel estimations cause destructive interferences in the accumulation resulting in lower values of Γ. This effect is visible in the PLP curve shown in Fig. 3b, where the amplitude decreases in the region of the sudden tempo change. As noted above, PLP curves can be regarded as a periodicity enhancement of the original novelty curve. Based on this observation, we compute a second tempogram now based on the PLP instead of the original novelty curve. Comparing the resulting tempogram (Fig. 3b) with the original tempogram (Fig. 3a), one can note a significant cleaning effect, where only the tempo information of the dominant pulse (and its harmonics) is maintained. This example shows how our PLP concept can be used in an iterative framework to stabilize local tempo estimations. Finally, Fig. 3c shows the manually generated ground truth onsets

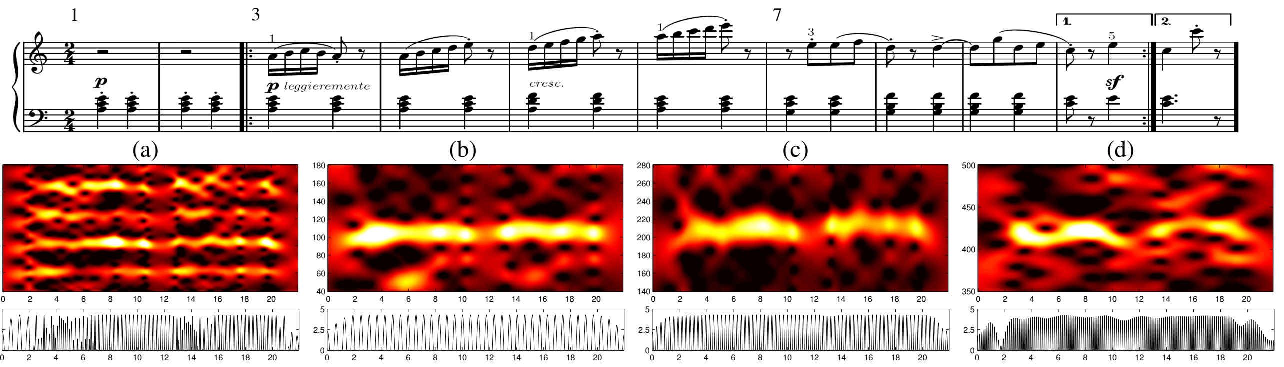


Figure 4: Beginning of the Piano Etude Op. 100 No. 2 by Burgmüller. Tempograms and PLP curves (KS $= 4$ sec) are shown for various sets Θ specifying the used tempo range (given in BPM). **(a)** $\Theta = [30 : 500]$ (full tempo range). **(b)** $\Theta = [40 : 180]$ (quarter note tempo range). **(c)** $\Theta = [140 : 280]$ (eighth note tempo range). **(d)** $\Theta = [350 : 500]$ (sixteenth note tempo range).

as well as the resulting tempogram (using the onsets as idealized novelty curve). Comparing the three tempograms of Fig. 3 again indicates the robustness of PLP curves to noisy input data and outliers.

In our final example, we look at the beginning of the Piano Etude Op. 100 No. 2 by Burgmüller, see Fig. 4. The audio recording includes the repetition and is played in a rather constant tempo. However, the predominant pulse level changes several times within the excerpt. The piece begins with four quarter note chords (measures $1-2$), then there are some dominating sixteenth note motives (measures $3-6$) followed by an eighth note pulse (measures $7-10$). The change of the predominant pulse level is captured by the PLP curve as shown by Fig. 4a. We now indicate how our PLP concept allows for incorporating prior knowledge on the expected tempo range to exhibit information on different pulse levels. Here, the idea is to constrain the set Θ of tempo parameters in the maximization (4) of Sect. 3. For example, using a constrained set $\Theta = [40 : 180]$ instead of the original set $\Theta = [30 : 500]$, one obtains the tempogram and PLP curve shown in Fig. 4b. In this case, the PLP curve correctly reveals the quarter note pulse positions as well as the quarter note tempo of 100 BPM. Similarly, using the set $\Theta = [140 : 280]$ ($\Theta = [350 : 500]$) reveals the eighth (sixteenth) note pulse positions and the corresponding tempos, see Fig. 4c (Fig. 4d). In other words, in the case there is a dominant pulse of (possibly varying) tempo within the specified tempo range Θ, the PLP curve yields a good pulse tracking on the corresponding pulse level.

In view of a quantitative evaluation of the PLP concept, we conducted a systematic experiment in the context of tempo estimation. To this end, we used a representative set of ten pieces from the RWC music database [7] consisting of five classical pieces, three jazz, and two popular pieces, see Table 1 (first column). The pieces have different instrumentations containing percussive as well as non-percussive passages of high rhythmic complexity. In this experiment, we investigated to what extend our PLP concept is capable of capturing local tempo deviations. Using the MIDI files supplied by [7], we manually determined the pulse level that dominates the piece. Then, for each MIDI file, we set the tempo to a constant value with regard to the respective dominant pulse level, [1] see Table 1 (second and third columns). The resulting MIDI files are referred to as *original MIDIs*. We then temporally distorted the MIDI files by simulating strong local tempo changes such as ritardandi, accelerandi, and fermatas. To this end, we divided the original MIDIs into 20-seconds segments and then alternately applied to each segment a continuous speed up or slow down (referred to as *warping procedure*) so that the resulting tempo of the dominant pulse fluctuates between $+30\%$ and -30% of the original tempo. The resulting MIDI files are referred to as *distorted MIDIs*. Finally, audio files were generated from the original and distorted MIDIs using a high-quality synthesizer.

To evaluate the tempo extraction capability of our PLP concept, we proceed as follows. Given an original MIDI, let τ denote the tempo and let Θ be the set of integer tempo parameters covering the tempo range of $\pm 40\%$ of the original tempo τ. This coarse tempo range reflects the prior knowledge of the respective pulse level (in this experiment, we do not want to deal with tempo octave confusions) and comprises the tempo values of the distorted MIDI. Based on Θ, we compute for each time position t the maximizing tempo parameter $\tau_t \in \Theta$ as defined in (4) of Sect. 3 for the original MIDI using various kernel sizes. We consider the local tempo estimate τ_t correct, if it falls within a 2% deviation of the original tempo τ. The left part of Table 1 shows the percentage of correctly estimated local tempi for each piece. Note that, even having a constant tempo, there are time positions with incorrect tempo estimates. Here, one reason is that for certain passages the pulse level or the onset information is not suited or simply not sufficient for yielding good local tempo estimations, e. g., caused by musical rests or local rhythmic offsets. For example, for the piece C022 (Brahms's Hungarian Dance No. 5), the tempo estimation is correct for 74.5% of the time parameters when using a kernel size (KS) of 4 sec. Assuming a constant tempo, it is not surprising that the tempo estimation stabilizes when using a longer kernel. In case of C022, the percentage increases to 85.4% for KS $= 12$ sec.

[1] In this experiment, we make the simplistic assumption that the predominant pulse does not change throughout the piece. Actually, this is not true for most pieces such as C003 (Beethoven's Fifth), C022 (Brahms's Hungarian Dance No. 5), or J001 (Nakamura's Jive).

			original MIDI				distorted MIDI			
Piece	Tempo	Level	4	6	8	12	4	6	8	12
C003	360	1/16	74.5	81.6	83.7	85.4	73.9	81.1	83.3	86.2
C015	320	1/16	71.4	78.5	82.5	89.2	61.8	67.3	71.2	76.0
C022	240	1/8	95.9	100.0	100.0	100.0	95.0	98.1	99.4	89.2
C025	240	1/16	99.6	100.0	100.0	100.0	99.6	100.0	100.0	96.2
C044	180	1/8	95.7	100.0	100.0	100.0	82.6	85.4	77.4	59.8
J001	300	1/16	43.1	54.0	60.6	67.4	37.8	48.4	52.7	52.7
J038	360	1/12	98.6	99.7	100.0	100.0	99.2	99.8	100.0	96.7
J041	315	1/12	97.4	98.4	99.2	99.7	95.8	96.6	97.1	95.5
P031	260	1/8	92.2	93.0	93.6	94.7	92.7	93.7	93.9	93.5
P093	180	1/8	97.4	100.0	100.0	100.0	96.4	100.0	100.0	100.0
average:			86.6	90.5	92.0	93.6	83.5	87.1	87.5	84.6
average (after iteration):			89.2	92.0	93.0	95.2	86.0	88.8	88.5	83.1

Table 1: Percentage of correctly estimated local tempi for the experiment based on original MIDI files (constant tempo) and distorted MIDI files for kernel sizes $\mathrm{KS} = 4, 6, 8, 12$ sec.

Anyway, the tempo estimates for the original MIDIs with constant tempo only serve as reference values for the second part of our experiment. Using the distorted MIDIs, we again compute the maximizing tempo parameter $\tau_t \in \Theta$ for each time position. Now, these values are compared to the time-dependent distorted tempo values that can be determined from the warping procedure. Analogous to the left part, the right part of Table 1 shows the percentage of correctly estimated local tempi for the distorted case. The crucial point is that even when using strongly distorted MIDIs, the quality of the tempo estimations only slightly decreases. For C022, the tempo estimation is correct for 73.9% of the time parameters when using a kernel size of 4 sec (compared to 74.5% in the original case). Averaging over all pieces, the percentage decreases from 86.6% (original MIDIs) to 83.5% (distorted MIDIs), for $\mathrm{KS} = 4$ sec. This clearly demonstrates that our concept allows for capturing even significant tempo changes. As mentioned above, using longer kernels naturally stabilizes the tempo estimation in the case of constant tempo. This, however, does not hold when having music with constantly changing tempo. For example, looking at the results for the distorted MIDI of C044 (Rimski-Korsakov, The Flight of the Bumble Bee), we can note a drop from 82.6% (4 sec kernel) to 59.8% (12 sec kernel).

Furthermore, we investigated the iterative approach already sketched for the Brahms example, see Fig 3b. Here, we use the PLP curve as basis for computing a second tempogram from which the tempo estimation is derived. As indicated by the last line of Table 1, this iteration indeed yields an improvement for the tempo estimation for the original as well as the distorted MIDI files. For example, in the distorted case with $\mathrm{KS} = 4$ sec the estimation rate raises from 83.5% (tempogram based on $\bar{\Delta}$) to 86.0% (tempogram based on Γ).

5. CONCLUSIONS

In this paper, we introduced a novel concept for extracting the predominant local pulse even from music with weak non-percussive note onsets and strongly fluctuating tempo. We indicated and discussed various application scenarios ranging from pulse tracking, periodicity enhancement of novelty curves, and tempo tracking, where our mid-level representation yields robust estimations. Furthermore, our representation allows for incorporating prior knowledge on the expected tempo range to adjust to different pulse levels. In the future, we will use our PLP concept for supporting higher-level music tasks such as music synchronization, tempo and meter estimation, onset detection, as well as rhythm-based audio segmentation. In particular the sketched iterative approach, as first experiments show, constitutes a powerful concept for such applications.

Acknowledgements: The research is funded by the *Cluster of Excellence on Multimodal Computing and Interaction* at Saarland University.

6. REFERENCES

[1] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler: "A Tutorial on Onset Detection in Music Signals," *IEEE Trans. on Speech and Audio Processing*, Vol. 13(5), 1035–1047, 2005.

[2] J. Bilmes: "A Model for Musical Rhythm," in *Proc. ICMC*, San Francisco, USA, 1992.

[3] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing: "On Tempo Tracking: Tempogram Representation and Kalman Filtering," *Journal of New Music Research*, Vol. 28(4), 259–273, 2001.

[4] S. Dixon: "Automatic Extraction of Tempo and Beat from Expressive Performances," *Journal of New Music Research*, Vol. 30(1), 39–58, 2001.

[5] D. P. W. Ellis: "Beat Tracking by Dynamic Programming," *Journal of New Music Research*, Vol. 36(1), 51–60, 2007.

[6] J. Foote and S. Uchihashi: "The Beat Spectrum: A New Approach to Rhythm Analysis," in *Proc. ICME*, Los Alamitos, USA, 2001.

[7] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka: "RWC Music Database: Popular, Classical and Jazz Music Databases," in *Proc. ISMIR*, Paris, France, 2002.

[8] A. Holzapfel and Y. Stylianou: "Beat Tracking Using Group Delay Based Onset Detection," in *Proc. ISMIR*, Philadelphia, USA, 2008.

[9] K. Jensen, J. Xu, and M. Zachariasen: "Rhythm-Based Segmentation of Popular Chinese Music," in *Proc. ISMIR*, London, UK, 2005.

[10] A. P. Klapuri, A. J. Eronen, and J. Astola: "Analysis of the meter of acoustic musical signals," *IEEE Trans. on Audio, Speech and Language Processing*, Vol. 14(1), 342–355, 2006.

[11] M. Müller: "Information Retrieval for Music and Motion," Springer, 2007.

[12] G. Peeters: "Template-based estimation of time-varying tempo," *EURASIP Journal on Advances in Signal Processing*, Vol. 2007, 158–171, 2007.

[13] J. Seppänen: "Tatum grid analysis of musical signals," in *Proc. IEEE WASPAA*, New Paltz, USA, 2001.

[14] E. D. Scheirer: "Tempo and beat analysis of acoustical musical signals," *Journal of the Acoustical Society of America*, Vol. 103(1), 588–601, 1998.

[15] R. Zhou, M. Mattavelli, and G. Zoia: "Music Onset Detection Based on Resonator Time Frequency Image," *IEEE Trans. on Audio, Speech, and Language Processing*, Vol. 16(8), 1685–1695, 2008.

ADAPTIVE MULTIMODAL EXPLORATION OF MUSIC COLLECTIONS

Dominik Lübbers[*†], **Matthias Jarke**[*]

[*]Informatik 5
RWTH Aachen University
Aachen, Germany

[†]Dept. Applied Information Technology
German University of Technology
Muscat, Sultanate of Oman

ABSTRACT

Discovering music that we like rarely happens as a result of a directed search. Except for the case where we have exact meta data at hand it is hard to articulate what song is attractive to us. Therefore it is essential to develop and evaluate systems that support guided exploratory browsing of the music space.

While a number of algorithms for organizing music collections according to a given similarity measure have been applied successfully, the generated structure is usually only presented visually and listening requires cumbersome skipping through the individual pieces.

To close this media gap we describe an immersive multimodal exploration environment which extends the presentation of a song collection in a video-game-like virtual 3-D landscape by carefully adjusted spatialized plackback of songs. The user can freely navigate through the virtual world guided by the acoustic clues surrounding him.

Observing his interaction with the environment the system furthermore learns the user's way of structuring his collection by adapting a weighted combination of a wide range of integrated content-based, meta-data-based and collaborative similarity measures.

Our evaluation proves the importance of auditory feedback for music exploration and shows that our system is capable of adjusting to different notions of similarity.

1. INTRODUCTION

Early work in Music Information Retrieval primarily concentrated on the development and evaluation of systems to support the identification of songs in a collection given a well-formulated query. According to Cunningham [1], this retrieval paradigm hardly matches the way we usually look for CDs in a music shop. Instead of searching for a dedicated album, participants in a user study showed a more exploratory browsing behaviour, which can be summarized as "shopping around" in contrast to "shopping for". This exploratory behaviour is however not completely chaotic: Users are reported to prefer some sort of structure in a music collection (e.g. a categorization according to genres), as long as this organization is intuitively understandable to them.

Even having a specific song in mind, we may find it difficult to articulate the information demand properly, if the name of the artist and the song title are unknown. Query by Example approaches like Query by Humming can only partly bridge this media discontinuity gap.

These reasons have led to an increased interest in exploration environments for music over the last years [2–4]. Most of theses approaches focus on visualizing a music collection with only standard playback functionality, which results in a media discontinuity problem in the opposite direction and does not exploit the human's capability to orientate himself in a complex environment of simultaneously playing spatialized sounds.

Therefore, we developed and evaluated an exploration prototype that provides an immersive virtual environment, in which the user can navigate guided by acoustic clues from song playbacks surrounding him.

As in previous approaches, the placement of pieces in this environment is based on a similarity function. The notion of similarity is known to be multifaceted, highly user-dependent and also influenced by the song collection at hand. We therefore allow the user to move songs in the environment as well as raise or lower borders between song clusters. Observing the user's interaction with the landscape we furthermore adapt a linear combination of content-based and collaborative similarity measures to best fit his understanding of similarity.

To our knowledge our prototype is therewith the first multimodal exploration environment which integrates an immersive virtual 3D-landscape of clustered songs with spatialized audio playback respecting humans' auditory perception limitations and furthermore adapts to the user's strategy of organizing his collection by learning the weights of a wide range of different music similarity measures.

In the next section we give a brief overview of related work on exploration environments for music collections. Then we list the integrated base similarity functions used as components of a user-adaptive similarity measure. The following section describes our exploration environment in detail. We continue with the explanation of the similarity measure adaption process, which is followed by results from an qualitative and quantitative evaluation of our system and concluded by some final remarks and an outlook on further research.

2. RELATED WORK

Over the last years, a number of proposals for visualizing music collections have been made.

Pampalk et al. reduce the audio signal of a song to the median of frame-based *Fluctuation Patterns*, which model loudness periodicities in different frequency bands of the signal [5]. These features are used to train a small-size rectangular Self-Organizing Map (SOM). They interpret the estimated song densities of the cells as the height profile of a map. Applying an appropriate color map generates an intuitive visualization of similar song clusters positioned on “Islands of Music” separated by blue water.

The approach by Moerchen et al. is conceptually similar [7]. Their work mainly differs in the use of a compact but highly discriminative content-based feature set and the distribution of the collection items over a larger, *emergent* SOM. Still, Moerchen et al. do not integrate any kind of acoustic presentation besides a standard playback functionality of a selected song.

In contrast to this, Hamanaka and Lee focus on audio-only exploration of a given song set [8] without the need for a display. By spatializing songs according to different pre-defined allocation schemes, a user wearing a special headphone has the impression of being surrounded by simultaneously playing sound sources from different directions. Sensors mounted on the headphone detect the movement of the head and allow the user to change focus to songs he perceives from left or right. This interaction promotes the impression of an immersive virtual environment. Additionally, he can narrow the range of sounding sources by putting his hands behind the ear and thereby fading out songs that are not placed directly in front of him. This resembles the *focus of perception* mechanism we introduced in [9] and supports humans’ ability to concentrate on specific sounds in a complex mixture, known as the cocktail party effect.

To our knowledge, the approach by Knees et al. is the first one that combines SOM-based structuring of music collections with three-dimensional visualization and auralization to an immersive multimodal exploration environment [10]. Their work extends the Island of Music metaphor by using the smoothed height profile of SOM cells to generate a virtual 3D-landscape that the user can intuitively explore. Songs in the neighborhood of the current position sound from the respective direction. Knees and et. do not implement a focus mechanism, which seems to be critized by one of the comments in their user study, that asks for a larger landscape especially when facing crowded regions.

All of the above exploration environments quantify similarity between songs according to a fixed measure, that is supposed to reflect a generic similarity understanding by the average user. Recognizing the diversity of the similarity notion, Pampalk et al. align three SOMs representing timbral, rhythmic and metadata-provided aspects and allow the user to gradually change between these presentations [11].

Baumann linearly combines content-based similarity with cultural similarity and text-based similarity of the lyrics [12]. The user can adjust the weights of this trimodal measure by moving a virtual joystick into the direction of the favoured similarity aspect.

Instead of forcing the user to learn the semantics of different similarity measures and to decide for the individual importance of them, we propose a machine learning strategy that induces the weights of each component from the user’s interaction with our immersive multimodal exploration environment.

Figure 1 depicts the stages involved in generating and adapting this environment. The following sections describe these phases in detail.

3. SIMILARITY

To model a user’s notion of similarity as precisely as possible, it is mandatory to combine a number of base similarity measures covering different musical aspects and let the system adapt their weights.

We therefore decided to integrate timbral similarity measures (based on stochastic MFCC models as proposed by Logan/Salomon [13] and Aucouturier/Pachet [14] or the 20-feature set proposed by Moerchen et al. [7]) as well as more rhythm-based measures (Fluctuation Patters and Periodicity Histograms [11]). Furthermore we calculate the average and variance of 15 frame-based audio features as provided by the MIRtoolbox library [15]. These features are of varying complexity, ranging from simple RMS values over spectral centroids and roughness measures to key clarity and tempo estimates.

Additionally, we use ID3 metadata to make contextual information available. In particular, we calculate the time period between the publication of two pieces. To group songs by the same artist even in the commonly encountered presence of small typing errors, we furthermore calculate the edit distance between ID3 artist strings.

These similarity measures are complemented by three collaborative approaches based on direct last.fm similarity links, last.fm top tags and co-occurrence on playlists published on Art of the Mix.

last.fm offers the compilation of recommended tracks to a personalized music stream based on the user’s profile. This requires the establishment of similarity links between tracks. last.fm allows access to this information by a web service that returns a number of similar tracks to a given song. Each of these similar tracks is assigned a match value that quantifies the degree of similarity scaled to 100 for the most similar song. We consider the presence of a direct similarity link as a strong indication of similarity, even if the match value might be low. Therefore we transform the match score with a compressed exponential function to a distance value. Averaging the mutual distances to guarantee symmetry leads to the following calculation for two tracks tr_i and tr_j:

$$d_{DL}(tr_i, tr_j) = 0.5(e^{-c_{DL}\cdot\frac{\mathrm{ms}_{tr_i}(tr_j)}{100}} + e^{-c_{DL}\cdot\frac{\mathrm{ms}_{tr_j}(tr_i)}{100}}),$$

where $\mathrm{ms}_{tr_i}(tr_j)$ denotes the match score of track tr_j in

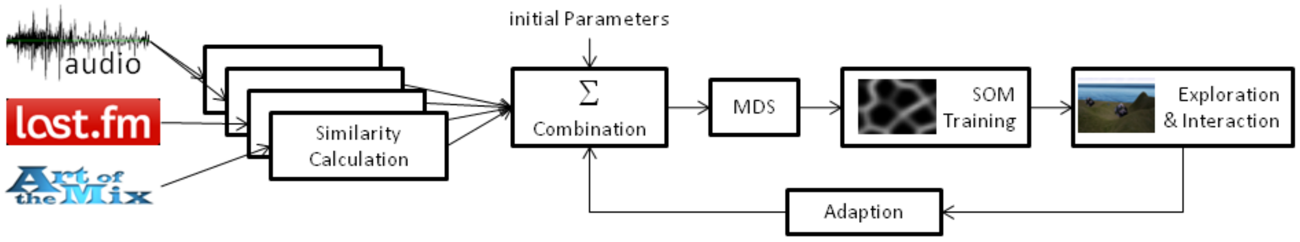


Figure 1. Data transformation stages for building and adapting the exploration environment.

the list of similar tracks to track tr_i if present and 0 otherwise. We empirically chose a value of $c_{DL} = 5$ for the compression factor.

While a track-based similarity measure is very specific, it may be difficult to find enough collaborative data for a reliable estimate. We therefore calculate the distance between the artists of two songs in the same way as above and combine it linearly with the track-based measure weighting the more precise track distance double.

Instead of assigning fixed genre categories to songs, last.fm allows users to tag tracks with arbitrary keywords favouring the emergence of a folksonomy over the definition of a static genre hierarchy. The comparison of these song descriptions is another valueable source of similarity. Retrieving the top tags for a song results in a list ranked according to the frequency used to annotate the song. Unfortunately, last.fm's `count` attribute does not quantify this per-track frequency but the overall popularity of a tag. Lacking further information, we consider the tags as natural language terms in a text about the track. This allows us to assume that the tag distribution follows Zipf's law and approximate tag frequencies by a Zipfian density function. Likewise, we do not have access to the ratio of tracks that are tagged with a certain keyword and have to estimate the inverse document frequency on the basis of the overall popularity of a tag.

These approximations can be used to weight the importance of a tag for a song according to the standard tf·idf scheme. The track-based top tag-similarity between two songs can finally be calculated as the cosine between aligned weight vectors. For the same reasons as above we also calculate top tag-similarity on artist level.

The last distance calculation we derive from collborative data is based on co-occurrences of songs on playlists (called *mixes*) that are published by users on the Art of the Mix portal[1]. We follow the assumption that two pieces occuring on the same list fit the same taste and can be considered as similar. To quantify this notion we use a simple overlap distance measure:

$$d_{AotM}(s_i, s_j) = 1 - \frac{|M(s_i) \cap M(s_j)|}{\min\{|M(s_i)|, |M(s_j)|\}},$$

where $M(s_i)$ denotes the set of mixes that contain song s_i. As done for the other collaborative measures, we combine this distance with its artist-based variant.

[1] www.artofthemix.org

Since some of the presented measures (like Logan/Salomon) are based on pairwise comparisons between songs, the composed distance values are arranged in a (symmetric) matrix. As the SOM training algorithm requires the representation of each item as a feature vector in Euclidean space, we apply multi-dimensional scaling (MDS) to find d-dimensional coordinates for each song such that the Euclidean distance between two song vectors resembles the distance matrix value (see figure 1). In our experiments we chose a value of $d = 20$, which matches the dimensionality of the data space used for the MusicMiner-SOM [7].

4. EXPLORATION ENVIRONMENT

4.1 SOM Training

As humans are used to intuitively estimate distances between points on a 2-dimensional plane, dimensionality reduction techniques that map high-dimensional data to low-dimensional representations while preserving distances as much as possible are popular data visualization strategies.

One of these techniques is the Self-Organizing Map (SOM) proposed by Kohonen,which arranges disjoint cells $\{y_i\}$ on a usually rectangular grid. Each y_i is associated with a *model vector* m_i from data space. We initialize the model vectors with linear combinations of the first two principial components of the song feature values according to the grid coordinate of their cell.

In each iteration t we randomly choose a data vector x_j and identify the cell bm with the closest model vector to x_j, i.e. that minimizes $||x_j - m_{bm}||$. The model vectors of this *Best Matching Unit* bm and its neighborhood are moved towards x_j according to the following equation:

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{i,bm}(t)[x_j - m_i(t)],$$

where $\alpha(t)$ denotes the learning rate at time t and $h_{i,bm}(t)$ quantifies the distance between x_i and bm, usually by some Gaussian function centered around bm. Since $\alpha(t)$ and $h_{i,bm}(t)$ decrease with each iteration and thereby weaken the adaption process with time, the map converges to a configuration where the Best Matching Units of similar data points are located close to each other.

In contrast to clustering algorithms like k-Means, a SOM is also capable of adaquately representing data points that lie in between clusters and reveals the macro-structure of the data space by retaining similarity relationships between clusters themselves.

The distribution of model vectors over the grid that is generated on the fly during the adaption contains additional valueable information about the similarity space: This information can be visualized by the U-Matrix [6], which assigns to each cell the average distance of its model vector to the model vectors of its neighbors. High values thereby indicate clear borders separating coherent regions of similar objects on the map.

4.2 Visual Presentation

Displaying these U-Matrix values and placing songs at their Best Matching Unit already yields an untuitively understandable visualization of the collection. But if we interpret the U-Matrix values as heights of a landscape we can generate a 3-D terrain and allow the user to leave his bird's eye-view on the music space in favor of becoming part in an immersive virtual environment.

Our prototype is based on Microsoft's game framework XNA 3.0 to realize efficient state-of-the-art visualization. We generate a high-resolution terrain mesh by bilinear interpolation of the U-Matrix height values and use a customized shader for visualization which appropriately combines sand, grass, mountain and snow textures according to the height.

By default, songs are visualized as small cubes textured with the cover image of their album if available. The position of a cube is mainly determined by the coordinates of the song's Best Matching Unit. To avoid clumping at grid points, we slightly move it towards the location in the immediate neighborhood where the bilinearly interpolated model vector is closest to the feature vector of the song.

The user can freely run through the landscape, move his head around and lift up to get an overview of the scenery. Figure 2 shows a screenshot of our environment taken from different elevation levels. The user is standing in (or over) a valley that contains songs from the German hiphop group *Fanta4*. As can be seen, these songs are clearly separated from different pieces by surrounding hills.

4.3 Auditory Presentation

Music is described best by music. This asks for the presence of acoustic information as guidance in the exploration process: Since humans are used to differentiate well between sound sources from different directions, exposing the user to simultaneously playing spatialized music facilitates efficient and well-informed navigation through the collection.

Fortunately, the above virtual environment can be extended naturally to incorporate the presentation of acoustic information, simply by associating each cube with a sound source playing the song from its location in the landscape.

As described in [9] the unrestricted simultaneous playback of many songs quickly overwhelms the user's auditory system and confuses more than it helps. Following ideas from visual perception we therefore define the point the user is currently looking at as the *Focus of Perception* and attenuate the volume of songs the more they deviate from the view direction. To allow for broad "listening around" as well as for clearly focussing on the sound in front we model the strength of this attenuation by a Gaussian function with user-adjustable variance. More precisely, the gain factor due to perception focussing is given as follows:

$$g_{PF}(\varphi) = e^{\frac{-\varphi^2}{\sigma^2}},$$

where φ denotes the angle between the direction to the song and the view direction and $\sigma^2 = \frac{-AoP}{\ln(gAoP)}$ is the variance for the user-adjustable *Angle of Perception* AoP, such that $g_{PF}(AoP) = gAoP$.

We describe the influence of a song's distance to its gain by an inverse distance model:

$$g_{Dist}(d) = \min(1, \frac{decSpeed}{d} - \frac{decSpeed}{minDist} + 1),$$

where d is the distance to the song, $decSpeed$ parameterizes the speed of gain decrease per distance unit and $minDist$ denotes the distance at which no attenuation takes place.

To summarize, the overall gain for a song s at location $\vec{p_s}$ assuming a listener's position $\vec{p}$ and a view direction $\vec{vd}$ is the product of its gain influences:

$$g(s,\vec{p}) = g_{Dist}(||\vec{p} - \vec{p_s}||) \cdot g_{PF}(\angle(\vec{vd}, \vec{p_s} - \vec{p})) \cdot g_{muff}(s,\vec{p}).$$

$g_{muff}(s,\vec{p})$ reduces the gain for a song, that is hidden behind a rise of the terrain. To generate the impression of a muffled sound this is complemented by a highcut filter.

Still, the simultaneous playback of all songs in the collection is too demanding (as well from an computational as from a perceptual point of view). We tested several song selection criteria and decided for a simple approach that guarantees perceptual separability and does not change the set of active sources when the user rotates his head: First, all songs in the neighborhood of the listener's position are sorted according to their gain factor. Following this order we then successively activate songs as long as they do not sound from a direction similar to the one of already playing songs.

4.4 User Interaction

A standard xBox 360 game controller can be used to navigate in the virtual world. Besides this, the user can customize the landscape as follows:

- Songs that seem to be misplaced in the opinion of the user can be moved easily.
- Alternatively, songs can be released to let the system find a new location during the next adaption cycle.
- Landmarks can be placed to emphasize and easily recover locations on the terrain. The user can choose between different sign types that can be labeled or textured with arbitrary images. Figure 2 shows two triangular landmarks.
- The terrain can be altered by raising or lowering its height at the position the user points to. This allows the formation of new separating hills between song

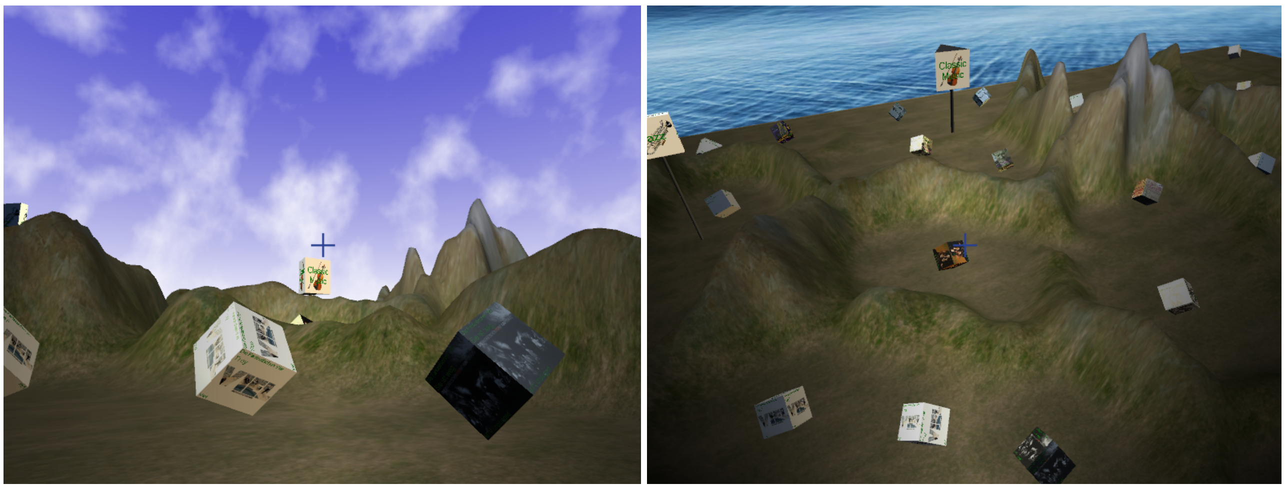


Figure 2. Screenshot of the exploration prototype: Views from different elevation levels

clusters that are perceived as different or the removement of borders between areas that the user judges similar.

5. USER ADAPTATION

As Cunningham observes, music listeners organize their personal collections according to different criteria. Some may sort their albums by the year of publication, some may cluster their music by genre, for others rhythmic content plays a dominant role. An exploration environment should be flexible enough to follow the user's organization strategy.

Instead of asking the user to articulate his structure principles explicitly we decided to learn his similarity notion from his interaction with the environment. Adapting the weights in the linear similarity model properly allows us to reposition songs that have been released by the user or to place new songs that are added to the collection.

The user can build or destroy separating hills between songs. To account for these terrain changes, we numerically integrate over the height profile (h_n) between the locations p_i and p_j and compare this to the situation before the change (h_o):

$$td_t(s_i, s_j) = \frac{1}{||\vec{p_i} - \vec{p_j}||} \cdot \left(\int_{\vec{p_i}}^{\vec{p_j}} (h_n(\vec{p}) - h_o(\vec{p}))d\vec{p} \right)$$

The combination of td_t with the Euclidean distance between the (interpolated) model vectors of two songs' locations on the map is stored in a *target distance* matrix. Each entry of this matrix is considered a training case for a linear regression learner, that adapts the weighting of the implemented base distances to approximate the target distance.

As figure 1 shows, the updated similarity model is subsequently used to rebuild the environment by the same process chain as before. To avoid drastic changes in the exploration space that potentially disorientate the user, we initialize the vector representation of each fixed song by its old value before the MDS optimization starts. Likewise, we guarantee topographic stability of the SOM by constantly taking a song's old location as its Best Matching Unit during training.

6. EVALUATION

We conducted a user study with nine participants showing different music taste, listening habits and experience with computer games.

In a first experiment we aurally presented an unknown song and measured the time needed to find it in a collection of about 100 tracks, that were randomly distributed over a flat exploration plane. Cover and metadata of the wanted song were not given to the user. We repeated the task for a different song and collection, this time providing the SOM-based organization. To eliminate effects from the choice of song and collection, we shuffled task and data for different participants.

A similar pair of experiments investigates the importance of spatialized acoustic clues when navigating through the exploration space by comparing this feature with standard media player functionality which requires to explicitly start and stop the playback of a song.

We found reductions in search time of 61% and 58% on average, which demonstrate, how significantly the user benefits from a well structured collection and acoustic clues during the exploration.

The last group of experiments evaluate the adaptation capabilities of our system to a user's notion of similarity: We asked the participants to customize a collection of 20 tracks by moving the songs and changing the terrain structure. Similar to a leave-one-out evaluation we successively release one song and compare its original position to the location that would be assigned by the SOM training. This *placement error* is calculated with and without executing the adaptation procedure. The first data series in figure 3 shows the relative difference between these two runs and reveals, that generally the adaptation works well, but reduces the placement error only slightly. One reason for that might be the that the initial similarity measure already captured the user's notion rather well.

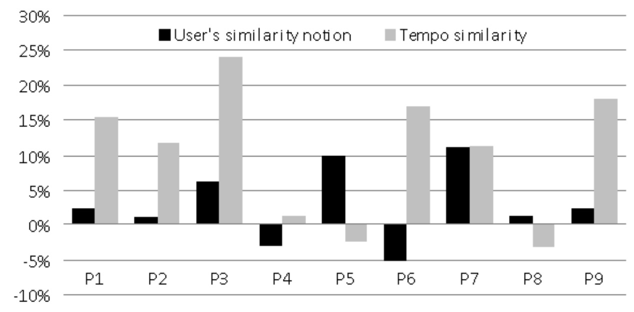


Figure 3. Relative reduction of placement error by adaptation to users' similarity notion

Therefore, we asked the users to organize the collection according to tempo independent of the genre and again computed the relative improvement in placement error. As can be seen from the second data series in figure 3 our system also adapts generally well to this more drastic change in similarity notion.

After these quantitative experiments we handed out an extensive questionnaire for qualitative evaluation. Study participants consistently judged the usability of the system as high but repeatedly proposed the addition of a 2-D map view to the environment to avoid disorientation in the exploration landscape.

7. CONCLUSION AND OUTLOOK

We presented an immersive multimodal exploration environment, that visualizes and auralizes music collections organized according to an user-adaptable similarity model, which combines content-based, meta-data-based and collaborative similarity measures. While our evaluation shows the general tractability of our approach, some open questions for further research remain:

So far, we did not focus on scalability issues in our work. We found, that collections of up to 400 songs are still manageable in our environment. Larger numbers of tracks require some form of hierarchical organization to remain accessible. We may can adopt ideas from [16] to extend the SOM-based placement algorithm.

Since they can model more complex relationships than vector-based distances, we deliberately integrated similarity measures that require pairwise computation of distances. Because of this the complexity of the similarity calculation stage is in $\mathcal{O}(n^2)$. To alleviate the scalability problems arising from this, one could restrict the calculation to some *anchor songs*. The MDS stage is already prepared to handle sparse distance matrices.

As shown by the evaluation, the adaption to the user's similarity notion still has room for improvement. A reason for this might be that a linear model is not expressive enough to capture the intended combination of base similarities. More complex models should therefore be investigated in future research.

8. REFERENCES

[1] S. Cunningham, N. Reeves, and M. Britland: "An ethnographic study of music information seeking: implications for the design of a music digital library" *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 5–16, 2003.

[2] M. Goto and T. Goto: "Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces" *Proc. ISMIR*, 2005.

[3] R. van Gulik, F. Vignoli, and H. van de Wetering: "Mapping Music in the Palm of Your Hand, Explore and Discover Your Collection" *Proc. ISMIR*, 2004.

[4] E. Pampalk and M. Goto: "Musicsun: A New Approach to Artist Recommendation" *Proc. ISMIR*, 2007.

[5] E. Pampalk, A. Rauber, and D. Merkl: "Content-based Organization and Visualization of Music Archives" *Proceedings ACM Multimedia*, 2002.

[6] A. Ultsch: "Self-Organizing Neural Networks for Visualization and Classification" *Proc. GfKI*, 1992

[7] F. Mörchen, A. Ultsch, M. Nöcker, and C. Stamm: "Databionic Visualization of Music Collections According to Perceptual Distance" *Proc. ISMIR*, 2005.

[8] M. Hamanaka and S. Lee: "Music Scope Headphones: Natural User Interface for Selection of Music" *Proc. ISMIR*, 2006.

[9] D. Lübbers: "soniXplorer: Combining Visualization and Auralization for Content-Based Exploration of Music Collections" *Proc. ISMIR*, 2005.

[10] P. Knees, M. Schedl, T. Pohle, and G. Widmer: "Exploring Music Collections in Virtual Landscapes" *IEEE MultiMedia*, Vol. 14, No. 3, 2007.

[11] E. Pampalk, S. Dixon, and G. Widmer: "Exploring Music Collections by Browsing Different Views" *Proc. ISMIR*, 2003.

[12] S. Baumann, T. Pohle, and S. Vembu: "Towards a Socio-cultural Compatibility of MIR Systems" *Proc. ISMIR*, 2005.

[13] B. Logan and A. Salomon: "A Music Similarity Function Based on Signal Analysis" *Proceedings ICME*, 2001.

[14] J.-J. Aucouturier and F. Pachet: "Finding Songs That Sound the Same" *IEEE Workshop on Model based Processing and Coding of Audio* 2002.

[15] O. Lartillot and P. Toiviainen: "A Matlab Toolbox for Musical Feature Extraction From Audio" *Proc. DAFx-07*, 2007.

[16] A. Rauber, E. Pampalk, and D. Merkl: "Using psycho-Acousic Models and Self-Organizing Maps to Create a Hierarchical Structuring of Music by Sound Similarity" *Proc. ISMIR 2002*

SINGING PITCH EXTRACTION FROM MONAURAL POLYPHONIC SONGS BY CONTEXTUAL AUDIO MODELING AND SINGING HARMONIC ENHANCEMENT

Chao-Ling Hsu[1] **Liang-Yu Chen**[1] **Jyh-Shing Roger Jang**[1] **Hsing-Ji Li**[2]

[1]MediaTek-NTHU Joint Lab
Dept. of Computer Science,
National Tsing Hua Univ., Taiwan
{leon, davidson833, jang} @mirlab.org

[2]Innovative Digitech-Enabled Applications & Services Institute (IDEAS)
Institute for Information Industry, Taiwan
lihsingji@iii.org.tw

ABSTRACT

This paper proposes a novel approach to extract the pitches of singing voices from monaural polyphonic songs. The hidden Markov model (HMM) is adopted to model the transition between adjacent singing pitches in time, and the relationships between melody and its chord, which is implicitly represented by features extracted from the spectrum. Moreover, another set of features which represents the energy distribution of the enhanced singing harmonic structure is proposed by applying a normalized sub-harmonic summation technique. By using these two feature sets with complementary characteristics, a 2-stream HMM is constructed for singing pitch extraction. Quantitative evaluation shows that the proposed system outperforms the compared approaches for singing pitch extraction from polyphonic songs.

1. INTRODUCTION

Melody, usually represented by the pitch contour of a lead vocal in a song, is considered as one of the most important elements of a song. It is broadly used in various applications, including singing voice separation, music retrieval, and musical genre classification.

Since Goto [1] proposed the first melody extraction system by employing a parametric model trained by statistical methods in 1999, more and more work has been proposed in the literature [2-8]. Because harmonic structures of a singing voice are very noticeable in spectrogram even in a polyphonic song, they are commonly used as cues for extracting the singing melody [1][4-6]. However, they neglect the contextual information of music.

Ryynänen et al. [8] used both acoustic and musicological models to generate hidden Markov models (HMMs) for a singing melody transcription system. The musicological models determine the transition probabilities between the adjacent notes. Li et al. [2] also utilized an HMM where the transition probability was estimated from the labeled training data. However, they only considered the transition between adjacent notes; the concurrent pitches generated by other musical instruments, such as chords, were not considered.

While the concurrent pitches are usually the obstacles in singing pitch extraction, we try to utilize them as the cues to extract the melody. Generally speaking, melody is composed of a series of notes and is decorated by chords. The chords here represent the concurrent pitches accompanying the melody. These notes and chords progress according to some underlying music rules to make the song euphonious. Therefore, we use an HMM to learn these rules from actual song data by observing their spectrograms. Note that we do not identify the chords explicitly. Instead, we use the energy distribution of each semitone to train the contextual audio model. In addition, in order to utilize the harmonics information as cues to extract the singing pitches, we also model the energy distribution of harmonics by using the proposed normalized sub-harmonic summation (NSHS) to enhance the harmonic structures of the sound sources especially for those of the singing voices. By synergizing these two techniques, the accuracy of singing pitch extraction is improved significantly.

The rest of this paper is organized as follows. Section 2 describes the proposed system in detail. The experimental results are presented in section 3, and section 4 concludes this work with possible future directions.

2. SYSTEM DESCRIPTION

Fig. 1 shows the overview of the proposed system. Two streams of features are extracted from the spectrogram and the NSHS map, respectively, of the input polyphonic song. A 2-stream HMM is then employed to decode the input songs into the most likely unbroken pitch vectors. On the other hand, the MFCCs (Mel-frequency cepstral coefficients) are extracted to perform the voiced/non-voiced detection. Lastly, the singing pitch vectors are produced by integrating the results of these two processes. The following subsections explain these blocks in detail.

2.1 Features Extraction from a Spectrum

This block extracts two types of features, including MFCCs and ESI (Energy at Semitones of Interests).

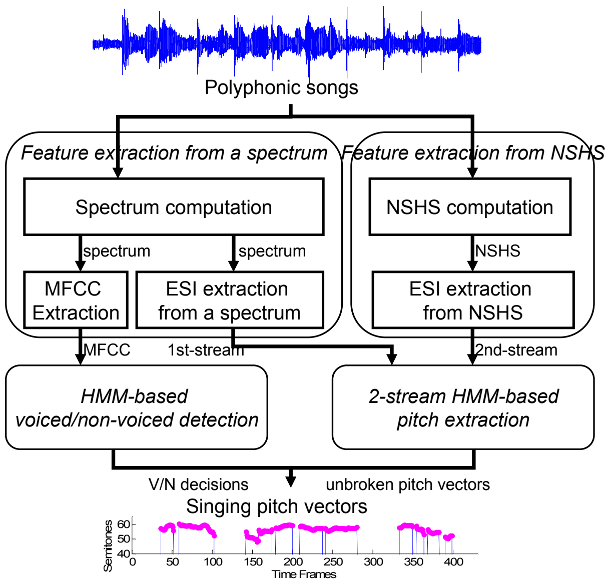


Figure 1. System overview

MFCCs are the features for a 2-state HMM for voiced/non-voiced detection. ESI is the 1st-stream feature for a 2-stream HMM for pitch extraction. Since most of the songs nowadays follow the twelve-tone equal temperament, it is intuitive to employ semitone scale to model the relations between melody and chords. For each integer semitone of interests within the range $[40,72]$, we identify its maximum energy as an element of the feature vector. Take semitone 69 for example, the search range in semitone is $[68.5,69.5]$, corresponding to a frequency bin of $[427.47,452.89]$ in terms of Hertz. Then we find the maximum power spectrum within this range as the feature associated with semitone 69. Since there are 33 elements within semitone of interests, the length of the feature vector of ESI is also 33.

More specifically, the ESI computed from a spectrum in the time frame t can be obtained as follows:

$$v_t(m) = \max_{f_m - \frac{f_m - f_{m-1}}{2} \le f < f_m + \frac{f_{m+1} - f_m}{2}} \left(P_t(f)\right), \tag{1}$$

where $P_t(*)$ is the power spectrum calculated from short time Fourier transform (STFT), $m = 0,1,..,M-1$, M is the total number of semitones that are taken into account, and f_m is the frequency of mth semitone in the selected pitch range.

Note that we also need to record the maximizing frequency within each frequency bin in order to reconstruct the most likely pitch contours.

2.2 HMM-based Voiced/Non-voiced Detection

This block employs a continuous 2-state HMM to decode the mixture input into voiced and non-voiced segments, similar to the one proposed by Fujihara et al. [9]. Note that the "voiced" here indicates the voiced singing voice, and "non-voiced" indicates the unvoiced singing voice and music accompaniments. Given the MFCC feature vectors $X = \{x_0, \cdots, x_t, \cdots\}$ of the input mixtures, the problem is to find the most probable sequence of voiced/non-voiced states, $\hat{S} = \{s_0, \cdots, s_t, \cdots\}$:

$$\hat{S} = \arg\max_S \left\{ p(s_0)p(x_0 \mid s_0) \prod_t \{p(s_t)p(x_t \mid s_t)p(s_t \mid s_{t-1})\} \right\}, \tag{2}$$

where $p(x \mid s)$ is the output likelihood of a state s, $p(s_t \mid s_{t-1})$ is the state transition probability from state s_{t-1} to s_t, and $p(s_t)$ is the prior of the state s_t. Note that $p(s_t \mid s_{t-1})$ and $p(s_t)$ can be obtained from the actual song data with manual annotations.

2.3 Features Extraction from NSHS

This block extracts the 2nd-stream feature vector which represents the energy distributions of the enhanced harmonic structures of singing voices. The harmonic structures can be enhanced by sub-harmonic summation (SHS) proposed by Hermes [10]:

$$H_t(f) = \sum_{n=1}^{N} h_n P_t(nf), \tag{3}$$

where $H_t(f)$ is the sub-harmonic summation value of the frequency f at time frame t, $P_t(*)$ is the power spectrum calculated from STFT, n is the index of harmonic components, N is the number of the harmonic components in consideration, and h_n is the weight indicating the contribution of the nth harmonic component. Usually we set $h_n = h^{n-1}$, where $h \le 1$. In order to further enhance the harmonics of singing voices, we propose the use of normalized SHS (NSHS) defined as follows:

$$\hat{H}_t(f) = \frac{\sum_{n=1}^{N_f} h_n P_t(nf)}{\sum_{n=1}^{N_f} h_n}, \tag{4}$$

where the number of harmonic components N_f depend on the frequency under consideration:

$$N_f = floor\left(\frac{0.5 f_s}{f}\right), \tag{5}$$

with f_s being the sampling rate. The reason of the modification is based on the observation that most of the energy in a song in located at the low frequency bins, and the energy of the harmonic structures of the singing voice seems to decay slower than that of instruments [2]. Therefore, when more harmonic components are considered, energy of the vocal sounds is further strengthened. Although some percussive instruments (e.g. cymbals)

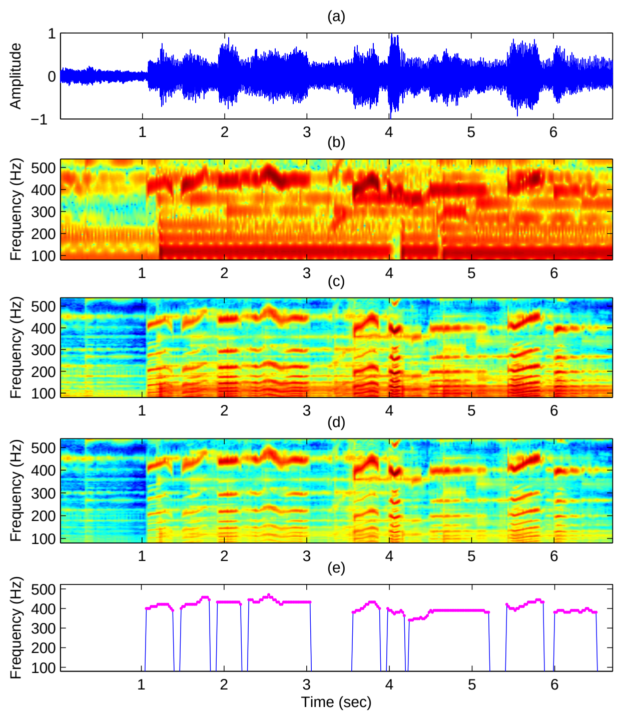


Figure 2. The energy distributions of a sample clip Amy_4_05 in MIR-1K dataset at 0 dB SNR. The distributions are computed within the frequency range [80.06, 538.58], or [39.5, 72.5] in terms of semitones. (a) The waveform of the mixture. (b) The spectrogram. (c) The SHS map. (d) The proposed NSHS map. (e) The manually labeled pitch vector of the singing voice.

present high energy at higher frequency bins, their non-harmonic nature does not affect the NSHS much.

Figure 2 illustrates the energy distributions of a traditional spectrogram, original SHS map, and the proposed NSHS map. By comparing the spectrogram in 2(b) and the SHS map in 2(c), it is obvious that most of the energy of accompaniments in the spectrogram is attenuated in the SHS map. However, the energy in the lower frequency bins remains high. The proposed NSHS map shown in 2(d) further attenuates the low-frequency energy and enhances the sub-harmonic structure of the singing voice. As a result, after the enhancement by the NSHS map, the pitch of the singing voice can be extracted much easier.

Based on the proposed NSHS, we can extract a 33-element feature vector of ESI for each given frame, as explained in Section 2.1. The feature vector is sent to the 2-stream HMM for pitch extraction.

2.4 2-Stream HMM-based Pitch Extraction

We employ a 2-stream HMM to model the relationship between the adjacent melody pitches and their corresponding audio context. Given the 1st-stream ESI feature vectors $V=\{v_0,\cdots,v_t,\cdots\}$ from spectrogram and the 2nd-stream ESI feature vectors $C=\{c_0,\cdots,c_t,\cdots\}$ from NSHS map, our goal is to find the most likely sequence of pitch states, $\hat{R}=\{r_0,\cdots,r_t,\cdots\}$:

$$\hat{R}=\arg\max_{R}\left\{p(r_0)p(v_0,c_0\mid r_0)\prod_t\{p(r_t)p(v_t,c_t\mid r_t)p(r_t\mid r_{t-1})\}\right\}, \tag{6}$$

where $p(r_t\mid r_{t-1})$ is the state transition probability from pitch state r_{t-1} to r_t, $p(r_t)$ is the prior of the pitch state r_t, and $p(v,c\mid r)$ is the joint output likelihood of the pitch state r defined as:

$$p(v,c\mid r)=p_v(v\mid r)p_c(c\mid r), \tag{7}$$

where $p_v(v\mid r)$ and $p_c(c\mid r)$ are the state likelihoods of feature vectors v and c, respectively, given the state r. This is a typical multi-stream HMM which is broadly used in speech processing [11]. The state likelihoods (or conditional observation likelihoods), transition probabilities, and priors of eq. (6) and (7) can all be obtained from the actual song data with manually annotated pitch contours.

Figure 3 shows the benefits of applying a 2-stream HMM instead of using a single-stream feature from either the spectrum or the NSHS. Each of the plots is a state-frame likelihood table where the vertical axis indicates the pitch state of each semitone and horizontal axis indicates time frames. The likelihood is computed for each state and time frame. All likelihood in the same time frame is normalized to zero mean and unity variance for better visualization. The ideal singing pitches are overlaid as solid lines. Figure 3(a) shows $p_v(v\mid r)$ of each state which utilizes audio context as cues to extract the singing pitches. Figure 3(b) shows $p_c(c\mid r)$ of each state which indicates the likelihood that an enhanced singing harmonic structure is presented, and Figure 3(c) shows the joint likelihood $p_v(v\mid r)p_c(c\mid r)$. Figure 3(d) and (e) illustrate the overall maximum likelihoods (up to a given frame time and pitch state) of single-stream HMMs using feature vectors V and C, respectively. More specifically, the value of each point in the figure represents the maximized accumulated likelihood of the previous pitch states sequence including the transition probabilities. Again, for better visualization, each column in these two tables is normalized to have zero mean and unity variance. The joint likelihood using the 2-stream HMM is shown in Figure 3(f). It can be observed that the likelihood of the singing pitch states at around 1.2 and 5.6 seconds are low in Figure 3(d), but they are recovered in Figure 3(f) by combining with the likelihood in Figure 3(e). In addition, the likelihood of the states that are not corresponding to the singing pitches between 1.5 and 5.3 seconds in Figure 3(e) are diminished in Figure 3(f) as well. Furthermore, both single-stream HMMs exhibit high likelihood for the singing pitch states. Therefore, after combining the likelihood using the 2-stream HMM, the likelihood of the singing pitch states are higher than that of the other states, and the pitches of singing voices can thus be extracted more accurately.

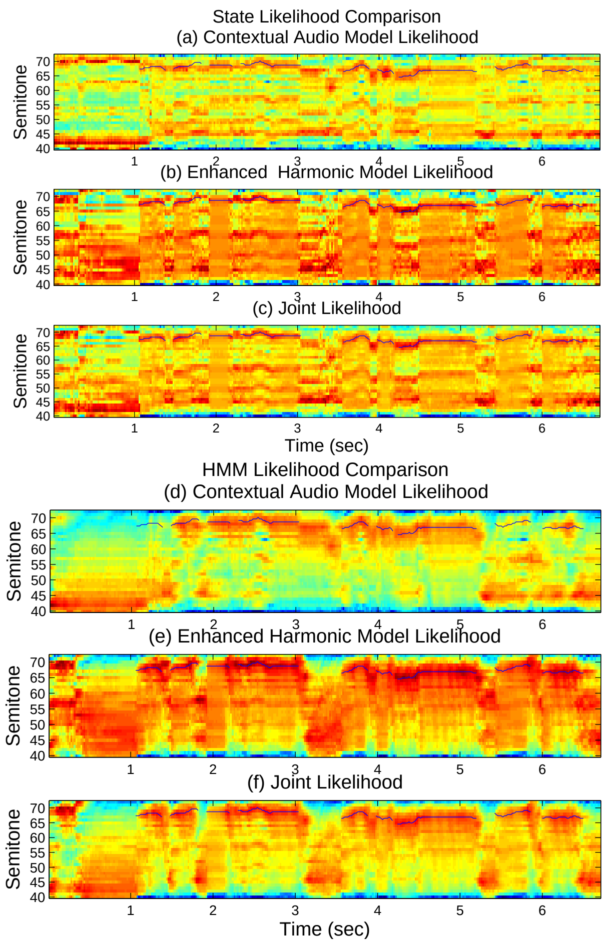


Figure 3. The state likelihood and HMM likelihood comparison for the clip Amy_4_05 in MIR-1K dataset at 0 dB SNR. (a) to (c) and (d) to (f) show the likelihood of contextual audio model, the likelihood of enhanced harmonic model, and the join likelihood of state likelihood and HMM likelihood, respectively. The solid line indicates the manually labeled pitch vector of the singing voice.

3. EVALUATION

Two datasets were used to evaluate the proposed approach. The first one, MIR-1K[1], is a publicly available dataset proposed in our previous work [12]. It contains 1000 song clips recorded at 16 kHz sample rate with 16-bit resolution. The duration of each clip ranges from 4 to 13 seconds, and the total length of the dataset is 133 minutes. These clips were extracted from 110 karaoke songs which contain a mixed track and a music accompaniment track. These songs were selected (from 5000 Chinese pop songs) and sung by our colleagues in the lab, consisting of 8 females and 11 males. Most of the singers are amateurs with no professional training. The music accompaniment and the singing voice were recorded at the left and right channels, respectively. The second dataset, called commercial set for short, contains 178 song clips from commercial CDs, and the total length of the dataset is about 25 minutes. The ground truth of the voiced/non-voiced segments and pitch values of the singing voices were first estimated from the pure singing voice and then manually adjusted for these two datasets.

All songs are mixed at 0 dB SNR, indicates that the energy of the music accompaniment is equal to the singing voice. Note that the SNRs for commercial pop songs are usually larger than zero, indicating that our experiments were set to deal with more adversary scenarios than the general cases.

	MIR-1K	Commercial set
Precision	87.48 %	91.14 %
Recall	86.03 %	91.78 %
Overall accuracy	81.52 %	87.12 %

Table 1. Performance of voiced/non-voiced detection

3.1 Evaluation for Voiced/Non-voiced detection

The evaluation was performed via two-fold cross validation with the MIR-1K dataset. The dataset was divided into two subsets of similar sizes (487 vs. 513, recorded by disjoint subjects). In addition, the commercial set was also evaluated by using all MIR-1K for training. The reason for not using the commercial set for training the voiced/non-voiced model is because its size is too small.

39-dimensional MFCCs (12 cepstral coefficients plus a log energy, together with their first and second derivatives) were extracted from each frame. The MFCCs were computed from STFT with a half-overlapped 40-ms Hamming window. Cepstral mean subtraction (CMS) was used to reduce channel effects.

Two 32-component GMMs were trained for voiced frames and non-voiced frames, respectively. All GMMs had diagonal covariance matrices. Parameters of the GMMs were initialized via k-means clustering algorithm and were iteratively adjusted via expectation-maximization (EM) algorithm with 30 iterations. Each of the GMMs was considered as a state in a fully connected 2-state HMM, where the transition probabilities and the weight of each GMMs were obtained through frame counts of the labeled dataset. For a given input song mixture, Viterbi algorithm was used to decode the mixture into voiced and non-voiced segments.

Table 1 shows the performance of voiced/non-voiced detection. The precision is the percentage of the frames that are correctly classified as voiced over the frames that are classified as voiced. The recall is the percentage of the frames that are correctly classified as voiced over all the voiced frames. The effects of the results will be discussed in the following subsections.

3.2 Evaluation for Singing Pitch Extraction

The MIR-1K dataset was divided into two subsets in the same way as subsection 3.1 for two-fold cross validation, and the commercial set was evaluated by using all MIR-1K for training. The spectrum of each frame was com-

[1] The MIR-1K dataset is available at http://unvoicedsoundseparation.googlepages.com/mir-1k

puted from STFT with a half-overlapped 40-ms window and zero padding to 2^{14}. In addition, the pitch range for computing ESI (for both spectra and NSHS) was [40-0.5, 72+0.5] in semitones or [80.06, 538.58] in Hertz, which is similar to the common singing frequency range used in [2]. The compression factor h for computing NSHS was set to 0.99. At last, a 33-dimentional feature vector v_t from spectra and a 33-dimentional feature vector c_t from NSHS were extracted for each frame.

Two diagonal 8-component GMMs, Γ_V and Γ_C, were trained for each of the 33 semitone models by using feature vectors v_t and c_t, respectively. Parameters of the GMMs were initialized via k-means clustering algorithm and were iteratively adjusted via EM algorithm with 30 iterations. Each of the $(\Gamma_V, \Gamma_C)_m$ pairs (with $m=[0,32]$) was considered as a state in an HMM, where the transition probabilities and the prior of each GMM were obtained through frame counts of the labeled dataset. For a given input mixture, Viterbi algorithm was used to decode the mixture into a sequence of pitch states $\hat{R}$. By tracking the maximizing frequency (which generates ESI at each semitone) for each pitch state, we can then reconstruct the optimum pitch contour.

In order to evaluate the proposed method, eight other approaches were used for comparison. For simplicity, we use SPEC and NSHS to indicate ESI that were extracted from a spectrum or a NSHS, respectively. In addition, HMM, DP, and MAX are used to indicate different schemes for extracting the singing pitches. More specifically, HMM represents the proposed HMM approach; DP represents the approach of dynamic programming over spectrum/NSHS directly (to be detailed next); MAX is simply maximum-picking over spectrum/NSHS.

The goal of the DP method is to find a path $f=[f_0,\cdots,f_i,\cdots,f_{n-1}]$ that maximizes the score function:

$$score(f,\theta)=\sum_{t=0}^{n-1} Y_t(f_t)-\theta\times\sum_{t=1}^{n-1}|f_t-f_{t-1}|, \qquad (8)$$

where $Y_t(f_t)$ is a feature vector extracted from spectrum/NSHS at the frame t and frequency f_t. The first term in the score function is the sum of energy of the pitches along the path, while the second term controls the smoothness of the path with the use of a penalty term θ (which is set to 2 in this study). If θ is larger, then the computed path are smoother. In particular, the MAX approach sets θ to be zero so that maximizing the above objective function is equivalent to maximum-picking of the features from spectrum/NSHS of each frame.

The DP method employs dynamic programming to find the maximum of the score function, where the optimum-valued function $D(t,m)$ is defined as the maximum score starting from frame 1 to t, with $f_t=m$:

$$D(t,m)=Y_t(m)+\max_{k\in[0,32]}\{D(t-1,k)-\theta\times|k-m|\}, \qquad (9)$$

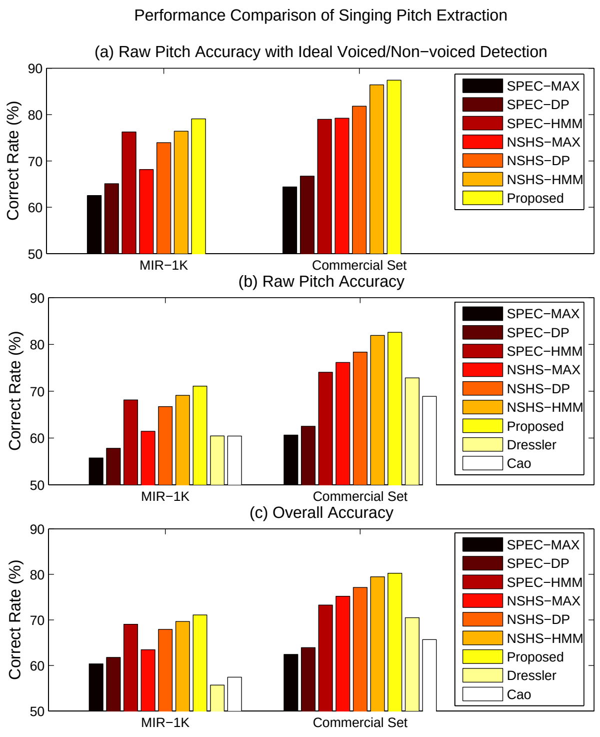


Figure 4. Performance comparison for singing pitch extraction. (a) Raw pitch accuracy with ideal voiced/non-voiced detection. (b) Raw pitch accuracy. (c) Overall accuracy.

where n is the number of frames, $t=[1,n-1]$, and $m=[0,32]$. The initial conditions are $D(0,m)=Y_0(m)$, and the optimum score is equal to $\max_{m\in[0,32]} D(n-1,m)$.

Moreover, the "Dressler" approach indicates a melody extraction method proposed by Dressler [4] which ranked first from 2005 to 2006 in the MIREX task of audio melody extraction. We obtained the software from her for comparison purpose. The "Cao" approach indicates the method proposed by Cao et al. [5], which was re-implemented by us for comparison.

Figure 4 shows the performance comparison for the singing pitch extraction. Figure 4(a) shows the raw pitch accuracy with ideal voiced/non-voiced detection, where the correct rate is computed over the frames that were labeled as voiced in the reference files. Figure 4(b) shows the raw pitch accuracy with automatically detected voiced/non-voiced segments. Figure 4(c) shows the overall accuracy where all frames are taken into account for computing the correct rate. In other words, Figure 4(a) shows the performance of the singing pitch extraction alone, assuming ideal voiced/non-voiced detection. On the other hand, the Figure 4(b) and (c) shows the performance in a practical situation where the results are affected by the errors of voiced/non-voiced detection. Since Dressler's and Cao's method perform singing voice detection implicitly, their performance is only shown for the cases of raw pitch and overall accuracy in Figure 4(b) and (c). Note that Dressler's method was designed not only to extract the melody from vocal songs, but also from non-vocal music which contains no singing voice, so the performance may not be as good as the other approaches that solely designed for vocal songs.

The proposed system achieved 71.10% and 80.24% overall accuracy in MIR-1K and commercial set, respectively. Experiments show that performance is significantly improved by applying the proposed HMM and by the NSHS in both datasets. Two points are worth noting. Firstly, while NSHS enhances the harmonic structures of both the singing voices and chords, the energy enhancement of chords is relatively weaker. Therefore, the improvement of using HMM over the MAX and DP approaches is much larger by using spectrum-based ESI than NSHS-based. This shows that the chord information embedded in spectrum-based ESI does help for extracting the singing pitches. Secondly, when spectrum-based ESI are replaced by NSHS-based ESI, the performance of MAX and DP is improved significantly. It shows that the NSHS does help for reducing the interference of non-singing pitches. By taking the advantages of both approaches, the proposed method therefore performs significantly better than the compared approaches.

4. CONCLUSIONS

In this paper, we propose a new singing pitch extraction system by employing a 2-stream HMM to model the relation between adjacent notes and between melody and chords. By modeling the energy distribution in spectrogram and in the proposed NSHS map, the performance is significantly improved. Besides, the improvement of the performance is quite similar in different datasets which confirms the robustness of the proposed approach.

The proposed NSHS only applies a simple weight function for harmonic components; the performance can be further improved by optimizing it with the training scheme proposed by Klapuri [13]. In addition, the raw pitch accuracy with ideal voiced/non-voiced detection of the proposed system is much higher than that of the overall accuracy (6~8%). Therefore it is also one of our future directions to improve the voiced/non-voiced detector by not only using MFCCs but also considering the voice vibrato information as proposed by Regnier et al. [14].

It is worth noting that the evaluation is performed by using our dataset, MIR-1K, which contains more song clips than that used in MIREX (less than 20 minutes, and only 7 minutes of them are publicly available). It allows researchers to evaluate and compare their systems with others easily by using the more comprehensive dataset.

5. ACKNOWLEDGEMENT

This study is conducted under the "III Innovative and Prospective Technologies Project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs, the Republic of China. The authors would also like to thank K. Dressler for providing her pitch tracking software.

6. REFERENCES

[1] M. Goto, “A Real-Time Music Scene Description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-World Audio Signals,” *Speech Communication*, vol. 43, no. 4, pp. 311–329, 2004.

[2] Y. Li and D. L. Wang, “Detecting Pitch of Singing Voice in Polyphonic Audio,” *IEEE ICASSP*, pp. 17–20, 2005.

[3] G. E. Poliner and D. P. W. Ellis, “A Classification Approach to Melody Transcription,” *6th ISMIR*, pp. 161-166, 2005.

[4] K. Dressler, “An Auditory Streaming Approach on Melody Extraction,” *Extended abstract for 7th ISMIR*, 2006.

[5] C. Cao, M. Li, J. Liu and Y. Yan, “Singing Melody Extraction in Polyphonic Music by Harmonic Tracking,” *8th ISMIR*, 2007.

[6] V. Rao and P. Rao, “Melody Extraction Using Harmonic Matching,” *Extended abstract for 9th ISMIR,* 2008.

[7] J.-L. Durrieu, G. Richard and B. David, “An Iterative Approach to Monaural Musical Mixture De-soloing,” *IEEE ICASSP*, pp. 105-108, 2009.

[8] M. Ryynänen and A. Klapuri, "Transcription of the Singing Melody in Polyphonic Music," *7th ISMIR*, pp. 222-227, 2006.

[9] H. Fujihara, M. Goto, J. Ogata, K. Komatani, T. Ogata, and H. G. Okuno, “Automatic Synchronization between Lyrics and Music CD Recordings Based on Viterbi Alignment of Segregated Vocal Signals,” *ISM*, pp. 257–264, 2006.

[10] D. J. Hermes, “Measurement of Pitch by Subharmonic Summation,” *Journal of Acoustic Society of America*, vol.83, pp. 257-264, 1988.

[11] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. J. Odell, D. Ollason, D. Povery, V. Valtchev, and P. C. Woodland: *The HTK Book (for HTK version 3.4)*, Cambridge University, 2006.

[12] C. L. Hsu and J. S. Jang, “On the Improvement of Singing Voice Separation for Monaural Recordings Using the MIR-1K Dataset,” *IEEE Trans. Audio, Speech, and Language Processing*, accepted.

[13] A. Klapuri, "Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes," *7th ISMIR*, 2006.

[14] L. Regnier and G. Peeters, “Singing Voice Detection in Music Tracks using Direct Voice Vibrato Detection,” *IEEE ICASSP*, pp. 1685-1688, 2009.

USABILITY EVALUATION OF VISUALIZATION INTERFACES FOR CONTENT-BASED MUSIC RETRIEVAL SYSTEMS

Keiichiro Hoashi† Shuhei Hamawaki‡ Hiromi Ishizaki† Yasuhiro Takishima† Jiro Katto‡
† KDDI R&D Laboratories Inc.
{hoashi, ishizaki, takisima}@kddilabs.jp
‡ Graduate School of Science and Engineering, Waseda University
{hamawaki, katto}@katto.comm.waseda.ac.jp

ABSTRACT

This research presents a formal user evaluation of a typical visualization method for content-based music information retrieval (MIR) systems, and also proposes a novel interface to improve MIR usability. Numerous interfaces to visualize content-based MIR systems have been proposed, but reports on user evaluations of such proposed GUIs are scarce. This research aims to evaluate the effectiveness of a typical 2-D visualization method for content-based MIR systems, by conducting comparative user evaluations against the traditional list-based format to present MIR results to the user. Based on the observations of the experimental results, we next propose a 3-D visualization system, which features a function to specify sub-regions of the feature space based on genre classification results, and a function which allows users to select features that are assigned to the axes of the 3-D space. Evaluation of this GUI conclude that the functions of the 3-D system can significantly improve both the efficiency and usability of MIR systems.

1. INTRODUCTION

The popularity of online music distribution services have provided an opportunity for users to access to millions of songs. Furthermore, the rapid spread of portable devices with large storage, *e.g.*, the iPod and 3G mobile phones, has also enabled common users to carry around music collections, which may consist of thousands of songs. These developments have prompted the need for effective music information retrieval (MIR) technologies, in order to ease the user burden to find songs which they want to listen to.

It is obvious that, for any MIR system, the usability of its interface is essential for the user to efficiently search for songs which match their preferences. However, while various GUIs for MIR systems have been proposed, reports on user evaluations of such GUIs are scarce, hence, the effectiveness and/or problems of visualizing MIR systems are yet to be formally clarified.

The objective of this research is to evaluate the effectiveness of MIR visualization, and derive what functions are necessary to improve its usability. In order to accomplish this objective, we first conduct a comparative user experiment between a typical 2-D visualization MIR interface, against the traditional list-based format. Through the analysis of this experiment, we verify if visualization actually contributes to improve the efficiency of MIR, and also derive potential problems of visualization in general. Based on the knowledge obtained from this analysis, we next propose an extended 3-D interface with several new functions, which aim to resolve the problems that have become apparent from the results of the prior experiment. Comparative experiments with the previous GUI indicate that the additional functions contribute to improve the efficiency and entertainability of MIR systems.

2. RELATED WORK

One of the initial research efforts to visualize content-based MIR is the *Islands of Music* application, developed by Pampalk [1]. This application utilizes self-organized maps to plot songs on a two-dimensional feature space, and expresses populated clusters in the feature space by illustrating "islands" of music. Furthermore, Lamere *et al.* have developed a visualization application called *Search Inside the Music* [2], which calculates audio-based similarity between songs in the music collection, and locates highly similar songs adjacently in a 3-D feature space.

Recently, there have been numerous reports to collect meta-information of songs and/or artists from the Web, and display the collected information on the user interface of MIR systems, to support the users' music searching process. *MusicRainbow* [3] is an application designed to discover artists, which maps similar artists on a circular "rainbow." The artist similarity is calculated based on acoustic analysis of the artists' songs. Labels of the artists are applied by analyzing Web information retrieved by using the name of the artists as the search query. Other applications to visualize music with web-based metadata include *MusicSun* [4], an extended application of *MusicRainbow*, as well as the work presented in [5, 6], etc.

As clear from the above descriptions of existing work, many proposals of user interfaces for MIR systems have been made in recent years. However, thorough user evaluations of such interfaces are scarce. Therefore, it is not apparent that the various functions, user interface designs, etc. which have been proposed in previous work, actually contribute to improve the overall usability of MIR systems.

Considering these problems, we set the motivation of our research to first evaluate the effectiveness of typical visualization interfaces, by comparison with the traditional list-based format to output MIR results. Furthermore, through the analysis of user logs of the experiment, we will clarify the advantages and drawbacks of visualization, and utilize this analysis to further improve usability of MIR visualization interfaces.

3. EVALUATION OF 2-D INTERFACE

Our first experiment is to evaluate a prototype 2-D interface for content-based MIR systems, by comparison with the traditional list-based interface. For the following experiment, 16 subjects (all Japanese students in computer science) have participated to work on an experiment task using the MIR systems. Details of the systems, and the evaluation experiment are as follows.

3.1 Systems

3.1.1 Feature extraction

For both MIR systems used in the experiment, the songs in the music collection are vectorized, based on the TreeQ algorithm developed by Foote [7]. For the initial training data of TreeQ, we use the songs and sub-genre metadata of the RWC Genre Database [8]. MFCCs are extracted as the features used for the TreeQ method. In order to optimize the feature space to suit the characteristics of the experimental music collection, we re-construct the feature space based on the algorithm proposed by Hoashi *et al* [9]. The final song vectors are generated based on the re-constructed feature space.

3.1.2 2-D interface

Based on the song vectors generated by the previous process, we have developed a prototype 2-D interface for our MIR system. In order to plot the song vectors to the 2-D space, the vectors are compressed to two dimensions, by conducting principal component analysis (PCA), and extracting the first two components of the PCA results.

A screenshot of this system is illustrated in Figure 1. The 2-D interface consists of two major components: the *macro feature space viewer*, which displays the entire "universe" of the music feature space, and the *local feature space viewer*, which displays a close-up view of the area where the user is interested in. In this system, users can first select their area of interest, by clicking on the macro feature space viewer. Next, users can listen to songs in the selected area, by clicking on the plots displayed in the local feature space viewer.

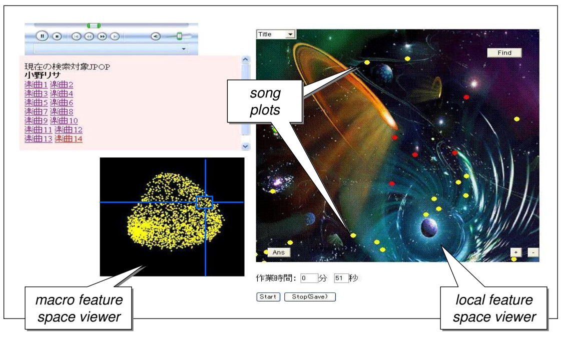


Figure 1. Screenshot of 2-D interface

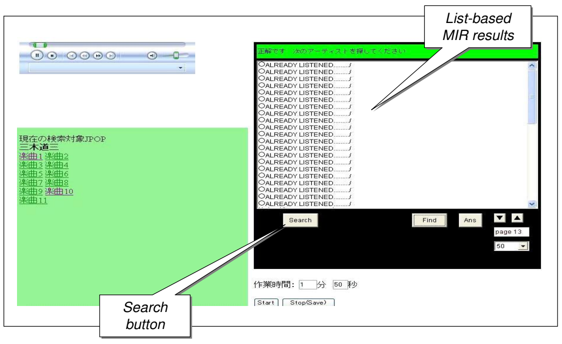


Figure 2. Screenshot of list-based interface

3.1.3 List-based interface

For comparison with the 2-D interface, we have also developed a list-based MIR system, which outputs the MIR results in a list-based format. This interface resembles the list format to present search results for typical Web search engines. A screenshot of this system is shown in Figure 2.

In this system, a random list of songs in the music collection is initially presented, with the title and artist information hidden to the user. Users are to search for the target songs, by first listening to the songs in this initial list to select a query, and clicking on the "Search" button to execute MIR. The vector similarity between the query song and all other songs are calculated, and the songs with high similarity are presented in the list, sorted accordingly to the similarity to the query song. Users can listen to the songs in the list and continue searching, by repeating the MIR procedure for the songs listed in the MIR results.

3.2 Experiment task

The task given to the subjects of the experiment is to use the two previous MIR systems, to search for songs performed by specific artists. For each experiment, a set of target songs, which are performed by a pre-specified Japanese artist, are added to the base collection, which consists of 723 Korean pop songs (hereafter referred to as *K-pop* songs). Naturally, the *K-pop* songs are unfamiliar to the subjects. The number of target songs ranges from 10 to 14 songs, depending on the target artist.

Prior to each experiment, the subjects are provided with a set of sample songs, which are performed by the target artist, but are different from the target songs added to the base collection. Based on the impression of listening to the sample songs, the subjects then use the MIR system to search for the target songs. A single task session finishes, when the subject has successfully discovered one of the target songs. Each experiment consists of three sessions. For each session, a different artist is assigned as the target. Each subject performs this experiment, for both the 2-D and list-based systems.

3.3 Evaluation measures

Comparative evaluation of the two systems are conducted by the following objective and subjective measures.

3.3.1 Objective evaluation measures

For objective evaluation, we measure the efficiency of the MIR experiment task, based on the following two measures:

- **Operation time** *(OTime)*: Time required to complete experiment task.
- **Number of song plays** *(PlayNum)*: Number of songs played to complete task.

3.3.2 Subjective evaluation measures

For subjective evaluation, each subject is asked to apply a five-ranked rating to the systems, for the following elements (1:Bad $\sim$ 5:Good):

- **Operability** *(Oper)*: Evaluation of the usability of the interface
- **Accuracy** *(Acc)*: Evaluation of the accuracy of MIR results
- **Explicitness** *(Expl)*: Easiness to grasp relationship between songs in the MIR results
- **Enjoyability** *(Enj)*: Evaluation of overall entertainability of system

3.4 Results

3.4.1 Comparison of evaluation measure results

In order to directly compare the two systems, we summarize the evaluation results by an election-like approach. For each subject, the evaluation value for each measure is compared, and a "vote" for the subject is cast to the system with the higher score. For the objective measures (*OTime*, *PlayNum*), the vote is cast to the system with the smaller value, since the efficiency of a superior system should result in lower *OTime* and *PlayNum*.

The resulting number of votes of all evaluation measure for the two systems, are illustrated in Figure 3. Note that, the votes of the subjects who could not complete the experiment task are omitted from the results of the objective measures (*OTime*, *PlayNum*).

Comparison of the number of votes for the objective measures in Figure 3, show that the number of subjects

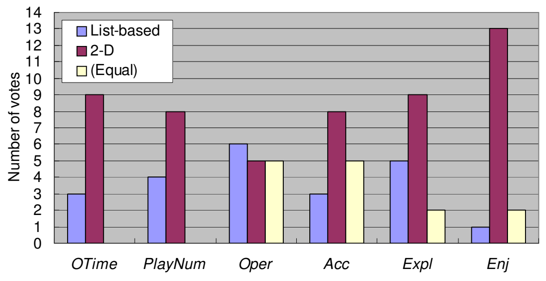


Figure 3. Number of votes for 2-D and list-based systems

who have completed the experiment task more efficiently with the 2-D interface is approximately twice as high as the list-based interface. This result indicates that the 2-D interface contributes to improve MIR efficiency. However, the vote results for *Oper* show that, subjectively, the operability of the two systems do not differ as much as the objective evaluation results suggest. A reason for this result is assumed to be that, many of the subjects are generally used to the traditional list-based interface, while the 2-D interface requires time to get acquainted with. Further analyses to discuss this result will be presented in the following section.

The results for the other subjective evaluation measures show that the 2-D interface has been more well-accepted than the list-based interface. The reason why the 2-D system has received more votes for *Expl* is obvious, since the list-based interface can only present the similarity between the query songs and other songs in the MIR results, while the 2-D interface not only displays the similarity to the query, but also the relative position between other songs that are plotted in the interface. An interesting observation is that the accuracy (*Acc*) of the 2-D system has received more votes, despite the fact that the feature extraction and vectorization methods of the systems are the same. This result indicates that the visualization of the feature space not only improves efficiency of MIR, but also applies a better impression about the accuracy of MIR results.

3.4.2 Analysis of user logs

For further analysis of the 2-D system, we analyze the user logs of the experiment. Figures 4 and 5 illustrate the *PlayNum* of all subjects, for the three target songs (hereafter referred to as *J-pop*{*1,2,3*}, respectively). For the first target song (*J-pop1*), *PlayNum* is generally lower for the list-based system, compared to that of the 2-D system. This indicates that the initial efficiency of MIR is higher for the list-based system, which is assumed to be the reason for the close voting results for *Oper* in Figure 3. However, as the experiment proceeds to the second and third target songs, a decreasing trend of *PlayNum* can be observed for the 2-D system, while no such trend is apparent for the list-based system. These observations show that users are able to search for music with high efficiency by using the 2-D system, as soon as they get acquainted with its interface.

Next, we analyze the search logs of 2-D system users,

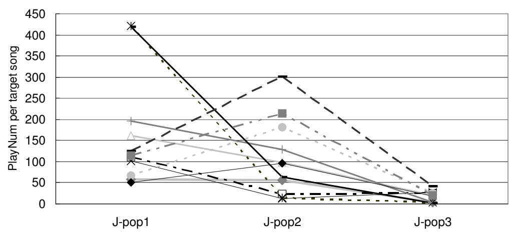


Figure 4. *PlayNum* per target song (2-D system)

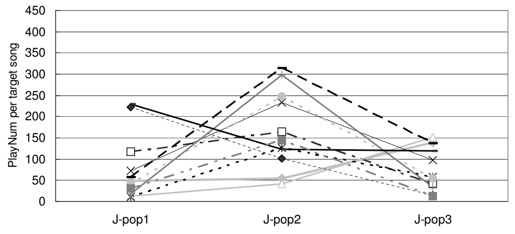


Figure 5. *PlayNum* per target song (list-based system)

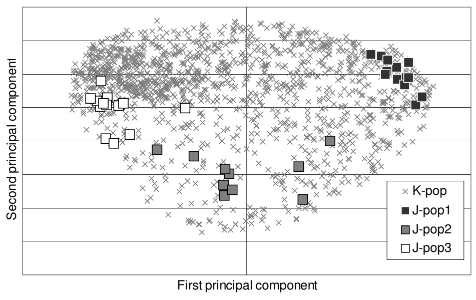


Figure 6. Song plots in the 2-D interface

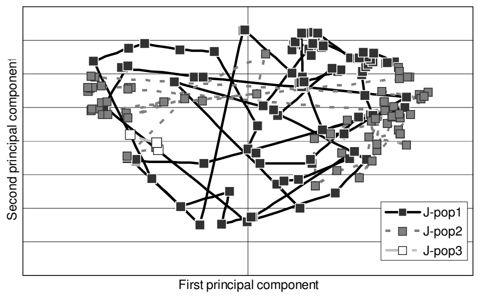


Figure 7. Search path of typical user of 2-D interface

to identify drawbacks of the 2-D system. Figure 6 illustrates the distribution of the *K-pop* and target song plots in the 2-D feature space, and Figure 7 shows the search path of a typical subject when using the 2-D interface. The search path for the first target song *J-pop1* shows that the subject first tries to grasp the characteristics of the feature space, by listening to songs that are plotted in various locations. In the next session, *i.e.*, the search for *J-pop2*, the log shows that the subject initially focuses on the edges of the feature space, where the acoustic features of the songs are assumed to be characteristic from the others, and gradually moves towards the center area. Finally, in the third session, it is clear that the subject is able to discover the target songs (*J-pop3*) with ease, assumably based on his/her experience from the previous sessions.

Overall, the above experimental results provide proof that the visualization interface of the prototype 2-D system contributes to improve the efficiency and usability of the MIR process. However, analysis of user logs also show that a major problem of the visualization interface is that users are required to experience the system sufficiently, in order to grasp its characteristics. Another interpretation of this result may be that, many users have experienced difficulty to comprehend the features, *i.e.*, the timbral features of the MFCC-based TreeQ vectors, that are used for the visualization of the songs in the MIR system.

4. 3-D MIR SYSTEM INTERFACE

In order to resolve the problems that have become apparent from the previous experiment, we next develop an extended visualization interface for content-based MIR. This system features a 3-D visualization interface with the following additional functions: (1) Selection of sub feature spaces based on genre classification, and (2) User selection of features which define the axes in the 3-D feature space. A screenshot of the 3-D system is illustrated in Figure 8. The objective of the first function is to provide an intuitional way to choose specific areas within the visualization feature space, in order to reduce the time required to grasp its characteristics. The second function aims to improve the understandability of the visualization interface, by allowing the users to select features which they prefer to use for MIR. Details of the system are presented in the following sections.

4.1 Selecting subspaces based on genre classification

A major problem of visualization, as clarified from the previous experiment, is the experience time required for the user to get acquainted to the visualization interface. In order to reduce this acquaintance time, we have added a function to the 3-D system, which enables the user to select "sub-spaces" in the macro feature space viewer, by utilizing genre classification results.

The sub-spaces are generated by conducting k-means clustering on all vectors of the songs included in the music collection, plus the song vectors of the RWC Genre Database [8], which were used as the initial training data to generate the tree-based vector quantizer. Next, labels are applied to the clusters, by referring to the sub-genre metadata of the RWC songs that are classified to each of the clusters. The number of clusters k was set to $k = 7$, based on preliminary experiments. The labels applied to the clusters are listed in Table 1.

Each sub-space is represented as a sphere in the macro feature space viewer of the 3-D system. The labels for each sub-space are listed below the viewer. Users can drag in the viewer to rotate the feature space, and click on the sphere which best represents their area of interest. This

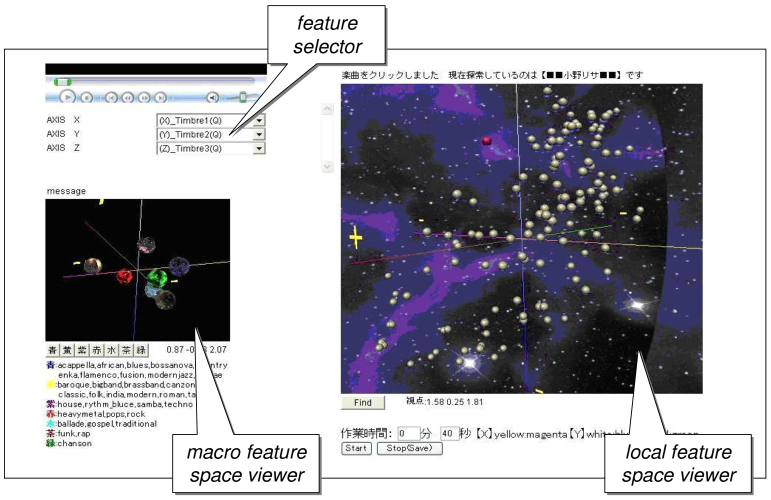


Figure 8. Screenshot of 3-D interface

Cluster ID	Labels
C_1	blues, bossanova, enka, flamenco, reggae
C_2	baroque, bigband, classic, folk, india
C_3	house, rhythm_blues, samba, techno
C_4	heavymetal, pops, rock
C_5	ballad, gospel, traditional
C_6	funk, rap
C_7	chanson

Table 1. Sub-space labels of 3-D system

function enables users to select areas in the feature space intuitively, thus is expected to resolve the try-and-error approach necessary to get acquainted with the previous 2-D system.

4.2 Selection of features for visualization

By clicking on a sub-space sphere in the macro feature space viewer, users can view the plots of the songs which belong in the selected sub-space (cluster), on the local feature space viewer. As illustrated in Figure 8, the song plots are displayed in a 3-D feature space.

Another problem of visualization is the unfamiliarity of the features used to visualize music. However, it is also clear that the appropriateness of features differ among individual users. For example, users with musical experience may consider keys or chords as adequate features for MIR, while casual music listeners may not be able to discriminate such high-level features of music.

In order to resolve this problem, we have added the *feature selector* function, which allows the user to define the features to be used in the 3-D feature space. By using the feature selector, system users can select a feature which corresponds to the x, y, z axes in the feature space. The list of features from which users can select from, are written in Table 2, along with the tools used to extract the features: TreeQ [10], MIRtoolbox [11], jAudio [12], and COFViewer [13]. Note that the "Timbre" features are equivalent to the PCA results of the TreeQ vectors utilized in the previous 2-D system.

Feature	Description	Tool
Timbre(1,2,3)	PCA results (first 3 components) of TreeQ vectors	TreeQ
Man-Woman	Categorization score of Male/Female vocal TreeQ classifier	TreeQ
Tempo	Average tempo	MIRtoolbox
Dynamics	Overall power of song	jAudio
Key	Basic key of song	COFViewer
TransKey	Transition ratio of keys	COFViewer
Mode	Degree of major/minor	MIRtoolbox
Stat(1,2,3)	PCA results (first 3 components) of jAudio features	jAudio

Table 2. List of features in 3-D system

5. EVALUATION OF 3-D INTERFACE

We have conducted a user experiment to evaluate the 3-D system. The objective of this experiment is to examine whether the problems of the 2-D system have been resolved by the additional features of the 3-D system. The task of the experiment, and the subjects are the same as the experiment described in Section 3. The measures used for evaluation are also the same as the previous experiment.

5.1 Results and discussions

First, we compare the usability of the 2-D and 3-D interfaces, by counting the votes of the subjects for all evaluation measures, similar to the evaluation in Section 3.4.1. Figure 9 illustrates the number of votes for the 2-D and 3-D systems.

The voting results for the objective measures, *OTime* and *PlayNum*, indicate that the 3-D system has contributed to improve the efficiency of MIR. Furthermore, the votes for the subjective measures show that the subjects have considered the 3-D system to be superior than the 2-D system, for all evaluation measures.

As mentioned in Section 3.4.2, a problem of the 2-D system is the time required to get used to the interface. In order to examine if the 3-D system has resolved this

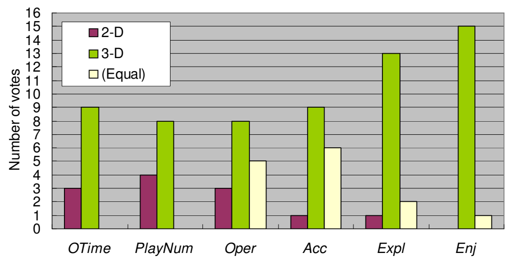


Figure 9. Number of votes for 2-D and 3-D systems

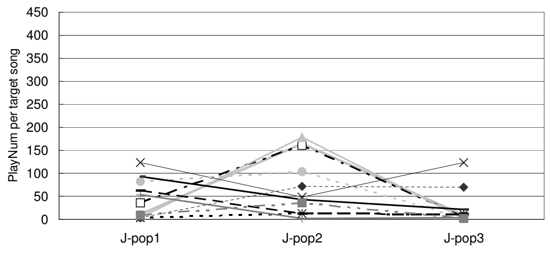


Figure 10. *PlayNum* per target song (3-D system)

problem, we analyze the *PlayNum* of the 3-D system for the three target songs, similar to the analysis presented in Figure 4. This result is illustrated in Figure 10.

Comparison of the results in Figures 4 and 10 show that, the *PlayNum* of the first target song *J-pop1* is lower for the 3-D system. This result shows that users have required less time to get acquainted with the interface of the 3-D system.

Next, in order to evaluate the usability of the feature selection function, we asked the subjects to select which features they considered useful for the experiment task, after the experiment (subjects are allowed to select multiple features in this questionnaire). The number of selections for each feature is listed in Table 3.

The results in Table 3 show that system users have selected features other than the timbre features that were used in the 2-D system. Furthermore, in this questionnaire, 11 of the 16 subjects have selected more than one feature as useful. These results indicate that users have proactively selected various features during the task, meaning that users have favorably accepted the feature selection function.

6. CONCLUSION

In this research, we have evaluated the effectiveness of visualization methods for content-based MIR, by conducting comparative user experiments of various MIR interfaces. The first experiment, which is based on a GUI with a typical 2-D visualization approach, has proved that visualization contributes to improve overall usability compared to the list-based interface, but also indicate problems for inexperienced users to comprehend the characteristics of the visualized feature space. Evaluation of the extended GUI, which is added new functions to resolve the previous problems, makes clear that the additional functions further contribute to improve the usability of MIR systems. Through

Feature	No of selections
Man-Woman	14
Tempo	8
Timbre	4
Dynamics	4
Key	1
Total	31

Table 3. Number of selections per feature in 3-D system

this research, we consider ourselves to have contributed to clarify general user requirements for the visualization of MIR systems, and also have proposed a successful approach to satisfy such needs.

7. REFERENCES

[1] E. Pampalk: Islands of Music: Analysis, Organization and Visualization of Music Archives, Master's thesis, Vienna University of Technology, 2001.

[2] P. Lamere, D. Eck: Using 3D Visualizations to Explore and Discover Music, Proc. of ISMIR 2007, 2007.

[3] E. Pampalk, M. Goto: MusicRainbow: A New User Interface to Discover Artists Using Audio-based Similarity and Web-based Labeling, Proc. of ISMIR 2006, 2006.

[4] E. Pampalk, M. Goto: MusicSun: A New Approach to Artist Recommendation, Proc. of ISMIR 2007, 2007.

[5] P. Knees, M. Schedl, T. Pohle, G. Widmer: An Innovative Three-dimensional User Interface for Exploring Music Collections Enriched with Meta-information from the Web, Proc. of ACM Multimedia 2006, 2006.

[6] M. Torrens, P. Hertzog, J.-L. Arcos: Visualizing and Exploring Personal Music Libraries, Proc. of ISMIR 2004, 2004.

[7] J. Foote: Content-based Retrieval of Music and Audio, Proc. of SPIE, Vol. 3229, pp. 138-147, 1997.

[8] M. Goto, H. Hashiguchi, T. Nishimura, R. Oka: RWC Music Database: Music Genre Database and Musical Instrument Sound Database, Proc. of ISMIR 2003, 2003.

[9] K. Hoashi, K. Matsumoto, F. Sugaya, H. Ishizaki, J. Katto: Feature Space Modification for Content-based Music Retrieval based on User Preferences, Proc. of ICASSP 2006, Vol. V, pp. 517-520, 2006.

[10] "TreeQ software," http://treeq.sourceforge.net/

[11] O. Lartillot: MIRtoolbox, http://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

[12] D. McEnnis, C. McKay, I. Fujinaga, P. Depalle: jAudio: A Feature Extraction Library, Proc. of ISMIR 2005, 2005.

[13] T. Inoshita, J. Katto: Key Estimation Using Circle of Fifth, Proc. of MMM 2009, 2009.

MUSIC PASTE: CONCATENATING MUSIC CLIPS BASED ON CHROMA AND RHYTHM FEATURES

Heng-Yi Lin† Yin-Tzu Lin‡ Ming-Chun Tien‡ Ja-Ling Wu†‡
National Taiwan University
†Department of Computer Science and Information Engineering
‡Graduate Institute of Networking and Multimedia
{waquey,known,trimy,wjl}@cmlab.csie.ntu.edu.tw

ABSTRACT

In this paper, we provide a tool for automatically choosing appropriate music clips from a given audio collection and properly combining the chosen clips. To seamlessly concatenate two different music clips without causing any audible defect is really a hard nut to crack. Borrowing the idea from the musical dice game and the DJ's strategy and considering psychoacoustics, we employ the currently available audio analysis and editing techniques to paste music sounded as pleasant as possible. Besides, we conduct subjective evaluations on the correlation between pasting methods and the auditory quality of combined clips. The experimental results show that the automatically generated music pastes are acceptable to most of the evaluators. The proposed system can be used to generate lengthened or shortened background music and dancing suite, which is useful for some audio-assisted multimedia applications.

1. INTRODUCTION AND MOTIVATION

Nowadays, more and more music lovers prefer to create their own music from the existing music audio collections, for the purpose of generating background music or dancing suite with specific length or composing a new song with all the favorite parts from different songs. However, they often confront difficulties in reaching a desirable result. The main problem lies in how to choose appropriate music clips from a large database and find out proper connecting-positions among these chosen clips. To our big surprise, studies on the relationship between the "hearing quality" and the "connecting-positions" in music combining has been long ignored. Conventionally, professional users would rely on their music sense and a few music theories to choose the clips and the connecting-positions, but the editing process is still try-and-error. As the amount of tasks increases, the process becomes time-consuming and labor-intensive. Therefore, the goal of this paper is two-fold: **(i)** providing a tool for automatically choosing appropriate music clips from given audio collections and combining the chosen clips as euphonious as possible, and **(ii)** conducting several experiments and investigating the relationship between pasting methods and the corresponding auditory quality. The ultimately combined music is named as "music paste" because it is just like the concept of pasting. The terminology "euphonious" is defined as follows: **(i)** listeners do not notice the transitions in the music paste, or **(ii)** listeners do notice the transition but they do not perceive the exact connecting-positions or the transitions sound pleasant to them.

2. RELATED WORK

2.1 Combine Music in Symbolic Domain

Combining two music clips in symbolic domain has been early studied. In the European classical era, preeminent composers developed a kind of musical dice game called Musikalische Würfelspiele [1] . Composers composed numbers of music clips for each measure in advance. While playing the game, players throw a dice to select the predetermined music clips. The action is performed for every measure. The generated music piece would not be strange because the music clip candidates for the same measure usually consist of the same chord or similar dominant tones. Based on this idea, Cope [2] conducted numerous experiments and developed a music-generating system. In the system, music clips of master composers have been analyzed and recombined to generate a new master style music piece.

The advantage of combining music clips in symbolic domain is that it causes less artifacts in auditory aspect. It is simple to transpose the midi clips to the same scale and directly combine two midi clips without causing artifacts while artifacts are usually inevitable in combined audio clips. However, the approaches in symbolic domain are not easy to be applied in audio domain due to the complication of polyphonic audio files. Moreover, current state of the art music transcription and separation techniques are not accurate enough to extract all the musical notes from polyphonic clips. Thus, the most commonly applicable editing operations in audio domain are only tempo change, remix, and concatenation.

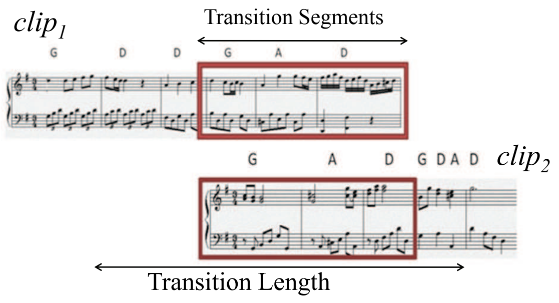


Figure 1. An example of pasting at the measures with the same chords.

2.2 Combine Music in Waveform Domain

Conventionally, what a DJ does can be treated as a human version of music-combining system. DJs often have talents for combining music clips appropriately. They can obtain hidden messages from the music clips by hearing even without the score information. In addition to choosing proper clips and connecting-positions, they also change the tempi around the connecting-positions of music clips to make the pasted music clips be pleasant for hearing. Based on this concept, Tristan [3, 4] proposed an automated DJ system by extracting auditory features, connecting the clips at rhythm-similar segments and aligning the beats of clips. However, the concatenated result may be discordant if these rhythm-similar segments are not pitch-similar. Thus, in this work, we adopt the chroma-based similarity measurement to solve this problem. Moreover, several useful schemes for filtering out dissimilar music clips are presented as well.

3. SYSTEM OVERVIEW

The key idea of the proposed system is as follows:

People usually anticipate the succeeding notes while listening to music [1]. Figure 1 shows an example of 2 input clips: $clip_1$, $clip_2$. Originally, each input clip fits people's expectation. To continue the expectation between the clips, we choose the most similar segments (pitch-similar and rhythm-similar, inspired by the musical dice game and automated DJ) between them as the connecting-position. Then, we can ensure that in the pasted music, from the beginning through the connecting-position to the end will all conform to people's anticipation. In this example, the last three chords of $clip_1$ are the same as the first three chords of $clip_2$. So, we connect these 2 clips by superimposing the beginning of $clip_2$ onto $clip_1$ at the position of the third last chord. We define the overlapping parts (the marked chords) as "transition segments." It will be determined by finding the most alike segments of the two combined clips. Besides, we use another term "transition length" to represent the length (in beat) we need for gradually adjusting the tempo from $clip_1$ to that of $clip_2$ if there is a discrepancy of tempi in these two clips.

The proposed system framework is illustrated in Figure 2. First, we extract all the features we need from music clips such as loudness, chroma, rhythm, and tempo. Then, we filter out dissimilar music clips by pair-wise comparisons.

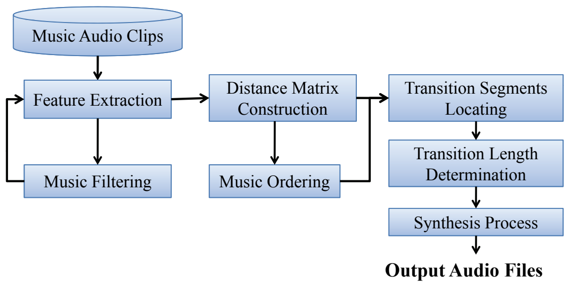


Figure 2. The block diagram of the proposed system.

Next, we construct a distance matrix by chroma and rhythm features. With this matrix, we determine the transition segments and decide an appropriate pasting ordering. After that, we determine the transition lengths and adjust the tempi within them. Then, we rearrange the volume within the transition segments and synthesize all the processed music clips. For better understanding, we will firstly describe how to paste two music clips in section 4. And the music ordering and filtering schemes for dealing with more than two clips will be illustrated in section 5.

4. CONCATENATION OF TWO CLIPS

In this section, we describe the process of pasting two music clips. We name these two clips as $clip_1$ and $clip_2$.

4.1 Transition Segments Locating

The common-used similarity/distance matrix [5] method is applied to measure the similarities between $clip_1$ and $clip_2$. We extract chroma-based [6] and rhythm features [7] per beat and then calculate their Euclidian distance. Thus, the smaller the values are, the more similar the segments are. Let $D_{c_{12}}(i,j)$ and $D_{r_{12}}(i,j)$ represent the chroma and rhythm distance values between $clip_1$'s i^{th} beat and $clip_2$'s j^{th} beat, respectively. That is,

$$D_{c_{12}}(i,j) = ||\vec{C}_{1i} - \vec{C}_{2j}||_2 \tag{1}$$

$$D_{r_{12}}(i,j) = ||\vec{R}_{1i} - \vec{R}_{2j}||_2 \tag{2}$$

where $\vec{C}_{1i}$ and $\vec{C}_{2i}$ denote $clip_1$'s i^{th} and $clip_2$'s j^{th} chroma vectors, respectively. And similarly, $\vec{R}_{1i}$ and $\vec{R}_{2i}$ represent the rhythm feature vectors. The two matrices $D_{c_{12}}(i,j)$ and $D_{r_{12}}(i,j)$ are linearly combined into a new matrix $D_{cr_{12}}$ (as shown in Eqn. (3)), which is the distance matrix we used for finding transition segments:

$$D_{cr_{12}}(i,j) = \alpha D_{c_{12}}(i,j) + (1-\alpha)D_{r_{12}}(i,j) \tag{3}$$

where $\alpha \in [0,1]$. We set α = 0.5 as default to equally consider the two features. Figure 3(a) depicts a distance matrix ($D_{cr_{12}}$) of 2 clips chosen from Chinese pop songs: "real man" ($clip_1$), "Let's move it" ($clip_2$). The darker the color is, the more similar the segments are. Since the transition segment of $clip_1$ and the transition segment of $clip_2$ should be similar beat by beat, we trace the values diagonally by applying overlapping window with L_{min} to L_{max} beats long and compute the average value within each window. We pick the windows with the minimum average

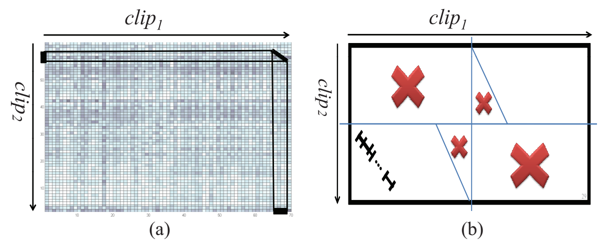


Figure 3. (a) The distance matrix of the two clips : "Real man" and "Let's move it." (b) The ignored areas.

value as the transition segments. Moreover, for the purpose of reducing the computational load and avoiding promptly switching clips, we consider only the last half of $clip_1$ and the first half of $clip_2$. Figure 3(a) shows the most similar segment we found. Figure 3(b) shows the ignored areas marked with thick crosses. The process is described by

$$[i^*, j^*, L^*] = \arg\min_{i,j,L} \frac{1}{L+1} \sum_{l=0}^{L} D_{cr_{12}}(i+l, j+l) \quad (4)$$

where $L \in [L_{min}, L_{max}]$, $i \geq \frac{N}{2}$, $j \leq \frac{M}{2}$, N and M are the total beat number of $clip_1$ and $clip_2$, respectively.

4.2 Transition Length Determination

In musical theory, tempo is defined as the speed of a given piece [8] , usually measured by the number of beats per minute (BPM). The tempo value at the i^{th} beat ($T(i)$) can be calculated as follows:

$$T(i) = \frac{60}{beat_{i+1} - beat_i} \quad (5)$$

where $beat_i$ and $beat_{i+1}$ are the time indices (in seconds) of the i^{th} and the $(i+1)^{th}$ beats of a clip extracted from the state-of-the-art tempo tracker: beatroot [9]. In order to gradually adjust the tempi from $clip_1$ to $clip_2$, the "transition length" should be long enough to let the difference between the adjacent $T(i)$ within the transition length small enough. An example is shown in Figure 4. To change the tempi from Tempo1 to Tempo2, the changing ratio (r_c) of the adjacent tempi within K beats is

$$r_c = \sqrt[K]{\frac{Tempo2}{Tempo1}} \quad (6)$$

By choosing a proper value of r_c, we can determine the minimum value of K. We adopt the concept of just noticeable difference [10] (JND) in the domain of psychoacoustics to determine r_c. JND is defined as the minimum difference of stimuli that people can perceive. These stimuli include loudness, tempo and pitch. According to *Weber's law*, the JND can be computed with the *Weber's Constant*. However, the *Weber's Constant* of tempo varies with changes in the environment. Thus, inspired by Thomas [11], we conduct experiments to find the JND of tempo on our music clip datasets. For quick (i.e. fast tempo) clips, we found out that the ratio of the tempi from 0.95 to 1.03 will not be perceived. For slow clips, the JND

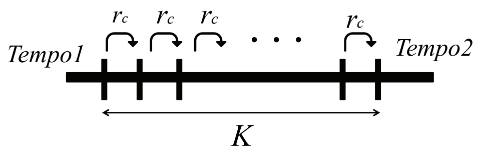


Figure 4. Diagram for changing tempi within a length K.

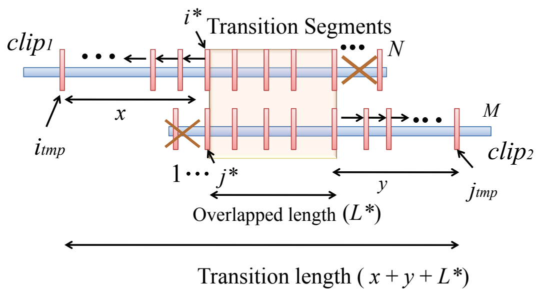


Figure 5. The sketch map of finding the transition length.

range is from 0.96 to 1.04. Nevertheless, real world music clips may contain more than one tempo, e.g. the pieces with accelerando or ritardando. Therefore, we developed Algorithm 1 to find the transition length and the target tempi T_{t1}, T_{t2}. The procedure is also illustrated in Figure 5. Then, we use phase vocoder [12] to adjust the tempi from T_1, T_2 to T_{t1}, T_{t2} , respectively. Figure 6 shows the corresponding results of two song clips: "Let's move it" and "Real man." The ratio of change appears like a linear decay because the ratios are usually very close to 1.

Algorithm 1

```
Input: the tempi of $clip_1$ and $clip_2$: $T_1(i), T_2(j)$, for $i = 1 \ldots N, j = 1 \ldots M$
1: for $x = 0$ to $i^*$, $y = 0$ to $(M - L^* - j^*)$ do
2:   $i_{tmp} \Leftarrow (i^* - x)$
3:   $j_{tmp} \Leftarrow (j^* + L^* + y)$
4:   $r_c \Leftarrow \sqrt[x+y+L^*]{\frac{T_2(j_{tmp})}{T_1(i_{tmp})}}$
5:   if $r_c$ is within JND then
6:     break
7:   end if
8: end for
9: $T_{t1}(i) \Leftarrow \begin{cases} T_1(i), & \text{for } i \leq i_{tmp} \\ T_1(i_{tmp}) \times r_c^{(i-i_{tmp})}, & \text{otherwise.} \end{cases}$
   $T_{t2}(j) \Leftarrow \begin{cases} T_2(j), & \text{for } i \geq j_{tmp} \\ T_2(j_{tmp}) \times r_c^{-(j_{tmp}-j)}, & \text{otherwise.} \end{cases}$
Output: the target tempi $T_{t1}, T_{t2}$
```

4.3 Synthesis Process

After changing the tempi, we align $clip_2$ at the start position of $clip_1$'s transition segment. Then, we apply crossfading on the transition segments. The effect of crossfading is achieved by using the log-transform method [3] because it better fits the actual human auditory system.

5. MUSIC FILTERING AND ORDERING

In this section, we describe the extra steps for dealing with more than two clips: music filtering and ordering.

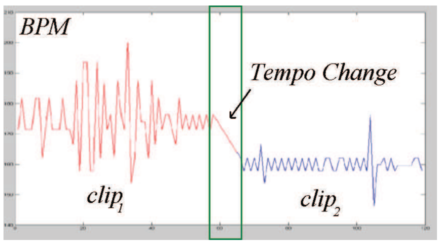


Figure 6. Tempi change in the transition length of the clips of "Let's move it" and "Real man."

5.1 Music Filtering

In order to reduce the probability of pasting quite distinct clips and the computational load in the ordering process (c.f. Section 5.2), we eliminate clips with extreme values by pair-wise comparison. A $clip_p$ is said to be extreme dissimilar and should be eliminated if there are more than half of the other clips ($clip_q$) in the database dissimilar to $clip_p$. The dissimilarity and similarity of any two clips are measured sequentially as follows.

5.1.1 Loudness Dissimilarity

The loudness dissimilarity is defined by the ratio $r_L(p,q)$ of the average loudness value of two clips $clip_p$ and $clip_q$, as shown in Eqn. (7)

$$r_L(p,q) = \frac{|Ld_p - Ld_q|}{Ld_p}, q = 1 \dots W, q \neq p \qquad (7)$$

where Ld_p and Ld_q are the average loudness values of the p^{th} and the q^{th} clips in the datasets and W is the total number of clips. The loudness values are computed by accumulating log-energy (in db) in all the frequency bands. $clip_p$ and $clip_q$ are said to be loudness-dissimilar if $r_L(p,q)$ is greater than a certain threshold. By Weber's law [10], the JND of loudness in db is 0.1, i.e. we will perceive the loudness change between $clip_p$ and $clip_q$ when the changing ratio ($r_L(p,q)$) is greater than 0.1. Since we have applied log-transform mechanism to smooth the change of volume in the sound effect module, we set the threshold value as 0.2 instead of the original strict standard.

5.1.2 Tempo Dissimilarity

Borrowing the concept in section 4.2, $clip_p$ and $clip_q$ are said to be tempo-dissimilar if there are not enough length for them to gradually adjusting the tempi from one to the other. The tempo dissimilarity is defined by $r_T(p,q)$:

$$r_T(p,q) = \left(\frac{T_q}{T_p}\right)^{\frac{1}{L_p+L_q}}, q = 1 \dots W, q \neq p \qquad (8)$$

where T_p and L_p are the minimal tempo value of the last quarter in $clip_p$ and the corresponding length from the position of T_p to the end of $clip_p$. Similarly, T_q and L_q are the maximal tempo value of the first quarter in $clip_q$ and the corresponding length, as shown in Figure 7. If $r_T(p,q)$ does not lie in the range of JND mentioned in section 4.2, there will not be enough transition length for changing tempi from $clip_p$ to $clip_q$ and they should be regarded as tempo-dissimilar.

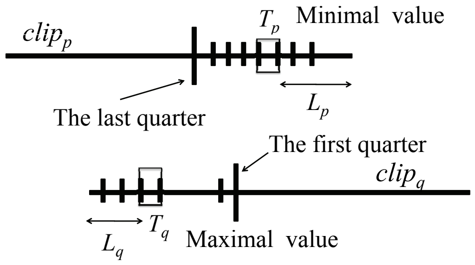


Figure 7. Finding tempo dissimilarity.

5.1.3 Chroma Histogram Similarity

In this module, we tend to avoid concatenating clips with different pitch distribution. The reason is as follows: the music paste will be unpleasant if we directly combine clips of different tonalities (e.g. C Major → e♭ minor) without modulation. Generally speaking, music clips with the same tonality contain similar pitch distributions. Thus, we construct a chroma histogram for each clip to represent its dominant pitch distribution and compare the clips by this histograms. For the 12 dimensional chroma vector ($\vec{C}_{pi}$) of the i^{th} beat in $clip_p$, we choose the index of its maximal value to represent the chroma dominant pitch (CM_{pi}) of this beat. That is,

$$CM_{pi} = \arg\max_u C_{pi}(u), u = 1 \dots 12 \qquad (9)$$

The chroma histogram of $clip_p$ (CH_p) is constructed from CM_{pi}'s. Inspired by the commonly used color histogram intersection method [14] in the computer vision field, we define the chroma histogram similarity between $clip_p$ and $clip_q$ by

$$S_H(p,q) = \frac{\sum_{u=1}^{12} \min(CH_p(u), CH_q(u))}{\sum_{u=1}^{12} CH_p(u)} \qquad (10)$$

where $q = 1 \dots W$, $p \neq q$. Analogous to the two previous subsections, $clip_p$ and $clip_q$ are viewed as dissimilar if $S_H(p,q)$ is less than 0.5.

5.2 Music Ordering

In the music ordering process, we tend to find an appropriate order to minimize the average distance values between each clip pair. For example, if the transition segments between $clip_1$ and $clip_3$ is not similar enough, maybe $clip_2$ can be the bridge of them. Besides, the transition segments from $clip_1$ to $clip_2$ may be less similar as compared with the transition segments from $clip_2$ to $clip_1$. Therefore, the ordering problem can be formulated as finding a path which goes through all clips in the datasets with minimum cost in the ordering matrix (D_o) defined as follows:

$$D_o(p,q) = \min_{i,j,L} \frac{1}{L+1} \sum_{l=0}^{L} D_{cr_{pq}}(i+l, j+l) \qquad (11)$$

where $L \in [L_{min}, L_{max}]$. To reduce the computation, we use a method analogous to the greedy algorithm but the path found cannot be guaranteed to reach the global optimum. The procedure is as follows:

	clip₁	***clip₂***	***clip₃***	***clip₄***
clip₁	0	0.3486	0.329	0.342
clip₂	0.3936	0	0.4704	0.4577
clip₃	0.2609	0.537	0	0.4806
clip₄	0.2898	0.4826	0.3732	0

Figure 8. An example of the ordering matrix for 4 clips.

1. Find the minimum value in the ordering matrix and set the corresponding two clips as the initial clips.
2. Find the minimum value in the row that corresponding to the last clip in the order found previously (each clip can only be visited once) and then add the corresponding clips to the order.
3. Repeat step 2 until all the values in the target row are larger than a predefined threshold or all clips have been visited.

Figure 8 shows an example of an ordering matrix constructed by four clips. First, we look for the minimum value in the matrix: 0.2609. We set the order as $3 \rightarrow 1$. Then, we check the values of first row: $\{0, 0.3486, 0.3290, 0.3420\}$. Since the first entry (0) represents $clip_1$ goes to $clip_1$ itself and the third entry (0.3290) means $clip_1$ goes to $clip_3$ again, we would not consider these two values. We find the minimum value of the rest: $\{0.3486, 0.3420\}$ is 0.3420. Thus, the order becomes $3 \rightarrow 1 \rightarrow 4$. Next, we check the fourth row and find 0.4826 is the only left value, so we compare it with the predefined threshold. If it is smaller than the threshold, the order would become $3 \rightarrow 1 \rightarrow 4 \rightarrow 2$. Otherwise, we would not concatenate $clip_2$ and the order would be just $3 \rightarrow 1 \rightarrow 4$. Currently, the threshold is 0.5.

6. EXPERIMENTS AND USER EVALUATIONS

The experiments are conducted on the basis of user evaluations. In order to reduce the impact of the prejudices, evaluators will not be informed the methods used in the testing sequences.

6.1 Overlap Length Discussion

Assuming that the smoothness of the results depends on the overlap length, we let 15 evaluators judge the music pastes with different overlap lengths. 8 sets of clips ($\approx$ 40 secs/clip) from different types of Chinese pop songs are used. We generate music pastes with 2 overlap lengths (force L^* = 4, 12 beats) and give each of them three different α values (c.f. Eqn. (3)) in the transition segments finding process. Figure 9 describes the overall results. The vertical axis represents the percentages of how many people prefer each method. We found that results with longer overlap length aren't really more acceptable than the shorter ones. The reason is probably that the similarity of transition segments decreases as the overlap length grows. Another observation is that the evaluator's acceptance varies with the types of the music clips. For instance, the accepted overlap length between two rap clips may be shorter than those of two lyric clips. Besides, over 60% of the evaluators preferred 4 beats as the overlapping length. Hence, we set the default overlap length to 4 beats long in the next section to compare the influence of different similarity measurements.

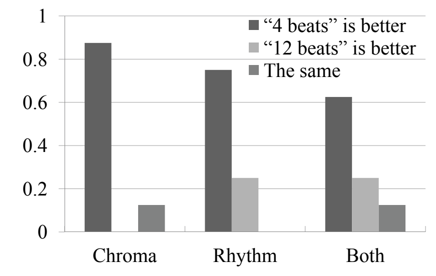


Figure 9. Overlap length comparison results.

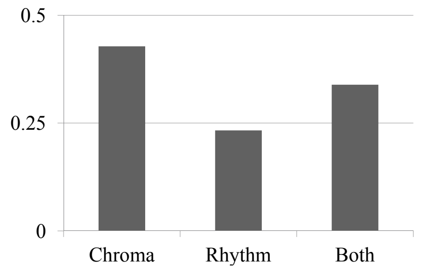


Figure 10. Comparison of 3 measurements.

6.2 Comparison of different similarity measurements

The similarity measurements we used for comparison are chroma only, rhythm only and both chroma and rhythm, i.e. α = 0, 1, 0.5. We utilized 8 sets of clips from songs in different languages. Fifteen evaluators gave scores from 1 to 10 to represent their satisfactions (higher score means better satisfaction) with respective to the feeling of intrusion. Figure 10 shows the percentages of how many people prefer each method. We found that chroma may be the most preferred measurement. Therefore, we choose the chroma measurements to conduct the following comparison with automated DJ.

6.3 Comparison with Automated DJ

In the automated DJ system [4], we can only use the music clips existing on the Amazon website and they should overlap at least 2 seconds. Thus, we selected 5 sets of pop songs available on Amazon and set the overlap length to 8 beats ($\approx$ 2 secs). Each set consists of two music pastes, one is generated by automated DJ and the other is by our approach. Seventeen evaluators participate in choosing their preferable method. Similarly, Figure 11 shows the percentages of how many people prefer each method. Overall speaking, the music pastes generated by our approach are promising and preferred as compared to those generated by the automated DJ. The acceptances may vary with different sets. For instance, our method is superior to automated DJ

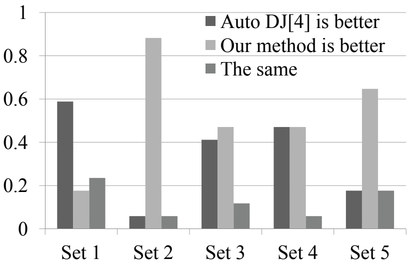


Figure 11. Comparisons with Automated DJ.

in a great level for set 2. The reason is that set 2 is a combination of a female voice and a male voice singing in quite different pitches. The results will be a little bit intrusive if we just concatenate these clips by rhythm-similar segments. Instead, we choose pitch-similar segments where the pitch of female voice downwards and the pitch of male voice upwards. The results would be more pleasant to hear.

6.4 Discussion

According to the above experimental results, we discovered the following factors affecting people's feeling toward the music paste: **(i) Language and lyrics.** The music pastes with unfamiliar languages will be probably more acceptable. In our datasets, half of sets are with unfamiliar languages to the evaluators. And 75% of this kind of pastes are scored higher than the average scores. This is probably because the intrusiveness increases when the lyrics of clips in familiar language conflict with each other. In contrast, it is not easy for evaluators to perceive the transition in clips with unfamiliar languages. **(ii) The ending position of phrases in the clips.** The music pastes will be probably more acceptable if the clips are transformed at the ending position of phrases. We have gathered statistics on our experiment datasets. There are 50% of the sets fit the mentioned condition. The scores are all higher than the average scores of all sets. The reason is probably that people's anticipation for the ending phrases will smooth the intrusiveness at the transition. **(iii) Familiarity with the music clips.** The music pastes are probably less acceptable if the evaluators have heard the music clips before. 71% of the evaluators gave higher score to unfamiliar music pastes than familiar ones. **(iv) The influence of vision.** The transition in music paste would be less noticeable if vision information involves. We combined one of our music pastes with a photo slideshow and let the evaluators view and listen again. Over 90% of evaluators gave higher scores to it because they almost did not notice the transition.

7. CONCLUSION AND FUTURE WORKS

In this paper, we provide a tool for automatically choosing proper music clips from a given audio collection and combining the chosen clips as euphonious as possible. We employ common auditory music features and borrow the concept from distance matrix to determine the transition segment and choosing music clips. The transition length is determined by Weber's Law. Besides, we apply phase vocoder to adjust the audio files and use cross-fading in synthesis process. Moreover, we conduct subjective evaluations on the correlation between pasting methods and auditory quality of combined clips. The overall experiment results show that the generated music pastes are acceptable to humans.

There are rooms for improving the proposed system. First, the pasting method is restricted to clips with similar enough transition segments. Perhaps the clips can be connected by automatically generating appropriate intermezzo or bridge music. Second, the proposed work can be meliorated by the improvement of music analysis techniques. More similarity measurements closer to style-similarity (timbre, rhythm) would improve the filtering process. Furthermore, more representative auditory features and similarity measurements, techniques for music structure analysis and phrase boundary extraction would help the process of locating transition segments. Third, studies on the variant overlapped length range in the transition segments are still worth investigating while currently the whole transition segments are overlapped. In the future, we will continue our investigation in these directions.

8. REFERENCES

[1] G. Loy: *Musimathics*, pp. 295–296, 347–350, The MIT Press, 2006.

[2] D. Cope: *Experiments in Musical Intelligence*, Madison, WI: A-R Editions, 1996.

[3] T. Jehan: "Creating music by listening," PhD thesis, MIT Media Lab, Cambridge, MA, 2005.

[4] T. Jehan: "This is My Jam," visited at Feb. 18, 2008; `http://thisismyjam.com/`

[5] M. Cooper, and J. Foote: "Automatic Music Summarization via Similarity Analysis," *Proceedings of the International Symposium on Music Information Retrieval* (ISMIR '02), Paris, France, 2002.

[6] C. A. Harte and M. B. Sandler: "Automatic chord identification using a quantised chromagram," *Proceedings of the Audio Engineering Society*, Spain, 2005.

[7] M. Cicconet: "Rhythm features," visited at Dec. 13, 2008; `http://w3.impa.br/~cicconet/cursos/ae/spmirPresentation.html`

[8] "Virginia Tech Multimedia Music Dictionary," visited at July 31, 2009; `http://www.music.vt.edu/musicdictionary/`.

[9] S. Dixon: "Evaluation of the Audio Beat Tracking System BeatRoot," *Journal of New Music Research,* Vol. 36, No. 1, pp. 39–50, 2007;

[10] G.T. Fechner: *Elements of psychophysics 1,* Holt, Rinehart & Winston, New York, 1860.

[11] Kim Thomas: "Just Noticeable Difference and Tempo Change," *Journal of Scientific Psychology,* May 2007.

[12] M. Dolson: "The phase vocoder: a tutorial," *Computer Music Journal*, Vol. 10, No. 4, pp. 14–27, 1986.

[13] C.J. Plack and R.P. Carlyon: "Loudness perception and intensity coding," *Hand book of Perception and Recognition 6: Hearing*, pp. 123-160, Editorial B.C.J. Moore, Academic Press, London, 1995.

[14] M. J. Swain and D. H. Ballard: "Color indexing," *International Journal of Computer Vision*, Vol. 7, No. 1, pp. 11–32, 1991.

MUSICAL BASS-LINE PATTERN CLUSTERING AND ITS APPLICATION TO AUDIO GENRE CLASSIFICATION

Emiru Tsunoo **Nobutaka Ono** **Shigeki Sagayama**
Graduate School of Information Science and Technology
The University of Tokyo, Japan
{tsunoo, onono, sagayama}@hil.t.u-tokyo.ac.jp

ABSTRACT

This paper discusses a new approach for clustering musical bass-line patterns representing particular genres and its application to audio genre classification. Many musical genres are characterized not only by timbral information but also by distinct representative bass-line patterns. So far this kind of temporal features have not so effectively been utilized. In particular, modern music songs mostly have certain fixed bar-long bass-line patterns per genre. For instance, while frequently bass-lines in rock music have constant pitch and a uniform rhythm, in jazz music there are many characteristic movements such as walking bass. We propose a representative bass-line pattern template extraction method based on k-means clustering handling a pitch-shift problem. After extracting the fundamental bass-line pattern templates for each genre, distances from each template are calculated and used as a feature vector for supervised learning. Experimental result shows that the automatically calculated bass-line pattern information can be used for genre classification effectively and improve upon current approaches based on timbral features.

1. INTRODUCTION

Due to the increasing size of music collections available on computers and portable music players, the need for music information retrieval (MIR) has surged recently. In particular, automatic genre classification from audio is a traditional topic of MIR and provides a structured way of evaluating new representations of musical content. In this task, not only instrumental information but also a bass-line information is thought to be important. For instance, bass parts in most rock songs consist of root notes of the chords in a uniform rhythm. In comparison to this, bass parts in jazz songs have a lot of characteristic movements which are called walking bass. If such representative bar-long bass-line patterns in music can be captured automatically as templates, they can potentially be used to characterize different music genres directly from audio signals.

In previous research, timbral features, rhythmic features and pitch features have been used for audio genre classification [1]. In this work, the timbral features were the most dominant and the pitch features used were not limited to the bass register. Studies more related to bass part extraction have been presented [2] where musical melodic and bass notes were modeled with Gaussian mixture model (GMM) and estimated. Using this pitch estimation method, Marolt [3] worked on the clustering of melodic lines using GMM. Researches using bass-line information for genre classification include McKay [4] and Tsuchihashi [5]. However the bass-line features discussed were based on overall statistics and did not represent directly temporal information.

In this paper, we discuss an approach for clustering unit bass-line patterns from a number of audio tracks and propose a feature vector based on the distances from templates for the applicationto genre classification. First, we emphasize only harmonic sounds of the audio tracks and estimate measure segments which divide tracks into measures. Then we propose a clustering method specialized to bass-line patterns based on the k-means clustering algorithm. For the purpose of an application to audio genre classification, the scheme to extract feature vector based on the bass-line patterns which contain temporal information is suggested. Finally, the effectiveness of the proposed bass-line pattern information for genre classification is verified experimentally.

2. BASS-LINE PATTERN CLUSTERING

2.1 Challenges in Bass-line Pattern Clustering

Bar-long bass-line patterns are frequently common and characteristic of a particular genre. In order to extract those representative patterns from audio signals, there are several challenges to be cleared. Especially in modern popular music, pieces comprise of both harmonic and percussive sounds and percussive components might disturb the bass-line analysis. Additionally, the bar lines of the music pieces need to be estimated. Another problem is that unit bass-line patterns are shifted in pitch according to the chord played. For example, a pattern consists of only root notes in a uniform rhythm, all notes in this pattern need to be pitch-shifted by the same amount of notes according to the chord, because the root note changes accompany with the chord changes.

Therefore the problems in extraction of representative bar-long patterns can be summarized as the following three problems:

I. audio signals may contain not only harmonic sounds but also percussive sounds,

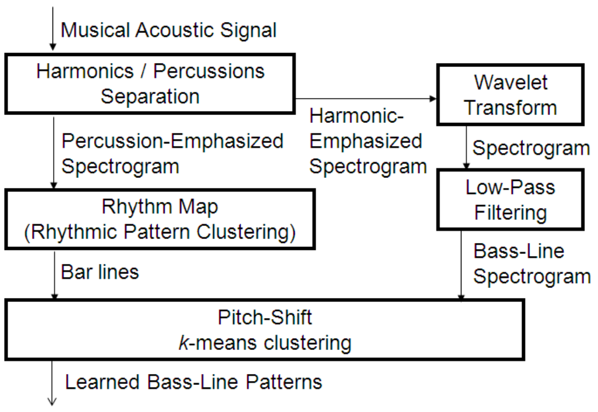

Musical Acoustic Signal
Harmonics / Percussions Separation
Wavelet Transform
Harmonic-Emphasized Spectrogram
Percussion-Emphasized Spectrogram
Spectrogram
Low-Pass Filtering
Rhythm Map (Rhythmic Pattern Clustering)
Bass-Line Spectrogram
Bar lines
Pitch-Shift k-means clustering
Learned Bass-Line Patterns

Figure 1. The flow diagram of the system.

II. measure segmentation is to be estimated, and
III. bass-line patterns are pitch-shifted according to chords.

In the next subsections, we describe our approach to solving these challenges. Fig. 1 illustrates the flow of the algorithm.

2.2 Emphasizing Harmonic Components

Generally, harmonic and percussive sounds are mixed in the observed spectrograms of audio pieces (the problem I). Therefore in order to perform bass-line pattern analysis it is useful to separate these components as a preprocessing step. We utilize the harmonic/percussive sound separation (HPSS) technique which we have proposed [6] that is based on the difference of general timbral features. By looking at the upper left figure in Fig. 2, a typical instance of spectrogram, one can observe that harmonic components tend to be continuous along the temporal axis in particular frequencies. On the other hand, percussive components tend to be continuous along the frequency axis and temporally short. Mask functions for separating the two components (harmonic and percussive) are calculated following a maximum a priori (MAP) estimation approach using the expectation maximization (EM) algorithm. Applying this approach to the shown spectrogram, harmonic and percussive components are separated and harmonic ones are emphasized (harmonic and percussive components are shown in the upper right and the lower left of Fig. 2 respectively). In order to capture only the bass part we apply low pass filtering to the harmonic-only spectrogram.

2.3 Bar Line estimation Using Percussive Clustering Method

There are many possible ways to solve problem II which is the estimation of bar lines. One way is beat tracking [8] in which onset, chord changes and drum patterns are used as cues. Instead, in the case where it is unknown how many beats are in one measure or songs do not always start with the head of the measure, the pattern matching approach is rather useful to estimate bar lines. Therefore the rhythm map which we have proposed [9] is employed.

The rhythm map is an approach to estimate representative bar-long percussive patterns and its segmentation (i.e. bar lines) simultaneously. This algorithm also requires the source separation method shown in previous subsection as a preprocessing and deals with the percussive-emphasized spectrogram. By iterating dynamic programming (DP) matching and updating the templates used for DP matching based on the k-means-clustering-like update rules, both segmentation and templates themselves are updated. After convergence, the multiple percussive patterns in the input song are learned and the optimal segmentation is obtained. We use this estimated segmentation as bar lines.

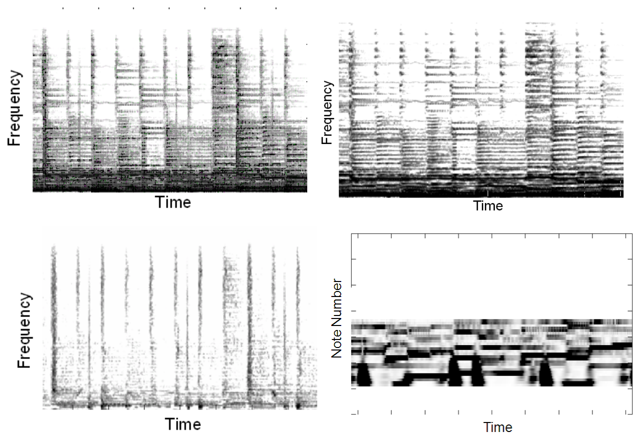


Figure 2. The original spectrogram (upper left), the harmonics-emphasized spectrogram (upper right) and the percussion-emphasized spectrogram (lower left) of a popular music piece (RWC-MDB-G-2001 No.6 [7]). The low-pass filtered logarithmic spectrogram calculated using wavelet transform from harmonics-emphasized spectrogram is shown in lower right figure.

2.4 Iterative Update of Bass-line Pattern Cluster

If there was no pitch shift (problem III) a simple k-means clustering approach can be used to estimate the representative bar-long bass-line patterns: Distances between each bar-long spectrogram pattern and centroid spectrogram patterns are calculated, and the centroids are updated by averaging the sample patterns. In order to deal with pitch-shift problem, we propose an new approach where every possible pitch-shift is compared in k-means framework.

Here we should note that both centroid template spectrogram and input spectrogram need to be logarithmic along frequency axis in order to consider pitch-shift. Because the musical notes are fashioned logarithmic in linear frequency domain. This kind of logarithmic spectrogram can be obtained by using wavelet transform. The lower right of Fig. 2 shows the logarithmic spectrogram which is processed low-pass filtering after Gabor wavelet transform whose frequency resolution is semitone (100 cents). The low-pass filtering (actually a band-pass filtering) was done by setting high and low frequency components to be zero. In this figure, the energy of bass drum is still dominant even after the source separation. That is because the bass drum sounds sometimes have long duration and the duration components also have the same feature with harmonic sounds which are continuous along the temporal axis. However these energies are thought not to be so harmful because when averaging spectrogram patterns to update templates, those parts become relatively small.

Mathematically, we modeled a bass-line template as a matrix and find out the most matched template for each bar-long spectrogram by calculating distances over all possible pitch-shift. The centroid bass-line pattern is defined as a matrix whose rows represent semitone numbers and whose columns are time instants. This matrix contains energy distribution of bar-long harmonic spectrogram. When the input piece has M measures, mth measure processed low-pass filtering can be written as X_m, the $I \times N$ matrix, where N is the number of semitones to capture in low frequency band and I is the resolution of time that divides a bar-long. On the other hand, kth bass-line pattern template can be written as the $I \times 2N$ matrix B_k. Since there are N notes to pitch-shift potentially, at least N rows of margin have to be provided.

As a distance measure of two bass-line pattern matrices, we use the following distance introduced by Frobenius norm:

$$d(X,Y) = \sum_{i=1}^{I}\sum_{n=1}^{N}\left(x_{i,n} - y_{i,n}\right)^2, \qquad (1)$$

where $x_{i,n}$ and $y_{i,n}$ are the (i,j)th entries of matrices X and Y. Considering pitch-shift, we can define the distance between input spectrogram X_m and template bass-line pattern B_k as

$$D(X_m, B_k) = \min_{1 \le n \le N} d(X_m, B_{k,n}) \qquad (2)$$

where $B_{k,n}$ is a $I \times N$ submatrix of B_k which is from nth row to $(n+N-1)$th row.

Like k-means algorithm, for every input bass-line pattern X_m, the distances are calculated according to Eq. (2) and classes and pitch-shift intervals are determined as

$$\hat{k}_m = \operatorname*{argmin}_{1 \le k \le K} D(X_m, B_k) \qquad (3)$$

and

$$\hat{n}_m = \operatorname*{argmin}_{1 \le n \le N} d(X_m, B_{\hat{k},n}). \qquad (4)$$

Then the update rule of a template pattern can be written as following, just by averaging patterns in a particular class,

$$B'_k = \frac{\sum_{m \in \{m|k=\hat{k}_m\}} S^{\hat{n}_m} E X_m}{\sum_{m \in \{m|k=\hat{k}_m\}} 1}, \qquad (5)$$

where E is an $N \times 2N$ extending matrix and S is a $2N \times 2N$ pitch-shift matrix respectively defined as

$$E = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \qquad (6)$$

$$S = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}. \qquad (7)$$

This update rule satisfies following equation as far as Euclidean distance metric is used,

$$\sum_{m \in \{m|k=\hat{k}_m\}} D(X_m, B'_k) \le \sum_{m \in \{m|k=\hat{k}_m\}} d(X_m, B'_{k,\hat{n}}) \le \sum_{m \in \{m|k=\hat{k}_m\}} D(X_m, B_{k,\hat{n}}), \qquad (8)$$

and convergence of iterative update is guaranteed. After the convergence, K centroid bass-line patterns B_k $(k = 1, \dots, K)$ are obtained.

3. BASS-LINE PATTERN FEATURE EXTRACTION

As the proposed clustering method applied to audio genre classification, there is a problem that the learned templates cannot be used directly. Ideally bass-line patterns for a particular genre would be fixed and would be automatically extracted perfectly. If that was the case then automatic genre classification could be performed simply by looking at which particular bass-line patterns are used in a music pieces by calculating the distance defined in Eq. (2). However in practice there is no guarantee that patterns are fixed for a particular genre or that their automatic extraction will be perfect. Therefore in many cases the bass-line patterns of a particular music piece will belong to more than one genre. To address these problems simultaneously we utilize a feature vector based on the distances from each template followed by statistical machine learning to automatically classify music genre. Supervised learning classifiers such as support vector machines (SVM) [10] can be used for this purpose.

One possible way to extract feature vector is calculating distances between every measure of input spectrogram and every template pattern following Eq. (2) and averaging them through whole an input piece. Even though there is a possibility for a music piece to belong to more than one genre templates, the distances between spectrogram in the input piece and learned templates are still affected, e.g., one measure spectrogram in blues song is close enough to the templates learned from blues collection even if its distance is not the smallest.

The mathematical definition of the feature vector is following. After bass-line pattern templates are learned, we have $K \cdot G$ templates when K templates are learned from G genres. Then the distances between input song which have M measures and learned templates are calculated. The averaged distances are obtained as follows:

$$d_l = \frac{1}{M}\sum_{m=1}^{M} D(X_m, B_l) \qquad (9)$$

where $1 \le l \le KG$ is the template number. The feature vector $\boldsymbol{x}$ can be written as

$$\boldsymbol{x} = (d_1, d_2, \dots, d_{KG})^T. \qquad (10)$$

We use this feature vector for a supervised learning classification to classify music genre.

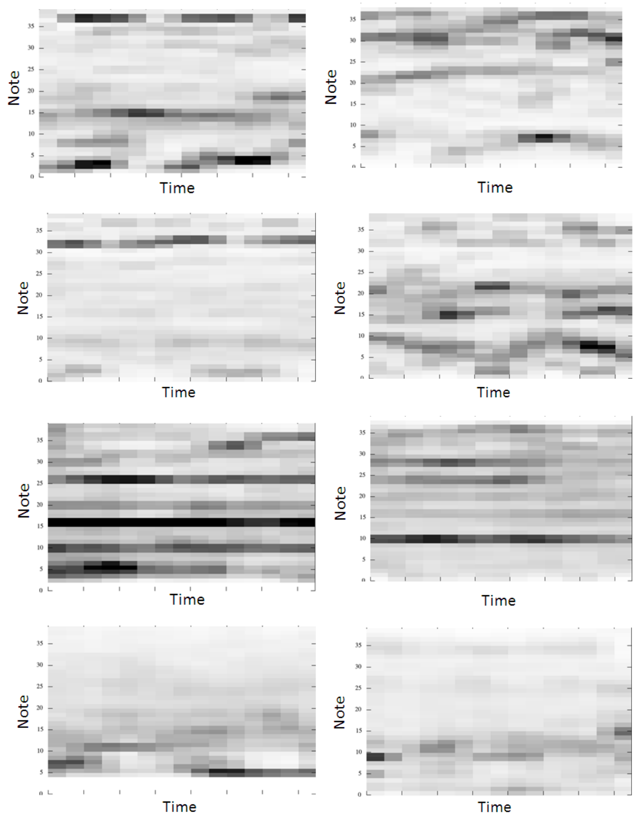


Figure 3. Two examples of learned bass-line templates from jazz, blues, rock, and hiphop, in descending order. While a lot of movements of bass notes are shown in jazz and blues, bass notes in rock are unchanged, and hiphop bass-line templates are quite sparse.

4. PROCEDURAL SUMMARY OF THE ALGORITHM

The overall algorithm can be summarized as follows:

1. Preprocessing
   (a) Emphasis of harmonic components using HPSS
   (b) Apply low-pass filtering
   (c) Estimate bar lines using rhythm map
2. Bass-line Pattern Clustering
   (a) Provide initial (random value) templates
   (b) Match the templates patterns with pitch-shifting by Eq. (2)
   (c) Update the template patterns following Eq. (5)
   (d) Iterate steps b and c until the convergence
3. Genre Classification
   (a) Calculate the distances between patterns (Eq. (10)) and use it as a feature vector characterizing a music pieces
   (b) Perform classification into genres using a machine learning technique

5. EXPERIMENTAL RESULTS

5.1 Dataset

Experiments with the proposed algorithms were conducted on the GTZAN dataset [1]. The dataset had 10 genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. The dataset had 100 songs per genre all of which were single-channel and sampled at 22.05kHz. All songs were processed harmonic-percussive sound source separation and wavelet transform. Then they were processed low-pass filtering and obtained the spectrogram only from 82.4Hz (E1) to 330.0Hz (E3). The reason we didn't use the spectrogram under 82.4Hz was the dominance of the energy of bass drums in that area.

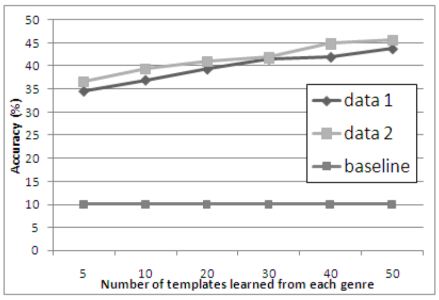


Figure 4. Classification accuracy using only bass-line features for each number of templates learned from one genre. The baseline accuracy of random classifier was 10.0%

5.2 Template Learning and Feature Extraction

First, common bass-line pattern templates were learned using the proposed algorithm for each genre. The proposed algorithm was implemented using audio processing framework *Marsyas*[1] which is open source software with specific emphasis on Music Information Retrieval (MIR) [11]. To generalize the learning templates part, we divided the dataset 50-50 into two parts randomly and obtained two sets of templates for each genre. In this experiment, the number of iteration was fixed to 15 times because it was enough to converge, and the number of the time resolution was fixed to 16.

The examples of learned templates of jazz, blues, rock and hiphop are shown in Fig. 3. While jazz bass patterns had some movements, rock patterns showed the horizontally straight lines. In blues bass-line templates, a typical swing rhythm is shown, and in hiphop templates, there are sparse bass notes because hiphop songs mostly have strong drum sounds but bass sounds are not so melodic.

After learning templates two sets of templates were obtained, and next, we extracted feature vector (Eq. (10)) using those template sets. The templates learned from data 1 were used to extract feature vectors of data 2, and vice versa.

5.3 Classification results

In order to train a classifier in the feature space, the "Weka" machine learning toolkit [12] was employed. All the results shown were based on 10-fold cross validation using a linear SVM as a classifier. The labeled data was split into 10 folds and each fold was used once for testing with the remaining 9 folds used for training the classifier to generalize the classification, for both of divided data sets.

[1] http://marsyas.sness.net/

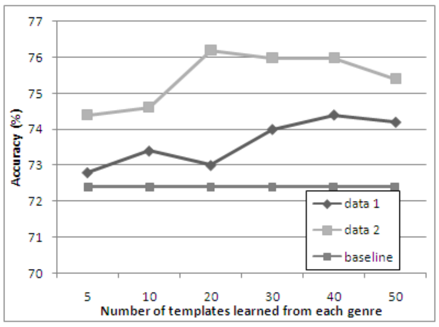


Figure 5. Classification accuracy using both bass-line and timbral features for each number of templates learned from one genre. The baseline accuracy of existing features was 72.4%.

Table 1. Genre classification accuracy using only bass-line pattern features and merged with timbral features.

Features	data 1	data 2
Baseline (random classifier)	10.0%	10.0%
Only bass-line (400 dim.)	42.0%	44.8%
Existing (Timbre, 68 dim.)	72.4%	72.4%
Merged (468 dim.)	74.4%	76.0%

The number of templates learned for each particular genre were decided experimentally. Fig. 4 shows the results using only the bass-line pattern features whose dimension was $10K$ when the number of templates learned from one genre was K. As can be seen the proposed features had enough information for genre classification and the accuracy improved as the number of templates was increased.

An existing state-of-the-art genre classification system which uses 68 dimensional timbral features like MFCC and spectrum centroids proposed by Tzanetakis was used for comparison. The system performed well on several audio classification tasks in MIREX 2008 [13]. Merging this timbral features and bass-line features, the classification accuracies shown in Fig. 5 were obtained. When the number of templates K was 40 the performance was the best, and when we increased the number K more than that the performance got worse. That was because the dimension of feature space became larger and curse of dimensionality was thought to occur. In particular case the number of templates K was fixed to 40 for each particular genre, the precise result is shown in Table 1 and the confusion matrices are shown in Table 2 and Table 3. These results were higher than existing genre classification systems that rely on timbral information and verify the effectiveness of the proposed bass-line patterns. One can see that classical was the most distinguished by the system and some reggae songs were mistaken for disco.

6. CONCLUSIONS

We discussed an approach for clustering common bar-long bass-line patterns for particular genres, and proposed a feature vector which represented relations to learned bass-line templates. We used HPSS technique to extract harmonic components from audio signals and rhythm map to estimate measure segmentation. After processing low-pass filtering, bass-line patterns were clustered using a new clustering method based on k-means clustering with pitch-shift. For audio genre classification, a new feature vector was defined as averaged distances from each template. Experiments over music pieces from various genres confirmed that the proposed algorithm can improve the accuracy of classification systems based on timbral information.

Future work includes more experiments with the parameters of the algorithms such as the resolution of time in templates. Additionally, other features than pattern distance vector can be devised. Combination with other features like percussive pattern can be done to improve further genre classification as well.

ACKNOWLEDGMENTS

The authors thank George Tzanetakis for sharing his dataset and some useful experimental discussions with us.

7. REFERENCES

[1] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transaction on Speech and Audio Processing*, 10(5):293–302, 2002.

[2] M. Goto. A real-time music-scene-description system: Predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43(4):311–329, 2004.

[3] M. Marolt. Gaussian mixture models for extraction of melodic lines from audio recordings. In *Proc. of ISMIR*, pages 80–83, 2004.

[4] C. McKay and I. Fujinaga. Automatic genre classification using large high level musical feature sets. In *Proc. of ISMIR*, pages 525–530, 2004.

[5] Y. Tsuchihashi and H. Katayose. Music genre classification from bass-part using som. In *IPSJ SIG Technical Reports*, volume 90, pages 31–36, 2006.

[6] N. Ono, K. Miyamoto, H. Kameoka, and S. Sagayama. A real-time equalizer of harmonic and percussive componets in music signals. In *Proc. of the 9th Int. Conf. on Music Information Retrieval*, pages 139–144, September 2008.

[7] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. Rwc music database: Music genre database and musical instrument sound database. In *Proc. of the 4th Int. Conf. on Music Information Retrieval*, pages 229–230, October 2003.

[8] M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, June 2001.

[9] E. Tsunoo, N. Ono, and S. Sagayama. Rhythm map: Extraction of unit rhythmic patterns and analysis of rhythmic structure from music acoustic signals. In *Proc. of ICASSP*, pages 185–188, 2009.

[10] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[11] G. Tzanetakis. *Marsyas-0.2: A Case Study in Implementing Music Information Retrieval System*, chapter 2, pages 31 – 49. Idea Group Reference, 2007. Shen, Shepherd, Cui, Liu (eds).

[12] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[13] Mirex 2008. http://www.music-ir.org/mirex/2008/.

Table 2. The confusion matrix of data 1 in the case 40 templates learned from each genre

classified as	bl	cl	co	di	hi	ja	me	po	re	ro
bl: blues	38	0	5	0	0	2	1	0	1	3
cl: classical	0	48	0	0	0	0	0	0	0	2
co: country	3	0	30	1	1	2	1	3	0	9
di: disco	0	0	1	36	1	1	1	2	2	6
hi: hiphop	0	0	1	6	34	0	1	2	4	2
ja: jazz	4	2	1	1	0	41	0	0	0	1
me: metal	0	0	2	1	0	0	43	0	0	4
po: pop	1	0	3	3	3	2	0	34	0	4
re: reggae	0	0	4	4	2	0	0	0	37	3
ro: rock	3	0	4	3	2	1	3	0	3	31

Table 3. The confusion matrix of data 2 in the case 40 templates learned from each genre

classified as	bl	cl	co	di	hi	ja	me	po	re	ro
bl: blues	38	0	5	2	0	2	2	0	1	0
cl: classical	0	47	0	0	0	3	0	0	0	0
co: country	4	0	40	0	0	0	0	0	0	6
di: disco	3	0	0	34	1	0	1	2	3	6
hi: hiphop	1	0	0	3	41	0	0	0	5	0
ja: jazz	3	1	0	0	0	43	2	0	0	1
me: metal	1	0	0	0	0	1	44	0	1	3
po: pop	4	0	1	2	1	0	0	35	3	4
re: reggae	4	0	0	9	4	2	0	3	27	1
ro: rock	6	0	3	5	0	0	3	2	0	31

UNSUPERVISED DETECTION OF COVER SONG SETS: ACCURACY IMPROVEMENT AND ORIGINAL IDENTIFICATION

Joan Serrà†, Massimiliano Zanin‡, Cyril Laurier†, and Mohamed Sordo†
† Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain.
‡ Universidad Autónoma de Madrid, Madrid, Spain.
joan.serraj@upf.edu, massimiliano.zanin@hotmail.com, {cyril.laurier,mohamed.sordo}@upf.edu

ABSTRACT

The task of identifying cover songs has formerly been studied in terms of a prototypical query retrieval framework. However, this framework is not the only one the task allows. In this article, we revise the task of identifying cover songs to include the notion of sets (or groups) of covers. In particular, we study the application of unsupervised clustering and community detection algorithms to detect cover sets. We consider current state-of-the-art algorithms and propose new methods to achieve this goal. Our experiments show that the detection of cover sets is feasible, that it can be performed in a reasonable amount of time, that it does not require extensive parameter tuning, and that it presents certain robustness to inaccurate measurements. Furthermore, we highlight two direct outcomes that naturally arise from the proposed framework revision: increasing the accuracy of query retrieval-based systems and detecting the original song within a set of covers.

1. INTRODUCTION

Cover song identification has been a very active area of study within the music information research (MIR) community over the last years [1]. Traditionally, cover song identification has been set up as a typical information retrieval (IR) task where queries are processed in a batch mode [2] (p. 74): the user submits a query (a song) and receives an answer back (a list of songs ranked by their relevance to the query). Such a setup has conditioned the way of implementing and evaluating cover song identification systems [1, 3].

Here we take a new qualitative approach by considering cover song sets instead of isolated cover song queries. More concretely, we automatically identify sets (or groups) of covers of the same underlying musical piece in a music collection. We do so by utilizing grouping algorithms on top of an existing cover song identification system. We focus on unsupervised clustering [4, 5] and community detection [6, 7] algorithms.

The usage of grouping algorithms can be intuitively justified from multiple perspectives. First, grouping algorithms are a natural choice given the output of current cover song identification systems. Because this usually consists in a set of pairwise distances [1], we can directly assume that the preliminary issues of a typical pattern grouping task [5], namely feature extraction and distance measurement, are appropriately dealt with. Second, grouping algorithms may help in obtaining more coherent answers for isolated queries. In particular, the answers to any query song belonging to a given cover set would coherently contain the other songs in the set (notice that this property is not ensured by the distance measurements nor by batch-mode query systems). Third, grouping algorithms may profit from second order cover song associations. For instance, if cover song pairs s_i, s_j and s_j, s_k are independently detected, the cover song relation between s_i and s_k could automatically be inferred. This way, the system would take advantage of these collaborative effects and, among other things, increase the overall accuracy. Fourth, grouping algorithms can provide insightful clues to the study of inter and intra-group relations (e.g. by using hierarchical clustering algorithms [4, 5]).

The Music Information Retrieval Evaluation eXchange (MIREX) provides a batch-mode query framework for evaluating cover song identification systems [2]. Nonetheless, some participants have started moving towards the cover set detection framework. This framework has been indirectly and scantily introduced in [8] and, simultaneously, in our previous work in [9]. In [8], a multidimensional scaling analysis was performed on the basis that the configuration of the music collection under study was known (i.e. the number of cover sets and their cardinality was a priori defined, and the latter was assumed to be constant for all sets). This analysis was shown to substantially increase the final system's identification accuracy. A comparable increase was also achieved by the post-processing step mentioned in [9], whose details we now disclose.

Below, we first present the grouping algorithms that we use for detecting cover song sets (Sec. 2). We then summarize the followed methodology (Sec. 3) and subsequently present the obtained results (Sec. 4). We also show two

[1] In this article we pragmatically use the term *distance* to refer to any similarity or dissimilarity measure between cover songs.
[2] `http://www.music-ir.org/mirex/2008/index.php/Audio_Cover_Song_Identification`

natural outcomes of detecting cover sets (Sec. 5): increasing the accuracy of batch-mode query systems (Sec. 5.1) and detecting the original song (Sec. 5.2). We finally close the article with a conclusions section (Sec. 6).

2. STUDIED ALGORITHMS

2.1 K-medoids

The K-medoids algorithm is a common choice when the computation of means is unavailable, as it solely operates on pairwise distances and can exhibit some advantages compared to the standard K-means algorithm [4]. We employ the K-medoids implementation of the TAMO[3] package, which incorporates several heuristics[4] to achieve an optimal K value [4]. We use the default parameters and try all possible heuristics provided in the aforementioned implementation.

2.2 Hierarchical clustering

We also consider four representative agglomerative hierarchical clustering methods [4, 5]: single linkage, complete linkage, group average linkage (UPGMA), and weighted average linkage (WPGMA). We use the hcluster[5] implementation with the default parameters, and we try different cluster validity criteria [4] such as[6] checking descendants for inconsistent values, or considering the maximal or the average inter-cluster cophenetic distance.

2.3 Proposed method 1

The present and subsequent methods perform community detection [6, 7] on a complex network [10]. A weighted complex network [11] can be easily built from pairwise distances. Given a music collection $S=\{s_i\}, i=1,...,N_S$, with N_S songs, we query a cover song identification system for each song and obtain a set of answers $A=\{A_i\}$, where A_i contains N_{A_i} tuples $\{s_j, d(s_i,s_j)\}$, $s_j \in S$, ranked from low to high according to the provided distance measure d. In our case, N_{A_i} can be different for each A_i and it can be significantly lower than N_S. As we do not expect cover songs to have high distances or ranks in A_i, we determine N_{A_i} by applying a distance and a rank threshold d_{Th} and r_{Th}, respectively. From A, we construct a graph G with N_S vertices ($V=\{v_i\}$, $V \leftrightarrow S$) and N_{A_i} weighted edges for each vertex (an edge with a weight $w_{i,j}=\frac{1}{d(s_i,s_j)+\epsilon}$, ϵ being an arbitrary small constant, e.g. $\epsilon=0.01$, is assigned between vertices v_i and v_j if $s_j \subset A_i$ or $s_i \subset A_j$).

The method performs community detection by just looking at connected vertices in G in such a way that all connected vertices are assigned to the same community. Therefore, this algorithm strongly relies on d_{Th} and/or r_{Th} parameters. This approach presents some analogies with the common nearest neighbor clustering approach [5].

[3] http://fraenkel.mit.edu/TAMO

[4] http://fraenkel.mit.edu/TAMO/documentation/TAMO.Clustering.Kmedoids.html

[5] http://code.google.com/p/scipy-cluster

[6] http://www.soe.ucsc.edu/~eads/cluster.html

2.4 Proposed method 2

The aforementioned approach could be further improved by reinforcing triangular connections in the complex network before the last step of checking for connected vertices. Following the intuitive reasonings advanced in Sec. 1, the algorithm proposed here tries to reduce the "uncertainty" generated by triplets of vertices connected by two edges. In other words, it tries to reinforce coherence in a triangular sense.

If v_i, v_j and v_j, v_k are respectively connected, we can induce more coherence by either creating a connection between v_i and v_k (i.e. forcing the existence of a triangle), or by deleting one of the existing edges. This coherence can be measured through an objective function f_{O} which considers complete and incomplete triangles in the whole graph G. We define f_{O} as a weighted difference between the number of complete triangles N_Θ and the number of incomplete triangles N_Φ (three vertices connected by only two links) that can be computed from a pair of vertices: $f_{\text{O}}(N_\Theta, N_\Phi)=N_\Theta-\alpha N_\Phi$. The constant α, which weights the penalization for having incomplete triangles in G, is set experimentally.

The method starts by building a complex network G as described in the previous section. Then, for each pair of vertices v_i, v_j, the value of f_{O} is calculated for two situations: (i) when an edge between v_i and v_j is artificially created and (ii) when such an edge is deleted. Then, the option which maximizes f_{O} is preserved and the adjacency matrix of G is updated as necessary. The process of assigning cover sets is again the connected vertices criterion.

2.5 Proposed method 3

We can substantially reduce the computation time of proposed method 2 by considering for the computation of f_{O} only those vertices whose connections seem to be uncertain. If the distance between two songs is extremely high or low, this means that the cover song detection system has clearly detected a match or a mismatch. Accordingly, we just consider the pairs of vertices whose edge weight is around $w_{\text{Th}}=\frac{1}{d_{\text{Th}}+\epsilon}$. In particular, for each vertex v_i, a pre-selection of adjacent vertices is performed according to a certain weight margin w_{Ma}, which we set manually. This way, v_j is associated to v_i for further processing iff $|w_{i,j}-w_{\text{Th}}|<w_{\text{Ma}}$, where $|\cdot|$ indicates absolute value. The process of building the initial complex network and assigning cover sets is the same as in Sec. 2.3.

2.6 Modularity optimization

We also consider the method in [12], which extracts the community structure from large networks based on the optimization of the network modularity [6, 7, 12]. This algorithm outperforms all other known community detection methods in terms of computational time while maintaining a high accuracy [12]. We use the implementation by Aynaud[7] and we build the initial network as in Sec. 2.3.

[7] http://perso.crans.org/~aynaud/communities/index.html

3. METHODOLOGY

3.1 Data

For the generality of our experiments, we employ two sources of data: artificial and real-world symmetric distance matrices. Artificial distances are randomly generated given a predefined noise level σ_ξ. More concretely, the distance $d(s_i,s_j)=d(s_j,s_i)$ between songs s_i and s_j is drawn from a normal distribution $\mathcal{N}(\mu,\sigma)$ with mean μ and standard deviation σ such that

$$d(s_i,s_j)=\begin{cases}0 & \text{if } i=j,\\ |\mathcal{N}(0,\sigma_\xi)| & \text{if } i>j \text{ and } s_i,s_j \text{ covers},\\ 1-|\mathcal{N}(0,\sigma_\xi)| & \text{otherwise.}\end{cases} \tag{1}$$

Real-world distances are provided by a state-of-the-art cover song identification system [13].

3.2 Experimental setups

A cover song music collection can be characterized by certain parameters constituting a setup [1]: the total number of songs N_S, the number of cover sets N_C the collection includes, the cardinality C of these cover sets, and the number of added noise songs [8] N_N. Because some setups can lead to wrong accuracy estimations [1], it is safer to consider several of them, including fixed and variable cardinalities. In our experiments we use the setups summarized in Table 1. For real-world data we use an extension of the music collection described in [13] which comprises a variety of genres and styles, as well as different types of covers. The characteristics of this music collection correspond to setup 3. For other setups we simply sample cover sets from setup 3 and repeat the experiments N_T times (number of trials, average accuracies reported). We either sample cover sets with a fixed cardinality ($C=4$, the expected cardinality of setup 3) or without fixing it (variable cardinality, $C=\nu$). When using artificial data, we construct the distance matrix following Eq. (1). As fixed cardinality we use $C=4$ as well, and as variable cardinality we take $\nu\sim\lfloor 2+e(1/2)\rfloor$, where $\lfloor\cdot\rfloor$ denotes floor value and $e(1/\lambda)$ corresponds to an exponential distribution [9] with rate parameter λ.

3.3 Evaluation measures

To evaluate batch-mode query systems we employ the mean of average precisions (MAP) over all queries [3, 14]. The average precision for a query s_i (AP$_i$) is calculated from the retrieved answer A_i as $\text{AP}_i=\frac{1}{C-1}\sum_{r=1}^{N_S-1}P_i(r)I_i(r)$, where P_i is the precision of the sorted list A_i at rank r, $P_i(r)=\frac{1}{r}\sum_{l=1}^{r}I_i(l)$, and I_i is a relevance function such that $I_i(z)=1$ if the song with rank z in A_i is a cover of s_i, $I_i(z)=0$ otherwise.

[8] By *noise songs* we mean songs that do not belong to any cover set included in the collection.

[9] An exponential distribution is used to imitate the distribution of C with setup 3 (see [13]). With $\lambda=2$, an expected value $\langle\nu\rangle=4$ is obtained.

Setup	Parameters				
	N_C	C	N_N	N_S	N_T
1.1	25	4	0	100	20
1.2	25	ν	0	$\langle 100\rangle$	20
1.3	25	4	100	200	20
1.4	25	ν	100	$\langle 200\rangle$	20
2.1	125	4	0	500	20
2.2	125	ν	0	$\langle 500\rangle$	20
2.3	125	4	400	900	20
2.4	125	ν	400	$\langle 900\rangle$	20
3	525	ν	0	2135	1

Table 1. Experimental setup summary. The $\langle\cdot\rangle$ delimiters denote expected value.

To evaluate cover set detection we resort to the classical F-measure with even weighting [2, 14]: $F=\frac{2PR}{P+R}$. Here P and R correspond to precision and recall, respectively. For our evaluation, we compute these quantities independently for all songs and average afterwards. More concretely, for each song s_i, we count the number of true positives TP$_i$ (i.e. the number of cover songs of s_i belonging to the the same group [10] as s_i), the number of false positives FP$_i$ (i.e. the number of songs belonging to the same group as s_i that are not covers of s_i), the number of false negatives FN$_i$ (i.e. the number of cover songs of s_i that do not belong to the same group as s_i), and average $P_i=\frac{\text{TP}_i}{\text{TP}_i+\text{FP}_i}$ and $R_i=\frac{\text{TP}_i}{\text{TP}_i+\text{FN}_i}$ across all N_S songs [11].

4. RESULTS

4.1 Artificial data

We first evaluate the algorithms' accuracy as a function of the noise level σ_ξ introduced to the artificial data for setups 1.1 to 1.4. Before computing the reported accuracies, we independently performed a parameter optimization for each algorithm [12], σ_ξ, and setup. Then, using the optimal parameters found for a given σ_ξ, we plot average F over 20 trials versus σ_ξ (Fig. 1). In general, we observe that the accuracy for all algorithms starts to drop for $\sigma_\xi>0.2$ until it reaches an $F<0.3$ for $\sigma_\xi>0.5$. We also see that the K-medoids and single-linkage algorithms are less robust to noise than the others. Low accuracies for the K-medoids algorithm might be explained by the difficulty to automatically set the correct K value. Furthermore, cover sets with variable cardinality, such as the ones used for setups 1.2 and 1.4, might further decrease the accuracy [4]. Low accuracies for the single linkage algorithm with respect to other hierarchical clustering algorithms confirm the findings in the literature [5]. UPGMA, and WPGMA accuracies are slightly higher than other algorithms under noise levels $\sigma_\xi\in[0.2,0.4]$.

[10] Through this subsection by *group* we mean the cover set detected by the evaluated algorithm.

[11] Note that, unlike other clustering evaluation measures [15], F is not computed on a per-cluster basis, but on a per-song basis.

[12] This parameter optimization was not critical to achieve near-optimal accuracies (see Sec. 4.2). We did not consider proposed method 2 at this stage due to its computational complexity (see Sec. 4.3).

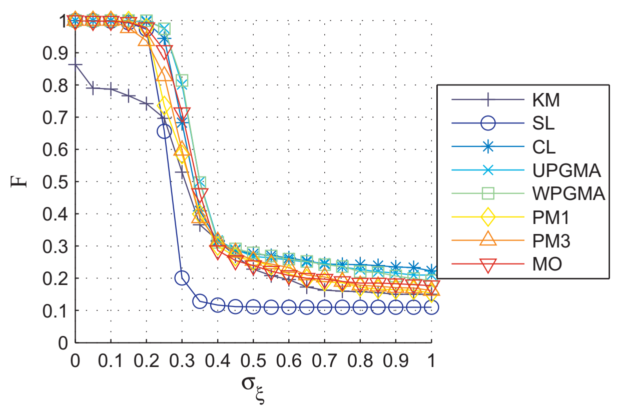


Figure 1. Accuracy under different noise levels for setup 1.4. Other setups lead to similar curves. Here and in subsequent tables and figures KM stands for K-medoids, SL for single-linkage, CL for complete-linkage, UPGMA for group average linkage, and WPGMA for weighted average linkage, PM for proposed method, and MO for modularity optimization.

Algorithm	Setup			
	2.1	2.2	2.3	2.4
KM	0.517	0.497	0.520	0.500
SL	0.656	0.690	0.654	0.683
CL	0.816	0.820	0.814	0.821
UPGMA	**0.895**	**0.900**	**0.897**	**0.902**
WPGMA	0.879	0.889	0.883	0.893
PM1	0.713	0.723	0.716	0.718
PM3	0.665	0.698	0.668	0.704
MO	0.721	0.735	0.716	0.744

Table 2. Accuracy (F) for artificial data with $\sigma_\xi = 0.25$.

To better assess the algorithms' performance, we show the accuracies achieved with setups 2.1 to 2.4 under a fixed noise level of $\sigma_\xi = 0.25$ (Table 2). Here, differences between accuracies can be better estimated, as we are employing a bigger music collection and quite a high noise level. We see that UPGMA and WPGMA definitely perform best under the specified σ_ξ.

4.2 Real-world data

To evaluate the algorithms' accuracy with real-world data we independently optimized all possible parameters for each algorithm on setups 1.1 to 1.4. Within this pre-analysis, we saw that the definition of a distance threshold [13] was, in general, the only critical parameter for all algorithms. Apart from this, all other parameters turned out not to be critical for obtaining near-optimal accuracies. Methods that had specially broad ranges of these near-optimal accuracies were K-medoids, proposed method 2, and all considered hierarchical clustering algorithms.

We report the accuracies with optimal parameters for setups 2.1 to 3 (Table 3). We see that accuracies for proposed methods 1 and 3 are comparable to the ones achieved

[13] Either cophenetic distances, d_{Th}, or r_{Th} (see Sec. 2).

Algorithm	Setup				
	2.1	2.2	2.3	2.4	3
KM	0.637	0.642	0.656	0.666	*n.c.*
SL	0.776	0.783	0.833	0.828	0.676
CL	0.777	0.768	0.860	0.853	0.756
UPGMA	0.797	0.812	0.865	**0.875**	**0.796**
WPGMA	0.800	0.804	**0.866**	0.862	0.788
PM1	0.804	**0.813**	0.853	0.853	0.751
PM2	0.761	0.738	*n.c.*	*n.c.*	*n.c.*
PM3	0.788	0.790	0.841	0.848	0.733
MO	**0.808**	0.809	0.856	0.858	0.762

Table 3. Accuracy (F) for real-world data. Due to algorithms' complexity, some results were not computed (denoted as *n.c.*, see Sec. 4.3).

by the other algorithms and, in some setups, even better. Overall, we corroborate the findings of the previous section, although differences between algorithms are not so large now. We hypothesize that these similar (as well as near-optimal) accuracies are due to the relatively good distance measure provided by the employed cover song identification system (we have already seen that these differences get stressed with artificial data).

4.3 Time performance

In the application of these techniques to big real world music collections, computational complexity is of great importance. To qualitatively evaluate this aspect, we report the average amount of time spent by each algorithm to achieve a solution for each setup (Fig. 2).

We see that K-medoids and proposed method 2 are completely inadequate for processing collections with more than 2000 songs (e.g. setup 3). The steep rise in the time spent by hierarchical clustering algorithms to find a cluster solution for setup 3 also raises some doubts as to the usefulness of these algorithms for huge music collections. Furthermore, the hierarchical clustering algorithms, as well as the K-medoids algorithm, take the full pairwise distance matrix as input. Therefore, with a music collection of, say,

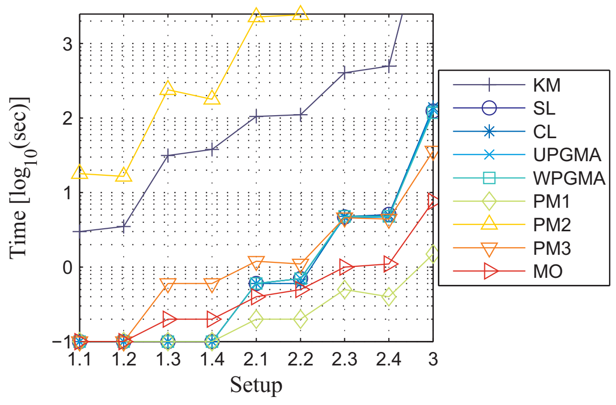


Figure 2. Average time performance for each considered setup. Algorithms were run with an Intel(R) Pentium(R) 4 CPU 2.40GHz with 512M RAM.

10 million songs, this distance matrix might be difficult to handle. In contrast, algorithms based on complex networks such as modularity optimization and proposed methods 1 and 3, only need a single list of answers A. Moreover, the length of the elements of this list can be very small due to d_{Th} and r_{Th} defined above (e.g. in our tests we found unnecessary to consider $r_{\mathrm{Th}} > 3$). It should also be noticed that the resulting network is sparse, i.e. the number of links is much lower than ${N_S}^2$ [10] and, therefore, calculations on such graphs can be strongly optimized both in memory requirements and computational costs as demonstrated, for instance, by [12]. Thus, methods based on complex networks, despite not achieving the highest accuracies for cover set detection, represent a robust and scalable option for processing big music collections.

5. OUTCOMES

5.1 Accuracy increase

In this section we show that the detection of cover song sets can improve the accuracy of batch-mode query systems. In particular, we study the relative MAP increase for the best cover set detection algorithms found. For comparison purposes, we take the output A of an initial batch-mode query system and transform it according to the grouping solution achieved. More concretely, we divide the distances in each A_i by the maximum distance value found and add an arbitrary constant $\beta > 1$ to the songs that are not detected as belonging to the cover set where s_i is included.

We plot the relative MAP increment versus the cardinality C of the cover sets for artificial data in Fig. 3. We show that one can get a MAP accuracy improvement of up to 25%. A comparable improvement can also be achieved in the case $C = \nu$. In general, it can be seen that the higher the cardinality, the higher the improvement we can get by detecting cover sets. Improvements for real-world data are much more modest as we do not have many sets with high C. With setup 2.4, average improvements are 2.8% for UPGMA, 2.1% for WPGMA, 5.1% for proposed method 1, and 4.9% for modularity optimization.

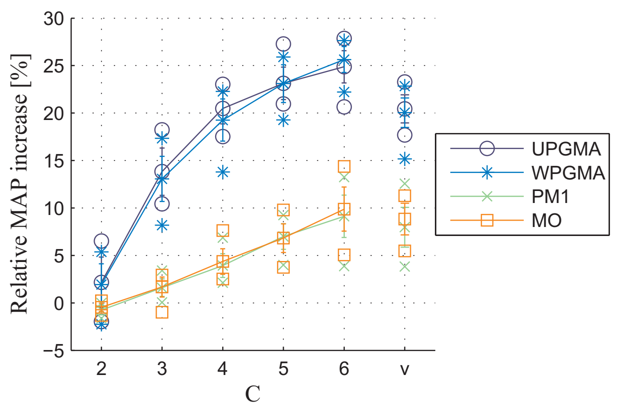


Figure 3. Relative MAP increment with artificial data. We use $\sigma_\xi = 0.25$ and, except C, the same parameters as in setups 2.3 and 2.4.

A further out-of-sample test was done within the MIREX 2008 audio cover song identification contest, where we submitted two versions of the same system [9] and obtained the two highest accuracies [14] achieved to date. The first version corresponded to the algorithm we use here for obtaining the real-world data [13]. The second version comprised the same algorithm, plus an additional post-processing step performing cover set detection based on proposed method 1 (the only method we had available at that time) and the maximum distance value normalization described in the present section. The MAP achieved with the former was 0.66 while the MAP of the latter was 0.75, which corresponds to a 13.6% relative increment. This increment is higher than the one achieved here with real-world data most probably because in the MIREX task $C = 10$.

5.2 Original detection

In the clustering context, many applications exploit compact cluster descriptions such as centroids or medoids [4, 5]. Analogously to cluster centroids and medoids, the cover set centroids and medoids can be determined. This way, the centroid of a cover set would correspond to the "average realization" and the medoid would correspond to the "prototype" [15] of the underlying musical piece. We here focus on the latter and leave the former for future consideration. In general, we could consider this prototype to be the most referential, influential, or inspirational song in a cover set (e.g. the musical piece covered by the majority of the other pieces). We here make an oversimplification and assume that the medoid of a cover song set corresponds to the original version.

To evaluate this option for detecting original songs we manually check for original versions in setup 3 of the used real-world data (we find 426 originals out of 525 cover sets) and we consider an ideal cover set detection algorithm (with all cover sets correctly estimated) as well as the best performing algorithms found in previous sections. For these last ones, we discard for evaluation the detected sets that do not contain an original.

To automatically determine the original song we take all possible pairwise distances within songs in a detected set and select the one which has a minimal distance sum to the other songs in the set. Let $\hat{S} = \{\hat{s}_j\}$, $j = 1, ..., \hat{C}$, be a set of detected covers with cardinality $\hat{C}$. Then, the index i of the prototype song corresponds to [16]

$$i = \operatorname*{argmin}_{j} \left\{ \sum_{\substack{k=1 \\ k \neq j}}^{\hat{C}} d(\hat{s}_j, \hat{s}_k) \right\}. \tag{2}$$

The results in Table 4 show the percentage of hits and misses, which can be compared to the null hypothesis of randomly selecting one song in the set. We observe that,

[14] `http://www.music-ir.org/mirex/2008/index.php/Audio_Cover_Song_Identification_Results`

[15] Standard, typical, or best example.

[16] Notice that analogous formulae can be derived by employing the notion of betweenness or closeness centrality in a complex network [10, 11].

Algorithm	C				
	2	3	4	5	6
Ideal	50.0	51.8**	39.2*	36.8*	41.7**
UPGMA	50.0	58.0**	55.0**	60.4**	50.4**
WPGMA	50.0	55.9**	46.6*	63.2**	47.5**
PM1	50.0	62.7**	61.3**	70.0**	50.0**
MO	50.0	62.8**	61.0**	70.0**	50.0**
Null	50.0	33.3	25.0	20.0	16.7
Ref. N_C	187	85	51	38	24

Table 4. Accuracies (%) for the original song detection task depending on C. The * and ** symbols denotes statistical significance at $p < 0.05$ and $p < 0.01$, respectively. The last line shows a referential N_C corresponding to the number of cover sets obtained with an ideal grouping solution.

in general, accuracies are around 50% with all considered cardinalities. This exactly corresponds to the null hypothesis of sets with $C = 2$ but, as soon as $C > 2$, accuracies become higher than the null hypothesis and statistical significance arises (statistical significance is assessed with the binomial test [16]). Discarding cover sets with no original song explains why accuracies for the algorithms studied here become slightly higher than the ones achieved by the ideal grouping algorithm. We hypothesize that our algorithms tend to split the ideal cover sets into "more coherent" or compact subsets and, therefore, within these, the method of Eq. 2 can better determine the original song.

6. CONCLUSIONS

In this paper we propose and study a framework for cover song identification based on the notion of cover sets that subsumes the current batch-mode query framework (the latter is naturally incorporated as an important part of the former). Through comprehensive experiments we show that the detection of cover sets is feasible, that it does not require extensive parameter tuning, and that it is quite robust to noisy distance measurements. Furthermore, we propose three versions of an unsupervised community detection algorithm that, when compared to existing state-of-the-art methods, achieve comparable accuracies with similar computation time (proposed method 3) or even faster (proposed method 1). We evidence that this new framework can provide new outcomes and can give raise to new applications. In addition to showing that cover set detection can substantially increase the accuracy (and coherence) of current systems, we here provide a proof-of-concept application to detect the original song within a cover set, which is inspired by the notion of cluster medoids.

Finally, we would like to highlight that, in spite of focusing on cover songs, the intuitive reasonings followed thoughout the paper could be as well applied to any IR batch-mode query system, and especially to other MIR systems (e.g. query-by-humming, query-by-example, audio fingerprinting, or music similarity).

7. ACKNOWLEDGMENTS

The authors would like to thank E. Gómez, M. Haro, P. Herrera, R. Marxer, and J. Salamon for useful discussions. This research has been partially funded by the EU-IP project PHAROS [17] IST-2006-045035.

8. REFERENCES

[1] J. Serrà, E. Gómez, and P. Herrera. *Audio cover song identification and similarity: background, approaches, evaluation, and beyond*. Springer, 2009. In press.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Books, 1999.

[3] J. S. Downie, M. Bay, A. F. Ehmann, and M. C. Jones. Audio cover song identification: Mirex 2006-2007 results and analyses. *Int. Symp. on Music Information Retrieval (ISMIR)*, pages 468–473, September 2008.

[4] R. Xu and D. C. Wunsch. *Clustering*. IEEE Press, 2009.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[6] S. Fortunato and C. Castellano. *Community structure in graphs*. Springer, Berlin, 2009.

[7] L. Danon, A. Díaz-Aguilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics*, 9008:09008, 2005.

[8] A. Egorov and G. Linetsky. Cover song identification with if-f0 pitch class profiles. *MIREX extended abstract*, September 2008.

[9] J. Serrà, E. Gómez, and P. Herrera. Improving binary similarity and local alignment for cover song detection. *MIREX extended abstract*, September 2008.

[10] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang. Complex networks: structure and dynamics. *Physics Reports*, 424:4–5, 2006.

[11] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proc. of the National Academy of Sciences*, 101:3747, 2004.

[12] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, 10:10008, 2008.

[13] J. Serrà, X. Serra, and R. G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*. Under review.

[14] C. D. Manning, R. Prabhakar, and H. Schutze. *An introduction to Information Retrieval*. Cambridge University Press, 2008. Available online: `http://www.informationretrieval.org`.

[15] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. *ACM Int. Conf. on Information and Knowledge Management*, pages 357–366, 2006.

[16] P. H. Kvam and B. Vidakovic. *Nonparametric statistics with applications to science and engineering*. John Wiley and Sons, Hoboken, New Jersey, USA, 2007.

[17] `http://www.pharos-audiovisual-search.eu`

USING MUSICAL STRUCTURE TO ENHANCE AUTOMATIC CHORD TRANSCRIPTION

Matthias Mauch, Katy Noland, Simon Dixon
Queen Mary University of London, Centre for Digital Music
{matthias.mauch, katy.noland, simon.dixon}@elec.qmul.ac.uk

ABSTRACT

Chord extraction from audio is a well-established music computing task, and many valid approaches have been presented in recent years that use different chord templates, smoothing techniques and musical context models. The present work shows that additional exploitation of the repetitive structure of songs can enhance chord extraction, by combining chroma information from multiple occurrences of the same segment type. To justify this claim we modify an existing chord labelling method, providing it with manual or automatic segment labels, and compare chord extraction results on a collection of 125 songs to baseline methods without segmentation information. Our method results in consistent and more readily readable chord labels and provides a statistically significant boost in label accuracy.

1. INTRODUCTION

The automatic extraction of chords from audio has applications in music retrieval, cognitive musicology, and automatic generation of lead sheets. In this work we present a technique that allows us to generate more authentic lead sheets than previously possible with automatic methods, by making use of musical structure. Much of musical structure is defined by repetition, a core principle in music [1, p. 229].

In popular songs a repeated *verse-chorus* format is common, in which the chord sequence is the same in all sections of the same type. In lead sheets, for better readability these sections would normally only be notated once, with repeats indicated. Our method mirrors this improvement by assigning the same chord progression to repeated sections. In addition, having found repeating sections, we have available several instances of a given chord sequence from which to estimate the chords, so we expect an improvement in estimation accuracy. We demonstrate the improvements in readability and accuracy using manually-annotated descriptions of the musical structure, and show that the improvement can also be achieved using an automatic structure annotation algorithm tailored to the task.

In Section 2 we describe related work. In Section 3 we describe the chord extraction method used and present a new segmentation technique that is tailored to our task of finding repeated chord sequences. We give examples of chord estimation with and without the segmentation technique in Section 4, and present quantitative chord estimation results in Section 5. In Section 6 we discuss our findings, and present our conclusions in Section 7.

2. RELATED WORK

The majority of approaches to automatic chord estimation rely on framewise chroma features [2] as a representation of the relative energy in each pitch class for a given time window, then apply some further processing to estimate the chords. When template-matching is used to identify chords, additional smoothing over time, for example by a median filter [3], is necessary due to musical variation and noise. Inference in hidden Markov models (HMMs) [4] simultaneously performs template-matching and smoothing. These methods treat chords as isolated features of the music, which is a considerable simplification. In reality, chords are heard in context, together with the melody, key, rhythm, form, instrumentation, and other attributes. Some chord estimation methods account for additional musical attributes during the estimation process such as key [5], or key and rhythm together [6, 7], which is a step towards a unified music analysis model.

In this work we extend the concept of unified music analysis by using repetition in the structure to enhance chord estimation. Dannenberg [8] shows that knowledge of the musical structure can greatly improve beat tracking performance, but to our knowledge the principle has not yet been applied to chord estimation.

Previous automatic music structure extraction techniques include those that primarily search for section boundaries, indicated by a sudden change in the feature of interest, which could be timbre [9], spectral evolution [10], or combinations of features [11]. A common approach is to cluster together frames that are similar, then label contiguous similar frames as a segment. However, this relies on a particular feature remaining approximately constant for the duration of a section. We are interested in chords, which do change during a section, so an approach that searches for repeated progressions [12, 13] is more appropriate for our purposes. Methods using this paradigm rely on a self-similarity matrix [14], which is a symmetric, square matrix that contains a measure of the similarity between every pair of frames. Repeated sections appear as parallel diagonal lines, and can be extracted with some post-processing, such as application of a low pass filter to reduce noise [15] followed by a thresholding operation to find contiguous frames with high similarity. In Section 3.3 we present a new variation which is similar to algorithms proposed by Ong [16] and Rhodes and Casey [17] and extracts repeated chord progressions of equal length.

3. METHOD

In a song, we call a chord sequence that describes a section such as the verse or chorus a *segment type*. Any segment type may occur one or more times in a song and we call each occurrence a *segment instance*. To make use of segment repetition as part of the chord estimation process, we rely on segment types whose instances are not only harmonically very similar, but also have the same length in beats (see Section 3.4). This is not required of a general purpose segmentation task, and hence generic segmentations are not directly utilisable. In Section 3.2 we describe how we preprocess manual segmentations to meet our needs. For automatic segmentation we choose to implement our own algorithm, which fulfills the above requirements by design (Section 3.3). First, we describe the method for calculating our basic features, beat-synchronous chromagrams (Section 3.1).

3.1 Beat-Synchronous Chromagrams

The automatic segmentation and chord estimation algorithms both rely on chroma features that are synchronised to the musical beat. The features represent the importance of each pitch class at the current beat. The initial, short chroma frames are generated from a note salience representation similar to a constant-Q transform, at a hopsize of 512 samples (46 ms) from audio that has been downsampled to 11025 Hz. For the chord extraction algorithm we split the salience representation to obtain separate bass and treble chromagrams, but the chromagram used by the segmentation algorithm covers both the bass and the treble range. For details see [18].

In order to produce beat-synchronous chromagrams we obtain a single chroma vector for each beat by taking the median (in the time direction) over all the chroma frames falling between two consecutive beat times. We use one of two sorts of beat times: manual or automatic. The collection of manual beat annotations covers 125 songs performed by the rock group The Beatles. The automatic beat times were extracted using Davies's automatic beat-tracker [19] on the same set of songs.

3.2 Manual Structural Segmentation

The manual structural segmentations cover the same 125 songs by The Beatles as we have beat annotations for: 29 songs were annotated for a previous project [1], and 96 were newly annotated for the present work. The basis for all annotations are Pollack's song analyses [20].

Every song contains several segment types, some of which have multiple instances. In some songs, the instances of a segment type differ in length. In that case, to fulfill the requirement of equal length instances, the segment type is divided to create one or more new segment types whose instances all have the same length. This may result in new segment types having only one instance in the song.

3.3 Automatic Segmentation Algorithm

The automatic segmentation method has two main steps: finding approximately repeated chroma sequences in a song, and a greedy algorithm to decide which of the sequences are indeed segments. We calculate the Pearson correlation coefficients between every pair of chroma vectors, which together represent a beat-wise self-similarity matrix $R = (r_{ij})$ of the whole song. This is similar to the matrix of cosine distances used by Ong [16]. In the similarity matrix, parallel diagonal lines indicate repeated sections of a song. In order to eliminate short term noise or deviations we run a median filter of length 5 (typically just more than one bar) diagonally over the similarity matrix. This step ensures that *locally* some deviation is tolerated.

We perform a search of repetitions over all diagonals in the matrix over a range of lengths. We assume a minimum length of $m_1 = 12$ beats and a maximum length of $m_M = 128$ beats for a segment, leading to a very large search space. We minimise the number of elements we have to compare by considering as section beginnings only those beats that have a correlation r greater than a threshold t_r, and assuming that section durations are quantised to multiples of four beats. We found that a value of $t_r = 0.65$ worked well. In future work we would like to learn t_r from data. We further reduce the search space by allowing segments to start only at likely bar beginnings. Likely bar beginnings are beats where the convolution of a function representing the likelihood of a change in harmony, and a kernel with spikes every two beats has a local maximum (details in [18]).

To assess the similarity of a segment of length l starting at beat i to another one of the same length starting at j we consider the diagonal elements

$$D_{i,j,l} = (r_{i,j}, r_{i+1,j+1}, \dots, r_{i+l,j+l}) \tag{1}$$

of the matrix $\mathcal{R}$. If the segments starting at i and j are exactly the same, then D_{ij} will be a vector of ones, and hence we can characterise a perfect match by

$$\min\{D_{i,j,l}\} = 1. \tag{2}$$

To accomodate variation arising in a practical situation, we relax the requirement (2) by using the empirical p-

[1] Segmentations available at `http://www.elec.qmul.ac.uk/digitalmusic/downloads/index.html#segment`.

quantile function [2] instead of the minimum (which is the 0-quantile), and choosing a segment threshold t_s lower than unity. The triple (i, j, l) hence describes a repetition, if

$$\text{quantile}_p\{D_{i,j,l}\} > t_s. \tag{3}$$

The two parameters $p = 0.1$ and $t_s = 0.6$ are chosen empirically. In future work we would like to learn these values from the ground truth data. The set of repetitions $\mathcal{R}_{il} = \{j : \text{quantile}_p\{D_{i,j,l}\} > t_s\}$ is then added to a list $\mathcal{L}$ of repetition sets, if it has more than one element j, i.e. if it actually describes at least one repetition. If two segments (i, j_1, l) and (i, j_2, l) overlap, only the index of the one with the higher score is retained in $\mathcal{R}_{il}$.

Each of the sets $\mathcal{R}_{il}$ represent a potential segment type, and its elements represent the start beats of instances of that segment type. However, there are typically many more repetition sets than there are segment types. To find repetition sets relating to actual segment types we use the heuristic of a music editor who tries to save paper: he will first take the repetition set in which $l \times |\mathcal{R}_{il}|$ is maximal, and then repeat this kind of choice on the remaining segments of the song, resulting in a greedy algorithm. The only exception to that rule is the case in which he finds that a sub-segment of a repetition is repeated more often than the whole segment. He then chooses the $\mathcal{R}_{il}$ pertaining to the sub-segment.

3.4 Using Repetition Cues in Chord Extraction

We use structural segmentation to combine several instances of a segment type in a song and then infer a single chord sequence from the combination.

The baseline is an existing chord labelling method [6], which extracts chords from beat-synchronous treble and bass chromagrams. Using a dynamic Bayesian network [21] similar to a hierarchical hidden Markov model the network jointly models metric position, chords and bass pitch class and infers the most probable sequence from the beat-synchronous chromagrams of the whole song. The method models four different chord classes: major, minor, diminished and dominant [3].

In order to integrate the knowledge of repeating segments, we split the chromagram for the whole song into smaller chromagram chunks, each belonging to one segment instance. If a segment type has more than one instance, all its chromagram chunks are averaged by taking the mean of the respective elements, thus creating a new chromagram chunk representing all instances of the segment type. The chord extraction is then performed on the newly generated chromagram chunk, and the estimated chords are transcribed as if they had been extracted at the individual segment instances.

4. EXAMPLES

In this section we present some example chord transcriptions with and without the segmentation technique, for the fully automatic method. Figure 1 shows a complete song segmentation, and indicates regions where the chord extraction was correct with and without the segmentation technique. Figures 2 and 3 show some excerpts on a larger scale, with the chord estimation detail visible. It is clear that the segmentation technique has had a defragmentation effect on the chord labels. A change in realisation of a repeated chord sequence between segment instances, such as a difference in melody, has in numerous places caused the standard transcription to incorrectly change chord, but when repeated segments are averaged these inconsistencies are removed. Examples include the `E:min` chord in the third row of Figure 2 and the fragmented `F♯` chords in the third row of Figure 3. This not only improves the chord accuracy (see Section 5), but also results in more natural transcriptions that include repeated chord progressions, so could be used to generate compact lead-sheets with each segment written exactly once. The figures demonstrate how the segmentation technique generates chord progressions that are indeed identical for all instances of a given segment type.

For a few songs the segmentation caused the chord estimation accuracy to decrease. Figure 4 shows an excerpt from *A Taste of Honey*, a song with one of the greatest reductions in chord accuracy due to segmentation. The transcription in the second row is good in general, but the long F sharp minor chord has been incorrectly labelled as major, an error that repeats three times in the song. The final chord in the song is F sharp major, and the segmentation algorithm has incorrectly marked this chord as a repetition of the minor chords earlier on. The problem is compounded by the behaviour of the automatic beat tracker at the end of the song: when the true beats stop, the beat tracker continues at a much faster tempo, which has caused the last chord to appear to have the same length in beats as the much longer (in seconds) F sharp minor chords throughout the song. This poor case, then, still produces a good transcription but with a parallel major-minor error caused in part by the beat tracker giving too much importance to the final chord.

5. QUANTITATIVE RESULTS

While the previous section has demonstrated how segmentation can help create consistent and more readily readable chord transcriptions, this section examines their overall performance. To that end we compare the six different combinations arising from two different beat annotations (manual and automatic) and three different segmentation annotations (manual, automatic, and none).

For each of the ground truth chords, we make a musical judgement regarding whether it should fall into one of the chord classes we investigate: major, minor, diminished, dominant or no chord. If there is no clear suitable mapping, for example for an augmented chord, our chord estimation will always be treated as incorrect. We use as an evaluation measure the relative correct overlap per song in physical time against a reference of Harte's chord tran-

[2] `http://www.mathworks.com/access/helpdesk/help/toolbox/stats/quantile.html`

[3] strictly speaking: major with a minor seventh

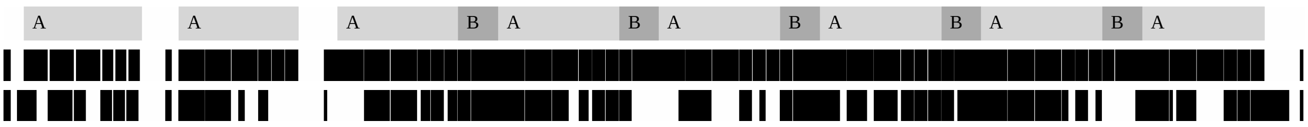


Figure 1. *Dizzy Miss Lizzy* (complete). First row: automatic segmentation. Second row: regions of correctly-labelled chords using segmentation. Third row: regions of correctly-labelled chords without using segmentation.

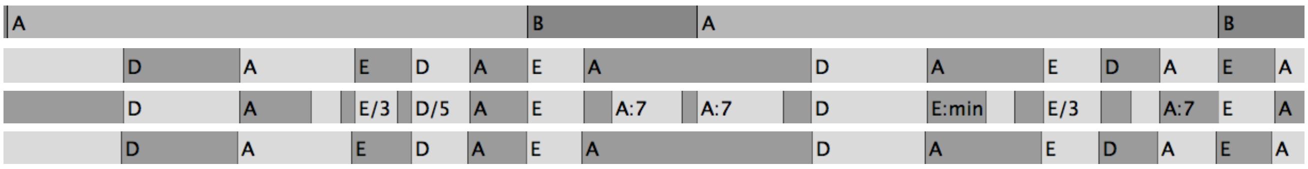


Figure 2. Extract from *Dizzy Miss Lizzy*. First row: automatic segmentation. Second row: automatic chord labels using segmentation. Third row: automatic chord labels without using segmentation. Fourth row: hand-annotated chord labels.

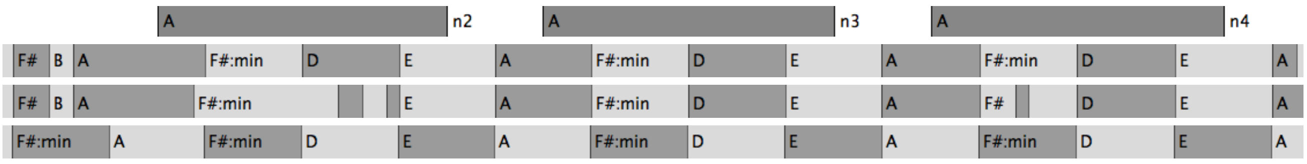


Figure 3. Extract from *Please Mister Postman*. First row: automatic segmentation. Second row: automatic chord labels using segmentation. Third row: automatic chord labels without using segmentation. Fourth row: hand-annotated chord labels.

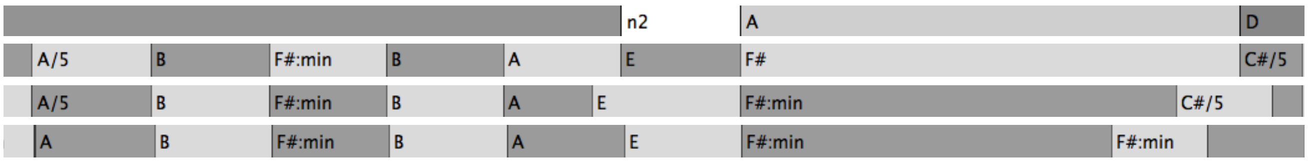


Figure 4. Extract from *A Taste of Honey*. First row: automatic segmentation. Second row: automatic chord labels using segmentation. Third row: automatic chord labels without using segmentation. Fourth row: hand-annotated chord labels.

scriptions [22], i.e.

$$\mathcal{O} = \frac{\text{summed duration of correct chords}}{\text{duration of song}}. \quad (4)$$

A chord is considered correct if its chord type matches that of the ground truth chord and its root note matches that of the ground truth or its enharmonic equivalent. In Table 1 we report mean overlap scores over the 125 songs. For completeness we also report the equivalent scores using the chord classes used in the MIREX chord detection task [23], in which only two chord classes are distinguished. We recommend that these numbers are used only to assess the approximate performance of the algorithm because—as can be seen in Figure 5—the distribution is multimodal with a wide spread, due to the large range of difficulty between songs. An evaluation method that takes into account these "row effects" is the Friedman analysis of variance [24] based on ranking the results per song. The associated p-value is below double precision, suggesting that at least one method is significantly different from the others. The multiple comparison analysis [4] in Figure 6 shows that the improvements due to segmentation cues for both manual segmentation and automatic segmentation are significant. Figure 7 illuminates why this is so: the use of segmentation information leads to an improved relative overlap score in most of the songs, for example, automatic segmentation improves accuracy on 74% of songs.

[4] http://www.mathworks.com/access/helpdesk/help/toolbox/stats/multcompare.html

Table 1 shows that the choice of segmentation method makes very little difference to our results, with a much greater difference caused by the beat annotation method. Since the automatic beat tracker was adjusted for quick tempos, several songs were tracked at double tempo with respect to the manual annotations, so our results suggest that the chord estimation method works better with higher beat granularity.

6. DISCUSSION

The method presented here is not tied to the individual algorithms. Using other chord extraction or segmentation methods could further improve results and shed more light on the performance of its constituent parts. As mentioned in Section 3.3 we plan to investigate the effects of training some of the segmentation parameters. It would also be in-

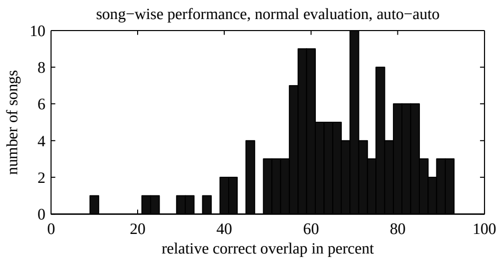


Figure 5. Relative correct overlap for the configuration using automatic beats and automatic segmentation: Histogram showing song frequencies. The clearly non-Gaussian distribution suggests that the mean correct overlap should not be the main evaluation technique.

configuration		four classes	MIREX
man. beat	man. segm.	64.4	71.8
	auto segm.	64.1	71.5
	no segm.	61.7	69.1
auto beat	man. segm.	66.4	73.7
	auto segm.	65.9	73.0
	no segm.	63.4	70.7

Table 1. Mean relative overlap in percent and mean rank results. The four classes measure is our preferred measure for this task. The MIREX measure gets higher scores, since it maps all chords to two classes, in particular dominant and major chords are taken to be equivalent.

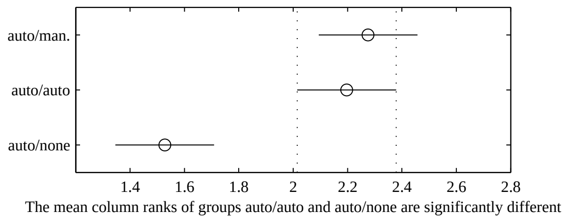


Figure 6. Multiple comparison test of the three best-performing variants (automatic beat extraction) at a confidence level of 99%, based on Friedman analysis of variance. The upper two rows show that of the two methods using manual (auto/man.) and automatic (auto/auto) segmentation significantly outperform the one without (auto/none), while the difference between automatic and manual segmentation is not significant.

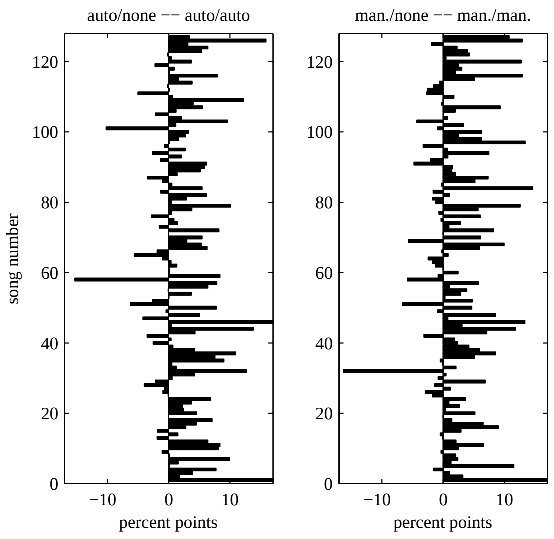


Figure 7. Song-wise improvement in correct relative overlap for the methods using segmentation cues: using automatic beats, automatic segmentation improves performance on 74% of songs (left); for manual beats, manual segmentation improves 68% of songs (right).

teresting to determine whether using the median (instead of the mean) to average chromagram chunks would lead to improvements for cases like *A Taste of Honey*, where one major chord has tipped the mean to the parallel major.

The present work focussed on early rock music. We expect that—given a good segmentation—improvements in recognition results could be even greater for jazz: while the extraction of chords in jazz is more difficult than in rock music due to improvisation and more complex chord types, the repetition of segment types is often more rigid.

The method to share information globally between segments we used for this work is a simple one. Integrating this process with the chord extraction itself is a more elegant solution, but would require structure learning.

7. CONCLUSIONS

We have shown that using knowledge of repeating structure in a song can improve chord recognition in two ways. Firstly, by design the chord estimates are more consistent between instances of the same segment type, which leads to a more natural transcription that could be used to generate realistic lead sheets with structure markings. Secondly, we have shown that our method of averaging the different instances of each segment type has significantly improved the measured chord accuracy. This is demonstrated by examples that show how non-repeating incorrect chord fragments are removed by the averaging process. The improvement is observed both when using manually-annotated beat times and segments, which shows that the principle is valid, and when using a fully-automatic method, which shows that the principle can be applied to real systems, and is effective even when there are some errors in the beat or segment labels.

The results we have presented support the wider hypothesis that unified music analysis improves estimation of individual features [6–8]. We would like to extend this approach in our future work to allow chord estimation to be informed by a complete musical context, including melody, tonality, timbre and metrical structure.

8. REFERENCES

[1] David Huron. *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press, 2006.

[2] Takuya Fujishima. Real time chord recognition of musical sound: a system using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 464–467, 1999.

[3] Christopher Harte and Mark Sandler. Automatic chord identifcation using a quantised chromagram. In *Proceedings of 118th Convention*. Audio Engineering Society, 2005.

[4] Juan P. Bello and Jeremy Pickens. A Robust Mid-level Representation for Harmonic Content in Music Signals. In *Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005, London, UK*, pages 304–311, 2005.

[5] Kyogu Lee and Malcolm Slaney. Acoustic Chord Transcription and Key Extraction From Audio Using Key-Dependent HMMs Trained on Synthesized Audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):291–301, February 2008.

[6] Matthias Mauch and Simon Dixon. Simultaneous estimation of chords and musical context from audio. To be published in IEEE Transactions on Audio, Speech, and Language Processing.

[7] Hélène Papadopoulos and Geoffroy Peeters. Simultaneous estimation of chord progression and downbeats from an audio file. In *Proceedings of the 2008 ICASSP Conference*, pages 121–124, 2008.

[8] Roger B. Dannenberg. Toward automated holistic beat tracking, music analysis, and understanding. In *Proceedings of the 6th International Conference on Music Information Retrieval*, London, 2005.

[9] S. Abdallah, K. Noland, M. Sandler, M. Casey, and C. Rhodes. Theory and evaluation of a Bayesian music structure extractor. In *Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005, London, UK*, pages 420–425, 2005.

[10] G. Peeters, A. La Burthe, and Xavier Rodet. Toward automatic music audio summary generation from signal analysis. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, Paris, 2002.

[11] Namunu C. Maddage. Automatic structure detection for popular music. *IEEE Multimedia*, 13(1):65–77, 2006.

[12] Meinard Müller and Frank Kurth. Towards structural analysis of audio recordings in the presence of musical variations. *EURASIP Journal on Advances in Signal Processing*, 2007.

[13] Masataka Goto. A chorus-section detecting method for musical audio signals. In *Proceedings of the 2003 IEEE Conference on Acoustics, Speech and Signal Processing*, pages 437–440, 2003.

[14] Jonathan Foote. Visualizing music and audio using self-similarity. In *Proceedings of the 7th ACM International Conference on Multimedia (Part 1)*, pages 77–80, 1999.

[15] Mark A. Bartsch and Gregory H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(4), February 2005.

[16] Bee Suan Ong. *Structural Analysis and Segmentation of Music Signals*. PhD thesis, Universitat Pompeu Fabra, 2006.

[17] Christophe Rhodes and Michael Casey. Algorithms for determining and labelling approximate hierarchical self-similarity. In *Proceedings of the 2007 ISMIR Conference, Vienna, Austria*, pages 41–46, 2007.

[18] Matthias Mauch. A chroma extraction method and a harmonic change detection function. Technical report, Queen Mary, University of London. Available at `http://www.elec.qmul.ac.uk/digitalmusic/papers/2009/Mauch09-C4DM-TR-09-05.pdf`.

[19] Matthew Davies. *Towards Automatic Rhythmic Accompaniment*. PhD thesis, Queen Mary University of London, London, UK, August 2007.

[20] Alan W. Pollack. Notes on... series, 1995. Available at `http://www.recmusicbeatles.com`.

[21] Kevin P Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[22] Christopher Harte, Mark Sandler, Samer A. Abdallah, and Emilia Gomez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005, London, UK*, pages 66–71, 2005.

[23] MIREX audio chord detection subtask, music information retrieval evaluation exchange, 2008. `http://www.music-ir.org/mirex/2008/index.php/Audio_Chord_Detection`.

[24] David Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338. ACM New York, USA, 1993.

VISUALISING MUSICAL STRUCTURE THROUGH PERFORMANCE GESTURE

J.MacRitchie, B.Buck and N.J.Bailey
Centre for Music Technology, University of Glasgow
{j.macritchie, b.buck, n.j.bailey}@elec.gla.ac.uk

ABSTRACT

A musical performance is seen as the performer's interpretation of a musical score, illuminating the interaction between the musical structure and implied emotive character [1]. It has been demonstrated that performers' physical gestures correlate with structural and emotional aspects of the piece they are performing and that this information can be decoded by an audience when presented with a visual-only performance [2].

This paper investigates the relationship between direction of physical movement and underlying musical structures. The Vicon motion capture system is used to record 3D movements made by nine university-level pianists performing Chopin preludes op.28 Nos 6 and 7. The examination of several pianists provides insight into the similarity and differences in gestures between performers and how these relate to structure.

Principal Component Analysis (PCA) of these performances and consequent analysis of variance reveals a relationship between extrema of the first six significant components and timing of phrasing structure in Prelude 7 where motion troughs consistently lag behind the occurence of phrase boundaries in the audio. This relationship is then examined for Prelude 6 which encompasses longer, expanded phrases and changes in rhythm. These expanded phrases are associated with elongated or split gestures, and variations of the motif with changes in movement.

1. INTRODUCTION

Structural communication in performance is well understood for parameters such as timing and dynamics, and certain relationships between these parameters and structures have been clarified [3]. We now know that performers tend to slow the tempo towards the end of a phrase and use dynamics to 'shape' a phrase often using a diminuendo towards the end. These are of course context-dependent as a performer can use the same parameters to mark different structural features [4]. Performers also have personal styles of playing and will not all use the same performance parameter to emphasise the same structural feature.

This personal style becomes a much bigger issue when treading into the field of physical gestures. No such straightforward relationships exist between physical gestures and musical structure, although it has been demonstrated that performers' gestures do contain information about the music being performed [5, 6]. Perception studies have also shown that typical audience members can accurately perceive information about tension and phrasing from visual-only performances [2].

This paper aims to establish relationships between body movement and the phrasing structure it attempts to convey, exploring how these change between different pieces of music. This investigation of phrasing and gesture relationships will be conducted by recording piano performances through the Vicon motion capture system in synchrony with audio recordings, and subsequent analysis of the movement results alongside traditional analyses of structure in the chosen pieces.

2. CHOPIN PRELUDES

Several factors fuelled the choice of two Chopin preludes for gestural analysis.

- In order to make general statements about phrasing structure, it was necessary to provide some amount of scientific control, otherwise the exercise could become tantamount to guesswork. Two pieces were therefore chosen to examine the progression of movement and structure relationships.
- The pieces' genre may have an effect on performance gestures, so ideally those from the romantic period would provide the most expressive performances.
- The motion capture system coped better with shorter recordings and so brief pieces were preferred.

The preludes chosen for this investigation were Preludes op.28 No.s 6 & 7. Prelude No. 7 in A major has a strict, rigid structure, with the rhythmically identical two bar phrase occurring eight times in total. As can be seen from Figure 1, this binary form 16-bar piece has the main boundary occurring exactly halfway through at bar 8, and a harmonic arrival point occurring with the chord at the end of bar twelve [7]. The two sections of the piece are thought to each contain a set of antecedent-consequent phrases.

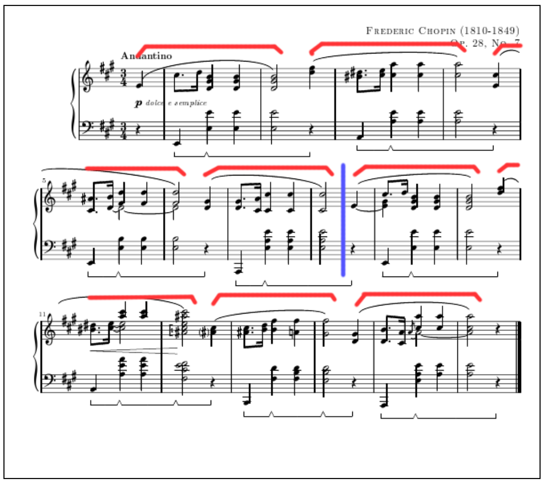


Figure 1. Chopin prelude op.28 No.7 with two-bar phrases marked in red and line boundary for end of first section.

Prelude No.6 in B minor comprises three sections [8] from bars 1-8, bars 9-22 and a coda section from bars 23-26. The first section represents an 'extended idea'. As seen in Figure 2, Chopin begins with another two-bar motif in B minor. This motif is repeated with a slightly higher pitch range in the next two bars. The first part of the motif is repeated again and then expands into a four bar phrase ending at bar 8, the first sectional boundary. The second section represents an expansion of this idea. At bar 9, The two-bar motif from the beginning is repeated with the next expansion moving into C major. A new four bar phrase is introduced at bar 15, answered by the consequent four bar phrase concluding on the tonic at bar 22 at the second sectional boundary. The piece ends with a slight coda in B minor in its final phrase [1]

As many different analyses of one piece can exist, each pianist's own interpretation of phrasing is noted within this experiment. Analysis of each pianist's performance of the rigidly structured prelude no.7 will provide an impression of performance style. This piece provides the opportunity to observe movements for each phrase in isolation before moving on to examine the other prelude containing slightly more complicated structures.

3. METHOD

3.1 Performance Motion Capture

Using the Vicon 3D-motion capture system [9], performances of the two selected Chopin preludes by nine highly trained pianists [2] were recorded. All pianists were asked

[1] This analysis of Chopin's Prelude Op.28 No.6 is combined from Kofi Agawu, V. 'Concepts of Closure and Chopin's opus 28' in Music Theory Spectrum 9:1–17, 1987. and comments made by Jennifer MacRitchie, University of Glasgow, and David Lewis and Christophe Rhodes, Goldsmiths, University of London

[2] These nine performers consisted of five music performance undergraduate students, four at the University of Glasgow and one at the University of Edinburgh, two postgraduate students from the Royal Scottish Academy of Music and Drama and two amateur pianists with more than ten years of performance experience. Each pianist was paid a one-off sum of £25 for their participation in the experiment.

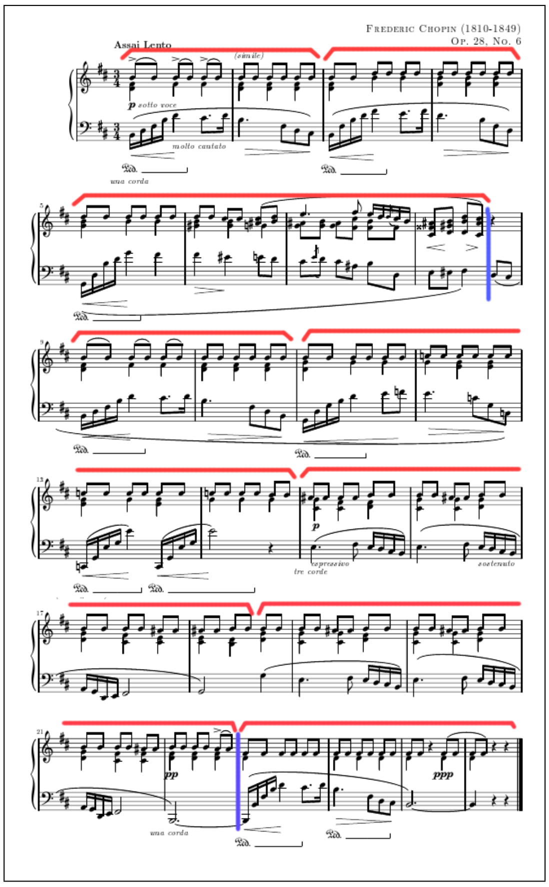


Figure 2. Chopin prelude op.28 No.6 with phrases marked in red and line boundaries for the ends of sections.

to play the pieces from memory in an effort to ensure an in-depth knowledge of both pieces. The only performance direction given to the pianists was to play as if they were in a normal concert setting. Pianists' interpretation of phrasing structure and gestural expression were taken by means of a self-report following the recording.

The Vicon motion capture system consists of twelve infra-red cameras placed around the room to ensure capture of a particular volume of space. Retro-reflective markers were placed onto a velcro jacket and hat worn by the performers in the configuration shown in the head and upper body model in Figure 3. This particular model combined the upper body model from Cutti et al. [10] with four reference markers for the head positions. Each camera tracks the coordinates of the 28 markers and triangulates their position in order to build a 3-D model of each performer. Each video was recorded at 120 frames per second in synchrony with an analogue input for the recorded sound. The models were then reconstructed by post-processing and any points where the cameras had failed to pick up a certain marker were filled with the estimation models available from the Vicon Nexus software.

Problems were encountered particularly with the markers placed on the elbows of the performers. As the markers were placed not directly onto the skin but onto a velcro jacket, there were several points in the recordings where the marker was lost by the camera as the jacket had moved

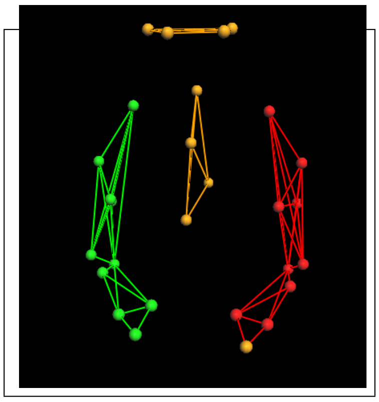


Figure 3. Vicon marker model for upper body of pianists.

round the elbow and displaced the marker. Although the Vicon interpolation algorithms filled most of these elbow gaps, the system is proprietary, so these algorithms are unavailable for inspection. The accuracy of reconstruction for these elbow gaps must therefore be considered suspect.

4. RESULTS

4.1 Motion Capture Analysis

As motion capture always produces such an overwhelming plethora of data, the traditional phrase analysis of each prelude provides us with points from which to start investigation of gestural cues at phrasing boundaries. Each performer's audio recording was annotated in Audacity [11] with the timings of the phrase boundaries explained in section 2. by a separate professional pianist. Each performer's own view of the phrase segmentation was also noted in case of any differences to traditional analysis. The pianists' self-reports also conveyed a wide view on the role of movement in performance, with some branding movements extra to sound productive ones as completely unnecessary and something they tried to limit, whilst others felt it vital to move in order to 'feel' the music they were performing. Although physical gestures in performance can be classified as movements necessary to the actual sound-production, or movements that are related to the music but not necessary for the actual sound (i.e. ancillary) [12], it is acknowledged that gestures may still be multi-functional. To view the overall general motion characteristics of each performer, principal component analysis (PCA) was performed through designated pca modules using singular value decomposition algorithms [13] on the complete set of motion data for each pianist. Each person's principal component score was mapped against the timings of each phrasing boundary to determine if there was a pattern of movement for each phrase. Reduced-dimension curves such as these are good at expressing a general overview but tend to lose some semantics of the actual movement being performed and so each individual marker is then also examined for reference to phrases, measures and beats.

Three pianists have been chosen to demonstrate the spread of results concisely. These pianists were chosen according to their ability, their standard deviation and variance of movement calculated for intra-performance data and also their views on movement during a performance. Performer 1 is a highly trained amateur pianist and had a small standard deviation of movement. Performer 2 is a conservatory trained postgraduate student and had a large standard deviation of movement, and Performer 3 is a music undergraduate student and had a mid-range standard deviation. Normalization of results allows the movements to be correlated with phrase structure independent of differences in amplitude. The arrows in each graph indicate the point in time where the last note of each phrase ends in the audio stream.

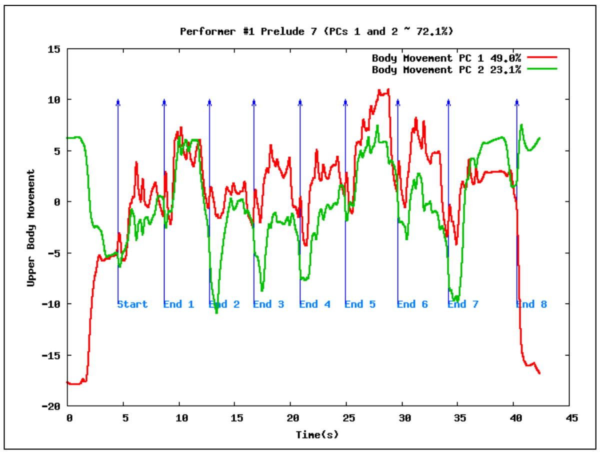


Figure 4. First two principal components of movement for prelude 7, Performer 1 mapped against phrase boundaries.

4.2 Prelude 7 in A major

Starting with prelude 7, in which the pianists self-reporting analysis agreed with the traditional phrase segmentation marked in section 2., Figures 4, 5 and 6 show the first two principal components accounting for around 70% of the overall movement. These appear to relate to the phrase boundaries as dictated by traditional music analysis. For each performer, the correlation between markers and the resultant PCA curves i.e. the loadings, clarified that instead of a few markers being prevalent in causing the most variance in motion, the PCA curves were a result of the variances in a combination of several markers and these differed slightly for each pianist.

Interestingly, Performer 1's self-report on conclusion of the recordings expressed the opinion that movement in performance did not convey any information on phrasing and that during performances, he/she attempted to minimize movements and facial expressions. However, Performer 1's movement, shown in Figure 4, shows a clear relationship between physical gestures and phrasing structure where each phrase boundary precedes a trough in the motion graph. The loadings for Performer 1 related highly to movements in the upper arms, the elbows, the wrists and the chest.

Performer 2's main component loadings consist of movement in every section of markers: the wrists, elbows, upper arms, shoulders, chest and the head. A pattern can

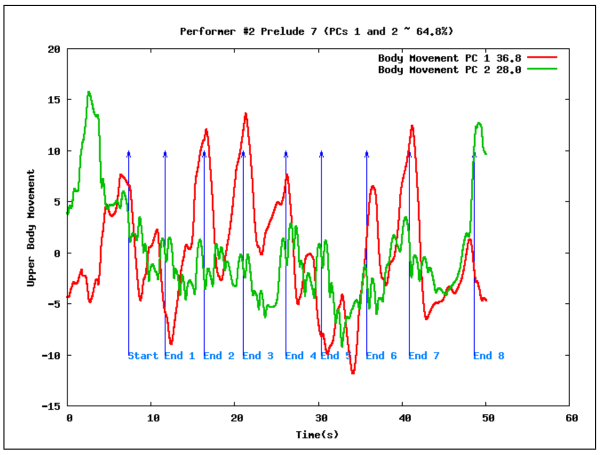


Figure 5. First two principal components of movement for prelude 7, Performer 2 mapped against phrase boundaries.

be seen in Figure 5 for the first principal component(red), where the phrase boundary occurs at a motion peak for all but phrase endings 1, 5 and 6 whose peaks precede the boundary in time. The second component (green) shows a relationship between the phrase endings and peaks in the graph.

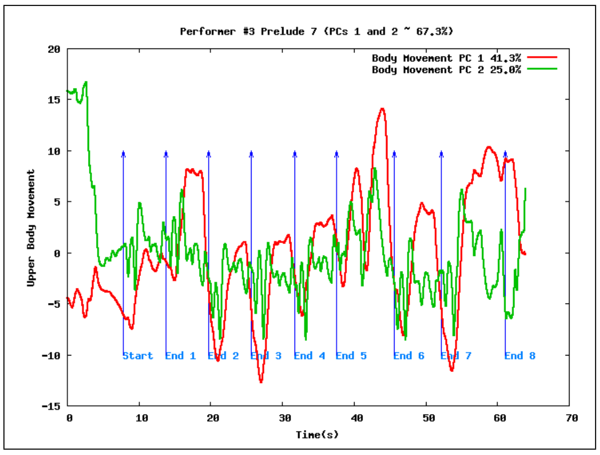


Figure 6. First two principal components of movement for prelude 7, Performer 3 mapped against phrase boundaries.

Performer 3's main component loadings relate to movements in the wrists, upper arms, chest and head. A pattern can be seen in Figure 6 where each phrase boundary precedes a trough in the main component in all phrases. The pattern of movement for each phrase in this component is changed slightly during phrase six. This could be to emphasise the harmonic arrival in bar 12 of the piece.

The addition of the weighted values of the first six principal component scores for each performer produces a visualisation of data accounting for more than 90% of the variance in motion, the weightings calculated from the percentage variance of each component over the dataset. These have been resampled with 10,000 points and time-warped to allow more direct comparison between performers. The distance between each audio phrase boundary is 0.1 and quoted means and standard deviations are calculated for the distances between the troughs of the motion trajectory and its corresponding phrase boundary. Figure 8 shows a pattern in all phrases except one, five and six (mean=0.0369,s.d=0.015). Phrases 1 and 5 are the first phrases in the each section of prelude 7 whilst phrase 6 contains the harmonic arrival point. Figures 7 and 9 show a clear relationship between the movement and phrase boundaries.(mean=0.0186,s.d=0.0116 and mean=0.0204,s.d=0.0068 respectively) The calibration of this hypothesis with prelude 7 now provides us with a useful tool to observe the structure of prelude 6 for the same performers.

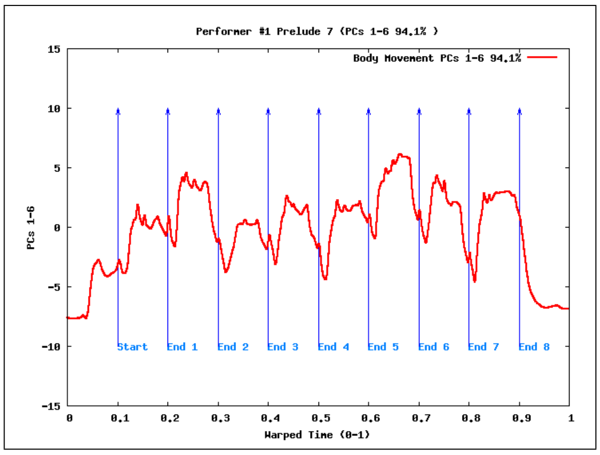


Figure 7. Combination of first six components for prelude 7, Performer 1 mapped against phrase boundaries.

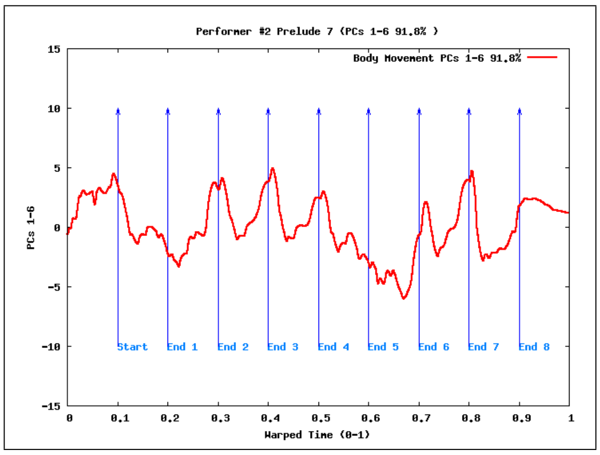


Figure 8. Combination of first six principal components of movement for prelude 7, Performer 2 mapped against phrase boundaries.

4.3 Prelude 6 in B minor

The initial two-bar motif in prelude 6 is in the left hand melody marked in the score seen in section 2. This motif is varied in the subsequent phrases, first in pitch for the second phrase, then also in rhythm for the third phrase ending at bar 8. This is echoed in the movements made by performers. This initial step looks at the first three

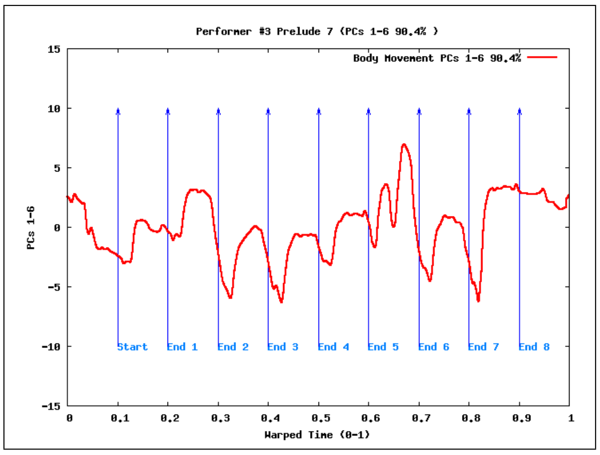


Figure 9. Combination of first six principal components of movement for prelude 7, Performer 3 mapped against phrase boundaries.

phrases of prelude 6 as agreement on phrase segmentation between analyses and performers' self-reports diverge from this point onwards. The means and standard deviations of distance between motion trough and phrase boundary are for the first three phrases only.

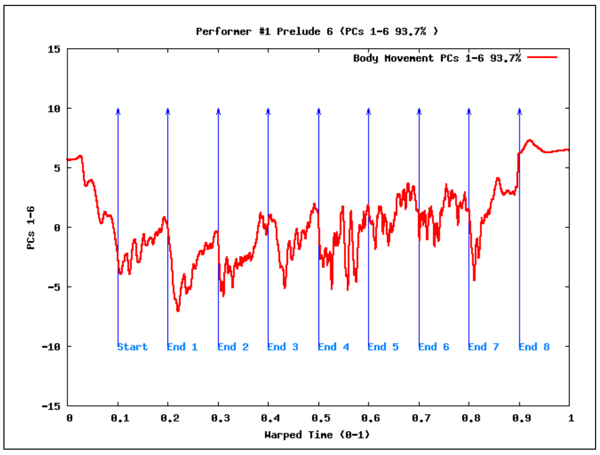


Figure 10. Combination of first six principal components of movement for prelude 6, Performer 1 mapped against phrase boundaries.

Performer 1's weighted combination of principal components of movement, shown in Figure 10, shows a distinct pattern of movement for the two-bar motif established in phrase one of the piece (mean=0.0223,s.d=0.0110). Its elongation in phrase 3 is mimicked by an elongated gestural movement. The change in motif at bar 15 beginning with a four bar phrase (phrase 5), is marked with a different pattern of movement. This movement is repeated as the consequent four bar phrase is played.

The first six principal components for Performer 2, as seen in Figure 11, also shows a clear pattern within the first three phrases (mean=0.0129,s.d=0.0115 this particular mean is negative as each trough occurs slightly before the phrase boundary). Interestingly, in phrase 3 where the original two-bar motif is expanded, we clearly see two separate movements. As the length of the phrase being performed is just under 12 seconds long, we refer to the theory of gestures being separated into gesture-units i.e. action-chunking [14]. At which points within a long phrase this action-chunking occurs is most likely related to the smaller rhythmical groupings within the particular phrase.

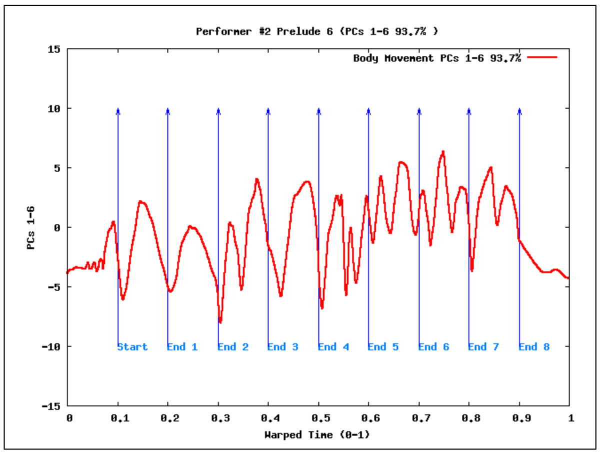


Figure 11. Combination of first six principal components of movement for prelude 6, Performer 2 mapped against phrase boundaries.

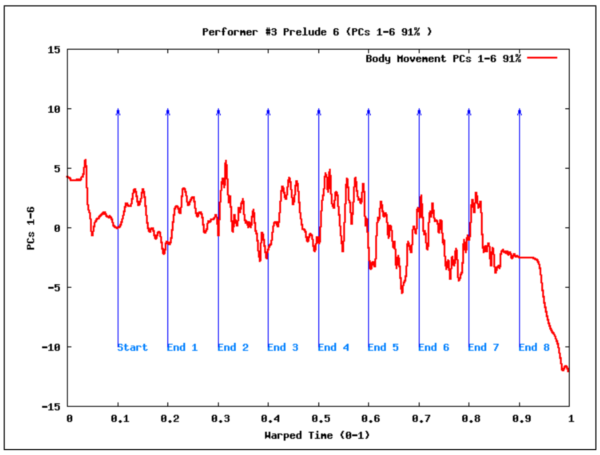


Figure 12. Combination of first six principal components of movement for prelude 6, Performer 3 mapped against phrase boundaries.

The first principal component for Performer 3 as seen in Figure 12 again shows this pattern between movement and phrasing (mean=-0.0069,s.d=0.0086). Again at phrase 3, we can see the beginning of the two-bar motif related movement before the phrase is expanded and the resulting gesture elongated as well.

4.4 Analysis Across All Performers

One-way ANOVAs were conducted to investigate the effects of performer style and phrasing motion. No significant effect of performer was found which, when coupled with variance analyses of individual performers, revealed that performers were consistent in their movement patterns throughout the piece, showing no significant difference between performers in their overall motion timing variance. A significant effect of trough location was found between performers (F=17.32, p<0.001). When considered with the minimal variance within performances, these results show that performers were consistent in the location of their coherant motion patterns (troughs) with respect to phrasing boundaries. As performers' interpretations of the latter part of prelude 6 differ, such straightforward comparisons cannot be performed for this piece.

Comparisons of pianists' singular marker movements for both preludes showed similarities in movement in the y axis (along the length of the keyboard) of every single marker of the upper body. Troughs in each plot occurred either slightly before or slightly after the audio phrase boundary. This changed between pianists but was consistent between performances of two preludes with differing structure and different melodies. Differences in pianists' marker movements showed for some pianists, a clear relationship with phrasing in all three axes of movement for each marker. Some used their heads to mark out phrasing whereas others preferred to use their upper body. These movements were performer specific and occurred across both pieces. Within some markers denoting phrasing, movements corresponding to the measures and beats within the piece were found. Markers which were not phrasing specific also appeared to highlight beats or measures of the piece. Despite this, the overall general movement reflected by the PCA shows a clear phrasing pattern.

5. CONCLUSIONS

Structural information appears to be inherent in pianists' directional movements across the three axes. Principal component analysis confirms the relationship between general movement in the upper body and head with composed structure. Variance analysis shows that each performer's general movement consistently lags behind the occurence of a phrase boundary in the audio stream. By examining pianists' movements in performances of Chopin's prelude op.28 no.7, it is confirmed that short phrases in isolation, with the same rhythmical pattern appear to invoke similar movements by the performers. Movement also appears to change when the motif is varied, as examined in performances of Chopin's prelude op.28 no.6. In comparison with the short two-bar motif, phrases with a longer duration have different elongated gestures and are sometimes split into sections in a process referred to as action-chunking.

This investigation provides the initial step of relating movements to phrases. Further investigations into effect of genre of the music being performed on the structural relationship with movement are required to make more general statements across a wider variety of music. Further steps for this research are to clarify whether physical gestures are related to other compositional or performance attributes. Empirically relating general movement to structural aspects of performed music contributes to the argument that ancillary performer movements may have a music-related function. This has implications for piano pedagogy and furthers the understanding of the wider relationship between music and movement.

6. ACKNOWLEDGEMENTS

The authors acknowledge Prof. Frank Pollick and researcher Lukasz Piwek of the Psychology Dept at the University of Glasgow for their support and access to equipment. This research is EPSRC funded.

7. REFERENCES

[1] Shaffer, L. Musical Performance as Interpretation *Psychology of Music*,23, 17–28, 1995.

[2] Vines,B. et al. Cross-modal Interactions in the Perception of Musical Performance. *Cognition*, 101, 80–113, 2006.

[3] Friberg, A. and Battel, G.U. Structural Communication in Parncutt, R. and McPherson, G.E. (Eds.) *The Science and Psychology of Music Performance*, Oxford University Press, 2002, 199–218

[4] Clarke, E.F. The Semiotics of Expression in Musical Performance *Contemporary Music Review*, 1998, 17, Part 2, 87–102

[5] Wanderley, M.M. Quantitative Analysis of Non-Obvious Performer Gestures Gesture and Sign Language in *Human-Computer Interaction: International Gesture Workshop*, 2001.

[6] Jane W. Davidson. Qualitative insights into the use of expressive body movement in solo piano performance: a case study approach. *Psychology of Music*, 35(3):381–401, 2007.

[7] Kresky, J. A Reader's Guide to the Chopin Preludes Greenwood Press, 1994.

[8] Kofi Agawu, V. Concepts of Closure and Chopin's opus 28 *Music Theory Spectrum*, 9:1–17, 1987.

[9] Motion Capture Systems from Vicon `http://www.vicon.com`.

[10] Cutti,A.G. et al. Soft tissue artefact assessment in humeral axial rotation *Gait and Posture*, 2005, 21, 341-349.

[11] Ardour - the new digital audio workstation. `http://www.ardour.org`.

[12] Cadoz, C. and Wanderley, M. Gesture-Music in Wanderley, M. and Battier, M. (ed.) *Trends in Gestural Control of Music* Iracam - Centre Pompidou, 2000

[13] Risvik, H. PCA Python module `http://folk.uio.no/henninri/pca_module`

[14] Godøy, R. I. Knowledge in Music Theory by Shapes of Musical Objects and Sound-Producing Actions. In M. Leman (ed.) *Music, Gestalt, and Computing*, Springer Verlag, Berlin (1997) pp. 106–110

FROM LOW-LEVEL TO SONG-LEVEL PERCUSSION DESCRIPTORS OF POLYPHONIC MUSIC

Martín Haro, Perfecto Herrera
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{martin.haro,perfecto.herrera}@upf.edu

ABSTRACT

We address here the automatic description of percussive events in real-world polyphonic music. By taking a pattern recognition approach we evaluate more than 2,450 object-level features. Three binary instrument-wise support vector machines (SVM) are built from a training set of more that 100 songs and 10 genres. Then, we use these binary models to build a drum transcription system achieving comparable results with state of the art algorithms. Finally, we present 17 song-level percussion descriptors computed from the imperfect output of the transcription algorithm. We evaluate the usefulness of the proposed descriptors in music information retrieval (MIR) tasks like genre classification, danceability estimation and Western vs. non-Western music discrimination. We conclude that the presented song-level percussion descriptors provide complementary information to "classic" descriptors, that can help in the previously mentioned MIR tasks.

1. INTRODUCTION

During the last decade the interest in the transcription of percussive instruments has grown and most of the work has focused on the problem of drum[1] transcription [1]. The aim of such systems is to obtain, from an audio signal, a representation of the type of percussion instrument played (instrument recognition), and when it has been played (temporal location).

The transcription of isolated or polyphonic drum sounds (i.e. without concurrent pitched sounds) can be considered a practically solved problem (e.g. see [2]). However, the automatic transcription of percussive events in polyphonic music is an open problem where there is still a lot of room for improvement.

Instead of focusing on a full transcription system, we consider that, when MIR of polyphonic music is addressed, an automatic music "description" approach should be taken. The main idea behind such an approach is to obtain "predicates" or labels that apply to a given music excerpt and usually this information goes beyond traditional music scores.

In this paper we present and evaluate several song-level percussion descriptors extracted from the output of an imperfect transcription system. The aim of these descriptors is to semantically describe general characteristics of within-song drum events such as drum-instrument degree of presence, drum-instrument relationships (i.e. inter-instrument ratios) and most-frequent inter-instrument intervals. Finally, we explore the usefulness of the proposed descriptors for some MIR tasks such as genre classification, danceability estimation and Western vs. non-Western music classification.

The paper is organized as follows: An overview on percussion transcription of polyphonic music is presented in section 2. In section 3 "full" and "relaxed" transcription systems are described. Next, song-level percussion descriptors are proposed and evaluated within several MIR tasks (section 4). Finally, section 5 presents some conclusions.

2. RELATED WORK

Most of the works on transcription of percussive events in polyphonic music have focused on transcription of drum kit sounds, specially on bass drum (BD), snare drum (SD) and hi-hat (HH) sounds. In Table 1 a summary of the most relevant works on drum transcription in polyphonic music is presented. It is worth to notice here that the presented results can not be directly compared since they were not evaluated on the same dataset. Sandvold et al. [3] used a combination of general and localized sound models. The correct classified instances obtained from the general model (C4.5 with AdaBoost) were manually parsed to a localized training set. Yoshii et al. [4] achieved the best results in the MIREX 2005[2] audio drum detection contest by using matching template spectrograms. The main idea here was to obtain a template spectrogram representation of a particular percussion instrument from a large training database of sounds. Thus, when analyzing a song, a template-adaptation algorithm was applied on every onset. A distance measure for template matching was used to try to minimize the spectral overlapping of other sounds. Dittmar's [5] system was also evaluated within

[1] The word "drum" refers to a standard Rock/Pop drum kit found in Western music.

[2] See MIREX web site: http://www.music-ir.org/evaluation/mirex-results/audio-drum/index.html for more information.

Authors	# songs	Approach	Main Algorithms	Overall	BD	SD	HH
Sandvold et al. [3]	25	Patt-Rec	local. model	0.924	0.951	0.931	–
Yoshii et al. [4]	50	Sp. Temp	Temp. match.	0.670	0.728	0.702	0.574
Dittmar [5]	50	S. Sep+Sp. Temp	NN ICA	0.588	0.606	0.581	0.585
Tanghe et al. [6]	50	Patt-Rec	SVM	0.615	0.688	0.555	0.601
Gillet and Richard [7]	20	Patt-Rec	SVM+local. model	0.840	0.824	0.842	–
Paulus and Klapuri [8]	45	Patt-Rec	HMM	0.697	0.795	0.655	0.660
Gillet and Richard [9]	28	S. Sep+Patt-Rec	SVM	0.678	0.695	0.583	0.755

Table 1. Summary of drum transcription systems in polyphonic music. Almost all works classify bass drum (BD), snare drum (SD) and hi-hat (HH) sounds, except for Gillet and Richard [7] (where only BD and SD were detected) and Sandvold et al. [3] (where BD, SD and Cymbal were computed)."Approach" considers Pattern Recognition (Patt-Rec), Source Separation (S. Sep), and Spectral Template (Sp. Temp). F-measures results (except for Sandvold et al. where accuracy was measured). Since different datasets were used, results can not be directly compared.

the MIREX contest. It combined source separation (Non-Negative ICA) with template matching algorithms. Tanghe et al. [6], another MIREX participant, presented an onset-based classification system using N-binary SVM as recognition algorithm (being N the number of instruments to detect). Gillet and Richard [7] also used N-binary SVM as classification algorithm. This system performed a band-wise harmonic/noise decomposition as pre-processing step to enhance the presence of unpitched instruments. A localized adapted model like the one presented in [3] was also evaluated. Paulus and Klapuri [8] presented an evaluation using Hidden Markov Models with a combination of spectral features and temporal descriptors calculated from long narrow-band frames. In a recent work by Gillet and Richard [9] a combination of source-separation and pattern-recognition algorithms was proposed. Two set of features were computed, one from the original audio signal and the other from a "drum enhanced" track obtained by source separation. These feature vectors were then classified by three binary SVM.

As can be seen from the literature review, there is still a lot of room for improvement in the problem of drum transcription in polyphonic music. But, instead of pursuing a perfect transcription system, we decided to explore the potential of song-level percussion descriptors to describe real-world music. In order to achieve that we implemented a simple drum transcription system. This system could be used later as a baseline for more complex implementations (e.g. based on source separation or harmonic/noise decomposition). Thus, we chose to follow a standard pattern-recognition approach, trained on a large set of sounds and audio features. The potential of this kind of basic system to describe percussive events in polyphonic music was not previously assessed.

3. TRANSCRIPTION EXPERIMENTS

3.1 Datasets

We used three song collections with proper annotations of percussive events. Two of them are publicly available and were used in previous studies on drum transcription.

ENST-Drums database [10]: This is the largest publicly available drum database. Since we wanted to detect drum events in "real" music, we decided to mix the provided "wet" drums and their accompaniment tracks without further changes on amplitude (-6dB drum level). From the obtained collection of 64 songs we randomly selected 30 seconds excerpts of each song and their labels.

MAMI database [11]: This database is a collection of 52 music fragments (30 seconds length) extracted from commercial CDs. We managed to gather 48 songs and aligned them with the provided annotations. This database was one of the three databases used in the MIREX 2005 audio drum detection contest.

In-house database [12]: This is a database of 30 annotated music excerpts (20 seconds length), extracted from commercial CDs. Since HH events were not specifically annotated, we only used the BD and SD labels.

Due to the number of instances and the musical importance of each instrument within the drum kit we decided to work with the following instruments classes: BD, SD and HH (including open and closed HH sounds). Finally, we obtained a large set of polyphonic music excerpts adding up a total of 142 songs labeled with three, possibly concurrent, tags. The final number of instances per instrument was: 6,407 for BD, 5,655 for SD and 8,400 for HH.

3.2 Descriptors

First, we computed a set of frame-level descriptors (frame-size of 46 ms, hop-size of 12 ms) namely: temporal descriptors (zero-crossing rate and lpc coefficients), spectral descriptors (e.g. centroid, complexity, crest, decrease, dissonance, energy, flatness, flux, kurtosis, pitch, rms, rolloff, skewness, spread, strong-peak), perceptual descriptors (MFCCs, Bark-bands and Bark-bands kurtosis, skewness and spread) and tonal descriptors (Harmonic Pitch Class Profile). See [13] and [14, p. 20] for an overview on these descriptors.

After this first step we computed a set of object-level descriptors [3] from the time series of each frame-level descriptor (about 12 frames per object). The computed object-level descriptors were:

[3] The term "object" is considered here as: every sound event starting from an onset and finishing 150 ms after (or in the next onset if this new onset falls within the 150 ms interval).

a) Amplitude-related object descriptors: mean, variance, minimum, maximum, skewness and kurtosis.

b) Time-related object descriptors: temporal skewness, temporal kurtosis, temporal centroid, max. and min. normalized position (normalized temporal position of the maximum, or minimum, value of the time series), slope (arctangent of the slope of the linear regression of the data), attack and decay (slope descriptor from initial, or end, point to the maximum point) and amplitude-normalized attack and decay.

At the end of this process we obtained about 2,400 descriptors for every sound object. See [14, p. 46] for a detailed explanation on the computed descriptors.

3.3 Model training

Since we were working with three drum categories that can occur at the same time, we decided to train N-binary (SVM) classifiers instead of one model with 2^N possible classes. In this context we have each trained model in charge of detecting the presence or absence of one particular instrument (e.g. SD or not-SD).

In order to have a more representative database for training purposes we mixed the ENST and the MAMI databases. Taking into account that the final system has to label pre-detected onsets we decided to train our models with labeled onsets. Thus, we performed an onset detection (by using an implementation of the onset algorithm proposed by Brossier in [15]) and we assigned the corresponding labels to every detected onset. Finally we split the database leaving 90% for training and reserving 10% to be used as independent test set. We called these databases 90%MIX and 10%MIX. The In-House database was also reserved as second independent testing set.

To build the SVM models we first used the correlation based feature selection (CFS) [16] algorithm in 10-fold cross-validation (CV) to identify the most informative object-level descriptors from the 90%MIX database. We chose only those descriptors selected in all CVs (i.e. 10 times) obtaining 56 relevant descriptors for BD (e.g. low Bark-bands, MFCCs, spectral-energy low and spectral flux), 77 for SD (e.g. mid Bark-bands, temporal lpc, MFCCs and spectral flatness) and 38 for HH (e.g. high Bark-bands, temporal lpc, MFCCs, spectral spread and spectral flatness). Then, we trained the SVM models with the selected descriptors of the 90%MIX database and evaluated their performance using 10-fold CV. We also applied these models to the testing sets (i.e. 10%MIX and In-House). The classification results for every labeled instance and every model (after a grid search of SVM parameters) can be seen in Table 2. We obtained averaged F-measure results of 0.806 and 0.782 for the training and testing sets respectively.

3.4 Full transcription

Since up to this step we had worked only with labeled onsets, the next step was to evaluate the learned models against all the ground truth labels in the datasets. In order to do that we implemented a complete drum transcription system. The three previously described databases were analyzed (ENST, MAMI and In-house) adding up a total of 142 songs (20 to 30 seconds length).

Instrument Model	**90%MIX**	**In-House**	**10%MIX**
bass drum	0.834	0.812	0.835
snare	0.778	0.687	0.773
hi-hat	0.806	—	0.802

Table 2. F-measure classification results after grid search of SVM parameters. Models were trained with 90%MIX database. Results were evaluated using 10-fold CV on each dataset.

The experiment set-up for evaluating the transcription capabilities of our system was as follows: a) Perform an onset detection on the audio excerpts (we used the same onset detector as in the model training step). b) Compute the descriptors used by each model on every onset plus 150 ms (or until the next onset). c) Apply the models to every set of descriptors to obtain the predicted labels. d) Evaluate the predicted results against the ground truth annotations (as in the MIREX 2005 contest, a range of ± 30 ms from the true times was allowed). After evaluating all 142 song excerpts, we obtained an overall result of 0.659 (F-measure) and per instrument F-measure results of 0.699 for BD, 0.652 for SD and 0.626 for HH. If we compare our system with the fully automatic systems described in section 2 (i.e. [4, 6, 8, 9]) we can see that our system obtained near state-of-the-art drum transcription performance with a quite simple pattern recognition algorithm. Nevertheless, these performances are still far from reliable transcriptions.

3.5 Relaxed Transcription

Taking into account that state-of-the-art algorithms are still far from yielding perfect transcriptions and that our final goal was to derive song-level percussion descriptors, we decided to evaluate the capacity of our transcription system to estimate the total number of drum events in a song (e.g. how many BD, SD or HH strikes a particular song has). These descriptors could be used to characterize a song as having, for example, a lot of SD, no HH, etc., hence they contribute to bridge the semantic gap [17]. In this experiment we considered as a "correct" decision the total number of instrument instances (e.g. HH events) in the whole audio file discarding time-information[4]. Using the same datasets as in the full transcription experiments we obtained, as expected, better classification performance (F-measure) for all classes (BD = 0.822, HH = 0.794 and SD = 0.698). The overall performance of this "relaxed" transcription system was 0.771 (F-measure). These results encouraged us to investigate if useful song-wise percussion descriptors could be computed.

[4] We define correct transcription (CTR) as the $\arg\min(TR, GT)$, being TR = transcription and GT = ground truth labels per instrument. Then, we compute $P = CTR/TR$ and $R = CTR/GT$ and finally $F = 2PR/(P+R)$.

4. SONG-LEVEL PERCUSSION DESCRIPTORS

4.1 Computed Descriptors

In [12] and [18] two percussion-related descriptors were presented and evaluated with promising results namely: *Percussion Index* (a ratio between the total number of detected percussion events and the number of detected onsets) and *Percussion Profile* (the relative amount of BD, SD, cymbals, and non-percussion events normalized by the total number of onsets). Following this idea of percussion related descriptors we decided to compute and evaluate the following song-level percussion descriptors (some of these descriptors appeared as suggested future work in [12] but, up to our knowledge, they have not been implemented nor evaluated yet).

Computed song-level percussion descriptors:

- **Percussion Profile:** The ratio between the number of detected percussion events and the number of detected onsets [18]. Computed for BD, SD, HH and drum (D) [5] (e.g. BD/total, SD/total).
- **Inter-Instrument Ratio:** The ratio among all percussive instrument events namely: BD/SD, BD/HH and SD/HH.
- **Instrument Per Minute:** The number of detected events per minute for BD, SD, HH and D.
- **Inter-Instrument Interval (iii):** The first and second peak values of the histogram of the differences between successive events. Thus, we computed: first and second-iii-peak for BD, SD and HH.

At the end of this process we obtained 17 song-level percussion descriptors for each song.

4.2 Evaluation

To investigate the correlation between the proposed percussion descriptors and the ground truth values we computed the percussion descriptors both from the ground truth labels (labeled onsets) and from the output of our transcription system. Then, we built a fractional ranking [6] for each descriptor (for every song) and computed the Pearson's correlation coefficient between both rankings. The correlation results showed large correlation values (i.e. $>$ 0.5) for 12 out of 17 proposed descriptors (only: BD/SD, second-iii-peak for the three instruments and first-iii-peak for HH presented correlation values below 0.5). These highly correlated values between our descriptors and descriptors computed from the ground truth labels were specially strong (i.e. $>$ 0.7) for D/total, D/min, HH/total and HH/min. These results suggest that the proposed descriptors have potential to describe the percussive content of a song.

[5] In this context "drum" means the number of detected onsets labeled by the system as BD, SD or HH.

[6] If the ordered vector to rank is A,B,C,D and B is equal to C (i.e. tie) the fractional ranking assigns the same mean position value in both cases, i.e. 1,2.5,2.5,4.

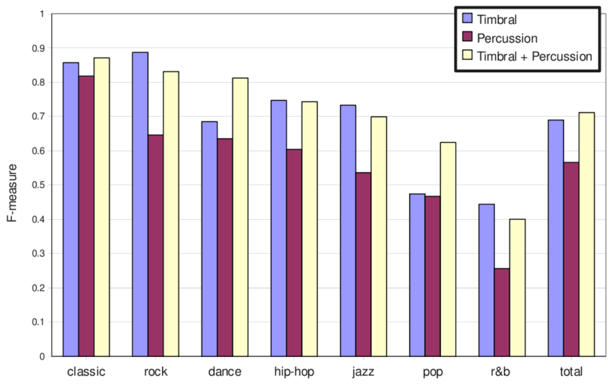


Figure 1. Genre classification results per genre and descriptor set. F-measure after 10-fold CV.

Next, we evaluated the usefulness of the percussion descriptors as features in several MIR tasks such as genre and sub-genre classification, danceability [7] and Western vs. non-Western music estimation. We decided to set-up a general methodology for evaluating the song-level percussion descriptors on every selected MIR task. Firstly, we computed, for each song in the dataset, the mean value of a set of "standard" descriptors to be used as baseline for the evaluation. Secondly, we computed the proposed song-level percussion descriptors on the same database. Thirdly, we selected a classification algorithm and we determined the "best" classification values for the "standard", "percussion" and "standard + percussion" descriptor sets. Finally, we evaluated the classification results by comparing F-measures and performing a Binomial test [20, p. 37] with 5% significance level (i.e. $\alpha = 0.05$). This Binomial test determines if the difference between correctly classified songs for each descriptor set is statistically significant or not. It is worth to notice that none of the songs used for training the SVM models were used in these evaluation experiments.

4.2.1 Genre

For genre classification we used an in-house database of 30 seconds excerpts extracted from 350 songs equally distributed among 7 genres: classic, dance, hip-hop, jazz, pop, r&b and rock. The computed "standard" descriptors were: Bark-bands, Bark-bands kurtosis, Bark-bands skewness, Bark-bands spread, spectral centroid, spectral crest, spectral decrease, spectral dissonance, spectral energy, spectral energy-band high, spectral energy-band low, spectral energy-band middle high, spectral energy-band middle low, spectral flatness, spectral flux, spectral hfc, spectral kurtosis, MFCCs, spectral skewness, spectral spread and temporal zero-crossing rate. We called "timbral" descriptors this set of 60 features. We used multi-class SVM as classification algorithm.

The genre classification results can be seen in Figure 1. From these results it is interesting to notice that by using the percussion descriptors only, good discrimination rates

[7] The easiness with which one can dance to a musical track [19].

Genre	T	P	T+P
ambient	0.531	0.433	0.588
drum'n bass	0.475	0.619	0.576
house	0.200	0.500	0.427
techno	0.369	0.269	0.380
trance	0.438	0.566	0.427
Average	**0.403**	**0.478**	**0.480**

Table 3. Electronic sub-genre classification. T = timbral, P = percussion and T+P = timbral+percussion. C4.5 classification algorithm. Results in F-measure after 10-fold CV.

can be achieved for classic ($F = 0.818$), rock, dance and hip-hop ($F \approx 0.600$) genres. The overall classification for the percussion-only data set was about 12 percentage points (pp) below "timbral" descriptors. When combining "timbral" and "percussion" descriptors a small improvement in the overall result was observed (+2.1 pp). It is worth to notice that big improvements were produced in dance (+12.7 pp) and pop (+15 pp) results, whereas results for rock and r&b decreased 5.6 and 4.4 pp respectively.

The Binomial test showed no statistically significant difference between "timbral + percussion" and "timbral" descriptors ($p = 0.1932$), but both sets evidenced significant differences with the "percussion" descriptor set ($p < 0.0001$ in both cases).

4.2.2 Electronic

For electronic sub-genre classification we performed our experiments on an in-house database of 270 songs (30 seconds length each) equally distributed among the following genres: ambient, drum'n bass, house, techno and trance. The computed descriptors were the same as in the genre experiment. Given that sub-genre and genre classification could be considered as very related tasks, we decided to try a different algorithm to gain some insight on the descriptors. Therefore, we used in this case the C4.5 decision tree algorithm for classification, since its output can be easily summarized into interpretable trees of descriptors.

Results for electronic sub-genre classification are depicted in Table 3. In this experiment we observed that classification results obtained by the "percussion" set outperformed "timbral" descriptors by 7.5 pp. The combination of "timbral" and "percussion" descriptors showed no significant difference with results from percussion-only descriptors in the overall classification result (although this combination seems to output more balanced classification rates among categories). In both "percussive" and "timbral + percussive" models the D/total and first and second iii-peak BD were the most informative descriptors.

The significance test corroborates the conclusions extracted from the F-measure results where "percussion" descriptors performed significatively better than "timbral" descriptors ($p = 0.0028$), "timbral + percussion" performed better than "timbral" ($p = 0.0058$) and no statistical difference between "percussion" and "percussion + timbral" descriptor sets was appreciated ($p = 0.4273$).

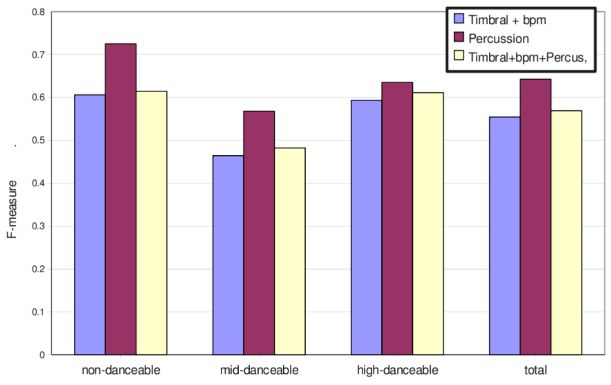


Figure 2. Danceability classification results after 10-fold CV.

4.2.3 Danceability

For danceability tests we used an in-house database of 374 song excerpts of 30 seconds equally distributed into three classes (i.e. non-dance., mid-dance. and high-dance.). We computed the same descriptors as in the genre experiment plus an estimation on the beats per minute (bpm) of the song [8], we called this descriptor set as "timbral + bpm". As in the genre experiments we decided to use the SVM algorithm (multi-class).

Results for Danceability tests are shown in Figure 2. From these results we can conclude that "percussion" descriptors performed better than both "timbral + bpm" and "timbral + bpm + percussion". Percussion-only descriptors outperformed by 8.9 pp and 7.4 pp "timbral + bpm" and "timbral + bpm + percussion" respectively, obtaining better results in all three categories. It is interesting to notice that percussion descriptors also outperformed obtained results by [19] which achieved an accuracy of 61.78% in classifying 225 songs into the same three categories by using a different and more complex approach.

The Binomial test on danceability results showed that "percussion" descriptors provided significatively better performance than the other two sets ($p = 0.0025$ for "timbral + bpm" and $p = 0.0074$ for "timbral + bpm + percussion"). The test also showed no statistical difference between "timbral + bpm" and "timbral + bpm + percussion" descriptor sets ($p = 0.3793$).

4.2.4 Western vs. non-Western music classification

For Western vs. non-Western experiments we used an in-house database of 139 Western songs from 16 genres including classical, jazz, rock, pop, religious and hip-hop, and 139 non-Western songs including songs from Africa, Arab States, Asia and the Pacific. The computed descriptors and classification algorithm were the same as in genre experiments.

The results for these experiments are shown in Table 4. Here we observed an almost linear increment in the classification rates starting by "timbral" descriptors with

[8] Since bpm is an important descriptor for danceability estimation we included it into the "standard" set. Otherwise, it would be too easy for our descriptors to outperform.

Class	T	P	T+P
Western	0,817	0,803	0,856
non-Western	0,747	0,828	0,833
Average	**0,782**	**0,816**	**0,844**

Table 4. Western vs. non-Western music classification. T = timbral, P = percussion and T+P = timbral+percussion. SVM classification algorithm, F-measure results after 10-fold CV.

$F = 0.782$ followed by "percussion" descriptors with $F = 0.816$ (+3.4 pp) and "timbral + percussion" with $F = 0.844$ (+2.8 pp from "percussion"). It seems clear that adding percussion descriptors helped in the process of Western vs. non-Western song discrimination. Is is also interesting to notice that classification results for non-Western music were much better when percussion descriptors were used (more than 8 pp above "timbral").

The significance test showed no statistically significant difference between "percussion + timbral" and "percussion" descriptors ($p = 0.1212$) and between "percussion" and "timbral" descriptors ($p = 0.1349$). The test also depicted statistical difference between "percussion + timbral" and "timbral" descriptors ($p = 0.0096$). See [21] for an in-depth study on Western vs. non-Western music classification.

5. CONCLUSIONS

Within the present work we have conducted several experiments in order to detect and describe percussive events in polyphonic music. Firstly, we built, by combining three databases, a large set of percussion-labeled songs. Secondly, we evaluated the capacity of an automatic drum transcription system, based on object-level features and three binary SVM models, to transcribe percussion events in polyphonic music. From the transcription results we extrapolated that our relatively simple algorithm can be placed among the top ranked ones, even though all these systems leave a lot of room for improvement. After performing "relaxed" transcription experiments we observed that our system can detect the total number of drum events in a song with an overall F-measure of 0.771. Finally, we presented 17 song-level percussion descriptors and we evaluated their usefulness among several MIR tasks. These preliminary results suggest that song-level percussion (i.e. "semantic") descriptors, even though they are based on imperfect transcriptions, can help in MIR tasks such as genre and sub-genre classification, danceability and Western vs. non-Western music estimation. It also seems clear that song-level percussion descriptors offer useful information that complements the one provided by classic "spectral" and "timbral" descriptors. This new information could also be exploited in music similarity tasks.

6. REFERENCES

[1] D. Fitzgerald and J. Paulus. *Unpitched Percussion Transcription*, chapter 5, pages 131–162. Signal Processing Methods for Music Transcription. Springer, 2006.

[2] P. Herrera, A. Dehamel, and F. Gouyon. Automatic labeling of unpitched percussion sounds. In *Proc. of AES 114th*, Amsterdam, The Netherlands, 2003.

[3] V. Sandvold, F. Gouyon, and P. Herrera. Percussion classification in polyphonic audio recordings using localized sound models. In *Proc. of ISMIR*, pages 537–540, Barcelona, Spain, 2004.

[4] K. Yoshii, M. Goto, and H. G. Okuno. Adamast: A drum sound recognizer based on adaptation and matching of spectrogram. In *MIREX*, London, UK, 2005.

[5] C. Dittmar. Drum detection from polyphonic audio via detailed analysis of the time frequency domain. In *MIREX*, London, UK, 2005.

[6] K. Tanghe, S. Degroeve, and B. De Baets. An algorithm for detecting and labeling drum events in polyphonic music. In *MIREX*, London, UK, 2005.

[7] O. Gillet and G. Richard. Drum track transcription of polyphonic music using noise subspace projection. In *Proc. of ISMIR*, pages 92–99, London, UK, 2005.

[8] J. Paulus and A. Klapuri. Combining temporal and spectral features in hmm-based drum transcription. In *Proc. of ISMIR*, pages 225–228, Vienna, Austria, 2007.

[9] O. Gillet and G. Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.

[10] O. Gillet and G. Richard. ENST-Drums: an extensive audiovisual database for drum. In *Proc. of ISMIR*, pages 156–159, Victoria, Canada, 2006.

[11] K. Tanghe, M. Lesaffre, S. Degroeve, M. Leman, B. De Baets, and J. Martens. Collecting ground truth annotations for drum detection in polyphonic music. In *Proc. of ISMIR*, pages 50–57, London, UK, 2005.

[12] V. Sandvold. Percussion descriptors. A semantic approach to music information retrieval. Master's thesis, University of Oslo, 2004.

[13] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, CUIDADO project, 2004.

[14] M. Haro. Detecting and describing percussive events in polyphonic music. Master's thesis, Universitat Pompeu Fabra, 2008.

[15] P. Brossier. *Automatic annotation of musical audio for interactive systems.* PhD thesis, Queen Mary University of London, 2006.

[16] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 359–366, San Francisco, USA, 2000.

[17] O. Celma, P. Herrera, and X. Serra. Bridging the music semantic gap. In *Workshop on Mastering the Gap: From Information Extraction to Semantic Representation*, volume 187, Budva, Montenegro, 2006. CEUR.

[18] P. Herrera, V. Sandvold, and F. Gouyon. Percussion-related semantic descriptors of music audio files. In *Proc. of AES 25th*, London, UK, 2004.

[19] S. Streich and P. Herrera. Detrended fluctuation analysis of music signals danceability estimation and further semantic characterization. In *Proc. AES 118th*, Barcelona, Spain, 2005.

[20] P. H. Kvam and B. Vidakovic. *Nonparametric Statistics with Applications to Science and Engineering*. Wiley, 2007.

[21] E. Gómez, M. Haro, and P. Herrera. Music and geography: Content description of musical audio from different parts of the world. In *Proc. of ISMIR*, Kobe, Japan, 2009.

MUSIC GENRE CLASSIFICATION USING LOCALITY PRESERVING NON-NEGATIVE TENSOR FACTORIZATION AND SPARSE REPRESENTATIONS

Yannis Panagakis[*] **Constantine Kotropoulos**[*,†]
[*]Dept. of Informatics
Aristotle University of Thessaloniki
Box 451 Thessaloniki, GR-54124, Greece
{panagakis,costas}@aiia.csd.auth.gr

Gonzalo R. Arce[†]
[†]Dept. of Electrical & Computer Engineering
University of Delaware
Newark, DE 19716-3130, U.S.A.
arce@ece.udel.edu

ABSTRACT

A robust music genre classification framework is proposed that combines the rich, psycho-physiologically grounded properties of auditory cortical representations of music recordings and the power of sparse representation-based classifiers. A novel multilinear subspace analysis method that incorporates the underlying geometrical structure of the cortical representations space into non-negative tensor factorization is proposed for dimensionality reduction compatible to the working principle of sparse representation-based classification. The proposed method is referred to as *Locality Preserving Non-Negative Tensor Factorization* (LPNTF). Dimensionality reduction is shown to play a crucial role within the classification framework under study. Music genre classification accuracy of 92.4% and 94.38% on the GTZAN and the ISMIR2004 Genre datasets is reported, respectively. Both accuracies outperform any accuracy ever reported for state of the art music genre classification algorithms applied to the aforementioned datasets.

1. INTRODUCTION

Despite the lack of a commonly agreed definition of music genre due to genre dependence on cultural, artistic, or market factors and the rather fuzzy boundaries between different genres, music genre is probably the most popular description of music content [1].

Psycho-physiology indicates that the acoustic stimulus is encoded in the primary auditory cortex by its spectral and temporal characteristics. This is accomplished by cells whose responses are selective to a range of spectral and temporal resolutions resulting into a neural representation. In particular, when the acoustic stimulus is either speech or music, its perceptual properties are encoded by slow spectral and temporal modulations [13, 18].

The appealing properties of slow spectro-temporal modulations from the human perceptual point of view and the strong theoretical foundations of sparse representations [4, 6] have motivated us to propose a robust framework for automatic music genre classification here. To this end, the auditory model [17] is used in order to map a given music recording to a three-dimensional (3D) representation of its slow spectral and temporal modulations with the same parameters as in [15]. This 3D representation is referred to as *cortical representation* and exploits the properties of the human auditory system [18]. The cortical representations form an overcomplete dictionary of basis signals for music genres, which is exploited for *sparse representation-based classification* (SRC) as proposed in [19]. That is, first each music recording is represented by its cortical representation. Second, each cortical representation is modeled as a sparse weighted sum of the basis elements (atoms) of an overcomplete dictionary, which stems from the cortical representations associated to training music recordings whose genre is known. If sufficient training music recordings are available for each genre, it is possible to express any test cortical representation as a compact linear combination of the dictionary atoms of the genre, where it actually belongs to. This representation is designed to be sparse, because it involves only a small fraction of the dictionary atoms and can be computed efficiently via ℓ_1 optimization. The classification is performed by assigning each test recording to the class associated with the dictionary atoms, that are weighted by non-zero coefficients.

Since we would like to build an overcomplete dictionary extracted from training cortical representations, the dimensionality of dictionary atoms must be much smaller than the cardinality of the training set. Such a dimensionality reduction facilitates the treatment of missing data, noise, and outliers. Conventional linear subspace analysis methods, such as Principal Component Analysis, Linear Discriminant Analysis, and Non-Negative Matrix Factorization (NMF) deal only with vectorial data. By vectorizing a typical 3D cortical representation of 6 scales, 10 rates, and 128 frequency bands, a vector of dimensions 7680×1 results. Handling such high-dimensional patterns is computationally expensive not to mention that eigenanalysis cannot be easily performed. Despite the implementation issues, by reshaping a 3D cortical representation into a vector the natural structure of the original data is destroyed. Thus, dimensionality reduction applied directly to tensors rather than their vectorized versions is desirable. Unsupervised multilinear dimensionality reduction techniques, such as Non-Negative Tensor Factorization (NTF) [2] or Multilinear Principal Component Analysis (MPCA) [12] as well as supervised ones including General Tensor Discriminant Analysis (GTDA) [20] or Discriminant Non-Negative Tensor Factorization (DNTF) [21] could be considered. However, the just mentioned methods do not take into account the geometrical structure of the original data space. To reduce tensor dimensions in a consistent manner with the working principle of SRC, we should guarantee that two data points, which are close in the intrinsic geometry of the original data space are also close in the new data space after multilinear dimensionality reduction. To this end, we propose a novel algorithm, where the geometrical information of the original data space is incorporated into the objective function optimized by NTF. In particular, we encode the geometrical information by constructing a nearest neighbor graph. Furthermore, the non-negativity of cortical representations is preserved to maintain their physical interpretation. The proposed method is referred to as Locality Preserving Non-Negative Tensor Factorization (LPNTF). We derive a multiplicative updating algorithm for LPNTF, which extracts features from the cortical representations. For comparison purposes, NTF, MPCA, GTDA, DNTF, and random projections are also tested.

Next, the features extracted by the aforementioned multilinear dimensionality techniques are classified by SRC. Performance comparisons are made against the SVMs employing a linear kernel. The reported genre classification accuracies are juxtaposed against the best ones achieved by the state of the art algorithms applied to the GTZAN and ISMIR2004 Genre datasets. More specifically, two sets of experiments are conducted. First, stratified ten-fold cross-validation is applied to the GTZAN dataset. The proposed genre classification method, that extracts features using the LPNTF, which are then classified by SRC (i.e. LPNTF plus SRC), yields an accuracy of 92.4%. Second, experiments on the ISMIR2004Genre dataset are conducted by adhering to the protocol employed during ISMIR2004 evaluation tests. This protocol splits the dataset into two equal disjoint subsets with the first one being used for training and the second one being used for testing. Features extracted by NTF, which are then classified by SRC, yield an accuracy of 94.38%. An accuracy of 94.25% was achieved when the LPNTF plus SRC framework is employed. To the best of the authors' knowledge, the just quoted genre classification accuracies are **the highest ever reported for both datasets**.

The paper is organized as follows. In Section 2, basic multilinear algebra concepts and notations are defined. The LPNTF is introduced in Section 3. The SRC framework, that is applied to music genre classification, is detailed in Section 4. Experimental results are demonstrated in Section 5 and conclusions are drawn in Section 6.

2. NOTATION AND MULTILINEAR ALGEBRA BASICS

Tensors are considered as the multidimensional equivalent of matrices (i.e., second-order tensors) and vectors (i.e., first-order tensors) [9]. Throughout this paper, tensors are denoted by boldface Euler script calligraphic letters (e.g. $\mathcal{X}$, $\mathcal{A}$), matrices are denoted by uppercase boldface letters (e.g. $\mathbf{U}$), and vectors are denoted by lowercase boldface letters (e.g. $\mathbf{u}$).

A high-order real valued tensor $\mathcal{X}$ of order N is defined over the tensor space $\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, where $I_i \in \mathbb{Z}$ and $i = 1, 2, \ldots, N$. Each element of tensor $\mathcal{X}$ is addressed by N indices, i.e. $x_{i_1 i_2\, i_3 \ldots i_N}$. Mode-$n$ unfolding of tensor $\mathcal{X}$ yields the matrix $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1\, \ldots I_{n-1} I_{n+1} \ldots I_N)}$. In the following, the operations on tensors are expressed in matricized form [9].

An N-order tensor $\mathcal{X}$ has rank 1, when it is decomposed as the outer product of N vectors $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(N)}$, i.e. $\mathcal{X} = \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \ldots \circ \mathbf{u}^{(N)}$. That is, each element of the tensor is the product of the corresponding vector elements, $x_{i_1 i_2 \ldots i_N} = u^{(1)}_{i_1}\, u^{(2)}_{i_2}\, \ldots\, u^{(N)}_{i_N}$ for all $i_n = 1, 2, \ldots, I_n$. The rank of an arbitrary N-order tensor $\mathcal{X}$ is the minimal number of rank-1 tensors that yield $\mathcal{X}$ when linearly combined. Next, several products between matrices will be used, such as the Kronecker product denoted by $\otimes$, the Khatri-Rao product denoted by $\odot$, and the Hadamard product denoted by $*$, whose definitions can be found in [9] for example.

3. LOCALITY PRESERVING NON NEGATIVE TENSOR FACTORIZATION

Let $\{\mathcal{X}_q|_{q=1}^{Q}\}$ be a set of Q non-negative tensors $\mathcal{X}_q \in \mathbb{R}_{+}^{I_1 \times I_2 \times \ldots \times I_N}$ of order N. Let us also assume that these Q tensors lie in a nonlinear manifold $\mathcal{A}$ embedded into the tensor space $\mathbb{R}_{+}^{I_1 \times I_2 \times \ldots \times I_N}$. Accordingly, we can represent such a set by a $(N+1)$-order tensor $\mathcal{A} \in \mathbb{R}_{+}^{I_1 \times I_2 \times \ldots \times I_N \times I_{N+1}}$ with $I_{N+1} = Q$. Conventional NTF operates in the Euclidean space and does not consider the intrinsic geometrical structure of the data manifold [2]. To overcome the just mentioned limitation of NTF, we propose LPNTF by incorporating a geometrically-based regularizer stemming from locality preserving projections [7] into the optimization problem to be solved.

Given $\{\mathcal{X}_q|_{q=1}^{Q}\}$, one can model the local structure of $\mathcal{A}$ by constructing the nearest neighbor graph $\mathcal{G}$. By exploiting the heat kernel function [7], one can define the elements of the weight matrix $\mathbf{S}$ of $\mathcal{G}$ as $s_{qp} = e^{-\frac{||\mathcal{X}_q - \mathcal{X}_p||^2}{\tau}}$ if $\mathcal{X}_q$ and $\mathcal{X}_p$ belong to the same class and 0 otherwise, where $||\ ||^2$ denotes the tensor norm [9]. Accordingly, the Laplacian matrix is defined as $\mathbf{L} = \mathbf{\Gamma} - \mathbf{S}$, where $\mathbf{\Gamma}$ is a diagonal matrix with elements $\gamma_{qq} = \sum_p s_{qp}$, i.e. the column sums of $\mathbf{S}$. Let $\mathbf{Z}^{(i)} = \mathbf{U}^{(N+1)} \odot \ldots \odot \mathbf{U}^{(i+1)} \odot \mathbf{U}^{(i-1)} \odot \ldots \odot \mathbf{U}^{(1)}$. Since the Laplacian matrix is analogous to Laplace-Beltrami operator on compact Riemannian manifolds [7], one can incorporate the local geometry of $\mathcal{A}$ into NTF by constructing the following objective function for LPNTF in matrix form:

$$f_{LPNTF}\big(\mathbf{U}^{(i)}|_{i=1}^{N+1}\big) = ||\mathbf{A}_{(i)} - \mathbf{U}^{(i)}\big[\mathbf{Z}^{(i)}\big]^T||^2 + \lambda \,\mathrm{tr}\Big\{\big[\mathbf{U}^{(N+1)}\big]^T \mathbf{L}\,\mathbf{U}^{(N+1)}\Big\}, \quad (1)$$

where $\lambda > 0$ is a parameter, which controls the trade off between goodness of fit to the data tensor $\mathcal{A}$ and locality preservation. Consequently, we propose to minimize (1) subject to the non-negativity constraint on factor matrices $\mathbf{U}^{(i)} \in \mathbb{R}_+^{I_i \times k}$, $i = 1, 2, \ldots N+1$, where k is the desirable number of rank-1 tensors approximating $\mathcal{A}$ when linearly combined.

Let $\nabla_{\mathbf{U}^{(i)}} f_{LPNTF} = \frac{\partial f_{LPNTF}}{\partial \mathbf{U}^{(i)}}$ be the partial derivative of the objective function $f_{LPNTF}(\mathbf{U}^{(i)}|_{i=1}^{N+1})$ with respect to $\mathbf{U}^{(i)}$. Since $\mathbf{U}^{(i)}$, $i = 1, 2, \ldots, N+1$, $\mathbf{\Gamma}$, and $\mathbf{S}$ are non-negative, the partial derivatives of the objective function can be decomposed as differences of two non-negative components denoted by $\nabla^+_{\mathbf{U}^{(i)}} f_{LPNTF}$ and $\nabla^-_{\mathbf{U}^{(i)}} f_{LPNTF}$, respectively. It can be shown that for $i = 1, 2, \ldots, N$ we have

$$\nabla_{\mathbf{U}^{(i)}} f_{LPNTF} = \underbrace{\mathbf{U}^{(i)}\big[\mathbf{Z}^{(i)}\big]^T \mathbf{Z}^{(i)}}_{\nabla^+_{\mathbf{U}^{(i)}} f_{LPNTF}} - \underbrace{\mathbf{A}_{(i)}\mathbf{Z}^{(i)}}_{\nabla^-_{\mathbf{U}^{(i)}} f_{LPNTF}}, \quad (2)$$

while for $i = N+1$ by invoking the definition of the Laplacian we obtain

$$\nabla_{\mathbf{U}^{(N+1)}} f_{LPNTF} = \underbrace{\mathbf{U}^{(N+1)}\big[\mathbf{Z}^{(N+1)}\big]^T \mathbf{Z}^{(N+1)} + \lambda\, \mathbf{\Gamma} \mathbf{U}^{(N+1)}}_{\nabla^+_{\mathbf{U}^{(N+1)}} f_{LPNTF}} - \underbrace{\big(\mathbf{A}_{(N+1)}\mathbf{Z}^{(N+1)} + \lambda\, \mathbf{S}\, \mathbf{U}^{(N+1)}\big)}_{\nabla^-_{\mathbf{U}^{(N+1)}} f_{LPNTF}}. \quad (3)$$

Following the strategy employed in the derivation of NMF [10], we can obtain an iterative alternating algorithm for LPNTF as follows. Given $N+1$ randomly initialized nonnegative matrices $\mathbf{U}^{(i)}|_{i=1}^{N+1} \in \mathbb{R}_+^{I_i \times k}$, a local minimum of the optimization problem (1) subject to non-negativity constraints can be found by the multiplicative update rule:

$$\mathbf{U}^{(i)}_{[t+1]} = \mathbf{U}^{(i)}_{[t]} * \frac{\nabla^-_{\mathbf{U}^{(i)}_{[t]}} f_{LPNTF}}{\nabla^+_{\mathbf{U}^{(i)}_{[t]}} f_{LPNTF}}, \quad (4)$$

where the division in (4) is elementwise and t denotes the iteration index. The multiplicative update rule (4) suffers from two drawbacks: (1) The denominator may be zero; (2) $\mathbf{U}^{(i)}_{[t+1]}$ does not change when $\mathbf{U}^{(i)}_{[t]} = \mathbf{0}$ and $\nabla_{\mathbf{U}^{(i)}[t]}$ $f_{LPNTF} < \mathbf{0}$. In order to overcome these drawbacks, we can modify (4) as in [11]. A robust multiplicative update rule for LPNTF is then

$$\mathbf{U}^{(i)}_{[t+1]} = \mathbf{U}^{(i)}_{[t]} - \frac{\bar{\mathbf{U}}^{(i)}_{[t]}}{\nabla^+_{\mathbf{U}^{(i)}_{[t]}} f_{LPNTF} + \delta} * \nabla_{\mathbf{U}^{(i)}_{[t]}} f_{LPNTF}, \quad (5)$$

where $\bar{\mathbf{U}}^{(i)}_{[t]} = \mathbf{U}^{(i)}_{[t]}$ if $\nabla_{\mathbf{U}^{(i)}_{[t]}} f_{LPNTF} \geq \mathbf{0}$ and σ otherwise. The paremeters σ, δ are predefined small positive numbers, typically 10^{-8} [11].

4. SPARSE REPRESENTATION-BASED CLASSIFICATION

For each music recording a 3D cortical representation is extracted by employing the computational auditory model of Wang *et al.* [17] with the same parameters as in [15]. Thus, each ensemble of recordings is represented by a 4th-order data tensor, which is created by stacking the 3rd-order feature tensors associated to the recordings. Consequently, the data tensor $\mathcal{A} \in \mathbb{R}_+^{I_1 \times I_2 \times I_3 \times I_4}$, where $I_1 = I_{scales} = 6$, $I_2 = I_{rates} = 10$, $I_3 = I_{frequencies} = 128$, and $I_4 = I_{samples}$ is obtained.

Determining the class label of a test cortical representation, given a number of labeled training cortical representations from N music genres is addressed based on SRC [19]. Let us denote by $\mathbf{A}_i = [\mathbf{a}_{i1}|\mathbf{a}_{i2}|\ldots|\mathbf{a}_{in_i}] \in \mathbb{R}_+^{7680 \times n_i}$ the dictionary that contains n_i cortical representations stemming from the ith genre as column vectors (i.e., atoms). Given a test cortical representation $\mathbf{y} \in \mathbb{R}_+^{7680}$ that belongs to the ith class, we can assume that $\mathbf{y}$ is expressed as a linear combination of the atoms that belong to the ith class, i.e.

$$\mathbf{y} = \sum_{j=1}^{n_i} \mathbf{a}_{ij}\, c_{ij} = \mathbf{A}_i\, \mathbf{c}_i, \quad (6)$$

where $c_{ij} \in \mathbb{R}$ are coefficients, which form the coefficient vector $\mathbf{c}_i = [c_{i1}, c_{i2}, \ldots, c_{in_i}]^T$. Let us, now, define the matrix $\mathbf{D} = [\mathbf{A}_1|\mathbf{A}_2|\ldots|\mathbf{A}_N] = \mathbf{A}_{(4)}^T \in \mathbb{R}_+^{7680 \times I_{samples}}$ by concatenating $I_{samples}$ cortical representations, which are distributed across N genres. Accordingly, a test cortical representation $\mathbf{y}$ that belongs to the ith genre can be equivalently expressed as

$$\mathbf{y} = \mathbf{D}\,\mathbf{c}, \quad (7)$$

where $\mathbf{c} = [\mathbf{0}^T|\ldots|\mathbf{0}^T|\mathbf{c}_i^T|\mathbf{0}^T|\ldots|\mathbf{0}^T]^T$ is the augmented coefficient vector whose elements are zero except those associated with the ith genre.

Since the genre label of any test cortical representation is unknown, we can predict it by seeking the sparsest solution to the linear system of equations (7). Let $||.||_0$ be the ℓ_0 quasi-norm of a vector, which returns the number of its non-zero elements. Formally, given the matrix $\mathbf{D}$ and the test cortical representation $\mathbf{y}$, sparse representation aims to find the coefficient vector $\mathbf{c}$ such that (7) holds and $||\mathbf{c}||_0$ is minimum, i.e.

$$\mathbf{c}^\star = \arg\min_{\mathbf{c}} ||\mathbf{c}||_0 \quad \text{subject to } \mathbf{D}\mathbf{c} = \mathbf{y}. \quad (8)$$

(8) is NP-hard due to the underlying combinational optimization. An approximate solution to (8) can be obtained by replacing the ℓ_0 norm with the ℓ_1 norm, i.e.

$$\mathbf{c}^\star = \arg\min_{\mathbf{c}} ||\mathbf{c}||_1 \quad \text{subject to } \mathbf{D}\,\mathbf{c} = \mathbf{y}, \quad (9)$$

where $||.||_1$ denotes the ℓ_1 norm of a vector. The optimization problem (9) can be solved by standard linear programming methods in polynomial time [5].

Since we are interested in creating overcomplete dictionaries derived from the cortical representations, the dimensionality of atoms must be much smaller than the training set cardinality. Thus, we can reformulate the optimization problem in (9) as follows:

$$\mathbf{c}^\star = \arg\min_{\mathbf{c}} ||\mathbf{c}||_1 \quad \text{subject to } \mathbf{W}\,\mathbf{D}\,\mathbf{c} = \mathbf{W}\mathbf{y}, \qquad (10)$$

where $\mathbf{W} \in \mathbb{R}^{k \times 7680}$ with $k \ll \min(7680, I_{samples})$ is a projection matrix. The projection matrix $\mathbf{W}$ can be obtained by LPNTF or any other multilinear dimensionality reduction technique, such as NTF [2], MPCA [12], GTDA [20], or DNTF [21]. Alternatively, one can even employ a random projection matrix whose elements are independently sampled from a zero-mean normal distribution, and each column is normalized to unit length as proposed in [19]. More particularly, when LPNTF, NTF, or DNTF is applied to the data tensor $\boldsymbol{\mathcal{A}}$, four factor matrices $\mathbf{U}^{(i)} \in \mathbb{R}_+^{I_i \times k}$, $i = 1, 2, 3, 4$, are obtained, which are associated to scale, rate, frequency, and sample modes respectively. The projection matrix $\mathbf{W}$ is given by either $\mathbf{W} = (\mathbf{U}^{(3)} \odot \mathbf{U}^{(2)} \odot \mathbf{U}^{(1)})^T$ or $\mathbf{W} = (\mathbf{U}^{(3)} \odot \mathbf{U}^{(2)} \odot \mathbf{U}^{(1)})^\dagger$, where $(.)^\dagger$ denotes the Moore-Penrose pseudoinverse. Accordingly, every column of $\mathbf{D}$ (i.e. vectorized cortical representation of a music recording) is a linear combination of the basis vectors, which span the columns of the basis matrix $\mathbf{W}^T$ with coefficients taken from the columns of matrix $[\mathbf{U}^{(4)}]^T$. That is, $\mathbf{D} = \mathbf{A}_{(4)}^T = \mathbf{W}^T [\mathbf{U}^{(4)}]^T$. For MPCA or GTDA, three factor matrices $\mathbf{U}^{(i)} \in \mathbb{R}^{I_i \times J_i}$, with $J_i < I_i$, $i = 1, 2, 3$, are obtained, which are associated to scales, rates, and frequencies, respectively. The columns of $\mathbf{D}$ are obtained by applying the projection matrix $\mathbf{W} = (\mathbf{U}^{(3)} \otimes \mathbf{U}^{(2)} \otimes \mathbf{U}^{(1)})^T$ or $\mathbf{W} = (\mathbf{U}^{(3)} \otimes \mathbf{U}^{(2)} \otimes \mathbf{U}^{(1)})^\dagger$ to vectorized training tensors $\text{vec}(\mathcal{X}_q)$. The dimensionality reduction of the original cortical representations data space has two benefits: (1) It reduces the computational cost of linear programming solvers for (9) [5]; (2) It facilitates the creation of a redundant dictionary out of training cortical representations.

A test cortical representation can be classified as follows. First, $\mathbf{y}$ is projected onto the reduced dimensionality space through the projection matrix $\mathbf{W}$ as $\hat{\mathbf{y}} = \mathbf{W}\mathbf{y}$. Then, the following optimization problem is solved

$$\mathbf{c}^\star = \arg\min_{\mathbf{c}} ||\mathbf{c}||_1 \quad \text{subject to } \mathbf{W}\,\mathbf{D}\,\mathbf{c} = \hat{\mathbf{y}}. \qquad (11)$$

Ideally, the coefficient vector $\mathbf{c}^\star$ contains non-zero entries in positions associated with the columns of $\mathbf{WD}$ associated with a single genre, so that we can easily assign the test auditory representation $\mathbf{y}$ to that genre. However, due to modeling errors, there are small non-zero elements in $\mathbf{c}^\star$ that are associated to multiple genres. To cope with this problem, each auditory modulation representation is classified to the genre that minimizes the ℓ_2 norm residual between $\hat{\mathbf{y}}$ and $\check{\mathbf{y}} = \mathbf{W}\,\mathbf{D}\,\vartheta_i(\mathbf{c})$, where $\vartheta_i(\mathbf{c}) \in \mathbb{R}^n$ is a new vector whose non-zero entries are only the elements in $\mathbf{c}$ that are associated to the ith genre [19].

5. EXPERIMENTAL EVALUATION

In order to assess both the discriminating power of the features derived by LPNTF applied to cortical representations for dimensionality reduction and the accuracy of sparse representation-based classification, experiments are conducted on two widely used datasets for music genre classification [3, 8, 14, 16]. The first dataset, abbreviated as GTZAN, was collected by G. Tzanetakis [16] and consists of 10 genre classes. Each genre class contains 100 audio recordings 30 sec long. The second dataset, abbreviated as ISMIR2004 Genre, comes from the ISMIR 2004 Genre classification contest and contains 1458 full audio recordings distributed across 6 genre classes. All the recordings were converted to monaural wave format at a sampling frequency of 16 kHz and quantized with 16 bits. Moreover, the music signals have been normalized, so that they have zero mean amplitude with unit variance in order to remove any factors related to the recording conditions. Since the ISMIR2004 Genre dataset consists of full length tracks, we extracted a segment of 30 sec just after the first 30 sec of a recording in order to exclude any introductory parts that may not be directly related to the music genre the recording belongs to. The cortical representation is extracted for the aforementioned segment of 30 sec duration for any recording from both datasets. The best reported music genre classification accuracies obtained for the aforementioned datasets are summarized in Table 1.

Reference	Dataset	Accuracy
Bergstra *et al.* [3]	GTZAN	82.5%
Holzapfel *et al.* [8]	ISMIR2004	83.5%
Pampalk *et al.* [14]	ISMIR2004	82.3%

Table 1. Best classification accuracies achieved by music genre classification approaches on standard datasets.

Following the experimental set-up used in [3, 15, 16], stratified 10-fold cross-validation is employed for experiments conducted on the GTZAN dataset. Thus each training set consists of 900 audio files. Accordingly, the training tensor $\boldsymbol{\mathcal{A}}_{GTZAN} \in \mathbb{R}_+^{6 \times 10 \times 128 \times 900}$ is constructed by stacking the cortical representations. The experiments on ISMIR 2004 Genre dataset were conducted according to the ISMIR2004 Audio Description Contest protocol. The protocol defines training and evaluation sets, which consist of 729 audio files each. Thus the corresponding training tensor $\boldsymbol{\mathcal{A}}_{ISMIR} \in \mathbb{R}_+^{6 \times 10 \times 128 \times 729}$ is constructed.

The projection matrix $\mathbf{W}$ is derived from each training tensor $\boldsymbol{\mathcal{A}}_{GTZAN}$ and $\boldsymbol{\mathcal{A}}_{ISMIR}$ by employing either LPNTF, NTF, DNTF, MPCA or GTDA. Throughout the experiments the value of λ in LPNTF was empirically set to 0.5, while the parameter τ of the heat kernel was set equal to 1. In order to determine automatically the parameters λ and τ one can apply cross-validation to the training set. However, the systematic setting of these parameters could be a subject of future research. In order to determine the dimensionality of factor matrices, the ratio of the sum of eigenvalues retained over the sum of all eigenvalues for each mode-n tensor unfolding is employed as in [12]. By using this ratio as a specification parameter, the number of retained principal components for each mode (e.g. scale, rate, and frequency) was determined, as is demon-

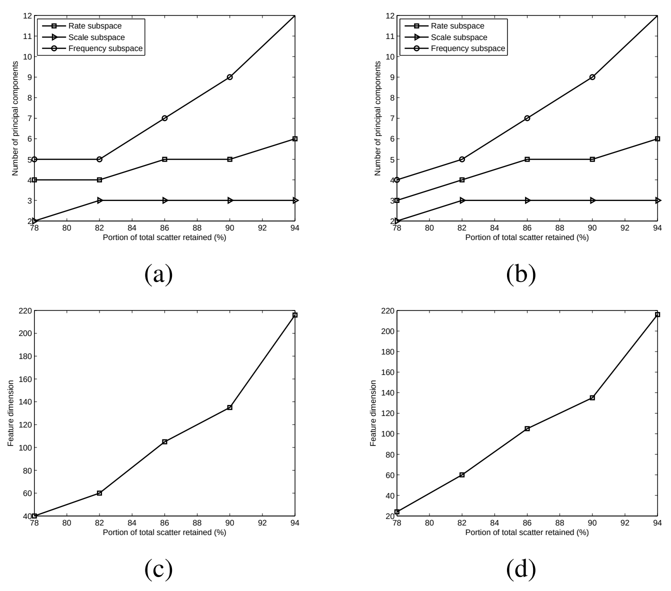


Figure 1. Total number of retained principal components in each mode (e.g. rate, scale, and frequency) as a function of the portion of total scatter retained for the: a) GTZAN dataset and b) ISMIR 2004 Genre dataset. Feature dimension as a function of the portion of the total scatter retained for the: c) GTZAN dataset and d) ISMIR 2004 Genre dataset.

strated in Figure 1 for the GTZAN and the ISMIR Genre 2004 datasets. The different subspace analysis methods are compared for equal dimensionality reduction. That is, the same $J_1 = J_{scales}$, $J_2 = J_{rates}$ and $J_3 = J_{frequencies}$ were used in MPCA and GTDA, while $k = J_1 J_2 J_3$ for LPNTF, NTF, and DNTF. The same value of parameter k is used in order to construct the random projection matrix. Since the low dimensional features obtained by the aforementioned multilinear dimensionality reduction algorithms are linearly combined for classification, SVMs with linear kernel are tested as alternatives to SRC.

In Figure 2, the classification accuracy achieved by the three different classifiers is plotted as a function of the portion of the total scatter retained, when various subspace analysis methods are applied to both GTZAN and ISMIR 2004 Genre datasets. On the GTZAN dataset the best classification accuracy (92.4%) was obtained when LPNTF extracts features, that are classified by SRC. In this case, $k = 135$, as shown in Figure 1(c). The standard deviation of the classification accuracy was estimated thanks to 10-fold cross-validation. At the best classification accuracy, its standard deviation was found to be 2%. The reported classification accuracy outperforms those listed in Table 1. The interval $\pm$ one standard deviation is overlaid in all plots for the various values of the portion of the total scatter retained.

On the ISMIR 2004 Genre dataset the best classification accuracy (94.38%) was obtained, when the NTF with $k = 135$ extracts the low dimensional features that are classified by SRC next. When the LPNTF with $k = 105$ extracts features that are classified by SRC next, the classification accuracy is found equal to 94.25%, that is very close to the best accuracy. Both accuracies outperform the previously reported ones, which are listed in Table 1.

It is seen that the classification accuracy obtained by LPNTF and SRC outperforms the accuracy obtained with features extracted by all other multilinear subspace analysis techniques, which are next classified by either SRC or linear SVMs, for all the values of the portion of the total scatter retained but one. Moreover, the classification accuracy obtained with features extracted by LPNTF, NTF, MPCA or GTDA that are subsequently classified by SRC exceeds 80% for both datasets despite the reduced dimensions of the feature space extracted that are plotted in Figure 1(c) and (d). The experimental results reported in this paper indicate that the dimensionality reduction is crucial, when SRC is applied to music genre classification. This was not the case for face recognition [19].

6. CONCLUSIONS

In this paper, a robust music genre classification framework has been proposed. This framework resorts to cortical representations for music representation, while sparse representation-based classification has been employed for genre classification. A multilinear subspace analysis technique (i.e. LPNTF) has been developed, which incorporates the underlying geometrical structure of the cortical representations with respect to the music genre into the NTF. The crucial role of feature extraction and dimensionality reduction for music genre classification has been demonstrated. The best classification accuracies reported in this paper outperform any accuracy ever obtained by state of the art music genre classification algorithms applied to both GTZAN and ISMIR2004 Genre datasets.

In many real applications, both commercial and private, the number of available audio recordings per genre is limited. Thus, it is desirable that the music genre classification algorithm performs well for such small sets. Future

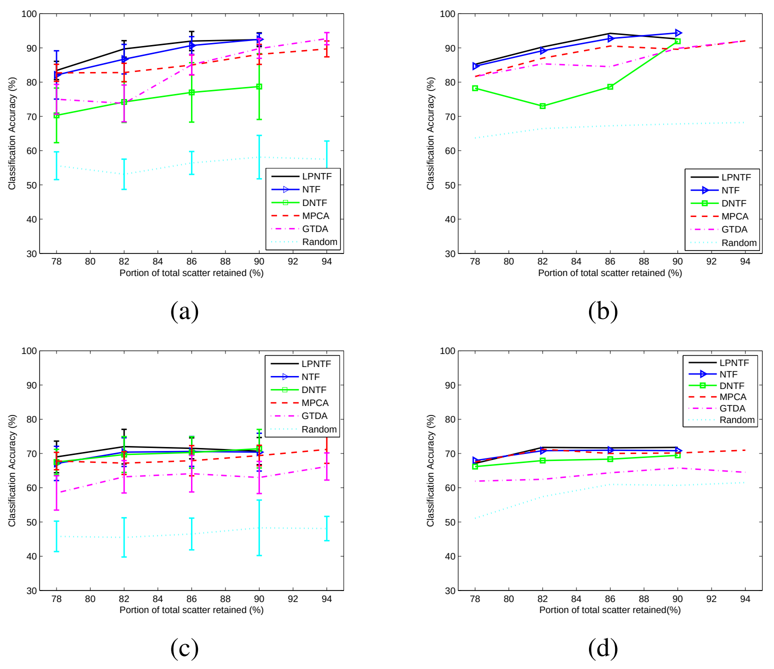


Figure 2. Classification accuracy for various feature extraction methods and classifiers. (a) SRC on GTZAN dataset; (b) SRC on ISMIR2004 Genre dataset; (c) Linear SVM on GTZAN dataset; (d) Linear SVM on ISMIR2004 Genre dataset.

research will address the performance of SRC framework under such conditions.

7. REFERENCES

[1] J. J. Aucouturier and F. Pachet: “Representing Musical Genre: A State of the Art,” *Journal of New Music Research*, pp. 83–93, 2003.

[2] E. Benetos and C. Kotropoulos: “A Tensor-Based Approach for Automatic Music Genre Classification,” *Proceedings of the European Signal Processing Conference*, Lausanne, Switzerland, 2008.

[3] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kegl: “Aggregate Features and AdaBoost for Music Classification,” *Machine Learning*, Vol. 65, No. 2-3, pp. 473–484, 2006.

[4] E. J. Candès, J. Romberg, and T. Tao: “Robust Uncertainty Principles: Exact Signal Reconstruction From Highly Incomplete Frequency Information,” *IEEE Transactions on Information Theory*, Vol. 52, No. 2, pp. 489–509, 2006.

[5] S. S. Chen, D. L. Donoho, and M. A. Saunders: “Atomic Decomposition by Basis Pursuit,” *SIAM Journal Scientific Computing*, Vol. 20, No.1 pp. 33–61, 1998.

[6] D. L. Donoho, “Compressed sensing,” *IEEE Transactions on Information Theory*, Vol. 52, No. 4, pp. 1289–1306, 2006.

[7] X. He and P. Niyogi: “Locality Preserving Projections,” *Advances in Neural Information Processing Systems*, Vol. 16, MIT Press, 2004.

[8] A. Holzapfel and Y. Stylianou: “Musical Genre Classification Using Nonnegative Matrix Factorization-Based Features,” *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16, No. 2, pp. 424–434, 2008.

[9] T. Kolda and B. W. Bader: “Tensor Decompositions and Applications,” *SIAM Review*, Vol. 51, No. 3, to appear.

[10] D. D. Lee and H. S. Seung: “Algorithms for Non-negative Matrix Factorization,” *Advances in Neural Information Processing Systems*, Vol. 13, pp. 556–562, 2001.

[11] C. -J. Lin: “On the Convergence of Multiplicative Update Algorithms for Nonnegative Matrix Factorization,” *IEEE Transactions on Neural Networks*, Vol. 18, No. 6, pp. 1589–1596, 2007.

[12] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos: “MPCA: Multilinear Principal Component Analysis of Tensor Objects,” *IEEE Transactions on Neural Networks*, Vol. 19, No. 1, pp 18–39, 2008.

[13] N. Mesgarani, M. Slaney, and S. A. Shamma: “Discrimination of Speech from Nonspeech Based on Multiscale Spectro-temporal Modulations,” *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 14, No. 3, pp. 920–930, 2006.

[14] E. Pampalk, A. Flexer, and G. Widmer: “Improvements of Audio-based Music Similarity and Genre Classification,” *Proceedings of the Sixth International Symposium on Music Information Retrieval*, London, UK, 2005.

[15] I. Panagakis, E. Benetos, and C. Kotropoulos: “Music Genre Classification: A Multilinear Approach,” *Proceedings of the Seventh International Symposium on Music Information Retrieval*, Philadelphia, USA, 2008.

[16] G. Tzanetakis and P. Cook: “Musical Genre Classification of Audio Signal,” *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 3, pp. 293–302, 2002.

[17] K. Wang and S. A. Shamma: “Spectral Shape Analysis in the Central Auditory System,” *IEEE Transactions on Speech and Audio Processing*, Vol. 3, No. 5, pp. 382–396, 1995.

[18] S. Woolley, T. Fremouw, A. Hsu, and F. Theunissen: “Tuning for Spectro-temporal Modulations as a Mechanism for Auditory Discrimination of Natural Sounds,” *Nature Neuroscience*, Vol. 8, pp. 1371–1379, 2005.

[19] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma “Robust Face Recognition via Sparse Representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 2, pp. 210–227, 2009.

[20] D. Tao, X. Li, X. Wu, and S. J. Maybank: “General Tensor Discriminant Analysis and Gabor Features for Gait Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 10, pp. 1700–1715, 2007.

[21] S. Zafeiriou: “Discriminant Nonnegative Tensor Factorization Algorithms,” *IEEE Transactions on Neural Networks*, Vol. 20, No. 2, pp. 217–235, 2009.

A QUANTITATIVE EVALUATION OF A TWO STAGE RETRIEVAL APPROACH FOR A MELODIC QUERY BY EXAMPLE SYSTEM

Justin Salamon
Music Technology Group
Universitat Pompeu Fabra, Barcelona, Spain
justin.salamon@upf.edu

Martin Rohrmeier
Centre for Music & Science, Faculty of Music,
University of Cambridge, United Kingdom
mrohrmeier@cantab.net

ABSTRACT

We present a two-stage approach for retrieval in a melodic Query by Example system inspired by the BLAST algorithm used in bioinformatics for DNA matching. The first stage involves an indexing method using n-grams and reduces the number of targets to consider in the second stage. In the second stage we use a matching algorithm based on local alignment with modified cost functions which take into account musical considerations.

We evaluate our system using queries made by real users utilising both short-term and long-term memory, and present a detailed study of the system's parameters and how they affect retrieval performance and efficiency. We show that whilst similar approaches were shown to be unsuccessful for Query by Humming (where singing and transcription errors result in queries with higher error rates), in the case of our system the approach is successful in reducing the database size without decreasing retrieval performance.

1. INTRODUCTION

The transition to digital media and the growing popularity of portable media devices over the past decade has resulted in much research into new ways of organising and searching for music. Of note are Content Based Music Retrieval (CBMR) systems [21] which search the musical content directly as opposed to using song meta-data for retrieval.

A specific case of CBMR is that of performing a melodic search in a collection of music, where the input query can be made either symbolically (e.g. text, score, MIDI controller) [8, 9, 19, 22] or by the user singing/humming the query, called Query by Humming (QBH) [4, 15]. For convenience we will refer to the symbolic input case as Query by Symbolic Example (QBSE). Both QBSE and QBH rely on an underlying model of melodic similarity [6]. In [17], a detailed review of algorithms for computing symbolic melodic similarity is provided. In recent years QBH systems have become increasingly popular, as they do not require musical knowledge such as playing an instrument or understanding musical notation. On the other hand, QBSE can be advantageous over QBH in certain cases – firstly, it affords more elaborate query specification, which might be preferred by advanced users and music researchers. Secondly, it does not require the automatic transcription of audio queries, which introduces additional errors into the queries.

In [4] a detailed comparative evaluation of different algorithms for QBH was carried out, comparing approaches based on note intervals, n-grams, melodic contour, HMMs and the Qubyhum system. The authors noted that the most successful approaches lacked a fast indexing algorithm, which is necessary in order to apply them to large databases. They studied a potential solution to the problem using a two-stage approach – an n-gram algorithm is used as a first stage for filtering targets in the database. The remaining targets are then passed to the second stage which uses a slower note interval matching algorithm which has better retrieval performance. The authors concluded that the approach was unsuccessful, as any significant improvement in search time resulted in a dramatic degradation in retrieval performance. They attributed this degradation to the errors introduced into the queries due to singing and transcription errors, which inhibited successful exact matching in the n-gram stage.

Nonetheless, other approaches for searching symbolic data exist for which efficient indexing is possible. Set-based methods (which also support polyphonic queries) have been shown to be effective for both matching and indexing – Clausen et al. use inverted files [3], Romming and Selfridge-Field use geometric hashing [16], Lemström et al. use index based filters [10] and Typke et al. use vantage objects [18]. For string based approaches (such as ours) many efficient indexing algorithms exist for metric spaces [2]. However, due to the melodic similarity measure used in our system (section 2.1.3), we can not use these algorithms and require an alternative solution (a recent solution to indexing non-metrics is also proposed in [20]).

In the following sections we present a two-stage indexing and matching approach for a QBSE system, inspired by the BLAST algorithm used in bioinformatics for DNA matching [5]. The first stage involves an indexing method (section 2.1.2) similar to the n-gram approach studied and evaluated in [4]. As our system avoids the need to transcribe user queries, the degree of errors in the queries depends only on the user, and is lower as a result. Consequently, it allows us to considerably reduce the database size for the second stage without degrading retrieval performance. In section 2.1.3 we present the second stage in which we perform matching using local alignment. We then detail the evaluation methodology used to evaluate our system, using real user queries utilising both short-term and long-term memory. Finally in the results section we show that this two-stage approach can be applied successfully in the case of QBSE, and study how the parameters of the indexing and matching algorithms affect retrieval performance and efficiency.

2. THE SONGSEER QBSE SYSTEM

2.1 System overview

SongSeer is a complete query by symbolic example system. The user interacts with the system through a graphical user interface implemented as a Java applet, allowing access from any web-browser [1] . The interface is further described in section 2.2. User queries are sent to a server application which contains a database of songs and performs the matching and returns the results to the client applet.

2.1.1 Query and target representation

The internal representation of user queries and database targets is based on the one proposed in [15] – pitch is represented as pitch intervals and rhythm as LogIOI Ratios (LogIOIR) [14]. This representation is independent of key and tempo, as well as concise, allowing us to match queries against targets even if they are played in another key or at a different tempo.

The targets are extracted from polyphonic MIDI files – every track in the MIDI file results in a single monophonic target. Tracks that are too short, as well as the drum track which is easily detectable are filtered out, but otherwise all tracks are considered. This allows the user to search for melodic lines other than the melody, though at the cost of considerably increasing the database size and adding targets which could possibly interfere with the search.

2.1.2 Indexing

The first stage in our two-stage approach is the indexing algorithm. Indexing is required in order to avoid comparing the query against every target in the database, thus improving system scalability an efficiency. As previously noted, our melodic matching is non-metric meaning we can not use existing indexing approaches, leading us to propose an alternative solution based on the BLAST algorithm.

The BLAST algorithm [5] was designed for efficiently searching for DNA and protein sequences in large databases. The steps of the original BLAST algorithm are the following: first, low-complexity regions or sequence repeats are removed from the query. Then, the query is cut into "seeds" – smaller subsequences of length n which are evaluated for an exact match against all words (of the same length) in the database using a scoring matrix. High scoring words (above a threshold T) are collected and the database is then scanned for exact matches with these words. The exact matches are then extended into high-scoring pairs (HSP) by extending the alignment on both sides of a hit till the score starts to decrease (in a later version gaps in the alignment are allowed). HSPs with scores above a cutoff score S are kept, and their score is checked for statistical significance. Significant HSPs are locally aligned, and an expectation value E is calculated for the alignment score. Matches with E smaller than a set threshold are reported as the final output.

Our indexing algorithm is based on this concept of preceding the slower local alignment stage with a fast exact matching stage. Given a query, we cut it into "seeds" as in BLAST, and search for exact matches in the database. This can be efficiently implemented by storing the targets of the database in a hash table where every key is a seed which hashes to a collection of all targets containing that seed. We also implement the idea of filtering less informative parts of the query as detailed in section 4.5. This first stage allows us to return a much reduced set of targets, which we then compare to the query using our matching algorithm. A crucial parameter of this approach is the seed size n – a longer seed will return less targets making the retrieval faster, but requires a longer exact match between query and target potentially reducing performance for queries which contain errors.

2.1.3 Matching

For determining the similarity of a query to a target, we use the dynamic programming approach for local alignment [13], similar to the one proposed in [15] with one significant difference – in an attempt to make the matching procedure more musically meaningful, we replace the *skip* and *replace* costs in the local alignment algorithm with *cost functions*. These functions determine the skip and replace cost based on the specific pitch intervals and LogIOIRs being compared. The underlying assumption is that some errors should be penalised less heavily than others, based on the errors we can expect users to make when making a query:

- Repeated notes – the user might repeat a note more or less times than in the original melody (for example when translating a sung melody into piano strokes). Thus, the penalty for skipping a repeated note should be reduced.
- Pitch contour – the user might not remember the exact pitch interval, but remember the pitch contour correctly (big/small jump up/down or same). Thus, the penalty for replacing an incorrect interval which has the correct contour should be reduced.
- Rhythm contour – the user might not remember the exact rhythm ratio between two notes, but remember the "rhythmic contour" correctly (slower, faster or same). Thus, the penalty for replacing an incorrect LogIOIR which has the correct contour should be reduced.
- Octave errors – the user might play the correct pitch class but in the wrong octave relative to the previous

[1] `http://www.online-experiments.com/SongSeer`

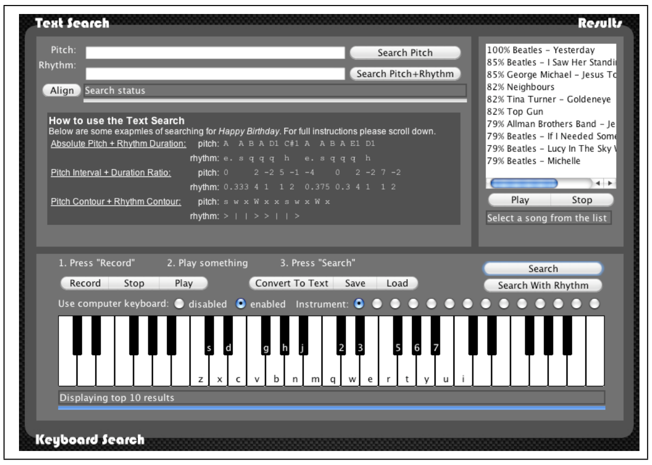
note. Thus, the penalty for replacing a note which is off by an integer number of octaves should be reduced.

Following this rationale, we define two cost parameters – a *full cost* and a *reduced cost*. When one of the aforementioned cases is detected the cost functions return the reduced cost, and in all other cases the full cost. Another issue is the relative importance we give to the pitch and rhythm match scores. As the pitch and rhythm of a query might be represented with different degrees of accuracy, the match scores should be weighted differently when combining them to obtain the final match score. To do this we introduce a *pitch factor* and *rhythm factor* which weight the pitch and rhythm scores when combining them.

By default, the *full cost* is set to 2 and the *reduced cost* to 1, and both pitch and rhythm factors are set to 1 (so that the pitch and rhythm scores are weighted equally and summed into the final score). In the results section we explain how these parameters are optimised based on real user queries.

2.2 The SongSeer GUI

The SongSeer GUI is displayed in Figure 1. Two query input methods are provided – a text search allowing to make textual queries similar to the ones supported by the Themefinder [8] system by Huron et al. (including pitch and rhythm contour), and a virtual keyboard that can be played using the mouse, the computer keyboard or a connected MIDI controller. Once a query is made the top 10 results are displayed back to the user with a percentage indicating the matching degree, and the user can select a song and play back the corresponding MIDI file stored in the database.



How to use the Text Search

Below are some exapmles of searching for *Happy Birthday*. For full instructions please scroll down.

```
Absolute Pitch + Rhythm Duration:   pitch: A  A B A D1 C#1 A  A B A E1 D1
                                   rhythm: e. s q q q  h   e. s q q q  h
Pitch Interval + Duration Ratio:    pitch: 0     2 -2 5 -1 -4    0   2 -2 7 -2
                                   rhythm: 0.333 4 1  1 2  0.375 0.3 4 1  1 2
Pitch Contour + Rhythm Contour:     pitch: s w x W x x s w x W x
                                   rhythm: > | | > > | | >
```

100% Beatles – Yesterday
85% Beatles – I Saw Her Standi
85% George Michael – Jesus To
82% Neighbours
82% Tina Turner – Goldeneye
82% Top Gun
79% Allman Brothers Band – Je
79% Beatles – If I Needed Some
79% Beatles – Lucy In The Sky V
79% Beatles – Michelle

Figure 1. The SongSeer user interface.

3. EVALUATION METHODOLOGY

3.1 Test collections and machines used

For the evaluation, we compiled a corpus of 1,076 polyphonic MIDI files of pop and rock music, including 200 songs by the Beatles. After ignoring short tracks and drum tracks, this translates into 6,541 targets in the database. For scalability tests we have also compiled several more corpora of increasing size, the largest containing 18,017 songs which translates into 88,034 targets.

All user experiments were run on standard pentium IV PCs running windows XP. The quantitative evaluation was run on a server machine with two Intel® Dual Core Xeon® 5130 @ 2GHz with 4MB Cache and 4GB RAM, running Linux 2.6.17-10 and Java HotSpot™ 64-Bit Server VM.

3.2 Collecting user queries

We conducted a user experiment with 13 participants of varying musical experience, ranging from amateur guitar players to music graduates. The first part of the experiment involved a usability test in which the subjects were asked to complete a set of tasks using the SongSeer interface, which also allowed them to familiarise themselves with the system. The second part involved the subjects making queries which would then be used to perform a quantitative evaluation of the system. For the purpose of the quantitative evaluation, all subjects were asked to play on the virtual keyboard, using either the mouse or computer keyboard.

To collect queries, subjects were presented with a list of 200 Beatles songs, and asked to record queries of songs they can remember from the list. This stage simulates the event where a user remembers part of a song they have not heard recently, and resulted in 63 "long term memory" queries. Next, subjects were asked to listen to 10 audio recordings of Beatles songs and then record a query, simulating the event where the user has recently heard the song, resulting in 123 "short term memory" queries. This gives us a total of 186 real user queries for system evaluation.

3.3 Evaluation metrics

As there is always only a single correct result (the database contained no cover versions), we use a metric based on the rank of the correct song based on match score, the *Mean Reciprocal Rank* (MRR) which is given by:

$$MRR = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{rank_i} \tag{1}$$

where K is the numbers of queries evaluated and $rank_i$ is the rank of the correct song based on the match score for query i. This is similar to taking the average rank of the correct song over all queries but is less sensitive to poor ranking outliers, and returns a value between $1/M$ and 1 (where M is the number of songs in the database) with higher values indicating better retrieval performance.

It is important to note however that as it is possible for several songs to have the same match score, they may share the same rank (in which case they are returned by alphabetical order in the results list). In order to evaluate performance from a user perspective (where the position of a song in the result list is significant), we introduce a second metric – the *ordered MRR* (oMRR) which is computed in the same way as the MRR but where the rank is based not on the match score but on the actual position of the song in the final results list.

4. RESULTS

4.1 Initial results

In order to asses the performance of our approach, we start by estimating a baseline performance for the problem. The baseline is estimated using the following procedure: Given a query, we randomly generate the rank of the correct song in the result list (between 1 and 1076). We repeat this process 99 times for the same query, saving the best randomly generated rank out of the 99 repetitions. We perform this procedure for all 186 queries, and use the saved ranks to compute an overall oMRR. This gives us an oMRR of 0.211 (with a variance of 0.051).

Next we turn to evaluate our algorithm. As a first step, we compute the MRR and oMRR taking only the pitch information into account, using a seed size $n = 3$ and the matching parameters set to their default values (full cost = 2, reduced cost = 1). The results are presented in Table 1.

Query Group	#Queries	MRR	oMRR
All queries	186	0.800	0.659
Long term memory	63	0.765	0.627
Short term memory	123	0.818	0.675

Table 1. Initial results.

The table shows that our oMRR values are significantly ($P < 10^{-10}$, Wilcoxon rank-sum test) higher than the baseline. Though results for the short term memory queries are slightly higher than for the long term memory queries, the difference is not statistically significant, and for the rest of the evaluation we use all the queries together. Finally, we observe that, as expected, the MRR values are higher than the oMRR values. This suggests that songs are not sufficiently distinguished using the default parameters.

4.2 The effect of rhythm on performance

We now include the rhythm information in the matching procedure, setting the pitch and rhythm factors to 1:1. The MRR and oMRR go down from 0.800 and 0.659 (pitch only) to 0.677 and 0.594 (pitch+rhythm) respectively. This indicates that giving rhythm equal importance as pitch degrades performance. We could argue that as the rhythm information is less detailed compared to the pitch information, it will match a greater set of songs, so when given equal importance as the pitch information it ends up degrading the results. Another possibility is that due to the use of a virtual keyboard rather than a real one users found it harder to accurately play the rhythm of a query.

4.3 Choice of seed size

As previously mentioned, in [4] it was shown that a two stage retrieval process for QBH was unsuccessful in reducing search time without considerably degrading retrieval performance. In Figure 2 we evaluate the effect of the seed size n in our indexing algorithm (the first stage of our two-stage approach) on retrieval performance and search time.

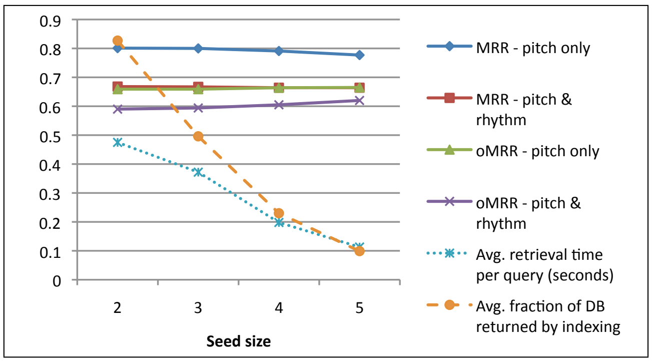


Figure 2. MRR, oMRR, retrieval time and DB reduction vs seed size.

Figure 2 shows that the number of targets to search returned by the indexing algorithm is almost halved every time we increase the seed size, and consequently the search time goes down. Interestingly, the MRR and oMRR values remain stable as we increase the seed size (only the MRR for pitch only shows slight signs of decline). Increasing the seed size n does however require the user query to contain at least n sequential correct pitch intervals, in addition to increasing memory and storage requirements for the database. All in all it is a trade-off between retrieval performance, efficiency and resource usage. For the rest of the evaluation we choose a seed size of 4, providing a significant reduction in database size (reduced to 23%) with practically no degradation of retrieval performance.

4.4 Parameter optimisation

In section 2.1.3 we introduced the notion of having a *full cost* and a *reduced cost* in the matching algorithm, and in section 4.2 we saw that giving rhythm equal importance as pitch in the matching is detrimental to retrieval performance. In this section we optimise these parameters, namely the ratio between the full and reduced costs, and the ratio between the pitch and rhythm factors.

To do so we divided the queries into two groups of roughly equal size – the optimisation is performed using the queries of group 1, and then validated on the queries of group 2. For the optimisation we use the *Simulated Annealing* approach for global optimisation [7]. The performance for the two query groups before optimisation is given in Table 2.

Query Group	Pitch:Rhythm Ratio	Full:Reduced Cost	MRR	oMRR
1	1:1	2:1	0.704	0.646
2	1:1	2:1	0.634	0.574

Table 2. Results for the groups before optimisation.

We start by optimising the pitch and rhythm weighting factors. The effect of these parameters on performance is visualised in Figure 3. The optimal pitch to rhythm ratio was found to be 3:1 (Table 3), and is used in all fur-

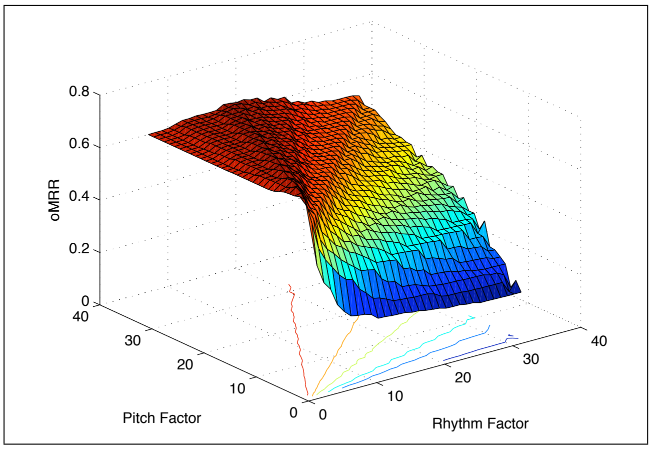


Figure 3. oMRR as a function of the pitch and rhythm factors.

ther evaluations. A slightly higher MRR value could be achieved by ignoring rhythm altogether, but at the cost of significantly reducing the oMRR indicating that the rhythm information is useful for distinguishing between targets which have the same pitch match score.

We next perform the optimisation for the cost parameters in the matching algorithm. The optimal values were found to be 3 for *full cost* and 1 for *reduced cost* (Table 3 last row). This suggests that the modified cost functions provide an improvement to performance, as otherwise the optimal full and reduced costs would have been equal to each other (Table 3 penultimate row).

Pitch:Rhythm Ratio	Full:Reduced Cost	MRR	oMRR
1:1	2:1	0.704	0.646
3:1	2:1	0.784	0.745
3:1	1:1	0.759	0.717
3:1	3:1	0.787	0.761

Table 3. Results for group 1 before and after pitch:rhythm optimisation and full:reduced optimisation.

Finally we compute the MRR and oMRR for query group 2 and for all queries together using the optimised parameters. Figure 4 shows that in all cases performance is improved, though only in the case of oMRR for all queries (Group 1&2) is the improvement statistically significant (p=0.018, Wilcoxon rank-sum test).

4.5 Seed filtering

Next we examine the distribution of seeds in the queries and the database, displayed in Figure 5.

Both distributions constitute a power law probability distribution, obeying a kind of Zipf's law for musical interval sequences [23]. Accordingly, the most frequent seeds comprise the largest proportion of the database but convey the least amount of information useful for distinguishing between songs. By filtering from the query the seeds which are most common in the database we can further reduce the

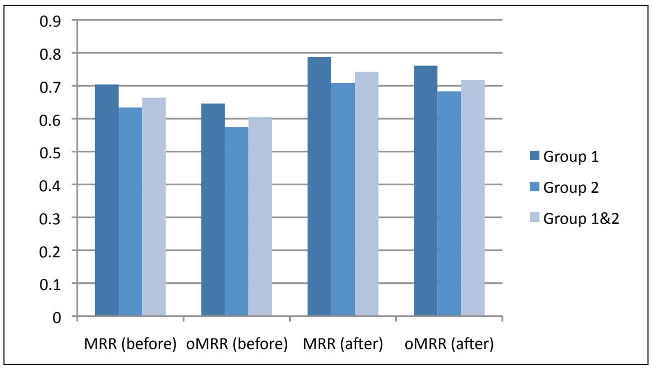


Figure 4. MRR and oMRR results, before and after optimisation.

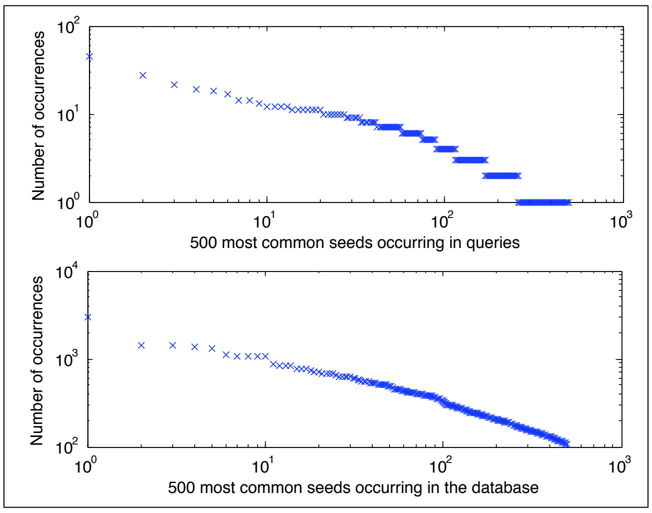


Figure 5. Seed distributions for queries and songs in the database.

fraction of the database returned by the indexing algorithm while maintaining retrieval performance.

When filtering just the three most common seeds in the database, we reduce the database size by a further 40% (from 23% to 14% of the original size) while the MRR and oMRR values go down by less than 1.3%. This concept could be further extended by introducing a seed weighting function, for example using $tf * idf$ [1] like weighting based on seed distributions. This could also help in ranking songs which have the same match score after the second stage, however we have not explored this option and leave it for future work.

4.6 Scalability

Finally, we evaluate how retrieval performance and efficiency are affected as we scale the database size up to 18,017 songs which translates into 88,034 targets. The results are presented in Figures 6 and 7.

First we note that our indexing algorithm provides a considerable reduction in the fraction of the database returned by the first stage, reducing it to 15% of its original size. Nonetheless, further work would be required for our approach to be applicable to collections of millions of

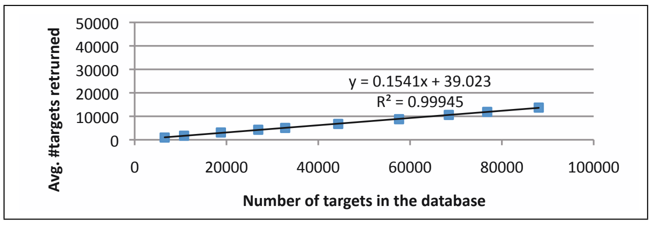


Figure 6. Avg. #targets returned by the first stage vs #targets in the database.

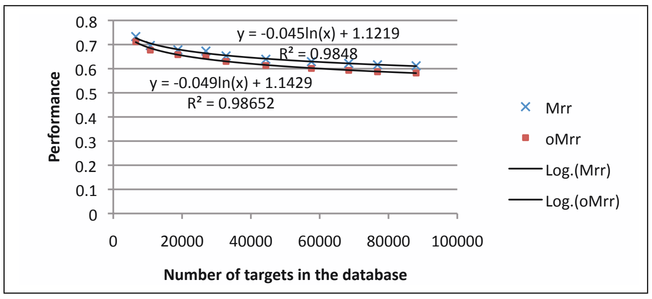


Figure 7. MRR and oMRR vs numbers of targets in the database.

songs, ideally obtaining a sublinear relation between the number of targets in the database and the number of targets returned by the indexing stage.

Next we note that both the MRR and oMRR decrease as $O(log(n))$ as the database size n is increased ($R^2 > 0.98$), indicating that performance scales well with database size.

5. CONCLUSIONS

In this paper we introduced a two-stage retrieval approach for a melodic QBSE system. We demonstrated that whilst for QBH systems similar approaches were unsuccessful, for QBSE this approach can successfully reduce the database size while maintaining high MRR values. We provided a detailed study of the effect of different parameters of the system, namely the seed size, the relative weighting of pitch and rhythm and the full and reduced costs in the matching algorithm.

Finally we consider some ideas for future work. Instead of considering almost every track in a MIDI file as a target, we could aim to extract only the most relevant melodic parts of the piece, as done in [11, 12]. Next, it would be interesting to further study the seed distributions in the queries and the database, which could help develop more elaborate seed filtering and/or a seed weighting scheme. [2]

6. REFERENCES

[1] R. Baeza-Yates and B. Riberto-Neto. *Modern Information Retrieval*. Addison Wesley and ACM Press, 1999.

[2] T. Bozkaya and M. Ozsoyoglu. Indexing Large Metric Spaces for Similarity Search Queries. *ACM Transactions on Database Systems*, 24(3):361–404, September 1999.

[3] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. PROMS: A Web-based Tool for Searching in Polyphonic Music. In *ISMIR Conference Proceedings*, 2000.

[4] R. B. Dannenberg, W. P. Birmingham, B. Pardo, N. Hu, C. Meek, and G. Tzanetakis. A Comparative Evaluation of Search Techniques for Query-by-Humming Using the MUSART Testbed. *Journal of the American Society for Information Science and Technology*, February 2007.

[5] W.J. Ewens and G. Grant. *Statistical Methods in Bioinformatics: An Introduction (Statistics for Biology and Health)*. Springer, 2nd edition, 2005.

[6] W. Hewlett and E. Selfridge-Field, editors. *Melodic Similarity: Concepts, Procedures and Applications*. MIT Press, Cambridge, 1998.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598), 1983.

[8] A. Kornstädt. Themefinder: A Web-based Melodic Search Tool. *Computing in Musicology*, 11:231–236, 1998.

[9] K. Lemström, V. Mäkinen, A. Pienimäki, M. Turkia, and E. Ukkonen. The C-BRAHMS Project. In *ISMIR Conference Proceedings*, pages 237–238, 2003.

[10] K. Lemström, N. Mikkilä, and V. Mäkinen. Fast Index Based Filters for Music Retrieval. In *ISMIR Conference Proceedings*, pages 677–682, 2008.

[11] S. T. Madsen, R. Typke, and G. Widmer. Automatic Reduction of MIDI Files Preserving Relevant Musical Content. In *6th International Workshop on Adaptive Multimedia Retrieval (AMR 2008)*, Berlin, Germany, 26-27 June 2008.

[12] C. Meek and W.P. Birmingham. Automatic Thematic Extractor. *Journal of Intelligent Information Systems*, 2003.

[13] G. Navarro and M. Raffinot. *Flexible Pattern Matching in Strings*. Cambridge University Press, Cambridge, UK, 2002.

[14] B. Pardo and W.P. Birmingham. Encoding Timing Information for Musical Query Matching. In *ISMIR Conference Proceedings*, Paris, France, 2002.

[15] B. Pardo, J. Shifrin, and W. Birmingham. Name That Tune: A Pilot Study in Finding a Melody From a Sung Query. *Journal of the American Society for Information Science and Technology*, 2004.

[16] C. A. Romming and E. Selfridge-Field. Algorithms for polyphonic music retrieval: the Hausdorff metric and geometric hashing. In *ISMIR Conference Proceedings*, pages 457–462, Vienna, Austria, 2007.

[17] R. Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Utrecht University, Netherlands, 2007.

[18] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R. Van Oostrum. Using Transportation Distances for Measuring Melodic Similarity. In *ISMIR Conference Proceedings*, pages 107–114, 2003.

[19] R. Typke, R. C. Veltkamp, and Wiering F. Searching Notated Polyphonic Music using Transportation Distances. In *Proceedings of the ACM Multimedia Conference*, pages 128–135, New York, 2004.

[20] R. Typke and A. Walczak-Typke. A Tunneling-Vantage Indexing Method for Non-Metrics. In *ISMIR Conference Proceedings*, pages 683–688, 2008.

[21] R. Typke, F. Wiering, and R. C. Veltkamp. A Survey of Music Information Retrieval Systems. In *ISMIR Conference Proceedings*, pages 153–160, 2005.

[22] A. L. Uitdenbogerd. N-gram Pattern Matching and Dynamic Programming for Symbolic Melody Search. In *Proceedings of the Third Annual Music Information Retrieval Evaluation eXchange*, Sept. 2007.

[23] D. H. Zanette. Zipf's Law and the Creation of Musical Context. *Musicae Scientiae*, 10(1):3–18, 2006.

[2] This work was supported in part by Microsoft Research through the European PhD Scholarship Programme, and the Programa de Formación del Profesorado Universitario (FPU) of the Ministerio de Educación de España.

A WEB-BASED APPROACH TO DETERMINE THE ORIGIN OF AN ARTIST

Sten Govaerts **Erik Duval**
K.U. Leuven
Department of Computer Science
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
{sten.govaerts, erik.duval}@cs.kuleuven.be

ABSTRACT

One can define the origin of an artist as the geographical location where he started his career. The origin is an important metadata element, because it can help to specify subgenres, be an indicator of regional popularity and improve recommendations. In this paper, we present six methods to determine the origin, based on Web data sources: one extracts data from Last.fm, two query Freebase and three analyze biographies. We evaluate the different methods with 11275 artists. Circa 55% of the artists can be classified using biographies. The best Freebase method can classify 26% and the Last.fm based method 7%. When comparing on accuracy, the Last.fm and Freebase methods perform similarly with around 90% accuracy. For the biography-based methods we achieve 71%. To improve coverage, a final, hybrid method achieves 77% accuracy and 60% coverage. The accuracy of the continent classification is 87%. As a showcase for our classifier, we developed a mashup application that displays, among others, information about the origin of artists from radio station playlists on a map.

1. INTRODUCTION

In a previous project [1], we developed a music player for hotels, restaurants and pubs. Peculiar to our approach is that a user can describe the music he wants by referring to a situation, rather than by defining the usual search criteria on artist, title, etc. This system uses almost 40 metadata fields, manually annotated by music experts of Aristo Music (http://www.aristomusic.com), which is a very time-consuming and expensive labor. Currently, the Aristo Music database contains around 58000 songs. The time-consuming metadata annotation process is difficult to scale: in the case of Aristo Music for instance, it limits the ability to penetrate new markets. In order to assist the experts, we already achieved some success in automating the annotation process for some metadata fields [2].

This paper reports on ongoing work in the MuziK project that focuses on automatically generating the metadata fields that are most costly to do manually and most relevant for end users. For this purpose, we rely on a variety of approaches: digital signal processing [3], web-based classification [4] and data analysis [5].

This paper focuses on one of the parameters: the origin of an artist, defined here as the geographical location where an artist started his musical career. This can be hard to determine sometimes. It can be seen as the country of the artist's first success, where he lived most of his life or where some of the group members live. For example Georg Friedrich Händel was born in Germany, but went to England where his career really took off.

1.1 Relevance

Although the origin of an artist can thus be quite fuzzy, it is a useful piece of metadata in many cases.

- Some subgenres are based on geographical location of the artist, for example Britpop and Viking Metal.
- It can also be a good indicator of the popularity of an artist in a region, as most artists are often most popular in their country of origin. An artist popularity visualization based on Last.fm data shows this by comparing two countries, http://hublog.hubmed.org/archives/001085.html. There are of course exceptions with an international carreer, like the Spice Girls.
- Recommendations can also improve by using the artist's origin. They can be tuned to the location of the listener. Some musically very similar songs can be good or bad recommendations, depending on the region (and dialect): for example, a small stage art genre in Belgium and the Netherlands (called "Kleinkunst") is mostly expressed in regional dialects and if the recommendation for a song from Antwerp is a song originating from Amsterdam it could break the atmosphere, although they are musically very similar.

The data in Figure 1, collected during an internal time management evaluation, shows the average time in seconds needed by a music expert to annotate different metadata fields. Origin is the 6th most expensive element, and rather close to the top. Providing the origin often requires manual lookup work, hence making it very expensive to annotate. Annotating continent takes much less because it is derived from origin. Target region is used for localized music distribution and is done in batch, making it faster. One can determine the origin in a more or less precise way. Sometimes, the country is not specific enough due to differences in regional music styles or linguistic,

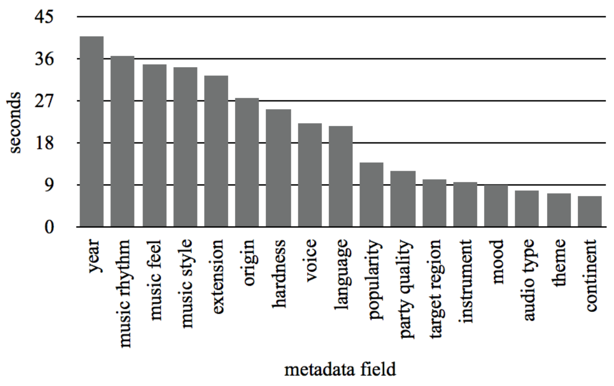


Figure 1. The average number of seconds to manually annotate a metadata field.

linguistic, cultural or religious differences within a country, as in for instance East coast vs. West coast rap. We rely on two levels: country and continent.

1.2 Related work

Today, specialized search engines enable searching for persons, which often requires linking different sources to a person. Researchers are using knowledge management systems to find experts [6]. Numerous examples can be found online: NNDB (http://www.nndb.com/), Pipl (http://pipl.com/), and Spock (http://www.spock.com/). They often offer information on birthplace and residence and some even recognize music artists. Except for Celma's demo [7], we did not find any related work specialized towards classifying the origin of music artists.

Origin is annotated by the experts based on either their personal knowledge of the artist or by looking up the origin information. Therefore, we try to automate this metadata field by extracting information from web data sources. In this paper, we present our approach to classify an artist's geographical origin. First, the web data sources and algorithms will be explained and evaluated in section 3. Afterwards, we discuss the concept and design of our mashup tool, which locates artists of radio station playlists as a demonstration of the technique, and present the conclusion and possible further work.

2. FINDING THE ORIGIN

Determining the origin of the artist will be very hard by using content-based techniques (e.g. by analyzing the signal). The best option is to analyze other data sources. In our quest to classify the origin we looked at a wide plethora of music related web resources. Most of these resources are in plain text, for example sites with reviews like Amazon.com, which makes it hard to extract geographical locations, but results can be achieved with for example named entity recognition [8]. One of the main problems is the use of the names of the inhabitants or adjectives of geographical locations, e.g. German vs. Germany. These kinds of words are called demonyms and are not recognized as geographical descriptors by most named entity recognizers, e.g. Open Calais (http://www.opencalais.com/). Luckily, more structured data sources are available: for instance, most Wikipedia artist pages contain a box with background information. Moreover, there are some online databases available for querying, e.g. Freebase (http://www.freebase.com). Tags are also a rich source of information and often contain geographical data. If an artist is really more popular in his country of origin (not proven), listening counts of an artist per country might also be an interesting source.

Our approach relies on several methods with different data sources, because no single data source covers all artists. In the remainder of this section, we will present our methods to determine the origin of an artist. Section 2.1 covers a screen scraping technique based on Last.fm (http://last.fm). Section 2.2 presents two approaches that rely on Freebase, and section 2.3 details a method to analyze an artist biography with demonyms.

2.1 Origin determination with Last.fm

Last.fm is a well-known music recommender system and a music community website with over 30 million users, making it a great resource of metadata for MIR, such as biographies and tags [9]. For some artists, Last.fm contains the origin and sometimes their different whereabouts over time, for example Radiohead is located in Abingdon, Oxfordshire, UK since 1986, according to http://www.last.fm/music/Radiohead.

We scrape the Last.fm artist page with Dapper (http://www.dapper.net/open/), which basically creates a web service out of unstructured data. Sometimes the data on Last.fm is incomplete: e.g. New York, without a country. To fill in these gaps, we use the Google Maps API (http://maps.google.com) for geo-coding to retrieve the ISO 3166-1 country code, used to identify the origin.

2.2 Origin determination with Freebase

Freebase [10] is a large collaborative semantic database, containing structured data, harvested from different resources, e.g. Wikipedia.org and MusicBrainz.org. Freebase allows querying through a REST-ful web service, using ontologies to describe the semantics and data interlinking. Different classes (/music/group_membership, /music/artist, /music/musical_group/member) describe music artists and within these, others (place_of_birth, nationality, origin, places_lived) describe geo-locations. We run a long complex query covering all the artist and geo-location classes of Freebase, which is used in 2 methods:

- **Based on the freebase origin class (freebase-origin):** the origin class is geo-coded with the Google Maps API to get the country code, which is the result.
- **Most frequent location (freebase-most_freq):** takes the nationality, birthplace and places where the artist lived and geocodes them all. The most occurring country code in all the locations is selected. Then out of all the locations with this selected country code, the most occurring city is selected. The location of that

city and country code is the result of the method. To conclude, this method returns the most frequent location of nationalities, birthplaces and residences of the group members.

2.3 Origin determination with biographies

Biographies often provide a lot of geographical information and thus can be a great source of information on the origin. As mentioned before, authors often describe geolocations with demonyms, for example "Anders Trentemøller is a Danish electronic musician…" (from http://www.last.fm/music/Trentemøller). We looked for natural language stemmers to transform demonyms, but none were found. One might be able to extend a stemmer with rules to cover all exceptions. We use a list of countries and their demonyms from Wikipedia, which also contains Anglo-Saxon cities and the states of the USA (http://en.wikipedia.org/wiki/Nationality). We noticed that the origin or residence of an artist is often mentioned in the first sentences of the biography. We implemented 3 variations that exploit this characteristic. The biography is split into natural language sentences and for every sentence the occurring demonyms and locations are noted.

- **Highest occurrence (bio-most_freq):** The result is the demonym or geographical location that occurs most often.
- **Favor first occurrences (bio-favour_1st):** For every encountered country code (cc) a list of sentence numbers s in which cc occurred is kept. Say, s_{tot} is the total number of sentences in the biography, then following formula is calculated for every country code cc:
$$R_{cc} = \sum_{i=0}^{length(s)} \frac{(s_{tot}+1)-s_i}{s_{tot}+1}$$
The result is the country code with the highest R_{cc}.
- **Weaker favoring first occurrences (bio-weak_favour_1st):** this method is equal to the previous, but another weighting function is applied for every country code cc:
$$R_{cc} = \sum_{i=0}^{length(s)} \frac{(s_{tot}+2)-s_i}{s_{tot}+1}$$
Again, the result is R_{cc}. This method tries to spread importance a bit more over the sentences.

3. EVALUATION

The approaches from section 2 are evaluated against a ground truth data set provided by Aristo Music. First, we describe the data set normalization, then the results are discussed and a combination of all methods is presented.

3.1 The data set

As ground truth, we use the origin and continent field from the Aristo Music database. They group metadata per song, although the origin is an artist property. For some artists, different songs indicate different origins, due to errors in the database. In those cases, we consider the origin that occurs most often. From the data set, we removed 10 artists that are not annotated with origin, as well as artists with an origin value like "Mixed" and "Others": these are used when for instance a group of artists collaborate, e.g. "George Michael & Aretha Franklin". These artists are removed from the ground truth – we identify them through connectors like "and", "feat." and "vs.". The Caribbean contains the Caribbean Sea and its islands and cannot be mapped to a single country code. Artists annotated with “Caribbean” are thus also removed from the ground truth based. The overall result is that 25% are removed: we keep 11275 artists (Table 1).

To classify continents, the origin is mapped to a continent lookup table. There are different ways to define continents, based on geography or political treaties. We use a Wikipedia table (http://en.wikipedia.org/wiki/List_of_-countries_by_continent_(data_file)). Table 1 also shows the distribution of artists over continents. Our data set reflects mainstream music taste in Europe, with a strong representation of North America and Europe.

As input for the methods, the artist page and biography is retrieved for all artists with the Last.fm API. All origins are geocoded to obtain the country code.

3.2 The results

Section 3.2.1 examines how well all approaches cover the data set. Section 3.2.2 discusses the accuracy of the results per method. Finally, we present a new method that increases the coverage and improves overall performance.

3.2.1 Coverage

The coverage is defined as the number of artists of the ground truth that have been determined. Figure 2 shows the percentage of artists for which an origin was found per method. For 59%, an origin can be found by at least one of the methods. The large difference in coverage is the main reason why we use multiple data sources.

Only a small percentage of Last.fm artist pages (7%) contain the origin. Freebase covers a bit more than 26%. Surprisingly, only 5% less artists have an explicit origin class in the Freebase ontology. For the three biography-based methods, the coverage is obviously the same (56%) and about double that of Freebase. As only 63% of artists

	#	%
total nb. artists original data	14880	100
total nb. artists cleaned data	11275	75,77
nb. artists removed	3605	24,23
nb. artists in Europe	7167	63,57
nb. artists in N. America	3612	32,04
nb. artists in S. America	195	1,73
nb. artists in Asia	141	1,25
nb. artists in Africa	80	0,71
nb. artists in Oceania	80	0,71

Table 1. The number of artists in the data set and for each continent in the cleaned data set.

have a biography, this means that almost all of the retrieved biographies contain geographical information. For 62% of the artists, an origin could be retrieved with at least one of the methods. The coverage of the combination method will be discussed later.

As discussed, the best method can only annotate 56% of the artists. One of the reasons is that Aristo Music is a Belgian company with a database that includes many local artists for whom neither biographies nor freebase entries are available. There is also a rich tradition of mardi gras music in Belgium, made by artists well known in a very small region (sometimes only a village or town) for a short period of the year. Their online presence is zero. There are also possible differences in artist name writing between Aristo Music and Last.fm. The latter often offers alternative pages for the different writings; most of these pages contain no biography or a very condensed one, disabling the biography-based methods.

3.2.2 Accuracy

The combination method makes use of the results of the other methods. Therefore the same data set cannot be used to evaluate all methods. A data set, called combo data set, contains 3000 randomly selected artists from the full data set to evaluate the combination method. The 8275 artists left are used to evaluate the Last.fm, Freebase and biography methods and is called the LFB data set. The coverage of the LFB and combo data set is almost equal to Figure 2 (up to about 1% difference).

The accuracy is defined as the number of correct classifications divided by the total number of artists covered by a method. The accuracy of the Last.fm, Freebase and biography-based methods for the continent and origin classification on the LFB data set is shown in Table 2. As expected, the continent classification is performing better, because small errors in the origin will not impact on the continent classification: for instance, a misclassification of a French artist as a German one will still result in a correct continent (Europe). The Last.fm screen scraping approach has the highest accuracy overall and the Freebase methods perform equally well and are close second. Their 95% confidence intervals overlap, so the best performer cannot be concluded, nor from performed t-tests.

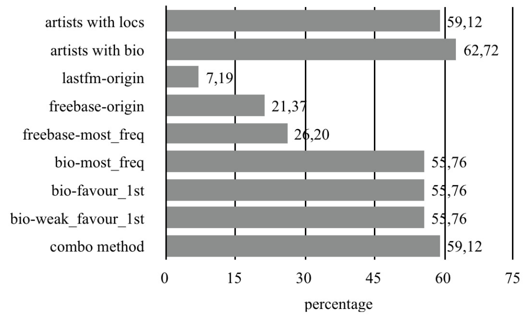


Figure 2. The percentage of annotated artists by the different methods.

methods	origin (%)	CI origin	cont. (%)	CI cont.
lastfm-origin	91,28	[89,06; 93,50]	95,32	[93,66; 96,98]
freebase-origin	90,60	[89,24; 91,96]	93,58	[92,44; 94,72]
freebase-most_freq	90,60	[89,37; 91,83]	94,36	[93,39; 95,33]
bio-most_freq	63,50	[62,11; 64,89]	77,21	[76,00; 78,42]
bio-favour_1st	70,95	[69,64; 72,26]	83,12	[82,04; 84,20]
bio-weak_favour_1st	66,16	[64,80; 67,52]	78,74	[77,56; 79,92]

Table 2. The accuracy and 95% confidence intervals for all methods for the classification of continent and origin on the LFB data set.

There is quite a drop in accuracy when using the biography as a source. Continent classification with biographies performs much better than origin. When comparing the results of the 3 different biography-based methods, it is clear that our assumption that the origin appears in the first sentences is valid, because the method favoring the first sentences most strongly, bio-favour_1st, is the best.

All methods have their issues. We noticed that some of them occur due to errors in geocoding of locations with the same name, e.g. Birmingham in USA and UK, or a mix-up between small geographical entities and their big neighboring countries, e.g. Jersey and England, Luxemburg and Belgium. Another problem occurs when different artists have the same name; in that case, they have to be identified on song level. Of course, there are also method dependent errors: for example, sometimes the Last.fm origin is given as a demonym instead of a country, e.g. “American” and “French”, and the geocoding will resolve this to another country, respectively to Americana, Brazil and Wattsburg, USA. The freebase-origin method often takes the country of birth as the origin of the artist, e.g. Akon was born in Senegal, but moved as a child to the USA. In the case of freebase-most_freq, it occurs that one band member dominates the locations, because Freebase contains many more geographical data on that one member as the rest. This is the case with Ry Cooder of Buena Vista Social Club: he originates in the USA, while this is a Cuban band. The Last.fm biography can contain multiple biographies from different artists with the same name. This of course confuses the biography-based methods.

In Figure 3, we can see the accuracy for every method for every continent. It is clear that the continent classification works better for Europe, North America and Oceania. This can be due to the strong representation of these continents in the ground truth or a stronger representation on the web (see Table 1). To be able to see a consistent trend, we need more data for the smaller continents. We noticed that the misclassifications of the biography-based methods for Asia can often be traced back to Japan, because of Japanese album releases. Some of the Belgian and Dutch artists are classified with former colonies. This happens when the biography contains the place of birth or performance locations. Bio-favour_1st outperforms the two other biography methods in Africa, Asia and South-

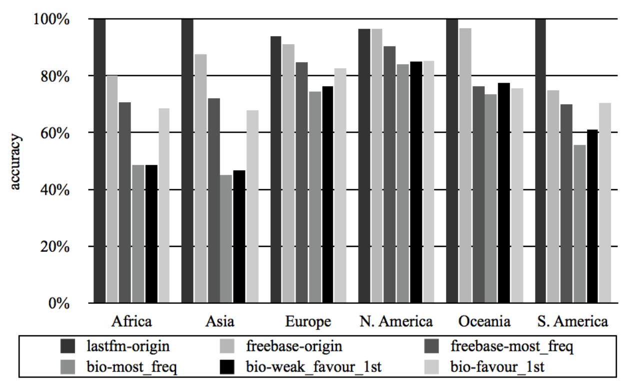


Figure 3. The accuracy of every method for every continent on the LFB data set.

America. Due to the high number of countries it is impossible to this on country level.

3.2.3 Improve by combination (combo_method)

We will now introduce a new method that maximizes coverage and improves performance beyond bio-favour_1st. The idea of the new method is to use all origins found by all methods to improve the coverage and select the origins smartly to improve accuracy.

The result is selected in an order based on the accuracy in Table 2. Since the 95% confidence intervals of the Last.fm and Freebase methods overlap, we don't know which one is significantly better. We used the highest accuracy and the spread of the confidence intervals to order. If a location of lastfm-origin is available, this is the result, because it has the highest accuracy. Otherwise freebase-most_freq is selected, because the confidence interval of freebase_most_freq lies encapsulated in that of freebase-origin, then freebase-origin and as last option bio-favour_1st. We can still improve this slightly by using the continent accuracy in Figure 3. Freebase-origin performs better in Africa, Asia and South-America. If the origin determined by freebase-origin is from these continents, this is selected before freebase-most_freq.

Obviously, the coverage of the new method (59%) for the complete data set equals the percentage found by all methods (Figure 2). The accuracy of the combo-method for classifying origin on the combo data set is shown in

	recall	precision	F-measure
US	0,8824	0,8468	0,8642
GB	0,8092	0,8513	0,8297
FR	0,7576	0,8117	0,7837
DE	0,6281	0,8444	0,7204
NL	0,6186	0,7849	0,6919
BE	0,5385	0,9825	0,6957
IT	0,6538	0,6939	0,6733
SE	0,8718	0,8293	0,8500
JM	0,7586	0,9565	0,8462
CA	0,8947	0,6071	0,7234

Table 4. Precision, recall and F-measure of the top countries categorized by combo_method.

methods	origin (%)	CI origin	cont. (%)	CI cont.
lastfm-origin	89,58	[85,26; 93,90]	94,79	[91,65; 97,93]
freebase-origin	90,85	[88,61; 93,09]	94,16	[92,33; 95,99]
freebase-most_freq	91,60	[89,65; 93,55]	94,70	[93,12; 96,28]
bio-most_freq	64,63	[62,33; 66,93]	77,81	[75,81; 79,81]
bio-favour_1st	71,04	[68,85; 73,22]	82,04	[80,19; 83,89]
bio-weak_favour_1st	66,26	[63,98; 68,54]	78,66	[76,69; 80,63]
combo_method	77,09	[75,12; 79,06]	86,29	[84,67; 87,90]

Table 3. The accuracy and 95% confidence intervals for all methods on the combo data set.

Table 3. The method with the highest coverage previously, bio-favour_1st, improves from 71% to 77% and for continent to 86%. The lower accuracy, compared to the Freebase and Last.fm methods, is caused by half of the results, which are from bio-favour_1st. The combo-method performs 14% less than the best method, freebase-most_freq, but covers more than double.

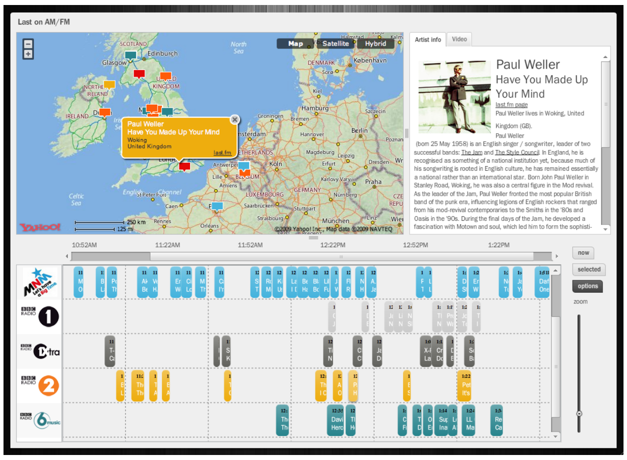
Table 4 shows the recall, precision and F-measure of the 10 countries with the most artists classified by combo_method. Overall the precision is higher than the recall. A higher precision is preferred for this task, because the main task of the classifier is to classify new artists. Belgium, the Netherlands and Germany have a rather low recall value. Currently, these countries are the main markets of Aristo Music, so there are comparatively more globally lesser-known artists in the database for these countries than others and thus less data available on the web. The lower precision for Canada is probably caused by misclassifications with US and mistakes by the biography analysis: for example, if the bio mentions "French speaking", then the artist may be classified with France.

4. MASHUP

As a showcase for the classifier, a mashup application was developed, that visualizes playlists from radio stations and locates the artists on a map, based on the origin classifier. We also link the data to YouTube videos, biographies and the Last.fm account.

The playlists are retrieved from Last.fm accounts, Twitter or scraped from radio station websites with Dapper. Yahoo! Pipes retrieves the data from these sources and adds the biography, pictures, track duration and Last.fm URL from the Last.fm API. Then we apply the origin classifier, implemented as a REST web service on Google App Engine. It requires the artist, the biography and the Last.fm artist page URL; e.g. http://artistlocator.appspot.com/?artist=Morrissey&bio=Test+bio&last_fm_url=http://www.last.fm/music/Morrissey. Figure 4, shows a screenshot of the mashup application, http://www.cs.kuleuven.be/~sten/lastonamfm/.

During its 24 days online, the mash-up attracted 220 individual visitors; almost 30% of them return. Most visitors come from Belgium, USA, France and UK. On 8 May, it was "Mashup of the Day" on Programmable Web (http://www.programmableweb.com/). A Belgian na-



Paul Weller
Have You Made Up Your Mind
last.fm page
Paul Weller lives in Woking, United Kingdom (GB).
Paul Weller
(born 25 May 1958) is an English singer / songwriter, leader of two successful bands: The Jam and The Style Council. In England, he is recognised as something of a national institution yet, because much of his songwriting is rooted in English culture, he has remained essentially a national rather than an international star. Born John Paul Weller in Stanley Road, Woking, he was also a central figure in the Mod revival. As the leader of the Jam, Paul Weller fronted the most popular British band of the punk era, influencing legions of English rockers that ranged from his mod-revival contemporaries to the Smiths in the '80s and Oasis in the '90s. During the final days of the Jam, he developed a fascination with Motown and soul, which led him to form the sophisti-

Figure 4. A screenshot of the mashup application.

tional radio station discussed it on air. Another radio station intends to use it to analyze playlists of competitors.

5. CONCLUSION AND FUTURE WORK

Our aim was not to build a very complex classifier, but to see how far we could get using simple techniques. Real life systems can often benefit from this, e.g. short computation time. This rather simple approach leads to decent performance. From 6 different methods to classify the origin, the 4 best performing are combined in the final classifier to maximize coverage to 59%. The resulting origins are selected from the classifier with the highest accuracy. This results in a final accuracy of 77% for origin classification and 86% for continent classification.

The classifier can probably benefit from applying data mining techniques. For example the biographies can be analyzed for words that co-occur for artists of the same country. Something similar could be trained on the Last.fm tags of artists with the same origin to extract geographical information from tags. Another idea is to use named entity recognition, like Open Calais, to extract birthplaces and other geographical facts. This could complement the demonym analysis to get more detailed information, for example city names.

One important way to improve the classifier is by increasing the coverage. The identification of an artist online has to be improved to get the correct biography. This can be done with online identification services, e.g. MusicBrainz. More data sources can also enable further improvement. People search engines sometimes show the birthplace and could thus be leveraged. Additional data sources, such as Belgian rock/pop catalogs, could be exploited to cover local artists. An obstacle might the intellectual properties of such collections. The web can be used as a whole to extract information from by for example crawling music related sites, using search engines for classification [4] or using links from semantic search engines, like http://sig.ma.

In any case, we believe that our classifier is accurate enough to automatically annotate the origin of an artist for applications like that of Aristo Music. This can remove the need for costly manual effort and enable scalable automation for an important metadata element.

6. ACKNOWLEDGEMENT

We gratefully acknowledge the support of IWT Vlaanderen on the project: "MuziK: Muziek op maat van de klant" under grant IWT 080226 and Aristo Music NV for the evaluation data, metadata annotation time management study and their musical expertise.

7. REFERENCES

[1] N. Corthaut, S. Govaerts, E. Duval. “Moody Tunes: The Rockanango Project”, Proc. of the 7th Int. Conf. on Music Inf. Retrieval, 2006, pp. 308-313.

[2] S. Govaerts, N. Corthaut, E. Duval. “Mood-ex-Machina: Towards Automation of Moody Tunes”, Proc. of the 8th Int. Conf. on Music Inf. Retrieval, 2007, pp. 347-350.

[3] G. Tzanetakis, P. Cook. “Musical genre classification of audio signals”, IEEE Trans. on Speech & Audio Proc., Jul. 2002, 10(5), 293-302.

[4] M. Schedl, T. Pohle, P. Knees, G. Widmer. “Assigning and visualizing music genres by web-based co-occurrence analyis”, Proc. of the 7th Int. Conf. on Music Inf. Retrieval, 2006.

[5] J. Bergstra, A. Lacoste, D. Eck. “Predicting genre labels for artists using FreeDB”, Proc. of the 7th Int. Conf. on Music Inf. Retrieval, 2006.

[6] I. Becerra-Fernandez. “The role of artificial intelligence technologies in the implementation of People-Finder knowledge management systems”, Knowl.-Based Syst., 2000, 13(5), pp. 315-320.

[7] Ò. Celma, M. Nunes, “GeoMuzik: A geographic interface for large music collections”, Proc. of the 9th Int. Conf. on Music Inf. Retrieval, 2008.

[8] B. Pouliquen, R. Steinberger, C. Ignat, T.De Groeve. "Geographical information recognition and visualization in texts written in various languages", SAC '04, 2004, pp. 1051-1058.

[9] Paul Lamere. “Social tagging and music information retrieval”, Journ. New Music Res., 37(2), 101–114.

[10] K. Bollacker, C. Evans, et al. “Freebase: a collaboratively created graph database for structuring human knowledge”, 2008, ACM SIGMOD, pp. 1247-1250.

[11] M. Meire, X. Ochoa, E.Duval, “SAmgI: Automatic metadata generation v2.0”, 2007, Proc. of EdMedia07, AACE, pp. 1195-1204.

CHRONICLE: REPRESENTATION OF COMPLEX TIME STRUCTURES

Wijnand Schepens
University College Ghent, Belgium
wijnand.schepens@hogent.be

ABSTRACT

Chronicle is a novel open source system for representing structured data involving time, such as music.

It offers an XML-based file format, object models for internal representation in various programming languages, and software libraries and tools for reading and writing XML and for data transformations.

Chronicle defines basic blocks for representing time-based information using events, a hierarchy of groups and instantiable templates. It supports two modes of timing: local timing within a group and association with other elements. The built-in mechanism for resolving time references can be used to implement both timescale mappings and tagging of information.

Chronicle aims to be a powerful and flexible foundation on which new file formats and software can be built. Chronicle focuses on structure and timing, but leaves the actual content free to choose. Thus format- or software-developers can specify their own domain-model. This makes it possible to make representations for different types of musical information (scores, performance data, ...) in different styles or cultures (CMN, non-western, contemporary, ...), but also for other domains like choreography, scheduling, task management, and so on. It is also ideal for structured tagging of audio and multimedia (movie subtitles, karaoke, synchronisation, ...) and for representing "internal" data used in music algorithms.

The system is organized in four levels of increasing complexity. Software developed for a specific level and domain will also accept lower level data, while users can choose to represent data in a higher level and use Chronicle tools to reduce the level.

1. INTRODUCTION

Since the beginning of computing hundreds of music encodings (representations, formats) have been invented, and new ones are still being developed today. For overviews see e.g. [1–3]. Part of the reason is that the landscape of forms of musical information is so large, ranging from audio to symbolic, from performance to score, from western to non-western, from classical to popular, from ancient to contemporary. Some of these terrains are relatively well established, even standardized, others are still being explored.

Most symbolic music encodings focus on common western music notation (CWMN). However, many new applications are pushing the limits of conventional encodings. Some examples:

- non common western music, e.g. traditional African music, medieval music
- special notations such as percussion notations, tablature, Gregorian plainchant
- extra data such as lyrics, choreography, instrument-specific notations, harmony. Also: auxiliary data for music software, e.g. chroma vectors, onset times, ...
- synchronization or alignment with multimedia, annotations and labeling

Existing encodings are not always suitable to accommodate the storage of these types of data. Developers are forced to invent their own encoding, or to abuse existing encodings such as MIDI [1].

The most important common factors in these applications are time and structure. We have developed a new system for dealing with structured data involving time, called *Chronicle*. The system deals with time and structure, but leaves the actual content or data free to choose. A developer has the freedom to represent the content in any form he wishes. Thus the Chronicle system can serve as the skeleton for a variety of applications, allowing the developer to concentrate on content-specific issues.

In the early 90s, one of the first systems dealing with time was HyTime [4] on which the symbolic music encoding SMDL [5] was based. Although both are international ISO/IEC standards, and have had a lot of influence on later initiatives, they have never been used in practice. There are a number of reasons for this, but the main reason is probably that the system was far too complex. Chronicle is similar in some aspects, but aims to be as simple as possible. Related efforts for symbolic music can be seen in Music Space [6] and SMI [7] or IEEE 1599 [8].

Similar time-based approaches can be found in the field of multimedia, notably with SMIL [9] and its recent offspring Timesheets [10]. For a more theoretical background please refer to [11]. These systems were designed to schedule multimedia presentations. The use of parallel and sequential groups of elements (sound, image, movie) can also be found in Chronicle, but Chronicle offers much more advanced mechanisms for annotations, timescale mappings, templates etc.

Chronicle is intended to aid developers of (music) software and file formats, by providing simple yet powerful and flexible basic building blocks and structuring mechanisms. It offers support for working with events, groups, relative time, parallel and sequential layout, timescales, timescale mappings, association, parametrized templates and more. The system was developed primarily for symbolic music, but is also applicable in other domains like audio, multimedia, choreography, job scheduling etc.

The Chronicle system consists of different parts:

- external representation: XML-based file format
- software tools for manipulating Chronicle XML files
- internal representations: object models (interfaces, classes, ...)
- software libraries for manipulating, storing, parsing...

Internal representations and libraries are developed in various programming languages [1] .

The main aim is to facilitate the development of new domain-specific encodings and software by providing internal data representations (in the form of interfaces and classes) and software-libraries for various tasks such as writing and parsing XML, processing events, manipulating structure, querying information etc.

2. DESIGN

2.1 Elements and ID's

The basic elements are *events* and *groups*. A group contains child-elements, which are either events or sub-groups. Sub-groups contain other elements and so on. The root-group is the common ancestor of all elements. The result is a hierarchy or tree-structure, comparable to a file system. Every element in the tree, except the root, has a *parent group*.

Every element is identified within its parent group by a unique integer number called *local* ID. By default elements are numbered 0, 1, 2, ..., but it is possible to override this by an explicit ID-definition.

Every element in the hierarchy is uniquely identified by its local ID, the local ID of its parent group, the local ID of its grandparent group etc. This sequence of local IDs is called the elements *global* ID.

It is convenient to introduce *path-notation*, where a global ID is specified by concatenating the local IDs top-down (starting from the root) separated by slashes, similar to a file path or URL. The root itself is notated by `/`. Thus, for example, global ID `/2/3` identifies the element with local ID 3 in the sub-group with local ID 2 in the root group. Optionally, an element can have a name (String) which is also unique within its parent group. Names can be used to embellish path-notation, e.g. `/part1/section3/4`.

[1] currently Java and ActionScript

Path expressions starting with `/` are called *absolute*, because they identify an element starting from the root. It is also possible to identify elements using a *relative* ID from within another element. As in a file-path, one can use ".." in path-notation to indicate the parent (group). For example `../../section/2` refers to the element with local ID 2 in a group named "section" in the grandparent group of the current element. As an alternative, the notation `@`*`name`* can be used to refer to an element named *name* anywhere up in the hierarchy. First the parent group is searched. If the element is not found there, the grandparent is searched, and so on, until the root is reached. This mechanism is similar to the lookup of variable names in nested scopes. For example, `@section/2` is equivalent to `../../section/2` if the grandparent contains a child named "section", but the parent doesn't.

2.2 Time

All elements have a timestamp, which is expressed either in *local time* or in *non-local time*.

Local time is represented by an integer number and is to be interpreted relative to the timescale of the parent group. The parent group has its own (start-)time and, optionally, a scale. An example in XML-form:

```
<group time="100" scale="4">
   <event time="10" />
</group>
```

In this fragment the "absolute" time of the event would be $100+4\times10 = 140$. If an element's time is not specified, it defaults to zero (local time). The scale defaults to one.

Alternatively, an element can specify its timestamp in non-local time using a *time reference*. A time reference is represented by a path-expression, either absolute or relative, which can be written using path-notation starting with `/`, `..` or `@`. If this path refers to an element which exists in the tree, then the referring element gets the same time as the element it refers to.

```
<event name="e1" time="100" />
...
<group>
  <event name="e2" time="@e1" />
</group>
```

In this example event "e2" refers to event "e1", so it also has time 100. Here we used `@e1` in the path expression. Equivalently, one could specify this using a relative path `../e1` or an absolute path.

Time references can also be chained (refer to a reference...) and mixed with local timing:

```
<group name="e1" time="100" />
  <event time="10" />
  <event time="20" />
</group>
...
<group time="@e1/0">
  <event name="e2" time="1" />
</group>
```

Here event "e2" uses local time, which is added to the parent group time, resulting in `@e1/1`. This path refers to the second element (id=1) of the group ”e1”. The result is that the time of event ”e2” resolves to $100 + 20 = 120$.

The technique of time references is very powerful. It can be used for associating elements with other elements (tagging...), but also to implement mappings between time-scales. This will be illustrated in section 3.

Chronicle also supports layout-schemes which allows automatic determination of element times. The most common examples are sequential and parallel layout which schedule elements resp. one after the other and at the same time.

2.3 Levels

Chronicle is organized in four levels of increasing complexity.

A level 0 file consists of one list of events. There is one global timescale. All events use local time relative to this timescale. The events are ordered in ascending time order: every event must have a time greater than or equal to that of its predecessor. No restrictions are imposed on the data carried by the events.

Level 1 adds the possibility of groups and nesting, with the restriction that groups have starting time equal to zero. This means that local times in groups are equivalent to ”absolute” times in the global timescale. In this (and higher) levels, the events needn’t be ordered in time.

In level 2 all elements, including groups, can specify their (start) time as a local time or using a time reference, and groups can use a layout-scheme to set child element times.

Finally, level 3 introduces templates and template instances. Note that every level is a subset of the higher levels. Chronicle provides tools to transform data to a lower level.

2.4 Domain

As noted in the introduction the Chronicle system only deals with structure and timing, not with the actual domain-specific content, which can be chosen freely. Different Chronicle *applications* can be developed for specific domains representing different types of musical information (scores, performance data, ...) in different styles or cultures (CMN, non-western, contemporary, ...), but also for other domains like choreography, scheduling, task management, and so on. It is also ideal for structured tagging of audio and multimedia (movie subtitles, karaoke, synchronisation of score and audio, ...) and for representing ”internal” data used in music algorithms (chroma, coordinates, ...)

In the XML-format domain-specific information is encoded *in the content of event-elements*, either as text or as one or more child elements. Additionally it is possible to give the event an attribute `type`. To illustrate this we show some possibilities for encoding a chord-symbol:

```
<event>Am7</event>
<event type="chord">Am7</event>
<event><chord>Am7</chord></event>
<event>
  <chord><root>A</root><kind>m7</kind></chord>
</event>
```

We have provided API’s to read and write Chronicle elements (event, group, ...), but it is up to the user to deal with event-content. Groups cannot carry data themselves, although it is possible to specify a group-type using the `type`-attribute. If necessary, extra events can be introduced in a group to encode group-related information.

3. EXAMPLE

We will demonstrate the key features of the Chronicle system and the level reduction process by means of an example. We present the material in XML-form, although it should be borne in mind that Chronicle also supports ”internal” representations and transformations in different programming languages.

```
<chronicle version="2.0" level="3"
  domain="http://.../leadsheet" />

<template name="note" >
  <group>
    <event time="0">
      <note_on  pitch="#pitch" />
    </event>
    <event time="#dur">
      <note_off pitch="#pitch" >
    </event>
  </group>
</template>

<template name="voice">
  <group>
    <group name="notes" layout="sequential">
      <instance model="/note"
                pitch="D4" dur="12" />
      <instance model="/note"
                pitch="D4" dur="12" />
      <instance model="/note"
                pitch="D4" dur="9" />
      <instance model="/note"
                pitch="E4" dur="3" />
      <instance model="/note"
                pitch="F#4" dur="12" />
      ... MORE NOTES ...
    </group>
    <group type="lyrics" time="@notes/0">
      <event time="0" type="lyric">Row,</event>
      <event time="1" type="lyric">row,</event>
      <event time="2" type="lyric">row</event>
      <event time="3" type="lyric">your</event>
      <event time="4" type="lyric">boat</event>
      ... MORE LYRICS ...
    </group>
  </group>
</template>

<group name="song" time="@ticks/0">
  <group name="canon" layout="sequential">
    <group time="0" name="first">
      <instance model="/voice" />
    </group>
    <group time="48" name="second">
      <instance model="/voice" />
    </group>
    <group time="96" name="third">
      <instance model="/voice" />
    </group>
    <group time="144" name="fourth">
      <instance model="/voice" >
    </group>
  </group>
  <instance name="unison" model="/voice">
</group>

<group name="ticks" time="@ms/0" >
```

```
  <event id="0"   time="0" />
  <event id="192" time="19200" />
  <event id="240" time="21600" />
</group>

</chronicle>
```

This example illustrates an encoding of the song ”Row, row, row your boat”. It is important to note that this is only one of many possible encodings.

After the XML-preamble, the file starts with a root-element `chronicle`. The attribute `version` specifies the version of the Chronicle system itself. The attribute `level` indicates the encoding level used, and the attribute `domain` specifies the domain (content types and structural restrictions). In this case the attribute points to a URI.

The first element defines a template `note`. A template is a kind of prototype which can be instantiated (copied) multiple times. Templates are useful for avoiding code-duplication. A template can have parameters, which makes it possible to vary the instances. In this case the `note`-template is used as a convenient way to bundle a note-on and note-off event. It represents a single note with a certain pitch and duration. The pitch is encoded as a simple string which indicates pitch class (e.g. `F#`) and register or octave (4th octave). This is not dictated by Chronicle but is a choice made by the domain-developer.

Next, the template `voice` defines notes and lyrics of the song. The group `song` instantiates five copies at different times, representing four voices sung in canon, followed by one in unison. Finally, group `ticks` relates the ticks-timescale to milliseconds.

The example is a level-3 encoding. We will now illustrate how it can be reduced to lower levels.

The first phase, which transforms from level-3 to level-2, is template instantiation, also called expansion. It is carried out bottom-up: first the innermost elements, then their parents and so on. In this case, the voice-template contains instances of note-templates which are instantiated expanded first, the parameters `#pitch` and `#dur` being substituted by their actual values defined in attributes `pitch` and `dur`. Subsequently, the four voice-instances in the song-group are expanded. Since this template has no parameter, the instances are exact copies. In the resulting level-2 file the templates have disappeared:

```
<chronicle version="2.0" level="2"
  domain="http://.../leadsheet" />

<group name="song" time="@ticks/0">
  <group time="0" name="first">
    ...
  </group>
  <group time="48" name="second">
    <group>
      <group name="notes" layout="sequential">
        <group>
          <event time="0">
            <note_on  pitch="D4" />
          </event>
          <event time="12">
            <note_off pitch="D4" />
          </event>
        </group>
        ... MORE NOTES ...
```

```
          <group>
            <event time="0">
              <note_on  pitch="E4" />
            </event>
            <event time="3">
              <note_off pitch="E4" />
            </event>
          </group>
          ... MORE NOTES ...
        </group>
        <group type="lyrics" time="@notes/0">
          <event time="0" type="lyric">Row,</event>
          <event time="1" type="lyric">row,</event>
          <event time="2" type="lyric">row</event>
          <event time="3" type="lyric">your</event>
          <event time="4" type="lyric">boat</event>
          ... MORE LYRICS ...
        </group>
      </group>
    </group>
  ... THIRD AND FOURTH VOICE ...
</group>

<group name="ticks" time="@ms/0" >
  <event id="0"   time="0" />
  <event id="192" time="19200" />
  <event id="240" time="21600" />
</group>

</chronicle>
```

Reduction from level-2 to level-1 is carried out in two phases. The first phase is the *layout* phase. In the example, the `notes`-group has sequential layout, which means that its elements must be scheduled one after the other. Technically, the (start) time of element $i + 1$ is equal to the (start) time of element i plus the duration of element i, for all i.

The duration of a group is equal to the largest local time of (grand)child events. If necessary the duration can also be specified explicitly. In the case of the notes, the duration of a note-group is equal to the time of the offset-event.

In the resulting file the element groups have acquired an explicit time, and the layout-indications are gone. In the example, the note-groups have times 0, $0 + 12 = 12$, $12 + 12 = 24$, $24 + 9 = 33$, $33 + 3 = 36$ and so on. This is illustrated in the following fragment which shows the onset-event of the fourth note (E4) in the second voice, and its ancestor groups:

```
<group name="song" time="@ticks/0">
  <group time="48">
    <group>
      <group name="notes">
        ...
        <group time="33">
          <event time="0">
            <note_on  pitch="E4" />
          </event>
```

The layout phase is followed by the *time resolution* phase. The timestamps of all elements are resolved in the manner illustrated in section 2.2.

Consider for example the onset of the fourth note (E4) in the fragment above. Following the way up from parent to parent, it can be seen that the additions result in a time equal to `@ticks/81`. This means that that note starts on tick 81.

The domain-developer has chosen to encode the lyrics in a separate group which is *associated* with the notes-group. In the group `lyrics` the event times are added to the group-time, yielding value `@notes/0`, `@notes/1` and so on. The `@notes`-reference points to the `notes`-group. Therefore the timestamps of the lyric-events are substituted by the timestamps of the note-groups which they refer to, in this case `@ticks/0`, `@ticks/12` and so on.

The reference `@ticks` in turn points to the `ticks`-group defined near the end of the example. As this group contains only three events with local ID 0, 192 and 240, a reference like `/ticks/12` doesn't point to a real element. Such a reference is called *virtual*. In that case times are resolved by a linear interpolation between real elements. In the example ID 0 maps to time 0 and ID 192 maps to 19200, so each tick in this range has a duration of 100 ms. This means, for instance, that `@ticks/12` resolves to `@ms/1200` which signifies that tick 12 occurs after 1200 milliseconds. Ticks in the range up to 240 have a duration equal to $(21600-19200)/(240-192) = 50$ ms. As a result, the fifth instance of the voice-template (named "unison") is played in double tempo.

The net effect is that the `ticks`-group defines a mapping between timescales. Note that the mechanism for resolving (local or non-local) times is used to accomplish two different goals: a. the lyrics are associated (tagged) to notes, b. the ticks timescale is mapped to the millesecond timescale.

Note that all references can be resolved to a form `ms/`x where x is an integer number. The reference `@ms/...` cannot be resolved any further - this is the "global" timescale. If we set all group times to zero, then the absolute times are equivalent to relative times. The result is a level-1 file, with the timescale `ms` specified in the root-element:

```
<chronicle version="2.0" level="1"
  domain="http://.../leadsheet"
  timescale="ms" />

<group name="song">
  <group name="first">
    ...
  </group>
  <group name="second">
    <group >
      <group name="notes" >
        <group>
          <event time="4800">
            <note_on  pitch="D4" />
          </event>
          <event  time="6000">
            <note_off pitch="D4" />
          </event>
        </group>
...
        ... MORE NOTES ...
        <group>
          <event time="8100">
            <note_on  pitch="E4" />
          </event>
          <event time="8400">
            <note_off pitch="E4" />
          </event>
        </group>
...
        ... MORE NOTES ...
      </group>
      <group type="lyrics" >
        <event time="4800" type="lyric">
Row, </event>
        <event time="6000" type="lyric">
row, </event>
        <event time="7200" type="lyric">
row </event>
        <event time="8100" type="lyric">
your </event>
        <event time="8400" type="lyric">
boat </event>
...
        ... MORE LYRICS ...
      </group>
    </group>
  </group>
  ...
  ... THIRD AND FOURTH VOICE ...
</group>
...
... TICKS ...
</chronicle>
```

To reduce from level-1 to level-0 one more transformation is needed. In this final phase groups are *serialized* or *flattened* into one long series of events, and they are ordered in ascending time order. In the resulting level-0 file, there are no more groups:

```
<chronicle version="2.0" level="0"
  domain="http://.../leadsheet"
  timescale="ms" />
...
<event time="4800">
  <note_on  pitch="D4" />
</event>
<event time="4800" type="lyric">Row,</event>
<event  time="6000">
  <note_off pitch="D4" />
</event>
...
<event time="8100">
  <note_on  pitch="E4" />
</event>
<event time="8100" type="lyric">your</event>
<event time="8400">
  <note_off pitch="E4" />
</event>
...
</chronicle>
```

4. IN PRACTICE

How can the Chronicle system be used in practice?

The XML-format can be used by software-developers as a convenient means to persist data. Chronicle provides easy-to-use libraries for writing and reading the XML-format. For new software projects it may even be advisable to use the Chronicle classes and interfaces as the basis for the object model, even if XML-persistence is not needed. It is up to the developer to specify the domain-model by constraining content types and level.

Consider, for example, a Chronicle-encoding of MIDI-information (see e.g. [1]) By its very nature, this application is ideal for a level-0 encoding, i.e. a flat list of simple events. The domain model establishes event-types (note-on, note-off, control change, program change) and their XML-encoding. A typical event could look like this:

```
<event time="196">
  <note_on key="100" velocity="127" channel="0">
</event>
```

Software applications operating on this data, for example for playing the music, typically process the events one by one. Whereas the processing software is happy to consume level-0 data, from a musical point of view it may be desirable to add some structure. This can be achieved by encoding in a higher level, using mechanisms such as grouping, association, sequences, templates etc. Chronicle tools can then be used to transform to level-0 and feed the reduced data to the processing software.

The nice thing is that users can choose the organization which best suits their needs. For example, one might choose a "part-by-part" organization using parallel groups for different instrument parts, or a "frame-by-frame" organization using a sequence of parallel note-groups. One can choose to use sequential groups for measures, for sections (movements). It is also possible to play around with associations, timescales and time-mappings, and so on.

One important point which hasn't been addressed is that Chronicle files can be embedded as subgroups within another chronicle file using an `include`-element. The mechanism of association by time-references can be used to add information to a group without changing it. This is especially useful if the target is an embedded group, or if it is external data such as an audio or multimedia file.

Chronicle is currently being tested for the encoding of leadsheets for *wikifonia.org*. In the near future, we are planning to create more domain-specific applications, and hope that other developers will do the same.

Documentation, tools and open source code can be found at **http://code.google.com/p/chronicle-xml/**.

5. REFERENCES

[1] E. Selfridge-Field: *Beyond MIDI: The Handbook of Musical Codes*, MIT Press, Cambridge MA, 1997.

[2] K. Ng, and P. Nesi: *Interactive Multimedia Music Technologies*, Information Science Publishing, 2007.

[3] R. Cover: "XML and Music," retrieved July 6, 2009 from http://xml.coverpages.org/xmlMusic.html, 2006

[4] C. Goldfarb: Standards: "HyTime: A standard for structured hypermedia interchange," *IEEE Computer magazine*, 24(8), 1991.

[5] D. Sloan: "Aspects of Music Representation in HyTime SMDL.," *Computer Music Journal*, 17, 51-59, 1993.

[6] J. Steyn: "Introducing Music Space," *Proceedings of the 4th Open Workshop of MUSICNETWORK: Integration of Music in Multimedia Applications*, Barcelona, 2004.

[7] G. Haus, and M. Longari: "A multi-layered, time-based music description approach based on XML," *Computer Music Journal*, 29, 70-85, 2005.

[8] L. Lucidovo: IEEE 1599: "a Multi-layer Approach to Music Description," *Journal of Multimedia*, 4(1), 2009.

[9] D. Bulterman, J. Jansen, et al.: "Synchronized Multimedia Integration Language (SMIL 3.0)," retrieved July 6, 2009, from http://www.w3.org/TR/2008/REC-SMIL3-20081201/, 2008.

[10] P. Vuorimaa, D. Bulterman, and P. Cesar: "SMIL Timesheets 1.0 - W3C Working Draft," retrieved July 6, 2009, from http://www.w3.org/TR/2008/WD-timesheets-20080110/, 2008.

[11] S. Boll, U. Klas, and W. Westermann: "A Comparison of Multimedia Document Models Concerning Advanced Requirements," *Technical Report Ulmer Informatik-Berichte* No 99-01, 1999.

EFFICIENT ACOUSTIC FEATURE EXTRACTION FOR MUSIC INFORMATION RETRIEVAL USING PROGRAMMABLE GATE ARRAYS

Erik M. Schmidt
MET-lab, Drexel University
eschmidt@drexel.edu

Kris West
IMIRSEL, University of Illinois
kris.west@gmail.com

Youngmoo E. Kim
MET-lab, Drexel University
ykim@drexel.edu

ABSTRACT

Many of the recent advances in music information retrieval from audio signals have been data-driven, i.e., resulting from the analysis of very large data sets. Widespread performance evaluations on common data sets, such as the annual MIREX events, have also been instrumental in advancing the field. These endeavors incur a large computational cost, and could potentially benefit greatly from more rapid calculation of acoustic features. Traditional, cluster-based solutions for large-scale feature extraction are expensive and space- and power-inefficient. Using the massively parallel architecture of the field programmable gate array (FPGA), it is possible to design an application specific chip rivaling the speed of a cluster for large-scale acoustic feature computation at lower cost. Recent advances in development tools, such as the Xilinx Blockset in Simulink, allow rapid prototyping, simulation, and implementation on actual hardware. Such devices also show potential for the implementation of MIR systems on embedded devices such as cell phones and PDAs where hardware acceleration would be an absolute necessity. We present a prototype library for acoustic feature calculation for implementation on Xilinx FPGA hardware. Furthermore, using a genre classification task we compare the performance of simulated hardware features to those computed using standard methods, demonstrating a nearly negligible drop in classification performance with the potential for large reductions in computation time.

1. INTRODUCTION

The extraction of appropriate acoustic features is the first step for nearly all audio-based music information retrieval applications. Many recent advances in MIR systems are the result of large-scale, data-driven analysis of audio samples. Such corpora may contain thousands (or even millions, in the case of some commercial databases) of audio files, and the accompanying analyses of these data sets demands vast computational resources. In seeking improved methods for music classification and understanding, researchers are constantly searching for more informative feature sets, which requires the ability to rapidly prototype and evaluate new features on very large databases. Additionally, performance evaluations, such as the annual MIREX events [1], of multiple approaches to specific application tasks on common data sets have proven to be invaluable for advancing the state-of-the-art in MIR research. These evaluations, however, are increasingly difficult to administer, since both the number of participants and the size of the data sets continues to grow annually.

The most common solution to problems having such computational demands involves an investment in computing clusters, which are expensive and inefficient (in terms of both their utilization of hardware resources and energy consumed). Using the massively parallel architecture of the field programmable gate array (FPGA) it is possible to achieve parallel processing on a scale similar to that of a small cluster (for specific applications) on a single chip. Current tools such as the Xilinx System Generator (XSG) for DSP [1] enable rapid prototyping of DSP algorithms in the graphical language of Simulink with the ability to incorporate hardware in the modeling and design loop. Additionally, any algorithm built using the Simulink Xilinx Blockset can be easily compiled into Verilog or VHDL code and incorporated into a larger hardware system design. One possible implementation would be MIR on embedded, mobile devices, such as cell phones and PDAs, where the computation of acoustic features would not be possible using the onboard CPU. Such a hardware acceleration system could be designed both for the computation of acoustic features, and evaluation of the decision function of a pre-trained classifier.

We have developed an acoustic feature extraction library, implemented using XSG, that can be synthesized directly on supported FPGA hardware. The library supports the calculation of both Mel-Frequency Cepstral Coefficients (MFCC) [2] and common Statistical Spectrum Descriptors (SSDs). Here, we present preliminary classification results using fixed-point MFCC features calculated by the XSG (simulating an FPGA implementation). The accuracy of these features is verified through their use in a genre classification task on a medium-sized audio database, and we demonstrate that the resulting classification performance is comparable to that of a floating-point MATLAB and double-precision Java implementation of similar feature cal-

[1] Xilinx System Generator: http://www.xilinx.com/ise/optional_prod/system_generator.htm

culation algorithms.

Furthermore, we also present a process for migrating acoustic features designed using MATLAB to the Xilinx System Generator in Simulink. The ultimate goal of this endeavor is to develop a system in which features can be rapidly and easily prototyped within Simulink, synthesized to Verilog hardware description language (HDL), and embedded into a larger standalone FPGA-based system-on-chip design. Tools developed for such a platform could be easily shared between members of the MIR research community and could potentially allow even an inexperienced HDL programmer to take advantage of the performance gained from implementing feature extraction in hardware.

2. BACKGROUND

Early efforts at implementing acoustic feature computation in hardware required systems built entirely from scratch, which is a significant and time-consuming endeavor when working directly with low-level hardware descriptor languages (HDLs). Prior work has almost exclusively targeted MFCC feature calculation for automatic speech recognition. For example, [3] presents an optimized algorithm for efficient computation of MFCCs using an FPGA implementation, while [4] focuses on implementing only the FFT sub-calculation in hardware for eventual use in MFCC computation. In the direction of easing the dependence on hardware arithmetic units, [5] proposes modifying the MFCC algorithm from a triangular filterbank to a mel-spaced rectangular filterbank, and their results demonstrate only a minimal decrease in classification accuracy. Other work has focused on full hardware system integration for speech recognition. In [6] an on-chip, retrainable hardware speech recognition system is presented using MFCC features and Hidden Markov Models (HMMs) for statistical pattern recognition.

The annual Music Information Retrieval Evaluation eXchange (MIREX) tasks, initiated in 2005, have become a core component of the field of MIR in terms of advancing and disseminating the latest research and results. With the number of tasks, participants, and data sets increasing annually, the evaluations have become exponentially more difficult to administer. The International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL), the organizers of MIREX, has traditionally gone above and beyond in order to accommodate a wide range of implementation platforms and architectures and retains a range of machines, including an ever-expanding cluster, for this purpose, resulting in additional complexity. This configuration, while offering a great deal of flexibility, limits their ability to take advantage of parallel processing implementations, which are still highly platform-specific. A 72-hour runtime cap is enforced for all submissions, but even so, recent evaluations have required up-to 1000 person-hours of effort. A significant speedup in feature computation could greatly reduce the runtime requirements for MIREX.

3. HARDWARE IMPLEMENTATION OF FEATURES

The Xilinx System Generator (XSG) tools in Simulink enable the prototyping of complex signal processing algorithms for hardware implementation in a relatively straightforward manner. For our initial implementation, we limited ourselves to analysis windows with a length of 512 samples and 50% overlap, 40 triangular mel-filters, and 20 DCT coefficients. These parameters were chosen because of their wide application in audio and music processing algorithms. Other research has shown that the number of filters and DCT coefficients can be greatly reduced while still maintaining adequate performance [7].

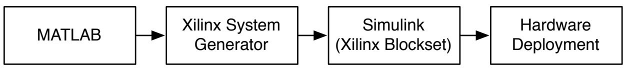


Figure 1. Implementation flowchart for audio feature extraction algorithms.

The first processing stage for our audio features requires a DFT calculation, which is performed using the pipelined Xilinx core FFT v5.0, producing real-time serial FFT data. Next, using two multipliers, we square the results of the real and imaginary parts and add them together to obtain the energy spectral density (ESD), which is quantized to 32-bits. In this case, ESD is preferred to the magnitude FFT because the square root function necessary to obtain magnitude is not easily implemented in hardware. XSG includes Coordinate Rotation Digital Computer (CORDIC) algorithms which can compute square roots as well as trigonometric, logarithmic, and division functions using only addition, subtraction, bitshift, and table lookup, but in a practical design these algorithms tend to take up large areas of the FPGA fabric and often incur large delays. In general, we avoided the use of these functions whenever possible.

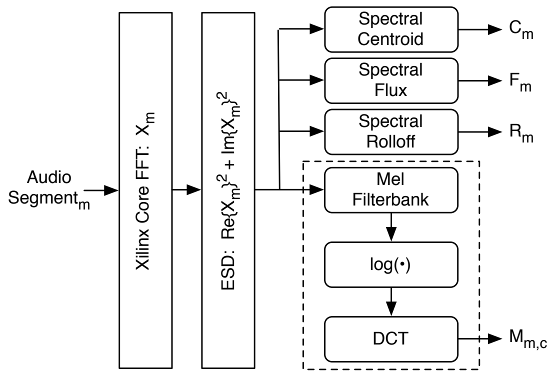


Figure 2. Hardware implementation of audio feature computation.

3.1 Mel-Frequency Cepstral Coefficients

3.1.1 Algorithm

Mel-frequency cepstral coefficients (MFCCs) are among the most widely used acoustic features in speech and audio processing. MFCCs are essentially a low-dimensional representation of the spectrum warped according to the mel-scale, which reflects the nonlinear frequency sensitivity of the human auditory system [2]. In our implementation, MFCCs are defined as

$$M_{m,c} = \sum_{b=1}^{40} \hat{X}_{m,b} \cos\left[c\left(b-\frac{1}{2}\right)\frac{\pi}{40}\right], c = 1, 2, ..., 20 \tag{1}$$

$$\hat{X}_{m,b} = \log\left(f_b[k]\left|\sum_{n=0}^{N-1} x_m[n]e^{-j\frac{2\pi nk}{N}}\right|\right), \tag{2}$$

where m represents the current frame. Normally, MFCCs are implemented over short-time segments, and accordingly our implementation divides the audio into overlapping segments and applies a Hanning window function to reduce edge effects. The Discrete Fourier Transform (DFT) of each short-time segment is computed using the FFT algorithm. The magnitude of the frequency components is determined and $f_b[k]$, the mel-spaced triangular filters (Figure 3), are applied via multiplication in the frequency domain. Continuing with the cepstrum calculation, the $\log$ of the mel-filtered energies is calculated ($\hat{X}_{m,b}$), which to some extent, serves to deconvolve the audio by transforming multiplications in the frequency-domain (and thus, convolutions in the time-domain) into additions. As a final step, the inverse DFT is applied to $\hat{X}_{m,b}$ using the DCT (sine components are not needed since the input is guaranteed to be real and even). This step is also used to reduce the dimensionality of the data to the desired quantity, and it has been shown that the DCT has the additional effect of decorrelating the vector of feature components [8].

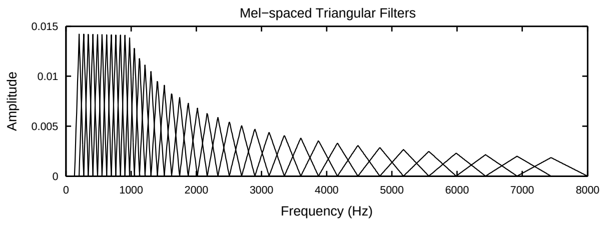


Figure 3. 40-band mel-warped triangular filterbank

3.1.2 FPGA Implementation

Calculation of the MFCC features consists of applying the mel-filterbank to the spectrum, taking the $\log$, and computing the DCT. Applying the mel-filterbank requires 40 read-only memory (ROM) elements, 40 multipliers, and 40 accumulators. A control register is placed on the output which is triggered by FFT completion to only allow the filterbank output to change once for every frame. To minimize the size of the ROM elements and the number of multiplies, the mel-filter coefficients are restricted to 16-bits. This stage is the most resource intensive part of the design due to the number of multipliers required.

Once the data is filtered, it is again serialized in order to compute the $\log$. This requires a wait of 512 samples until the next FFT frame is supplied, therefore ample time is available to serialize the 40 filter band values. Using a single CORDIC $\log$, all 40 values are computed and subsequently quantized to 32-bits.

The final step involves computation of the DCT, where the DCT coefficients are stored in 20 ROM units and have been quantized to 16-bit resolution. In addition, this step requires 20 multipliers and 20 accumulators. The output is triggered using a simple control register such that the output values change only once every 512 values. Since no further processing is required, this data is decimated by a factor of 512 and returned as the final output.

3.2 Statistical Spectrum Descriptors

In music and audio processing, Statistical Spectrum Descriptors (SSDs) are often related to timbral texture [9]. For each spectral shape function, we begin by dividing the data into short-overlapping segments, applying a Hanning window, and computing the magnitude DFT.

3.2.1 Spectral Centroid

Spectral centroid is defined as the weighted-average (center of mass) of the spectrum,

$$C_m = \frac{\sum_{k=0}^{K-1} F[k]|X_m[k]|}{\sum_{k=0}^{K-1} |X_m[k]|}, \tag{3}$$

where $X_m[k]$ is the DFT of short-time segment m, and $F[k]$ is a vector of frequencies corresponding to the bins of the magnitude spectrum.

Computation of the spectral centroid requires a multiplication, two accumulators, and a division. Here the spectrum is summed with one accumulator and the spectrum, multiplied by the respective spectral bin values, is summed by the other accumulator. After passing a control register to ensure only one value is returned for each frame, the result is divided by the CORDIC divider provided by XSG.

3.2.2 Spectral "Flux"

Spectral flux is defined as the Euclidean distance between successive spectral frames. We compute the square of this feature, which is defined as follows:

$$F_m = \sum_{k=0}^{K-1} (|X_m[k]| - |X_{m-1}[k]|)^2. \tag{4}$$

Again, $X_m[k]$ is the discrete spectrum of the current analysis frame m and $X_{m-1}[k]$ is the spectrum of the previous frame.

Spectral flux is the simplest of all of the spectral shape features to compute. The hardware consists of a 512 sample delay block to maintain a copy of the previous frame, an adder, a multiplier, and an accumulator. An important note is that we are not computing the square root, but simply the sum of the squares as to avoid the use of additional CORDIC functions.

3.2.3 Spectral Rolloff

Spectral rolloff is defined as the frequency beneath which a given proportion of the total spectral energy lies, typically 85%:

$$R_m = \frac{f_s}{K} r_m = \frac{f_s}{K}\left(\arg_{r_m} \sum_{k=0}^{r_m} |X_m[k]| = 0.85 \sum_{k=0}^{K-1} |X_m[k]|\right). \tag{5}$$

Here $|X_m[k]|$ is the magnitude of the k-th frequency sample of the current frame and r_m is the frequency sample number that produces the desired 85% rolloff.

The core of the spectral rolloff implementation consists of two accumulators. The first accumulator sums the spectrum to obtain the total energy and multiplies it by 0.85, where as the second sums the delayed spectrum until the total energy reaches 85%. Once this value is obtained, the frequency value of the correspond spectral bin is returned. Of these features, spectral rolloff is probably the most robust to quantization effects as it is returning values of an already discretized function.

4. HARDWARE PERFORMANCE AND USAGE

The initial hardware target for this project is Digilent's Virtex-II Pro Development System. The Xilinx Virtex-II FPGA on the board (XC2VP30) contains 13,969 slices, 136 18-bit multipliers, 2,448Kb of block RAM, and two PowerPC Processors. While the available amount of FPGA fabric is highly constrained, at an academic discounted cost of $299.00 USD, the board is an attractive target, and its widespread adoption in education creates greater opportunities for algorithm experimentation and deployment.

Implementing only the ESD algorithm on this chip requires 299 slices, 8 18-bit multipliers, and no block RAM. Considering the total size and resource restrictions the most limiting factor is the number of multipliers required. Using all 136 multipliers and 5,083 slices we could compute the ESD for up to 17 analysis frames in parallel. For a single FFT implementation, the first frame requires 1123 clock cycles to compute and subsequent frames require an additional 512 cycles. With 17 in parallel, the first set of frames will still need 1123 clock cycles, although we will now output 17 frames at a time. Assuming the fabric is clocked at 80MHz, a conservative clock speed, a breakdown of performance is shown in Table 1.

We compared the hardware performance using a single FFT unit on an FPGA to that of the M2K toolkit [10] and MATLAB on a set of 600 audio clips, each 30 seconds in duration (the data set used in the classification task is detailed in the next section). The M2K and MATLAB features were calculated using a single processor core of a 2.4 GHz Intel Core 2 CPU. Table 2 reveals the computation times for each feature set, averaged across all 600

FFTs	First frame (μs)	Subsequent frames (μs)	Three secs (ms)	Thirty secs (ms)
1	14.04	6.400	3.316	33.08
17	14.04	0.376	0.206	1.953

Table 1. Performance of spectrogram calculation on simulated hardware.

clips, and we observe that there is more than an order of magnitude difference between the hardware and software implementations. While MATLAB and M2K each produce results in just under a half second for each 30 second clip, the hardware requires only around 33ms. Additionally, it can be seen from Table 3 that as the number of FFT units available on the FPGA increases, the hardware can achieve sub 1ms computation times.

Toolkit	Computation Time (s)
M2K	0.483
MATLAB	0.364
FPGA	0.033

Table 2. Comparison of feature computation times between software and hardware implementations.

In its current form, implementing the full system-on-chip (including MFCC and SSD calculations) with a single FFT unit requires 15,542 slices, 145 18-bit multipliers, and 1,080 Kb of block RAM, which exceeds the capacity (in terms of slices and multipliers) of the hardware target. With some additional optimization, perhaps trading off some parallelism to reduce hardware resource utilization, we believe the full system could be implemented on the targeted Virtex-II VC2VP30-based system. Other products in the Virtex-II FPGA family provide additional hardware resources, which offer the possibility of combining multiple feature computation engines on a single chip. For example, the Virtex-II XC2VP100 provides sufficient slices and multipliers to easily accommodate 3 feature computation engines. Additionally, certain development boards contain multiple FPGA chips, adding further parallelization opportunities. With two FPGAs, we could potentially accommodate 6-8 computation engines on a single system board.

The current design requires 2048 clock cycles to produce the first output for the spectral shape features and 2560 for MFCCs. As with the ESD, each additional frame takes 512 cycles and compute time decreases proportionally with the addition of each parallel computational engine. More specific timing results for a single computation engine (again clocked at a conservative 80 MHz) are provided in Table 3, as well as theoretical performance numbers if six parallel feature computation engines could be implemented on a single system. Although such development hardware is currently quite expensive (approximately $10K, on the order of a small cluster), the performance is potentially faster by more than an order of magnitude.

Feature	Comp. Engines	First frame (μs)	Three secs (ms)	Thirty secs (ms)
S. Shape	1	25.60	3.328	33.09
S. Shape	6	25.60	0.576	5.536
MFCC	1	32.00	3.334	33.10
MFCC	6	32.00	0.582	5.542

Table 3. Performance of feature extraction on simulated hardware.

5. CLASSIFICATION EXPERIMENT

In order to confirm the efficacy of the hardware extracted features, we have conducted an evaluation of genre classification systems based on the features produced and compared its performance to that of classifiers based on the same features computed in MATLAB and the M2K toolkit [10]. We conducted two experiments using a collection of 600 tracks drawn from the Magnatune collection of Creative Commons licensed music [11], divided into six genres. An overview of the collection is given in table 4.

Genre	Number of tracks
Ambient	100 tracks
Classical	100 tracks
Electronic	100 tracks
Ethnic	100 tracks
Jazz and Blues	100 tracks
Rock	100 tracks
Total	600 tracks

Table 4. Composition of dataset for genre classification task.

Pampalk [12] identifies the potential for the over-fitting of the characteristics of a particular artist to inflate accuracy scores in the evaluation of audio content-based genre classification systems, particularly when evaluating systems on small collections. Hence, we have conducted both artist-filtered and conventional cross-validated classification experiments based on 5-fold random 80:20 splits and 5-fold stratified cross-validation, respectively.

5.1 Pre-processing of Features

The feature extractors yield a very large number of feature vectors for each track (based on 23 ms windows with 50% overlap), and in order to effectively and efficiently classify tracks, the features must be summarised to produce a smaller number of more informative vectors. One approach that has been effectively used by many authors [12–15] is to summarise the distribution of feature frames over the track. This may be performed by, for example, estimating the parameters of a single Gaussian distribution or a mixture of Gaussians. However, [16] and [12] provide experimental evidence that the performance of techniques based on mixtures of Gaussian distributions are at best equal to that of single Gaussian based approaches, making their extreme additional computational cost impossible to justify. This lack of additional discriminative power for the use of mixture distributions is at odds with research in many other audio indexing problems [16]. Hence, in our evaluation the feature stream is summarised as a flattened single Gaussian distribution (mean vector and and flattened upper triangular covariance matrix).

5.2 Classification Algorithms

The classification algorithms tested were drawn from the M2K toolkit [10] and Weka [17] and include: Fisher's Criterion Linear Discriminant Analysis (LDA) [18], Classification and Regression Trees (CART) [19], a first-order linear Support Vector Machine (based on John C. Platt's Sequential Minimal Optimisation, SMO, algorithm [20]) and the J48 decision tree algorithm [17].

5.3 Classification Results

The results of the artist-filtered and unfiltered classification experiments are given in Tables 5 and 6, respectively. For each classification method, the highest-performing feature set is highlighted in bold.

Classifier Feature set	CART	J48	LDA	Linear SMO
M2K	**36.43%**	**36.79%**	35.70%	45.36%
Matlab	34.24%	35.15%	37.16%	46.63%
FPGA	34.24%	34.97%	**37.89%**	**49.00%**

Table 5. Artist-filtered classification results.

Classifier Feature set	CART	J48	LDA	Linear SMO
M2K	41.67%	**44.50%**	41.17%	**59.17%**
Matlab	38.83%	40.83%	40.67%	56.67%
FPGA	**42.67%**	39.83%	**41.33%**	57.50%

Table 6. Cross-validated classification results.

6. DISCUSSION AND FUTURE WORK

The results above demonstrate that the implementation of acoustic feature computation in hardware can potentially reduce computation times by orders of magnitude, accompanied, at worst, by a nearly negligible decrease in classification accuracy. This initial implementation demonstrates a potential pathway for migrating MATLAB feature extraction code to the Xilinx Blockset in Simulink and ultimately to hardware. The current implementation, however, does not allow for easy deployment on FPGA hardware and additional challenges lie in integrating FPGA-based feature computation in a full MIR evaluation system.

Our next step is to create a custom platform for full hardware deployment in a standalone system. In this system, based on the Virtex-II Pro development board, the local computer will communicate with the FPGA board via gigabit ethernet. The onboard hardware PowerPC cores will ease development for the standalone platform by allowing us to write C code to manage the communication link with a host PC and the data flow in and out of the feature extraction logic.

Such a custom FPGA platform would allow algorithms to be quickly designed and tested in Simulink, and then compiled into Verilog code to be synthesized into the larger project. Given a full deployment system, it will be possible to run the system at much higher clock speeds than in the Simulink simulation and hardware-in-the-loop co-simulation. The fully deployed system will ultimately be a massively parallel system where multiple analysis windows are processed simultaneously.

7. ACKNOWLEDGEMENTS

The authors would like to thank the International Music Information Retrieval Systems Evaluation Lab (IMIRSEL) at the University of Illinois at Urbana Champaign for their help in implementing the experiments reported. This work is partially supported by National Science Foundation grant IIS-0644151.

8. REFERENCES

[1] J.S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[2] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-28, No. 4:357–366, August 1980.

[3] J. C. Wang, J. F. Wang, and Y. S. Weng. Chip design of MFCC extraction for speech recognition. *Integration*, 32(1-2):111–131, Jan 2002.

[4] M. Nilsson and K. K. Paliwal. Speaker verification in software and hardware. *Microelectronic Engineering Research Conference*, 2001.

[5] W. Han, C. Chan, C. Choy, and K. Pun. An efficient MFCC extraction method in speech recognition. *Proceedings 2006 IEEE International Symposium on Circuits and Systems (ISCAS)*, page 4, Jan 2006.

[6] S. Nedevschi, R. Patra, and E. Brewer. Hardware speech recognition for user interfaces in low cost, low power devices. *Design Automation Conference, 2005. Proceedings. 42nd*, pages 684 – 689, May 2005.

[7] S. Sigurdsson, K. B. Petersen, and T. Lehn-Schiøler. Mel frequency cepstral coefficients: An evaluation of robustness of MP3 encoded music. In *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR)*, 2006.

[8] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the First International Symposium on Music Information Retrieval (ISMIR)*, October 2000.

[9] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE Transactions on*, 10(5):293–302, 2002.

[10] J. S. Downie, A. F. Ehmann, and D. Tcheng. Music-to-knowledge (M2K): a prototyping and evaluation environment for music information retrieval research. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 676–676, New York, NY, USA, 2005. ACM.

[11] J. Buckman. Magnatune: Mp3 music and music licensing, April 2006.

[12] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Johannes Kepler University, Linz, March 2006.

[13] K. West. *Novel techniques for Audio Music Classification and Search*. PhD thesis, School of Computing Sciences, University of East Anglia, Norwich, United Kingdom, September 2008.

[14] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, August 2001.

[15] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of ISMIR 2005 Sixth International Conference on Music Information Retrieval*, 2005.

[16] J.-J Aucouturier. *Ten Experiments on the Modeling of Polyphonic Timbre*. PhD thesis, University of Paris 6, France, June 2006.

[17] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. *ICONIP/ANZIIS/ANNES*, pages 192–196, 1999.

[18] K. West and S. Cox. Features and classifiers for the automatic classification of musical audio signals. In *Proceedings of ISMIR 2004 Fifth International Conference on Music Information Retrieval*, 2004.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks/Cole Advanced books and Software, 1984.

[20] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods: Support Vector Learning*, 1999.

ACCELERATING QUERY-BY-HUMMING ON GPU

Pascal Ferraro
LaBRI - U. Bordeaux 1, France
PIMS/CNRS - U. Calgary, Canada
ferraro@cpsc.ucalgary.ca

Pierre Hanna
LaBRI - U. Bordeaux 1,
France
pierre.hanna@labri.fr

Laurent Imbert
Lirmm - CNRS, France
PIMS/CNRS - U. Calgary, Canada
laurent.imbert@lirmm.fr

Thomas Izard
Lirmm - U. Montpellier 2,
France
thomas.izard@lirmm.fr

ABSTRACT

Searching for similarities in large musical databases has become a common procedure. Local alignment methods, based on dynamic programming, explore all the possible matchings between two musical pieces; and as a result return the optimal local alignment. Unfortunately these very powerful methods have a very high computational cost. The exponential growth of musical databases makes exact alignment algorithm unrealistic for searching similarities. Alternatives have been proposed in bioinformatics either by using heuristics or by developing faster implementation of exact algorithm. The main motivation of this work is to exploit the huge computational power of commonly available graphic cards to develop high performance solutions for Query-by-Humming applications. In this paper, we present a fast implementation of a local alignment method, which allows to retrieve a hummed query in a database of MIDI files, with good accuracy, in a time up to 160 times faster than other comparable systems.

1. INTRODUCTION

One of the main goal of music retrieval systems is to find musical pieces in large databases given a description or an example. These systems compute a numeric score on how well a query matches each piece of the database and rank the music pieces according to this score. Computing such a degree of resemblance between two pieces of music is a difficult problem. Three families of methodologies have been proposed [1]. Approaches based on index terms generally consider N-grams techniques [2,3], which count the number of common distinct terms between the query and a potential answer. Geometric algorithms [4–6] consider geometric representations of music and compute distances between objects. Techniques based on string matching [7] are generally more accurate as they can take into account errors in the query or in the pieces of music of the database. This property is of major importance in the context of music retrieval systems since audio analysis always induces approximations. Moreover, some music retrieval application require specific robustness. Query by humming (QbH), a music retrieval system where the input query is a user-hummed melody, is a very good example. Since the sung query can be transposed, played faster or slower, without degrading the melody, retrieval systems have to be both transposition and tempo invariant. Edit distance algorithms, mainly developed in the context of DNA sequence recognition, have been adapted in the context of music similarity [8]. These algorithms, based on the dynamic programming principle, are generalisations of a local sequence alignment method proposed by Smith and Waterman [9] in the early 80's. Applications relying on local alignment are numerous and include cover detection [10], melody retrieval [8], Query-by-Humming [11], Query-by-Tapping [12], structural analysis, comparison of chord progressions [13], etc. Local alignment approaches usually provide very accurate results as shown at the recent editions of the Music Information Retrieval Evaluation eXchange (MIREX) [14].

Alignment algorithms are powerful and optimal: they always find the best alignment. However, they are also very time consuming. This drawback considerably limits their use for musical applications. For biological applications, heuristics such as BLAST and FASTA can be used to speed-up local sequence alignment while allowing for multiple regions of local similarity. These heuristics are valuable, but they may fail to report hits or may report false positives. In order to get more accurate results faster implementations of exact alignment algorithms are therefore of primary importance.

Graphics Processing Units (GPUs) have recently received lots of attention thanks to their extensive computing resources. Not only are the latest generations of GPUs very powerful graphic engines, they can also be used for General Purpose computation (GPGPU) [15]. With the recent evolutions of GPUs' architecture into a unified, highly parallel programmable processor, and the development of programming tools and high-level programming languages such as NVIDIA's CUDA, GPUs have become a very attractive, low-cost alternative to the traditional microprocessors for computationally demanding applications that can be expressed as data-parallel computations, *i.e.* the same program is executed on many data elements in parallel. This type of parallelism is well suited to the problem of QbH on very large scale music databases, although it also brings new challenges regarding memory operations and computational resource allocations. In this paper, we present an implementation of a variant of Smith-Waterman based on local transpositions which illustrates the advantages of recent graphic cards as computation platforms.

In Section 2 we present the general concepts of sequence alignment and a variant based on local transpositions well suited to musical applications. Our parallel implementation on GPU is detailed in Section 3. Several tests and comparisons are presented in Section 4.

2. ALIGNING TWO MUSIC

In this section, we briefly present the QbH system experimented. Following Mongeau and Sankoff [8], any monophonic piece can be represented by a sequence of notes, each given as a pair *(pitch, length).* Several alphabets and sets of numbers have been proposed to represent pitches and durations [3]. In the following, we are using the interval relative representation, *i.e.* the number of semitones between two successive notes reduced modulo 12. In the context of QbH applications, this representation presents the huge advantage to be transposition invariant.

2.1 General Sequence Alignment

Sequence alignment algorithms are widely used to compare strings. They evaluate the similarity between two strings t and q given on an alphabet A, and of respective sizes $|t|$ and $|q|$. Formally an alignment between t and q is a string z on the alphabet of pairs of letters, more precisely on $(A \cup \{\epsilon\}) \times (A \cup \{\epsilon\})$, whose projection on the first component is t and the projection on the second component is q. The letter ϵ does not belong to the alphabet A. It is often substituted by the symbol "-" and is called a *gap*. An aligned pair of z of type (a, b) with $a, b \in A$ denotes the *substitution* of the letter a by the letter b. A pair of type $(a, \text{-})$ denotes a *deletion* of the letter a. Finally, an aligned pair of type $(\text{-}, b)$ denotes the *insertion* of the letter b. A score $\sigma(t_i, q_j)$ is assigned to each pair (t_i, q_j) of the alignment. The score S of an alignment is then defined as the sum of the costs of its aligned pairs. Computational approaches to sequence alignment generally fall into two categories: *global alignments* and *local alignments*. Calculating a global alignment is a form of global optimization that forces the alignment to span the entire length of all query sequences. By contrast, local alignments identify regions of similarity within long sequences that are often widely divergent overall. In Query-by-Humming applications, since the query is generally much shorter than the reference, one favours local alignment methods.

Both alignment techniques are based on dynamic programming [9, 16, 17]. Given two strings t and q, alignment algorithms compute a $(|t| + 1) \times (|q| + 1)$ matrix T such that:

$$T[i, j] = S(t[0 \dots i], q[0 \dots j]),$$

where $S(t[0 \dots i], q[0 \dots j])$ is the optimal score between the subsequences of t and q ending respectively in position $0 \le i \le |t|$ and $0 \le j \le |q|$. Dynamic programming algorithms can compute the optimal alignment (either global or local) and the corresponding score in time $\mathcal{O}(|t| \times |q|)$ and memory $\mathcal{O}(\min\{|t|, |q|\})$ (see [9] for details).

2.2 Local Transposition

Queries produced by human beings can, not only be totally transposed, but can also be composed of several parts that are independently transposed. For example, if the original musical piece is composed of different harmonic voices, the user may sing different successive parts with different keys. In the same way, pieces of popular music are sometimes composed of different choruses sung based on different tonic. A sung query may imitate these characteristics. Moreover, errors in singing or humming may occur, especially for users that are not trained to perfectly control their voice like professional singers. From a musical point of view, sudden tonal changes are disturbing. However, if these changes last during a long period, they may not disturb listeners. Figure 1 shows an example of query having two local transpositions.

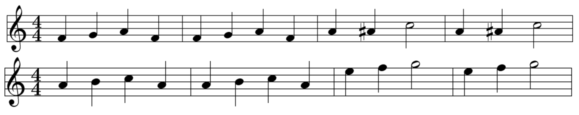


Figure 1. Example of a monophonic query not transposed (top) and a monophonic query with two local transpositions (bottom).

The two pieces in Figure 1 sound very similar, although the two resulting sequences are very different. This problem has been addressed in [18] by defining a local transposition algorithm. It requires to compute multiple score matrices simultaneously, one for each possible transposition value. The time complexity is $\mathcal{O}(\Delta \times |q| \times |t|)$, where Δ is the number of local transposition allowed during the comparison (for practical applications, Δ is set up to 12). Our experiments, presented in section 4, show that the local transposition algorithm provides a much better result.

2.3 Pitch/Duration Scoring Scheme

The quality of an alignment-based algorithm heavily depends on the scoring function. Results may differ significantly whether one uses a basic scoring scheme or a more sophisticated scoring function [7]. For our experiments, we use the scoring schemes introduced in [8] and [7], where the score between two notes depends on the pitch, the duration and the consonance of both notes. For example, the fifth (7 semitones) and the third major or minor (3 or

4 semitones) are the most consonant intervals in Western music [19]. The score function between two notes is then defined as a linear combination of a function σ_p on pitches (its values are coded into a matrix) and a function σ_d on durations as:

$$\sigma(a, b) = \alpha \cdot \sigma_p(a, b) + \beta \cdot \sigma_d(a, b).$$

The cost associated to a gap only depends on the note duration. Finally a penalty (a negative score) is also applied to each local transposition.

3. PARALLEL IMPLEMENTATION

3.1 GPU Architecture

Both AMD and NVIDIA build architectures with unified, massively parallel programmable units, which allow programmers to target that programmable unit directly instead of dividing work across multiple hardware units. More precisely, a GPU contains many streaming multiprocessors (MPs) each containing several elements including several cores, also called streaming processors (SPs), and various types of on-chip shared memories and registers. The MPs also share some constant memory areas with very fast access and a global uncached large memory with relatively low throughput and long latency. For example, the NVIDIA GeForce 9800 GX2 used for our experiments (see Section 4) is a dual GPU engine with 256 cores (128 per GPU) running at 1.5 GHz. These cores are regrouped into 2×16 MPs which share a global memory of 1GB with a 512-bit interface width providing a throughput of 128 GB/sec (64 GB/sec per GPU).

The MPs creates, manages, and executes concurrent threads in hardware with zero scheduling overhead. To manage hundred of threads running several different programs, the multiprocessor employs an architecture called SIMT (single-instruction multiple-thread), which resembles SIMD (single-instruction multiple-data) vector organizations, *i.e.*, single instruction controls multiple processing elements. Unlike SIMD vector machines, SIMT enables programmers to write thread-level parallel code for independent, scalar threads, as well as data-parallel code for coordinated threads.

3.2 GPU Computing with CUDA

Our implementation uses CUDA, a general purpose parallel computing framework developed and distributed by NVIDIA for use with their recent GPUs [1]. CUDA can be seen as an extension of C that allows developers to define C functions, called *kernels* to be executed N times in parallel by N different CUDA *threads*. CUDA threads may access data from multiple memory spaces during their execution. CUDA's programming model assumes that the CUDA threads execute on a separate *device*, whereas the rest of the program runs on a CPU. In other words, the GPU operates as a coprocessor to the *host* running the C program. Both the host and device maintain their own memory areas, allowing for concurrent programming between the CPU and the GPU(s). CUDA kernels must be compiled into binary code using `nvcc`, a C compiler for CUDA. Note that `nvcc` supports C++ programming for host functions but kernels must be written in C, possibly with templates. `nvcc` also supports device emulation.

[1] CUDA was introduced in November 2006 along with the G80 series. CUDA can be downloaded for free from `http://www.nvidia.com/object/cuda_home.html`. A list of CUDA-enabled product is available at `http://www.nvidia.com/object/cuda_learn_products.html`.

3.3 CUDA implementation of QbH

The process of evaluating how well each piece of music in a database match a query (sung or hummed in the case of QbH), and rank the music pieces according to this score can be parallelized at different levels. As explained earlier, our implementation uses a variant of Smith-Waterman. Although it is possible to do so [20], our choice was not to parallelize the implementation of Smith-Waterman itself, as this approach could only provide significant improvements for extremely large sequences. In contrast, our CUDA implementation optimizes the arithmetic intensity (the ratio of arithmetic operations to memory operations) by computing in parallel all the scores of a query with every piece of music in the database. If the database contains N pieces of music, our program virtually launches N kernels executing the Smith-Waterman algorithm in parallel. The main challenges are therefore to optimize the resource allocations and the memory operations.

After the query has been converted from its original format (typically a wave audio file), we store it in a special memory area called the *texture* memory, which allows for very fast read/write operations. The texture memory is shared among all threads (Fig. 2). The database usually contain too many pieces of music to be stored in any of the cached, fast memories (texture, constant, shared memory). Therefore, all the pieces of music are stored in the global memory. On a GPU, the global memory is not cached, so it is extremely important to follow the right access pattern to get maximum memory bandwidth. Throughput of memory operations is 8 operations per clock cycle, plus 400 to 600 clock cycles of memory latency. Under some size and alignment conditions, the device is capable of reading data from global memory in a single load instruction. Moreover, the memory bandwidth can be used most efficiently when the simultaneous memory access by all the active threads can be coalesced into a single memory transaction. (For more details, see [21].)

In order to satisfy all these constraints, the pieces of music of the database are store in an array of `float2`, a CUDA structure containing two 32-bit floats, which stores the pitch and duration of each note. Although the size and alignment properties are fulfilled by this data type, storing the pieces of music sequentially, one after the other, would be very inefficient since simultaneous reading by all the threads in a single transaction would be impossible. Instead, if the database contain N pieces of music, we organize the data in memory as a one dimensional array, such that its first N entries correspond to the first note (pitch, duration) of each piece; then, the next N entries correspond to the second note of each piece, etc.

Each thread performs its computations on its own matrix, more exactly on its $\Delta = 12$ transposition matrices. In order to minimize the amount of required memory, we only store the current row of each matrix. Moreover, to optimize memory alignment, allocation is based on the query's fixed size rather than the pieces of music's variable sizes. Finally, in order to allow simultaneous read/write operations by the active threads, the matrices are not stored at consecutive addresses but rather using the same strategy as the database. Fig. 2 describes the device architectures.

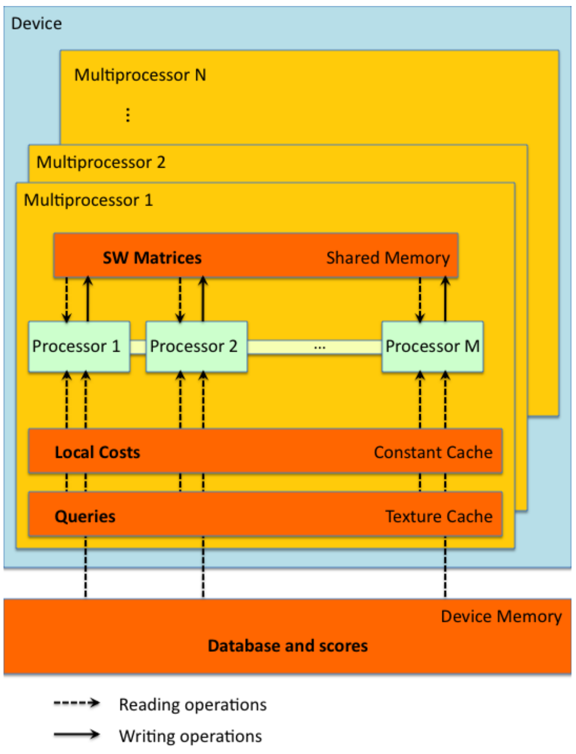


Figure 2. nVidia GPUs architecture. Each multi-processor executes both the conversion of queries from audio files to a vector of notes (stored in the texture memory) and the comparison between the query and each reference (stored in the device memory). Each processor store its intermediate Smith-Waterman matrices (only one row) in its own shared memory space. Constants costs and intermediate values are respectively stored in constant and shared memories.

4. TESTS AND RESULTS

4.1 Benchmark

Our experiments are based on the query data corpus proposed for the QbH tasks at the MIREX 2007 and 2008 and three different noise databases. Roger Jang's corpus is composed of 2797 queries, along with 48 ground-truth MIDI files[2], with the particularity that all queries start at the beginning of the references. The first database (called DB1) consists of the 48 ground-truth MIDIs and a subset of 2000 MIDI noise files from the Essen Collection[3]. The whole Essen Collection, made of 5982 files together with the 48 ground-truth MIDIs files, is called DB2. Finally, since the ground-truth MIDIs are rather short while Essen collection mainly consists of long data files, we also consider a third database, called DB3, proposed during the MIREX 2005, which is a subset of the RISM A/II (International inventory of musical sources) collection, composed of 17433 short excerpt of real world compositions.

We have tested our implementation on three different platforms. Their characteristics are given in Table 1. For each algorithm we have measured the time on both the CPU alone and the CPU together with the GPU used as a parallel coprocessor.

	OS	CPU	GPU
W1	NVidia Tesla Workstation running Linux, CUDA v2.1	3GHz Intel Core 2 Duo	NVidia GeForce 9800 GX2, 512 MB memory
W2	Mac Pro running Mac OS X 10.5, CUDA v2.2	Two 2.8GHz Intel Xeon Quad Core	NVidia GeForce 8800 GT, 512 MB Memory
L1	MacBook Pro running Mac OS X 10.5, CUDA v2.2	2.53 GHz Intel Core 2 Duo	NVidia GeForce 9400 M, shared memory with CPU

Table 1. Characteristics of our three platforms

Regarding the algorithms, we have implemented the original Smith-Waterman algorithm (SW) and our extension based on local transposition alignment (LT). Since the queries are known to be at the beginning of the references, we have implemented variants of the above algorithms that only compare the query with the beginning of each MIDI files (in Table 2 we only report timings for size of the query plus 10 notes). These variants are respectively called SW10 and LT10.

We evaluate the quality of our music retrieval system using two measures. The Mean Reciprocal Rank (MRR), *i.e.* the average of the reciprocal ranks of the first correct answer, computed for a sample of N queries as

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r_i},$$

where r_i is the rank of the first correct answer for the ith query. And the top-X ratio which reports the proportion of queries for which $r_i \leq X$.

4.2 Smith-Waterman vs Local Transposition Alignment

We first evaluate the quality, given in terms of MRR and top-5 ratio, of SW and LT on the three databases. For

[2] `http://www.cs.nthu.edu.tw/~jang`

[3] `http://www.esac-data.org/`

DB1, SW reaches a MRR of 0.274 and a top-5 ratio of 30.3%, while LT reaches a MRR of 0.684 and a top-5 ratio of 75.4%. The MRR increases when the size of the query is taken into account during the comparison: SW10 reaches a MRR of 0.295 and a top-5 ratio of 32.3%, while LT10 reaches a MRR of 0.732 and a top-5 ratio of 79.4%. We observe the same behaviour for DB2 and DB3. Fig. 3 shows that the MRRs obtained for databases of different sizes remains roughly the same. The local transposition alignment method provides better results than Smith-Waterman.

For a suggestive comparison, the method submitted by Wu and Li, which performed best at MIREX 2008 reaches a MRR of 0.9 on a well choosen subset of 2000 MIDI files from the Essen collection. However, since we could not find the original database used for the MIREX 2008 competition, our DB1 consists of 2000 randomly chosen MIDI files from the Essen collection. It therefore contains several copies of the same noise files, which automatically impacts the quality of our results.

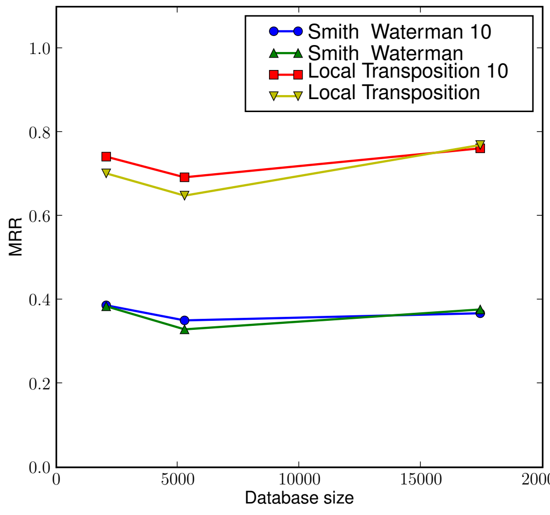


Figure 3. MRRs obtained for SW, SW10, LT, LT10 on our three databases

4.3 CPU vs GPU

Although LT provides very good results in terms of quality, it is very time consuming. On our fastest CPU (W2), a Mac Pro equipped with two 2.8 GHz Intel Xeon Quad Core processors, the analysis on DB1 takes more than 326 minutes with LT10 ($\sim$ 7 sec. per query) and more than 595 minutes with LT ($\sim$ 13 sec. per query). This computation time even reaches 1282 minutes for LT10 ($\sim$ 27.5 sec. per query) on the largest database DB3, which contains more than 17000 MIDI files.

As shown in Table 2, our CUDA implementations provide impressive speed-ups. The local transposition algorithms (LT and LT10) which gives the highest MRRs and top-X ratios, are up to 162 times faster than their CPU counterpart. This is achieved for DB3 on our Mac Pro configuration W2; the analysis of more than 17000 MIDI files using LT10 was completed in 473 seconds ($\sim$ 0.16 s per query). Note also that the local transposition algorithm is perfectly adapted to parallel implementations as it is only slightly slower than SW.

It is important to remark that our CUDA implementation leads to significant improvements even on our lightest configuration (L1), a laptop not designed to perform heavy graphic computations and which embeds a cheap, on-chip graphic card (NVidia 9400 M). The analysis of DB1 only takes 470 seconds, *i.e.* $\sim$ 0.16 s per query. During the MIREX 2008, the fastest implementation for the analysis of a database similar to DB1 was performed in 1699 seconds on a AMD Athlon XP 2600+ running at 1.9GHz. Our fastest implementation, running on W1, completed the analysis of the 2797 queries in only 301 seconds, that is almost 6 times faster.

	Config.	SW10	SW	LT10	LT
DB1	L1	7:33	7:33	7:50	40:54
	W1	4:12	5:05	5:01	20:16
	W2	6:00	6:00	6:01	14:42
DB2	L1	7:53	7:51	15:10	79:45
	W1	6:55	6:54	6:10	25:46
	W2	6:18	6:18	6:16	20:40
DB3	L1	9:20	9:19	41:31	44:48
	W1	5:15	6:05	10:09	25:27
	W2	7:25	7:27	7:53	21:15

Table 2. Timings of the different algorithms on various GPUs and databases in mm:ss

5. CONCLUSIONS

Local transposition alignment algorithms are very powerful. Using QbH as an experimental application, our variant of Smith-Waterman lead to very good results. We believe that this type of algorithm would give even better results for other music retrieval systems, such as cover detection, where the query is significantly larger and contains fewer errors than a sung or hummed query. Our implementation takes advantage of the immense computing resources offered by the most recent graphic cards. These low-cost devices regroup hundreds of cores that can operate in parallel and sufficient memory to store large musical databases. A great care must be taken when programming memory operations as a bad allocation strategy can have a significant impact on the computation time. At this time, we have not yet optimized the pre-processing phase of the system. In particular, the analysis and conversion of the queries (wave audio files) is running exclusively on the CPU and takes between 75-90% of the overall computation time. Our next task will be to implement this stage on GPU using the CUDA CUFFT library. We anticipate significant improvements in terms of speed.

6. ACKNOWLEGMENT

This work has been partially sponsored by the French ANR SIMBALS (JC07-188930) and ANR Brasero (ANR-06-BLAN-0045) projects.

7. REFERENCES

[1] N. Orio. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1(1):1–90, 2006.

[2] S. Doraisamy and S. Rüger. Robust polyphonic music retrieval with N-grams. *Journal of Intelligent Information Systems*, 21(1):53–70, 2003.

[3] A. L. Uitdenbogerd. *Music Information Retrieval Technology*. PhD thesis, RMIT University, Melbourne, Victoria, Australia, July 2002.

[4] E. Ukkonen, K. Lemström, and V. Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proceedings of the 4th International Conference on Music Information Retrieval, ISMIR 2003*, pages 193–199, 2003.

[5] R. Typke, R. C. Veltkamp, and F. Wiering. Searching notated polyphonic music using transportation distances. In *Proceedings of the ACM Multimedia Conference*, pages 128–135, 2004.

[6] R. Typke and A. Walczak-Typke. A tunneling-vantage indexing method for non-metrics. In *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008*, pages 351–352, 2008.

[7] P. Hanna, P. Ferraro, and M. Robine. On optimizing the editing algorithms for evaluating similarity between monophonic musical sequences. *Journal of New Music Research*, 36(4):267–279, 2007.

[8] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[9] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[10] J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16:1138–1151, 2008.

[11] R. B. Dannenberg, W. P. Birmingham, B. Pardo, N. Hu, C. Meek, and G. Tzanetakis. A comparative evaluation of search techniques for query-by-humming using the MUSART testbed. *Journal of the American Society for Information Science and Technology*, 58(5):687–701, 2007.

[12] P. Hanna and M. Robine. Query by tapping system based on alignment algorithm. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 2009. (to appear).

[13] J. P. Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007*, pages 239–244, September 2007.

[14] J. S. Downie, M. Bay, A. F. Ehmann, and M. C. Jones. Audio cover song identification: MIREX 2006-2007 results and analyses. In *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008*, pages 51–56, 2008.

[15] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[16] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[17] D. Gusfield. *Algorithms on Strings, Trees and Sequences – Computer Science and Computational Biology*. Cambridge University Press, Cambridge, 1997.

[18] J. Allali, P. Ferraro, P. Hanna, and C. Iliopoulos. Local transpositions in alignment of polyphonic musical sequences. In *String Processing and Information Retrieval Symposium, 14th International Symposium, SPIRE 2007, Proceedings*, volume 4726 of *Lecture Notes in Computer Science*, pages 26–38. Springer, October 2007.

[19] F. J. Horwood. *The Basis of Music*. Gordon V. Thompson Limited, Toronto, Canada, 1944.

[20] Y. Liu, W. Huang, J. Johnson, and S. Vaidya. GPU accelerated Smith-Waterman. In *Computational Science, ICCS 2006*, volume 3994 of *Lecture Notes in Computer Science*, pages 188–195. Springer, 2006.

[21] NVIDIA CUDA. *Programming Guide*, April 2009. Version 2.2. Available at `http://www.nvidia.com/object/cuda_home.html`.

LEARNING TO CONTROL A REVERBERATOR USING SUBJECTIVE PERCEPTUAL DESCRIPTORS

Zafar Rafii
EECS Department
Northwestern University
Evanston, IL, USA
ZafarRafii2011@u.northwestern.edu

Bryan Pardo
EECS Department
Northwestern University
Evanston, IL, USA
pardo@northwestern.edu

ABSTRACT

The complexity of existing tools for mastering audio can be daunting. Moreover, many people think about sound in individualistic terms (such as "boomy") that may not have clear mappings onto the controls of existing audio tools. We propose learning to map subjective audio descriptors, such as "boomy", onto measures of signal properties in order to build a simple controller that manipulates an audio reverberator in terms of a chosen descriptor. For example, "make the sound less boomy". In the learning process, a user is presented with a series of sounds altered in different ways by a reverberator and asked to rate how well each sound represents the audio concept. The system correlates these ratings with reverberator parameters to build a controller that manipulates reverberation in the user's terms. In this paper, we focus on developing the mapping between reverberator controls, measures of qualities of reverberation and user ratings. Results on 22 subjects show the system learns quickly (under 3 minutes of training per concept), predicts users responses well (mean correlation coefficient of system predictiveness 0.75) and meets users' expectations (average human rating of 7.4 out of 10).

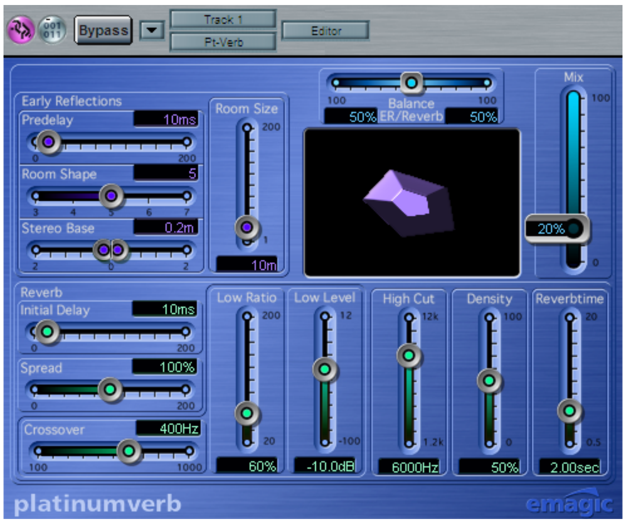
1. INTRODUCTION

In recent decades, many audio production tools have been introduced to enhance and facilitate music creation. Often, these tools are complex and conceptualized in terms ("high cut","density") that are unfamiliar to many users. This makes learning these tools daunting, especially for inexperienced users.

One solution would be to redesign the standard interfaces to manipulate audio in terms of commonly used descriptors (e.g. "warm" or "enveloping"). This can be problematic, since the meanings of many words used to describe sound differ from person to person or between different groups [1]. For example, the audio signal properties associated with "warm" and "clear" have been shown to vary between English speakers from the UK and the US [2]. Since it may not be possible to create general controllers for terms whose meaning varies between groups, we propose mapping descriptive terms onto the controls for audio tools on a case-by-case basis.

While there has been much work on adaptive user interfaces [3], there has been relatively little on personalization of audio tools. A previous study showed success in personalizing an equalization tool [4]. Here, we propose to simplify and personalize the interface to one of the most widely applied classes of audio effect: *Reverberation*.

Reverberation is created by the reflections of a sound in an enclosed space causing a large number of echoes to build up and then slowly decay as the sound is absorbed by the walls and air [5]. The reflections modify the perception of the sound, its loudness, timbre and spatial characteristics [6]. Reverberation can be simulated using multiple feedback delay circuits to create a large, decaying series of "echoes" [7], and many reverberation tools have been built. Fig. 1 shows the interface of a typical reverberation tool. Note the 7 buttons and 14 sliders that control parameters (such as "density") whose meaning in this context are unfamiliar to the average person and to many musicians.



Predelay 10ms
Room Shape 5
Stereo Base 0.2m
Room Size 10m
Balance ER/Reverb 50% 50%
Mix 20%
Initial Delay 10ms
Spread 100%
Crossover 400Hz
Low Ratio 60%
Low Level -10.0dB
High Cut 6000Hz
Density 50%
Reverbtime 2.00sec

Figure 1. *Logic Audio*'s *Platinumverb* complex interface.

We propose a system that learns an audio concept a user has in mind ("boomy", for example) and builds a simple reverberation controller to manipulate sound in terms of that descriptor. By automatically adapting the interface to an individual user's conceptual space, we hope to bypass the creative bottleneck caused by complex interfaces and individual differences in the meaning of descriptive terms.

The paper is organized as follows. The method used to map descriptive terms onto audio signal characteristics is described in Section 2. Section 3 presents the reverberation control used to perceptually alter sound. Experimental evaluation of the approach is described in Section 4. Finally, conclusions are given in Section 5.

2. LEARNING DESCRIPTIVE TERMS

We now give an overview of the process by which the system learns to build a controller that controls the reverberation of a signal in terms of a user-defined descriptive word.

2.1 The Training Process

In the training process, the user is presented with the *Perceptual Learner* interface shown in Fig. 2. The user selects a descriptive word (such as "boomy" or "church-like") to teach the system. The user is then presented with a series of audio examples generated from an original audio file and processed by the reverberator using a variety of reverberation settings. The reverberation settings used are chosen to explore the space of likely parameter settings for a digital reverberator, as described in Section 4.

The user moves a slider to rate each audio example on how well it represents the audio descriptor. Ratings range from 1 (captures the concept perfectly) to -1 (does not capture it at all). Training typically takes about 30 ratings (around two minutes for a five-second file). Fig. 3 illustrates the process.

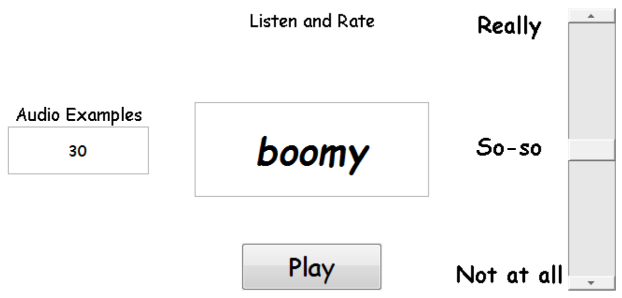


Figure 2. Interface of the Perceptual Learner.

2.2 Mapping Signal Statistics to User Ratings

The system collects five impulse response measures (described in Section 3.2) for the reverberation applied to each example rated by the user. Once user ratings are collected, the system relates user ratings to each of the five measures using *linear regression*. This lets us build a model that predicts the expected user rating, given a reverberation impulse response signal characterized by these measures.

This mapping is used to build a controller that lets the user easily manipulate the audio in terms of the descriptor

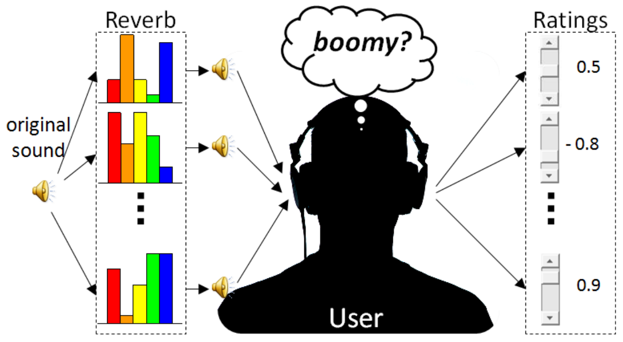


Figure 3. The training process: (1) audio examples are generated from an original sound using a reverberator set to a variety of parameter settings (5 control parameters shown by 5 different bars); (2) the user listen to the audio examples and uses a slider to rate how well each one fits the audio concept she/he has in mind.

(such as "boomy") using a simple slider as shown in Fig. 4. This slider affects all five reverberation measures in parallel, although not necessarily in the same direction. For example, "boomy" may be positively correlated with central time and negatively correlated with spectral centroid.

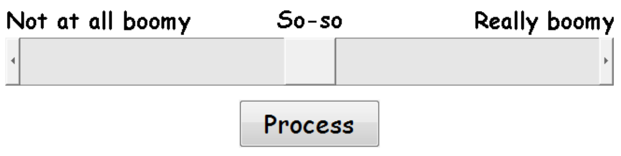


Figure 4. Interface of the Perceptual Controller.

3. THE REVERBERATION CONTROL

To build the new interface, we must map human feedback to reverberation controls. We do not, however, map user feedback directly to parameters for a specific reverberator, but onto measures of the reverberation (Section 3.2). This lets us use mappings learned using one reverberator to control another one, chosen later. The only requirement is that both reverberators have known mappings between control parameters and reverberation measures.

3.1 The Digital Reverberator

The approach we describe, while not tied to any particular reverberation approach, works best if the reverberator can generate a wide variety of impulse response functions on the fly. Thus, rather than use a convolution reverberator that selects from a fixed library of impulse responses, we have developed a digital stereo reverberation unit inspired by Moorer's work [8]. The reverberator, shown in Fig. 5, is easy to manipulate through the control parameters. The reverberation measures described in Section 3.2 can be derived easily as functions of those parameters. This is important for learning a mapping between human feedback and reverberator settings.

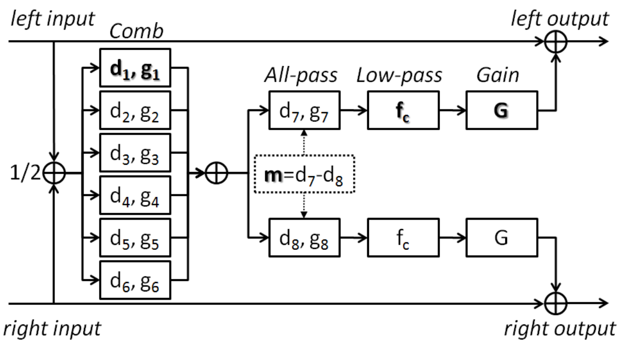


Figure 5. The digital stereo reverberation unit.

The reverberator uses six *comb* filters in parallel to simulate the complex modal response of a room by adding echoes together. Each comb filter is characterized by a delay factor d_k and a gain factor g_k (k=1..6). The delay values are distributed linearly over a ratio of 1:1.5 with a range between 10 and 100 msec, so that the delay of the first comb filter d_1, defined as the longest one, determines the other delays. The gain factor of the first comb filter g_1 has the smallest gain and has a range of values between 0 and 1. Although a comb filter gives a non-flat frequency response, a sufficient number of comb filters in parallel with equal values of *reverberation time* helps to reduce the spectral coloration.

An *all-pass* filter is added in series to increase the *echo density* produced by the comb filters without introducing spectral coloration, and doubled into two channels to simulate a more "natural sounding" reverberation in stereo. The all-pass filter is characterized by a delay factor d_a of 6 msec and a gain factor g_a fixed to $\frac{1}{\sqrt{2}}$. A small difference m is introduced between the delays to insure a difference between the channels, therefore the delays become $d_7 = d_a + \frac{m}{2}$ for the left channel and $d_8 = d_a - \frac{m}{2}$ for the right channel. The range of values for $m = d_7 - d_8$ is then defined between 0 and 12 msec. Note that to prevent exactly overlapping echoes, the delay values for the comb and the all-pass filters are set to the closest inferior prime number of samples.

To simulate air and walls absorption, a first-order low-pass filter of gain g_c defined from its cut-off frequency f_c is added at each channel [9]. f_c ranges between 0 and half of the frequency sampling f_s. Finally, a gain parameter G, whose range of values is between 0 and 1, controls the wet/dry effect. In summary, a total of only five independent parameters are needed to control the reverberator: $\boldsymbol{d}_1$, $\boldsymbol{g}_1$, $\boldsymbol{m}$, $\boldsymbol{f}_c$ and $\boldsymbol{G}$. The other parameters can be deduced from them according to the relations above.

3.2 The Reverberation Measures

We now define five measures commonly used to characterize reverberation and describe formulae to estimate values for these measures in terms of the parameters for our reverberator. For details on how we derive these formulae, we refer the reader to [10].

- *Reverberation Time* (T_{60}) is defined as the time in sec required for the reflections of a direct sound to decay by 60 dB below the level of the direct sound [5]. Based on the reverberation time of the comb filter and the other gains, we estimated the reverberation time of the whole reverberation unit as follows in Eq. 1.

$$T_{60} = \max_{k=1..6} \left(d_k \, \log \left(\frac{10^{-3}}{g_a \, (1-g_c) \, G} \right) / \log g_k \right) \quad (1)$$

- *Echo Density* (D_t) is defined as the number of echoes per second at a time t. In practice, we computed the average echoes per second between time 0 and time t. We estimated the echo density of the whole reverberation unit at time t = 100 msec, as a combination of echo densities of the digital filters, as follows in Eq. 2.

$$D = \frac{t}{d_a} \sum_{k=1}^{6} \frac{1}{d_k} \quad (2)$$

- *Clarity* (C_t) describes the ratio in dB of the energies in the impulse response p before and after a given time t. It provides indication of how "clear" the sound is [11]. The definition of C_t in discrete time is given by Eq. 3.

$$C_t = 10 \, \log_{10} \left(\sum_{n=0}^{t} p^2[n] / \sum_{n=t}^{\infty} p^2[n] \right) \quad (3)$$

We estimated the clarity of the whole reverberation unit at t = 0, the arrival time of the direct sound, as shown in Eq. 4, assuming that the total energy of the reverberator is a linear combination of the energies of its filters.

$$C = -10 \, \log_{10} \left(G^2 \, \frac{1-g_c}{1+g_c} \sum_{k=1}^{6} \frac{{g_k}^2}{1-{g_k}^2} \right) \quad (4)$$

- *Central Time* (T_C) is the "center of gravity" of the energy in the impulse response p, [11], defined in discrete time by Eq. 5.

$$T_C = \sum_{n=0}^{\infty} n p^2[n] / \sum_{n=0}^{\infty} p^2[n] \quad (5)$$

Based on the same assumption as for clarity, we estimated the central time of the whole reverberation unit as the combination of central times of the filters, as follows in Eq. 6.

$$T_C = \sum_{k=1}^{6} \frac{d_k {g_k}^2}{(1-{g_k}^2)^2} / \sum_{k=1}^{6} \frac{{g_k}^2}{1-{g_k}^2} \; + \; d_a \quad (6)$$

- *Spectral Centroid* (S_C) is the "center of gravity" of the energy in the magnitude spectrum P of the impulse response p, defined in discrete time by Eq. 7, where f_s is the sampling frequency.

$$S_C = \sum_{n=0}^{f_s/2} n P^2[n] / \sum_{n=0}^{f_s/2} P^2[n] \quad (7)$$

We estimated the spectral centroid of the whole reverberation unit from the characteristics of its low-pass filter, as follows in Eq. 8.

$$S_C = \frac{\sum_{n=0}^{f_s/2} \frac{n}{1 + {g_c}^2 - 2\, g_c \cos(2\pi n)}}{\sum_{n=0}^{f_s/2} \frac{1}{1 + {g_c}^2 - 2\, g_c \cos(2\pi n)}} \quad (8)$$

Based on the relations between the parameters defined in section 3.1, the measures can be redefined as five functions of five independent parameters: $\boldsymbol{T}_{60}(\boldsymbol{d}_1, \boldsymbol{g}_1, \boldsymbol{f}_c, \boldsymbol{G})$, $\boldsymbol{D}(\boldsymbol{d}_1, \boldsymbol{m})$, $\boldsymbol{C}(\boldsymbol{g}_1, \boldsymbol{f}_c, \boldsymbol{G})$, $\boldsymbol{T}_C(\boldsymbol{d}_1, \boldsymbol{g}_1, \boldsymbol{m})$ and $\boldsymbol{S}_C(\boldsymbol{f}_c)$.

Note that these functions are not entirely invertible, especially for d_1 and g_1. When necessary, we estimate d_1 and g_1 from a reverberation measure by using tables of values.

4. EVALUATION

We have implemented the system in Matlab on a PC with an Intel Core2 Quad CPU of 2.66GHz and 6GB of RAM. The system was evaluated by 22 participants, 14 males and 8 females, between the ages of 18 and 29. All reported normal hearing and were native English speakers. 10 had a little or no musical background and 12 had a strong musical background i.e. practicing one or several instruments, more than 1 hour per week and for more than 10 years, or more than 6 hours per week and for more than 6 years.

All audio examples created were based on a 5.5 sec anechoic recording of an unaccompanied singing male sampled at 44,100 Hz. Prior to the study, a database of 1024 impulse response functions was generated using the reverberator described in Section 3.1. These impulse response functions were selected to evenly cover a range of the five reverberation measures (Section 3.2). The *Reverberation Time* ranged from 0.5 to 8 sec, the *Echo Density* from 500 to 10,000 echoes/sec, the *Clarity* from -20 to 10 dB, the *Central Time* from 0.01 to 0.5 sec, and the *Spectral Centroid* from 200 to 11,025 Hz (no low-pass filtering). These ranges were chosen by audio inspection so that they evenly cover a range of "good" values in the space of reverberation measures leading to natural sounding reverberation.

4.1 Experiment

Study participants were seated in a quiet room with a computer that controlled the experiment and recorded the responses. The stimuli were presented binaurally over headphones. Participants were allowed to adjust the sound level prior to starting the study. Prior to beginning the study, participants were quickly trained on the task. Each participant participated in a single one-hour session.

Each participant was asked to rate the same five descriptive words: *bright*, *clear* (two common audio descriptors), *boomy* (often related to reverberation), *church-like* and *bathroom-like* (related to models of space, respectively a church and a bathroom). These words were presented to each participant in a random order. For each descriptive word, the participant was asked to perform three tasks.

First, the participant was asked to rate a series of 60 audio examples. For each example the participant heard the audio modified by an impulse response function. The participant moved an on-screen slider (Fig. 2) to indicate the extent to which each sound exemplified the current word descriptor. Values ranged from 1 (captures the concept perfectly) to -1 (does not capture it at all). These 60 audio examples contained 35 examples chosen randomly from our database of 1024 examples. We then duplicated 25 of the 35 and added the duplicates to the set in random order, for a total of 60 examples. The 25-example duplicate set was used to measure consistency of user responses, while the 35-example training set was for system training. A previous study showed that around 25 examples are sufficient to model a user's preferences for an equalization controller [4], which is a closely related task.

Once the first task was completed, the system created a model of the effect of each reverberation measure on the user ratings, as described in Section 2.2. The data set used was the user ratings of the 35 non-duplicate examples in the first task. The new model was used to select a new set of audio examples. This set contained 11 audio examples chosen to evenly cover the range of user ratings from -1 to 1 (as predicted by the learned model) and 14 audio examples selected at random, for a total of 25. The participant was asked to rate the 25 new audio examples as she/he did in the first part.

Finally, the system used the learned model to build a slider (Fig. 4) that controls reverberation in terms of the learned descriptor. The controller mapped 11 audio examples chosen to evenly cover the range of user ratings from -1 to 1 onto slider positions. As the slider is moved to a new location, a different variant of the sound is played. This let the participant move the slider to change the degree of the effect. The user was asked to play with the controller for as long as neccesary to get a feel for how well it worked. The user was then asked to rate how well it manipulated the sound in terms of the learned descriptive word. *Human ratings* ranged from 0 (really bad) to 10 (really good).

4.2 Results

The average training time, over all the descriptive words and the participants, was 4 min 2 sec. Since only 35 of the 60 user-ratings in the first task were actually used for training the system, a model for a descriptive word was learned in only 2 min 20 sec of training (the mean time for a user to rate for 35 examples).

User consistency on a descriptive word was measured by computing the within-user correlation coefficient on rating the 25 pairs of duplicate examples in the first task. Average *user consistency* over all words and users was 0.65.

System predictiveness (how well the system learned) for a descriptive word was measured by computing the correlation coefficient between the user's observed ratings and the system's prediction of the user ratings on the second set of user-rated examples. *System predictiveness* was 0.75, averaged over all words and users.

System predictiveness was measured on a different data set than user consistency, so the results are not directly comparable. That said, the consistency of user ratings on matched pairs of stimuli gives an indication that one might not be able to expect significantly better predictive power than that shown by our approach.

Average *human rating* over all words and users given to the final controller was 7.4 out of 10. This means that overall, the participants felt the system succeeded in providing a controller that lets the user manipulate the sound in terms of the descriptive words.

Mean correlation coefficients between user ratings and each of the five control measures (Section 3.2) used to generate the audio examples are shown in Tables 1 and 2. Table 1 shows values for the 10 participants with little or no musical background. Table 2 shows these values for the 12 participants with strong musical background.

	bright	*clear*	*boomy*	*bath'*	*church*
training time	4'07"	3'34"	4'35"	4'29"	4'57"
Reverb. Time	-0.02	0.19	-0.07	0.02	0.08
Echo Density	-0.01	-0.10	0.13	0.06	0.03
Clarity	**0.39**	**0.33**	0.08	0.06	0.16
Central Time	-0.45	-0.51	**0.28**	-0.22	**0.52**
Spec. Centroid	**0.36**	**0.38**	-0.23	0.08	0.02

Table 1. Average training time and correlation coefficients of the measures for the five descriptive words over the participants with little or no musical background.

	bright	*clear*	*boomy*	*bath'*	*church*
training time	3'14"	3'42"	4'17"	4'05"	3'36"
Reverb. Time	0.03	0.21	-0.04	-0.04	0.06
Echo Density	-0.02	-0.06	0.06	0.01	0.02
Clarity	0.29	**0.44**	0.14	-0.08	-0.03
Central Time	-0.17	-0.57	**0.46**	0.06	**0.70**
Spec. Centroid	**0.42**	0.29	-0.21	0.13	-0.03

Table 2. Average training time and correlation coefficients of the measures for the five descriptive words over the participants with strong musical background.

As we can see, participants with strong musical background completed the training more quickly. Both groups showed similar results for the *user consistency*, the *system predictiveness* and the *human ratings*. However there are relevant differences between these two groups in which signal measures most affect ratings of examples.

For both groups, *bright* and *clear* show overall a correlation with the *Clarity* and the *Spectral Centroid*, and a negative correlation with the *Central Time*. Table 1 indicates that participants with little or no musical background may have confounded *bright* and *clear*, while they seem distinct to people with strong musical background. Indeed, we should expect *bright* to be more correlated with the *Spectral Centroid* and *clear* with the *Clarity*, as shown in Table 2 by participants with strong musical background.

That said, *user consistency*, *system predictiveness* and *human ratings* are reasonably high on these words for both groups, even though the definitions of these words clearly vary between groups. These results indicate people with little musical experience can still define these terms with enough consistency for the system to model their preferences and provide a useful controller.

Boomy shows a significant correlation with the *Central Time* (in bold) and a negative correlation with the *Spectral Centroid*. Participants with strong musical background showed higher correlation with the *Central Time*. Furthermore, the distribution of the correlation coefficients of the measures for participants with strong musical background has a smaller standard deviation, which means that they showed a common understanding of the concept, while the standard deviation for participants with a little or no musical background is higher, especially for the *Central Time* and the *Spectral Centroid*, which means that the definition of the concept varied more greatly between them.

Table 3 highlights how well the system performs on a descriptive word where there was substantial disagreement between individuals. The table compares the correlation coefficients of the measures, the *system predictiveness* correlation coefficients, and the *human ratings* between four participants with little or no musical background for the descriptive word "boomy".

boomy	*user 11*	*user 12*	*user 13*	*user 22*
Reverb. Time	0.01	-0.04	-0.10	-0.18
Echo Density	**0.26**	-0.08	0.24	0.01
Clarity	-0.43	**0.10**	0.14	**0.36**
Central Time	-0.33	-0.17	**0.69**	*-0.32*
Spec. Centroid	*-0.74*	*-0.58*	*-0.15*	0.17
predictiveness	0.90	0.77	0.86	0.79
human ratings	7.0	10.0	8.0	8.0

Table 3. Comparison of the results between four participants with little or no musical background for *boomy* (the highest correlation coefficient is in bold and the highest negative correlation coefficient is in italic, for each user).

We can see that the correlation coefficients of the audio measures are very different from one participant to another, and yet, the *system predictiveness* and the *human ratings* are high. Again, this indicates our approach worked well to personalize a controller for each of these individuals, despite the variation in their personal definition of *boomy*.

Participants showed great variation in their responses to *bathroom-like* and distributions of the correlation coefficients between acoustic measures and user ratings show high standard deviation, especially for the *Clarity*, the *Central Time* and the *Spectral Centroid*. Table 4 compares the results for *bathroom-like* between four different participants: users 03 and 08 have a strong musical background, and users 12 and 13 have little or no musical background. Correlation coefficients of the measures are very different between participants, yet the *system predictiveness* and the *human ratings* are high.

Church-like shows overall a high correlation with the *Central Time* (in bold), especially for participants with a strong musical background. The distribution of the correlation coefficients of the measures show also significant standard deviation, especially for participants with little or no

bathroom-like	*user 03*	*user 08*	*user 12*	*user 13*
Reverb. Time	-0.14	0.13	0.05	0.07
Echo Density	0.10	-0.09	0.21	0.17
Clarity	-0.02	0.12	0.25	*-0.63*
Central Time	**0.78**	*-0.44*	0.01	**0.74**
Spec. Centroid	*-0.27*	**0.47**	**0.60**	-0.09
predictiveness	0.83	0.77	0.81	0.93
human ratings	7.0	8.0	10.0	7.0

Table 4. Comparison of the correlation coefficients of the measures between four different users for *bathroom-like*.

musical background. The same conclusions can be drawn here: participants have their own way of understanding the concept, and overall the system succeeds in grasping it to build a controller which meets participants' expectations.

Overall, *clear* shows the best mean results across all users: *user consistency*, 0.73, *system predictiveness*, 0.85, and *human rating*, 8.5. Overall, *bathroom-like* shows the worst results: *user consistency*, 0.62, *system predictiveness*, 0.62, and *human rating*, 6.8. Fig. 6 shows the distributions over all the participants of the *user consistency* and *system predictiveness* correlation coefficients, and the *human ratings* for *clear* and *bathroom-like*.

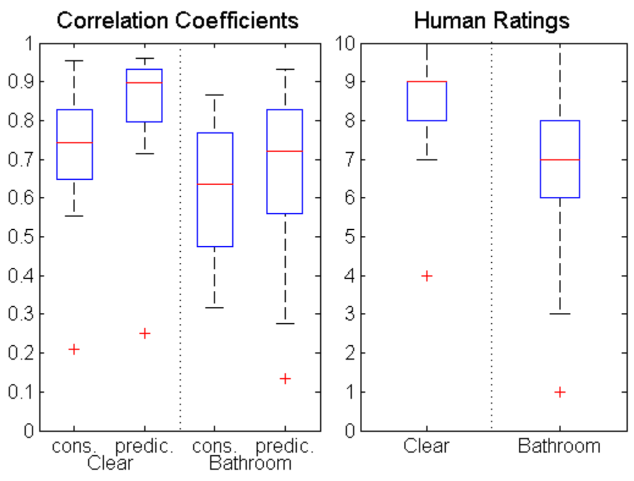


Figure 6. Left boxplot: distributions of *user consistency* and *system predictiveness* correlation coefficients for the best performing word: *clear* (left) and the worst performing word: *bathroom* (right) ; right boxplot: distributions of *human ratings* for *clear* (left) and *bathroom* (right).

5. CONCLUSION

A method for mapping subjective terms onto perceptual audio measures useful for digital reverberation control has been presented. This lets us build a simple controller to manipulate sound in terms of a subjective audio concept, bypassing the bottleneck of complex interfaces and individual differences in descriptive terms. The evaluation of our system showed that audio descriptors can be effectively and rapidly learned and controlled with this method.

Our study showed that people have different definitions of the same descriptor, and yet our system succeeds in learning an individual's concept so that people are satisfied with the final controller. This supports our contention that individualizing controllers is a useful approach.

There are a number of directions we expect to take in this work. We wish to conduct a more grounded psychoacoustic study to determine meaningful ranges for the set of reverberation measures. Finally, joint learning of controls for multiple audio effects (reverberation and equalization, for example) can be considered, to span a wider range of possible manipulations of sound. This work was supported by NSF grant number IIS-0757544.

6. REFERENCES

[1] Mihir Sarkar, Barry Vercoe, and Yang Yang. "Words that Describe Timbre, A Study of Auditory Perception Through Language," *Language and Music as Cognitive Systems Conference*, Cambridge, UK, May 2007.

[2] Alastair C. Disley and David M. Howard. "Spectral correlates of timbral semantics relating to the pipe organ," *Joint Baltic-Nordic Acoustics Meeting*, Mariehamn, Aland, Finland, 8-10 June 2004.

[3] Victor Alvarez-Cortes, Benjamin E. Zayas-Perez, Victor Huga Zarate-Silva, and Jorge A. Ramirez Uresti. "Current Trends in Adaptive User Interfaces: Challenges and Applications," *Electronics, Robotics and Automotive Mechanics Conference*, pp. 312-317, 2007.

[4] Andrew T. Sabin and Bryan Pardo. "Rapid learning of subjective preference in equalization," *125th Audio Engineering Society Convention*, San Francisco, CA, USA, 2-5 October 2008.

[5] Carl R. Nave. *HyperPhysics*, Georgia State University, Atlanta, GA, USA, 2006, http://hyperphysics.phy-astr.gsu.edu/hbase/hph.html.

[6] Pavel Zahorik. "Perceptual Scaling of Room Reverberation," *Journal of the Acoustical Society of America*, Vol. 15, No. B, pp. 2598-2598, 2001.

[7] Manfred R. Schroeder and Benjamin F. Logan. "'Colorless' Artificial Reverberation," *Journal of the Audio Engineering Society*, Vol. 9, No. 3, July 1961.

[8] James A. Moorer. "About This Reverberation Business," *Computer Music Journal*, July 1979.

[9] Fernando A. Beltrán, José R. Beltrán, Nicolas Holzem, and Adrian Gogu. "Matlab Implementation of Reverberation Algorithms," *Journal of New Music Research*, Vol. 31, No. 2, pp. 153-161, June 2002.

[10] Zafar Rafii and Bryan Pardo. "A Digital Reverberator controlled through Measures of the Reverberation," Northwestern University, EECS Department, Technical Report NWU-EECS-09-08, 2009.

[11] Fons Adriaensen. "Acoustical Impulse Response Measurement with ALIKI," *4th International Linux Audio Conference*, Karlsruhe, Germany, 27-30 April 2006.

INTERACTIVE GTTM ANALYZER

Masatoshi Hamanaka
University of Tsukuba
hamanaka@iit.tsukuba.ac.jp

Satoshi Tojo
Japan Advanced Institute of Science and Technology
tojo@jaist.ac.jp

ABSTRACT

We describe an interactive analyzer for the generative theory of tonal music (GTTM). Generally, a piece of music has more than one interpretation, and dealing with such ambiguity is one of the major problems when constructing a music analysis system. To solve this problem, we propose an interactive GTTM analyzer, called an automatic time-span tree analyzer (ATTA), with a GTTM manual editor. The ATTA has adjustable parameters that enable the analyzer to generate multiple analysis results. As the ATTA cannot output all the analysis results that correspond to all the interpretations of a piece of music, we designed a GTTM manual editor, which generates all the analysis results. Experimental results showed that our interactive GTTM analyzer outperformed the GTTM manual editor without an ATTA. Since we hope to contribute to the research of music analysis, we publicize our interactive GTTM analyzer and a dataset of three hundred pairs of a score and analysis results by musicologist on our website http://music.iit.tsukuba.ac.jp/hamanaka/gttm.htm, which is the largest database of analyzed results from the GTTM to date.

1. INTRODUCTION

We have been constructing a music analyzer based on the generative theory of tonal music (GTTM) [1]. The GTTM is composed of four modules, each of which assigns a separate structural description to a listener's understanding of a piece of music. These four modules output a grouping structure, a metrical structure, a time-span tree, and a prolongational tree. The main advantage of implementing the GTTM on a computer is to acquire tree structures called time-span and prolongational trees from the surface structure of a piece of music. The time-span and prolongational trees provide melody morphing, which generates an intermediate melody between two melodies with a systematic order [2]. It can also be used for performance rendering [3-5] and reproducing music [6] and provides a summarization of the music. This summarization can be used as a representation of a search, resulting in music retrieval systems [7].

In computer implementation of music theory [1, 8-10], we have to consider two types of ambiguity in music analysis. One involves human understanding of music, and the other concerns the representation of music theory. The former tolerates our subjective interpretation, while the latter is caused by the incompleteness of formal theory, and the GTTM is not an exception. Therefore, due to the former's ambiguity, we assume there is more than one correct result.

In our previous work, we proposed the exGTTM (machine-executable extension of GTTM) and constructed an automatic time-span tree analyzer (ATTA) to avoid the latter type of ambiguity, introducing as many parameters as possible [11, 12]. Whenever we find a correct result that the exGTTM cannot generate, we add new parameters with proper values to improve the result.

However, the ATTA has been clumsy for the first type of ambiguity. Even an identical melody can be played in different ways to represent different feelings since the ATTA cannot output the different analysis results in the same melody repetition. To solve this problem, we developed a GTTM manual editor that manually alternates the analysis results of the ATTA, according to the user's interpretations of a piece of music.

However, the ATTA still exhibits problems concerning the latter type of ambiguity. For example, the GTTM consists of feed-back operations from higher- to lower-level in the tree structure; however, no detailed description and only a few examples are given. To solve this problem, we developed a GTTM process editor, which enables seamless change of the automatic analysis process with an ATTA and the manual edit process with a GTTM manual editor. Therefore, a user can acquire the target analysis results by iterating the automatic and manual processes interactively and easily reflect his or her interpretations on a piece of music.

This paper is organized as follows. We present an overview of our interactive GTTM analyzer, which consists of the ATTA, GTTM manual editor, and GTTM process editor in Section 2, propose a manual editing method of the GTTM manual editor in Section 3, propose a process editing method of the GTTM process editor in Section 4, and present experimental results and conclusions in Sections 5 and 6, respectively. Finally, we provide in the appendix the data format of the analyzing results of the GTTM, which we publicize along with those of the interactive GTTM analyzer.

2. INTERACTIVE GTTM ANALYZER

Figure 1 is a screenshot of the viewer of our interactive GTTM analyzer. There is a sequence of notes displayed in a piano roll format. Below the notes is a grouping structure, which is graphically presented as several levels of arcs. The grouping structure is intended to formalize the intuitive belief that tonal music is organized into groups that are in turn composed of subgroups. Below the grouping structure is a metrical structure. The metrical structure describes the rhythmical hierarchy of the piece by identifying the position of strong beats at the levels of a quarter note, half note, one measure, two measures, four measures, and so on. Strong beats are illustrated as several levels of bars. Above the notes, there is a time-span tree. The time-span tree is a binary tree, which is a hierarchical structure describing the relative structural importance of notes that differentiate the essential parts of the melody from the ornamentation. Below the time-span tree is a prolongational tree, a binary tree that expresses the structure of tension and relaxation in a piece of music.

Figure 2 is an overview of our interactive GTTM analyzer, consisting of an ATTA, a GTTM manual editor, and a GTTM process editor. The ATTA consists of a grouping structure, a metrical structure, and time-span tree analyzers. We have been developing a prolongational tree analyzer. Hamanaka et al. explain the details of the ATTA [11].

The GTTM manual editor consists of grouping, metrical, time-span, prolongational, and Tonal Pitch Space editors. The Tonal Pitch Space [12] is a music theory for chord progression composed by Lerdhal, who is one of the authors of the GTTM. Although the GTTM includes rules that require the analysis results of chord progression, the ATTA uses such rules by adopting the results of the Tonal Pitch Space.

The analyzing process with the ATTA and GTTM manual editor is complicated, and sometimes a user is confused as to what he or she should do in the next process, as there are three analyzing processes in the ATTA and five editing processes in the GTTM manual editor. A user may iterate the ATTA and manual edit processes multiple times. To solve this problem, we propose a GTTM process editor, which presents candidates for the next process of analysis, and a user only needs to change the process, just by selecting the next process.

We use an XML format for all the input and output data structures of our interactive GTTM analyzer. Each analyzer and editor of our analyzer works independently, but they are integrated with the XML-based data structure.

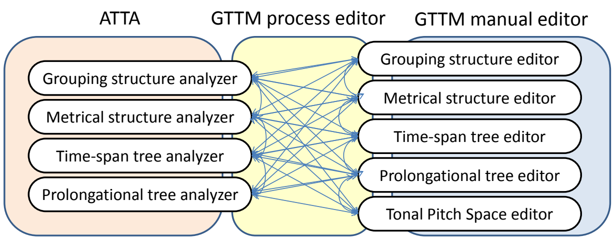
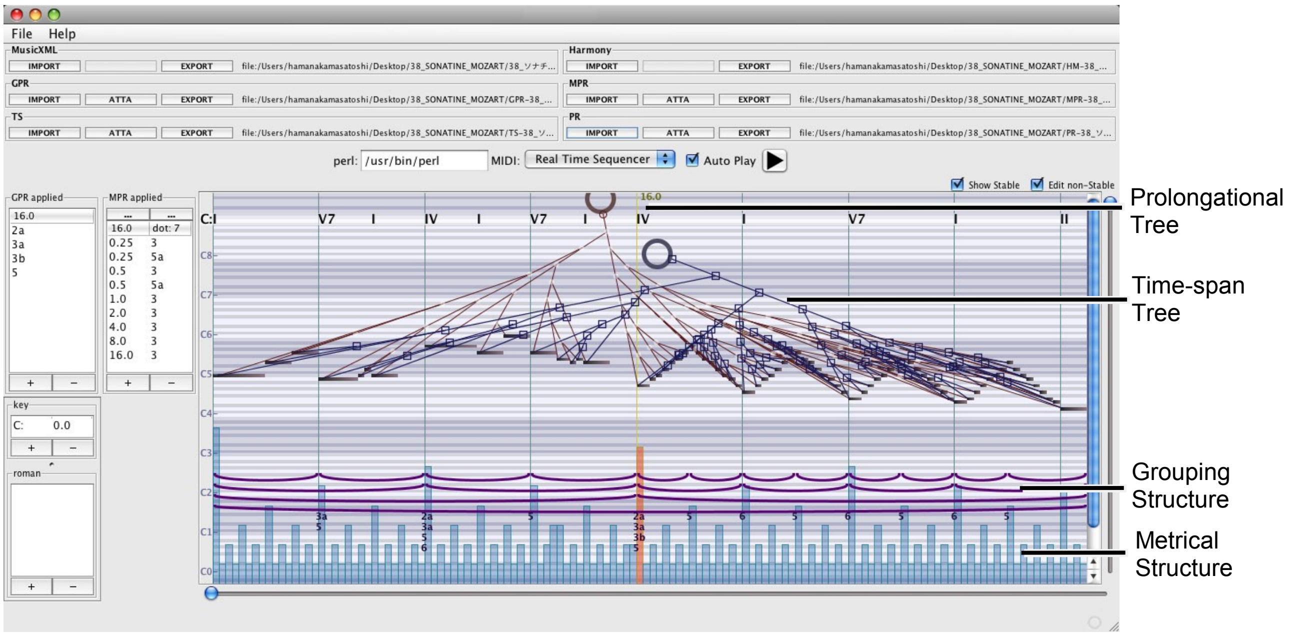

ATTA
GTTM process editor
GTTM manual editor
Grouping structure analyzer
Metrical structure analyzer
Time-span tree analyzer
Prolongational tree analyzer
Grouping structure editor
Metrical structure editor
Time-span tree editor
Prolongational tree editor
Tonal Pitch Space editor

Figure 2. Overview of interactive GTTM analyzer.

3. GTTM MANUAL EDITOR

In some cases, the ATTA may produce a preferable result, which reflects the user's interpretation, but in other case it may not. When a user wants to change the analysis result according to his or her interpretation, he or she can use the GTTM manual editor. We describe the method for editing and constructing a musical structure of the GTTM using the GTTM manual editor.



file:/Users/hamanakamasatoshi/Desktop/38_SONATINE_MOZART/38_ソナチ...
file:/Users/hamanakamasatoshi/Desktop/38_SONATINE_MOZART/GPR-38_...
file:/Users/hamanakamasatoshi/Desktop/38_SONATINE_MOZART/TS-38_ソ...
file:/Users/hamanakamasatoshi/Desktop/38_SONATINE_MOZART/HM-38_...
file:/Users/hamanakamasatoshi/Desktop/38_SONATINE_MOZART/MPR-38_...
file:/Users/hamanakamasatoshi/Desktop/38_SONATINE_MOZART/PR-38_ソ...

perl: /usr/bin/perl MIDI: Real Time Sequencer ✓ Auto Play

GPR applied: 16.0, 2a, 3a, 3b, 5

...	...
16.0	dot: 7
0.25	3
0.25	5a
0.5	3
0.5	5a
1.0	3
2.0	3
4.0	3
8.0	3
16.0	3

key: C: 0.0

Figure 1. Screenshot of interactive GTTM analyzer.

3.1 Grouping structure editor

Figure 3 is a screenshot of our interactive GTTM analyzer in editing a grouping structure. The color of the target group and all its subgroups turn red after selection with a mouse. Then we can open a popup menu by right clicking the mouse. There are four operations in the popup menu, *divide this group and create subgroup*, *divide this group*, *delete*, and *delete descendant*.

To change a position of a grouping boundary, a user first delete the groups which adjoin the boundary then divide the upper level (global level) group and create new subgroups where he or she wants to create a boundary. By left clicking a grouping boundary, the user sees the rules that are applied to the boundary and he or she can add or delete these rules.

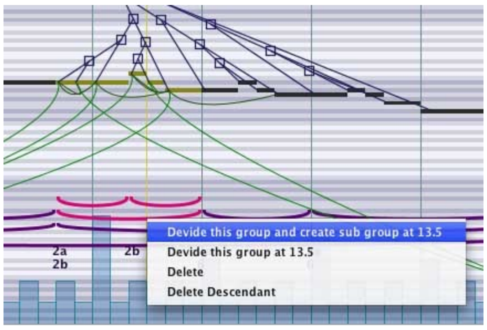

2a
2b
2b
Devide this group and create sub group at 13.5
Devide this group at 13.5
Delete
Delete Descendant

Figure 3. Screenshot of grouping structure editor.

3.2 Metrical structure editor

Although the metrical structure analyzer in the ATTA performs fairly well [11], a user may want to slightly edit the metrical structure. In which case, he or she applies the metrical structure editor, and changes the strength level of a beat by dragging a bar up or down. At the same time, he or she sees the rules that are applied to the bar and can add or delete these rules.

While editing beat strength, a user may break hierarchical metrical structures. In other words, the results of the metrical structure editor sometimes do not hold for the metrical preference rules. This problem can be solved using the GTTM process editor, which we discuss in Section 4.

3.3 Time-span tree editor

In the time-span tree, each branch has a head represented by a square in the time-span tree editor, and a user can move the head by dragging another branch. Figure 4 is a screenshot of dragging a head. The light blue branch is the former position, and the dark blue branch is the latter position. A user can select a type for each head by opening the popup menu among those four types, *ordinary*, *fusion*, *transformation*, and *cadential retention*.

3.4 Prolongational tree editor

The process of the prolongational tree editor is the same as that for the time-span tree. The prolongational tree is constructed by reconnecting the heads based on the time-span tree. There are head connection constraints of the

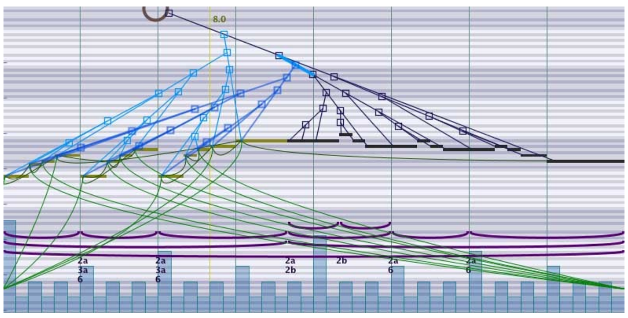


Figure 4. Screenshot of when dragging head.

prolongational tree. When a head connection of a prolongational tree is ill -formed, the GTTM process editor automatically opens the popup menu and displays candidates for a solution.

3.5 Tonal Pitch Space editor

The reason we include a Tonal Pitch Space editor in our interactive GTTM analyzer is that the editor provides quantitative grounds for the prolongational tree to be hierarchical. Therefore, analyzing the Tonal Pitch Space with the prolongational tree improves analyzing performance.

4. GTTM PROCESS EDITOR

There are two types of rules in the GTTM, which are *well-formedness* and *preference*. Well-formedness rules are necessary conditions for the rules assignment of a structure as well as the restrictions on the structure. When more than one structure satisfies the well-formedness rules, the preference rules indicate the superiority of one structure over another.

In the GTTM, the analysis sequence proceeds from the grouping structure, secondly to the metrical structure, next to the time-span tree, and finally to the prolongational tree. However, the GTTM contains feedback links from higher- to lower-level structures. For example, grouping preference rule 7 (GPR7) (time-span and prolongational stability) prefers a grouping structure that results in a more stable time-span and/or prolongation reduction. Therefore, to analyze with feedback link rules, we need to perform several analyzing processes by trial and error. The GTTM process editor helps in this repetition by performing three functions, data inputting, history recording, and process controlling.

4.1 Data inputting

Data inputting helps with the input of analysis results, which are prepared by another user or analyzer. For example, we do not have an automatic analyzer for the Tonal Pitch Space in our interactive GTTM analyzer; however, attempts have been made to implement the Tonal Pitch Space, so we can use those results [13].

We can add new rules to the ATTA using data inputting. For example, grouping preference rule 6 (GPR6) is a rule for parallelism in a grouping structure; however, the GTTM does not define the decision criteria for construing whether two or more segments are parallel or not. Therefore, many implementations of GPR6 would be possible, although we propose only one of them. By adding a new rule to the ATTA, we can control a new adjustable parameter for the new rule, GPR6+, which is the new implementation of GPR6.

4.2 History recording

History recording records the operation of analysis, and a user can return to the previous phase of analysis. History recording enables the copying and pasting of several operations of analysis while editing parallel phrases.

In the GTTM, there are few descriptions of the reasoning and working algorithms needed to compute the analysis results, especially for the time-span and prolongational trees. By using history recording, we look forward to storing the analysis knowledge, which improves automatic analysis.

4.3 Process controlling

Process controlling enables seamless change of the analysis process by using the ATTA and the manual edit process by using the GTTM manual editor, representing candidates for the next process of analysis. The representation method differs depending on the number of candidates for the next process.

4.3.1 One candidate

When there is only one candidate process, the process-controlling function automatically executes the process. For example, when a user edits the strongest beat in Figure 5a in the 2nd level, the hierarchical metrical structure is broken because in level 3 of Figure 5b there are three weak continuous beats, and the metrical well-formedness rule 2 (MWFW2) does not hold. MWFR2 requires that strong beats are spaced either two of three beats apart at each metrical level. The process editor automatically alternately produces strong and weak beats in level 3 (Figure 5c). If there is a higher metrical structure than level 3, the metrical analyzer of the ATTA automatically analyzes after level 3 and constructs a hierarchical metrical structure reflecting the user's intention.

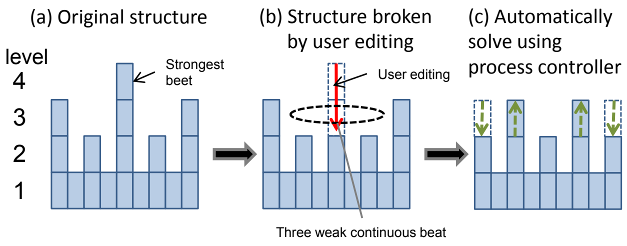

(a) Original structure
(b) Structure broken by user editing
(c) Automatically solve using process controller
level
4
3
2
1
Strongest beet
User editing
Three weak continuous beat

Figure 5. Automatically correct broken metrical structure.

4.3.2 A few candidates

When there are a few candidates, the process controlling function automatically opens the popup menu and shows the candidates. For example, if there is a grouping structure, as shown Figure 6a, and a user deletes a group at the upper left (Figure 6b), the grouping structure of Figure 6b is broken since grouping well-formedness rule 3 (GWFR3) does not hold. GWFR3 requires constraints that a group may contain smaller groups. To solve this problem, there are only two processes:

- Delete all the groups at the same level of the deleted group (Figure 6c).
- Extend the grouping boundary of the left end of the right group of the deleted group to the left end of that deleted group (Figure 6d).

The next process can be executed by selecting one of the two processes displayed in the popup menu.

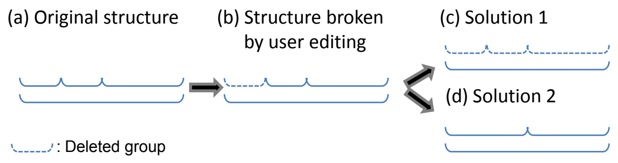


Figure 6. Two types of solutions for broken grouping structure.

4.3.3 Many candidates

When there are many candidates, the process-controlling function selects and shows the top-ten candidates from the history recording. The candidates are ordered depending on the similarity of the history. For example, after editing a time-span tree with the time-span tree editor, executing a grouping analyzer or metrical analyzer in the ATTA is ranked in the upper levels because there are rules for feedback link such as GPR7 or metrical preference rule 9 (MPR9). GPR7 (time-span and prolongational stability) is a link from the time-span and prolongational trees to the grouping structure, and MPR9 (time-span interaction) is a link from the time-span tree to the metrical structure.

We have not implemented the original ATTA on GPR7 and MPR9. In this paper, we omit the details of the implementation of these rules due to space limitations.

5. EXPERIMENTAL RESULTS

We asked a musicologist expert to manually analyze the score data faithfully with regard to the GTTM using our interactive GTTM analyzer. The musicologist collected three hundred 8-bar-long, monophonic, classical music pieces including notes, rests, slurs, accents, and articulations entered manually with music notation software called *Finale* [14]. The musicologist needed ten to twenty minutes for analyzing a piece. Three other experts crosschecked these results.

We measured the operating time for acquiring the target analysis results of our interactive GTTM analyzer and compared it with that of the GTTM manual editor without an ATTA. For the target analysis, we used one hundred pieces from the three hundred pairs of scores and correct data of grouping structure, metrical structure, and time-span tree. We did not use the prolongational tree in this measurement since its analyzer is still under development. As a result, our interactive GTTM analyzer outperformed the GTTM manual editor without an ATTA (Table 1).

Melodies	Interactive GTTM analyzer	GTTM manual editor
1 Grande Valse Brillante	326 sec	624 sec
2. Moments Musicaux	541 sec	791 sec.
3. Turkish March	724 sec	1026 sec
4. Anitras Tanz	621 sec	915 sec.
5. Valse du Petit Chien	876sec.	1246 sec.
	:	:
Total (100 melodies)	575 sec.	891 sec.

Table 1. Operation time of interactive GTTM analyzer and GTTM manual editor.

6. CONCLUSION

We developed a music analyzer called the interactive GTTM analyzer, which derives the grouping structure, metrical structure, time-span tree, and prolongational tree of the GTTM. The analyzer also derives analysis results of chord progression based on the Tonal Pitch Space. The analyzer consists of an automatic GTTM analyzer, called an ATTA, a GTTM manual editor, and a GTTM process editor. By using the process editor, a user can seamlessly change the analysis process of the ATTA and that of the manual editor. The experimental results show that our interactive GTTM analyzer outperformed the GTTM manual editor without an ATTA.

Since the original grouping rules of GTTM are based on monophonic melodies, we have implemented our system faithfully observing the theory. In the future, however, we plan to include harmonic analysis to complement the original theory and to target homophonic music.

7. REFERENCES

[1] Lerdahl, F., and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, Massachusetts, 1983.

[2] Hamanaka, M., Hirata, K., and Tojo, S.: "Melody morphing method based on GTTM', *Proceedings of the 2008 International Computer Music conference* (ICMC2008), pp. 155-158, 2008.

[3] Todd, N. "A Model of Expressive Timing in Tonal Music". *Musical Perception*, 3:1, 33-58, 1985.

[4] Widmer, G. "Understanding and Learning Musical Expression", *Proceedings of 1993 International Computer Music Conference* (ICMC1993), pp. 268-275, 1993.

[5] Hirata, K., and Hiraga, R. "Ha-Hi-Hun plays Chopin's Etude", *Working Notes of IJCAI-03 Workshop on Methods for Automatic Music Performance and their Applications in a Public Rendering Contest*, pp. 72-73, 2003.

[6] Hirata, K., and Matsuda, S. "Annotated Music for Retrieval, Reproduction, and Sharing", *Proceedings of 2004 International Computer Music Conference* (ICMC2004), pp. 584-587, 2004.

[7] Hirata, K., and Matsuda, S. "Interactive Music Summarization based on Generative Theory of Tonal Music". *Journal of New Music Research*, 32:2, 165-177, 2003.

[8] Cooper, G., and Meyer, L. B. *The Rhythmic Structure of Music*. The University of Chicago Press, 1960.

[9] Narmour, E. *The Analysis and Cognition of Basic Melodic Structure*. The University of Chicago Press, 1990.

[10] Temperley, D. *The Congnition of Basic Musical Structures*. MIT press, Cambridge, 2001.

[11] Hamanaka, M., Hirata, K., and Tojo, S. "Implementing 'A Generative Theory of Tonal Music'", *Journal of New Music Research*, 35:4, 249-277, 2006.

[12] Lerdahl, F. *Tonal Pitch Space*, Oxford University Press, 2001.

[13] Sakamoto, S., and Tojo, S.: " Harmony Analysis of Music in Tonal Pitch Space", Information Processing Society of Japan SIG Technical Report, Vol. 2009, May 2009 (in Japanese).

[14] PG Music Inc.: Finale, Available online at: http://www.pgmusic.com/finale.htm, 2009.

[15] Recordare, LLC.: MusicXML 2.0 Tutorial, Available online at http://www.recordare.com/xml/musicxml-tutorial.pdf, 2009.

[16] Recordare, LLC.: Dolet 4 for Finale, Available online at http://www.recordare.com/finale/index.html, 2009.

[17] W3C. XML Pointer Language (XPointer). http://www.w3.org/TR/xptr/, 2002.

[18] W3C. XML Linking Language (XLink) Version 1.0. http://www.w3.org/TR/xlink/, 2001.

APPENDIX: PUBLICLY AND DATA FORMAT

We publicize our interactive GTTM analyzer and database of three hundred pairs of scores and correct data at the following URL.

http://music.iit.tsukuba.ac.jp/hamanaka/gttm.htm

We believe that the exhibition of this kind of resource is important for the music information-researching community. The interactive GTTM analyzer is the first application for acquiring time-span trees and prolongational trees. We hope to benchmark the analyzer to other systems, which will be constructed.

We use the XML as the import and export format since the XML format is extremely qualified to express hierarchical musical structures.

MusicXML

As a primary input format, we chose MusicXML [15] because it provides a common 'interlingua' for music notation, analysis, retrieval, and other applications. For exporting MusicXML from finale we use a plug-in called *Dolet* [16].

GroupingXML

We designed Grouping.XML as an import and export format for hierarchical grouping structures. The GroupingXML has *group*, *note*, and *applied* elements. All *note* elements are inside hierarchical *group* elements. The *applied* elements are located between the end of a group tag and the start of the next group tag, which is where the grouping preference rules (GPRs) are applied. Figure 7a shows a simple example of GroupingXML.

MetricalXML

We designed MetricalXML as an import and export format for metrical structures. MetricalXML has *metric* elements, which require a *dot* attribute, an *at* attribute, and *applied* elements. The *dot* attribute indicates the strength of each beat. The *at* attribute indicates the time from the start of the piece. The *applied* element requires a *level* attribute and a *rule* attribute. In the metrical structure analysis, metrical preference rules (MPRs) are applied to each hierarchy of a dot. The *level* attribute indicates the interval of dots. If there is an onset of a note at the beat position, the *note* element is inserted before the end of the metric element (Figure 7b)

Time-spanXML, ProlongationalXML

The Time-spanXML has *ts*, *primary*, and *secondary* elements. The *ts* element has a *time-span* attribute, a *leftend* attribute, and a *rightend* attribute. Therefore, the *ts* element indicates the length and position of the time-span in a piece of music. In the *ts* element, there is a *head* element, which requires a *note* element indicating the most salient note in the time-span tree. If there is more than one note in the time-span, we can divide the time-span in two parts. One includes the head, and the other does not. The *primary* element in the *ts* element has a next-level *ts* element that corresponds to the time-span, which includes the upper level head. The *secondary* element in the *ts* element has a next-level *ts* element that corresponds to the time-span, which does not include the upper level head (Figure 7c).

We do not explain ProlongationalXML because its structure is similar to that of the time-span tree.

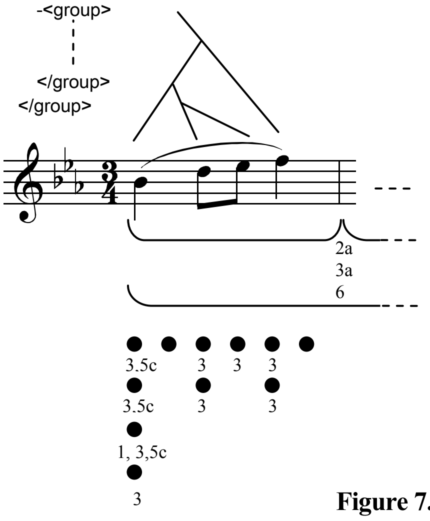
Tonal Pitch SpaceXML

The Tonal Pitch SpaceXML has *region* elements. Inside the *region* elements there are *chord* elements, and inside the *chord* element there are *note* elements.

Note that *note* elements in GroupingXML, MetricalXML, Time-spanXML, ProlongationalXML, and Tonal Pitch Space-XML are connected to *note* elements in MusicXML using Xpointer [17] and Xlink [18].

(a) GroupingXML

```
-<group>
  -<group>
    +<note id="P1-1-1"/>
    +<note id="P1-1-2"/>
    +<note id="P1-1-3"/>
    +<note id="P1-1-4"/>
  </group>
  <applied  rule=”  2a”  />
  <applied  rule=”  3a”  />
  <applied  rule=”  6”  />
  -<group>
     ¦
     ¦
  </group>
</group>
```



(b) MetricalXML

```
<metric dot="6" at="0.0">
    <applied level="0.5" rule="3"/>
    <applied level="0.5" rule="5c"/>
    <applied level="1.0" rule="3"/>
    <applied level="1.0" rule="5c"/>
    <applied level="3.0" rule="1"/>
    <applied level="3.0" rule="3"/>
    <applied level="3.0" rule="5c"/>
    <applied level="6.0" rule="3"/>
    +<note id="P1-1-1"/>
</metric>
<metric dot="1" at="0.5"/>
<metric dot="2" at="1.0">
    <applied level="0.5" rule="3"/>
    <applied level="1.0" rule="3"/>
    +<note id="P1-1-2"/>
</metric>
<metric dot="1" at="1.5">
    <applied level="0.5" rule="3"/>
    +<note id="P1-1-3"/>
</metric>
<metric dot="2" at="2.0">
    <applied level="0.5" rule="3"/>
    <applied level="1.0" rule="3"/>
    +<note id="P1-1-4"/>
</metric>
<metric dot="1" at="2.5"/>
```

(c) Time-spanXML

```
<ts timespan="3.0" leftend="0.0" rightend="3.0">
  <head> <note id="P1-1-4" /> </head>
  <primary>
    <ts timespan="1.0" leftend="2.0" rightend="3.0">
      <head> <note id="P1-1-4" /> </head>
    </ts>
  </primary>
  <secondary>
    <ts timespan="2.0" leftend="0.0" rightend="2.0">
      <head> <note id="P1-1-1" /> </head>
      <primary>
        <ts timespan="1.0" leftend="0.0" rightend="1.0">
          <head> <note id="P1-1-1" /> </head>
        </ts>
      </primary>
      <secondary>
        <ts timespan="1.0" leftend="1.0" rightend="2.0">
          <head> <note id="P1-1-2" /> </head>
          <primary>
            <ts timespan="0.5" leftend="1.0" rightend="1.5">
              <head> <note id="P1-1-2" /> </head>
            </ts>
          </primary>
          <secondary>
            <ts timespan="0.5" leftend="1.5" rightend="2.0">
              <head> <note id="P1-1-3" /> </head>
            </ts>
          </secondary>
        </ts>
      </secondary>
    </ts>
  </secondary>
</ts>
```

Figure 7. GroupingXML, MetricalXML, and Time-spanXML

ESTIMATING THE ERROR DISTRIBUTION OF A TAP SEQUENCE WITHOUT GROUND TRUTH

Roger B. Dannenberg
Carnegie Mellon University
School of Computer Science

Larry Wasserman
Carnegie Mellon University
Department of Statistics

ABSTRACT

Detecting beats, estimating tempo, aligning scores to audio, and detecting onsets are all interesting problems in the field of music information retrieval. In much of this research, it is convenient to think of beats as occuring at precise time points. However, anyone who has attempted to label beats by hand soon realizes that precise annotation of music audio is not possible. A common method of beat annotation is simply to tap along with audio and record the tap times. This raises the question: How accurate are the taps? It may seem that an answer to this question would require knowledge of "true" beat times. However, tap times can be characterized as a random distribution around true beat times. Multiple independent taps can be used to estimate not only the location of the true beat time, but also the statistical distribution of measured tap times around the true beat time. Thus, without knowledge of true beat times, and without even requiring the existence of precise beat times, we can estimate the uncertainty of tap times. This characterization of tapping can be useful for estimating tempo variation and evaluating alternative annotation methods.

1. INTRODUCTION

Tempo estimation and beat tracking are considered to be fundamental tasks of automatic music analysis and understanding. To evaluate machine performance in these sorts of tasks, it is useful to have audio annotated with beat times. We often assume that beat times are obvious and easily measured, usually through manual annotation. In some sense this is a fair assumption. Humans are good at detecting beats, especially in popular music, and human performance is generally better than machine performance. Most research simply accepts human-generated data as correct.

In cases where the goal is simply to get close to "true" beat times, or to estimate tempo (which can be a long-term average), ignoring potential tapping errors might be reasonable. However, it is troubling to assume errors do not matter without any way to test this assumption. Furthermore, there are some cases where automated methods can deliver quite precise results. For example, onset detection and beat detection in piano music can rely on fast onsets to obtain precise times in the millisecond range automatically. It seems unlikely that humans can tap or otherwise annotate beat times with this degree of precision, so how can we evaluate automatic labels?

The main goal of this work is to characterize the quality of human beat and tempo estimates in prerecorded audio data. A simple approach to this problem is to synthesize music from known control data such as MIDI, using control timing as the "ground truth" for beat times. This approach offers a clear connection to an underlying sequence of precise times, and after a human taps along with the music, some simple statistics can describe the distribution of actual tap times relative to the "true" beats. The problem here is that "real" music seems more complicated: Musicians are somewhat independent, adding their own timing variations, both intentional and unintentional. Musicians play instruments with varying attack times and they sometimes place their note onsets systematically earlier or later than the "true beat" times. How can we know that tapping to carefully controlled synthesized music is indicative of tapping to music in general?

We present an alternative approach in which multiple independent taps to beat-based music are used to estimate a distribution around the underlying "true" beat time. We assume that a true but hidden beat time exists and that observed tap times are clustered around these true times. In addition, we assume "all beats are the same" in the sense that observed tap times for one beat have the same distribution as observed tap times around any other beat. (This assumption will be discussed later.)

It should be apparent that different forms of tapping (tapping with different kinds of audio feedback, tapping with hands or feet, tapping by or while performing a musical instrument) will have subtle implications for the positioning and distribution of the tap times. Our techniques enable us to explore these differences but say nothing about whether one is more correct than another. In other words, there may be different implied "true" beat times for different tapping conditions.

In addition to estimating the distribution of tap times from multiple independent taps, our technique can estimate the distribution of another source of tap times. For example, we will show how a single set of foot tap times captured in a live performance can be used to estimate the accuracy of foot tapping, again without any ground truth. Our technique is interesting because it does not require any manual time estimation using visual editing, ground truth, or acoustical analysis, yet it gives us the ability to describe any sequence of estimated beat times as a probabilistic distribution around the underlying "true" beats.

By collecting data from real music audio examples, we can get a sense not only of the location of beats but the uncertainty of these locations. Since studies of expressive timing and tempo are normally based on beat time estimates, it is important to characterize uncertainty. In real-time computer music performance, estimating tempo and predicting the time of the next beat is an important problem. A good model of tapping and uncertainty can help to clarify the problem and analyze proposed solutions. There is also the potential to apply our model to the evaluation of automated beat tracking systems and to compare their performance to human tapping. Finally, models of timing and tempo change can help to build better beat tracking systems, which must reconcile prediction from past beat estimates using a steady-tempo hypothesis with new but uncertain beat estimates allowing the system to adapt to tempo change.

2. RELATED WORK

Previous studies have looked directly at tapping and synchronization. Michon [1] studied synchronization to sequences of clicks, and Mecca [2] studied human accompanists and how they adapt to tempo change. Wright [3] studied perceptual attack time, the perceived time or distribution of times at which a tone is perceived to begin. Dixon et al. [4] studied tapping to a short musical excerpt with expressive timing. There is a substantial literature on the perception of beats and rhythmic grouping [5]. The automatic detection of beats and tempo also has a long history of study [6,7]. The Mazurka project [8] has published beat times estimated using acoustic data from expressive performances.

Computer accompaniment [9] is a popular topic in the computer music literature and this work is closely related to ours. Tempo change in computer accompaniment has been modeled using Bayesian belief networks [10]. Our study of beat estimation and tempo in fact addresses shortcomings of existing computer accompaniment systems. In particular, computer accompaniment is usually based on score following, which assumes that a score exists and that there are audio signals to be matched to the score [9]. In reality, popular music often involves improvisation and other deviations from the score (if any), so the computer system must be "aware" of beats, measures, and cues in order to perform effectively with live players [11].

Conducting is another means for synchronizing computers to live performers and another example of human indication of beats. Various conducting systems have been created using traditional conducting gestures as well as simple tapping interfaces [12]. These studies are closely related to our work because any conducting system must sense beat times and make predictions about the tempo and the next beat time. Our work extends previous work by measuring human performance in tapping along to music. The sequential drum [13] and radio drum [14] of Max Mathews are also in the category of conducting systems. These emphasize expressive timing and multidimensional gestural control.

The "virtual orchestra" concept [15, 16] is also related. Virtual orchestras have been created to accompany dance, opera, and musical theater. Most if not all of this work is commercial and proprietary, so it is not known what techniques are used or how this work could be replicated, making any comparative studies impractical. Certainly, a better understanding of beat uncertainty and tempo estimation could contribute to the performance of these systems.

3. THE MODEL AND ASSUMPTIONS

We are interested in characterizing information obtained from tapping to music audio. In an ideal world, we would first label the audio with precise beat times. For example, we might ask a subject to tap by hand many times along with the music, measure the tap times, and compute the mean tap time $\widehat{\theta}_1, \widehat{\theta}_2, \ldots$ for each beat. Presumably, these mean tap times estimate and converge to a precise underlying or "hidden" time θ_i for each beat. In this way, beat times can be estimated with arbitrary precision given sufficient data. Once beat times are estimated, we can study other tap sequences. For example, given a sequence of foot tap times F_i we might like to estimate the distribution of timing errors: $\Delta_i = F_i - \theta_i$. If we ignore the difference between $\widehat{\theta}_i$ and θ_i, it is simple to compute the mean and standard deviation of Δ_i or simply to plot a histogram to characterize the distribution.

It should be noted that the outcome (the distribution of Δ_i) is a distribution over timing errors throughout the entire piece, not a distribution for a particular beat. Timing errors and the distributions of individual beats might be interesting things to study, but these are not considered by our model.

Unfortunately, tapping along to music requires much time, care, and concentration. We want to achieve the same results without tapping along to music many times. In fact, if we make a few assumptions about Δ_i, we only need to tap twice. Then, given a measured sequence of times F_i, we can estimate the corresponding distribution Δ_i.

The assumptions are that, first, Δ_i is normal. We will show some evidence that Δ_i obtained from actual tap data is in fact approximately normal. The second assumption is that the sequence of true beat times θ_i is well defined and the same for all tap sequences. So for example, if we want to compare foot taps to hand taps, we need to assume that the underlying "true" beats for each sequence are the same. Alternatively, if we want to measure the tap distribution of several subjects, we must assume they all share the same true beats.

In practice, we are seldom concerned about absolute shifts (subject A always perceives beats 10ms earlier than subject B). But introducing a time offset to a collection of tap times, say from subject A, generally increases the estimated variance of the tap times. If we believe that an offset may reflect individual differences, sensors, or calibration problems, then we can simply estimate and subtract off the offset. (Details will follow.) In that case the only assumption is that the "true" beat times for any two sequences of taps are the same except for a constant (but unknown) time offset.

4. ESTIMATING THE DISTRIBUTION

To estimate the distribution of actual tap times, let $\theta_1, \theta_2, \ldots$ denote the true beat times. (These will remain unknown.) Also, we will collect two sets of hand taps at times H_i^1, H_i^2. We assume that these times are normally distributed around the true beat times:

$$H_i^1, H_i^2 \sim \mathrm{Normal}(\theta_i, \sigma^2). \tag{1}$$

An unbiased estimate of σ^2 is

$$\widehat{\sigma}^2 = \frac{1}{2n}\sum_{i=1}^{n}(H_i^1 - H_i^2)^2. \tag{2}$$

Thus, with only two sets of taps generated under the same conditions, we can estimate the distribution of the taps relative to the true beat times. It should be mentioned that H_i^1 and H_i^2 must correspond to the same true beat. If, for example, one tap is missing from H^1, then some differences $(H_i^1 - H_i^2)$ will be increased by one beat. In practice, taps rarely differ by more than 150ms and beats are typically separated by 500ms or more (taps can be every 2, 3, or 4 beats if the tempo is faster), so errors are simple to find and correct.

What if H^2 has a constant offset relative to H^1? Since we assume the distribution around the true beat should be the same for both sequences, the mean of their differences $\bar{d}$:

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n}(H_i^1 - H_i^2) \tag{3}$$

should be zero. We can "correct" any constant offset (estimated by $\bar{d}$) by replacing H_i^2 by $(H_i^2 + \bar{d})$.

Now suppose we have another set of tap times generated by a different source, for example foot taps or taps from another subject. What is the distribution of these taps? Given H_i^1 and H_i^2, we only need one set of taps (one tap per beat) from the new source.

Let F_i be the new set of tap times, and let $\Delta_i = F_i - \theta_i$. The problem is to estimate the distribution of the Δ_i's. Let us begin by defining

$$\widehat{\Delta}_i = F_i - \widehat{\theta}_i \tag{4}$$

where $\widehat{\theta}_i$ is an estimate of θ_i. For these estimates, we will use

$$\widehat{\theta}_i = \frac{H_i^1 + H_i^2}{2}. \tag{5}$$

From the assumption that F_i is normal, $F_i \sim N(\theta_i, \tau^2)$, it follows that

$$\widehat{\Delta}_i \sim N\left(0, \tau^2 + \frac{\sigma^2}{2}\right) \tag{6}$$

Here, τ^2 is due to random variation in F_i and $\frac{\sigma^2}{2}$ is due to uncertainty in our estimates of the true beat times. Now, if we let s^2 be the expected sample variance of the $\widehat{\Delta}_i$, we obtain

$$s^2 = \tau^2 + \frac{\sigma^2}{2} \tag{7}$$

and hence

$$\tau^2 = s^2 - \frac{\sigma^2}{2} \tag{8}$$

Thus,

$$\Delta_i \sim N\left(0, s^2 - \frac{\sigma^2}{2}\right) \tag{9}$$

We already have an estimate of σ^2, and we can estimate s^2 using the sample variance $\widehat{s}^2$ of $\widehat{\Delta}_i$. Substituting $\widehat{\sigma}^2$ for σ^2 and $\widehat{s}^2$ for s^2, we can estimate the distribution of Δ_i and thus the accuracy of taps from the new source even without any ground truth for beat times. All we need are two additional sets of times obtained by tapping along with the music.

5. GENERALIZATION TO N SEQUENCES

This approach can be generalized to multiple tap sequences. For example, taps from many different subjects might be combined. Suppose that N tap sequences, $H_i^1, \ldots, H_i^N$ are normally distributed with means θ_i and variance σ^2. We estimate means and variance as follows:

$$\widehat{\theta}_i = \frac{1}{N}\sum_{j=1}^{N} H_i^j \tag{10}$$

and

$$\widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} s_i^2 \tag{11}$$

where

$$s_i^2 = \frac{1}{N-1}\sum_{j=1}^{N}(H_i^j - \widehat{\theta}_i)^2. \tag{12}$$

Defining $\widehat{\Delta}_i$ again as in (4), we generalize (6) to

$$\widehat{\Delta}_i \sim N(0, \tau^2 + \frac{\sigma^2}{N}). \tag{13}$$

Letting S^2 be the expected sample variance of the $\widehat{\Delta}_i$,

$$\Delta_i \sim N(0, \tau^2) = N\left(0, S^2 - \frac{\sigma^2}{N}\right). \tag{14}$$

Again, we can estimate S^2 using the sample variance $\widehat{S}^2$ of $\widehat{\Delta}_i$ and estimate the variance of Δ_i as $\widehat{S}^2 - \frac{\widehat{\sigma}^2}{N}$.

6. IS Δ_I NORMAL?

Our analysis assumes that the distribution of Δ_i is normal. We collected some taps to music synthesized from MIDI with note onsets quantized to exact beat times and smoothly varying but mostly constant tempo. Figure 1 shows a histogram of differences between the 117 "true" beats and hand-tapped (by one of the authors) beats, corresponding to Δ_i. To characterize the error, we use the mean of the absolute difference (MAD) between tapped beats and true beats after adjusting an absolute time offset to obtain a mean difference of zero. For this condition, the MAD is 16.13ms. Although the extreme values of +/-60ms seem quite large, the MAD value of 16.13ms compares favorably to the typical value of 10ms cited as the just noticeable difference (JND) for timing deviation [17]. (Since our goal is to describe a representation and its theory, we show only a couple of typical examples from data collected from a growing collection of songs, performers, and tappers.)

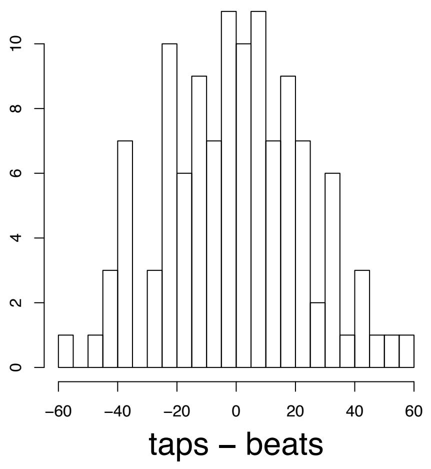


Figure 1. *Histogram of deviations (in ms) of hand tap times from "true" (MIDI) beat times.*

Using live acoustic music, two sets of hand tap times were collected, and Figure 2 shows differences between corresponding hand tap times. In this example, the music was from a big-band jazz rehearsal. Again, the data is from one of the authors, but it is typical of other data we have collected. This differs from Figure 1 in that the time differences are between two tap times to acoustic music rather than between a tap time and a known beat time in synthesized music. The standard deviation is 26ms. As with the MIDI-related data, the general shape of the histogram appears to be Gaussian, so the Normality assumption is at least reasonable. A Shapiro-Wilks test of Normality on data in Figures 1 and 2 yields values of $W = 0.9829$ (p-value = .6445), and $W = 0.9882$ (p-value = .2625), suggesting again that Normality is reasonable.

7. EXAMPLE

We are interested in characterizing foot tapping as an indicator of beat times. For our data used in Figure 2, $\widehat{\sigma} = 32.77$ms (standard error 2.006ms).

Even before collecting hand tap times, we recorded audio and foot tap times from a live performance. The foot taps are sensed by a custom pedal that uses a force-sensitive resistor (FSR) to control the frequency of a low-power analog oscillator [15]. The audio output from the pedal can be sent to one channel of a stereo recording in synchrony with

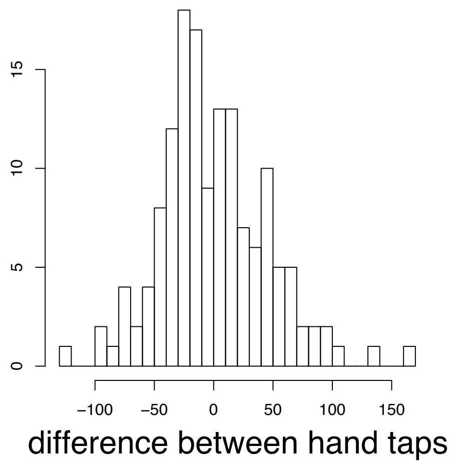


Figure 2. *Histogram of differences (in ms) between two sets of hand tap times to audio recording of a live performance.*

the live music on the other channel. Later, the "foot pedal channel" can be analyzed to detect foot taps with precise synchronization to the music audio. We then used Sonic Visualizer [18] to record hand taps (twice) while listening to the music channel of the recording.

Finally, using the analysis described in Section 4, we obtain a mean of 0 and a standard deviation of 37.2ms. This number reflects a particular condition involving the type of music, the other players involved, the interference task of performing, and possibly individual differences. Thus, we are not suggesting that one can give meaningful numbers for the accuracy of hand-tapped or foot-tapped beats in general, only that for any given situation, the taps can be accurately and efficiently characterized without a ground truth for beat times.

This example is interesting because it is impossible to obtain more than one set of foot taps or ground truth from a live performance, yet our technique still provides an estimate of the foot tap error distribution.

8. DISCUSSION

Because beats are hidden and perceptual, there are multiple ways to characterize beats. Just as there are differences between pitch (a percept) and fundamental frequency (a physical attribute), a distinction can be made between perceived beat times and acoustic event times. Some research relies on acoustic events to estimate beat times. While objective and often precise, these times are subject to various influences including random errors and physical characteristics of the instrument [19], so even acoustic times are the result of human perception, cognition, and action. After all, performing within an ensemble requires a perception of the beat and precise timing, so it is not all that different from tapping.

Furthermore, polyphony creates ambiguity because note onsets are often not synchronized. In fact, there is good evidence that note onsets are deliberately *not* placed on "the beat," at least in some important cases [20]. Therefore, this work attempts to identify and characterize *perceptual* beat times through tapping.

Even this approach has limitations. As seen in our example data, beat times are characterized as distributions rather than precise times, reflecting the limited information available from a small number of tap sequences. Moreover, all distributions are assumed to have the same variance. In music with a steady beat, this seems to be a reasonable assumption. In music with expressive timing, rubato, etc., one would expect some beats to be more accurately tapped than others. Learning from repeated listening can affect tapping times [4]. We suspect learning is a bigger factor in music with expressive timing where subjects might learn to anticipate timing variations. In music with a steadier tempo, any learning effect should be minimal.

The "meaning" of variance (τ^2) merits discussion. One interpretation is that the perceived beat time is very precise, but there are limitations in motor control that give rise to variation in tap times. Another interpretation is that the perception of beat times is not consistent from one listening to the next, resulting in different tap times. If different subjects tap, variance could arise from a difference between subjects. Ultimately, τ^2 models real data, so a more detailed model may not be relevant. On the other hand, experiments might be able isolate and characterize different influences on tap timing.

Using a limited amount of "field recording" data, we observed that foot tap timing can be approximated by a normal (Gaussian) random distribution around the "true" beat time. This is suggested by histograms as well as a Shapiro-Wilks test of Normality. The observed variance is almost certainly dependent upon the clarity of the beat, the steadiness of tempo, the skill of the tapper, interference tasks including playing an instrument while tapping, and other factors. The good news is that the method is practical and inexpensive, and the method can be used to study all of these factors.

Many studies in Computer Music, Music Information Retrieval, and Music Perception depend upon estimates of beat times and tempo variation. The techniques described here offer a principled way to go about characterizing the uncertainty of beat times obtained by tapping.

9. APPLICATIONS AND FUTURE WORK

The goal of this paper is to describe a representation of beat timing, the underlying estimation theory, and a practical way to use this representation. Current work is examining data from many sources with the goal of understanding the range of uncertainty (τ^2) observed under different conditions, and perhaps factors that account for differences. Also, experiments could study the degree to which tap time variance results from perceptual uncertainty vs motor control.

One of our goals is to create music systems that perform with live musicians using techniques based on work in Music Information Retrieval. Beat tracking, gesture sensing, analysis of mood, and other aspects of a performance all provide important input to an automated music performer.

In the area of beats and tempo, the techniques presented here are being used to analyze data from a variety of performances. For synchronization to live performers, the data will help us to tune systems that accurately predict the next beat time, allowing an artificial performer to play accurately on the beat. Beat timing variation implies tempo change. Modeling tap times probabilistically can help to distinguish between random timing errors and true tempo change. For example, preliminary analysis has shown that, depending upon the amount of tempo variation in a piece of music, estimating tempo using the previous 6 to 18 beats gives the best prediction of the next beat time. This work is closely related to beat tracking systems where smoothing over beats can help the system stay on track, but smoothing over too many beats makes the system unable to follow tempo changes.

Another application is in the construction and evaluation of score-to-audio alignment systems. While scores have precise beat times, audio recordings do not. By substituting alignment times for foot tap times (F_i in (4)), we can measure score alignment quality without any ground truth.

Audio labeling is another application. We might like to compare beat labels based on audio features to perceptual beat times. Since tap times might have a large variance, one is tempted to conclude that precise audio-based labels are more reliable. With our techniques, this can be tested. Another issue with labeling is the reliability of hand-labeled audio using an audio editor. This is a very difficult task where one might expect to see individual differences among human labelers. The lack of ground truth makes it difficult to evaluate different labelers. Our method might be useful because it does not need the ground truth to provide an analysis.

Finally, it is interesting to study tempo in the abstract. In live performances we have tapped to, we have found substantial tempo changes (on the order of 10%) during solos with a rhythm section where the tempo is nominally steady. As with live synchronization, one must be careful to avoid attributing tempo change to jitter in tap times, and a characterization of the tap time distribution helps to identify true tempo changes.

10. CONCLUSION

Our work concerns the analysis of beat times in music with a fairly steady beat. Our live data collection and analysis indicate that foot tap timing can be modeled well as a Gaussian distribution around a "true" but unknown beat time. We have introduced a new technique for estimating tapping accuracy that *does not require the accurate identification of underlying beats.* By comparing foot tap data (or data from other sources) to multiple hand taps on the same music, we are able to estimate the standard deviation and thus characterize the uncertainty in the tapping data. A major strength of this approach is that a one-time, irreproducible sequence of taps such as from a live performance can be analyzed in terms of accuracy without ground truth for "true" beat times.

11. ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of Nathaniel Anozie to the initial exploration and analysis of this data, and Cosma Shalizi for helpful consultation and discussions. This work was supported in part by Microsoft Research through the Computational Thinking Center at Carnegie Mellon.

12. REFERENCES

[1] J. A. Michon. *Timing in Temporal Tracking*. Van Gorcum, Assen, NL, 1967.

[2] M. Mecca. Tempo following behavior in musical accompaniment. Carnegie Mellon University, Department of Philosophy, Pittsburgh, PA, USA (Masters Thesis), May 1993.

[3] Matt Wright. *Computer-Based Music Theory and Acoustics*. PhD thesis, Stanford University, CA, USA, March 2008.

[4] S. Dixon, W. Goebl, and E. Cambouropoulos. Perceptual smoothness of tempo in expressively performed music. *Music Perception*, 23(3):195–21, 2006.

[5] P. Fraisse. Rhythm and tempo. In D. Deutsch, editor, *The Psychology of Music*, pages 149–80. Academic Press, New York, 1st edition edition, 1982.

[6] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1):34–54, 2005.

[7] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, September 2006.

[8] Craig. Sapp. Comparative analysis of multiple musical performances. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, pages 497–500, 2007.

[9] R. B. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Communications of the ACM*, 49(8):39–43, August 2006.

[10] C. Raphael. Synthesizing musical accompaniments with bayesian belief networks. *Journal of New Music Research*, 30:59–67, 2000.

[11] A. Robertson and M. D. Plumbley. B-keeper: A beat tracker for real time synchronisation within performance. In *Proceedings of New Interfaces for Musical Expression (NIME 2007)*, pages 234–237, 2007.

[12] R. B. Dannenberg and K. Bookstein. Practical aspects of a midi conducting program. In *ICMC Montreal 1991 Proceedings*, pages 537–540, San Francisco, 1991. International Computer Music Association.

[13] M. V. Mathews and C. Abbot. The sequential drum. *Computer Music Journal*, 4(4):45–59, Winter 1980.

[14] M. V. Mathews and W. A. Schloss. The radio drum as a synthesizer controller. In *Proceedings of the 1989 International Computer Music Conference*, San Francisco, 1989. Computer Music Association.

[15] Roger B. Dannenberg. New interfaces for popular music performance. In *NIME '07: Proceedings of the 7th International Conference on New Interfaces for Musical Expression*, pages 130–135, New York, NY, USA, 2007. ACM.

[16] Gregory M. Lamb. Robo-music gives musicians the jitters. *The Christian Science Monitor*, December 14, 2006.

[17] A. Friberg and J. Sundberg. Perception of just noticeable time displacement of a tone presented in a metrical sequence at different tempos. Technical report, STL-QPSR, Vol. 34, No. 2-3, pp. 49–56, 1993.

[18] C. Cannam, C. Landone, M. Sandler, and J. P. Bello. The sonic visualizer: A visualization platform for semantic descriptors from musical signals. In *ISMIR 2006, 7th International Conference on Music Information Retrieval*, pages 324–327, 2006.

[19] Patrik N. Juslin. Five facets of musical expression: A psychologist's perspective on music performance. *Psychology of Music*, 31(3):273–302, 2003.

[20] A. Friberg and A. Sundstrom. Swing ratios and ensemble timing in jazz performance: Evidence for a common rhythmic pattern. *Music Perception*, 19(3):333–349, 2002.

USING ACE XML 2.0 TO STORE AND SHARE FEATURE, INSTANCE AND CLASS DATA FOR MUSICAL CLASSIFICATION

Cory McKay
CIRMMT
McGill University
cory.mckay@mail.mcgill.ca

John Ashley Burgoyne
CIRMMT
McGill University
ashley@music.mcgill.ca

Jessica Thompson
Music Technology
McGill University
jessica.thompson@mail.mcgill.ca

Ichiro Fujinaga
CIRMMT
McGill University
ich@music.mcgill.ca

ABSTRACT

This paper introduces ACE XML 2.0, a set of file formats that are designed to meet the special representational needs of research in automatic music classification. Such standardized formats are needed to facilitate the sharing and long-term storage of valuable research data. ACE XML 2.0 is designed to represent a broad range of musical information clearly using a flexible, extensible, self-contained and formally structured framework. An emphasis is placed on representing extracted feature values, feature descriptions, instance annotations, class ontologies and related metadata.

1. INTRODUCTION

Many music information retrieval (MIR) research projects involve three core tasks: collecting and annotating ground-truth data; extracting feature values from instances; and training classification models using machine learning. These tasks require well-designed data representations, as insufficiently expressive representations can prevent learning algorithms from accessing valuable information.

Representational formats also have an important impact on the ability of MIR researchers to share valuable data with one another, particularly since ground-truth datasets can be expensive to acquire. Legal restrictions on distributing such datasets make the ability to share extracted feature values and ground-truth annotations particularly valuable. The absence of expressive, flexible, well-defined and well-supported standardized representational formats tends to result in individual research laboratories generating their own in-house data, with consequent wasteful repeated effort and lower quality data.

Standardized file formats are also needed to facilitate compatibility of MIR toolkits such as CLAM, jMIR, Marsyas, MIRtoolbox and Sonic Visualiser. Powerful packages such as these each have their own advantages, and a common representational format is needed if research performed using different toolkits is to be combined.

Standardized file formats are also needed to facilitate the evaluation and comparison of techniques from different research groups, something that has become apparent in the yearly Music Information Retrieval Evaluation Exchange (MIREX) competition [1]. The lack of an accepted standardized representational format necessitates the development of custom formats for each sub-task, which results in compromises with respect to expressivity and longevity. For example, the MIREX audio genre classification competition is carried out each year using ground-truth where each piece is labeled with only one genre label, despite general recognition that this is an unrealistic limitation that compromises results. The availability of standardized formats such as ACE XML that can be easily used to associate multiple classes with each instance could help to address such problems.

ACE XML 2.0 is proposed as a standard for representing information associated with the application of machine learning to music, including feature values, instance labels, class ontologies and associated metadata. ACE XML 2.0 has been developed as part of the Networked Environment for Musical Analysis (NEMA) [2] project, a multinational and multidisciplinary effort to create a general music information processing infrastructure.

2. ALTERNATIVE REPRESENTATIONS

There are a number of existing approaches that can be used to represent information related to automatic music classification. One is to simply store such information as raw binary data, such as Matlab [3] MAT files. Although this can be an easy and efficient way of storing data, it has problems with respect to portability, readability and longevity. Customized software is needed to parse or write each binary file type, and such software is often proprietary and can only be expected to have a limited life span.

Text files are an alternative to binary files. Although they are usually less space efficient, they address the weaknesses of binary files with respect to longevity, portability and readability. They can also be structured in a variety of standardized ways, ranging from simple delimited formats like CSV to markup languages like XML [4].

Weka ARFF [5] is a text-based format designed for general machine learning. Although ARFF files are currently the closest thing to a standard in the MIR community, they do have some significant limitations, such as inabilities to associate windows with instances, to group the values of a feature array, to store important metadata and to associate multiple classes with a single instance.

Another approach is to store feature values in audio files themselves, such as SDIF [6]. This technique can have a limited expressivity with respect to pertinent metadata, however, and is not appropriate for dealing with mixtures of cultural, symbolic and audio data.

Music Ontology [7] is one of the few representational frameworks designed specifically with MIR in mind, and it has many admirable strengths. It can represent musical ontologies of essentially any kind using RDF [8].

Music Ontology has a much broader scope than ACE XML, but is arguably less suited specifically to machine learning and automatic music classification, despite its advantages over ACE XML in other MIR domains. The advantages of ACE XML relative to semantic web solutions in general include greater conciseness; a reduced need for markup not directly relevant to the core problem domain; a lower barrier to entry for non-ontologists, particularly with respect to simplicity and convenience; a cleaner and more explicit structuring that is advantageous from a machine learning perspective; human readability, which is useful for application debugging and development; a self-contained nature that avoids the network dependence of RDF that can cause problems with respect to data integrity, robustness and accessibility, particularly considering the typically large size of feature data; and the simplicity of relying on only a single technology that is well-known in the MIR community (i.e., XML).

3. AN OVERVIEW OF ACE XML

3.1 General Overview

The primary design priorities behind ACE XML 2.0 are the maximization of expressivity, flexibility and extensibility while at the same time maintaining as much simplicity, accessibility and structure as possible.

There are four core ACE XML file types: *Feature Value, Feature Description, Instance Label* and *Class Ontology*. These file types hold, respectively, feature values extracted from instances; abstract information about features and their extraction parameters; class labels associated with particular instances and their subsections, as well as general metadata about instances; and ontological relationships between abstract classes. These XML file types may each be used independently, or they may be packaged with one another if desired (see Section 3.5).

The ACE XML file types are explained individually in Section 4, although space constraints prohibit more detailed descriptions. The XML DTDs shown in Figures 1 to 4 do specify their functionality in greater detail, however. Sample code excerpts for each of the four core ACE XML file types are also provided in Figures 5 to 8. It is important to note that these excerpts only demonstrate a reduced subset of ACE XML's expressivity, however, as many of the ACE XML elements and attributes are optional so that they can be included only when appropriate. This makes it possible to use simple and concise files by default, while maintaining the potential for much greater expressivity when needed.

ACE XML consists of multiple file types rather than just one because of the advantages, with respect to data portability and reusability, of explicitly separating fundamentally different types of information. One might, for example, extract features once from a large number of recordings and then reuse the resulting Feature Value file for multiple purposes, such as classification by performer, composer, genre and mood.

ACE XML is implemented in XML partly because it is a standardized format for which parsers are widely available. XML is also very flexible while maintaining the ability to structure data formally and clearly. XML is also relatively easily readable by both humans and machines.

ACE XML 2.0 is a significantly updated and expanded version of the earlier ACE XML 1.1, which was originally designed specifically for use with ACE [9]. It became apparent that certain important types of information could not be expressed with ACE XML 1.1, so ACE XML 2.0 was developed in order to address these needs and to make ACE XML useful to the MIR community outside the specific scope of ACE.

3.2 jMIR Support

jMIR [10] is a powerful suite of software applications developed for use as MIR research tools. Each of the jMIR applications reads and writes ACE XML, something that provides ACE XML early adopters with a powerful set of tools that are ready for immediate use:

- *jAudio:* An audio feature extractor.
- *jSymbolic:* A MIDI feature extractor.
- *jWebMiner:* A feature extractor that extracts cultural and demographic information from the web.
- *ACE:* A meta-learning system for machine learning.
- *jMusicMetaManager:* Software for managing and cataloguing large musical datasets.
- *Codaich, Bodhidharma MIDI, SAC:* research datasets.

3.3 Incorporating ACE XML into Other Software

A key factor in the effectiveness of any effort to encourage researchers to adopt new file formats is the ease with which they can incorporate the formats into their own software. Open-source code libraries are therefore currently in the process of being implemented to support ACE XML 2.0. These libraries are implemented in Java in order to increase portability, and do not rely on any additional technologies that might require special installation. They will provide functionality for parsing, writing and merging ACE XML files; for submitting search queries in JDOQL or SQL; and for performing various utility functions such as translating ACE XML data to and from Weka data. They will also include standard data structures that external code can access via a simple and well-documented API or a GUI ACE XML editor.

3.4 Linking ACE XML 2.0 to External Resources

It can be advantageous to associate instances, features or classes with various types of external information. Although ACE XML 2.0 can represent a broad range of metadata internally, the strong structuring that makes ACE XML advantageous for machine learning ultimately imposes limitations relative to the much more freely structured RDF, for example.

ACE XML addresses this issue by permitting the use of RDF-like triples via the optional *uri* XML element and

```
<!ELEMENT ace_xml_feature_value_file_2_0 (comments?,
             related_resources?, instance+)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT related_resources (feature_value_file*,
             feature_description_file*, instance_label_file*,
             class_ontology_file*, project_file*, uri*)>
<!ELEMENT feature_value_file (#PCDATA)>
<!ELEMENT feature_description_file (#PCDATA)>
<!ELEMENT instance_label_file (#PCDATA)>
<!ELEMENT class_ontology_file (#PCDATA)>
<!ELEMENT project_file (#PCDATA)>
<!ELEMENT uri (#PCDATA)>
<!ATTLIST uri predicate CDATA #IMPLIED>
<!ELEMENT instance (instance_id, uri*, extractor*, coord_units?,
                    s*, precise_coord*, f*)>
<!ELEMENT instance_id (#PCDATA)>
<!ELEMENT extractor (#PCDATA)>
<!ATTLIST extractor fname CDATA #REQUIRED>
<!ELEMENT coord_units (#PCDATA)>
<!ELEMENT s (uri*, f+)>
<!ATTLIST s b CDATA #REQUIRED e CDATA #REQUIRED>
<!ELEMENT precise_coord (uri*, f+)>
<!ATTLIST precise_coord coord CDATA #REQUIRED>
<!ELEMENT f (fid, uri*, (v+ | vd+ | vs+ | vj))>
<!ATTLIST f type (int | double | float | complex | string)
            #IMPLIED>
<!ELEMENT fid (#PCDATA)>
<!ELEMENT v (#PCDATA)>
<!ELEMENT vd (#PCDATA)>
<!ATTLIST vd d0 CDATA #REQUIRED d1 CDATA #IMPLIED
             d2 CDATA #IMPLIED d3 CDATA #IMPLIED
             d4 CDATA #IMPLIED d5 CDATA #IMPLIED
             d6 CDATA #IMPLIED d7 CDATA #IMPLIED
             d8 CDATA #IMPLIED d9 CDATA #IMPLIED>
<!ELEMENT vs (d+, v)>
<!ELEMENT d (#PCDATA)>
<!ELEMENT vj (#PCDATA)>
```

Figure 1: XML DTD for the ACE XML 2.0 Feature Value file format.

```
<!ELEMENT ace_xml_feature_description_file_2_0 (comments?,
             related_resources?, global_parameter*, feature+)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT related_resources (feature_value_file*,
             feature_description_file*, instance_label_file*,
             class_ontology_file*, project_file*, uri*)>
<!ELEMENT feature_value_file (#PCDATA)>
<!ELEMENT feature_description_file (#PCDATA)>
<!ELEMENT instance_label_file (#PCDATA)>
<!ELEMENT class_ontology_file (#PCDATA)>
<!ELEMENT project_file (#PCDATA)>
<!ELEMENT uri (#PCDATA)>
<!ATTLIST uri predicate CDATA #IMPLIED>
<!ELEMENT feature (fid, description?, related_feature*, uri*,
             scope, dimensionality?, data_type?, parameter*)>
<!ELEMENT fid (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT related_feature (fid, relation_id?, uri*,
                           explanation?)>
<!ELEMENT relation_id (#PCDATA)>
<!ELEMENT explanation (#PCDATA)>
<!ELEMENT scope (#PCDATA)>
<!ATTLIST scope overall (true|false) #REQUIRED
                sub_section (true|false) #REQUIRED
                precise_coord (true|false) #REQUIRED>
<!ELEMENT dimensionality (uri*, size*)>
<!ATTLIST dimensionality orthogonal_dimensions CDATA #REQUIRED>
<!ELEMENT size (#PCDATA)>
<!ELEMENT data_type (#PCDATA)>
<!ATTLIST data_type type (int | double | float | complex |
                          string) #REQUIRED>
<!ELEMENT global_parameter (parameter_id, uri*, description?,
                            value?)>
<!ELEMENT parameter (parameter_id, uri*, description?, value?)>
<!ELEMENT parameter_id (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

Figure 2: XML DTD for the ACE XML 2.0 Feature Description file format.

```
<!ELEMENT ace_xml_instance_label_file_2_0 (comments?,
             related_resources?, instance+)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT related_resources (feature_value_file*,
             feature_description_file*, instance_label_file*,
             class_ontology_file*, project_file*, uri*)>
<!ELEMENT feature_value_file (#PCDATA)>
<!ELEMENT feature_description_file (#PCDATA)>
<!ELEMENT instance_label_file (#PCDATA)>
<!ELEMENT class_ontology_file (#PCDATA)>
<!ELEMENT project_file (#PCDATA)>
<!ELEMENT uri (#PCDATA)>
<!ATTLIST uri predicate CDATA #IMPLIED>
<!ELEMENT instance (instance_id, misc_info*, related_instance*,
                    uri*, coord_units?, section*,
                    precise_coord*, class*)>
<!ATTLIST instance role (training | testing | predicted)
                    #IMPLIED>
<!ELEMENT instance_id (#PCDATA)>
<!ELEMENT related_instance (instance_id, relation_id?, uri*,
                            explanation?)>
<!ELEMENT relation_id (#PCDATA)>
<!ELEMENT explanation (#PCDATA)>
<!ELEMENT misc_info (info_id, uri*, info)>
<!ELEMENT info_id (#PCDATA)>
<!ELEMENT info (#PCDATA)>
<!ELEMENT coord_units (#PCDATA)>
<!ELEMENT section (uri*, class+)>
<!ATTLIST section begin CDATA #REQUIRED
                  end CDATA #REQUIRED>
<!ELEMENT precise_coord (uri*, class+)>
<!ATTLIST precise_coord coord CDATA #REQUIRED>
<!ELEMENT class (class_id, uri*)>
<!ATTLIST class weight CDATA "1">
<!ATTLIST class source_comment CDATA #IMPLIED>
<!ELEMENT class_id (#PCDATA)>
```

Figure 3: XML DTD for the ACE XML 2.0 Instance Label file format.

```
<!ELEMENT ace_xml_class_ontology_file_2_0 (comments?,
             related_resources?, class+)>
<!ATTLIST ace_xml_class_ontology_file_2_0 weights_relative
             (true|false) #REQUIRED>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT related_resources (feature_value_file*,
             feature_description_file*, instance_label_file*,
             class_ontology_file*, project_file*, uri*)>
<!ELEMENT feature_value_file (#PCDATA)>
<!ELEMENT feature_description_file (#PCDATA)>
<!ELEMENT instance_label_file (#PCDATA)>
<!ELEMENT class_ontology_file (#PCDATA)>
<!ELEMENT project_file (#PCDATA)>
<!ELEMENT uri (#PCDATA)>
<!ATTLIST uri predicate CDATA #IMPLIED>
<!ELEMENT class (class_id, misc_info*, uri*, related_class*,
                 sub_class*)>
<!ELEMENT class_id (#PCDATA)>
<!ELEMENT misc_info (info_id, uri*, info)>
<!ELEMENT info_id (#PCDATA)>
<!ELEMENT info (#PCDATA)>
<!ELEMENT related_class (class_id, relation_id?, uri*,
                         explanation?)>
<!ATTLIST related_class weight CDATA "1">
<!ELEMENT relation_id (#PCDATA)>
<!ELEMENT explanation (#PCDATA)>
<!ELEMENT sub_class (class_id, relation_id?, uri*,
                     explanation?)>
<!ATTLIST sub_class weight CDATA "1">
```

Figure 4: XML DTD for the ACE XML 2.0 Class Ontology file format.

its associated *predicate* attribute. This enables links to be specified to external resources of essentially any kind without compromising ACE XML's structured and self-contained design philosophy. In particular, it makes it easy to link ACE XML files to large RDF ontologies.

3.5 ACE XML 2.0 Project and ZIP Files

Although it is beneficial to be able to specify the information encapsulated in each of the four ACE XML formats in separate files, in practice users will want to use multiple ACE XML files together. The *ACE XML 2.0 Project* file format facilitates this by providing functionality for linking ACE XML (and other) resources together.

It is also possible to package multiple associated ACE XML files together into a single *ACE XML 2.0 ZIP* file for simplified storage and distribution. This is also advantageous because of the reduced file sizes resulting from data compression. Of course, the original ACE XML files may be extracted from this ACE XML ZIP file whenever desired. The supporting ACE XML software includes functionality for automatically generating, accessing and otherwise processing ACE XML Project and ZIP files.

4. THE CORE ACE XML 2.0 FILE FORMATS

This section provides descriptions of each of the four core ACE XML file formats: *Feature Value, Feature Description, Instance Label* and *Class Ontology*.

4.1 Feature Value Files

Feature Value files are used to express feature values that have been extracted from instances that are to be classified or used as training data. There is no assumed association with any specific kind of data, and so features may be extracted from audio recordings, symbolic recordings, textual or numeric cultural data, images of album art, etc.

Features may be extracted from instances as a whole (e.g., an entire score), from subsections of instances (e.g., audio analysis windows) or from a mixture of the two. Subsections may or may not overlap, may or may not be of equal size and may or may not cover an instance comprehensively. Each instance or subsection may also contain an arbitrary and potentially differing number of features, which makes it possible to omit features when appropriate or if they are unavailable.

Each instance in a Feature Vector file has an *instance_id* tag that may be used to associate it with class labels and metadata stored in an Instance Label file. Similarly, each feature has an *fid* tag that may be used to associate it with feature metadata stored in a Feature Description file. Other information that can be represented in a Feature Value file includes the data type (integer, double, string, etc.) of the feature, the feature extractor used to extract the feature values and links to external resources.

ACE XML 2.0 allows feature values to be expressed using any one of four methodologies, including one that is based on JavaScript Object Notation (JSON) [11]. Each such representation has its own advantages with respect to the maximum dimensionality of feature arrays, the ability to represent sparse feature arrays, human readability and space efficiency. An example of the most flexible (but not most space efficient) of these options is shown in Figure 5. This option allows feature arrays of any dimensionality and size to be represented, including sparse arrays and arrays that vary in size.

4.2 Feature Description Files

Feature Description files are used to express abstract information about features. These files do not specify actual feature values, as this information is instead specified in Feature Value files.

The information that may be represented in Feature Description files includes: details of pre-processing required before feature extraction (e.g., downsampling); feature extraction parameters; notations as to whether features are associated with instances as a whole or only with instance subsections; the dimensionality and size of each feature (i.e., a single-value feature, a feature vector or a feature array); the data type of each feature; qualitative feature descriptions; relationships between different features; and links to external resources.

There are other possible applications for Feature Description files beyond simply using them to represent information associated with Feature Value files. Examples include catalogues of features that can be extracted by particular feature extraction applications and lists of features and associated parameters that have been found to be useful for particular music classification applications.

4.3 Instance Label Files

Instance Label files are used to specify class labels and miscellaneous metadata about instances. These files are typically used to express ground-truth annotations or predicted labels, but there are certainly other uses as well.

Class labels may be assigned to instances as a whole, to subsections of instances, or to both. Subsections may be overlapping and may be of varying sizes. Weighted multi-class membership is also permitted. Additional information that may be associated with instances and their subsections includes the source of the class labels (e.g., a listener survey); whether the class label(s) for an instance are predicted labels or ground-truth; relationships of any kind between instances (e.g., one is a cover song of another); miscellaneous field-labeled qualitative metadata (e.g., the performer or composer of a piece); and links to external resources. Instance Label files may be linked with Feature Value files using matching *instance_id* tags and with Class Ontology files using matching *class* tags.

```
<instance>
   <instance_id>An Artificial Instance</instance_id>
   <f>
      <fid>A Single Value Feature</fid>
      <v>1</v>
   </f>
   <f>
      <fid>Feature Array of Value</fid>
      <vs><d>0</d><d>0</d><v>1</v></vs>
      <vs><d>0</d><d>1</d><v>2</v></vs>
      <vs><d>0</d><d>2</d><v>3</v></vs>
      <vs><d>1</d><d>0</d><v>11</v></vs>
      <vs><d>1</d><d>1</d><v>22</v></vs>
      <vs><d>1</d><d>2</d><v>33</v></vs>
   </f>
</instance>
```

Figure 5: An excerpt from a sample ACE XML 2.0 Feature Value file indicating two artificial features extracted from a single instance. The first feature has a value of *1,* and the second is the 2 by 3 feature array: *[[1,2,3],[11,22,33]]*. In practice, a Feature Value file could contain multiple such instances as well as features extracted from subsections of instances.

```
<feature>
   <fid>Beat Histogram</fid>
   <description>Tempo histogram calculated using
             Autocorrelation.</description>
   <related_feature>
      <fid>Tempo Peak</fid>
      <relation_id>derivative feature</relation_id>
   </related_feature>
   <scope overall="true" sub_section="false"
         precise_coord="false"></scope>
   <dimensionality orthogonal_dimensions="1">
      <size>161</size>
   </dimensionality>
   <data_type type="double"></data_type>
   <parameter>
      <parameter_id>normalized</parameter_id>
      <value>true</value>
   </parameter>
</feature>
```

Figure 6: An excerpt from a sample ACE XML 2.0 Feature Description file indicating information about a single feature called *Beat Histogram*. It is noted that *Beat Histogram* is related to another feature called *Tempo Peak* that can be calculated from the *Beat Histogram* feature, that *Beat Histogram* is configured to be extracted only for files as a whole, that it consists of a single vector of size 161, that feature values are stored as doubles and that the values are normalized. In practice, a Feature Description file would contain multiple such *feature* clauses, each for a different feature.

```
<instance role="predicted">
   <instance_id>C:\Symbolic\piece_42.midi</instance_id>
   <coord_units>ms</coord_units>

   <section begin="0" end="85673">
      <class>
         <class_id>Sonata Exposition</class_id>
      </class>
   </section>
   <section begin="85674" end="278894">
      <class>
         <class_id>Sonata Development</class_id>
      </class>
   </section>
   <section begin="278895" end="525419">
      <class>
         <class_id>Sonata Recapitulation</class_id>
      </class>
   </section>

   <class weight="3">
      <class_id>Haydn</class_id>
   </class>
   <class weight="1">
      <class_id>Mozart</class_id>
   </class>
</instance>
```

Figure 7: An excerpt from a sample ACE XML 2.0 Instance Label file specifying class labels for a MIDI file. As indicated by the *role* attribute, the labels are predicted classifier outputs. The subsections are classified by form and the overall instance is classified by composer. The classification system has expressed that this piece is three times as likely to be by Haydn than by Mozart. In practice, an Instance Label file would contain multiple such *instance* clauses.

```
<class>
   <class_id>Robert Johnson</class_id>
</class>

<class>
   <class_id>Muddy Waters</class_id>
   <related_class weight="10">
      <class_id>Robert Johnson</class_id>
      <relation_id>Influenced By</relation_id>
   </related_class>
   <related_class weight="1">
      <class_id>Eric Clapton</class_id>
      <relation_id>Influenced By</relation_id>
   </related_class>
</class>

<class>
   <class_id>Eric Clapton</class_id>
   <related_class weight="30">
      <class_id>Robert Johnson</class_id>
      <relation_id>Influenced By</relation_id>
   </related_class>
   <related_class weight="10">
      <class_id>Muddy Waters</class_id>
      <relation_id>Influenced By</relation_id>
   </related_class>
</class>
```

Figure 8: An excerpt from an artificial ACE XML 2.0 Class Ontology file indicating class labels consisting of names of Blues musicians. A type of relationship between classes is also specified, namely musicians influenced by other musicians. In this example, there is no relationship from Robert Johnson to the other musicians because he was not influenced by them. Both of the other musicians are influenced by Johnson, however. Clapton is more influenced by Johnson than by Muddy Waters, and Muddy Waters is strongly influenced by Johnson but only slightly influenced by Clapton, as indicated by the *weight* values.

4.4 Class Ontology Files

Class Ontology files are used to specify candidate class labels for a particular classification domain as well as weighted ontological relationships between classes. These files do not, however, specify the labels of any actual instances, as this is the domain of Instance Label files.

The ability to specify ontological class structuring has several important benefits. From a musicological perspective, it provides a simple, machine-readable way of specifying a variety of musical relationships. From a machine learning perspective, it has the dual advantages of enabling the use of powerful hierarchical classification methodologies that exploit this structuring, as well as learning schemes that utilize weighted penalization to punish "better" misclassifications less severely as training proceeds.

The information that may be expressed in Class Ontology files includes weighted taxonomical links to other classes; weighted general ontological links to other classes; structured or unstructured descriptions of such links; miscellaneous qualitative structured metadata (e.g., the birthplace of a composer if music is being classified by composer); and links to external resources.

5. CONCLUSIONS

This paper has emphasized the need for more effective representational formats for use in MIR and automatic music classification research. ACE XML 2.0 was presented as a solution to these needs. It is hoped that ACE XML will help to facilitate communication and data sharing between research groups involved in the computational study of music and correspondingly increase the efficiency and quality of research.

Much more detailed information, including sample ACE XML 2.0 files and an in-depth ACE XML 2.0 manual, are available at *jmir.sourceforge.net*.

6. FUTURE RESEARCH

Future work will focus on continuing to produce developer tools to help facilitate the integration of ACE XML functionality into other software. Once work is completed on implementing the ACE XML 2.0 support software (the ACE XML 1.1 software is already complete) in Java it will then be ported to other languages, such as Python, C++ and Matlab. There are also plans to write ACE XML 2.0 plug-ins for the popular MIR software toolkits and to implement tools for translating ACE XML to other representational formats. The upgrading of all jMIR components from ACE XML compatibility 1.1 to ACE 2.0 compatibility is a particular priority.

Another priority is the continuing extension and overall improvement of the ACE XML standard. This will include the expression of more strictly constrained rules specified using XSD or Relax NG schemas.

The publication of a common repository for data stored in ACE XML files is another key goal. This will enable such data to be posted and shared amongst researchers. This will also be a forum where best practices and extensions to the ACE XML standard can be discussed and agreed upon by the MIR community. Indeed, ideas from the MIR community for future improvements to ACE XML in general are very welcome, and upgrades to the file formats will continue, with the provision that backwards compatibility is maintained.

7. ACKNOWLEDGEMENTS

The authors would like to thank the *Andrew W. Mellon Foundation* and the *Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)* for their generous financial support, as well as the members of the *Networked Environment for Musical Analysis (NEMA)* group for their valuable critiques and suggestions.

8. REFERENCES

[1] MIREX 2009. Retrieved 11 May 2009, from http://www.music-ir.org/mirex/2009.

[2] NEMA. Retrieved 11 May 2009, from http://nema.lis.uiuc.edu.

[3] Mathworks. Retrieved 11May 2009, from http:// www.mathworks.com.

[4] Ray, E. T. 2003. *Learning XML.* Sebastopol, CA: O'Reilly Media.

[5] Witten, I. H., and E. Frank. 2005. *Data mining: Practical machine learning tools and techniques.* New York: Morgan Kaufman.

[6] Burred J. J., C. E. Cella, G. Peeters, A. Röbel, and D. Schwarz. 2008. Using the SDIF Sound Description Interchange Format for audio features. *Proceedings of the International Conference on Music Information Retrieval.* 427–32.

[7] Raimond, Y., S. Adbdallah, M. Sandler, and F. Giasson. 2007. The Music Ontology. *Proceedings of the International Conference on Music Information Retrieval.* 417–22.

[8] Powers, S. 2003. *Practical RDF.* Sebastopol, CA: O'Reilly Media.

[9] McKay, C., R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga. 2005. ACE: A framework for optimizing music classification. *Proceedings of the International Conference on Music Information Retrieval.* 42–9.

[10] McKay, C., and I. Fujinaga. 2009. jMIR: Tools for automatic music classification. Accepted for publication in the *Proceedings of the International Computer Music Conference.*

[11] JSON. Retrieved 11 May 2009, from http://json.org.

AN EFFICIENT MULTI-RESOLUTION SPECTRAL TRANSFORM FOR MUSIC ANALYSIS

Pablo Cancela **Martín Rocamora** **Ernesto López**
Universidad de la República, Instituto de Ingeniería Eléctrica, Montevideo, Uruguay
{pcancela,rocamora,elopez}@fing.edu.uy

ABSTRACT

In this paper we focus on multi-resolution spectral analysis algorithms for music signals based on the FFT. Two previously devised efficient algorithms (efficient constant-Q transform [1] and multiresolution FFT [2]) are reviewed and compared with a new proposal based on the IIR filtering of the FFT. Apart from its simplicity, the proposed method shows to be a good compromise between design flexibility and reduced computational effort. Additionally, it was used as a part of an effective melody extraction algorithm.

1. INTRODUCTION

Many automatic music analysis algorithms, such as those intended for melody extraction or multiple pitch estimation, rely on a spectral representation of the audio signal, typically the discrete Short Time Fourier Transform (STFT). A key issue that arises is the compromise between time and frequency resolution. The frequency components of a Discrete Fourier Transform (DFT) are equally spaced and have a constant resolution. However, in polyphonic music a higher frequency resolution is needed in the low and mid frequencies where there is a higher density of harmonics. On the other hand, frequency modulation gets stronger as the number of harmonic is increased, requiring shorter windows for improved time resolution. Thus, a multi resolution spectral representation is highly desired for the analysis of music signals. In addition, computational cost is a critical issue in real time or demanding applications so efficient algorithms are often needed.

In this context several proposals have been made to circumvent the conventional linear frequency and constant resolution of the DFT. The constant-Q transform (CQT) [3] is based on a direct evaluation of the DFT but the channel bandwidth Δf_k varies proportionally to its center frequency f_k, in order to keep constant its quality factor $Q = f_k/\Delta f_k$ (as in Wavelets). Center frequencies are distributed geometrically, to follow the equal tempered scale used in Western music, in such a way that there are two frequency components for each musical note (although higher values of Q provide a resolution beyond the semitone). Direct evaluation of the CQT is very time consuming, but fortunately an approximation can be computed efficiently taking advantage of the Fast Fourier Transform (FFT) [1].

Various approximations to a constant-Q spectral representation have also been proposed. The bounded-Q transform (BQT) [4] combines the FFT with a multirate filterbank. Octaves are distributed geometrically, but within each octave, channels are equally spaced, hence the log representation is approximated but with a different number of channels per octave. Note that the quartertone frequency distribution, in spite of being in accordance with Western tuning, can be too scattered if instruments are not perfectly tuned, exhibit inharmonicity or are able to vary their pitch continuously (e.g. glissando or vibrato). Recently a new version of the BQT with improved channel selectivity was proposed in [5] by applying the FFT structure but with longer kernel filters, a technique called Fast Filter Bank. An approach similar to the BQT is followed in [6] as a front-end to detect melody and bass line in real recordings. Also in the context of extracting the melody of polyphonic audio, different time-frequency resolutions are obtained in [2] by calculating the FFT with different window lengths. This is implemented by a very efficient algorithm, named the Multi-Resolution FFT (MR FFT), that combines elementary transforms into a hierarchical scheme.

In this paper we focus on multi-resolution spectral analysis algorithms for music signals based on the FFT. Two previously devised efficient algorithms that exhibit different characteristics are reviewed, namely, the efficient CQT [1] and the MR FFT [2]. The former is more flexible regarding Q design criteria and frequency channel distribution while the latter is more efficient at the expense of design constrains. These algorithms are compared with a new proposal based on the Infinite Impulse Response (IIR) filtering of the FFT (IIR CQT), that in addition to its simplicity shows to be a good compromise between design flexibility and reduced computational effort.

2. FIR Q TRANSFORM IMPLEMENTATIONS

2.1 Efficient constant Q transform

As stated in [3] a CQT can be calculated straightforwardly based on the evaluation of the DFT for the desired components. Consider the kth spectral component of the DFT:

$$X[k] = \sum_{n=0}^{N-1} w[n]x[n]e^{-j2\pi kn/N}$$

where $w[n]$ is the temporal window function and $x[n]$ is the discrete time signal. In this case the quality factor for a certain frequency f_k equals k, since $Q_k = f_k/\Delta f = f_k N/f_s = k$. This corresponds to the number of periods in the time frame for that frequency. The digital frequency is $2\pi k/N$ and the period in samples is N/k. In the CQT the length of the window function varies inversely with frequency (but the shape remains the same), so that N becomes $N[k]$ and $w[n]$ becomes $w[n,k]$. For a given frequency f_k, $N[k] = f_s/\Delta f_k = f_s Q_k/f_k$. The digital frequency of the kth component is then given by $2\pi Q/N[k]$, the period in samples is $N[k]/Q$ and always Q cycles for each frequency are analyzed. The expression for the kth spectral component of the CQT is then [1],

$$X^{cq}[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} w[n,k]x[n]e^{-j2\pi Qn/N[k]}. \quad (1)$$

Direct evaluation of equation (1) is time consuming, so an efficient algorithm for its computation has been proposed in [1]. The CQT can be expressed as a matrix multiplication, $X^{cq} = x \cdot T^*$, where x is the signal row vector of length N ($N \geq N[k]\ \forall k$) and T^* is the complex conjugate of the temporal kernel matrix T whose elements $T[n,k]$ are,

$$T[n,k] = \begin{cases} \frac{1}{N[k]} w[n,k]e^{-j2\pi Qn/N[k]} & \text{if } n < N[k] \\ 0 & \text{otherwise} \end{cases}$$

Computational effort can be improved if the matrix multiplication is carried out in the spectral domain. Using Parseval's relation for the DFT, the CQT can be expressed as,

$$X^{cq}[k] = \sum_{n=0}^{N-1} x[n]T^*[n,k] = \frac{1}{N}\sum_{k'=0}^{N-1} X[k']K^*[k',k] \quad (2)$$

where $X[k']$ and $K[k',\cdot]$ are the DFT of $x[n]$ and $T[n,\cdot]$ respectively. Spectral kernels are computed only once taking full advantage of the FFT. In the case of conjugate symmetric temporal kernels, the spectral kernels are real and near zero over most of the spectrum. For this reason, if only the spectral kernel values greater than a certain threshold are retained, there are few products involved in the evaluation of the CQT (almost negligible compared to the computation of the FFT of $x[n]$).

It is important to notice that although the original derivation of the CQT implies a geometrical distribution of frequency bins, it can be formulated using other spacing, for instance a constant separation. In the following, linear spacing is used to put all the compared algorithms under an unified framework.

[1] A normalization factor $1/N[k]$ must be introduced since the number of terms varies with k.

2.2 Multi-resolution FFT

A simple way to obtain multiple time-frequency resolutions is through the explicit calculation of the DFT using different frame lengths. In [2], an efficient technique is proposed where the DFT using several frame lengths is computed by means of the combination of the DFT of small number of samples, called elementary transforms. The idea arises from the observation that a transform of frame length N can be split into partial sums of L terms (assuming $N/L \in \mathbb{N}$),

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-\frac{j2\pi kn}{N}} = \sum_{c=0}^{\frac{N}{L}-1} \sum_{n=cL}^{(c+1)L-1} x[n]e^{-\frac{j2\pi kn}{N}}. \quad (3)$$

Each inner sum in equation 3 corresponds to the DFT of length N of a sequence $x_c[n]$, where $x_c[n]$ is an L samples chunk of $x[n]$, time-shifted and zero padded,

$$x_c[n] = \begin{cases} x[n], & cL \leq n < (c+1)L \\ 0, & \text{otherwise.} \end{cases}$$

So, it is possible to obtain a DFT of a frame of size N from N/L elementary transforms of frame size L, defined as

$$X_l[k] = \sum_{n=0}^{L-1} x[n + lL]e^{-j2\pi kn/N},\ l = 0, ..., \frac{N}{L} - 1.$$

To that end, it is enough to add the elementary transforms modified with a linear phase shift to include the time shift of $x_c[n]$, as stated by the shifting theorem of the DFT,

$$X[k] = \sum_{l=0}^{\frac{N}{L}-1} X_l[k]e^{-j2\pi kl/N}. \quad (4)$$

This procedure can be generalized to compute the DFT of any frame of length $M = rL$ by adding r elementary transforms ($r = 1, ..., N/L$) in the equation 4, which results in N/L possibles spectral representations with frequency resolutions of $f_s/(rL)$.

The computation of the multi-resolution spectrum from a combination of elementary transforms requires the windowing process to be done by means of convolution product in the frequency domain. Temporal windows of the form

$$w[n] = \sum_{m=0}^{\frac{M}{2}} (-1)^m a_m \cos\left(\frac{2\pi}{M}mn\right) \quad (5)$$

are suitable for this purpose because its spectrum has only few non-zero samples. Due to the fact that windowing is applied over zero-padded transforms, it is convenient to consider a periodic time window of the same length of the DFT to avoid the appearance of new non-zero samples of the window spectrum. In this case, the spectrum of a window of the form of equation 5 results in

$$W[k] = \sum_{m=0}^{\frac{M}{2}} (-1)^m \frac{a_m}{2} \left(\delta\left[k - m\frac{N}{M}\right] + \delta\left[k + m\frac{N}{M}\right]\right)$$

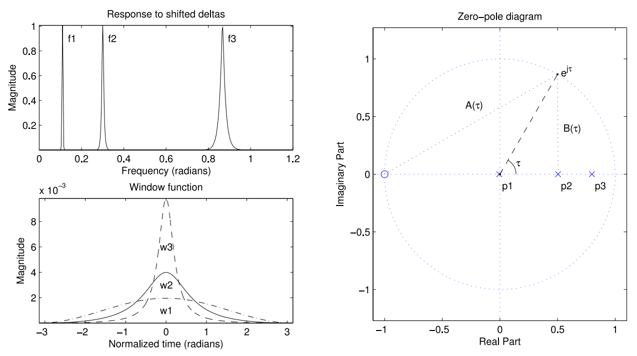


Figure 1. Zero-Pole diagram and IIR filters responses for three different input sinusoids of frequencies $f_1 = 0.11$, $f_2 = 0.30$ and $f_3 = 0.86$ radians.

For example, in Hann and Hamming windows only a_0 and a_1 are not zero and so its DFT contains solely three non-zero samples. As a counterpart, the restriction that $N/M = N/(rL) \in \mathbb{N}$ must be imposed, reducing the possible number of resolutions to $\log_2(N/L) + 1$.

3. IIR Q TRANSFORM

3.1 FIR/IIR Filterbank

The proposed methods define a Finite Impulse Response (FIR) filterbank with different impulse responses for different frequencies. The result of applying one of these filters can be regarded as multiplying the frame with a time window, which defines the time/frequency resolution. Variable windowing in time can also be achieved applying an IIR filterbank in the frequency domain. Let us define the k^{th} filter as a first order IIR filter with a pole p_k, and a zero z_k, as,

$$Y_k[n] = X[n] - z_k X[n-1] + p_k Y_k[n-1] \qquad (6)$$

Its Z transform is given by,

$$H_{f_k}(z) = \frac{z - z_k}{z - p_k}.$$

Here, $H_{f_k}(z)$ evaluated in the unit circle $z = e^{j\tau}$ represents its time response, with $\tau \in (-\pi, \pi]$ being the normalized time within the frame. A different time window for each frequency bin is obtained by selecting the value of the k^{th} bin as the output of the k^{th} filter.

The design of these filters involves finding the zero and pole for each k such that $w_k(\tau) = |H_{f_k}(e^{j\tau})|$, where $\tau \in (-\pi, \pi]$ and $w_k(\tau)$ is the desired window for the bin k. When a frame is analyzed, it is desirable to avoid discontinuities at its ends. This can be achieved by placing the zero in $\tau = \pi$, that is $z_k = -1$. If we are interested in a symmetric window, $w_k(\tau) = w_k(-\tau)$, the pole must be real. Considering a causal realization of the filter, p_k must be inside the unit circle to assure stability, thus $p_k \in (-1, 1)$. Figure 1 shows the frequency and time responses for the poles depicted in the zero-pole diagram.

This IIR filtering in frequency will also distort the phase, so a forward-backward filtering should be used to obtain a zero-phase filter response. Then, the set of possible windows that can be represented with these values of p_k is,

$$w_k(\tau) = \frac{(1-p_k)^2}{4}\left[\frac{A(\tau)}{B(\tau)}\right]^2 = \frac{(1-p_k)^2(1+\cos\tau)}{2(1+p_k^2 - 2p_k\cos\tau)} \qquad (7)$$

where $A(\tau)$ and $B(\tau)$ are the distances to the zero and the pole, as shown in Figure 1, and $g_k = (1-p_k)^2/4$ is a normalization factor [2] to have 0 dB gain at time $\tau = 0$, that is, $w_k(0) = 1$.

While this filter is linear and time invariant (in fact frequency invariant [3]) a different time window is desired for each frequency component. Computing the response of the whole bank of filters for the entire spectrum sequence and then choosing the response for only one bin is computationally inefficient. For this reason, a Linear Time Variant (LTV) system, that consists in a Time Varying (TV) IIR filter, is proposed as a way to approximate the filterbank response at the frequency bins of interest. It will no longer be possible to define the filter impulse response, as this could only be done if the filters were invariant to frequency shifts.

3.2 LTV IIR System

Selecting a different filter response of the filterbank for each frequency bin can be considered as applying an LTV system to the DFT of a frame. The desired response of the LTV for a given frequency bin is the impulse response of the correspondent filter.

Any LTV system can be expressed in the matrix form, $Y = K.X$ where K is the linear transformation matrix (also referred as Greens matrix) and, in this case, X is the DFT of the signal frame. A straightforward way to construct K for any LTV system is to set its i^{th} column as the response to a shifted delta $\delta[n-i]$, which is named Steady State Response (SSR).

The approach followed in this work consists in approximating the LTV system by a single TV IIR filter, assuming that the LTV system has a slow time varying behavior and that its SSR can be implemented by an IIR filterbank. Then it is verified that the approximation is sufficiently good for our purposes. In the case of variable windowing to obtain a constant Q, these assumptions hold, as time windows for two consecutive frequency bins are intended to be very similar, and the LTV system can be implemented by an IIR filterbank as seen before.

A direct way of approximating the IIR filterbank is by a first order IIR of the form of equation 6, but in which the pole varies with frequency ($p = p[n]$),

$$Y[n] = X[n] + X[n-1] + p[n]Y[n-1]. \qquad (8)$$

With an appropriate design, it reasonably matches the desired LTV IIR filterbank response, and its implementation has low computational complexity.

[2] This normalization factor can be calculated from the impulse response evaluated at $n = 0$, or by the integral of the time window function.

[3] Note that we will use the usual time domain filtering terminology in spite of the fact that filtering is performed in the frequency domain.

3.3 Time Varying IIR filter design

A question that arises is how to design the TV IIR filter in order to have a close response to that of the LTV IIR filterbank. Several design criteria have been proposed in the literature [7], that may depend on the problem itself.

The TV IIR can also be represented by a matrix K_v in a similar way as the LTV filterbank, so the design can be done as in [7], by minimizing the normalized mean square error, $E = ||K - K_v||_2/||K||_2$. In this work, the adopted design criteria is to impose the windows behavior in time in order to obtain the desired constant Q. Then, the error is regarded as the difference between the desired Q and the effective obtained value. It becomes necessary to define an objective measure of Q. Usually the quality factor of a passband filter is defined as the ratio between the center frequency and the bandwidth at 3 dB gain drop. In our case the filtering is done in the frequency domain, so it is reasonable to measure Q in the time domain. Given that Q represents the number of cycles of an analyzed frequency component in the frame, it makes sense to define Q as the number of cycles within the window width at a certain gain drop, for example 3 dB. If τ'_k is the time at this drop for frequency f_k, $w_k(\tau'_k) = 10^{-\frac{3}{20}} w(0) \triangleq w'_k$, then $\tau'_k = Q/(2f_k)$. This definition allows the comparison of Q for methods with different window shapes. Note however, that a similar measure of Q can be formulated in the frequency domain.

In the proposed approach the first step is to design an IIR filterbank that accomplishes the constant Q behavior. Then, a TV IIR filter is devised based on the poles of the filterbank. Finally a fine tuning is performed to improve the steadiness of the Q value for the TV IIR filter. In the following section, this procedure is described in detail.

3.3.1 *Proposed design*

Following the definition of Q in time, the poles of the IIR filterbank can be calculated from equation 7 as the solution of a second order polynomial: $(2w'_k - \cos(\tau'_k) - 1)p_k^2 + (2 + 2\cos(\tau'_k) - 4w'_k\cos(\tau'_k))p_k + 2w'_k - \cos(\tau'_k) - 1 = 0$.

Then, a simple and effective design of the TV IIR filter consists in choosing for each frequency bin the corresponding pole of the IIR filterbank, that is $p[n] = p_k$, with $k = n$. The Q factors obtained with this approach are close to the desired constant value but with a slight linear drift. This result shows that the slow variation of the LTV system allows an approximation by a single TV IIR with a little deviation that can be easily compensated by adding the same slope to the desired Q value at each bin. Figure 3 shows the Q curve for the original and compensated designs.

Another design consideration is that for low frequencies a constant Q would imply a longer window support than the frame time. It becomes necessary to limit the time τ'_k to a maximum time τ_{max}, such that $2\,\tau_{max}$ is smaller than the frame time. This limitation of τ'_k to a maximum value must be done in a smooth way. Let $\bar{\tau}'_k$ be a new variable that represents the result of saturating τ'_k. The transition can be implemented with a hyperbola whose asymptotes are $\bar{\tau}'_k = \tau'_k$ and $\bar{\tau}'_k = \tau_{max}$, so that $(\bar{\tau}'_k - \tau_{max})(\bar{\tau}'_k - \tau'_k) = \delta$,

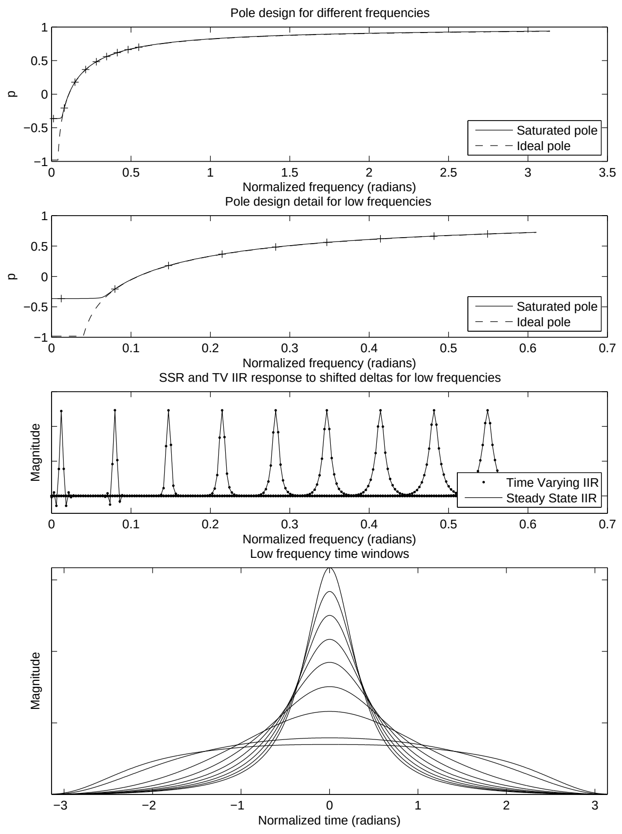


Figure 2. Detail of poles design. Pole locations for the ideal and saturated design. Impulse responses at low frequencies for the TV IIR and the Steady State, along with corresponding TV IIR time windows.

where δ is a constant that determines the smoothness of the transition.

The selection of τ_{max}, affects the behavior of the transform in low frequencies. Choosing a small τ_{max} compared to the frame time gives poor frequency resolution. On the contrary, if τ_{max} is set to a value close to the frame time, a better resolution is expected, but some distortion appears. This is because the time windows get close to a rectangular window for low frequencies. The spectrum of these windows has big side lobes, introducing Gibbs oscillations in the representation. Additionally, as a time window for low frequency approaches to a rectangular shape, its response to an impulse vanishes more slowly, so it becomes necessary to calculate the response for some negative frequency bins, adding extra complexity. In practice it is reasonable to choose an intermediate value of τ_{max}, e.g. $\tau_{max} \approx 0.7\pi$, such that only for very low frequencies the transform exhibits non constant Q. Figure 2 shows details of the described poles design.

3.3.2 *TV IIR filtering and zero-padding in time*

It is common practice to work with a higher sampling frequency of the spectrum, typically obtained by zero-padding in time. In this case the TV IIR filter design changes, as the signal support becomes $(-\tau_1, \tau_1]$ with $0 < \tau_1 < \pi$. Then, the discontinuity to be avoided at the ends of the frame ap-

```
p = design_poles(NFFT,Q);
X = fft(fftshift(s));
Y'(1) = X(1);
for i = 2:NFFT/2
    Y'(n) = X(n-1) + X(n) + p(n)Y'(n-1);
end
Y(n) = Y'(NFFT/2);
for i = NFFT/2-1:-1:1
    Y(n) = Y'(n+1) + Y'(n) + p(n)Y(n+1);
end
```

Table 1. Pseudocode of the TV IIR filter. First, the poles and normalization factor are designed given the number of bins (NFFT) and the Q value. Then the FFT of the signal frame s is computed after centering the signal at time 0. Finally the forward-backward TV IIR filtering is performed for that frame.

pears at $\pm\tau_1$, so a couple of zeros at $\pm\tau_1$ have to be placed instead of the zero at π. Window properties outside this support are irrelevant, as windowed data values are zero. The design of poles has to take into account the new zeroes and the time re-scaling, but windows with similar properties are obtained.

3.3.3 Implementation

The method implementation[4] is rather simple, as can be seen in the pseudocode of Table 1. A function to design the poles is called only once and then the forward-backward TV IIR filtering is applied to the DFT of each signal frame. The proposed IIR filtering applies a window centered at time 0, so the signal frame has to be centered before the transform. To avoid transients at the ends, the filtering should be done circularly using a few extra values of the spectrum as prefix and postfix. Their lengths can be chosen so as truncation error lies below a certain threshold, for instance 60 dB.

4. METHODS COMPARISON

4.1 Frequency scale

Depending on the context of the music analysis application different frequency grids may be preferred. To this respect, the efficient CQT method can be designed for any arbitrary frequency spacing. On the contrary, the MR FFT and the IIR CQT are constrained to a linear frequency scale because they rely on the DFT. This spacing typically implies an oversampling at high frequencies to conform with the minimum spacing at low frequencies.

4.2 Effective quality factor

The analyzed methods have different flexibility to define an arbitrary Q at each frequency. The efficient CQT offers the freedom to set any possible Q for every bin. The MR FFT allows choosing the resolution for every bin from a reduced set not enabling an arbitrary Q. On the other hand, the TV IIR filter allows any Q value for any frequency but with the constraint that it evolves slowly with frequency. This holds particularly well in the case of a constant Q transform, so the IIR CQT can give any constant Q with a fairly simple design. Figure 3 shows the obtained Q with the different methods. It can be observed that the MR FFT has a bounded Q due to the resolution quantization.

[4] The complete code is available at `http://iie.fing.edu.uy/~pcancela/iir-cqt`.

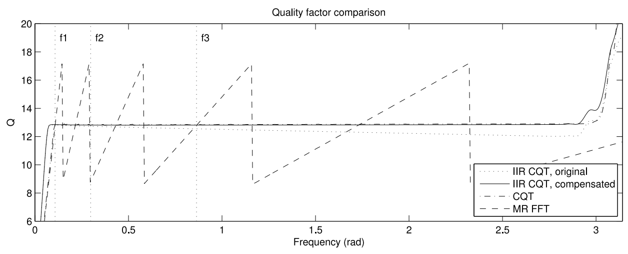


Figure 3. Comparison of the effective Q for a target value of 12.9 given the definition of 3.3. This value gives 34 cycles within the window, as commonly used in the CQT.

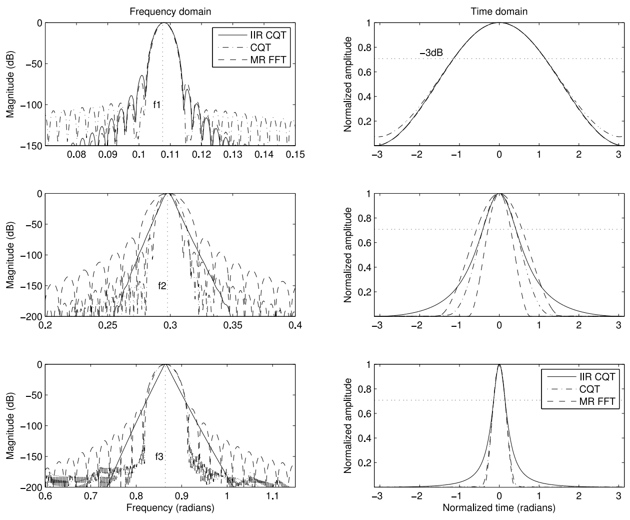


Figure 4. Windows comparison at frequencies f_1, f_2 and f_3 for the different methods. At f_1 and f_3 the three methods have the same Q, while at f_2 the MR FFT can not achieve the desired Q. For this reason, the two nearest MR FFT windows are considered at f_2. CQT and MR FFT are computed using a Hamming and Hann windows respectively.

4.3 Windows properties

The spectral and temporal characteristics of windows at three different frequencies are shown in Figure 4 for each method. At frequency f_1, IIR CQT time window behaves like a Hann window. For lower frequencies it exhibits a flatter shape to extend the range of constant Q (see Figure 2). For higher frequencies, the main lobe of the obtained windows has a steeper drop up to -50 dB compared to a conventional Hann or Hamming window. As a counterpart, time resolution is slightly diminished. Note that the selected drop value in the definition of Q sets the location in this compromise.

4.4 Computational complexity

The three algorithms are compared based on the number of real floating point operations performed in mean for each frequency bin. All of them compute the DFT of a non windowed frame, so these operations are not considered.

The number of operations in the efficient CQT depends on the length of the frequency kernels. This length varies with Q and is different for different frequency bins. For the Q and threshold values used in Figures 3 and 4 (Q_{CQT} = 34, Q = 12.9, th = 0.0054), `NFFT` = 2048 and $f_s = 44100$ Hz, the frequency kernel length varies from 1 to 57 coefficients, which implies a mean number of 27 real multiplications and 27 real additions. This result depends on the threshold and inversely on Q. The MR FFT takes advantage of the hierarchical implementation of the FFT to compute the transform, so the windowing in the frequency domain needs only 3 complex sums and 2 multiplications for each bin. The total number of real floating point operations is then, 4 multiplications and 6 additions. The IIR CQT involves a forward and backward IIR filtering with a variable real pole and a zero, followed by a real normalization (see Table 1 for a pseudocode). As the frequency components are complex values, the necessary number of real operations to compute each bin is 6 multiplications and 8 additions (plus a negligible number of extra operations due to the circularly filtering approximation).

5. APPLICATIONS AND RESULTS

Finally, two different examples of the spectral analysis of polyphonic music using the proposed IIR CQT method are shown in Figure 5 together with conventional spectrograms. As it is expected in a constant Q transform, it can be noticed that singing voice partials with high frequency slope tend to blur in the spectrogram but are sharper in the IIR CQT. This improved time resolution in high frequencies also contributes to define more precisely the note onsets, as can be seen in the second example (e.g. the bass note at the beginning). Moreover, in the low frequency band, where there is a higher density of components, the IIR CQT achieves a better discrimination, due to the fact that its time windows are flatter than typically used windows. At the same time, frequency resolution for the higher partials of notes with a steady pitch is deteriorated.

The proposed IIR CQT method was used as part of the spectral analysis front-end of a melody extraction algorithm submitted to the MIREX Audio Melody Extraction Contest 2008, performing best on Overall Accuracy [5]. Although the constant Q behavior of the spectral representation is just a small component of the algorithm, the results may indicate that the usage of the IIR CQT is appropriate.

6. CONCLUSIONS

In this work a novel method for computing a constant Q spectral transform is proposed and compared with two existing techniques. It shows to be a good compromise between the flexibility of the efficient CQT and the low computational cost of the MR FFT. Taking into account that it was used in the spectral analysis of music with encouraging results and that its implementation is rather simple, it seems to be a good spectral representation tool for audio signal analysis algorithms.

[5] The MIREX 2008 evaluation procedure and results are available at `http://www.music-ir.org/mirex/2008/index.php/Audio_Melody_Extraction`.

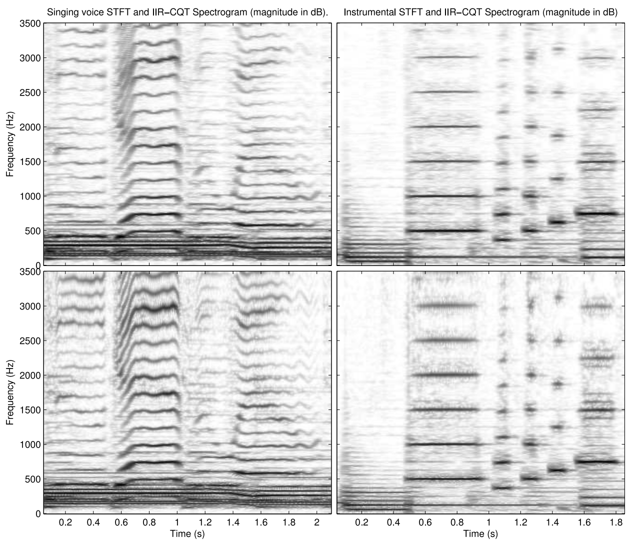


Figure 5. STFT and IIR CQT for two audio excerpts, one with a leading singing voice and the other, instrumental music.

7. REFERENCES

[1] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *JASA*, vol. 92, no. 5, pp. 2698–2701, 1992.

[2] K. Dressler, "Sinusoidal Extraction Using and Efficient Implementation of a Multi-Resolution FFT," in *Proceedings of the DAFx-06*, (Montreal, Canada), 2006.

[3] J. C. Brown, "Calculation of a constant Q spectral transform," *JASA*, vol. 89, no. 1, pp. 425–434, 1991.

[4] K. L. Kashima and B. Mont-Reynaud, "The bounded-Q approach to time-varying spectral analysis," Tech. Rep. STAN-M-28, Stanford University, 1985.

[5] F. C. C. B. Diniz, I. Kothe, S. L. Netto, and L. W. P. Biscainho, "High-Selectivity Filter Banks for Spectral Analysis of Music Signals," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, 2007.

[6] M. Goto, "A Real-time Music Scene Description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-world Audio Signals," *Speech Communication (ISCA Journal)*, vol. 43, no. 4, pp. 311–329, 2004.

[7] J. S. Prater and C. M. Loeffler, "Analysis and design of periodically time-varying IIR filters, with applications to transmultiplexing," *IEEE Transactions on Signal Processing*, vol. 40, no. 11, pp. 2715–2725, 1992.

EVALUATION OF MULTIPLE-F0 ESTIMATION AND TRACKING SYSTEMS

Mert Bay **Andreas F. Ehmann** **J. Stephen Downie**
International Music Information Retrieval Systems Evaluation Laboratory
University of Illinois at Urbana-Champaign
{mertbay, aehmann, jdownie}@illinois.edu

ABSTRACT

Multi-pitch estimation of sources in music is an ongoing research area that has a wealth of applications in music information retrieval systems. This paper presents the systematic evaluations of over a dozen competing methods and algorithms for extracting the fundamental frequencies of pitched sound sources in polyphonic music. The evaluations were carried out as part of the Music Information Retrieval Evaluation eXchange (MIREX) over the course of two years, from 2007 to 2008. The generation of the dataset and its corresponding ground-truth, the methods by which systems can be evaluated, and the evaluation results of the different systems are presented and discussed.

1. INTRODUCTION

A key aspect of many music information retrieval (MIR) systems is the ability to extract useful information from complex audio, which may then be used in a variety of user scenarios such as searching and organizing music collections. Among these extraction techniques, the goal of multiple fundamental frequency (multi-F0) estimation is to extract the fundamental frequencies of all (possibly concurrent) notes within a polyphonic musical piece. The extracted representations usually either take the form of a 1) list of pitches vs. time; or, 2) a MIDI-like representation that contains individual notes and their onset and offset times. These representations represent an intermediary between the audio and the score. While automatic transcriptions systems concern themselves with generating the actual score of music being analyzed, the intermediate representation generated by multi-F0 systems is useful in its own right. Such information can be very useful for other MIR systems as higher level features: to define the structure of the song, to make a better search or recommendation based on the score, or for F0-guided source separation. Recently, there has been great interest in multi-F0 estimation.

To understand the current state of art, starting in 2007, the MIREX [3] organized a multi-F0 evaluation task. This task can be considered as an evolution and superset of the previous MIREX audio melody extraction tasks. For more information on audio melody extraction, we refer the reader to [13]. The MIREX multiple-F0 task consists of two subtasks built around the two pitch representations mentioned earlier. The first subtask is called *Multiple-F0 Estimation* (MFE). In MFE, systems are required to return a list of active pitches at fixed time steps (analysis frames) of a polyphonic recording. The second subtask is called *Note Tracking* (NT). In the NT subtask, systems are required to return the note F0, onsets and offsets of note events in the polyphonic mixture, similar to a piano-roll representation.

The MIREX multiple-F0 task attracted many researchers from around the world. In the 2007 MFE subtask, there were a total of 16 algorithms from 12 labs. For the NT subtask, there were 11 algorithms from 7 labs. In 2008, there were a total of 15 algorithms from 10 labs for MFE and 13 algorithms from 8 labs for NT.

This paper serves to discuss the current performance of multi-F0 systems and to analyze the results of the MIREX algorithm evaluations. The paper is organized as follows. The rest of Section 1 describes the main approaches and challanges to MFE and NT. Section 2 describes the evaluation process. Section 2.1 describes the dataset and Section 2.2 defines the evaluation metrics. Section 3 discusses the results and some approaches from the MIREX 2007 and 2008 MFE and NT subtasks. Section 4 provides some concluding remarks.

1.1 An Overview of Multiple-F0 Estimation and Note Tracking Methods

There are many methods for F0 estimation and note tracking and an in-depth coverage of the many possible techniques is beyond the scope of this paper. Instead, we will provide a very brief overview of methods. Table 1 shows the participants of the MIREX 2007 and 2008 MFE and NT subtasks and their proposed methods. All systems use a time-frequency representation of the input signal as a front-end. The time-frequency representations include short-time Fourier transforms [1,2,6,10,11,13,15], auditory filter banks [16,17], wavelet decompositions [5] and sinusoidal analysis [18]. Characteristics of the spectrum such as harmonicity [5,10,14,17,19], spectral smoothness [11], onset synchronicity of harmonics [18] are often used to extract

F0s either by grouping harmonics together or calculating scores for different F0 hypotheses.

A large cross-section of techniques use nonnegative matrix factorization (NMF) to decompose the observed magnitude spectrum into a sparse basis. Fundamental frequencies can then be determined for each basis vector, and the onsets/offsets are computed from the amplitude weight of each basis throughout a piece. Some systems follow classification approaches which attempt to find pre-trained notes in the mixture. In general, it is possible to categorize the methods used into two groups in terms of how they approach polyphony. In the first group, systems extract F0s for the predominant source in the polyphonic mixture. The source is subsequently canceled or suppressed and the next predominant F0 is estimated. This procedure goes on iteratively until all sources are estimated. In the second group, systems attempt to estimate all F0s jointly.

2. EVALUATION

Extracting pitch information from polyphonic music is a difficult problem. This is why we choose to subdivide the task into the two MFE and NT subtasks. MFE defines a lower level representation for multiple-F0 systems. In this subtask, the systems estimate the F0s of active sources for each analysis frame.In many multi-F0 systems, frame-level F0 estimation is a precursor to the NT subtask. In the NT subtask, the systems are required to report the F0, onset and offset times of every note in the input mixture. Originally, additional timbre-tracking subtasks were envisioned for the MIREX multi-F0 task. Timbre tracking requires that the systems return the F0 contour and the notes of each individual source (e.g., oboe, flute, etc.) separately. However these subtasks were canceled due to lack of participation.

2.1 Creating the Dataset and the Ground-truth

The MIREX multi-F0 dataset consists both of recordings of a real-world performance and pieces generated from MIDI. The real-world performance is a recording of *L. van Beethoven Variations from String Quartet Op.18 N.5.* which is adapted and arranged for a woodwind quintet which consists of bassoon, clarinet, flute, horn and oboe. The piece was chosen due to its highly contrapuntal nature where the lines of each instrument are fairly different but sound harmonious when played together. Also, the predominant melodies alternate between instruments. The recording was done at the School of Music at the University of Illinois at Urbana-Champaign. First, the members of the quintet were recorded playing together where each performer was close mic‘ed. Second, each part was then recorded in complete isolation while the performer listened to and played along with the other parts previously recorded through headphones. The rerecording was done in isolation because there was significant bleed through of other sources into each instruments microphone during the ensemble recording. The MIREX 2007 dataset consisted of five different 30-second sections that were chosen from the nine minute recording. The MIREX 2008 data set added two more 30-second sections for a total of seven. The sections were chosen based on high activity of all sources. The isolated instruments from those sections were mixed to form mixtures starting from duet (two polyphony) to quintet (five polyphony). This results in four clips per section where each clip is generated by introducing an extra instrument to the mixture. There was no normalization during mixing, so each source‘s loudness in the mixture depends on how it was performed by the musician.

To create the ground-truth set, monophonic pitch detectors were used on the isolated instrument tracks using a 46 ms window and a 10 ms hop size. The pitch detectors used were *Wavesurfer*, *Praat* and *YIN*. The pitch contours generated were manually inspected and corrected by experts to get rid of common monophonic pitch detector errors such as voiced / unvoiced detection and octave errors. To create the ground-truth for the NT subtask, the isolated instrument recordings were annotated by hand to determine each note’s onset, offset and its F0 by inspecting the extracted monophonic pitch contour, the time domain amplitude envelope and the spectrogram of the recording.

The second, MIDI-based, portion of the dataset comes from two different sources. The first set was generated by [18] by creating monophonic tracks rendered and synthesized from MIDI files using real instrument samples from the RWC database [8]. The monophonic tracks were created such that no notes overlap so that each frame in the track is strictly monophonic. The ground-truth for MFE was extracted using *YIN*. The ground-truth for the NT subtask was generated using the MIDI file. Two 30-seconds sections with 4 clips from two to five polyphony were used from this data. The second set, which was used only for the note tracking subtask, was generated by [12] by recording a MIDI-controlled *Disklavier* playback piano. Two one-minute clips were used from this dataset for the note tracking subtask. The ground-truth was generated using the MIDI files.

2.2 Evaluation Methods and Metrics

This section describes the evaluation methods used in MIREX 2007 and 2008. The MFE and NT subtasks have different methods for evaluation.

2.2.1 Multi-F0 Estimation Evaluation

As mentioned earlier, the multi-F0 task represents a frame-level estimation of F0s where submitted systems were required to report active F0s every 10 ms. Many different metrics are used to evaluate this subtask. We begin by defining precision, recall and F-Measure as:

$$Precision = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t)} \tag{1}$$

$$Recall = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FN(t)} \tag{2}$$

$$\textit{F-measure} = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

Systems	Code	Front End	F0-Est Method	Note Tracking method	Ref
Cont	AC	STFT	NMF with sparsity constraints	NMF with sparsity constraints	[2]
Cao, Li	CL	STFT	Subharmonic sum, cancel-iterate	N/A	[1]
Yeh et al.	YRC	Sinusoidal an.	Joint Estimation based on spectral features	HMM tracking	[18]
Poliner, Ellis	PE	STFT	SVM classification	HMM tracking	[13]
Leveau	PL	Matching pursuit	Matching Pursuit with harmonic atoms	N/A	[10]
Raczyński et al.	SR	Constant-Q trans.	Harmonicity constrained NMF	N/A	[14]
Durrieu et al.	DRD	STFT	GMM source model, cancel-iterate	N/A	[4]
Emiya et al.	EBD	STFT	Derived from note tracking	HMM Tracking	[6]
Egashira et al.	EOS	Wavelets	Derived from note tracking	EM fit of Harmonic Temp. Models	[5]
Groble	STFT	MG	Scoring on pre-trained pitch models.	N/A	[9]
Pertusa, Iñesta	PI	STFT	Joint Estimation based on spectral features	Merge notes	[11]
Reis et al.	RFF	STFT	Derived from note tracking	Genetic Alg.	[15]
Ryynänen, Klapuri	RK	Auditory model	Derived from note tracking	HMM note and key models	[16]
Vincent et al.	EBD	ERB filter-bank	Derived from note tracking	Harmonicity constrained NMF	[17]
Zhou, Reiss	ZR	RTFI	N/A	Harmonic grouping, onset detection	[19]

Table 1. Summary of submitted multi-F0 and note tracking systems.

Since not all sources are active during any given analysis frame, the number of F0s in each time step of the ground-truth varies with time. For that reason, TP, FP and FN are defined as a function of time (frame index, t) as follows: "true positives" $TP(t)$ are calculated for frame t, based on the number F0s that correctly correspond between the ground-truth F0 set and the reported F0 set for that frame. "False positives" $FP(t)$ are calculated as the number of F0s detected that do not exist in the ground-truth set for that frame. The notion of "false negatives" $FN(t)$ however, becomes more problematic. We first begin by defining the notion of a negative. We define negatives based on the maximum polyphony of a each musical clip. Therefore, a quartet clip has a polyphony of four. Negatives in the ground-truth for each frame are calculated as the difference of the total polyphony and the number of F0s in the ground-truth. Similarly, the number of negatives for each frame in the reported F0 transcriptions are the difference between the total polyphony and the number of reported F0s. Therefore, the false negatives for each frame, $FN(t)$, is calculated as the difference between the number of reported negatives at frame t and the number of negatives in the ground-truth at frame t. Therefore, false negatives represent the number of active sources in the ground-truth that are not reported. The $TP(t)$,$FP(t)$ and $FN(t)$ are summed across all frames to calculate the total number of TPs, FPs and FNs for a given musical clip. From these measures, we can calculate an overall accuracy score as:

$$Accuracy = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t) + FN(t)} \qquad (4)$$

This is a measure of overall performance bounded between 0 and 1 where 1 corresponds to perfect transcription. However, it does not explain the types of errors that can happen. Therefore, we turn our attention to measures which better identify the types of errors multi-F0 systems make. We first note that not every instrument is active at every time frame. For example, an instrument in the mixture might be inactive through most of a piece's duration and active for only a relatively short amount of time. There are different kind of errors that can happen in estimating and reporting F0 candidates. An F0 of a source can be missed altogether, substituted with a different F0, or an extra F0 can be inserted ("false alarm" or false positive). To explain these types of errors, a measure called the frame-level transcription error score defined by [7] and used for music transcription by [12] is used. The benefit of this error measure is that this single error score can be decomposed into the three aforementioned types of errors, namely a miss, substitution, or false alarm. The total error score is defined as

$$E_{tot} = \frac{\sum_{t=1}^{T} \max(N_{ref}(t), N_{sys}(t)) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)} \qquad (5)$$

where $N_{ref}(t)$ is the number of F0s in the ground-truth list for frame t, $N_{sys}(t)$ is the number of reported F0s and $N_{corr}(t)$ is the number of correct F0s for that frame. This error counts the number of returned F0s that are not correct (they are either extra or substituted F0s) and the number of F0s that are missed. The total error is calculated by summing the frame level errors and normalizing by the the total number of F0s in the ground-truth. The maximum bound of this error score is directly correlated with the number of F0s returned. Not returning anything will result in a score of 1 while perfect transcription will yield a score of 0. However, the total error is not necessarily bounded by 1. This total error can be decomposed into the sum of three sub-errors. The substitution error is defined as

$$E_{sub} = \frac{\sum_{t=1}^{T} \min(N_{ref}(t), N_{sys}(t)) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)} \qquad (6)$$

The substitution error counts the number of ground-truth F0s for each frame that were not returned, but some other incorrect F0s were returned instead. These types of errors can be considered substitutions. This score is bounded between 0 and 1.

Missed errors are defined as

$$E_{miss} = \frac{\sum_{t=1}^{T} \max(0, N_{ref}(t) - N_{sys}(t))}{\sum_{t=1}^{T} N_{ref}(t)} \qquad (7)$$

which counts the number of F0s in the ground-truth that were missed by the system with no substitute F0s being returned. This error is also bounded between 0 and 1.

False alarms are defined as

$$E_{fa} = \frac{\sum_{t=1}^{T} \max(0, N_{sys}(t) - N_{ref}(t))}{\sum_{t=1}^{T} N_{ref}(t)} \tag{8}$$

which counts the number of extra F0s returned that are not substitutes. Every extra F0 after the number of F0s in the ground-truth list is counted as false alarm. The upper bound of this error depends on the number of F0s returned. All errors are normalized by the total number of F0s in the ground-truth. The error is good measure for this task because it enables us to explain different types of errors and can also provide a single measure for comparison.

2.2.2 *Note Tracking Evaluation*

In the note tracking subtask, systems are required to return a list of notes where each note is designated by its F0, onset and offset time. The evaluation of this subtask is more straightforward then the frame-level subtask. We can think of the ground-truth list as a fixed collection of events where each event is defined by three variables, F0, onset and offset. Due to the difficulty of detecting offsets in a highly polyphonic mixture, the evaluations were calculated using two different scenarios. In the first scenario, a returned note event is assumed to be correct if its onset is within a +/-50 millisecond range of a ground-truth onset and its F0 is within +/- a quarter tone (3%) of the ground-truth pitch. Here, the offset times are ignored. In the second scenario, in addition to the previous onset and pitch requirements, the correct returned note is required to have an offset time within 20% of ground-truth note's duration around the ground-truth note's offset value, or within 50 milliseconds of the ground-truth note's offset, whichever is larger. For these two cases, precision, recall and F-measure are calculated where true positives are defined as the returned notes that conform to the previously mentioned requirements and false positives were defined as the ones that do not. We also define an additional measure called Overlap Ratio (OR). The OR for a ith correct note in the returned list is defined as

$$\text{OR}_i = \frac{min(t_{i,off}^{ref}, t_{i,off}^{sys}) - max(t_{i,on}^{ref}, t_{i,on}^{sys})}{max(t_{i,off}^{ref}, t_{i,off}^{sys}) - min(t_{i,on}^{ref}, t_{i,on}^{sys})} \tag{9}$$

where $t_{i,off}^{sys}$ and $t_{i,on}^{sys}$ are the offset and the onset times of the correctly returned note and $t_{i,off}^{ref}$ and $t_{i,on}^{ref}$ are the offset and onset times of the corresponding ground-truth note. An average OR score is a good measure of how much the correct returned note overlaps with the corresponding ground-truth note. This information is especially useful when the correct notes are calculated based on the onset only.

3. RESULTS AND DISCUSSION

The evaluation results of two iterations of the MIREX multi-F0 estimation task (2007-2008) are presented here. We first turn our attention to the frame-level MFE subtask. Figure 1 shows the precision, recall, and accuracy scores for all submitted MFE systems over the two years. In general, systems have improved in accuracy over the course of the two years.

In Figure 2, a bar graph of the total error is shown for each of the systems. Each total error bar is subdivided into the three types of errors that constitute it namely, miss errors, substitution errors, and false alarm errors. It is evident that different systems present different trade-offs in terms of the types of errors. Referring back to Fig. 1, one can see that some systems have a very high precision compared to their accuracy such as those by PI, EBD and PE [6,11,13]. PI has the highest precision in both years. The reason behind this is that most of the F0s reported by these systems are correct, but they tend to under-report and miss a lot of active F0s in the ground-truth. This type of behavior is also evident in Fig. 2. While PI systems have the lowest total error score, there are very few false alarms compared to miss errors. PI achieves a low number of local false positives by taking into account a temporal salience of each combination of pitches. The results are post-processed by either merging/ignoring note events or using a weighted directed acyclic graph (wDAG).

Similarly, EBD and PE use hidden Markov models for temporal smoothing, and also have a relatively high miss error. RK [16] and YRC [18] have balanced precision, recall, as well as a balance in the three error types, and as a result, have the highest accuracies for MIREX 07 and MIREX 08, respectively. On the other hand, some systems like half of the CL submissions, have a high recall compared to their precision accuracy. CL returned a fixed (maximum) number of F0s for every frame regardless of the input polyphony in order to maximize recall.

The top two submissions share similar approaches. Both YRC and PI(1,2) generate a pool of candidate F0s for each frame and combine the candidates into hypotheses to jointly evaluate the present F0s. YRC first estimates an adaptive noise level, and extracts sinusoidal components. The algorithm then extracts F0 candidates until all the sinusoidal components are explained in the signal, as well as a polyphony inference stage that estimates the number of concurrent sources. All combinations of F0 candidates are evaluated by a score function based on smoothness and harmonicity, among others, and the best set is chosen. Finally, a tracking method is performed by first connecting F0 candidates across frames to establish candidate trajectories and then pruning them using HMMs. PI takes a similar approach in that, once again, joint F0 hypotheses are evaluated using saliency scores based on properties such as spectral smoothness and candidate loudness. Post-processing either takes into account local signal characteristics taken from adjacent frames or uses wDAGs for F0 note merging or pruning. The top performing algorithm from 2007, RK uses an auditory inspired model for anal-

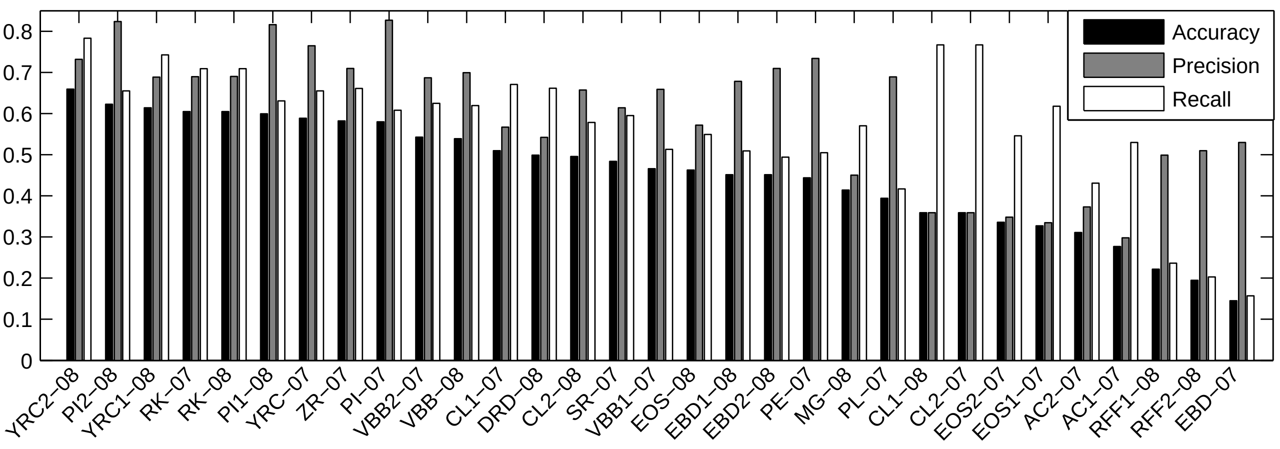


Figure 1. Precision, recall and accuracy for MIREX 07 and MIREX 08 MFE subtask ordered by accuracy.

ysis, and uses HMMs for note models and for note transitions, after a musical key estimation stage, in an attempt to incorporate some musicological information into the process.

For the NT subtask, Fig. 3 shows the precision, recall, and F-measures of the onset-offset based evaluation of the note tracking systems. We notice that in the NT onset-offset evaluation, performance is relatively poor. The likely explanation of this performance stems from the difficulty in properly defining an offset ground-truth in the data sets. In the woodwind data set, offset ground-truth was defined on the monophonic recordings of each track where the offset was labeled at very low loudness. Once mixed, other signals can dominate the low level of a source at the tail end of its decay such that the offset within the mixture is somewhat ambiguous. For the MIDI-generated piano dataset, offset is defined based on the MIDI file, and does not take into account the natural decay and the reverberation of the piano. Therefore, in the woodwind dataset, the offset time may be overestimated, whereas in the MIDI-generated dataset, the offset may be underestimated. Due to the inherent difficulty of properly defining offset, we also evaluate based strictly on note onset. The onset-based evaluation results of the NT subtask can be seen in Fig. 4. More detailed results and significance tests can be found at the MIREX wiki pages. [1]

4. CONCLUSION

Inspecting the methods used and their performances, we cannot make generalized claims as to what type of approach works best. In fact, statistical significance testing showed that the top three methods were not significantly different. However, systems that go beyond simple frame-level estimation methods and incorporate temporal constraints or other note tracking methods seem to perform better. It is plausible that timbral/instrument tracking can improve MFE even more. A future direction for evaluation would then be to add an instrument tracking subtask that

[1] http://www.music-ir.org/mirex/2007/index.php/MIREX2007_Results
http://www.music-ir.org/mirex/2008/index.php/MIREX2008_Results

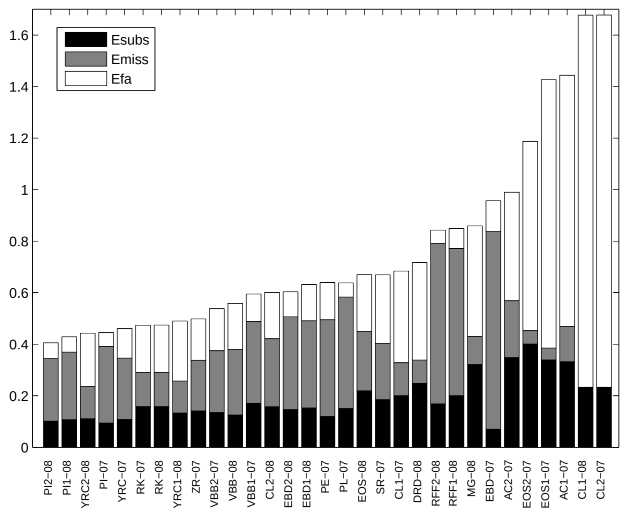


Figure 2. Error scores for MIREX 07 and MIREX 08 MFE subtask ordered by total error.

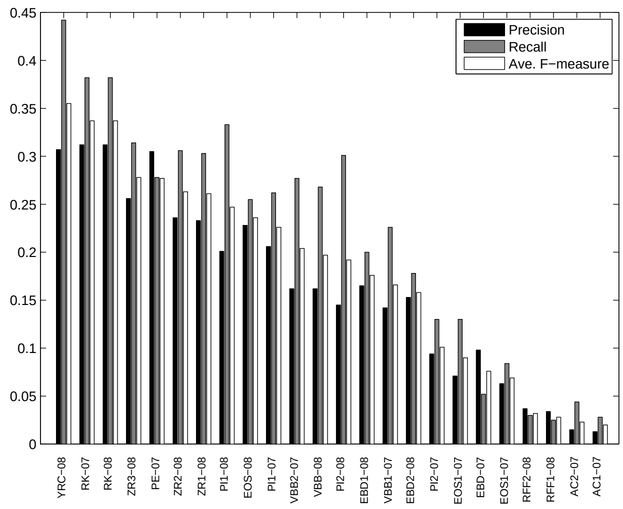


Figure 3. Precision, recall and F-measure based on note onset and offset for the MIREX 07 and MIREX 08 NT subtask.

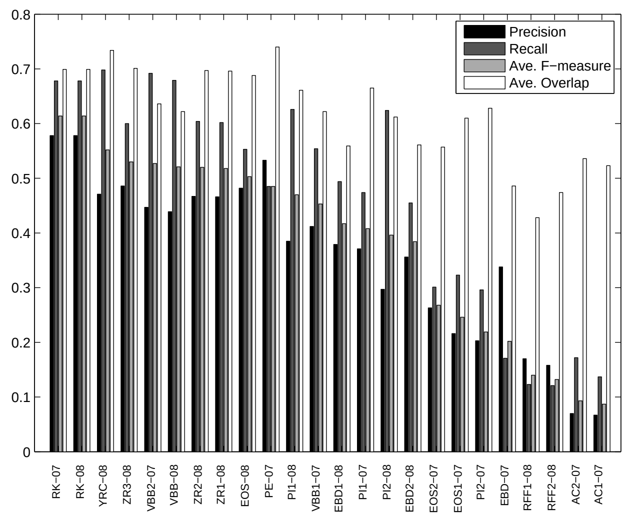


Figure 4. Precision, recall and F-measure based on note onset only for the MIREX 07 and MIREX 08 NT subtask.

would lead to a more complete music transcription task. The music transcription field is advancing but the problem is still far from being solved and there is a great room for improvement.

5. REFERENCES

[1] C. Cao and M. Li. Multiple F0 Estimation in Polyphonic Music, Avaliable at http://www.music-ir.org/mirex/2008/abs/mirex08_multiF0_Cao.pdf.

[2] A. Cont, S. Dubnov, and D. Wessel. Realtime multiple-pitch and multiple-instrument recognition for music signals using sparse non-negative constraints. In *Proceedings of the International Conference on Digital Audio Effects (DAFx). Bordeaux, France*, 2007.

[3] J.S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[4] J.L. Durrieu, G. Richard, and B. David. Singer melody extraction in polyphonic signals using source separation methods. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008*, pages 169–172, 2008.

[5] K. Egashira, N. Ono, and S. Sagayama. Sequential Estimation of Multiple Fundamental Frequencies Through Harmonic-Temporal-Structured Clustering, Avaliable at http://www.music-ir.org/mirex/2008/abs/F0_egashira.pdf.

[6] V. Emiya, R. Badeau, and B. David. Multipitch estimation of inharmonic sounds in colored noise. In *Proc. Int. Conf. Digital Audio Effects (DAFx), Bordeaux, France*, pages 93–98, 2007.

[7] J.G. Fiscus, N. Radde, J.S. Garofolo, A. Le, J. Ajot, and C. Laprun. The rich transcription 2005 spring meeting recognition evaluation. *Lecture Notes in Computer Science*, 3869:369, 2006.

[8] M. Goto. Development of the RWC music database. In *Proceedings of the 18th International Congress on Acoustics (ICA 2004)*, volume 1, pages 553–556, 2004.

[9] M. Groble. Multiple fundamental frequency estimation, Avaliable at http://www.music-ir.org/mirex/2008/abs/F0_groble.pdf.

[10] P. Leveau, D. Sodoyer, and L. Daudet. Automatic Instrument Recognition in a Polyphonic Mixture using Sparse Representations. In *Proc. of Int. Conf. on Music Information Retrieval (ISMIR), Vienne, Autriche*, 2007.

[11] A. Pertusa and J.M. Inesta. Multiple fundamental frequency estimation using Gaussian smoothness. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008*, pages 105–108, 2008.

[12] G.E. Poliner and D.P.W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007:1–9, 2007.

[13] G.E. Poliner, D.P.W. Ellis, A.F. Ehmann, E. Gomez, S. Streich, and B. Ong. Melody Transcription From Music-Audio: Approaches and Evaluation. *IEEE Transactions on Audio Speech and Language Processing*, 15(4):1247, 2007.

[14] S.A. Raczynski, N. Ono, and S. Sagayama. Multipitch analysis with harmonic nonnegative matrix approximation. In *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, pages 381–386, 2007.

[15] G. Reis, N. Fonseca, F.F. de Vega, and A. Ferreira. Hybrid Genetic Algorithm Based on Gene Fragment Competition for Polyphonic Music Transcription. *Lecture Notes in Computer Science*, 4974:305, 2008.

[16] M. P. Ryynanen and A. Klapuri. Polyphonic music transcription using note event modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005*, pages 319–322, 2005.

[17] E. Vincent, N. Bertin, and R. Badeau. Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008*, pages 109–112, 2008.

[18] C. Yeh. *Multiple fundamental frequency estimation of polyphonic recordings*. PhD thesis, Ph. D. dissertation, Universit Pierre et Marie Curie, Paris, Jun, 2008.

[19] R. Zhou and J.D. Reiss. A Real-Time Frame-Based Multiple Pitch Estimaiton Method Using The Resonator Time-Frequency Image, Avaliable at http://www.music-ir.org/mirex/2008/abs/F0_zhou.pdf.

We thank Andrew W. Mellon Foundation for their financial support.

SCALABILITY, GENERALITY AND TEMPORAL ASPECTS IN AUTOMATIC RECOGNITION OF PREDOMINANT MUSICAL INSTRUMENTS IN POLYPHONIC MUSIC

Ferdinand Fuhrmann, Martín Haro, Perfecto Herrera
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{ferdinand.fuhrmann,martin.haro,perfecto.herrera}@upf.edu

ABSTRACT

In this paper we present an approach towards the classification of pitched and unpitched instruments in polyphonic audio. In particular, the presented study accounts for three aspects currently lacking in literature: model scalability to polyphonic data, model generalisation in respect to the number of instruments, and incorporation of perceptual information. Therefore, our goal is a unifying recognition framework which enables the extraction of the main instruments' information. The applied methodology consists of training classifiers with audio descriptors, using extensive datasets to model the instruments sufficiently. All data consist of real world music, including categories of 11 pitched and 3 percussive instruments. We designed our descriptors by temporal integration of the raw feature values, which are directly extracted from the polyphonic data. Moreover, to evaluate the applicability of modelling temporal aspects in polyphonic audio, we studied the performance of different encodings of the temporal information. Along with accuracies of 63% and 78% for the pitched and percussive classification task, results show both the importance of temporal encoding as well as strong limitations of modelling it accurately.

1. INTRODUCTION

Instrument recognition is one of the big problems of current research in music information retrieval (MIR). Automatic indexing and retrieval of audio data are basic concepts to efficiently administrate and navigate through big datasets. Providing the information about the instrumentation of audio tracks via an automatic recognition system can highly facilitate these operations. Besides, such a system provides higher-level musical information, which helps to narrow the well-known semantic gap [1].

Computational recognition of musical instruments makes use of the intrinsic properties of, and differences between, each of the target categories. In the case of pitched instruments, where the sound is mostly composed of quasi-harmonic components, these are the amplitudes and frequency positions of the components and their evolution in time. The time-varying spectral envelope, an eminent feature for pitched instrument recognition [2], can be estimated out of them. For percussive instruments, properties such as attack and decay time, or frequency coverage, are properties which allow to distinguish between them [3]. While these specific characteristics can be determined without big problems in the case of a monophonic recording, the problem gets harder in polyphonic audio. Since the co-occurrence of multiple sound sources is producing overlapping frequency components, information extracted from the raw audio is often ambiguous and only partially useful for discriminating between several musical instruments. Without any preprocessing based on source separation, which is still not mature enough, models derived from simplified scenarios seem to imply strong limitations for the use on polyphonic audio. However, we hypothesize that, by providing a well suited dataset, a coarse – but MIR useful – modelling of predominant instruments directly from polyphonic audio is possible.

Below follows a short review of the current state of the art in computational musical instrument recognition. In Sec. 3 we substantiate our work and provide details about the general concepts we used for tackling the problem. Sec. 4 gives insights in the used data, the developed algorithms, and shows the experimental results. In the subsequent discussion we point out capacities as well as limitations of the chosen techniques and, finally, Sec. 6 concludes this article.

2. RELATED WORK

In current literature there exists a great unbalance between the amount of studies dealing with recognition of pitched instruments from polyphonic data and the amount of publications studying the monophonic case. Since the latter is not addressed in this paper, we refer to [4] for a comprehensive overview. Regarding the scarce publications addressing the more complex scenario, Kitahara et al. [5] presented a method to eliminate unreliable feature data caused by source inference for instrument recognition in artificial polyphonic mixtures. Linear discriminative analysis (LDA) was used to enhance features which discriminate best the five categories. The features were extracted from the harmonic structures of the corresponding instruments and used to train multivariate gaussian prototypes. Additional post-processing was applied by integrating the frame-wise a-posteriori probabilities and incorporating higher-level musical knowledge to get the final classification. In a more recent work, Every [6] manually annotated songs from a commercially available collection according to perceptually dominant instruments along with their corresponding pitches. A great set of audio descriptors was extracted from the raw audio in an unsupervised system, where the feature vectors were clustered and the resulting accuracies were measured. A strategy for tackling the problem at hand from a complete different direction was presented by Essid et al. [7]. Unlike trying to isolate the instruments present in the mixture, the whole audio was classified considering the more frequent combinations of them. Therefore, a suitable taxonomy was automatically generated by clustering a training corpus. Statistical models were built for each of the derived categories and used to classify unseen instances.

Regarding the recognition of percussive events in polyphonic music, work has focused on transcription of most common drum kit sounds (i.e. Bass Drum, Snare Drum and Hi-Hat sounds). For an excellent overview of studies on drum transcription up to 2006 see [3]. In a more recent work Paulus and Klapuri [8] evaluated a system based on Hidden Markov Models (HMM). In addition to standard spectral features, temporal features were derived from sub-band envelopes using 100 ms windows. Slight improvements in transcription accuracy were reported by incorporating this temporal information. Gillet and Richard [9] used source separation as preprocessing to obtain a drum-enhanced signal. A set of features was computed from both original and "enhanced" signals. Classification was derived from previously trained support vector machines (SVM) on an experimental database consisting of 28 songs from the ENST database (see Sec. 4.1 for an overview of this database).

Examining the literature review above we detect three main gaps in which we substantiate our present work. More precisely, we miss the aspect of *polyphonic scalability*, i.e. a detailed research about the application of current methods for instrument recognition to highly polyphonic audio. Second, there does not exist, to our knowledge, a study accounting for *instrument generality*, i.e. presenting a consistent methodology incorporating multiple instruments from different musical styles for tackling this problem. Finally, we see a clear need for incorporating *temporal characteristics* within the recognition process when working with statistical models, as this information is known to be important but often neglected.

3. CONCEPTUAL OVERVIEW

The present study is thought as a first step towards assessing the aforementioned gaps. We propose a methodology using statistical recognition techniques to build independent classification systems for 11 pitched and 3 unpitched instruments. Ground truth obtained from mostly manually created collections is used for training the models, gathered only from real world music. Additionally, we evaluate the importance and modelling accuracy of temporal aspects by comparing systems using different encodings of temporal information. What follows is a more detailed description of the concepts addressed within this work.

Polyphonic scalability. In this paper we are taking an approach of learning the time-frequency characteristics of musical instruments directly from polyphonic data. Our aim is to label a given audio excerpt with the name(s) of the most salient instrument(s). There exist some evidence that the performance of a recognition system is improved when the polyphonic context is incorporated into the training process [10]. As we focus on the application of a recognition algorithm on commercially available music, we introduce the least simplified conditions and work directly with real world data, all containing predominant instruments plus accompaniment.

Instrument generality. We included the recognition of pitched as well as unpitched (percussive) instruments in our study. As they imply obvious differences in their sound characteristics, both groups have to be treated in a slightly different way for computational processing. Percussive instruments produce a high energetic, impulsive sound and carry the main information in a relative short time interval (typically between 100 and 200 ms), whereas pitched instruments tend to have a quasi-harmonic and continuous tone ranging from very short to medium long durations (several seconds). Moreover, percussive sounds produce a spectrum in which their energy is scattered among the frequency bins, whereas pitched instruments have a frequency representation with peaks at quasi-integer multiples of their fundamental. Furthermore, a clear pitch dependency of the spectrum can be observed with pitched instruments unlike the more fixed spectral patterns of drum sounds.

Pitched Instruments: We consider an instrument to be "pitched" if it is able to produce a continuous, quasi-harmonic sound. Ten pitched instruments (Cello, Clarinet, Flute, acoustic and electric Guitar, Hammond Organ, Piano, Saxophone, Trumpet and Violin) are used in this study, being a good representation for most of the possible instrumentations in real world music of Western culture. We also include the human singing voice as an extra category in the corpus, as it can bee seen as frequently used pitched instrument in pop and rock music.

Unpitched Instruments: Due to the importance of the drum kit in Western popular music we decided to concentrate our research efforts on this particular set of percussive instruments. Likewise, because of the number of available instances and the musical relevance of each instrument within the drum kit, we work with the following, most common in literature, instrument classes: Bass Drum (BD), Snare Drum (SD) and Hi-Hat (HH).

Temporal characteristics. We incorporate temporal information in our statistical modelling process. According to experimental findings in literature, the human auditory system uses temporal aspects as an important cue for the recognition of musical instruments [2], but this information is often neglected in related studies. Together with quasi static properties of the sound, its evolution in time shapes the basics of human timbre perception. Therefore we directly compare different encodings of the temporal information in our statistical models. Doing that, we do not only examine the applicability of these aspects to computational musical instrument recognition. We also study how far they can be modelled directly from the polyphonic audio.

In [11] the modelling of temporal aspects was already analyzed by comparing the performance of a monophonic to a polyphonic similarity task. Timbre similarity was evaluated by both static systems, ignoring temporal aspects, and algorithms incorporating temporal behaviour. The used data consisted of isolated sound samples for the monophonic similarity task, and one song from The Beatles, segmented into its individual notes, for the polyphonic case. The authors concluded their work by stating that the frame-based analysis of polyphonic audio is not suited for modelling any temporal related properties. Moreover, as the performance of the dynamic systems was superior for the monophonic analysis and the same amount inferior for the polyphonic scenario, they identified the polyphony itself to be the root of all evil. However, we try to tackle this problem of polyphony by using big, diverse datasets and show that there still remain temporal aspects which can be modelled, if not for similarity retrieval, at least for sound source recognition.

4. METHOD

4.1 Data

A key concept for a successful modelling of musical instruments from polyphonic audio is the quality and the representativeness of the used data. We used two public available datasets to form the corpus for the recognition of percussive instruments, and developed our own collection for the pitched instrument identification task. Therefore we manually gathered sound samples from the three considered super-genres of Western music (jazz, classical and pop/rock), all extracted from commercial available recordings.

The objective for the creation of the dataset for the pitched instruments was to assemble excerpts of polyphonic audio in which the target instrument is playing continuously and is easily audible for a human listener. Each audio excerpt was then labelled with its predominating instrument (double-checked by two human experts), thus assigning more than one instrument to an audio excerpt was not allowed. After all, a corpus containing about 2,500 audio files was created, each one taken from a different recording. We tried to equally distribute the data among the three above-mentioned super-genres in order to cover most musical styles and combinations of instruments.

In the case of percussive instruments we used two publicly available collections with proper annotations of percussive events, namely the ENST-Drums database [12] and the MAMI database [13]. The first one is the largest publicly available drum database which provides "wet" and "dry" (see [12] for detailed information) drum tracks, as well as the respective accompaniment tracks. We decided to work with the "wet" drums and their accompaniment. From the obtained collection of 64 songs we randomly selected 30 second excerpts of every song and its labels. The MAMI database is a collection of 52 annotated music fragments extracted from commercial audio recordings. We managed to gather 48 songs and aligned them with the provided annotations. Finally, we mixed the ENST and the MAMI databases in order to have a representative database for training purposes. Thus we obtained a large set of polyphonic music excerpts adding up a total of 112 songs labelled with three, possibly concurrent, tags.

4.2 Algorithm Processing

Our approach towards assessing the information encoded by the different instruments and developing suitable models is based on classical pattern recognition techniques. First, we extract segments from the audio file containing the target instrument. For the drum recognition algorithm we generate excerpts based on onset detection: either we take a segment starting from the onset and lasting for 150 ms or, if the next onset falls within the following 150 ms, we take the inter-onset-interval. In the end, we include every so-generated excerpt in the dataset. For the pitched instruments we randomly extract a maximum of four 2.5 s long segments from each audio file. This grants a big amount of variability in the polyphonic background, which accompanies the main instrument. The length of 2.5 s was empirically determined and showed superior performance over shorter durations, whereas no significant improvement could be observed by using longer excerpts.

These segments are then framed with a fixed framesize of 46 ms and hopsize of 12 ms using a Blackman-Harris windowing function and audio features are extracted for every frame. We use a big amount of spectral, cepstral, and tonal features, all of them are well known audio descriptors and will not be discussed here. For a comprehensive overview of standard audio features we refer the interested reader to [14].

The frame-wise extraction results in a time series of feature vectors, consisting of the raw feature values. This two dimensional representation (features versus frames) is further processed by describing the evolution in time of each audio feature. We compute standard statistical measures like mean, variance from both the actual and the delta values, as well as more specific quantities accounting for the temporal information. The full set of the applied functions together with a short description is listed in Table 1. Finally, we derive one vector with a dimension of 2,023 representing the audio content of the extracted segment.

To decrease the complexity of the problem we perform feature selection on our data. For our experiments we search for the best subset of descriptors in the feature space, taking their correlation with the respective classes and their intercorrelation inside the subset into account [16]. We apply a 10 fold cross-validated feature selection to return

name	description
mean	mean of the values
var	variance of the values
dmean	mean of the delta values
dvar	variance of the delta values
max-norm-pos	location of the maximum
min-norm-pos	location of the minimum
attack	slope of the attack
decay	slope of the decay
slope	overall slope
t-centroid	temporal centroid of the values
t-skewness	temporal skewness of the values
t-kurtosis	temporal kurtosis of the values

Table 1. Applied functions to describe temporal aspects of the raw feature values. See [15] for details on their implementation.

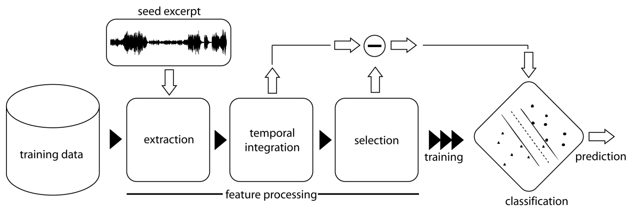


Figure 1. Block diagram of the training and recognition process. Black arrows indicate the training process while white ones show the prediction cycle. Note the decoupled modules of *extraction*, *temporal integration* and *selection* in the feature processing stage.

both a discriminative and compact set of descriptors. This procedure reduces the dimensionality of the vectors by a factor of 20, which significantly lowers the computation time of the following steps.

The feature vectors are then used to train SVMs, powerful classifiers for complex classification tasks. As the SVM is a binary classifier by definition, different strategies for combining the data and training the SVMs were tested to apply them to the multi-class problem. For our drum recognition system we utilized a balanced one–versus–all schema, where one SVM discriminates between the target category and an artificial one, consisting of a mixture of the remaining classes. Hence, each classifier determines the presence or absence of the respective class. In the case of pitched instruments we use a balanced one–versus–one algorithm with pair-wise coupling (PWC) [17], which performed superior than the one–versus–all approach in preceding experiments. Here, the final decision about the class membership is made by combining the output probabilities of all binary SVMs. The so generated models are then used to predict the labels of new data, represented as feature vectors. Fig. 1 shows an overview of the presented algorithm with a detailed view on the feature processing stage.

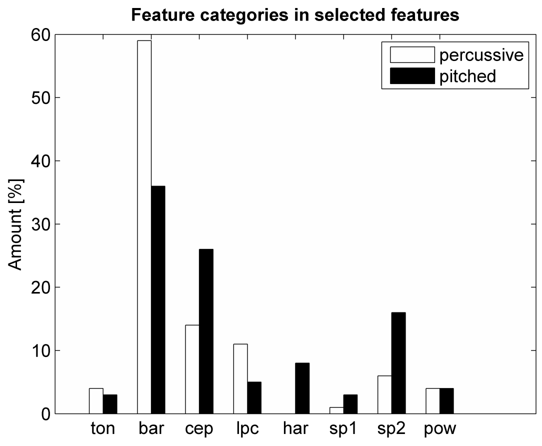


Figure 2. Relative occurrences of different feature categories in the final feature selection, applied to the full set of descriptors. The categories are derived in respect to the acoustic facets the features represent. See text for a detailed description.

4.3 Experiments and Results

First we evaluated the application of our features in the context of the pitched and the unpitched classification tasks. We grouped all selected descriptors in respect to the acoustic facets they represent: 8 categories were derived to evaluate the relative importance of the raw features when extracted from polyphonic data. In particular, the categories included *ton* (HPCPs, pitch salience), *bar* (Barkband energies), *cep* (MFCCs), *lpc* (LPCs), *har* (tristimuli, inharmonicity, odd2even), *sp1* (the four spectral moments), *sp2* (crest, rolloff,...), and *pow* (RMS, 3-bandenergies). Their relative occurrences are shown in Fig. 2. Furthermore, to assess the importance of temporal information encoded in the selected descriptors, we again grouped all of them into three new subsets. According to their modelling of the temporal information we derived the categories μ/σ^2 (only the average and deviation of the values), Δ (coarse encoding of the temporal characteristics in the delta descriptors), and time (detailed modelling of temporal aspects). Fig. 3 shows the results.

For the final evaluation of the recognition systems we split our data into two sets of 90 and 10% of their sizes. 10 fold cross validation was performed on the 90% dataset while the remaining 10% were used as an independent hold-out test set. Performance was measured by the resulting classification accuracy. Furthermore, to evaluate the effectiveness of the descriptors derived by the temporal integration of the raw feature values, we compared the performance of 3 different feature subsets. The first subset consisted of the full set of features, the second contained the average and the variance of both the actual and the delta feature values and subset 3 only included the mean and the variance of the raw values. Hence, we look at different encodings of the temporal information and their application for the recognition process. For all three groups we performed the above described feature selection procedure

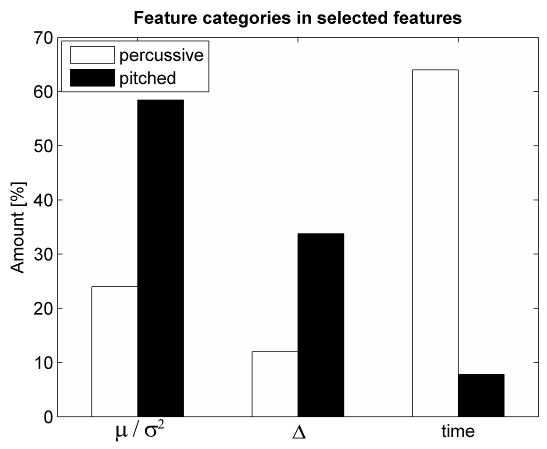


Figure 3. Relative occurrences of different descriptor categories in the final feature selection, applied to the full set of descriptors. The categories represent increasing encodings of the temporal information.

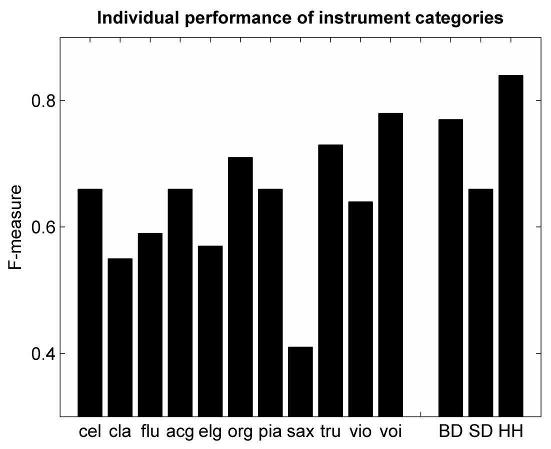


Figure 4. System performance in correctly identifying individual instrument categories. F-measures of the 10 Fold Cross Validation on the $\mu/\sigma^2 + \Delta$ feature subset.

data	full set	μ/σ^2+Δ	μ/σ^2
Pitched	63.1 / 50.3%	63.4 / 50.3%	61.1 / 47.2%
Drums	77.8 / 78.1%	77.6 / 82.6%	74.7 / 78%

Table 2. Performance comparison of different feature subsets with decreasing incorporation of temporal information. Accuracy of 10 fold cross validation (left values) and the hold-out set (right values) is shown (values for drums represent averaged individual accuracies).

before classification. The resulting accuracies can be seen in Table 2. Additionally, Fig. 4 provides details about the system performance in correctly identifying the individual classes on subset 2.

A binomial test [18, p. 37] revealed a significant difference between the cross-validated accuracies in the first two columns of Table 2 and those from the third column, for both pitched and unpitched instruments (p-value of the null hypothesis $\leq 10^{-3}$). Obviously, no statistical significance was found by comparing the first two columns for both recognition algorithms.

5. DISCUSSION

The above presented results show the capacities of the chosen approach as well as some clear limitations. By using a big, well suited dataset covering all relevant musical styles and a selected set of audio descriptors, the algorithm is able to learn the time-frequency characteristics of different musical instruments even from polyphonic data to a certain degree. This indicates that, although the target instrument is partly masked by various accompanying sounds, there still exist information in the audio data which is correlated with the main instrument. Moreover, our selected audio features can be used for extracting this intrinsic information from complex mixtures.

Looking at the results of the grouping experiment presented in Fig. 2 we can infer that both Barkband energies and cepstral features play a major role in discriminating between instrument classes. In particular, in the drum recognition the Barkbands form about 60% of all selected descriptors. This confirms that the spectral energy distribution is an important characteristic of percussive instruments. In the case of pitched instruments, the number of cepstral and spectral descriptors selected indicates the importance of the spectral envelope for recognition.

However, apart from the imbalance of categories (11 vs. 3), the performance differences between the pitched and percussive recognition indicate that the former task is more complex than the latter. This can also be derived from the fact that the characteristics of pitched instruments are more difficult to capture with the current audio features. As the required information is carried in a few frequency bins, a lot of noise due to overlapping components is incorporated into the feature values. Furthermore, percussive sounds generally carry more energy at the same time scale and therefore exhibit a more robust feature extraction.

Examining the performance of the individual instruments (Fig. 4), we can observe that the Snare Drum performs worse in respect to Bass Drum and Hi-Hat. As the latter ones only cover the very low and high frequency regions respectively, the Snare Drum has to compete with several other instruments in the same region, which degrades the systems' performance in correctly labelling Snare Drum sounds. Regarding pitched instruments, the weakness of the saxophone can be explained by its variety inside the class (e.g. Bass, Baritone, Tenor, and Alto), in contrary to other classes (e.g. hammond organ). Interestingly, the singing voice performs best among the pitched instruments, which was not expected. As an additional support of these observations, it is worth mentioning that similar results were obtained by the hold-out test set.

Nevertheless, compared to the human ability to recognize sounds, which is still the measure of all things, the results clearly indicate an inferior performance of our approach. First, the evaluation of the detailed temporal modelling of our audio features shows that, when extracted from polyphony, the resulting descriptors are not very discriminative between different instruments. We could not observe any improvement in performance when they were incorporated for the drum recognition task, even if a majority of the selected descriptors are describing fine temporal characteristics (see Fig. 3). Moreover, hardly any of these descriptors are selected for the pitched model. This implies that in the context of polyphony a detailed modelling becomes impossible for both short (percussive) and longer time-scale analysis (pitched), and that the remaining temporal aspects are best encoded in the coarse delta descriptors. That all strengthens the fact that temporal information is important for recognition but also shows the problems of modelling it accurately. These outcomes partly conform with the results presented in [11] by identifying fine temporal modelling of polyphonic audio as very fragile descriptors but proving the more coarse delta coefficients to be powerful, even when extracted from polyphonic data. Secondly, even if there would be some headroom for improvements, the algorithm will never be able to solve certain, dead-easy for humans, recognition tasks. Therefore we clearly see the need for different approaches in this area, starting from new audio representations to new algorithms for polyphonic processing. Enhanced signal processing as a front end system coupled with a complete probabilistic architecture (both bottom-up and top-down) could help to discover new paths, where an explicit source separation is not needed. Moreover, the integration of different knowledge sources could increase performance, as one solution might not always be applicable to all problems at hand.

6. CONCLUSIONS

In this paper we addressed three open gaps in automatic recognition of instruments from polyphonic audio. First we showed that by providing extensive, well designed datasets, statistical models are *scalable* to commercially available polyphonic music. Second, to account for *instrument generality*, we presented a consistent methodology for the recognition of 11 pitched and 3 percussive instruments in the main western genres classical, jazz and pop/rock. Finally, we examined the importance and modelling accuracy of *temporal characteristics* in combination with statistical models. Thereby we showed that modelling the temporal behaviour of raw audio features improves recognition performance, even though a detailed modelling is not possible. Results showed an average classification accuracy of 63% and 78% for the pitched and percussive recognition task, respectively. Although no complete system was presented, the developed algorithms could be easily incorporated into a robust recognition tool, able to index unseen data or label query songs according to the instrumentation.

ACKNOWLEDGEMENTS

The authors want to thank Joan Serrà and Emilia Gómez for their help in improving the quality and style of this publication. This research has been partially funded by the EU-IP project PHAROS [1] , IST-2006-045035.

7. REFERENCES

[1] X. Serra, R. Bresin, and A. Camurri, "Sound and music computing: Challenges and strategies," *Journal of New Music Research*, vol. 36, no. 3, pp. 185–190, 2007.

[2] S. McAdams, S. Winsberg, S. Donnadieu, G. DeSoete, and J. Krimphoff, "Perceptual scaling of synthesized musical timbres: common dimensions, specificities, and latent subject classes," *Psychological Research*, vol. 58, no. 3, pp. 177–192, 1995.

[3] D. FitzGerald and J. Paulus, "Unpitched percussion transcription," in *Signal Processing Methods for Music Transcription*, pp. 131–162, Springer, 2006.

[4] P. Herrera, A. Klapuri, and M. Davy, "Automatic classification of pitched musical instrument sounds," in *Signal Processing Methods for Music Transcription*, pp. 163–200, Springer, 2006.

[5] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. Okuno, "Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–16, 2007.

[6] M. Every, "Discriminating between pitched sources in music audio," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 267–277, 2008.

[7] S. Essid, G. Richard, and B. David, "Instrument recognition in polyphonic music based on automatic taxonomies," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 68–80, 2006.

[8] J. Paulus and A. Klapuri, "Combining temporal and spectral features in hmm-based drum transcription," in *Proc. of ISMIR*, 2007.

[9] O. Gillet and G. Richard, "Transcription and separation of drum signals from polyphonic music," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 16, no. 3, pp. 529–540, 2008.

[10] D. Little and B. Pardo, "Learning musical instruments from mixtures of audio with weak labels," in *Proc. of ISMIR*, 2008.

[11] J. Aucouturier and F. Pachet, "The influence of polyphony on the dynamical modelling of musical timbre," *Pattern Recognition Letters*, pp. 654–661, 2007.

[12] O. Gillet and G. Richard, "ENST-drums: an extensive audiovisual database for drum," in *Proc. of ISMIR*, 2006.

[13] K. Tanghe, M. Lesaffre, S. Degroeve, M. Leman, B. D. Baets, and J. Martens, "Collecting ground truth annotations for drum detection in polyphonic music," in *Proc. of ISMIR*, 2005.

[14] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," tech. rep., IRCAM, 2004.

[15] M. Haro, "Detecting and describing percussive events in polyphonic music," Master's thesis, Universitat Pompeu Fabra, Spain, 2008.

[16] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proc. of Int. Conf. on Machine Learning*, pp. 359–366, 2000.

[17] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," *Annals of Statistics*, pp. 451–471, 1998.

[18] P. H. Kvam and B. Vidakovic, *Nonparametric Statistics with Applications to Science and Engineering*. Wiley, 2007.

[1] `http://www.pharos-audiovisual-search.eu`

MUSICAL INSTRUMENT RECOGNITION IN POLYPHONIC AUDIO USING SOURCE-FILTER MODEL FOR SOUND SEPARATION

Toni Heittola, Anssi Klapuri and Tuomas Virtanen
Deparment of Signal Processing, Tampere University of Technology
toni.heittola@tut.fi, klap@cs.tut.fi, tuomas.virtanen@tut.fi

ABSTRACT

This paper proposes a novel approach to musical instrument recognition in polyphonic audio signals by using a source-filter model and an augmented non-negative matrix factorization algorithm for sound separation. The mixture signal is decomposed into a sum of spectral bases modeled as a product of excitations and filters. The excitations are restricted to harmonic spectra and their fundamental frequencies are estimated in advance using a multipitch estimator, whereas the filters are restricted to have smooth frequency responses by modeling them as a sum of elementary functions on the Mel-frequency scale. The pitch and timbre information are used in organizing individual notes into sound sources. In the recognition, Mel-frequency cepstral coefficients are used to represent the coarse shape of the power spectrum of sound sources and Gaussian mixture models are used to model instrument-conditional densities of the extracted features. The method is evaluated with polyphonic signals, randomly generated from 19 instrument classes. The recognition rate for signals having six note polyphony reaches 59%.

1. INTRODUCTION

The majority of research on the automatic recognition of musical instruments until now has been made on isolated notes or on excerpts from solo performances. A comprehensive review of proposed approaches on isolated note based recognition can be found in [1]. In recent years, there has been increasing research interest on more demanding and realistic multi-instrumental polyphonic audio. Most of the proposed techniques extract acoustic features directly from the signal, avoiding the source separation [2,3].

In polyphonic mixtures consisting of multiple instruments, the interference of simultaneously occurring sounds is likely to limit the recognition performance. The interference can be reduced by first separating the mixture into signals consisting of individual sound sources. In addition to the analysis of mixtures of sounds, sound source separation has applications in audio manipulation and object-based coding.

Many sound source separation algorithms aim at separating the most prominent harmonic sound from the mixture. Usually they first track the pitch of the target sound and then use the harmonic structure and sinusoidal modeling in the separation. A separation system based on this approach has been found to improve the accuracy of a singer identification in background music [4,5]. Sinusoidal components can also be grouped based on grouping cues such as common onset times, and the recognition can be done using the amplitudes of the grouped sinusoidal partials [6]. Instrument-specific harmonic models trained using instrument-specific material can achieve separation and recognition simultaneously [7].

Recently, many separation algorithms have been proposed which are based on matrix factorization of the mixture spectrogram. The methods approximate the magnitude $x_t(k)$ of the mixture spectrum in frame t and at frequency k as a weighted sum of basis functions as

$$\hat{x}_t(k) = \sum_{m=1}^{M} g_{m,t} b_m(k), \tag{1}$$

where $g_{m,t}$ is the gain of basis function m in frame t, and $b_m(k)$, $m = 1, \dots, M$ are the bases. This means that the signal is represented as a sum of components having a fixed spectrum and a time-varying gain. The decomposition can be done, e.g., using independent component analysis (ICA) or non-negative matrix factorization (NMF), the latter usually leading to a better separation quality [8]. The advantage of the methods is their ability to learn the spectral characteristics of each source from a mixture, enabling separation of sources which overlap in time and frequency. Instrument recognition systems based on the decomposition obtained with ICA have extracted the features from the estimated spectral basis vectors [9] or from the reconstructed time-domain signals [10].

A shortcoming of the basic spectrogram decompositions is that each pitch of each instrument has to be represented with a unique basis functions. This requires a large amount of basis functions, making the separation and classification difficult. Virtanen and Klapuri [11] proposed to model each spectral basis vector as a product of an excitation and a filter. The excitation models the time-varying pitch produced by a vibrating element such as a string, which can be shared between instruments, whereas the filter models

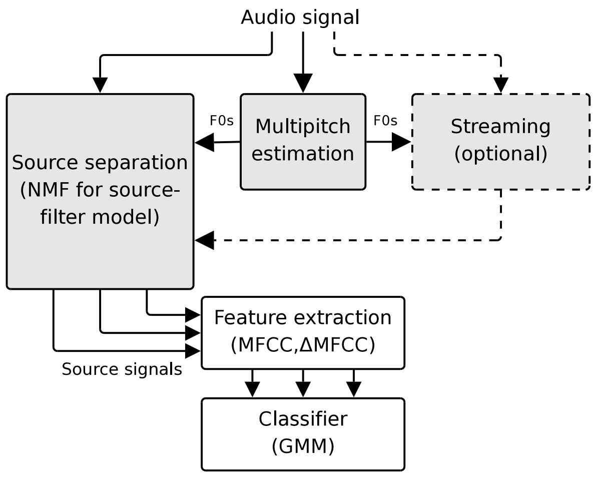


Figure 1. System overview.

the unique resonant structure of each instrument. To improve the performance of the method, FitzGerald proposed to model the excitations of different fundamental frequencies explicitly as a sum of sinusoids at the harmonic frequencies [12]. Badeau et al. [13] used also a harmonic model for the excitation, but they modeled the filter using a moving-average model, resulting in a smooth frequency response. Vincent et al. [14] modeled the spectral basis vectors as a weighted sum of harmonically constrained spectra having a limited frequency support. In this model the weights of each frequency band parametrized the rough spectral shape of the instrument.

In this paper, we present a novel approach to sound separation by using source-filter model in the context of musical instrument recognition. The mixture signal is decomposed into a sum of spectral bases modeled as a product of excitations and filters. The excitations are restricted to harmonic spectra and their fundamental frequencies are estimated in advance using a multiple pitch estimator, whereas the filters are restricted to have smooth frequency responses by modeling them as a sum of elementary functions on Mel-frequency scale. The pitch and timbre information are used in organizing individual notes into sound sources ("streaming"). Separated streams are recognized with a Gaussian mixture model (GMM) classifier. The system is evaluated with randomly mixed polyphonic signals using the Real World Computing (RWC) database [15] and sounds from 19 different instruments.

2. METHOD

An overview of the system is shown in Figure 1. Multipitch estimation is first employed to estimate the pitches in each analysis frame. The estimated pitches are used in the streaming algorithm to form temporally continuous streams of notes. Signals corresponding to individual sources are estimated using NMF for source-filter model. Features are extracted from the signals and they are classified using a GMM classifier. These processing steps are explained in detail in the following.

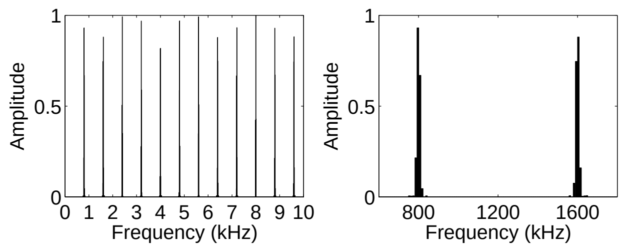


Figure 2. An example excitation spectrum $e_{n,t}(k)$ corresponding to pitch value 800 Hz. The entire spectrum is shown on the left and a closer view of a small portion of it on the right.

2.1 Signal model

In the proposed signal model, each basis $b_m(k)$ in (1) is expressed as a product of an excitation spectrum $e_{n,t}(k)$ and a filter $h_i(k)$. This leads to the model

$$\hat{x}_t(k) = \sum_{n=1}^{N}\sum_{i=1}^{I} g_{n,i,t} e_{n,t}(k) h_i(k) \tag{2}$$

for the magnitude $x_t(k)$ of the discrete Fourier transform $x_t(k)$ of Hamming-windowed signal in frame t. The excitations $e_{n,t}(k)$ are assumed to correspond to the pitch values of individual notes $n = 1, \ldots, N$ at times $t = 1, \ldots, T$, and the filters $h_i(k)$ are assumed to correspond to the spectral shapes of instruments $i = 1, \ldots, I$. We model only magnitude spectra of the excitations and filters, and therefore they restricted to non-negative real values. All combinations of excitations and filters are allowed, since we do not know in advance which instrument has produced which note. A polyphonic signal consists of several excitation and filter combinations occurring simultaneously or in sequence.

The excitations $e_{n,t}(k)$ are generated based on pitch values obtained from a multipitch estimator. For simplicity, we assume that the number of notes (pitch values) N is the same in all frames t. The multipitch estimator finds the pitches $F_t(n)$, $n = 1, \ldots, N$ in each frame t, and based on these, the corresponding excitation spectra $e_{n,t}(k)$ are calculated which consist of sinusoidal components with unity amplitudes at integer multiples of the corresponding pitch, $F_t(n)$. Figure 2 shows the excitation spectrum corresponding to pitch 800 Hz. Variation in amplitude appears in the figure since the partial frequencies do not fall exactly on spectral bins.

The filter $h_i(k)$ is further represented as a linear combination of fixed elementary responses:

$$h_i(k) = \sum_{j=1}^{J} c_{i,j} a_j(k) \tag{3}$$

where we chose the elementary responses $a_j(k)$ to consist of triangular bandpass magnitude responses, uniformly distributed on the Mel-frequency scale $f_{\mathrm{Mel}} = 2595\log_{10}(1+f_{\mathrm{Hz}}/700)$. The bases are illustrated in Fig. 3.

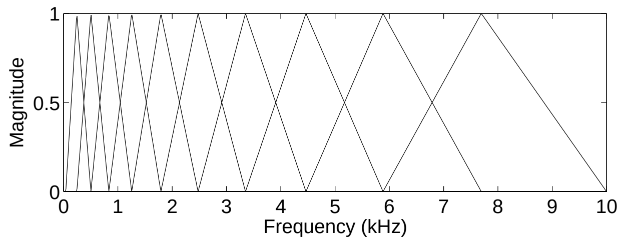


Figure 3. The elementary responses used to represent the filters $h_i(k)$. Triangular responses are uniformly distributed on the Mel-frequency scale.

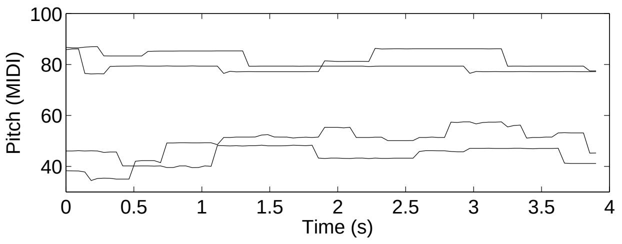


Figure 4. Output of the multipitch estimator for a mixture signal consisting of four simultaneous sounds.

Substituting (3) to (2) gives the final signal model

$$\hat{x}_t(k) = \sum_{n=1}^{N}\sum_{i=1}^{I} g_{n,i,t} e_{n,t}(k) \sum_{j=1}^{J} c_{i,j} a_j(k) \tag{4}$$

In this model, $e_{n,t}(k)$ are obtained as described above, $a_j(k)$ are fixed in advance, and therefore only $g_{n,i,t}$ and $c_{i,j}$ remain to be estimated using the proposed augmented NMF algorithm. The coefficients $c_{i,j}$ determine the spectral shape (filter) of instrument i, $i = 1, \ldots, I$, and the gains $g_{n,i,t}$ determine the amount of contribution from instrument i to note n at time t. Note that all instruments are allowed to play the same note simultaneously. Further constraints to associate each excitation with only one filter (instrument) are described in Sec. 2.3.

The amount of parameters to estimate is much smaller in the proposed model (4) than in the traditional model (1). This is because the traditional model practically requires a separate basis spectrum for each pitch/instrument combination. In the proposed model, the different notes coming from instrument i are represented by a single basis function (filter) $h_i(k)$, multiplied by the excitation spectra $e_{n,t}(k)$ to produce different pitch values. The smaller amount of parameters improves the reliability of the estimation. Furthermore, in the traditional model, the bases $b_m(k)$ have to be clustered to their respective sources after the estimation, whereas in the proposed model this takes place automatically.

2.2 Estimating the excitation spectra $e_{n,t}(k)$

The multipitch estimator proposed by Klapuri in [16] is used to estimate the note pitches $F_t(n)$, $n = 1, \ldots, N$ in each analysis frame t. Figure 4 illustrates the output of the multipitch estimator for a polyphonic signal consisting of four simultaneous sounds. Based on the pitch value $F_t(n)$, the corresponding excitation $e_{n,t}(k)$ is constructed which consists of Hamming-windowed sinusoidal components at integer multiples of the pitch value $F_t(n)$. These "harmonic combs" extend over the entire frequency range considered and have a unity magnitude for all the harmonics. An example excitation spectrum is shown in Figure 2.

2.3 Streaming algorithm to link excitations with filters

In the described model, all combinations of excitations and filters are allowed. In other words, all instruments (filters) i can play all the detected notes (excitations) n simultaneously. In a realistic situation, however, it is more likely that each note is played by one instrument and only occasionally two or more instruments play the same note.

Parameter $g_{n,i,t}$ controls the salience of each excitation and filter combination in each frame. Robustness of the parameter estimation can be improved if the excitations $e_{n,t}(k)$ can be tentatively organized into "streams", where a stream consists of the successive notes (excitations) coming from a same instrument (filter). Here stream i corresponds to the instrument i and we assume that the number of simultaneous notes N is equal to the number of filters I. The output of the streaming is a label sequence $\ell_t(n)$, where $\ell_t(n) = i$ indicates that the note (excitation) n at time t comes from instrument i. Even though the stream formation algorithm described here is imperfect, it is very helpful in initializing the augmented NMF algorithm that will be described in Sec. 2.4.

Let us introduce a state variable q_t that corresponds to a certain stream-labelling of the excitations $e_{n,t}(k)$, $n = 1, \ldots, N$ at time t. The number of different labellings (and states) is equal to $I!$, that is, the number of different permutations of numbers $1, \ldots, I$. For convenience, the different permutations of numbers $1, \ldots, I$ are stored as columns in a matrix $[\mathbf{U}]_{n,q}$ of size $(N \times I!)$.

A candidate solution to the streaming problem can be represented as a sequence of states $\mathbf{Q} = (q_1 q_2 \ldots q_T)$. The goodness of a candidate state sequence $\mathbf{Q}$ is defined so that it is proportional to the cumulative frame-to-frame variation of acoustic features extracted within each stream. Two types of acoustic feature vectors $\mathbf{z}_t(n)$ were investigated: pitch ($\mathbf{z}_t(n) \equiv F_t(n)$) and Mel-frequency cepstral coefficients (MFCCs) calculated from a spectrum that was constructed by picking only the spectral components corresponding to the harmonic partials of excitation n from the mixture spectrum $x_t(k)$. More exactly, the goodness Γ of a candidate solution $\mathbf{Q}$ given the features $\mathbf{z}_t(n)$ is defined by

$$\Gamma(\mathbf{Q}|\,\{\mathbf{z}_t\,(n)\}_{1:T,1:N}) = \gamma(q_1)\prod_{t=2}^{T}\gamma(q_t|q_{t-1}) \tag{5}$$

where the frame-to-frame feature similarity is calculated by

$$\gamma(q_t|q_{t-1}) = -\sum_{n=1}^{N} \left\| \mathbf{z}_t([\mathbf{U}]_{n,q_t}) - \mathbf{z}_{t-1}([\mathbf{U}]_{n,q_{t-1}}) \right\| . \tag{6}$$

The above goodness measure basically assumes that F0s of consecutive sounds coming from the same instrument usually have only small variations, or using MFCC features, that the spectral shapes of consecutive sounds from a same instrument are similar. Initial goodness values $\gamma(q_1)$ are defined to be zero for all other states except for $q_1 = 1$ for which we set $\gamma(q_1 = 1) = 1$. This removes the ambiguity related to the ordering of different streams.

The most likely sequence $\mathbf{Q} = (q_1 q_2 \ldots q_T)$ given the observed features $\mathbf{z}_t\,(n)$ is a search problem

$$\hat{\mathbf{Q}} = \arg\max_{\mathbf{Q}} \Gamma(\mathbf{Q}|\, \{\mathbf{z}_t\,(n)\}_{1:T,1:N}) \tag{7}$$

which can be straightforwardly solved using the Viterbi algorithm. The output of the streaming (associating each excitation with only one filter) is not fixed rigidly, but is used in initializing the NMF parameter estimation algorithm, as will be described below.

2.4 NMF algorithm for parameter estimation

The spectra $h_i(k)$ can be viewed as the magnitude responses of the filters, and therefore it is natural to restrict them to be entrywise non-negative. This is achieved using non-negative coefficients $c_{i,j}$. Furthermore, the model can be restricted to be purely additive by limiting the gains $g_{n,i,t}$ to be non-negative. NMF estimates the bases and their gains by minimizing the reconstruction error between the observed magnitude spectrogram $x_t(k)$ and the model $\hat{x}_t(k)$ while restricting the parameters to non-negative values.

Commonly used measures for the reconstruction error are the Euclidean distance, and divergence d, defined as

$$d(x, \hat{x}) = \sum_{k,t} x_t(k) \log \frac{x_t(k)}{\hat{x}_t(k)} - x_t(k) + \hat{x}_t(k) \tag{8}$$

The divergence is always non-negative, and zero only when $x_t(k) = \hat{x}_t(k)$ for all k and t. An algorithm that minimizes the divergence for the traditional signal model (1) has been proposed by Lee and Seung [17]. In their algorithm, the parameters are initialized to random non-negative values, and updated by applying multiplicative update rules iteratively. Each update decreases the value of the divergence.

We propose an augmented NMF algorithm for estimating the parameters of the model (4). Multiplicative updates which minimize the divergence (8) are given by

$$c_{i,j} \leftarrow c_{i,j} \frac{\sum_{n,t,k} r_t(k) g_{n,i,t} e_{n,t}(k) a_j(k)}{\sum_{n,t,k} g_{n,i,t} e_{n,t}(k) a_j(k)} \tag{9}$$

$$g_{n,i,t} \leftarrow g_{n,i,t} \frac{\sum_{j,k} r_t(k) e_{n,t}(k) c_{i,j} a_j(k)}{\sum_{j,k} e_{n,t}(k) c_{i,j} a_j(k)} \tag{10}$$

where $r_t(k) = \frac{x_t(k)}{\hat{x}_t(k)}$ is evaluated using (4) before each update. The overall estimation algorithm is given as follows:

1. Estimate excitations $e_{n,t}(k)$ using multipitch estimator and the procedure explained in Sec.2.2. Initialize the gains $g_{n,i,t}$ and the filter coefficients $c_{i,j}$ with absolute values of Gaussian noise.
2. Update the filter coefficients $c_{i,j}$ using (9).
3. Update the gains $g_{n,i,t}$ using (10)
4. Repeat steps 2-3 until the changes in parameters are sufficiently small.

In our experiments we observed that the divergence (8) is non-increasing under each update. If streaming is used, initial $g_{n,i,t}$ are multiplied with small factor 0.001 for n that do not belong to stream i. Using a small non-zero value favours the streamed excitations to be associated with the filter i, but this does not exclude the possibility of that the NMF algorithm will "correct" the streaming when the gains are updated during the algorithm. The streaming based on F0 values or MFCC values is far from perfect but yet improves the robustness of the parameter estimation with the NMF algorithm.

2.5 Reconstruction of instrument-wise spectrograms

Spectrograms corresponding to a certain instrument i can be reconstructed by using (4) and limiting the sum over i to one value only:

$$\hat{x}_{i,t}(k) = \sum_{n=1}^{N} g_{n,i,t} e_{n,t}(k) \sum_{j=1}^{J} c_{i,j} a_j(k) \tag{11}$$

Spectrogram of instrument i is reconstructed as

$$y_{i,t}(k) = \frac{\hat{x}_{i,t}(k)}{\hat{x}_t(k)} x_t(k) \tag{12}$$

where the denominator is calculated using (4) and summing over all i.

Time-domain signals are generated by using phases of the mixture signal and inverse discrete Fourier transform.

2.6 Classification

Mel-frequency cepstral coefficients (MFCC) are used to represent the coarse shape of the power spectrum of the separated instrument-wise signals. MFCCs are calculated from the outputs of a 40-channel filterbank which occupies the band from 30Hz to half the sampling rate. In addition to the static coefficients, their first time derivatives approximated with a three-point first-order polynomial fit are used to describe the dynamic properties of the cepstrum.

Gaussian mixture models are used to model instrument-conditional densities of the features. The parameters for the GMM are estimated with Expectation Maximization (EM) algorithm from the training material. Amount of Gaussian distributions in the mixture model was fixed to 32 for each class. In order to prevent acoustic mismatch between the training material and the testing material, models are trained with the separated signals. In the training stage, the perfect streaming is used with a prior knowledge about the sources in the signals. In the classification stage, likelihoods of the features are accumulated over the signal for the instrument classes, energy weighting the likelihoods with the RMS energy in the corresponding frame, and the classification is performed with maximum-likelihood classifier.

3. EXPERIMENTS

The proposed algorithm is evaluated in the musical instrument recognition task with generated polyphonic signals. The streaming algorithm is evaluated using both the pitch and the timbre information. "no separation" denotes a system where instrument recognition is done without separation directly from the mixture signal. "no streaming" denotes a system where the NMF is initialized (step 1 in Section 2.4) with random values $g_{n,i,t}$. "streaming (given F0s)" denotes system where time-varying pitches of sources were given in advance. "streaming (est. F0s)" denotes system where the pitches were estimated with the multipitch estimator and used in automatic streaming. "streaming (timbre)" uses timbre information in streaming and the F0s used for estimating the timbre were given in advance. Prior information of polyphony is used in all systems.

3.1 Acoustic Data

Polyphonic signals are generated by linearly mixing samples of isolated notes from the RWC musical instrument sound database. Nineteen instrument classes are selected for the evaluations (accordion, bassoon, clarinet, contrabass, electric bass, electric guitar, electric piano, flute, guitar, harmonica, horn, oboe, piano piccolo, recorder, saxophone, trombone, trumpet, tuba) and the instrument instances are randomized either into training (70%) or testing (30%) set. The polyphonic signals are generated from these sets, 500 cases for the training and 100 cases for the testing.

Four-second polyphonic signals are generated by randomly selecting instrument instances and generating random note sequences for them. For each instrument, the first note in a note sequence is taken randomly from the uniform distribution specified by the available notes in the RWC database for the instrument instance. The next notes in the sequence are taken from a normal distribution having a previous note as the mean and the standard deviation σ (being 6 semitones if not mentioned otherwise). Unisonal notes are excluded from the note sequence. The notes are randomly truncated to have length between 100 ms and one second. Signals from each instrument are mixed with equal mean-square levels. Examples of test signals are available at `www.cs.tut.fi/~heittolt/ismir09/`.

3.2 Evaluation Results

The separation quality was measured by comparing the separated signals with the reference ones. The signal-to-noise ratio (SNR) of a separated signal is estimated as

	Polyphony				
	2	3	4	5	6
no streaming	4.9	2.6	2.1	1.5	1.2
streaming (est. F0s)	7.2	4.0	2.5	1.7	1.2
streaming (timbre)	7.6	4.4	3.3	2.4	1.9

Table 1. Average signal-to-noise rations (in dB) for different system configurations.

	Polyphony					
	1	2	3	4	5	6
no separation	62.0	18.7	12.1	13.7	24.4	29.5
no streaming	62.0	49.5	42.1	42.6	39.3	42.7
streaming (given F0s)	62.0	59.0	58.0	57.9	57.8	56.0
streaming (est. F0s)	61.0	60.2	53.5	56.7	55.2	53.8
streaming (timbre)	62.0	57.6	51.9	57.0	55.9	59.1

Table 2. F-measures (%) for different system configurations.

$$SNR = 10\log_{10}\frac{\Sigma_t s(t)^2}{\Sigma_t (s(t)-\hat{s}(t))^2}, \tag{13}$$

where $s(t)$ is the reference signal and $\hat{s}(t)$is the separated signal. The average signal-to-noise ratios obtained for different system configurations are given in Table 1.

In instrument recognition, balanced F-measure is used as metric in the evaluations. The recall R is calculated as the ratio of correctly recognized instrument labels to sum of the correctly recognized instrument labels and unrecognized instrument labels. The precision P is calculated as the ratio of correctly recognized instrument labels to all instrument labels produced by the system. The F-measure is calculated from these two values as $F = 2RP/(R + P)$.

The evaluation results for different system configurations are given in Table 2. The system without separation uses the prior knowledge about the polyphony of the signal to find same amount of instruments directly from the mixture signal. This increases the random guess rate as the polyphony increases. The proposed approach using separation as a pre-processing gives rather steady performance regardless of the polyphony and gives reasonable performance already without the streaming algorithm. The streaming algorithm improves the results evenly, giving 10-15% increase in performance. The pitch and the timbre information based streaming gives same level of accuracy, though the pitch information seems to give slightly more robust performance. The estimated fundamental frequencies work almost as well as the given frequencies. The evaluation results for different types of polyphonic signals

	Polyphony					
σ	1	2	3	4	5	6
3	51.0	59.9	53.5	57.3	57.6	54.6
6	62.0	57.6	51.9	57.0	55.9	59.1
12	72.0	63.1	53.7	57.5	55.4	57.8

Table 3. F-measures (%) for different polyphonic signal conditions with the timbre based streaming.

are given in Table 3. The proposed system gives quite consistent results with all levels of the polyphony and when varying the standard deviation σ from 3 to 12 semitones. The slight variations in some cases are due to the randomization of used instruments for different polyphony levels.

4. CONCLUSIONS

In this paper, we proposed a source-filter model for sound separation and used it as a preprocessing step for musical instrument recognition in polyphonic music. The experimental results with the generated polyphonic signals were promising. The method gives good results when classifying into 19 instrument classes and with the high polyphony signals, implying a robust separation even with more complex signals. When recognizing the instrument from a sequence of several notes, it seems that the remaining slight separation artefacts average out to quite neutral noise, whereas the information related to the target instrument is consistent and leads to a robust recognition. Even when the F0s are estimated automatically, they provide sufficiently accurate information to get reasonable results. [1]

5. REFERENCES

[1] P. Herrera-Boyer, A. Klapuri, and M. Davy. Automatic classification of pitched musical instrument sounds. In *Signal Processing Methods for Music Transcription*, pages 163–200. Springer, 2006.

[2] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. Okuno. Musical instrument recognizer "instrogram" and its application to music retrieval based on instrumentation similarity. *International Symposium on Multimedia*, pages 265–274, 2006.

[3] S. Essid, G. Richard, and David.B. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions on Audio, Speech & Language Processing*, 14(1):68–80, 2006.

[4] H. Fujihara, T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Singer identification based on accompaniment sound reduction and reliable frame selection. In *Proc. ISMIR 2005*, pages 329–336, 2005.

[5] A Mesaros, T. Virtanen, and A. Klapuri. Singer identification and polyphonic music using vocal separation and pattern recognition methods. In *Proc. ISMIR*, pages 375–378, 2007.

[6] J. J. Burred, A. Röbel, and T. Sikora. Polyphonic musical instrument recognition based on a dynamic model of the spectral envelope. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 173–176, 2009.

[7] P. Leveau, E. Vincent, G. Richard, and L. Daudet. Instrument-specific harmonic atoms for mid-level music representation. *IEEE Transactions on Audio, Speech & Language Processing*, 16(1):116–128, 2008.

[8] T. Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Trans. Audio, Speech and Language Processing*, 15(3):1066–1074, 2007.

[9] P. Jincahitra. Polyphonic instrument identification using independent subspace analysis. In *Proceeding of the IEEE International Conference on Multimedia and Expo*, pages 1211–1214. IEEE, 2004.

[10] P. S. Lampropoulou, A. Lampropoulos, and G. Tsihrintzis. Musical instrument category discrimination using wavelet-based source separation. In *New Directions in Intelligent Interactive Multimedia*, pages 127–136. 2008.

[11] T. Virtanen and A. Klapuri. Analysis of polyphonic audio using source-filter model and non-negative matrix factorization. In *Advances in Models for Acoustic Processing, Neural Information Processing Systems Workshop*, 2006.

[12] D. FitzGerald, M. Cranitch, and E. Coyle. Extended nonnegative tensor factorisation models for musical source separation. *Computational Intelligence and Neuroscience*, 2008.

[13] R. Badeau, V. Emiya, and B. David. Expectation-maximization algorithm for multi-pitch estimation and separation of overlapping harmonic spectra. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009.

[14] E. Vincent, N. Bertin, and R. Badeau. Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 109–112, 2008.

[15] M. Goto, T. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Music genre database and musical instrument sound database. In *Proc. ISMIR*, pages 229–230, 2003.

[16] A. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proc. ISMIR*, pages 216–221, 2006.

[17] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, October 1999.

[1] This work was financially supported by the Academy of Finland.

HARMONICALLY INFORMED MULTI-PITCH TRACKING

Zhiyao Duan
Northwestern University
Evanston, IL, USA
zhiyaoduan00@gmail.com

Jinyu Han
Northwestern University
Evanston, IL, USA
jinyuhan@gmail.com

Bryan Pardo
Northwestern University
Evanston, IL USA
pardo@northwestern.edu

ABSTRACT

This paper presents a novel system for multi-pitch tracking, i.e. estimate the pitch trajectory of each monophonic source in a mixture of harmonic sounds. The system consists of two stages: multi-pitch estimation and pitch trajectory formation. In the first stage, we propose a new approach based on modeling spectral peaks and non-peak regions to estimate pitches and polyphony in each single frame. In the second stage, we view the pitch trajectory formation problem as a constrained clustering problem of pitch estimates in all the frames. Constraints are imposed on some pairs of pitch estimates, according to time and frequency proximity. In clustering, harmonic structure is employed as the feature. The proposed system is tested on 10 recorded four-part J. S. Bach chorales. Both multi-pitch estimation and tracking results are very promising. In addition, for multi-pitch estimation, the proposed system is shown to outperform a state-of-the-art multi-pitch estimation approach.

1. INTRODUCTION

Multi-pitch analysis is a fundamental research problems in music information retrieval (MIR). Pitch analysis results can provide helpful information for many other applications, such as automatic music transcription, audio source separation, content-based music search, etc. This task, however, remains challenging and existing methods do not match human ability in either accuracy or robustness.

Due to the complexity of multi-pitch analysis, researchers try to break it into different subproblems. *Multi-pitch Estimation (MPE)* [1, 2] usually refers to estimating pitches and the number of concurrent pitches (polyphony) in each single time frame. Based on MPE, *Note Formation* [3, 4] forms notes using pitch estimates in adjacent frames.

These two subproblems are important, however, they do not constitute the whole multi-pitch analysis problem. In a piece of polyphonic music consisting of several sound sources (usually different instruments), estimating pitches and finding the pitch trajectory for each underlying source, is a more advanced problem, which we call *Multi-pitch Tracking (MPT)*.

Multi-pitch tracking is similar to stream organization in Auditory Scene Analysis (ASA) [5], which is described as a grouping process of distinct acoustic events into a single perceptual entity, i.e. a stream. In [5], Bregman describes two main grouping processes in stream organization. *Simultaneous grouping* is to "integrate components that occur at the same time in different parts of the spectrum". *Sequential grouping* is to "put together events that follow one another in time". It is noted that MPE involves only simultaneous grouping, where harmonics of a sound source are integrated into a single pitch. Based on MPE, note formation also involves sequential grouping, where proximate pitches are integrated in a small time scale into notes. MPT also involves sequential grouping, but in a much larger time scale, i.e. the whole music piece.

Multi-pitch tracking is closely related to monaural source separation, which in fact estimates more information, i.e. the timbre of each source. A number of source separation systems [6–8] are built on multi-pitch estimation results. Other methods [9] usually train prior source models from solo excerpts. Those methods [10, 11] not relying on pitch estimation results nor prior source models utilize the same amount of information as MPT, but they are only tested on mixtures of two or three sounds.

Multi-pitch tracking is a difficult problem. Even for humans, only highly trained musicians have the ability to track concurrent pitch trajectories in listening. Therefore, some researchers proposed to only detect the main melody line and bass line of a polyphonic music [12, 13].

To our knowledge, few systems address the multi-pitch tracking problem. Kameoka et al. [14] proposed a multi-pitch analyzer based on harmonic temporal structured clustering that jointly estimates pitch, intensity, onset and duration of each underlying source. Chang et al. [15] presented a multi-pitch tracking system that and tracks pitches into note contours using Hidden Markov Models. Although these methods track concurrent pitch trajectories of multiple sound sources, they are only evaluated in the multi-pitch estimation and note formation level [14, 15]. Lagrange and Tzanetakis [16] proposed a sound source separation method that streams spectral peaks. It was not, however, evaluated on pitch tracking performance [16].

In this paper, we propose a system to address the MPT problem. Our system differs from previous systems in several ways. We start with multi-pitch estimation, where we

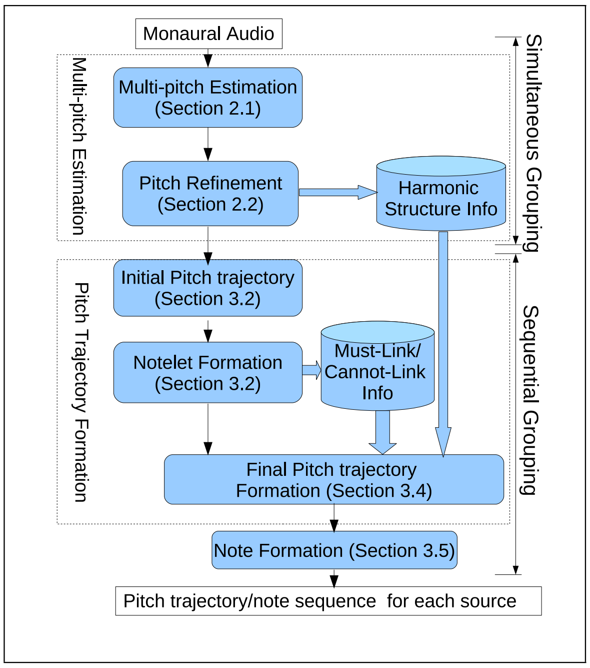

Monaural Audio
Multi-pitch Estimation
Multi-pitch Estimation (Section 2.1)
Pitch Refinement (Section 2.2)
Harmonic Structure Info
Simultaneous Grouping
Pitch Trajectory Formation
Initial Pitch trajectory (Section 3.2)
Notelet Formation (Section 3.2)
Must-Link/ Cannot-Link Info
Sequential Grouping
Final Pitch trajectory Formation (Section 3.4)
Note Formation (Section 3.5)
Pitch trajectory/note sequence for each source

Figure 1. System overview.

propose a new method that models spectral peaks and non-peak regions. Given pitch estimates for individual frames, we cast pitch trajectory formation as a constrained clustering problem, where each cluster corresponds to a trajectory. Harmonic structure is used as the feature of each pitch in clustering. Finally, note formation happens after pitch trajectories are formed, instead of forming notes and then placing them in streams, as previous systems have done.

Figure 1 shows the system structure. We address the MPT problem in two stages. The first stage is *multi-pitch estimation* (Section 2), where pitches and polyphony in each frame are estimated, and then refined using estimates in neighboring frames. The second stage is *pitch trajectory formation* (Section 3). Initial pitch trajectories are formed by grouping pitch estimates across frames according to pitch height. Within each initial trajectory, pitch estimates that are close in frequency and contiguous in time are grouped to form *notelets*. Final pitch trajectories are obtained through constrained clustering of pitch estimates, where must-link constraints are imposed on pitch pairs in each notelet and cannot-link constraints are imposed on pitch pairs of concurrent notelets. From the view of Auditory Scene Analysis, the first stage is simultaneous grouping and the second stage is sequential grouping.

2. MULTI-PITCH ESTIMATION

2.1 Multi-pitch Estimation In A Single Frame

In multi-pitch estimation, we break the music audio into 46 ms frames with a 10 ms hop size. We estimate polyphony and pitches in each time frame, but do not group estimates across adjacent frames into notes or trajectories. We view this problem (given polyphony N) as a Maximum Likelihood parameter estimation problem in the frequency domain. The parameters to be estimated are the set of pitches $\boldsymbol{\theta} = \{F_0^1, \cdots, F_0^N\}$. The observation is the spectrum, which is represented as spectral *peaks* and *non-peak regions*. Peaks are detected using the peak detection algorithm in [10]. The non-peak region is defined as the set of frequencies falling more than a quarter-tone from any observed peak.

For harmonic sounds, peaks typically appear only near integer multiples of F0s. We try to find the set of F0s with harmonics that maximize the probability of the occurrence of observed peaks, and minimize the probability that they have harmonics in non-peak regions.

Thus, the likelihood function can be defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{peak}}(\boldsymbol{\theta}) \cdot \mathcal{L}_{\text{non-peak region}}(\boldsymbol{\theta}) \qquad (1)$$

To model $\mathcal{L}_{\text{peak}}(\boldsymbol{\theta})$, each detected peak k is represented by its frequency f_k and amplitude a_k. We assume conditional independence between peaks, given a set of F0s. In addition, we consider the probability that a peak is *normal* (caused by some harmonic) or *spurious* (caused by other reasons). Then the peak likelihood is defined as

$$\begin{aligned}\mathcal{L}_{\text{peak}}(\boldsymbol{\theta}) &= \prod_{k=1}^{K} p\left(f_k, a_k|\boldsymbol{\theta}\right) &(2)\\ &= \prod_{k=1}^{K} \sum_{s_k} p\left(f_k, a_k|s_k, \boldsymbol{\theta}\right) P\left(s_k|\boldsymbol{\theta}\right) &(3)\end{aligned}$$

where K is the number of detected peaks; s_k is the binary variable to indicate whether the k-th peak is normal ($s_k = 0$) or spurious ($s_k = 1$).

In modeling $p\left(f_k, a_k|s_k = 0, \boldsymbol{\theta}\right)$, we notice that a normal peak may be generated by several F0s when their harmonics overlap at the peak position. In this case, however, the probability is conditioned on multiple F0s, which leads to a combinatorial problem in training we wish to avoid. Instead, we adopt the binary masking assumption [17], i.e. each peak is generated by only one F0, the one having the largest likelihood to generate the peak. Thus $p\left(f_k, a_k|s_k = 0, \boldsymbol{\theta}\right)$ is approximated by $\max_{F0 \in \boldsymbol{\theta}} p(f_k, a_k|F_0)$. For each $F_0 \in \boldsymbol{\theta}$, a harmonic number h_k is calculated as the nearest harmonic position of F_0 from f_k. Then,

$$\begin{aligned}p\left(f_k, a_k|F_0\right) &= p\left(f_k|F_0\right) p\left(a_k|f_k, F_0\right) &(4)\\ &= p\left(d_k|F_0\right) p\left(a_k|f_k, h_k\right) &(5)\end{aligned}$$

where $d_k = f_k - F_0 - 12\log_2 h_k$, is the frequency deviation of the k-th peak from its corresponding harmonic position in the logarithmic frequency domain.

In modeling $p\left(f_k, a_k|s_k = 1, \boldsymbol{\theta}\right)$, $\boldsymbol{\theta}$ can be ignored, since spurious peaks are assumed to be unrelated to F0s:

$$p\left(f_k, a_k|s_k = 1, \boldsymbol{\theta}\right) = p\left(f_k, a_k|s_k = 1\right) \qquad (6)$$

To model $\mathcal{L}_{\text{non-peak region}}(\boldsymbol{\theta})$, it is noted that instead of telling us where F0s or their ideal harmonics should be, the non-peak regions tell us where they should not be. A good set of F0s would predict as few harmonics as possible in the non-peak regions. Therefore, we define the non-peak region likelihood in terms of the probability of

From polyphonic (random chord) data	
$P(s_k\|\boldsymbol{\theta})$	approximated with $P(s_k)$, then learned as the proportion of spurious peaks
$p(d_k\|F_0)$	approximated with $p(d_k)$, then learned as a GMM from normal peaks
$p(a_k\|f_k, h_k)$	learned non-parametrically from normal peaks
$p(f_k, a_k\|s_k = 1)$	learned as a 2-D single Gaussian from spurious peaks
From monophonic (individual note) data	
$P(e_h = 1\|F_0)$	learned non-parametrically from normal peaks

Table 1. Model parameters that are learned from the training data.

not detecting any harmonic in the non-peak regions, given an assumed set of F0s. Again, with the conditional independence assumption of non-peak regions, the likelihood can be defined as:

$$\mathcal{L}_{\text{non-peak region}}(\boldsymbol{\theta}) = \prod_{F_0 \in \boldsymbol{\theta}} \prod_{\substack{hF_0 \in \mathcal{F}_{\text{np}} \\ h \in 1..H}} 1 - P(e_h = 1|F_0) \quad (7)$$

where N is the polyphony; e_h is a binary variable that indicates whether the h-th harmonic of F_0 is detected; $\mathcal{F}_{\text{np}}$ is the set of frequencies in the non-peak regions; H is the largest harmonic number we consider.

A harmonic not being detected in the non-peak region is because the corresponding peak in the source signal was weak or not present (e.g. even harmonics of clarinet). Therefore, this probability can be learned from monophonic training data, i.e. the monophonic notes used to generate the polyphonic training data.

The above probabilities are learned in monophonic or polyphonic training data, as described in Table 1. The monophonic training data are monophonic notes selected from the University of Iowa data set[1]. The polyphonic training data are randomly mixed chords of polyphony from 2 to 6, generated by mixing the above mentioned monophonic notes. Spectral peaks and non-peak regions are detected and collected in all the frames of the training data. Ground-truth F0s are obtained by YIN [18], a single pitch detection algorithm, on each note prior to mixing. The frequency deviation of each peak from the nearest harmonic of any ground-truth F0 is calculated. If the deviation is less than a quarter-tone, the peak is labeled normal, otherwise spurious. The "a quarter-tone" threshold is used here was selected in accordance with the standard tolerance used in measuring correctness of F0 estimation.

So far, given a set of F0s $\boldsymbol{\theta}$, its likelihood $\mathcal{L}(\boldsymbol{\theta})$ can be calculated in Eq. (1). The search space of the maximum likelihood solution $\hat{\boldsymbol{\theta}}$, however, is very large. We constrain this problem with a greedy search strategy. We start from an empty set $\hat{\boldsymbol{\theta}}^0$. In each iteration, we add an F0 estimate to $\hat{\boldsymbol{\theta}}^n$ to get a new set $\hat{\boldsymbol{\theta}}^{n+1}$ that gets maximum likelihood among all the possible values of the newly added F0. This iteration terminates when $n = N$, the given polyphony.

If the polyphony is not given, we need to decide when to terminate. Since $\mathcal{L}(\hat{\boldsymbol{\theta}}^n)$ increases with n, we propose a simple threshold-based method. The minimum number of F0s that achieves a likelihood greater than the threshold is returned as the polyphony estimate:

$$\begin{aligned} N = & \min_{1 \le n \le M} n, \\ s.t. \quad & \Delta(n) \ge T \cdot \Delta(M) \end{aligned} \quad (8)$$

where $\Delta(n) = \mathcal{L}(\hat{\boldsymbol{\theta}}^n) - \mathcal{L}(\hat{\boldsymbol{\theta}}^1)$ is the maximum increase of likelihood that could be achieved when the polyphony is set to be n. M is the maximum allowed polyphony which is set to 9 in all experiments; T is a learned threshold which is set to 0.88 (a value chosen by a machine learner using the monophonic and polyphonic training data). This polyphony estimation method works well on a large data set containing both music pieces and block musical chords with polyphony from 1 to 6.

2.2 Refine Pitch Estimates Using Neighboring Frames

There are often insertion, deletion and substitution errors in the multi-pitch estimation of a single frame. We propose a refinement method using estimates in neighboring frames: For each frame t, we build a weighted histogram in the frequency domain, where each bin corresponds to a semitone in the pitch range. Then, a triangular weighting function centered at t is imposed on a neighborhood of t, whose radius is r frames. The refined polyphony estimate N is calculated as the weighted average of polyphony estimates in all the frames in this neighborhood. The N bins with the highest histogram values are selected to reconstruct refined pitch estimates. For each of these bins, if there is an original pitch estimate in frame t that falls inside this bin, the original pitch estimate is used as the refined pitch estimate. Otherwise, the refined pitch estimate is calculated as the weighted average frequency of all the pitch estimates in this neighborhood that fall inside this bin. In our system, the radius r is set to 9 frames. This method removed a number of inconsistent estimation errors.

3. PITCH TRAJECTORY FORMATION

Given pitch estimates in all frames, we view pitch trajectory formation as a constrained clustering problem, where each *pitch trajectory* corresponds to a cluster.

3.1 Constrained Clustering

Constrained clustering [19,20] is a class of semi-supervised learning algorithms that make use of domain knowledge during clustering. Constraints can be imposed on different levels, where the instance level is the simplest and most common one. In instance-level constraints, there are two

[1] http://theremin.music.uiowa.edu/

basic forms: *must-link* and *cannot-link*. A must-link constraint specifies that two instances should be assigned to the same cluster. A cannot-link constraint specifies that two instances should not be assigned to the same cluster.

For our pitch trajectory formation problem, we adopt these two constraints. A must-link is imposed between two similar pitches in adjacent frames, since they probably belong to the same trajectory. A cannot-link is imposed between two pitches within a frame, due to our assumption that sources are monophonic. We then formulate the clustering problem to minimize the intra-class distance J, as the K-means algorithm does:

$$J = \sum_{k=1}^{K} \sum_{x_i \in T_k} \|\mathbf{x}_i - \mathbf{c}_k\|^2 \tag{9}$$

where K is the number of pitch trajectories; $\mathbf{x}_i$ is a feature vector in trajectory T_k and $\mathbf{c}_k$ is the mean feature vector in trajectory T_k; $\|\cdot\|$ denotes the Euclidean distance.

Wagstaff et al. [19] proposed a constrained K-means clustering algorithm, which iteratively changes the cluster labels of all points, without violating any constraint. For our multi-pitch tracking problem, however, this algorithm does not work. The reason is that almost every pitch estimate has must-links or cannot-links to other points, so it is almost impossible to change a pitch's label without violating any constraint. In addition, Davidson and Ravi [20] proved that finding a feasible solution, i.e. a label assignment without violating any constraint, of a clustering problem containing cannot-link constraints is NP-complete.

We now propose an iterative greedy algorithm that finds a low-cost (in terms of Eq. (9)) assignment of pitches to trajectories within a reasonable time.

3.2 Initial Pitch Trajectory Formation

Although the general problem is NP-complete, we can remove some constraints to make it tractable. A trivial example is to remove all constraints, where random label assignment can be a solution.

Instead of random assignment, we utilize the intrinsic structure of polyphonic music to assign initial labels. Note that pitch trajectories (e.g. melody and baseline) are roughly ordered in pitch, although sometimes they interweave. Since there are at most K pitches in each frame, we sort them from high to low and assign labels from 1 to K.

Then must-links are imposed on similar pitches that are in adjacent frames *and* have the same initial trajectory label. The maximal must-link difference between pitches in adjacent frames is set to 0.3 semitones (30 cents). Pitches connected by must-links form a short trajectory, which we call a *notelet*, since it is supposed to be some part of a note. Once notelets are formed, cannot-links are imposed between all pitches in two notelets that overlap more than 30ms. We say that two such notelets are in a *cannot-link relation*. We allow the 30ms overlap within a melodic line as it may be reverberation or ringing of a string. We chose conservative values for these parameters to ensure that they are reasonable for common real-world scenarios.

This results in an initial set of track assignments that is reasonably correct. By assigning trajectory labels on a frame-by-frame basis, assuming a fixed small number of trajectories, and only forming notelets from within sets of adjacent pitch estimates of the same trajectory, we greatly reduce the space searched. This provides a reasonable starting point and bypasses the NP-completeness problem.

Before showing how we improve on this initial solution by minimizing Eq. (9), we first explain the feature we use.

3.3 Harmonic Structure

Feature vectors in Eq. (9) should have the property that they are close in the same trajectory and far in different trajectories. Pitch height is not suitable because the underlying pitch estimates may show octave errors and the melodic lines overlap in range. Harmonic structure has been proven to be a good choice [10, 11] for music played by harmonic instruments, which is defined as the vector of relative amplitudes of harmonics of a pitch. Harmonic structures of the same instrument are similar, even if their pitches and loudness are different. On the other hand, different instruments usually have very different harmonic structures [10].

We calculate harmonic structure as follows: First, harmonics of each pitch are found from spectral peaks. For overlapping harmonics of different pitches, the peak likelihood in Eq. (4) is used to distribute energy to each pitch. Harmonic structures are then normalized to have the same total energy. The first fifty harmonics are used here.

3.4 Final Pitch Trajectory Formation

We start from the initial set of pitch trajectories created in Section 3.2, where pitches are assigned trajectory labels based on pitch height, and then placed into notelets based on time and pitch proximity. All pitch estimates comprising a notelet share the same trajectory label. All notelets that share a label form a *trajectory*. We now consider reassigning notelets to different trajectories to minimize the cost function in Eq. (9).

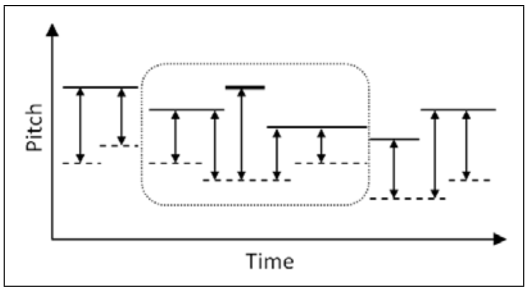


Figure 2. Illustration of a swap-set (the rounded rectangle) for a notelet (the bold solid line). Solid and dashed lines represent notelets in trajectory T_k and T_l, respectively. Cannot-link relations are indicated by arrows.

Suppose we want to change the trajectory of a notelet $\mathfrak{n}$ from T_k to T_l. We cannot do this in isolation, since there may be a notelet in T_l that overlaps $\mathfrak{n}$ and we assume monophonic pitch trajectories. We could simply swap the trajectories for two overlapping notelets. This, however may

```
For each notelet $\mathfrak{n}$, with trajectory $T_k$
    $J_0$ = cost of current trajectory assignments (Eq. (9))
    $J_{best} = J_0$
    For each trajectory $T_l$, such that $T_l \neq T_k$
        Find the swap-set between $T_l$ and $T_k$ containing $\mathfrak{n}$
        $J$ = (Eq. (9)) if we swap every notelet in the swap-set
        If $J < J_{best}$, $J_{best} = J$, End
    End
    If $J_{best} < J_0$, Perform swap that produces $J_{best}$, End
End
```

Table 2. The pitch trajectory formation algorithm.

cause a chain reaction, since the swap may cause new overlaps within a trajectory. Instead, we select two trajectories and pick a notelet $\mathfrak{n}$ from one of the trajectories. We then find all notelets in these two trajectories, T_k and T_l that connect to $\mathfrak{n}$ via a path of cannot-link relations (defined in Section 3.2). We call this the *swap-set*, as illustrated in Figure 2. These are the notelets affected by a potential trajectory swap between two notelets. All notelets in a swap set are swapped together, rather than individually, and the new trajectory are evaluated with the cost function in Eq. (9). Table 2 describes the process we use to swap trajectories for notelets until the trajectories of all notelets reach fixed points. In our experiment, this usually takes 3 to 4 rounds (where a round is a traversal of all notelets).

3.5 Note Formation

After pitch trajectories are formed, we form notes in each trajectory from the notelets. Two notelets are considered to be in the same note if the time gap between them is less than 100ms and their frequency difference is less than 0.3 semitone. Then the pitches in the gap are reconstructed using the average frequency of the note. Notes of length less than 100ms are considered spurious and removed. The 0.3 semitone parameter is the same as the one in imposing must-links. The 100ms parameter is set without tuning to adapt to the tempo and note lengths of the test music.

4. EXPERIMENT

The proposed system was tested on 10 real music performances, totaling 330 seconds of audio. Each performance was of a four-part Bach chorale, performed by a quartet of instruments: violin (Track 1), clarinet (Track 2), tenor saxophone (Track 3) and bassoon (Track 4). Each musician's part was recorded in isolation at 44.1 kHz, while the musician listened to the others through headphones. Audio mixtures were created by summing the four tracks. In testing, each piece was broken into 46 ms frames with center times spaced every 10 ms.

The ground-truth pitch trajectories of each testing piece were estimated using YIN [18] with manual corrections on monophonic sound tracks prior to mixture. The ground-truth notes (frequency, onset and offset) for each source were obtained by segmenting its pitch trajectory manually.

We evaluate the proposed system at the frame-level. For each estimated pitch trajectory, a pitch estimate in a frame

%	Initial		Final	
No.	Precision	Recall	Precision	Recall
1	66.9±3.2	66.9±3.2	**82.7±4.9**	**71.3±5.5**
2	52.3±6.5	52.1±6.5	**64.1±8.8**	**52.4±7.9**
3	61.0±8.5	59.3±8.7	**78.3±11.5**	**70.8±12.7**
4	81.8±5.0	65.3±7.4	**82.6±5.4**	**73.9±5.8**

Table 3. Frame-level evaluation results (Mean±Std) for each pitch trajectory. Track No. from 1 to 4 corresponds to the 4 parts of the quartets. "Initial" refers to the initial pitch trajectory formation (Section 3.2); "Final" refers to the final pitch trajectory formation (Section 3.4)

%	Precision	Recall
Klapuri06 [1]	87.2±2.0	66.2±3.4
Multi-pitch estimation (MPE)	84.9±1.7	**79.9±2.9**
Multi-pitch tracking (MPT)	**88.6±1.7**	77.0±3.5

Table 4. Frame-level evaluation results in the mixture instead of each trajectory.

is called correct if it deviates less than a quarter-tone from the pitch in the ground-truth pitch trajectory. Then precision and recall are calculated for each pitch trajectory by

$$\text{Precision} = \frac{\#\text{cor}}{\#\text{est}} \qquad \text{Recall} = \frac{\#\text{cor}}{\#\text{ref}} \tag{10}$$

where #cor, #est and #ref are the number of correctly estimated, estimated and reference pitches, respectively.

Table 3 presents the average frame-level evaluation results on the 10 pieces. The final tracking results are compared with the initial tracking results obtained from Section 3.2, which serves as a baseline obtained from multi-pitch estimation. For all four tracks, the proposed pitch trajectory formation method significantly improves either precision or recall from the baseline method. For Track 3, this improvement is up to 17.3% in precision and 11.5% in recall. In addition, in both initial and final tracking, results of Track 1 and 4 are better than Track 2 and 3. This is in accordance with previous researchers' observations that melody and baseline are easier to transcribe than other source streams [12].

Table 4 presents the frame-level evaluation results. We compare our system with Klapuri06 [1], a state-of-the-art multi-pitch estimation approach. We used Klapuri's code and suggested parameter settings. We can see that the best precision is obtained by MPT (Section 2+3), while the best recall is obtained by MPE (Section 2). Klapuri06 gets a high precision, which is indistinguishable from MPT, but its recall is much lower than both MPE and MPT. From MPE to MPT, precision has a significant improvement of 3.7%, while the two recalls are indistinguishable, considering the variances. It indicates that pitch trajectory formation improves the multi-pitch estimation results.

We also evaluate our system at the note-level in the mixture, as other researchers do [3,4,15]. It is evaluated in two ways. In the first way (Onset), a note estimate is correct if its frequency (average over all pitch estimates in this note) deviates less than a quarter-tone from a ground-truth note, and its onset time differs less than 50ms/100ms from the

%	Precision	Recall	AOR
Onset(50ms)	49.1±4.8	58.0±4.8	76.7±2.8
Onset(50ms)+Off	34.9±5.3	41.2±5.8	87.8±1.8
Onset(100ms)	65.5±4.7	77.4±3.1	73.7±3.1
Onset(100ms)+Off	46.0±5.5	54.3±5.5	85.1±2.3

Table 5. Note-level evaluation results in the mixture instead of each trajectory.

ground-truth onset time. The second way (Onset+Off) has the same requirements for frequency and onset time, but also requires that each offset time estimate deviate from the true offset time by less than 20% of the true note duration. Precision and recall are calculated by Eq. (10), where #cor, #est and #ref are the number of correctly estimated, estimated, and reference notes, respectively. *Average Overlap Ratio (AOR)* between correctly estimated notes and their ground-truth notes is calculated as

$$\text{AOR} = \text{Average}\left(\frac{\min\{\text{offsets}\} - \max\{\text{onsets}\}}{\max\{\text{offsets}\} - \min\{\text{onsets}\}}\right) \quad (11)$$

Given a correctly estimate note and its corresponding ground truth note, “onsets” is the set of onset times for both the estimated and the and the true note and ”offsets” is similarly defined. “Average” is over all correctly estimated notes.

Table 5 shows the note-level evaluation results. In Onset (100ms), both precision and recall are promising, and AOR value is high. This indicates that our system outputs good note estimates in both frequency and duration (offset subtracts onset). However, either reducing the onset threshold from 100ms to 50ms or adding the offset criterion decreases precision and recall significantly. This indicates that our system does not estimate the absolute onset/offset times precisely. This is not a surprise, since our current system does not have an onset/offset detection module.

5. CONCLUSION

Wee presented a novel system for multi-pitch tracking. Our system first estimates pitches and polyphony in each time frame. Then pitch trajectories are formed by constrained clustering pitch estimates across frames. Our system achieved promising results on ten real music pieces.

Currently, when clustering the pitches into trajectories, only harmonic structure is used. Future work includes incorporating musicological information into clustering.

This work was supported by National Science Foundation grant IIS-0643752.

6. REFERENCES

[1] A. Klapuri: “Multiple fundamental frequency estimation by summing harmonic amplitudes,” *Proc. ISMIR*, pp. 216-221, 2006.

[2] M. Davy, S. J. Godsill, and J. Idier: “Bayesian analysis of western tonal music,” *J. Acoust. Soc. Am.*, Vol. 119, No. 4, pp. 2498-2517, 2006.

[3] M. Ryynänen and A. Klapuri: “Polyphonic music transcription using note event modeling,” *Proc. WASPAA*, pp. 319-322, 2005.

[4] G. E. Poliner and D. Ellis: “A Discriminative Model for Polyphonic Piano Transcription” *EURASIP Journal on Advances in Signal Processing*, Vol. 2007, 2007.

[5] A. S. Bregman: *Auditory Scene Analysis*, Cambridge, MA: MIT Press, 1990.

[6] T. Virtanen and A. Klapuri: “Separation of harmonic sounds using multipitch analysis and iterative parameter estimation,” *Proc. WASPAA*, pp. 83-86, 2001.

[7] M. Bay and J. W. Beauchamp: “Harmonic source separation using prestored spectra,” *Proc. ICA*, pp. 561-568, 2006.

[8] J. Woodruff, Y. Li and D. Wang: “Resolving overlapping harmonics for monaural musical sound separation using pitch and common amplitude modulation,” *Proc. ISMIR*, 2008.

[9] E. Vincent: “Musical source separation using time-frequency source priors,” *IEEE Trans. Audio Speech Language Process.*, Vol. 14, No. 1, pp. 91- 98, 2006.

[10] Z. Duan, Y. Zhang, C. Zhang and Z. Shi: “Unsupervised single-channel music source separation by average harmonic structure modeling,” *IEEE Trans. Audio Speech Language Process.*, Vol. 16, No. 4, pp. 766-778, 2008.

[11] M. Kim, S. Choi: “Monaural music source separation: Nonnegativity, sparseness, and shift-invariance,” *Proc. ICA*, pp. 617-624, 2006.

[12] M. Goto: “A real-time music scene description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signal,” *Speech Communication*, Vol. 43, pp. 311-329, 2004.

[13] J. Eggink and G. J. Brown: “Extracting melody lines from comlex audio,” *Proc. ISMIR*, 2004.

[14] H. Kameoka, T. Nishimoto, and S. Sagayama: “A multipitch analyzer based on harmonic temporal structured clustering,” *IEEE Trans. on Audio, Speech and Language Processing*, Vol. 15, No. 3, pp. 982-994, 2007.

[15] W.-C. Chang, A. W.Y. Su, C. Yeh, A. Roebel and X. Rodet: “Multiple-F0 tracking based on a high-order HMM model,” *Proc. DAFx*, 2008.

[16] M. Lagrange and G. Tzanetakis: “Sound source tracking and formation using normalized cuts,” *Proc. ICASSP*, 2007.

[17] Ö. Yılmaz and S. Rickard: “Blind separation of speech mixtures via time-frequency masking,” *IEEE Trans. Signal Process.*, Vol. 52, No. 7, pp. 1830-1847, 2004.

[18] A. de Cheveigné and H. Kawahara: “YIN, a fundamental frequency estimator for speech and music,” *J. Acoust. Soc. Am.*, Vol. 111, pp. 1917-1930, 2002.

[19] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl: “Constrained K-means clustering with background knowledge,” *Proc. ICML*, 2001.

[20] I. Davidson and S. S. Ravi: “Clustering with constraints: feasibility issues and the k-means algorithm,” *Proc. SDM*, 2005.

CONTINUOUS PLSI AND SMOOTHING TECHNIQUES FOR HYBRID MUSIC RECOMMENDATION

Kazuyoshi Yoshii **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST)
{k.yoshii,m.goto}@aist.go.jp

ABSTRACT

This paper presents an extended probabilistic latent semantic indexing (pLSI) for hybrid music recommendation that deals with rating data provided by users and with content-based data extracted from audio signals. The original pLSI can be applied to collaborative filtering by treating users and items as discrete random variables that follow multinomial distributions. In hybrid recommendation, it is necessary to deal with musical contents that are usually represented as continuous vectorial values. To do this, we propose a continuous pLSI that incorporates Gaussian mixture models. This extension, however, causes a severe local optima problem because it increases the number of parameters drastically. This is considered to be a major factor generating "hubs," which are items that are inappropriately recommended to almost all users. To solve this problem, we tested three smoothing techniques: multinomial smoothing, Gaussian parameter tying, and artist-based item clustering. The experimental results revealed that although the first method improved nothing, the others significantly improved the recommendation accuracy and reduced the hubness. This indicates that it is important to appropriately limit the model complexity to use the pLSI in practical.

1. INTRODUCTION

The musical tastes of users of online music distribution services that provide millions of items are strongly influenced by the characteristics of the music automatically recommended by those services. Users often have difficulty retrieving unknown items they might like. In such case, users consider recommendations and get aware of what kinds of items are their favorites. When only popular items are always recommended, users are not exposed to items they might enjoy more and get used to enjoying only the "safe" recommendations. This in turn strengthens the tendency to recommend only popular items. In other words, there is a severe limitation in *serendipity* of music discovery. In fact, this negative-feedback phenomenon has been observed in many services based on collaborative filtering.

We aim to enhance the serendipity by transforming the

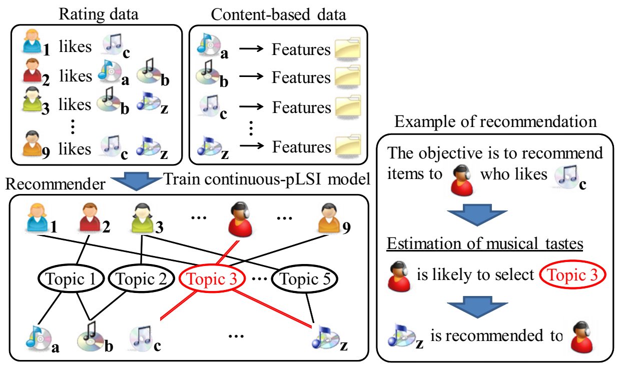


Figure 1. Hybrid recommender based on continuous pLSI.

passive experience in which users only receive "default" recommendations into an interactive experience in which users can freely customize (personalize) those recommendations. To achieve this, it is necessary to let users clearly understand and express their own musical tastes that are estimated as bases of making default recommendations. The conventional reasoning like "You like A, so you would like B because other users who like like A also like B" is a *relative* expression of musical tastes. We aim to obtain a *direct* expression of each user's musical tastes that is easy to use as a basis for interactive recommendation.

A promising way to do this is to use probabilistic latent semantic indexing (pLSI) based on a multi-topic model, which has been originally used for document modeling [1]. The model includes latent variables corresponding to the concepts of topics. How likely a document and a word cooccur is predicted by stochastically associating each document and word with a topic. Documents and words that are strongly associated with the same topic are likely to occur jointly. The model can be applied to collaborative filtering based on rating histories by treating documents and words as users and items [2]. Given a user, we can predict how likely each item is purchased by estimating how likely the user chooses each topic. The musical tastes of users can be expressed as the strength of user-topic associations.

As shown in Figure 1, we propose continuous pLSI for hybrid recommendation that enhances serendipity by combining rating data with content-based data extracted from musical audio signals. Specifically, Gaussian mixture models (GMMs) are built into the collaborative filtering model of pLSI in order to address continuous vectorial data. Unlike the major collaborative methods relying on heuristics [3,4], the pLSI model can be extended in a consistent manner because it is flexible and has a theoretical basis.

The continuous pLSI, however, suffers from a serious local optima problem because a number of parameters linearly increases according to data size. This causes the *hub* phenomenon [5], in which specific items are almost always recommended to users regardless of their rating histories. Thus, the serendipity of recommendations is insufficient. Although a similar probabilistic model was proposed for genre classification [6], this problem was not addressed.

To solve this problem, we propose three smoothing techniques: multinomial smoothing, Gaussian parameter tying, and artist-based item clustering. The first technique is expected to avoid overfitting and the other two reduce the model complexity. We compared the effectiveness of these techniques experimentally.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 explains a model of the continuous pLSI. Section 4 describes the three smoothing techniques. Section 5 reports our experiments. Section 6 summarizes the key findings of this paper.

2. RELATED WORK

Music recommendation is an important topic today in the field of music information processing. Conventional studies on recommendation have been intended to deal with textual data (documents and words). In addition, many researchers have proposed various ideas to make the most of content-based data that is automatically extracted from musical audio signals. For example, Logan [7] proposed a content-based recommender based on the cosine distance between a user's favorite items and non-rated items. Magno and Sable [8] reported subjective experiments showing that a content-based recommender competes against Last.fm (a collaborative recommender) and Pandora (a recommender based on manual annotations) in terms of user satisfaction. These reports indicate the synergistic effect of integrating rating data with content-based data. Hybrid recommenders have been actively investigated recently. Celma *et al.* [9] used both content-based similarity and user profiles given in RSS feeds to choose suitable items. Tiemann *et al.* [10] integrated two weak learners (social and content-based recommenders) by using an ensemble learning method.

Another important issue is how to present recommendations to users. Donaldson and Knopke [11] visualized the relationships of recommended items in a two dimensional space. Lamere and Maillet [12] proposed a transparent and steerable interface for a recommender based on crowds of social tags. A common concept of these studies seems to be that users had better actively explore or control recommendations. This would result in enhanced serendipity.

The existence of hubs has recently been recognized as a serious problem. Interestingly, this problem was not reported in the field of text-based recommendation. In music recommendation and retrieval, GMMs are generally used to represent the distributions of acoustic features. Aucouturier *et al.* [5] pointed out that this kind of modeling tends to create hubs that are wrongly evaluated as similar to all other items. Berenzweig [13] concluded that the hub phenomenon is related to the curse of dimensionality. Chordia *et al.* [14] discussed content-based recommendation based on the Earth-Movers distance between GMMs of individual items. They empirically found that a homogenization method can improve performance [15]. Hoffman *et al.* [16] tried to solve this problem by using the hierarchical Dirichlet process (HDP) for modeling content-based data. Unlike the GMM, the HDP represents each item as a mixture of an unfixed number of Gaussians. The number is automatically adjusted to match the data complexity. In addition, the same set of Gaussians is used to model all items, with only the mixture weights varying from item to item. This is similar to Gaussian parameter tying.

3. CONTINUOUS PLSI

This section explains a continuous pLSI model and a training method suitable for efficient parallel processing.

3.1 Problem Statement

We define several symbols from a probabilistic viewpoint. Let $U = \{u_1, u_2, \cdots, u_{|U|}\}$ be the set of all users, where $|U|$ is the number of them, and let $V = \{v_1, v_2, \cdots, v_{|V|}\}$ be the set of all items, where $|V|$ is the number of them. Let u and v be *discrete* random variables respectively taking the values of one member of U and one member of V. Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{|V|}\}$ denote content-based data that is a set of D-dimensional feature vectors extracted from individual items. Let $\boldsymbol{x}$ be a *continuous* random variable in the D-dimensional space. Probabilistic distributions are represented as $p(\text{variable})$ or $p(\text{variable1}|\text{variable2})$, e.g., a discrete distribution $p(u)$ or a conditional continuous distribution $p(\boldsymbol{x}|u)$. For example, probabilities or probability densities are given by $p(u = u_i)$ or $p(\boldsymbol{x} = \boldsymbol{x}_j|u = u_i)$, which are simply described as $p(u_i)$ or $p(\boldsymbol{x}_j|u_i)$.

As to available rating data, we mainly assume implicit ratings such as purchase histories or listening counts, which are recorded automatically even when users do not explicitly express their preferences for individual items. In general, the number of implicit ratings tends to be much larger than that of explicit ratings. We thus think that the former are more suitable to probabilistic approaches because for them the sparseness problem is less serious.

The total available data (combinations of rating data and content-based data) is given by $\boldsymbol{O} = \{(u_{(1)}, v_{(1)}, \boldsymbol{x}_{(1)}), \cdots, (u_{(N)}, v_{(N)}, \boldsymbol{x}_{(N)})\}$, where $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)})$ $(1 \le n \le N)$ is a user-item-feature co-occurrence that user $u_{(n)}$ has purchased (viewed or listened to) item $v_{(n)}$ with feature $\boldsymbol{x}_{(n)}$ and N is the number of co-occurrences. Let $c(u, v)$ be the number of times that co-occurrence $(u, v, \boldsymbol{x})$ was observed. Obviously, $N = \sum_{u,v} c(u, v)$. An easy way to utilize explicit ratings (e.g., numerical rating scores such as the numbers of "stars" in an one-to-five scale rating system adopted by Amazon.com) is to set the value of $c(u, v)$ to one if a user u likes item v, i.e., the rating score is greater than a neutral score (three stars). Alternatively, we could use rating scores for weighting $c(u, v)$.

The final objective is to estimate the probabilistic distribution $p(v|u)$, which indicates how likely it is that user u likes item v. Recommendations are then made by ranking items not rated by user u in a descending order of $p(v|u)$.

3.2 Model Formulation

The graphical representation of a continuous pLSI model is shown in Figure 2. This is an extended version of three-way aspect models [17, 18] in which all variables are discrete. We assume that users, items, and features are conditionally independent through latent topics. In other words, once a latent topic is specified, there is no mutual information between three kinds of variables. Although this seems a strong assumption, it is a reasonable way to avoid the local optima problem. Introducing a dependency edge from items to features in order to model the real world accurately would increase the number of parameters drastically.

The pLSI model can explain the process generating co-occurrence $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)})$. Let $Z = \{z_1, \cdots, z_{|Z|}\}$ be a set of *topics*, where $|Z|$ is the number of them. Let z be a latent variable that takes the value of one of Z. Each topic can be regarded as a *soft* cluster that is simultaneously associated with users and items. That is, each user and each item stochastically belong to one of the topics. The model thus treats triplet $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)})$ as incomplete data that is latently associated with $z_{(n)} \in Z$. The complete data is given by quartet $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)}, z_{(n)})$. An interpretation of the generative process is that user $u_{(n)}$ stochastically selects topic $z_{(n)}$ according to his or her taste $p(z_{(n)}|u_{(n)})$, and $z_{(n)}$ stochastically generates item v and its features $\boldsymbol{x}$ in turn. For convenience, we let $\boldsymbol{S}$ be $\{z_{(1)}, \cdots, z_{(n)}\}$.

A unique feature of the continuous pLSI is that $p(\boldsymbol{x}|z)$ is modeled with a Gaussian mixture model (GMM) in order to deal with continuous observation $\boldsymbol{x}$. Let M be the number of mixtures (Gaussian components). Each topic $z_k \in Z$ has a GMM defined by the mixing proportions of Gaussians $\{w_{k,1}, \cdots, w_{k,M}\}$ and their means and covariances $\{\boldsymbol{\mu}_{k,1}, \cdots, \boldsymbol{\mu}_{k,M}\}$ and $\{\boldsymbol{\Sigma}_{k,1}, \cdots, \boldsymbol{\Sigma}_{k,M}\}$. As in the original pLSI, $p(u)$, $p(z|u)$, and $p(v|z)$ are multinomial distributions. We practically use an equivalent definition of the model obtained by focusing on $p(z)$, $p(u|z)$, and $p(v|z)$. The parameters of these multinomial distributions are simply given by (conditional) probability tables of target variables. Let $\boldsymbol{\theta}$ be the set of all parameters of $|Z|$ GMMs and $|Z|(1 + |U| + |V|)$ multinomial distributions.

3.3 Model Training

The training method we explain here uses the Expectation-Maximization (EM) algorithm [19] and is a natural extension of previous methods [17, 18] (c.f., discrete HMM v.s. continuous HMM). Instead of maximizing the incomplete log-likelihood, $\log p(\boldsymbol{O})$, the EM algorithm maximizes the expected complete log-likelihood $E_{\boldsymbol{S}}[\log p(\boldsymbol{S}, \boldsymbol{O})]$ iteratively, where $E_z[f(z)]$ means an expected value of function $f(z)$ with respect to $p(z)$; $E_z[f(z)] = \sum_z p(z)f(z)$.

The complete likelihood of $(u, v, \boldsymbol{x}, z)$ is given by

$$p(u, v, \boldsymbol{x}, z) = p(z)p(u|z)p(v|z)p(\boldsymbol{x}|z). \tag{1}$$

This can be easily calculated for given observations when parameters $\boldsymbol{\theta}$ are obtained.

In the E-step we define a Q function as

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{\text{current}}) = E_{\boldsymbol{S}}[\log p(\boldsymbol{S}, \boldsymbol{O})] \tag{2}$$

$$= \sum_{u,v} c(u, v) \sum_z p(z|u, v, \boldsymbol{x}) \log p(u, v, \boldsymbol{x}, z), \tag{3}$$

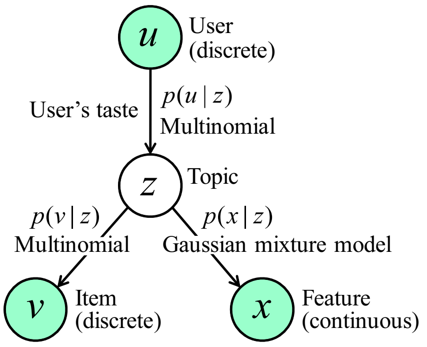


Figure 2. Graphical representation of continuous pLSI.

where $p(z|u, v, \boldsymbol{x})$ is a posterior distribution of latent variable z and can be calculated by using the current parameters $\boldsymbol{\theta}_{\text{current}}$ as follows:

$$p(z|u, v, \boldsymbol{x}) = \frac{p(u, v, \boldsymbol{x}, z)}{\sum_z p(u, v, \boldsymbol{x}, z)}. \tag{4}$$

In the M-step we update the current parameters by maximizing Eqn. (3). Note that $\log p(u, v, \boldsymbol{x}, z)$ can be decomposed into $\log p(z) + \log p(u|z) + \log p(v|z) + \log p(\boldsymbol{x}|z)$. This means that the parameters of each distribution can be updated independently. To update $p(z)$, for example, we only focus on a term related to $p(z)$ as follows:

$$Q_{p(z)} = \sum_{u,v} c(u, v) \sum_z p(z|u, v, \boldsymbol{x}) \log p(z). \tag{5}$$

Using a Lagrange multiplier λ for a constraint of probability standardization, we define a new function $F_{p(z)}$ as

$$F_{p(z)} = Q_{p(z)} + \lambda \left(1 - \sum_z p(z)\right). \tag{6}$$

We then calculate the differential of Eqn. (6) with respect to $p(z)$ and set it to zero as follows:

$$\frac{\partial F_{p(z)}}{\partial p(z)} = \frac{1}{p(z)} \sum_{u,v} c(u, v)p(z|u, v, \boldsymbol{x}) - \lambda \equiv 0. \tag{7}$$

The updated distribution $p(z)$ can be obtained by

$$p(z) = \frac{\sum_{u,v} c(u, v)p(z|u, v, \boldsymbol{x})}{\sum_{u,v,z} c(u, v)p(z|u, v, \boldsymbol{x})}. \tag{8}$$

The other two multinomial distributions $p(u|z)$ and $p(v|z)$ can be similarly updated as follows:

$$p(u|z) = \frac{\sum_v c(u, v)p(z|u, v, \boldsymbol{x})}{\sum_{u,v} c(u, v)p(z|u, v, \boldsymbol{x})}, \tag{9}$$

$$p(v|z) = \frac{\sum_u c(u, v)p(z|u, v, \boldsymbol{x})}{\sum_{u,v} c(u, v)p(z|u, v, \boldsymbol{x})}. \tag{10}$$

To update continuous distribution $p(\boldsymbol{x}|z)$, we focus on

$$Q_{p(\boldsymbol{x}|z)} = \sum_{u,v} c(u, v) \sum_z p(z|u, v, \boldsymbol{x}) \log p(\boldsymbol{x}|z) \tag{11}$$

$$= \sum_{u,v} c(u, v) \sum_{k=1}^{K} p(z_k|\cdot) \log \sum_{m=1}^{M} p(y_{k,m})p(\boldsymbol{x}|z_k, y_{k,m}), \tag{12}$$

where to improve legibility we wrote $p(z|u, v, \boldsymbol{x})$ as $p(z|\cdot)$.

$y_k \in \{y_{k,1}, \cdots, y_{k,M}\}$ is a latent variable that indicates which Gaussian in the GMM of topic z_k generates $\boldsymbol{x}$. $p(y_k)$ represents a probability distribution over M Gaussians, i.e., $p(y_{k,m}) = w_{k,m}$, and $p(\boldsymbol{x}|z_k, y_{k,m})$ is the likelihood that feature $\boldsymbol{x}$ is generated from a Gaussian indicated by $y_{k,m}$. Because the logarithmic operation for the summation makes Eqn. (12) hard to maximize directly, we focus on the expected value of $Q_{p(\boldsymbol{x}|z)}$ with respect to y_k:

$$E_{y_k}[Q_{p(\boldsymbol{x}|z)}] = \sum_{u,v} c(u,v) \sum_{k=1}^{K} p(z_k|\cdot)$$
$$\sum_{m=1}^{M} p(y_{k,m}|\boldsymbol{x}, z_k)\Big(\log w_{k,m} + \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{k,m}, \boldsymbol{\Sigma}_{k,m})\Big), \quad (13)$$

where $p(y_{k,m}|\boldsymbol{x}, z_k)$ is a posterior probability given by

$$p(y_{k,m}|\boldsymbol{x}, z_k) = \frac{p(y_{k,m})p(\boldsymbol{x}|z_k, y_{k,m})}{\sum_{m=1}^{M} p(y_{k,m})p(\boldsymbol{x}|z_k, y_{k,m})}. \quad (14)$$

To obtain optimized $w_{k,m}$, we define the following function by introducing a Lagrange multiplier β:

$$F_{w_k} = E_{y_k}[Q_{p(\boldsymbol{x}|z)}] + \beta\left(1 - \sum_{m=1}^{M} w_{k,m}\right). \quad (15)$$

Calculating the partial partial differential of Eqn. (15) with respect to $w_{k,m}$ and setting it to zero, we obtain

$$w_{k,m} = \frac{\sum_{u,v} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)}{\sum_{m=1}^{M}\sum_{u,v} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)}. \quad (16)$$

Setting the partial differential of Eqn. (13) to zero, the mean and variance $\boldsymbol{\mu}_{k,m}$ and $\boldsymbol{\Sigma}_{k,m}$ are obtained by

$$\boldsymbol{\mu}_{k,m} = \frac{\sum_{u,v} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)\boldsymbol{x}}{\sum_{u,v} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)}, \quad (17)$$

$$\boldsymbol{\Sigma}_{k,m} = \frac{\sum_{u,v} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)(\boldsymbol{x} - \boldsymbol{\mu}_{k,m})^2}{\sum_{u,v} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)}. \quad (18)$$

Given a user u_i, recommendations are made by evaluating $p(v|u_i) = \sum_z p(v|z)p(z|u_i)$, where $p(z|u_i)$ is proportional to $p(z)p(u_i|z)$ and indicates the musical tastes of user u: how likely it is user u_i selects (likes) topic z.

3.4 MapReducing EM Algorithm

Computational efficiency, a very important issue in music recommendation when the database and model become large, is especially critical when used data cannot be loaded on the memory of a single machine. Elegant implementations, however, have scarcely been addressed.

A remarkable advantage of pLSI-based recommenders is that we can easily implement them in parallel processing environments that consist of multiple machines such as clusters. Google News, for example, uses a distributed computation framework called MapReduce [20].

We can implement the continuous pLSI by using MPI or Hadoop [21]. Suppose we have $G_U G_V$ machines (CPUs). Let $\{U_1, \cdots, U_{G_U}\}$ and $\{V_1, \cdots, V_{G_V}\}$ be exclusive sets of users and items, where $U_1 \cap \cdots \cap U_{G_U} = U$ and $V_1 \cap \cdots \cap V_{G_V} = V$. To update $p(z)$, for example, we calculate

$$c_z(U_i, V_j) = \sum_{u \in U_i, v \in V_j} c(u,v)p(z|u,v,\boldsymbol{x}). \quad (19)$$

This can be separately calculated in each machine. To calculate $p(z|u,v,\boldsymbol{x})$, we need only $p(z)$, $p(u|z)$ $(u \in U_i)$, $p(v|z)$ $(v \in V_j)$, and $p(\boldsymbol{x}|z)$. The number of parameters of these distributions is much smaller than the total number of parameters. Finally, we can get an integrated result by

$$p(z) \propto \sum_{1 \le i \le G_U, 1 \le j \le G_V} c_z(U_i, V_j). \quad (20)$$

4. SMOOTHING TECHNIQUES

To avoid overfitting, one needs to use appropriate smoothing techniques. In our study, we use three techniques to improve accuracy and reduce hubness: multinomial smoothing, Gaussian parameter tying, and artist-based item clustering. The first relaxes the excessive inclination of multinomial parameters, and the others limit model complexity.

4.1 Multinomial Smoothing

We add a conjugate prior called a Dirichlet distribution to a Q function as a regularization term. To estimate $p(z)$, for example, we consider the following function:

$$Q'_{p(z)} = Q_{p(z)} + \mathrm{Dir}(\boldsymbol{\alpha}), \quad (21)$$

where $\boldsymbol{\alpha}$ is a set of K parameters of a Dirichlet distribution. This results in the additive smoothing method. We set all parameters to 1.0001. Maximizing $Q'_{p(z)}$, we get

$$p(z) = \frac{\sum_{u,v} c(u,v)p(z|u,v,\boldsymbol{x}) + \alpha - 1}{\sum_z \Big(\sum_{u,v} c(u,v)p(z|u,v,\boldsymbol{x}) + \alpha - 1\Big)}. \quad (22)$$

The updating formulas of the other multinomial distributions are similarly given by

$$p(u|z) = \frac{\sum_v c(u,v)p(z|u,v,\boldsymbol{x}) + \alpha - 1}{\sum_u \left(\sum_v c(u,v)p(z|u,v,\boldsymbol{x}) + \alpha - 1\right)}, \quad (23)$$

$$p(v|z) = \frac{\sum_u c(u,v)p(z|u,v,\boldsymbol{x}) + \alpha - 1}{\sum_v \left(\sum_u c(u,v)p(z|u,v,\boldsymbol{x}) + \alpha - 1\right)}. \quad (24)$$

4.2 Gaussian Parameter Tying

We force all GMMs to share the same set of Gaussians and differ from each other in the mixing proportions of those Gaussians. In the context of HMMs, this is called a tied mixture model. The new updating formulas are given by

$$\boldsymbol{\mu}_{k,m} = \frac{\sum_{u,v,m} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)\boldsymbol{x}}{\sum_{u,v,m} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)}, \quad (25)$$

$$\boldsymbol{\Sigma}_{k,m} = \frac{\sum_{u,v,m} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)(\boldsymbol{x} - \boldsymbol{\mu}_{k,m})^2}{\sum_{u,v,m} c(u,v)p(z_k|\cdot)p(y_{k,m}|\boldsymbol{x}, z_k)}. \quad (26)$$

4.3 Artist-based Item Clustering

We replace *item-based* distribution $p(v|z)$ with *artist-based* distribution $p(a|z)$, where variable a represents one of the artists in the database. Let A be a set of items sung by artist a, That is, let all items be grouped according to their artist names. We train an *artist-based* model for users, artists, and features by iteratively updating $p(a|z)$ as follows:

$$p(a|z) = \frac{\sum_{u,v \in A} c(u,v)p(z|u,v,\boldsymbol{x})}{\sum_{u,v} c(u,v)p(z|u,v,\boldsymbol{x})}. \quad (27)$$

Score	5	4	3	2	1
Counts	**5336**	**1458**	457	211	333
Ratio	68.5%	18.7%	5.86%	2.71%	4.27%

Table 1. Distribution of rating scores.

To recommend items rather than artists, we then construct an *item-based* model by replacing $p(a|z)$ with $p(v|z)$. To do this, we use an incremental training method [18] that re-estimates a distribution of unknown items $p(v|z)$ without affecting other trained distributions $p(z)$, $p(u|z)$,$(\boldsymbol{x}|z)$:

$$p(v|z) = \frac{\sum_u c(u,v) \frac{p(z)p(u|z)p(\boldsymbol{x}|z)}{\sum_z p(z)p(u|z)p(\boldsymbol{x}|z)}}{\sum_{u,v} c(u,v) \frac{p(z)p(u|z)p(\boldsymbol{x}|z)}{\sum_z p(z)p(u|z)p(\boldsymbol{x}|z)}}. \tag{28}$$

5. EVALUATION

We experimentally evaluated the continuous pLSI in terms of accuracy and hubness by using various combinations of the smoothing techniques.

5.1 Data

The music items we used were Japanese songs recorded in single CDs that were ranked in weekly top-20 sales rankings from Apr. 2000 to Dec. 2005. To use these items, we need real implicit ratings $c(u, v)$ such as purchase histories and listening counts, but most online services do not release such data to the public. We therefore instead collected explicit ratings (numbers of “stars” ranging from one to five) from Amazon.co.jp by using official APIs [22] that let us download almost all the information available from Amazon.co.jp [22]. For reliable evaluation, we excluded users who had rated fewer than two items and excluded items that had been rated less than two times. As a result, $|U|$ was 1872 and $|V|$ was 1400. The number of artists was 471. If a rating score given to item v_j by user u_i was greater than three (the neutral score), we set $c(u_i, v_j)$ to the score. Otherwise, we set $c(u_i, v_j)$ to zero. In other words, we considered only positive ratings. A similar approach has been used previously [23]. Note that, as shown in Table 1, the distribution of rating scores was strongly skewed. The density of 6794 positive ratings (scores 4 and 5) was 0.259% in the user-item co-occurrence table.

With regard to the content-based data, we focused on vocal features because all the items included singing voices that strongly affected the musical tastes of users. To extract these features from polyphonic audio signals, we used a method proposed by Fujihara *et al.* [24]. We calculated a 13-dimensional feature vector at each frame where singing voices were highly likely to be included, concatenated the mean and variance of the feature vectors in each item into a 26-dimensional vector, and then used principal component analysis to compress the dimensionality to 20 ($D = 20$).

5.2 Protocols

To test all combinations of the three smoothing techniques, we prepared eight models of the continuous pLSI. For convenience, throughout Section 5, the multinomial smoothing, Gaussian parameter tying, and item clustering are respectively called SM1, SM2, and SM3. The number of latent variables was 256 ($|Z| = 256$). Although the number of mixtures was 32, when SM1 was disabled it was set to 1 in order to avoid overfitting.

	Disabled	SM1	SM2	SM1&2
Disabled	4.65	4.29	6.18	6.57
SM3	7.10	6.72	**19.4**	19.3

Table 2. Expected utility of recommendations: Higher scores indicate better performance.

	Disabled	SM1	SM2	SM1&2
Disabled	5.94	5.81	**6.39**	6.36
SM3	5.98	5.81	6.36	6.34

Table 3. Entropy of recommendations: Higher scores indicate better performance (fewer hubs).

We conducted 10-fold cross validation by splitting the positive explicit ratings into ten groups. Nine groups were used for making recommendations with the eight models. The other group was considered to be not observed and was used for evaluating the recommendations.

5.3 Measures

Recommendation results given as ranked lists of items were evaluated in terms of accuracy and hubness.

To calculate accuracy, we used the expected utility of a ranked list [25], which for each user is defined as

$$R_u = \sum_{r=1}^{|V|-\#(\text{rated items})} \frac{\max(\text{score}_{u,r} - 3, 0)}{2^{(r-1)/(\gamma-1)}}, \tag{29}$$

where $\text{score}_{u,r}$ is the rating score that user u *actually* gave the r-th ranked item although the item was considered a non-rated item (the score was hidden) in model training. When $\text{score}_{u,r}$ was not available, its value was set to 3. γ is a viewing half-life based on the assumption that the probability that a user views an r-th ranked item is twice the probability that the user views an $(r + \gamma)$-th ranked item. We set γ to 5 as in the literature [25]. R_u was not sensitive to the value of γ. The total score is given by

$$R = 100 \frac{\sum_u R_u}{\sum_u R_u^{\max}} \quad (0 \leq R \leq 100), \tag{30}$$

where $R_u^{\max}$ is the maximum achievable utility if all items with available scores given by user u had been at the top of the ranked list in order of those scores. Basically, higher values indicate better performance, but note that the probability of recommending known items is high.

We propose the following hubness measure based the entropy of recommendations:

$$H = -\sum_{j=1}^{|V|} \frac{t(j)}{|U|} \log \left(\frac{t(j)}{|U|} \right), \tag{31}$$

where $t(j)$ is the number of times that item v_j was recommended with the highest (top 1) ranking. A larger H (higher entropy) indicates a smaller bias in how many times each item is recommended.

5.4 Results

As shown in Table 2, the accuracies of recommendations were greatly improved by using SM3. This can be explained from two aspects: the relationship between items and features and that between items and users. First, the items of each artist tend to be similar to each other in their musical features. Second, most users of Amazon.co.jp tend to like any of the items of the few artists they like. This would be a common tendency of the users of many online music distribution services. Therefore, SM3 reduced the complexity of the model while preserving almost all the information of the rating data.

SM2 improved the accuracy of recommendations made regardless of the combinations in which it was used. Interestingly, recommendations obtained by jointly using SM2 and SM3 were much more accurate than those made when these techniques were used independently. SM1, on the other hand, slightly reduced the accuracy because it is based on additive smoothing. It is known that its approximation errors are larger than those of the other smoothing methods such as the Good-turing method.

Table 3 shows hubness of recommendations. SM2 significantly reduced the hubness while the SM1 and SM3 had no gains. This is consistent with the results reported by Hoffman *et al.* [16], who found that HDP and vector quantization (VQ) did not produce many hubs. VQ can be considered as a *hard* clustering version of the tied GMM, which is a *soft* clustering model.

We conclude that combining SM2 and SM3 is the best approach to improving performance. In our experiments, it yielded accuracy comparable with that of conventional methods of collaborative filtering.

6. CONCLUSION

This paper has presented a continuous-pLSI-based model for hybrid music recommendation. The model uses GMMs to represent distributions of acoustic features extracted from musical audio signals. As in the original pLSI, users and items are assumed to follow multinomial distributions. We developed an algorithm for parameter estimation and implemented it in a parallel processing environment. Experimentally testing the abilities of three smoothing techniques —multinomial smoothing, Gaussian parameter tying, and artist-based item clustering—, we found that using the second and third techniques to adjust model complexity significantly improved the accuracy of recommendations and that the second technique could also reduce hubness.

In the future, we plan to introduce conjugate priors of all distributions (GMMs and multinomial distributions) into the continuous pLSI to enable full Bayesian estimation. Extending latent Dirichlet allocation (LDA) [23] and HDP-LDA [26] are worth considering.

Acknowledgement: This study was partially supported by Grant-in-Aid for Young Scientists (Start-up) 20800084.

7. REFERENCES

[1] T. Hofmann and J. Puzicha. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[2] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, pages 688–693, 1999.

[3] P. Resnick, N. Iacovou, M. Sushak, and P. Bergstrom. GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.

[4] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, pages 21–23, 2005.

[5] J.-J. Aucouturier, F. Pache, and M. Sandler. The way it sounds: Timbre models for analysis and retrieval of polyphonic music signals. *IEEE Transactions of Multimedia*, 7(6):1028–1035, 2005.

[6] P. Ahrendt, J. Larsen, and C. Goutte, Co-occurrence models in music genre classification. In *MLSP*, pages 24–252, 2005.

[7] B. Logan. Music recommendation from song sets. In *ISMIR*, pages 425–428, 2004.

[8] T. Magno and C. Sable. A comparison of signal-based music recommendation to genre labels, collaborative filtering, musicological analysis, human recommendation, and random baseline. In *ISMIR*, pages 161–166, 2008.

[9] O. Celma, M. Ramírez, and P. Herrera. Foafing the music: A music recommendation system based on RSS feeds and user preferences. In *ISMIR*, pages 464–467, 2005.

[10] M. Tiemann, S. Pauws, and F. Vignoli. Ensemble learning for hybrid music recommendation. In *ISMIR*, pages 179–180, 2007.

[11] J. Donaldson and I. Knopke. Music recommendation mapping and interface based on structural network entropy. In *ISMIR*, pages 181–182, 2007.

[12] P. Lamere and F. Maillet. Creating transparent, steerable recommendations. Late breaking in ISMIR, 2008.

[13] A. Berenzweig. *Anchors and Hubs in Audio-based Music Similarity*. PhD thesis, Columbia University, 2007.

[14] P. Chordia, M. Godfrey, and A. Rae. Extending content-based music recommendation: The case of Indian classical music. In *ISMIR*, pages 571–576, 2008.

[15] M. Godfrey and P. Chordia. Hubs and homogeneity: improving content-based music modeling. In *ISMIR*, pages 307–312, 2008.

[16] M. Hoffman, D. Blei, and P. Cook. Content-based musical similarity computation using the hierarchical Dirichlet process. In *ISMIR*, pages 349–354, 2008.

[17] A. Popescul, L. Ungar, D. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *UAI*, pages 437–444, 2001.

[18] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. An efficient hybrid music recommender system using an incrementally-trainable probabilistic generative model. *IEEE Transactions on ASLP*, 16(2):435–447, 2008.

[19] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society*, B-39(1):1–38, 1977.

[20] A. Das, M. Datar, and A. Garg. Google news personalization: Scalable online collaborative filtering. In *WWW*, pages 271–280, 2007.

[21] Hadoop. http://hadoop.apache.org/core.

[22] Amazon web services. http://aws.amazon.com.

[23] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Machine Learning Research*, 3:993–1022, 2003.

[24] H. Fujihara, T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. Singer identification based on accompaniment sound reduction and reliable frame selection. In *ISMIR*, pages 329–336, 2005.

[25] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[26] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet processes. *JASA*, 101(476):1566-1581, 2006.

STEERABLE PLAYLIST GENERATION BY LEARNING SONG SIMILARITY FROM RADIO STATION PLAYLISTS

François Maillet, Douglas Eck, Guillaume Desjardins
DIRO, Université de Montréal, CIRMMT
Montreal, Canada
{mailletf,eckdoug,desjagui}@iro.umontreal.ca

Paul Lamere
The Echo Nest
Somerville, USA
paul.lamere@gmail.com

ABSTRACT

This paper presents an approach to generating steerable playlists. We first demonstrate a method for learning song transition probabilities from audio features extracted from songs played in professional radio station playlists. We then show that by using this learnt similarity function as a prior, we are able to generate steerable playlists by choosing the next song to play not simply based on that prior, but on a tag cloud that the user is able to manipulate to express the high-level characteristics of the music he wishes to listen to.

1. INTRODUCTION

The celestial jukebox is becoming a reality. Not only are personal music collections growing rapidly, but online music streaming services like Spotify[1] or Last.fm[2] are getting closer everyday to making all the music that has ever been recorded instantly available. Furthermore, new playback devices are revolutionizing the way people listen to music. For example, with its Internet connectivity, Apple's iPhone gives listeners access to a virtually unlimited number of tracks as long as they are in range of a cellular tower. In this context, a combination of personalized recommendation technology and automatic playlist generation will very likely form a key component of the end user's listening experience.

This work's focus is on providing a way to generate steerable playlists, that is, to give the user high-level control over the music that is played while automatically choosing the tracks and presenting them in a coherent way. To address this challenge, we use playlists from professional radio stations to learn a new similarity space based on song-level audio features. This yields a similarity function that takes audio files as input and outputs the probability of those audio files being played successively in a playlist. By using radio station playlists, we have the advantage of having a virtually unlimited amount of training data. At the same time, we are able to generalize the application of the model to any song for which we have the audio files. We believe this will be the case in any real-life application we can foresee for the model.

Furthermore, we use the concept of a steerable tag cloud [2] to let the user guide the playlist generation process. Tags [6], a type of meta-data, are descriptive words and phrases applied to any type of item; in our case, music tracks. Tags are words like like *chill*, *violin* or *dream pop*. They have been popularized by Web 2.0 websites like Last.fm, where users can apply them to artists, albums and tracks. The strength of tags, especially when used in a social context, lies in their ability to express abstract concepts. Tags communicate high-level ideas that listeners naturally use when describing music. We tag all tracks in our playlists using an automatic tagging system [7] in order to ensure that they are all adequately tagged. Then, given a seed song, the learnt similarity model is used to preselect the most probable songs to play next, after which the similarity between the user's steerable tag cloud and each of the candidate songs' cloud is used to make the final choice. This allows users to steer the playlist generator to the type of music they want to hear.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of related work in music similarity and playlist generation. Section 3 explains how the radio station playlists data set was collected and assembled. Section 4 presents the creation and evaluation of our new similarity space. Section 5 explains how we propose implementing a steerable playlist generator. Finally, section 6 explores future research avenues.

2. PREVIOUS WORK

An increasing amount of work is being conducted on automatic playlist generation, with considerable focus being placed on the creation of playlists by means of acoustic or meta-data similarity [10–14].

More recently, connectivity graphs derived from music social networks are being used to measure similarity. For example, [5] uses network flow analysis to generate playlists from a friendship graph for MySpace[3] artists. In [4], the authors use Last.fm collaborative filtering data

[1] http://www.spotify.com
[2] http://www.last.fm

[3] http://www.myspace.com

to create a similarity graph considering songs to be similar if they have been listened to by similar users. They then embed the graph into a Euclidean space using LMDS, where similar artists would appear near one another.

Another approach [3] uses a case-based reasoning system. From its pool of real human-compiled playlists, the system selects the most relevant ones in regards to the user's one song query and mixes them together, creating a new playlist.

We are aware of only one other attempt to use radio station playlists as a source of data. In [1] radio station playlists are used to construct a weighted graph where each node represents a song and each arc's weight is the number of times the two songs are observed one after the other. From the graph, the authors are able to infer transition probabilities between songs by creating a Markov random field. Our approach is similar, with the advantage that we can generalize to songs not observed in the training data.

3. CONSTRUCTING THE DATA SET

Our model is trained on professional radio station playlists. For this experiment, we consider a playlist to be a sequence of 2 or more consecutive plays uninterrupted by a commercial break. Suppose a radio station plays the tracks t_a, t_b and t_c one after the other, we will consider $\{t_a, t_b\}$ and $\{t_b, t_c\}$ as two 2-song sequences $\in \mathbb{S}^2$, and $\{t_a, t_b, t_c\}$ as one 3-song sequence $\in \mathbb{S}^3$. We consider the sequences $\{t_a, t_b\}$ and $\{t_b, t_a\}$ as two distinct sequences. The model's output will thus be non-symmetric in regards to the order in which the songs are presented.

The playlist data we used came from two sources which we will cover in section 3.1.

3.1 Playlist sources

3.1.1 Radio Paradise

Radio Paradise [4] (RP) is a free Internet-streamed radio station that defines its format as "eclectic online radio." RP provided us with playlist data including every play from January 1st 2007 to July 28th 2008 (575 days). The data consists of 195,692 plays, 6,328 unique songs and 1,972 unique artists.

3.1.2 Yes.com

Yes.com is a music community web site that provides, among other things, the playlists for thousands of radio stations in the United States. Developers are able to access the playlist data via a free web based API [5] that returns the data in JSON format. One API call allows the developer to get a list of radio stations, either by searching by genre, by name or even by proximity to a given ZIP code. Then, for each retrieved station, the API provides access to that station's play history for the last 7 days. The self-assigned and non-exclusive genres of the available radio stations cover all major musical styles. The stations we used to build our own dataset were not chosen for any particular reason. Rather, we made a few searches with the API by genre until we obtained enough data for our work, that is 449 stations. The proportion of stations' non-exclusive association with the different genres is detailed in Table 1.

[4] http://www.radioparadise.com

[5] http://api.yes.com

Table 1. Proportion of the 449 Yes radio stations associated with each genre. Because the genres are non-exclusive the sum of the percentages is > 100.

Latin	11.2%	Christian	11.4%
Country	20.6%	Hip-Hop	17.2%
Jazz	4.3%	Metal	14.1%
Pop	23.3%	Punk	1.6%
Rock	39.4%	R&B/Soul	13.6%

We used data mined from the Yes API from November 13th 2008 to January 9th 2009 (57 days), totaling 449 stations, 6,706,830 plays, 42,027 unique songs and 9,990 unique artists.

Unlike RP, Yes did not provide any indication of where the commercial breaks were located in the list of plays. We inferred where they were by looking at the interval between the start time of every pair of consecutive songs. As we will explain in section 4.1, we only used sequences made of tracks for which we had the audio files. This allowed us to calculate song length and to infer when commercials were inserted. Specifically, if the second of the two songs did not start within ± 20 seconds of the end of the first one, we assumed that a commercial break had been inserted and thus treated the two songs as non-sequential. This approach is more precise than the method used in [1], where breaks were inserted if the elapsed time between two songs was greater than 5 minutes.

3.2 Putting the data together

Combining all the data yielded 6,902,522 plays, with an average of 15,338 plays per station. As we will explain in 4.1, the features we used as input to our model required us to have access to each of the songs' audio file. Of the 47,044 total songs played in the playlists we used, we were able to obtain the audio files for 7,127 tracks. This reduced the number of distinct usable song sequences to 180,232 and 84,668 for the 2 and 3-song sequence case respectively. The sequences for which we had all the audio files were combinations from 5,562 tracks.

Finally, we did not possess a set of explicit negative examples (i.e. two-song sequences that a radio station would never play). In order to perform classification we needed examples from both the positive and negative class. To address this, we considered any song sequence that was never observed in the playlist as being a negative example. During training, at each new epoch, we randomly sampled a new set of negative examples matched in size to our positive example set. With this strategy it is possible that we generated false-negative training examples (i.e. two-song sequences that we didn't see as positive examples in our data set but that in fact a radio station would play). However, since we resample new negative examples after every training epoch, we do not repeatedly show the model the same false-negative pair, thus minimizing potential impact on model performance.

4. SONG SIMILARITY MODEL

4.1 Features

We use audio-based features as input to our model. First, we compute 176 frame-level autocorrelation coefficients for lags spanning from 250ms to 2000ms at 10ms intervals. These are aggregated by simply taking their mean. We then down sample the values by a factor of two, yielding 88 values. We then take the first 12 Mel-frequency cepstral coefficients (MFCC), calculated over short windows of audio (100ms with 25ms overlaps), and model them with a single Gaussian (G1) with full covariance [16]. We unwrap the values into a vector, which yields 78 values.

We then compute two song-level features, danceability [9] and long-term loudness level (LLML) [8]. Danceability is a variation of detrended fluctuation analysis, which indicates if a strong and steady beat is present in the track, while the LLML gives an indication of the perceived loudness of the track. Both of these features yield a single numeric value per song.

These 4 audio features are concatenated to form an 180 dimensional vector for each track.

4.2 Learning models

We formulate our learning task as training a binary classifier to determine, given the features for a sequence of tracks, if they form a song sequence that has been observed in our radio station playlists. If a sequence has been observed at least once, it is considered a positive example. As mentioned above, the negative examples are randomly sampled from the pool of all unseen sequences.

We use three types of learning models in our experiments: logistic regression classifiers, multi-layer perceptrons (MLP) and stacked denoising auto-encoders (SdA). Logistic regression, the simplest model, predicts the probability of a song-sequence occurrence as a function of the distance to a linear classification boundary. The second model, a multi-layer perceptron, can also be interpreted probabilistically. It adds an extra "hidden" layer of nonlinearity, allowing the classifier to learn a compact, nonlinear set of basis functions.

We also use a type of deep neural network called a stacked denoising auto-encoder (SdA) [17]. The SdA learns a hierarchical representation of the input data by successively initializing each of its layers according to an unsupervised criterion to form more complex and abstract features. The goal of this per-layer unsupervised learning is to extract an intermediate representation which preserves information content whilst being invariant to certain transformations in the input. SdAs are exactly like neural networks with the exception that they have multiple hidden layers that are initialized with unsupervised training.

In our experiments, the models operated directly on pairs (or 3-tuples in the case of predicting sequences of length 3) of audio features. The input x of our model is thus a vector of length $180 \cdot n$, with $n \in \{2, 3\}$, formed by concatenating the features of each track into a single vector.

We used 75% of our unique pairs/triplets for training, keeping 12.5% for validating the hyper-parameters and 12.5% for testing. We did not perform any cross-validation.

4.3 Similarity evaluation

Measuring the quality of the similarity space induced by the model is not easy and highly subjective. We will first look at its performance on the learning task (4.3.1), and then try to evaluate it in a more qualitative way (4.3.2).

4.3.1 Learning performance

Classification errors for the different models we trained are presented in Table 2. The errors represent the proportion of real sequences that were classified as false sequences by each model, or vice versa, on the test set, for the best combination of hyper-parameters.

While the logistic regression clearly lacks learning capacity to adequately model the data, the MLPs and SdAs have similar performance. SdAs have been shown to outperform MLPs in complex image classification tasks ([17]) but were unable to learn a significantly better representation of the features we are using for this task. This could mean that the feature set was not sufficiently rich or that the task was simply too difficult for the hierarchical model to find any kind of compositional solution to the problem.

Table 2. Classification errors on the test set for the different models we trained as well as a random baseline. SdA-n represents an SdA with n hidden layers.

Model	2-song seq.	3-song seq.
random	50.00%	50.00%
logistic regression	31.73%	21.08%
MLP	8.53%	5.79%
SdA-2	8.38%	5.58%
SdA-3	8.62%	5.78%

4.3.2 Retrieval evaluation

By using the original radio station playlists as the ground truth, we can evaluate the retrieval performance of our model. The evaluation is done using *TopBucket* (TB) [7], which is the proportion of common elements in the two top-N lists.

Constructing the ground truth from the playlists is done as follows. Each 2-song sequence $S_n^2 \in \mathbb{S}^2$ is made up of tracks $\{t_n^1, t_n^2\}$ and has been observed $|S_n^2|$ times. We construct one top list $\mathcal{L}_{t_i} \forall t_i \in \mathbb{T}$, as the set of all sequences S_n^2 for which $t_n^1 = t_i$, ordered by $|S_n^2|$. $\mathcal{L}_{t_i}$ essentially gives a list of all the tracks that have followed t_i ordered by their occurrence count. In the 3-song sequence case, we construct a top list $\mathcal{L}_{\{t_i,t_j\}}$ for pairs of tracks since in

Table 3. Retrieval performance based on the *TopBucket* (TB) measure of our models compared to random, popularity-biased random, acoustic similarity and autotags similarity. Each score represents the average percentage (and standard deviation) of songs in the ground truth that were returned by each model.

	2-song sequences		**3-song sequences**	
Model	TB10	TB20	TB5	TB10
random	0.25%±0.16%	0.58%±1.75%	0.11%±1.45%	0.25%±1.52%
popularity-biased random	1.01%±3.15%	2.19%±3.34%	0.51%±3.17%	0.96%±3.15%
acoustic (G1C)	1.37%±3.80%	2.37%±3.73%	0.63%±3.48%	1.61%±4.01%
autotags (Cosine distance)	1.43%±3.98%	2.34%±3.86%	0.58%±3.34%	2.26%±4.89%
logistic regression	2.47%±5.08%	6.41%±6.40%	0.20%±2.00%	1.16%±3.40%
MLP	16.61%±14.40%	23.48%±13.17%	7.72%±13.92%	20.26%±17.85%
SdA-2	13.11%±12.05%	19.13%±11.19%	7.25%±13.66%	21.74%±19.75%
SdA-3	13.17%±11.31%	18.22%±10.04%	9.74%±18.00%	26.39%±22.74%

practice, a playlist generation algorithm would know the last two songs that have played.

For our experiments, we used the top 10 and 20 elements and 5 and 10 elements for the 2 and 3-song sequence case respectively. The results, shown in Table 3, represent the average number of common elements in the ground truth's top list and each of the similarity models' for every song.

Because most sequences were only observed once ($|S_n^2| = 1$), we were often in the situation where all the sequences in $\mathcal{L}_{t_i}$ had an occurrence of 1 ($\forall S_n^2 \in \mathcal{L}_{t_i} : |S_n^2| = 1$) and the number of sequences in $\mathcal{L}_{t_i}$ was greater than N for a top-N list. Because in such a situation there was no way to determine which sequences should go in the top-list, we decided to extend the top-N list to all the sequences that had the same occurrence count as the N^{th} sequence. In the 2-song sequence case, we also ignored all sequences that had $|S_n^2| = 1$ to keep the top-N lists from growing a lot larger than N. Ignoring as well all the songs that did not have at least N tracks in their top-list, we were left, in the 2-song sequence case, with 834 songs that had an average of 14.7 songs in their top-10 list and 541 songs with an average of 28.8 songs in their top-20 list. In the 3-song sequence case, the top-5 list was made up of 1,181 songs with a 6.6 top songs average and the top-10 list was composed of 155 songs with an average of 13.8 top songs.

We compared our model's retrieval performance to two other similarity models. First, we computed the similarity in the space of autotags [7] from the cosine distance over song's tags vector [2]. The second comparison was performed by retrieving the most acoustically similar songs. Acoustic similarity was determined by using G1C [15] which is a weighted combination of spectral similarity and information about spectral patterns. We also compared our model to a popularity-biased random model that probabilistically chooses the top songs based on their popularity. Each song's popularity was determined by looking at the number of sequences it is part of.

In the 3-song sequence case, for the autotag and acoustic similarity, we represent the similarity $sim(\{t_1, t_2\}, t_3)$ as the mean of $sim(t_1, t_3)$ and $sim(t_2, t_3)$.

The results of Table 3 clearly show that there is more involved than simple audio similarity when it comes to reconstructing sequences from radio station playlists. The performance of the audio and autotag similarity are indeed significantly lower than models that were trained on actual playlists.

Furthermore, the TB scores of Table 3 are from the models that have the best classification error (see Table 2). It is interesting to note that some models with a worst classification error have better TB scores. While classification is done by thresholding a model's certainty at 50%, TB gives an indication of the songs for which a model has the highest certainty. Since these are the songs that will be used when generating a playlist, this metric seems more appropriate to judge the models. The relation between classification error and TB scores is a topic for further investigation.

5. STEERABLE PLAYLIST GENERATION

While the model presented above is able to build a similarity space in which nearby songs fit well together in a playlist, it does not provide a mechanism for allowing the user to personalize the sequence for a given context. To address this, final song selection was done using the *Aura* [6] [2] recommendation engine from Sun Microsystems Labs. Aura is able to generate transparent and steerable recommendations by working with a textual representation — a tag cloud — of the items it is recommending. Specifically it finds the most similar items to any other in its pool by computing the cosine distance on their respective tag clouds. It can also explain to the user why an item is recommended by showing the overlap between tag clouds.

We use Aura as a means to allow users to personalize ("steer") the playlist generation by allowing them to create a personal tag cloud that represents the music they wish to listen to. In order to generate tag clouds for our tracks, we used *Autotagger* [7], a content-based machine learning model. This model is designed to generate a tag cloud (specifically a weighted vector of 360 music-relevant words) from an audio track, thus allowing us to use Aura's cosine distance measure to compute the similarity between

[6] `http://www.tastekeeper.com`

each track and the user's personal cloud.

5.1 Steps for generating a steerable playlist

Our playlist generation algorithm works as follows :

1. A seed track $t_s \in \mathbb{T}$ is selected amongst all possible tracks.

2. The similarity model is used to compute transitional probabilities between the seed song and all other ones (with more similar songs having higher transition probabilities), keeping only the top φ, or thresholding at a certain transition probability ρ. Let $\mathcal{T}$ be the group of these top songs:

$$\mathcal{T} = \underset{t_i \in \mathbb{T} \setminus t_s}{\arg\max}^{\varphi} \mathcal{M}(t_s, t_i) \tag{1}$$

3. The user is then invited to create a tag cloud $\mathcal{C}_U$ by assigning weights to any of the 360 tags in the system. In this way the cloud is personalized to represent the mood or type of songs the user would like to hear. The higher the weight of a particular tag, the more impact it will have on the selection of the next song.

4. Autotagger is used to generate a tag cloud $\mathcal{C}_{t_j}$ for all tracks $t_j \in \mathcal{T}$. The cosine distance ($cd(\cdot)$) between these tag clouds and $\mathcal{C}_U$ is used to find the song that best matches the abstract musical context the user described with his or her cloud:

$$t_{min} = \underset{t_j \in \mathcal{T}}{\arg\min}\, cd(\mathcal{C}_U, \mathcal{C}_{t_j}) \tag{2}$$

5. The track t_{min} is selected to play next. Since the system is transparent, we can tell the user we chose the song t_{min} because it has a certain transition probability from the seed song but also because its tag cloud overlapped with $\mathcal{C}_U$ in a particular way. The user can then go back and modify the tag cloud $\mathcal{C}_U$ to influence how subsequent songs will be selected.

Naturally, a lot of extra factors can be used when determining which song to play in step 4. For instance, we could consider the user's taste profile to take into account what types of songs he normally likes, mixing his current steerable cloud to the one representing his musical tastes. We could also include a discovery heuristic to balance the number of novel songs selected as opposed to ones the user already knows.

5.2 Example playlists

To illustrate the effect of the steerable tag cloud, we generate two playlists seeded with the same song but with very different steerable clouds. The first 9 iterations of both playlists are shown in Table 4. The effect of the cloud is clearly visible by the different direction each playlist takes. In our view, this transition is done smoothly because it is constrained by the underlying similarity model.

To visualize the similarity space and the playlist generating algorithm, we compute a full track-to-track similarity matrix and reduce its dimensionally using the t-SNEE [18] algorithm (see Figure 1). We chose t-SNEE because it tends to retain local distances while sacrificing global distances, yielding an appropriate two-dimensional visualization for this task (i.e. the distance between very similar songs is more important to us than the relative global placement of, e.g., jazz with respect to classical). We have overlaid the trajectory of the two playlists in Table 4 to illustrate their divergence.

Table 4. Both the following playlists are seeded with the song *Clumsy* by *Our Lady Peace*. To give a clear point of reference, we use the tag clouds of actual songs as the steerable cloud. The *soft* tag cloud is made up of the tags for *Imagine* by *John Lennon* and the *hard* tag cloud with the tags for *Hypnotize* by *System of a Down*.

Soft tag cloud
Viva la Vida by Coldplay
Wish You Were Here by Pink Floyd
Peaceful, Easy Feeling by Eagles
With or Without You by U2
One by U2
Fields Of Gold by Sting
Every Breath You Take by The Police
Gold Dust Woman by Fleetwood Mac
Enjoy The Silence by Depeche Mode
Hard tag cloud
All I Want by Staind
Re-Education (Through Labor) by Rise Against
Hammerhead by The Offspring
The Kill by 30 Seconds To Mars
When You Were Young by The Killers
Hypnotize by System of a Down
Breath by Breaking Benjamin
My Hero by Foo Fighters
Turn The Page by Metallica

6. CONCLUSIONS

We have demonstrated a method for learning song similarity based on radio station playlists. The learnt model induces a new space in which similar songs fit well when played successively in a playlist. Several classifiers were evaluated on a retrieval task, with SdAs and MLPs performing better than other similarity models in reconstructing song sequences from professional playlists. Though we were unable to show that SdAs outperform MLPs, we did show much better performance than logistic regression and measures such as G1C over standard audio features. Furthermore we argue that our model learns a direct similarity measure in the space of short song sequences rather than audio or meta-data based similarity. Finally, we showed a way of doing steerable playlist generation by using our similarity model in conjunction with a tag-based distance measure.

Though this model is only a first step, its power and simplicity lie in the fact that its two components play very different but complementary roles. First, the similarity model does the *grunt work* by getting rid of all unlikely candidates, as it was trained specifically for that task. This then greatly facilitates the steerable and fine-tuned selection of the subsequent track based on the textual aura, as most of

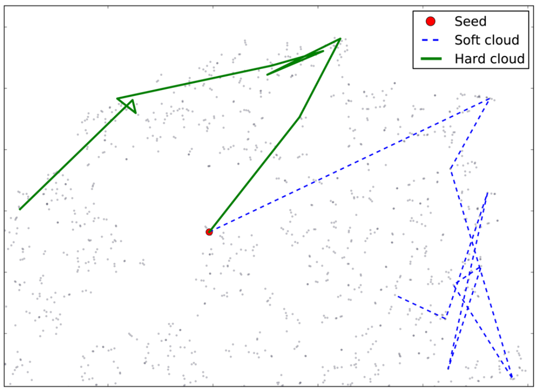


Figure 1. Part of the 2-d representation of the track-to-track similarity matrix generated by a 2-song sequence SdA model. The trajectories of the two playlists described in Table 4 are overlaid over the tracks. Both playlists are seeded with the same song, which is represented by the bigger dot. Each playlist diverges because of the steerable tag cloud that is guiding its generation.

the obvious bad picks have already been removed.

Future work should attempt to use the number of occurrences of each sequence to give more importance to more reliable sequences. Also, the model might learn a better similarity space by being trained with richer features as input. For example, adding meta-data such as tags, a measure of popularity, the year the song was recorded, etc., might prove helpful. Such a richer input space is likely necessary to show a performance gain for SdAs over competing probabilistic classifiers. Our experiments also led us to believe that an increase in the quality of the learnt similarity could probably be attained by simply adding more training data, something that can be easily accomplished as thousands of songs are played on the radio everyday.

7. REFERENCES

[1] R. Ragno, CJC Burges, C. Herley: "Inferring similarity between music objects with application to playlist generation," *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pp. 73–80, New York, USA, 2005

[2] S. Green, P. Lamere, J. Alexander, F. Maillet: 'Generating Transparent, Steerable Recommendations from Textual Descriptions of Items," *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, USA, 2009

[3] C. Baccigalupo, E. Plaza: "Case-based sequential ordering of songs for playlist recommendation," *Lecture Notes in Computer Science*, Vol. 4106, pp. 286–300, 2006.

[4] O. Goussevskaia, M. Kuhn, M. Lorenzi, R. Wattenhofer: "From Web to Map: Exploring the World of Music," *IEEE/WIC/ACM International Conference on Web Intelligence*, Sydney, Australia, 2008.

[5] B. Fields, C. Rhodes, M. Casey, K. Jacobson: "Social Playlists and Bottleneck Measurements: Exploiting Musician Social Graphs Using Content-Based Dissimilarity and Pairwise Maximum Flow Values," *Proceedings of the 9th International Conference on Music Information Retrieval*, Philadelphia, USA, 2008.

[6] P. Lamere: "Social Tagging And Music Information Retrieval," *Journal of New Music Research*, Vol. 37, No. 2, 2008.

[7] T. Bertin-Mahieux, D. Eck, F. Maillet, P. Lamere: "Autotagger: a model for predicting social tags from acoustic features on large music databases," *Journal of New Music Research*, Vol. 37, No. 2, pp. 115–135, 2008.

[8] E. Vickers: "Automatic long-term loudness and dynamics matching," *Proceedings of the AES 111th Convention*, New York, USA, 2001.

[9] S. Streich: "Music Complexity: a multi-faceted description of audio content," Ph.D. Dissertation, UPF, Barcelona, 2007.

[10] J.J. Aucouturier, F. Pachet: "Scaling up Music Playlist Generation," *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, Vol. 1, pp. 105–108, Lausanne, Switzerland, 2002.

[11] B. Logan: "Content-based playlist generation: Exploratory experiments," *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002

[12] B. Logan: "Music Recommendation from Song Sets," *Proceedings of the 5th International Conference on Music Information Retrieval*, Barcelona, Spain, 2004.

[13] E. Pampalk, T. Pohle, G. Widmer: "Dynamic Playlist Generation Based on Skipping Behaviour," *Proceedings of the 6th International Conference on Music Information Retrieval*, London, UK, 2005.

[14] A. Flexer, D. Schnitzer, M. Gasser, G. Widmer: "Playlist Generation Using Start and End Songs," *Proceedings of the 9th International Conference on Music Information Retrieval*, Philadelphia, USA, 2008.

[15] E. Pampalk: "Computational Models of Music Similarity and their Application in Music Information Retrieval, Ph.D. dissertation, Vienna University of Technology, 2006.

[16] M. Mandel, D. Ellis: "Song-Level Features and Support Vector Machines for Music Classication," *Proceedings of the 6th International Conference on Music Information Retrieval*, London, UK, 2005.

[17] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol: "Extracting and Composing Robust Features with Denoising Autoencoders," *International Conference on Machine Learning*, 2008

[18] L.J.P. van der Maaten, G.E. Hinton: "Visualizing High-Dimensional Data Using t-SNE," *Journal of Machine Learning Research*, 9(Nov):2579-2605, 2008.

EVALUATING AND ANALYSING DYNAMIC PLAYLIST GENERATION HEURISTICS USING RADIO LOGS AND FUZZY SET THEORY

Klaas Bosteels
Ghent University, Gent, Belgium
klaas.bosteels@ugent.be

Elias Pampalk
Last.fm Ltd., London, UK
elias@last.fm

Etienne E. Kerre
Ghent University, Gent, Belgium
etienne.kerre@ugent.be

ABSTRACT

In this paper, we analyse and evaluate several heuristics for adding songs to a dynamically generated playlist. We explain how radio logs can be used for evaluating such heuristics, and show that formalizing the heuristics using fuzzy set theory simplifies the analysis. More concretely, we verify previous results by means of a large scale evaluation based on 1.26 million listening patterns extracted from radio logs, and explain why some heuristics perform better than others by analysing their formal definitions and conducting additional evaluations.

1. INTRODUCTION

In January 2009, Arbitron and Edison Research measured the popularity of digital music platforms by means of a survey of 1,858 American people aged 12+ .[1] They estimated that 42 million Americans tune to online radio on a weekly basis, which is more than twice their number from 2005, and claim that the number of 12+ year old Americans owning a digital music player increased from 14% in 2005 to 42% in 2009. They also found that the vast majority of these people own an Apple iPod or iPhone. Evidently, the Apple products dominate their market, which is commonly attributed to their innovating design and user interfaces. The recent "Genius" feature is a nice example of such innovation. Using this feature, users can automatically create coherent playlists by selecting a *seed song*, i.e., an example of a song of interest, and pressing a single button. Many of the popular online radio stations are similar in concept. The user supplies one or more seeds, and the system generates a corresponding list of tracks that is turned into a custom radio station. Hence, automatic playlist generation can be seen as a technology that is, to some extent, responsible for the recent growth established by certain digital music platforms, and its commercial importance is likely to increase further in the near future.

This paper is about simple heuristics for automatically generating playlists. More precisely, we will discuss simple rules of thumb for choosing the song to be played next, given a set of candidate songs. This set of candidates can consist of all available tracks, but usually it is restricted to a limited subset. In order to avoid repetition, for instance, the set of candidates has to be restricted to the songs that have not been played yet. A realistic scenario is to select the candidates using some other method, effectively turning the heuristic into an enhancement rather than a playlist generation method on its own.

A very simple way to improve upon random selection, is to repeatedly choose the candidate that is most similar to a given seed song [1]. This *playlist generation heuristic* is said to be static because the song sequence is completely determined from the seed, without taking any additional user input into account. Dynamic heuristics, on the other hand, rely on user feedback to dynamically improve the selection process [2]. For example, the aforementioned static heuristic can be made dynamic by letting it pick the song that is most similar to any of the accepted songs, where the set of accepted songs consists of the seed song as well as all tracks that were not skipped [3]. When there is no given seed, the set of accepted songs can initially be empty and the next track can be chosen at random until there is at least one accepted song. This latter heuristic could easily be added to any system that returns multiple candidate songs for being played next.

Putting it in one sentence, we discuss simple dynamic playlist generation heuristics in this paper. In comparison with alternative techniques, such heuristics are interesting because they (i) are simple and thus easy to compute and implement, and (ii) can easily be added as an enhancement to many existing playlist generation systems.

2. RELATED WORK

Dynamic playlist generation can be seen as a special case of the well-known *relevance feedback* paradigm from information retrieval [4]. In this paradigm, the user is asked to give explicit feedback by labeling results as either relevant or irrelevant, which leads to additional information that can be used by the system to refine the search strategy and generate a better list of results. Several rounds of feedback can be conducted, each bringing the results closer to the user's implicit target concept. Hence, dynamic playlist generation is basically relevance feedback with the returned set of results restricted to one item. In case of this paper, the feedback taken into account is also implicit rather than explicit, but there is no reason to as-

[1] http://www.arbitron.com/study/digital_radio_study.asp

sume that dynamic playlist generation heuristics could not be based on more explicit feedback like “thumbs up/down” buttons instead of skipping behavior.

Over the past few years, relevance feedback has received quite a lot of research attention. In particular, some relevance feedback techniques have already been applied to music information retrieval, including training a decision tree [5], a vector quantizer [6], and an SVM [7]. Since the number of examples is very low when the returned set of results is restricted to one item [2], using these machine learning techniques for dynamic playlist generation might be problematic, however. For the custom-tailored heuristics described in this paper, this is less of a problem. Moreover, their simplicity can be considered an additional advantage from a computational and implementation point of view. Furthermore, as we already mentioned in the introductory section, the described heuristics can also be thought of as a refinement that can be added to a more complex relevance feedback system.

3. FORMALIZATION

The definition of playlist generation heuristics can be formalized using fuzzy set theory [8]. In this section, we explain this formalization in detail, since we rely on it extensively in the subsequent sections.

3.1 Fuzzy Sets

Let U denote a universe, i.e., a (crisp) set of considered objects. A *fuzzy set* F in U is a $U \to [0,1]$ mapping that associates a degree of membership $F(u)$ with each element u from U [9]. The higher $F(u)$, the more u is a member of F. In particular, u fully belongs to F when $F(u) = 1$, and $F(u) = 0$ implies that u is not at all an element of F. We use the notation $\mathcal{F}(U)$ for the class of fuzzy sets in U, which can be regarded a superclass of $\mathcal{P}(U)$, the class of crisp sets in U. A *(binary) fuzzy relation* L in U is a fuzzy set in $U \times U$, i.e., $L \in \mathcal{F}(U \times U)$ [9].

The fuzzy set Sim_X, with X a crisp set in the (finite) universe C of songs to be explored, is the main stepping stone towards the fuzzy formalization. It is given by

$$Sim_X(u) = \max_{x \in X} M(u, x) \tag{1}$$

for all $u \in U$, where U is the subset of C consisting of all candidate songs. In this definition, M is a fuzzy relation in C such that each relationship degree $M(c,d)$, with $(c,d) \in C^2$, corresponds to the degree to which c is similar to d. Putting it in words, Sim_X is a fuzzy set in U such that $Sim_X(u)$ can be interpreted as the degree to which u is similar to any song in X.

In order to obtain a crisp set of tracks from a fuzzy song set, we rely on the following formal operator:

$$X\rceil F = \left\{ x \in X \mid F(x) = \max_{y \in X} F(y) \right\} \tag{2}$$

for all $X \in \mathcal{P}(C)$ and $F \in \mathcal{F}(C)$, i.e., $X\rceil F$ is the crisp set consisting of the elements from X with the greatest membership degree in F. Using this operator, we can formally define the dynamic heuristic discussed in the introductory section of this paper as $U\rceil Sim_A$, with A the set of all accepted songs. In practice, the set $U\rceil Sim_A$ will be a singleton most of the time, but theoretically speaking it can contain up to $|U|$ elements. We can choose one element at random when $|U\rceil Sim_A| > 1$, however, since each song from the set can be considered equally suitable for being played next. In the remainder of this paper, we silently assume that this procedure is followed for all introduced heuristics, i.e., we will define the heuristics as crisp sets and assume that one element is chosen at random when this set has several members.

3.2 Operations on Fuzzy Sets

The set-theoretic operations complement, intersection, and union can be generalized to fuzzy sets as follows:

$$(\mathrm{co}_{\mathcal{N}} F)(u) = \mathcal{N}(F(u)) \tag{3}$$

$$(F \cap_{\mathcal{T}} G)(u) = \mathcal{T}(F(u), G(u)) \tag{4}$$

$$(F \cup_{\mathcal{S}} G)(u) = \mathcal{S}(F(u), G(u)) \tag{5}$$

for each $u \in U$, with $F, G \in \mathcal{F}(U)$, $\mathcal{N}$ a *negator*, $\mathcal{T}$ a *t-norm*, and $\mathcal{S}$ a *t-conorm*. We restrict the sheer number of possibilities by only considering the widely-used standard negator N_{S} given by $N_{\mathrm{S}}(x) = 1 - x$ for all $x \in [0,1]$, the three prototypical t-norms [10] given by

$$T_{\mathrm{M}}(x,y) = \min(x,y) \tag{6}$$

$$T_{\mathrm{P}}(x,y) = x \cdot y \tag{7}$$

$$T_{\mathrm{L}}(x,y) = \max(x + y - 1, 0) \tag{8}$$

for all $x, y \in [0,1]$, and their duals

$$S_{\mathrm{M}}(x,y) = \max(x,y) \tag{9}$$

$$S_{\mathrm{P}}(x,y) = x + y - x \cdot y \tag{10}$$

$$S_{\mathrm{L}}(x,y) = \min(x + y, 1) \tag{11}$$

for all $x, y \in [0,1]$. In the remainder, we will abbreviate $\mathrm{co}_{N_{\mathrm{S}}}$ by co since N_{S} is the only negator we consider.

For this paper, however, we mainly need a generalized set-theoretic difference, which can be obtained by defining

$$(F \setminus_{\mathcal{I}} G)(u) = N_{\mathrm{S}}(\mathcal{I}(F(u), G(u))) \tag{12}$$

for every u from U, with $F, G \in \mathcal{F}(U)$ and $\mathcal{I}$ an *implicator*. We consider two ways of generating implicators in this paper, namely, *S-implicators* and *R-implicators*. The S-implicator induced by a t-conorm $\mathcal{S}$ and the standard negator N_{S} is the $[0,1]^2 \to [0,1]$ mapping $\mathcal{I}_{\mathcal{S}}$ defined as $\mathcal{I}_{\mathcal{S}}(x,y) = \mathcal{S}(N_{\mathrm{S}}(x), y)$, for all $x, y \in [0,1]$, and the R-implicator induced by a t-norm $\mathcal{T}$ is the $[0,1]^2 \to [0,1]$ mapping $\mathcal{I}_{\mathcal{T}}$ given by, for all $x, y \in [0,1]$, $\mathcal{I}_{\mathcal{T}}(x,y) = \sup\{\gamma \in [0,1] \mid \mathcal{T}(x,\gamma) \leq y\}$. For the above-mentioned prototypical t-norms and the corresponding t-conorms, this leads to the following implicators:

$$I_{S_{\mathrm{M}}}(x,y) = \max(1 - x, y) \tag{13}$$

$$I_{S_{\mathrm{P}}}(x,y) = 1 - x + x \cdot y \tag{14}$$

$$I_{S_{\mathrm{L}}}(x,y) = \min(1 - x + y, 1) \tag{15}$$

$$I_{T_{\mathrm{M}}}(x, y) = \begin{cases} 1 & \text{if } x \leq y \\ y & \text{otherwise} \end{cases} \tag{16}$$

$$I_{T_{\mathrm{P}}}(x, y) = \begin{cases} 1 & \text{if } x \leq y \\ \frac{y}{x} & \text{otherwise} \end{cases} \tag{17}$$

$$I_{T_{\mathrm{L}}}(x, y) = I_{S_{\mathrm{L}}, N_{\mathrm{S}}}(x, y) \tag{18}$$

for all $x, y \in [0, 1]$.

3.3 Formal Heuristics

Having the operations on fuzzy sets at our disposal, we can incorporate the set R of all rejected songs by replacing Sim_A in $U \rceil Sim_A$ with a set-theoretic expression in terms of both Sim_A and Sim_R. The heuristic defined as $U \rceil (Sim_A \setminus_{\mathcal{I}} Sim_R)$, for instance, selects the songs that are similar to an accepted song but not similar to any rejected ones, as illustrated by Fig. 4(a). By taking into account the fact that $(U \rceil F) \rceil F = U \rceil F$ for each $F \in \mathcal{F}(U)$, we can easily define slightly more fine-grained heuristics, however. Instead of replacing Sim_A in $U \rceil Sim_A$, we can first rewrite $U \rceil Sim_A$ as $(U \rceil Sim_A) \rceil Sim_A$ and then replace only the first occurrence of Sim_A, which effectively leads to heuristics of the form $(U \rceil P) \rceil Sim_A$, where P is a set-theoretic expression in Sim_A and Sim_R. We call this expression P the *preselection expression*, since it implements a preselection step that precedes further filtering by Sim_A. As values for P, we consider the set-theoretic expressions illustrated by Fig. 4(a), Fig. 4(b), Fig. 4(c), which leads to the following heuristics:

$$H_{\mathrm{a}}^{\mathcal{I}} = (U \rceil (Sim_A \setminus_{\mathcal{I}} Sim_R)) \rceil Sim_A \tag{19}$$

$$H_{\mathrm{b}} = U \rceil Sim_A \tag{20}$$

$$H_{\mathrm{c}}^{\mathcal{I}} = (U \rceil \mathrm{co}(Sim_R \setminus_{\mathcal{I}} Sim_A)) \rceil Sim_A \tag{21}$$

For the value of the parameter $\mathcal{I}$ in $H_{\mathrm{a}}^{\mathcal{I}}$ and $H_{\mathrm{c}}^{\mathcal{I}}$, we will consider the five different implicators discussed in the previous subsection, namely, $I_{S_{\mathrm{M}}}$, $I_{S_{\mathrm{P}}}$, $I_{S_{\mathrm{L}}}$, $I_{T_{\mathrm{M}}}$, and $I_{T_{\mathrm{P}}}$.

The contour plots in Fig. 1 show how the implemented preselection strategy varies for the considered implicators. For each preselection expression P, there exists a corresponding $[0, 1]^2 \to [0, 1]$ mapping p such that $P(u) = p(Sim_A(u), Sim_R(u))$ for all $u \in U$. Table 1 lists these mappings for all considered (non-trivial) preselection expressions, and the plots in Fig. 1 each illustrate one of these mappings. Essentially, these plots provide a top view of the three-dimensional plots for the $[0, 1]^2 \to [0, 1]$ mappings. More precisely, the lines connect points for which the illustrated mapping yields the same value, leading to a partitioning of the $[0, 1]^2$ square into different areas. The darker the area, the higher the values returned by the mapping in this area. Hence, songs u for which the point $(Sim_A(u), Sim_R(u))$ is in a dark area are given preference by the preselection strategy in question.

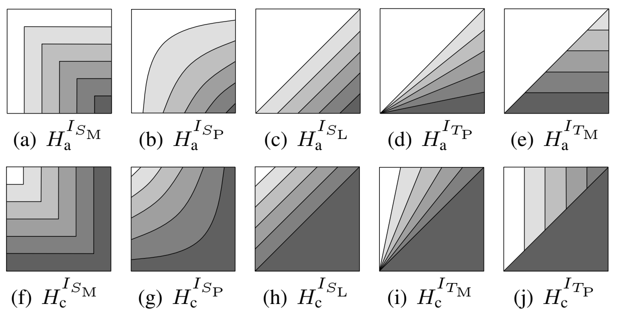
All previously-introduced playlist generation heuristics can be formalized in this way [8]. In particular, the well-performing heuristic defined as

> For each candidate song, let d_a be the distance to the nearest accepted, and let d_s be the distance to the nearest skipped. If $d_a < d_s$, then add the candidate to the set S. From S play the song with smallest d_a. If S is empty, then play the candidate song which has the best (i.e. the lowest) d_a/d_s ratio.

in [2], is equivalent to $H_{\mathrm{c}}^{I_{T_{\mathrm{P}}}}$. In addition to being more concise and precise, the formal definition of this heuristic was also obtained more systematically and is easier to analyse, as we will demonstrate later on in this paper.

preselection expression	$[0, 1]^2 \to [0, 1]$ mapping
$Sim_A \setminus_{I_{S_{\mathrm{M}}}} Sim_R$	$\min(x, 1 - y)$
$Sim_A \setminus_{I_{S_{\mathrm{P}}}} Sim_R$	$x - x \cdot y$
$Sim_A \setminus_{I_{S_{\mathrm{L}}}} Sim_R$	$\max(x - y, 0)$
$Sim_A \setminus_{I_{T_{\mathrm{M}}}} Sim_R$	$\begin{cases} 0 & \text{if } x \leq y \\ 1 - y & \text{otherwise} \end{cases}$
$Sim_A \setminus_{I_{T_{\mathrm{P}}}} Sim_R$	$\begin{cases} 0 & \text{if } x \leq y \\ 1 - \frac{y}{x} & \text{otherwise} \end{cases}$
$\mathrm{co}(Sim_R \setminus_{I_{S_{\mathrm{M}}}} Sim_A)$	$\max(1 - y, x)$
$\mathrm{co}(Sim_R \setminus_{I_{S_{\mathrm{P}}}} Sim_A)$	$1 - y + y \cdot x$
$\mathrm{co}(Sim_R \setminus_{I_{S_{\mathrm{L}}}} Sim_A)$	$\min(1 - y + x, 1)$
$\mathrm{co}(Sim_R \setminus_{I_{T_{\mathrm{M}}}} Sim_A)$	$\begin{cases} 1 & \text{if } y \leq x \\ x & \text{otherwise} \end{cases}$
$\mathrm{co}(Sim_R \setminus_{I_{T_{\mathrm{P}}}} Sim_A)$	$\begin{cases} 1 & \text{if } y \leq x \\ \frac{x}{y} & \text{otherwise} \end{cases}$

Table 1. Corresponding $[0, 1]^2 \to [0, 1]$ mappings for the considered (non-trivial) preselection expressions.


(a) $H_{\mathrm{a}}^{I_{S_{\mathrm{M}}}}$ (b) $H_{\mathrm{a}}^{I_{S_{\mathrm{P}}}}$ (c) $H_{\mathrm{a}}^{I_{S_{\mathrm{L}}}}$ (d) $H_{\mathrm{a}}^{I_{T_{\mathrm{P}}}}$ (e) $H_{\mathrm{a}}^{I_{T_{\mathrm{M}}}}$

(f) $H_{\mathrm{c}}^{I_{S_{\mathrm{M}}}}$ (g) $H_{\mathrm{c}}^{I_{S_{\mathrm{P}}}}$ (h) $H_{\mathrm{c}}^{I_{S_{\mathrm{L}}}}$ (i) $H_{\mathrm{c}}^{I_{T_{\mathrm{M}}}}$ (j) $H_{\mathrm{c}}^{I_{T_{\mathrm{P}}}}$

Figure 1. Contour plots illustrating the preselection strategies of the considered instances of $H_{\mathrm{a}}^{\mathcal{I}}$ and $H_{\mathrm{c}}^{\mathcal{I}}$. Every candidate song u corresponds to a point $(Sim_A(u), Sim_R(u))$ in each of these plots, and the points in the darker areas are given preference by the strategy in question.

4. BASIC EVALUATION

The evaluations described in [2] and [8] are all based on the fairly simplistic assumption that a song is a good addition to a playlist when it is from the same genre as the seed. For this paper, however, we evaluated the considered heuristics using patterns extracted from Last.fm radio logs. More precisely, we looked for sequences of 22 tracks for which the last two tracks did not both get accepted or rejected, i.e., one of them got accepted while the other got rejected. Tracks were considered accepted when the user listened to more than 50% of them. In order to make sure that the extracted patterns represent genuine user interactions, we imposed two additional restrictions: (i) at least 5 and at most 15 of the first 20 tracks got accepted, and (ii) the last song of the sequence was not the last song of a listening session. In this way, we avoid problems like, e.g., the user falling asleep or getting distracted while listening to the radio station, or the last song being considered a skip whereas the user really just turned off the radio while this song was playing.

All of the patterns used for our evaluation were extracted from log files produced by Last.fm "playlist" radio stations, which basically shuffle randomly through user-generated lists of tracks. Last.fm provides its users the ability to create and share playlists, and subscribers can listen to these playlists in random shuffle mode when they contain at least 45 playable tracks by 15 different artists. We considered 1,260,271 patterns extracted from log files generated by such stations, involving 53,768 unique listeners and 516,261 different tracks from 70,306 artists.

The similarity values used for our evaluation were derived from tag data using the well-known cosine similarity measure [4], i.e., songs to which Last.fm users applied the same tags were considered similar to each other. Since the values from $[0, 1]$ obtained in this way can directly be interpreted as membership degrees, we did not have to apply any normalization procedures in order to obtain the fuzzy relation M on which the definition of Sim_X is based.

For each considered pattern, we made every heuristic choose between the last two tracks based on the acceptance history for the 20 previous tracks, and counted how many times they picked the wrong one. More formally, each pattern corresponds to a (A, R, r, w) tuple, where A and R are the sets of accepted and rejected songs, respectively, and r and w are the right and the wrong choice. The *failure rate* for a given heuristic is then obtained by putting $U = \{r, w\}$ for each pattern and counting how many times w is returned by the heuristic.

Fig. 2 shows the results of our basic evaluation. The circles mark the failure rates, and the lines through them represent the 95% binomial confidence intervals computed by approximating the binomial distribution with a normal distribution. These results roughly confirm the findings obtained in [8]. Again, $H_c^{I_{S_L}}$ and $H_c^{I_{T_P}}$ perform significantly better than the other heuristics, although the difference between $H_c^{I_{T_P}}$ and $H_c^{I_{S_P}}$ is just barely significant in this case. It still remains unclear why exactly these two heuristics perform best, however, which is precisely the motivation for the subsequent sections of this paper.

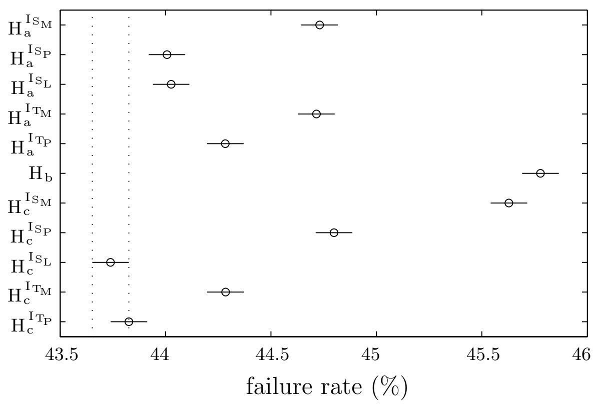


Figure 2. Results of the basic evaluation. The circles mark the failure rates, and the lines represent the 95% binomial confidence intervals.

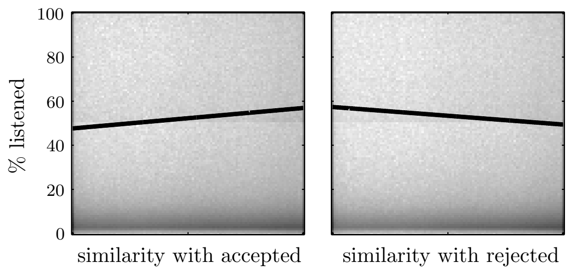
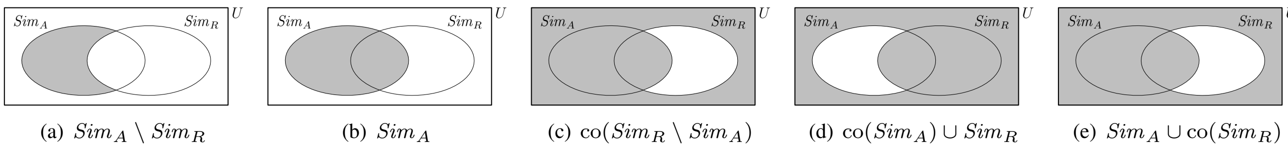


Figure 3. Two-dimensional histograms for the (similarity with accepted tracks, % listened) and (similarity with rejected tracks, % listened) pairs corresponding to the considered patterns. Darker regions contain more pairs, and the thick black lines were obtained using linear regression.

5. INCONSISTENT USER PREFERENCES

With each pattern considered for our basic evaluation, we can associate two pairs of the form (similarity with accepted tracks, % listened), one for the right choice and another for the wrong one. Similarly, we can also associate two pairs of the form (similarity with rejected tracks, % listened) with each pattern. Fig. 3 shows the distribution of these pairs for the considered patterns. The two thick black lines in this figure are the linear regression lines, i.e., the best-fitting straight lines through all of the points in terms of least squares. As illustrated by these regression lines, users apparently tend to avoid songs that are similar to the skipped tracks in favor of the ones similar to the tracks that were not skipped, which is the main assumption behind the dynamic heuristics discussed in this paper. However, the regression lines are only slightly tilted, suggesting that the user preferences are often driven by reasons unrelated to the (computed) similarity with the accepted or rejected tracks. We say such preferences are *inconsistent*, and distinguish the resulting *inconsistent skipping behavior* into two categories: (i) an *inconsistent accept* occurs when an accepted song is either similar to a rejected track, or not similar to any of the accepted ones, and (ii) an *inconsistent reject* occurs when a rejected song is similar to an accepted track or not similar to any rejected tracks. In the context of a radio station, for instance, an eclectic user might not mind when a song is not similar to any of the already accepted songs, leading to an inconsistent accept. On the other hand, the user might reject a particular track because she happens to dislike the corresponding artist for certain (unmeasurable) reasons, even though the track is very similar to the already accepted songs, resulting in an inconsistent reject which might in turn lead to inconsistent accepts, since the user is likely to accept songs that are similar to



(a) $Sim_A \setminus Sim_R$ (b) Sim_A (c) $\mathrm{co}(Sim_R \setminus Sim_A)$ (d) $\mathrm{co}(Sim_A) \cup Sim_R$ (e) $Sim_A \cup \mathrm{co}(Sim_R)$

Figure 4. The dark areas in these Venn diagrams depict the main set-theoretic expressions considered in this paper.

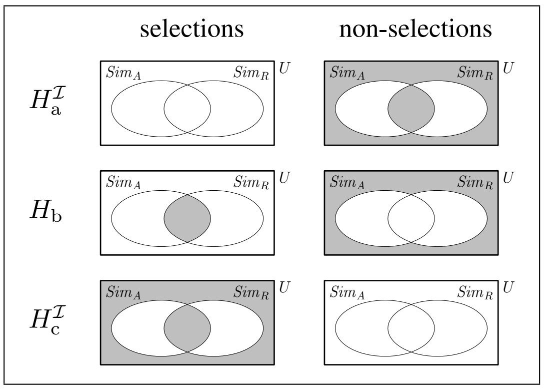

selections
non-selections
$H_{\mathrm{a}}^{\mathcal{I}}$
H_{b}
$H_{\mathrm{c}}^{\mathcal{I}}$
Sim_A
Sim_R
U

Figure 5. The inconsistent selections and non-selections area for all considered heuristics.

the inconsistently rejected track.

Now, by thinking of the fuzzy sets as if they were crisp sets, we can intuitively determine how well the preselection expressions from the heuristics comply with inconsistent user preferences. A song u from U selected by a crisp preselection expression P can lead to an inconsistent accept when either $u \notin Sim_A$ or $u \in Sim_R$. Hence, the area corresponding to potential inconsistent accepts for a preselection expression P is the intersection of P with the set-theoretic expression shown by Fig. 4(d). We call this area the *inconsistent selections area*. Similarly, we can define the *inconsistent non-selections area* as the intersection of $\mathrm{co}(P)$ and the expression shown by Fig. 4(e). The larger the inconsistent selections area, the better the preselection expression complies with inconsistent accepts, and the larger the inconsistent non-selections area, the better it complies with inconsistent rejects.

Fig. 5 shows the inconsistent selections and non-selections area for all considered heuristics. Judging from this figure, $H_{\mathrm{a}}^{\mathcal{I}}$ should perform best when inconsistent rejects occur more frequently than inconsistent accepts, $H_{\mathrm{c}}^{\mathcal{I}}$ is expected to perform best when inconsistent accepts are more common, and $H_{\mathrm{b}}^{\mathcal{I}}$ should perform similarly under both circumstances. In order to verify these theoretical insights, we conducted some additional evaluations.

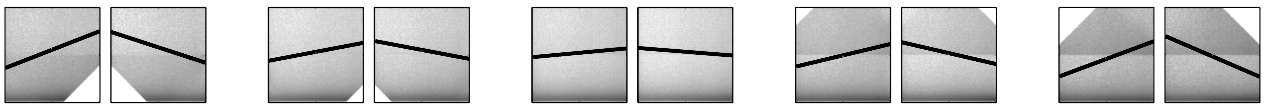
6. ADDITIONAL EVALUATIONS

By disregarding some of the extracted patterns, we can control the level of inconsistent accepts and rejects. As illustrated by Fig. 6(a), for example, the relative number of inconsistent accepts can be increased by ignoring all patterns for which either $s_{\mathrm{A}} - l_{\mathrm{A}} > 0.6$ or $1 - s_{\mathrm{R}} - l_{\mathrm{R}} > 0.6$ holds, with $(s_{\mathrm{A}}, l_{\mathrm{A}})$ and $(s_{\mathrm{R}}, l_{\mathrm{R}})$ a (similarity with accepted tracks, % listened) and a (similarity with rejected tracks, % listened) pair, respectively, corresponding to the pattern in question. Disregarding patterns in this way actually removes inconsistent rejects, but this effectively leads to a higher percentage of inconsistent accepts in the obtained dataset. Increasing the level of inconsistent rejects can be done analogously. By considering several cut-off values, we generated 9 different datasets that gradually move from a high level of inconsistent accepts to a high level of inconsistent rejects, as illustrated by Fig. 6.

We conducted the basic evaluation for every generated dataset, which led to the plots shown in Fig. 7. In accordance with Fig. 5, the dash-dotted line is below the dashed one in the left part of each of these plots, whereas it is always above the dashed one in the right part. The solid line, on the other hand, is roughly symmetrical along the dotted vertical divider, which also complies nicely with Fig. 5. Although their magnitudes vary a lot depending on the used value for the implicator $\mathcal{I}$, the differences in performance are clearly visible in each subfigure, confirming the insights we obtained by analysing the formal definitions of the heuristics.

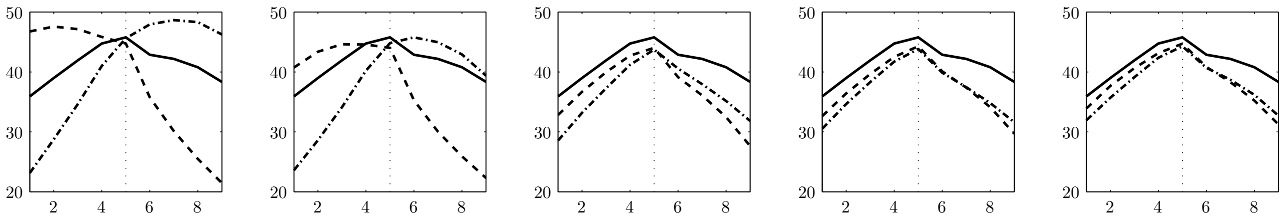
Now that we linked the performance of the heuristics to inconsistent user preferences, we can finally explain why the failure rate for the best performing instance of $H_{\mathrm{c}}^{\mathcal{I}}$ is significantly smaller than those for all instances of $H_{\mathrm{a}}^{\mathcal{I}}$ in Fig. 2. The reason for this is simply that the full collection of extracted listening patterns contains more inconsistent accepts than inconsistent rejects, which can easily be demonstrated by reducing the granularity of the two-dimensional histograms from Fig. 3 and summing up the counts for certain bins. For instance, we can get a rough idea of the number of inconsistent accepts by considering merely four bins and summing up the counts for the bins highlighted in Fig. 8(a). Similarly, we can roughly determine the number of inconsistent rejects by summing up the counts for the bins highlighted in Fig. 8(b). The following numbers were obtained in this way: 1,222,094 inconsistent accepts and 1,186,155 inconsistent rejects. Moreover, the dataset illustrated by Fig. 6(a) consists of 554,614 patterns, while the one corresponding to Fig. 6(e) is made up of only 440,171 patterns. Hence, the original dataset indeed seems to contain more inconsistent accepts than inconsistent rejects. The difference is not that large, however, which explains why there is only a very small gap between the performance of $H_{\mathrm{c}}^{I_{S_{\mathrm{L}}}}$ and $H_{\mathrm{a}}^{I_{S_{\mathrm{P}}}}$ in Fig. 2.

Note that Fig. 7 also illustrates that $I_{S_{\mathrm{L}}}$ can be seen as a balanced compromise between the extremes $I_{S_{\mathrm{M}}}$ and $I_{T_{\mathrm{M}}}$. For the other implicators, the measured performance tends to vary a lot for different heuristics, but $I_{S_{\mathrm{L}}}$ rarely leads to significantly worse performance than any of the other considered implicators. In Fig. 2 as well in as all empirical



(a) dataset 1 (b) dataset 3 (c) dataset 5 (d) dataset 7 (e) dataset 9

Figure 6. Two-dimensional histograms that illustrate how the 9 generated datasets gradually move from a high level of inconsistent accepts to a high level of inconsistent rejects.



(a) I_{S_M} (b) I_{S_P} (c) $I_{S_\mathrm{L}} = I_{T_\mathrm{L}}$ (d) I_{T_P} (e) I_{T_M}

Figure 7. Results of the additional evaluations for $H_\mathrm{a}^\mathcal{I}$ (- -), H_b (–), and $H_\mathrm{c}^\mathcal{I}$ (-·-). The numbers along the horizontal axis are dataset identifiers, while the vertical axis shows failure rate percentages.

results described in [8], $H_\mathrm{a}^{I_{S_\mathrm{L}}}$ and $H_\mathrm{c}^{I_{S_\mathrm{L}}}$ perform at least as well as all other instances of $H_\mathrm{a}^\mathcal{I}$ and $H_\mathrm{c}^\mathcal{I}$, respectively.

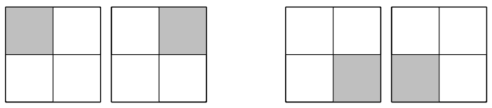
7. CONCLUSION AND FUTURE WORK

The mathematical apparatus from the theory of fuzzy sets proves to be very convenient for defining dynamic playlist generation heuristics. Using the described fuzzy framework, we obtained definitions that are not only systematic and both concise and precise, but also intuitively clear and easy to analyse. We relied on this latter benefit to relate the performance of the considered heuristics to inconsistent user preferences. More precisely, we established that $H_\mathrm{a}^\mathcal{I}$ performs best when inconsistent rejects occur more frequently than inconsistent accepts, that $H_\mathrm{c}^\mathcal{I}$ can be expected to perform best when inconsistent accepts are more common, and that $H_\mathrm{b}^\mathcal{I}$ performs similarly under both circumstances. We clearly confirmed these theoretical insights by means of a new methodology for evaluating playlist generation heuristics based on listening patterns extracted from radio logs, which allowed us to conduct accurate experiments using massive amounts of data.

Since we mainly focussed on comparing the heuristics with each other in this paper, it still remains largely unclear to what extent they can improve the performance of a particular playlist generation system. Future work should try to measure the performance impact of the considered heuristics on specific playlist generations systems, and compare them with potential alternatives. In order to obtain a fairer comparison, the underlying fuzzy relation M could then be based on a more advanced similarity measure than simple tag-based cosine similarity.

8. ACKNOWLEDGEMENTS

The first author would like to thank the Fund for Scientific Research–Flanders (FWO) for funding this research.



(a) inconsistent accepts (b) inconsistent rejects

Figure 8. Categorization of certain bins from the coarse-grained two-dimensional histograms.

9. REFERENCES

[1] B. Logan. Content-based playlist generation: Exploratory experiments. In *Proc. ISMIR Intl. Conf. on Music Info. Retrieval*, 2002.

[2] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. *Proc. ISMIR Intl. Conf. on Music Info. Retrieval*, 2005.

[3] B. Logan. Music recommendation from song sets. In *Proc. ISMIR Intl. Conf. on Music Info. Retrieval*, 2004.

[4] H. Blanken, A. de Vries, H. Blok, and L. Feng, editors. *Multimedia retrieval*. Springer, 2007.

[5] S. Pauws and B. Eggen. PATS: Realization and user evaluation of an automatic playlist generator. In *Proc. ISMIR Intl. Conf. on Music Info.Retrieval*, 2002.

[6] K. Hoashi, K. Matsumoto, and N. Inoue. Personalization of user profiles for content-based music retrieval based on relevance feedback. In *Proc. ACM Intl. Conf. on Multimedia*, 2003.

[7] M. Mandel, G. Poliner, and D. Ellis. Support vector machine active learning for music retrieval. *Multimedia Systems*, 12:3–13, 2006.

[8] K. Bosteels and E. Kerre. A fuzzy framework for defining dynamic playlist generation heuristics. *Fuzzy Sets and Systems*. To appear.

[9] L. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[10] E. Klement, R. Mesiar, and E. Pap. *Triangular norms*. Kluwer Academic Publishers, 2000.

SMARTER THAN GENIUS?
HUMAN EVALUATION OF MUSIC RECOMMENDER SYSTEMS.

Luke Barrington **Reid Oda**[*] **Gert Lanckriet**
Electrical & Computer Engineering, [*]Cognitive Science
University of California, San Diego
lukeinusa@gmail.com roda@ucsd.edu gert@ece.ucsd.edu

ABSTRACT

Genius is a popular commercial music recommender system that is based on collaborative filtering of huge amounts of user data. To understand the aspects of music similarity that collaborative filtering can capture, we compare Genius to two canonical music recommender systems: one based purely on artist similarity, the other purely on similarity of acoustic content. We evaluate this comparison with a user study of 185 subjects. Overall, Genius produces the best recommendations. We demonstrate that collaborative filtering can actually capture similarities between the acoustic content of songs. However, when evaluators can see the names of the recommended songs and artists, we find that artist similarity can account for the performance of Genius. A system that combines these musical cues could generate music recommendations that are as good as Genius, even when collaborative filtering data is unavailable.

1. INTRODUCTION

The popularity of the online radio station Pandora.com (20 million users) and Apple iTunes' "Genius" feature (released in September 2008 and available to over 10 million registered iTunes users) has brought the perennial MIR research topic of music similarity and recommender systems into the public spotlight. Apple, the largest music retailer in the world, collects massive amounts of data about music purchase and listening habits of its users. Our experiments demonstrate that collaborative filtering of this data allows Genius to produce better music recommendations than systems based on simple metadata- or content-based analysis. However, Genius fails on music for which collaborative filtering data is unavailable, such as the huge volume of undiscovered content in the "long tail" of the music market.

In this paper, we seek to understand the musical cues that Genius' collaborative filtering identifies to capture music similarity. We can then develop MIR recommender systems that use the same cues, without the need for massive amounts of user data. Since we do not have access to the collaborative filtering input to the Genius algorithm, we compare its output to two canonical recommender systems where we have complete knowledge of their available musical information. We discover that, despite not basing its recommendations directly on the audio content, collaborative filtering can capture information about acoustic similarity, as well as metadata similarity, for playlist *generation*. Using a blind user study, we determine the influence of certain metadata (e.g., familiarity, affinity, visibility) and musical factors (e.g., styles, sounds, artists) on playlist *evaluation* .

2. THE BLACK ART OF PLAYLIST GENERATION

A playlist is a collection of songs grouped together under a particular principle. The principle could be general, such as "rock songs from the 70's" or personal like "songs that remind me of Melanie". Cunningham et al.[1] make the distinction between playlists and "mixes". While a mix can have abstract themes and the sequence of songs is important, a playlist simply embodies a mood or desired emotional state or acts as a background to an activity (work, romance, sports, etc.). The order of songs in a playlist is not important and it is often played on shuffle. Cunningham et al.'s user study reports that 50 percent of requests for help in creating a playlist included a song as an example. Our work focuses on this "query by example" paradigm where the user provides a song as a query or "seed" and the recommender system's task is to generate a playlist of more music that somehow "fits well" with the seed song. The meaning of "fits well" may depend on a variety of the factors below.

2.1 Factors that impact playlist generation

Playlists may be generated (either automatically or by hand) to reflect a *mood*, accompany an *activity* or explore *novel songs* for music discovery. Recommendations can be based on similarity to one or more seed examples or songs may be grouped based on semantic descriptions. The top organization schemes for playlists in [1] were similar *artists*, *genres* and *styles* so we focus on the impact of these factors for automating playlist generation.

2.2 Factors that influence playlist evaluation

It is rare that a playlist is rated explicitly by the conditions used to generate it. The playlist's purpose plays a large role in evaluating it. Since music is often experienced within a social context[2], factors such as song *popularity*, *familiarity* and the perception of the recommender system as an *expert* can play a large role in the perceived quality of the playlist. Even systems that generate novel or serendipitous playlists for song discovery must include some familiar and relevant items to inspire users to trust the recommender system[3]. This may be achieved by offering some *transparency* of the recommendations, e.g., by showing matching artists or using descriptive tags.

3. MUSIC RECOMMENDER SYSTEMS

A variety of approaches to music recommendation and playlist generation have been proposed by the MIR community. Aucouturier and Pachet [4] used acoustic similarity to group songs together. Flexer et al. [5] propose using KL divergence between acoustic song models to make a playlist that transitions coherently from a start to an end song. Xiao et al.[6] describe songs' acoustic content using automatically generated tags drawn from a variety of semantic categories. They derive a music similarity metric by learning the optimum weighting of these categories and find genre similarity to be the most important predictor of subjective evaluations.

Fields et al.[7] extract social-network flow between artists on MySpace and use the resulting artist association metadata to build playlists. Vignoli and Pauws [8] designed a recommender system that allows users to control how acoustic timbre information is combined with genre, mood, year and tempo metadata. The resulting playlists rated higher than less transparent controls in a user evaluation.

3.1 Two Types of Recommender System

Section 2 details a variety of influences that may be used by music recommender systems but they can be broadly categorized into two different approaches:
Content-based systems "listen" to the audio content of the music and build playlists by finding songs that sound similar (e.g., [4, 5]) or that have similar semantic descriptions (e.g., [9, 6]). For example, the popular online radio station Pandora.com [1] employs professional musicologists to listen to each of the 1 million songs in its "music genome" database and objectively characterize their acoustic content using 400 semantic descriptors (e.g., major or minor tonality, the amount of syncopation, the gender of the vocalist, etc.).
Metadata-based systems use information associated with the music that is not directly related to the acoustic content such as artist names (e.g., [7]), genre or other tag information, purchase data, popularity, etc. For example, the Genius playlist algorithm uses collaborative filtering based on the purchase history of millions of iTunes users (i.e., listeners who bought *this* song also bought *that* song).

For this paper, we evaluate the Genius recommender system against one content-based and one metadata-based approach to generating playlists, as well a system that generates playlists randomly. All systems take a seed song and return a playlist of five recommended songs. Each algorithm that we consider is described in detail below.

3.2 Genius

The iTunes Genius recommender system [2] uses the Gracenote MusicID service[10] to fingerprint songs in a user's music library and identify the name of the song, artist, album, etc. This metadata is then used to identify the songs in Genius' database. Although the exact details of the algorithm are a trade-secret of Apple Inc., Genius appears to use collaborative filtering to compare the seed song's metadata to iTunes' massive database of music sales (over 50 million customers who have purchased over 5 billion songs), as well as play history and song rating data collected from iTunes users [3] . When it is first initialized, Genius analyzes a user's music library and compiles all of the collaborative filtering data necessary to build playlists from the library, based on any given seed song. While this fingerprinting and database communication takes some time ($\sim$ 1 hour for our 12,000-song library), the only acoustic analysis involved seems to be fingerprinting for the purpose of metadata information and not content-based recommendation.

Informal experiments with Genius give some clues into its operation and verify that it does not use content analysis directly. For example, if we delete the ID3 metadata information associated with a given MP3 file, or add a song to the library which is unknown to Gracenote (e.g., a new recording by an obscure band), Genius fails to recommend any music. Furthermore, if we choose a seed song that is very atypical of the style of the artist or album that features the song, Genius recommends music that represents the more common aspects of the artist. For example, using the seed song "Beautiful World", a country-folk ballad that is an outlying anomaly on the album "Renegades" by the metal band "Rage Against the Machine", Genius recommends a playlist of aggressive, thrash-metal songs by bands such as "Incubus" and "Nirvana". Although these artists are related to the seed artist, the sound and style of the resulting playlist is very dissimilar to that of the seed song. Based on this analysis, we expect Genius to perform well when recommending playlists based on popular seed songs but to suffer when analyzing less well-known music.

3.3 Artist Similarity

To provide a second, more transparent playlist algorithm that, like Genius, is not based on acoustic analysis, we consider building playlists based on artist similarity. The social music-streaming website last.fm offers lots of user-generated information about songs and artists [4] . In particular, for any given artist, our artist similarity system retrieves a ranked list of the 100 most similar artists to the seed song's artist. We use this last.fm metadata to build a playlist by moving down the ranked list and choosing a random song by each artist that we find in our library. Comparisons between these music recommendations and Genius' playlists will illuminate the degree to which collaborative filtering captures **artist similarity**.

3.4 Semantic Similarity of Automatic Tags

To examine Genius' ability to capture **acoustic similarity** between songs, we compare it to a purely content-based approach. This recommender system is modeled on Pandora.com in that it finds similar songs by matching semantic

[1] `www.Pandora.com`
[2] Our experiments use Genius incorporated in iTunes version 8.0.
[3] Based on `http://www.apple.com/pr/` as well as a meeting between the authors and iTunes in January 2009.
[4] `www.last.fm/api`

descriptions of the audio content. Pandora's semantic data and its music library are proprietary, so we recreate a similar system using computer audition.

We use an automatic tagging algorithm, described in detail in [11], to describe any song using 149 different semantic tags. These tags include descriptors of the genre, emotion, instruments, vocals and usages of the song. For a given song, the output of this "auto-tagger" is a set of probabilities that indicate the relevance of each tag to the song. These probabilities may be interpreted as the parameters of a "semantic" multinomial distribution that characterizes the song, just as a human listener might use words to describe a song's acoustic content (e.g., "very jazzy, features a lot of saxophone and piano, and good to listen to on a date").

The auto-tag system computes similarity between two songs by comparing the Kullback-Liebler (KL) divergence between their semantic multinomial distributions. To build a playlist, we return songs with minimum KL-divergence from the seed song. Abstracting multimedia representations using semantics has shown improvements over direct feature-based similarity for retrieval of images[12], video[**?**] and sound effects[9] and this system was among the top four performing algorithms in the 2007 MIREX audio similarity challenge [13].

4. PLAYLIST EVALUATION EXPERIMENT

One of the biggest challenges when designing music recommendation systems lies in evaluating any proposed method. There is no standard "ground truth" data set on which to test, let alone train, music similarity algorithms. Widely available surrogates for similarity exist, such as deciding that songs should be deemed "similar" if they come from similar genres [6, 14], artists [15] or albums [9]. Playlists can be evaluated by examining their intersection with existing, human-generated playlists [16, 6] but this requires that the same music libraries be used to generate both the new and the reference playlists.

A more accurate, but less scalable or flexible approach uses humans to evaluate music recommendations. This was the approach taken in the 2007 MIREX contest[13] and, though great effort was required to collect this information, the resulting evaluation was very rich. This data has not been released to the MIR community. Human computation games such as Tag-A-Tune[17] may provide another source of human-derived music similarity data.

Since the goals of this paper are both to evaluate the performance of different music recommender systems in various simulated scenarios *and* determine the factors that influence these evaluations, we built a new platform for humans to evaluate playlists as well as collect information about the strengths and weaknesses of each system.

4.1 The Interface

A new subject arriving at the experiment website sees brief instructions explaining the task and the playlist evaluation procedure. The subject then logs in, so that they can return to the experiment at a later date and not repeat trials.

A single evaluation or "trial" consists of three stages: 1) Listen to and evaluate a seed song. 2) Listen to and evaluate 2 playlists. 3) Indicate factors that influenced the playlist evaluation. 50 seed songs were chosen in advance and, on each trial, one seed song is randomly assigned to a subject (without repetition). In the first stage, the subject listens to the seed song and rates how *familiar* they are with the song and how much they *like* the song, both on a 5-point scale.

Once the subject rates the seed song, stage 2 displays two playlists, each containing 5 songs, generated by one of the 4 possible recommender systems (Genius, Artist Similarity, Similar Tags or a random playlist). The two systems in a given trial are randomly chosen but not the same. The subject can listen to the songs from each playlist in any order or re-listen to the seed song by pressing corresponding play buttons. Beside each song is a button to indicate any bad song that "doesn't fit" in the playlist. After listening to the playlists, the subject evaluates which playlist is better, on a 5-point scale: "Playlist 1 much better", "Playlist 1 somewhat better", "Equal", "Playlist 2 somewhat better", "Playlist 2 much better".

After choosing the winning playlist, stage 3 asks the subject to indicate factors that influenced their evaluation. Six factors are presented that may have affected the subject's evaluation of why *either* playlist was a good match with the seed song: similar **sounds, genres, artists, energy, instrumentation** and **lyrics**. These factors only examine the relationship between the seed song and the playlists. Other factors (e.g., usage, time) are assumed to be implicit in the choice of the seed song and are not tested in this work. The subject can select as many factors as they deem relevant or indicate that the factor was not relevant (this choice was not pre-selected) before continuing on to the next trial. Subjects can quit at any stage and their progress is saved.

4.2 The Music

The playlists are built from the authors' personal music library of over 12,000 relatively popular songs that span the most common genres of Western popular music, with very little music outside these genres. The genres include rock, alternative, punk, soft rock, classic rock, folk, pop, electronica, experimental, blues, jazz, soul and hip-hop. The 50 seed songs were chosen to represent these genres in proportions roughly equal to those observed in the library [5] . For each seed song, we pre-calculate a five-song playlist using each of the recommender systems described in Section 3.

We used 30-second song clips, beginning 30 seconds from the start of the song. 30 seconds is generally enough to give a good impression of a song (e.g., it is a standard length for previewing songs in online music stores) while being sufficiently short to make each trial manageable, since subjects are required to listen to 11 clips.

In half the trials, song and artist names for both the seed song and the playlist songs are *hidden* from subjects. This allows us to investigate the influence of the song and, in particular, artist names on subjects' evaluation of playlists. In certain music recommendation scenarios, listeners may read the names of the songs they hear (e.g., album track-lists, record stores, music players such as iTunes, WinAmp,

[5] The list of seed songs and music library can be found at `http://cosmal.ucsd.edu/cal/projects/playlist/`

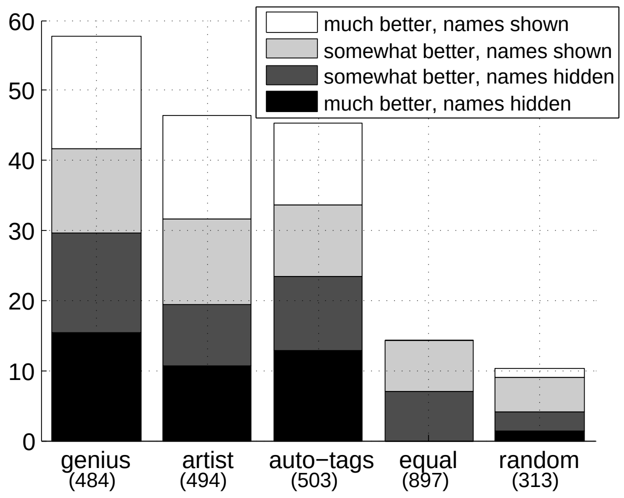


Figure 1. Percent wins for each music recommender system, divided over trials where the song names were hidden or shown. X-axis displays the system (and number of trials where this system was presented). Y-axis displays the percentage of trials where the system was the winner.

and last.fm) while in other cases, they only hear - and do not see - the playlist (e.g., most stereos, iPods on shuffle mode, on the radio, at parties or clubs).

5. RESULTS

Experimental subjects were recruited from psychology and engineering classes at UCSD, via email to friends, colleagues and the Music-IR mailing list and from online blogs and social networks. During the three week experiment, 185 subjects completed 894 trials with, on average, 4.8 trials per subject, including a maximum of 44 trials by one user. Seed songs were chosen randomly (without repeating a seed for any subject) and each of the 50 seed songs was presented an average of 18 times with a minimum of 11 and a maximum of 30 trials. Each of the three playlist generation methods was presented in at least 638 trials and each of the 150 playlists (a {seed song, playlist method} pair) was presented in, on average, 11.8 trials.

Figure 1 displays how often each recommender system won, as a proportion of all the trials in which it appeared. Figure 2 indicates how each system fared against the others, in head-to-head comparisons. The fading between colors in Figures 2-4 indicates the variance over 50 random subsamplings of 75% of the data for each condition. It is clear that Genius outperforms both the Artist Similarity and Similar Tags methods in most cases although a more detailed examination is given below.

5.1 Trial Lengths

Table 1 demonstrates that subjects spent, on average, 226 seconds on each trial, indicating that they listened to almost all of each 30-second song clip (11 songs x 30 seconds = 330 seconds). This time was significantly less for trials where the song and artist names were visible (196 seconds) and significantly longer when the names were hidden (258 seconds), indicating that subjects were often able to evaluate playlists (or, at least, some of the songs in a playlist) simply by looking at the names of the song and artist. Thus, we expect trials where the names were hidden to estimate better the impact of the “sounds” of the songs while those with names shown will demonstrate the impact of artist similarity.

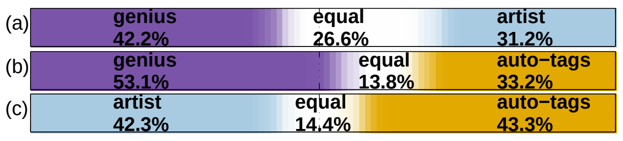


Figure 2. Head-to-head playlist comparisons over all conditions. Ignoring the equal votes, all systems are significantly better than random and Genius is significantly better than the content-based system using similar tags (Chi-square test for fit to a uniform distribution, $\alpha = 0.05$). All other differences are not significant.

Trial Length (sec)	Mean	Median
All Trials	226	150
Names Shown	258	165
Names Hidden	196	139

Table 1. Average seconds spent per trial as well as for trials where the song and artist names were shown or hidden.

5.2 Knowing the Names

Visibility of song and artist names had a large influence on how subjects evaluated each playlist. Showing the names benefited the metadata-based systems where, as evidenced by the shorter time spent on these trials, subjects made use of this metadata information to make their evaluations. Comparisons between each pair of algorithms are summarized in Figure 3. Of particular note is the comparison between the two metadata-based systems. When the names are shown, we see in Figure 3(a) that subjects actually rate the Artist Similarity playlists slightly better than the Genius playlists. This may indicate that social or visual cues are, at times, more salient than acoustic similarity or that, given some “explanation” of how a playlist is built, listeners are more forgiving of acoustic mismatches [3]. However, when the names are hidden, and subjects must base their judgements on the acoustics alone, Genius is overwhelmingly superior (Figure 3(b)).

5.3 Familiar and Liked Songs

The effects of subjects’ familiarity with the seed song is shown in Figure 5. The effect of affinity for the seed song was qualitatively almost identical and is not shown. In both cases, Genius benefits from decreased familiarity or affinity while the Artist Similarity method suffers. In other words, when subjects did not know (or like) a song, and presumably could make less use of artist associations, they preferred Genius’ recommendations. This is a strong indication that Genius does not just average over artists but determines song-specific similarities. The only statistically-significant

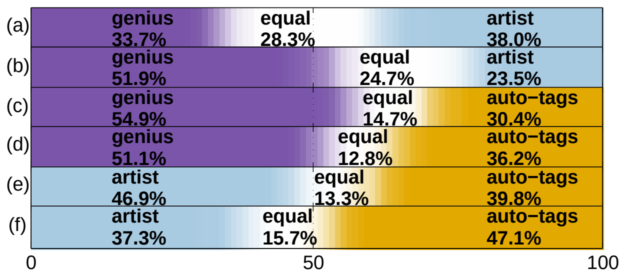


Figure 3. Head-to-head playlist comparisons over the condition where **song and artist names are shown** (a),(c)&(e) **or hidden** (b),(d)&(f). When names are shown (a), Artist Similarity outperforms Genius, but suffers significantly when names are hidden (b) (Chi-square test for independence, $\alpha = 0.05$). The content-based system always benefits when names are hidden (d),(f), forcing subjects to consider acoustics.

change between these conditions (familiar / unfamiliar or liked / not liked) is the reversal in ratings for the Artist and Tag Similarity methods. When familiar with the seed song, subjects were able to appreciate similar artists in the Artist Similarity playlists but, in the absence of this prior knowledge, acoustic similarity prevailed.

5.4 Content Similarity from Collaborative Filtering

We have seen that collaborative filtering finds similarities between songs, not just artists. Can collaborative filtering based on usage and purchase metadata actually capture similarity in acoustic content? To answer this question, we consider trials where subjects were *unfamiliar* with the seed song and where the names of the songs and artists were *hidden*. This removes the influence of song familiarity and artist associations so that subjects' evaluations are based only on acoustic similarity. We also required that subjects *like* the seed song so that they had sufficient motivation and experience with the genre to make relevant evaluations (many subjects reported that they felt unwilling or unable to evaluate songs they disliked). The outcome is shown in Figure 4 where it can be seen that Genius now performs at the same level as the system based solely on acoustic content. This agrees with the findings of Baccigalupo et al.[16] who provide evidence that information about song associations discovered from social playlists can be used to derive genre affinities i.e., collaborative filtering data can be used to derive aspects of acoustic similarity.

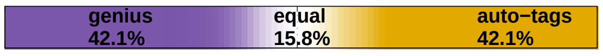


Figure 4. Genius captures content. When subjects were unfamiliar with a seed song that they liked and had no information about song and artist names (26 trials), Genius matches the performance of the content-based system.

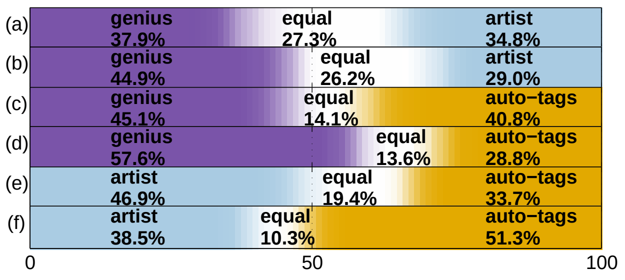


Figure 5. Head-to-head playlist comparisons over the condition where subjects are **familiar** (a),(c),&(e) **or unfamiliar** (b),(d)&(f) **with the seed song**. There is a significant difference between (e) and (f) where the content-based Tag Similarity system is more effective than Artist Similiarity when the seed song is not familiar. (Chi-square test for independence, $\alpha = 0.05$).

5.5 Bad Songs

Table 2 examines the "bad songs" in each playlist that subjects felt did not fit well with the seed song. Overall, the playlist with fewer bad songs won in 81% of trials.

Genius	1.30
Similar Artist	1.18
Similar Tags	1.33
Random	2.56

Trial Winner	1.20
Trial Loser	1.80

Table 2. Average "bad songs" in playlists from each system as well as the average for the winner and loser of each trial.

6. SMARTER THAN GENIUS

While Genius performs as well or better than the metadata- and content-based systems on our test collection of popular music, it is unable to make recommendations from the large "long tail" of new, undiscovered music. We now consider how a music recommender system could take advantage of both content-based information and metadata, when available, to perform as well or better than Genius, without the need for massive amounts of user data.

6.1 Balancing Content and Metadata

Table 3 quantifies the competing influences of artist and acoustic similarity. We show the average artist similarity and auto-tag KL divergence between the seed songs and all the songs from playlists generated by each recommender system. These measures are also shown for all the bad songs. By design, the content-based system has minimum KL and, although it can only access artist information indirectly through acoustics, it captures artist similarity at a better-than-random level. Though they produce good recommendations, both Genius and the Artist Similarity systems have significantly higher KL. This indicates that simply minimizing divergence between semantic descriptions will not produce perfect recommendations. Likewise, recommending similar artists is not sufficient as many bad songs had high artist similarity. Table 2 indicates that a recommender should avoid bad songs with very large semantic

	Artist Similarity	Tag KL Divergence
Genius	19.8	0.81
Similar Artists	**44.5**	0.89
Similar Tags	5.0	**0.14**
Random	1.1	1.15
Bad Songs	16.9	1.20

Table 3. Average artist similarity (between 0 and 100) and auto-tag KL divergence (larger means less similar) between a seed song and playlist songs recommended by each system as well as for bad songs produced by all systems.

differences (high KL divergence) while also making sure to include some clearly similar artists. For example, in 14 of the 50 playlists tested, Genius recommended a song by the *same* artist as the seed song, a simple way to enhance perception of the relevance of the recommendations.

6.2 Musical factors influencing playlist evaluations

Stage 3 of our experiment asked subjects to indicate how well the playlist songs matched the seed song on six different musical cues: similar style (genre), sound, artist, energy, instruments and lyrics. Subjects could indicate that a particular factor was most relevant to either playlist, even the one they had deemed inferior in stage 2. Figure 6 displays the percentage of trials where each system best manifested these factors. Genius playlists often match the styles (47%) and sounds (53%) of the seed song while, predictably, the content-based Similar Tags system rarely returns similar artists (26%). The percentages below the x-axis in Figure 6 indicate how often each factor was cited as a favorable influence (subjects were not required to mark these influences). Similarity between the sound of the seed and the playlist was the most frequently cited factor (82%) while similar lyrics rarely influenced playlist evaluation (36%).

7. CONCLUSIONS

We find that Genius' collaborative filtering approach, which essentially captures the wisdom of the crowds, performs well on a test collection of popular music. By removing evaluator bias resulting from artist names and song familiarity, we show that Genius captures song-specific aspects of acoustic similarity, as can be derived from a purely content-based system. Thus, for exploring the long tail, a content-based recommender can be expected to perform similarly to Genius, *if* collaborative filtering data were available.

We discover that seeing song and artist names has a significant effect on how a playlist is evaluated, indicating that recommender systems must be designed with applications in mind. We highlight the most influential factors on similarity evaluation and suggest that balancing content analysis to avoid bad songs with metadata similarity to provide transparent recommendations can help build smarter music recommender systems.

8. ACKNOWLEDGEMENTS

We thank Sarah Creel and Brian McFee for their advice. GL is supported by NSF grant DMS-MSPA 062540922.

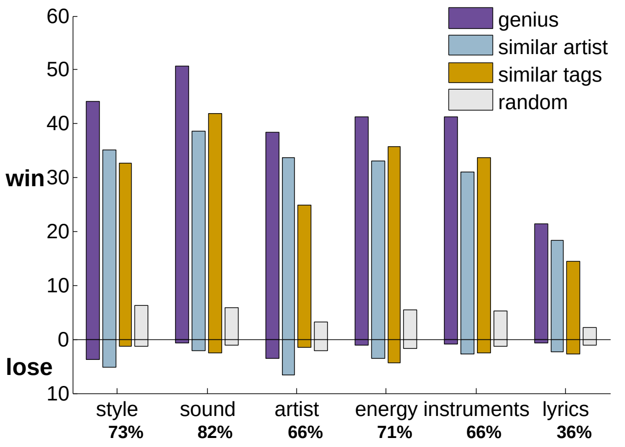


Figure 6. Musical factors influence song similarity. Y-axis shows how often each recommender system best matched musical factors of the seed song, averaged over trials evaluating that system (win or lose). Below the x-axis is the percentage of total trials where each factor was an influence.

9. REFERENCES

[1] S. Cunningham, D. Bainbridge, and A. Falconer. More of an art than a science: Supporting the creation of playlists and mixes. In *ISMIR*, 2006.

[2] M. Salganik, P. Dodds, and D. Watts. Experimental study of inequality and unpredictability in an artificial cultrural market. *Science*, 311(5762):854–856, 2006.

[3] O. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *RecSys*, 2008.

[4] J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *ISMIR*, 2002.

[5] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. In *ISMIR*, 2008.

[6] L. Xiao, L. Liu, F. Seide, and J. Zhou. Learning a music similarity measure on automatic annotations with application to playlist generation. In *ICASSP*, 2009.

[7] B. Fields, C. Rhodes, and M. Casey. Social playlists and bottleneck measurements: Exploiting musician social graphs using content-based dissimilarity and pairwise maximum flow values. In *ISMIR*, 2008.

[8] F. Vignoli and S. Pauws. A music retrieval system based on user-driven similarity and its evaluation. In *ISMIR*, 2005.

[9] L. Barrington, A.B. Chan, D. Turnbull, and G. Lanckriet. Audio information retrieval using semantic similarity. In *ICASSP*, 2007.

[10] Gracenote Inc. Automatic identification of sound recordings. US Patent Number 7,328,153, 2008.

[11] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, February 2008.

[12] N. Rasiwasia, N. Vasconcelos, and P. Moreno. Bridging the gap: Query by semantic example. *IEEE Trans. on Multimedia*, 2007.

[13] J.S.Downie. Music Information Retrieval eXchange. *ISMIR*, 2007.

[14] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *ISMIR*, 2005.

[15] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A large-scale evalutation of acoustic and subjective music-similarity measures. *Computer Music Journal*, pages 63–76, 2004.

[16] C. Baccigalupo, E. Plaza, and J. Donaldson. Uncovering affinity of artists to multiple genres from social behavior data. In *ISMIR*, 2008.

[17] E. Law and L vonAhn. Input-agreement: A new mechanism for collecting data using human computation games. In *ACM CHI*, 2009.

TAG INTEGRATED MULTI-LABEL MUSIC STYLE CLASSIFICATION WITH HYPERGRAPH

Fei Wang, Xin Wang, Bo Shao, Tao Li
Florida International University
{feiwang,xwang009,bshao001,taoli}@cs.fiu.edu

Mitsunori Ogihara
University of Miami
ogihara@cs.miami.edu

ABSTRACT

Automatic music style classification is an important, but challenging problem in music information retrieval. It has a number of applications, such as indexing of and searching in musical databases. Traditional music style classification approaches usually assume that each piece of music has a unique style and they make use of the music contents to construct a classifier for classifying each piece into its unique style. However, in reality, a piece may match more than one, even several different styles. Also, in this modern Web 2.0 era, it is easy to get a hold of additional, indirect information (e.g., music tags) about music. This paper proposes a multi-label music style classification approach, called *Hypergraph integrated Support Vector Machine* (*HiSVM*), which can integrate both music contents and music tags for automatic music style classification. Experimental results based on a real world data set are presented to demonstrate the effectiveness of the method.

1. INTRODUCTION

Music styles (e.g., Dance, Urban, Pop, and Country) are one of the top-level descriptions of music content. Consequently, automatic *Music Style Classification* (*MSC* for short) is a key step for modern music information retrieval systems [7]. There has already been some work toward automatic music style classification. For example, Qin and Ma [10] introduce an MSC system that takes MIDI as data source and mines frequent patterns of different music. Zhang and Zhou [18] present a study on music classification using short-time analysis along with data mining techniques to distinguish among five music styles. Zhou *et al.* [19] propose a Bayesian inference based decision tree model to classify the music into pleasurable and sorrowful music. Although these methods are highly successful, two major limitations exist.

- These are single-label methods in that they can assign only one style label, but many pieces of music may map to more than one style.
- They only make use of the music content information. However, with the rapid development of web technologies, we can easily obtain much richer information of the music (e.g., tags, lyrics, and user comments). How to incorporate these pieces of information into the MSC process effectively is a problem worthy of researching.

THE WORK IS PARTIALLY SUPPORTED BY NSF GRANTS IIS-0546280, DMS-0844513, AND CCF-0830659.

In this paper, we propose a multi-label MSC method that can integrate three types of information: (1) audio signals (MFCC coefficients, STFT, DWCH); (2) music style correlations; (3) music tag information and correlations. Specifically, we construct two hyper-graphs, one on music style labels and the other on music tags, where the vertices on the hypergraphs correspond to the data points, and the hyperedges correspond to the music styles and the tags, respectively. We first integrate those two hypergraphs to obtain a unified hypergraph. Next, assuming that similar music tends to have similar style labels on the hypergraph, we propose a new, SVM-like multilabel ranking algorithm. The algorithm uses a hypergraph Laplacian regularizer and can be efficiently solved by the dual coordinate descent method. Finally, we propose a predictor of the size of label set to determine the number of labels assigned to for each piece of music independently. To demonstrate the efficiency and effectiveness of our proposed method, we conducted a set of experiments applying the method to a real world data.

The rest of this paper is organized as follows. In Section 2 we briefly introduce preliminaries on our key concept, the hypergraph. In Section 3 we describe our HiSVM algorithm. We describe in Section 4 the audio features extracted from the data set as well as the style and tag information of the data set. We present the results of experiments in Section 5 and conclude the paper in Section 6.

2. PRELIMINARIES

A *hypergraph* is a generalization of a graph, in which edges, called hyperedges, may connect any positive number of vertices [1, 11]. Formally, a hypergraph $\mathcal{G}$ is a pair $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of *vertices* and $\mathcal{E} \subseteq 2^{\mathcal{V}} - \emptyset$ is a set of *hyperedges*. An *edge-weighted hypergraph* is one in which each hyperedge is assigned a weight. We use $w(e)$ to denote the weight given to e. The *degree* of a hyperedge e, denoted as $\delta(e)$, is the number of vertices in e. For a standard graph (sometimes called a "2-graph") the value of δ is 2 for all edges. The degree $d(v)$ of a vertex v is $d(v) = \sum_{v \in e, e \in \mathcal{E}} w(e)$. The *vertex-edge incidence matrix* $\mathbf{H} \in \mathbb{R}^{|V| \times |E|}$ is defined as: $h(v, e) = 1$ if $v \in e$ and 0 otherwise. We thus have

$$d(v) = \sum\nolimits_{e \in \mathcal{E}} w(e)h(v, e) \tag{1}$$

$$\delta(e) = \sum\nolimits_{v \in \mathcal{V}} h(v, e). \tag{2}$$

Let $\mathbf{D}_e$ (respectively, $\mathbf{D}_v$ and $\mathbf{W}$) be the diagonal matrix whose diagonal entries are $d(v)$ (respectively, $\delta(e)$, and $w(e)$).

The *graph Laplacian* is the discrete analog of the Laplace-Beltrami operator on compact Riemannian manifolds [12]. The *graph Laplacian* has been widely used in unsupervised learning (e.g., spectral clustering [9]) and semi-supervised learning (e.g. [16, 20]). Below we will sketch a commonly used algorithm by Chung [3], called the *Clique Expansion Algorithm*, for constructing the hypergraph Laplacian.

The Clique Expansion Algorithm constructs a traditional 2-graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$ from the original hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and views the Laplacian of $\mathcal{G}_c$ to be the Laplacian of $\mathcal{G}$. Suppose $\mathcal{V}_c = \mathcal{V}$ and $\mathcal{E}_c = \{(u, v) | u, v \in e, e \in \mathcal{E}\}$. The edge weight $w_c(u, v)$ of $\mathcal{G}_c$ is defined by

$$w_c(u, v) = \sum\nolimits_{u,v \in e, e \in E} w(e) \tag{3}$$

An interpretation of this definition is that the edge weight matrix, $\mathbf{W}_c$, of $\mathcal{G}_c$ can be expressed as

$$\mathbf{W}_c = \mathbf{HWH}^T \tag{4}$$

Let $\mathbf{D}_c$ be the diagonal matrix such that

$$\mathbf{D}_c(u, u) = \sum_v w_c(u, v).$$

Then the combinatorial Laplacian, $\mathbf{L}_c$, of $\mathcal{G}_c$ is given by

$$\mathbf{L}_c = \mathbf{D}_c - \mathbf{W}_c = \mathbf{D}_c - \mathbf{HWH}^T \tag{5}$$

and the normalized Laplacian, $\mathbf{L}_n$, is given by

$$\mathbf{L}_n = \mathbf{I} - \mathbf{D}_c^{-1/2}\mathbf{HWH}^T\mathbf{D}_c^{-1/2}. \tag{6}$$

From Eq. (5) and (6), we have

$$\mathbf{L}_n = \mathbf{D}_c^{-1/2}\mathbf{L}_c\mathbf{D}_c^{-1/2}. \tag{7}$$

In our music style classification, we construct two hypergraphs: the style hypergraph $\mathcal{G}_s$ and the tag hypergraph $\mathcal{G}_t$. The vertices of $\mathcal{G}_s$ and $\mathcal{G}_t$ are simply the data points. The hyperedges of $\mathcal{G}_s$ correspond to the style labels, i.e., each hyperedge in $\mathcal{G}_s$ contains all the data points that belong to a specific style category. Similarly, each hyperedge of $\mathcal{G}_t$ contains all the data points that own the corresponding tag. Figure 1 shows an intuitive example on the music style and tag hypergraphs.

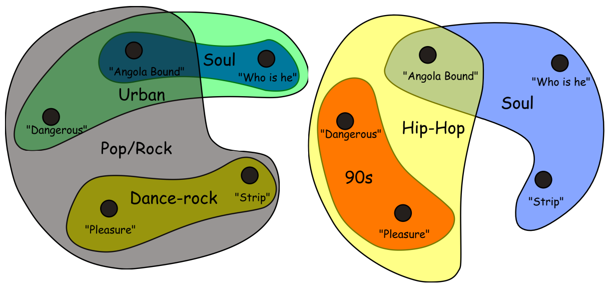


Figure 1. An example of the music style (left) and tag (right) hypergraph. The nodes on the hypergraphs correspond to the music "Angola Bond", "Who is he", "Dangerous", "Pleasure", and "Strip". The regions of different colors correspond to the different hyperedges. The hyperedges correspond to music styles in the left panel and to music tags in the right panel.

3. MULTI-LABEL LEARNING WITH HYPERGRAPH REGULARIZATIONS

In this section we will present in detail our proposed multi-label classification algorithm with hypergraph regularization. Suppose there are n training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where each instance $\boldsymbol{x}_i$ is drawn from some domain $\mathcal{X} \subseteq \mathbb{R}^m$ and its label y_i is a subset of the output label set $\mathcal{Y} = \{1, \cdots, k\}$. For example, if x_i belongs to categories 1, 3, and 4, then $y_i = \{1, 3, 4\}$. We use $\mathbf{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^T$ to represent the data feature matrix.

Our basic strategy is to solve the multi-label learning by combing a label ranking problem and a label number prediction problem. That is, for each instance we produce a ranked list of all possible labels, estimate the number of labels for the instance, and then select the predicted number of labels from the list.

Label ranking is the task of inferring a total order over a predefined set of labels for each given instance [5]. Generally, for each category, we define a linear function $f_i(x) = \langle \boldsymbol{w}_i, \boldsymbol{x} \rangle + b_i$ $(i = 1, \cdots, k)$, where $\langle \cdot, \cdot \rangle$ is the inner product and b_i is a bias term. One often deals with the bias term by appending to each instance an additional dimension

$$\boldsymbol{x}^T \leftarrow [\boldsymbol{x}^T, 1], \quad \boldsymbol{w}_i^T \leftarrow [\boldsymbol{w}_i^T, b_i] \tag{8}$$

then the linear function becomes $f_i(\boldsymbol{x}) = \langle \boldsymbol{w}_i, \boldsymbol{x} \rangle$. The goal of label ranking is to order $\{f_i(\boldsymbol{x}), i = 1, \cdots, k\}$ for each instance $\boldsymbol{x}$ according to some predefined empirical loss function and complexity measures. Elisseeff and Weston [6] apply the large margin idea to multi-label learning and present an SVM-like ranking system, called Rank-SVM, given as follows:

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i=1}^{k} \|\boldsymbol{w}_i\|^2 + C\sum_{i=1}^{n} \frac{1}{|y_i||\overline{y}_i|} \sum_{(p,q) \in y_i \times \overline{y}_i} \xi_{ipq} \\ \text{s.t.} \quad & \langle \boldsymbol{w}_p - \boldsymbol{w}_q, \boldsymbol{x}_i \rangle \geq 1 - \xi_{ipq}, (p, q) \in y_i \times \overline{y}_i \\ & \xi_{ipq} \geq 0 \end{aligned} \tag{9}$$

where $C \geq 0$ is a penalty coefficient that trades off the empirical loss and model complexity, $\overline{y}_i$ is the complementary set of y_i in $\mathcal{Y}$, $|y_i|$ is the cardinality of the set y_i, i.e., the number of elements of the set y_i, and $\xi_{ipq}(i = 1, \cdots, n; (p,q) \in y_i \times \overline{y}_i)$ are slack variables. The margin term $\sum_{i=1}^{k} \|\boldsymbol{w}_i\|^2$ controls the model complexity and improves the model generalization performance. Although this approach performs better than Binary-SVM in many cases, it still does not model the category correlations clearly. Next, we will describe how to construct a hypergraph to exploit the category correlations and how to incorporate the hypergraph regularization into the problem in the form of Eq. (9).

3.1 Basic Framework

To model the correlations among different categories effectively, a hypergraph is built where each vertex corresponds to one training instance and a hyperedge is constructed for each category which includes all the training instances relevant to the same category. Here, we apply the Clique Expansion algorithm to construct the similarity matrix of the hypergraph. It means that the similarity of two instances is proportional to the sum of the weights of their common categories, thereby captures the higher order relations among different categories. This kind of hypergraph structure was used in the feature extraction by spectral learning [13]. However, we consider how to apply the relation information encoded in the hypergraph to directly design the multi-label learning model. Intuitively, two instances tend to have a large overlap in their assigned categories if they share high similarity in the hypergraph. Formally, this smoothness assumption can be expressed using the hypergraph Laplacian regularizer, trace($\widehat{\mathbf{F}}^T\mathbf{L}\widehat{\mathbf{F}}$). Therefore we can introduce the smoothness assumption into problem Eq. (9) and obtain

$$\begin{aligned}
\min \quad & \frac{1}{2}\sum_{i=1}^{k}\|\boldsymbol{w}_i\|^2 + \frac{1}{2}\lambda \text{trace}(\widehat{\mathbf{F}}^T\mathbf{L}\widehat{\mathbf{F}}) + \\
& C\sum_{i=1}^{n}\frac{1}{|y_i||\overline{y}_i|}\sum_{(p,q)\in y_i\times\overline{y}_i}\xi_{ipq} \\
\text{s.t.} \quad & \langle \boldsymbol{w}_p - \boldsymbol{w}_q, \boldsymbol{x}_i\rangle \geq 1 - \xi_{ipq}, (p,q) \in y_i \times \overline{y}_i \\
& \xi_{ipq} \geq 0 \qquad (10)
\end{aligned}$$

Here $\widehat{\mathbf{F}}$ is the matrix of label prediction; that is, it is the $n \times k$ matrix $(f_j(\boldsymbol{x}_i))$, $1 \leq i \leq n$, $1 \leq j \leq k$. Also, $\mathbf{L}$ is the $n \times n$ hypergraph Laplacian and $\lambda \geqslant 0$ is a constant that controls the model complexity in the intrinsic geometry of input distribution.

3.2 Optimization Strategy

Problem (10) is a linearly constrained quadratic convex optimization problem. To solve it, we first introduce a dual set of variables, one for each constraint, i.e., $\alpha_{ipq} \geq 0$ for $\langle \boldsymbol{w}_p - \boldsymbol{w}_q, \boldsymbol{x}_i\rangle - 1 + \xi_{ipq} \geq 0$ and η_{ipq} for $\xi_{ipq} \geq 0$. After some linear algebraic derivation, we obtain the dual of Problem (10) as

$$\begin{aligned}
\min g(\boldsymbol{\alpha}) \quad = \quad & \frac{1}{2}\sum_{p=1}^{k}\sum_{h,i=1}^{n}\beta_{ph}\beta_{pi}\boldsymbol{x}_h^T(I + \lambda X^T L X)^{-1}\boldsymbol{x}_i \\
& -\sum_{i=1}^{n}\sum_{(p,q)\in y_i\times\overline{y}_i}\alpha_{ipq} \\
\text{s.t.} \quad & 0 \leq \alpha_{ipq} \leq \frac{C}{|y_i||\overline{y}_i|} \qquad (11)
\end{aligned}$$

where $\boldsymbol{\alpha}$ denotes the set of dual variables α_{ipq} and I is the $(m+1) \times (m+1)$ identity matrix.

Once the variables α_{ipq} that minimize $g(\boldsymbol{\alpha})$ are obtained, we can compute w_p by

$$\boldsymbol{w}_p = (I + \lambda X^T L X)^{-1}\sum_{i=1}^{n}\beta_{pi}\boldsymbol{x}_i \qquad (12)$$

where

$$\beta_{pi} = \sum_{(j,q)\in y_i\times\overline{y}_i} t_{ijq}^p \alpha_{ijq} \qquad (13)$$

$$t_{ijq}^p = \begin{cases} 1 & j = p \\ -1 & q = p \\ 0 & \text{if } j \neq p \text{ and } q \neq p \end{cases} \qquad (14)$$

Compared to the primal optimization problem, the dual has $k(m+1)$ less variables and includes simple box constraints. The dual can be solved by the dual coordinate descent algorithm shown in Algorithm 1.

Algorithm 1 A dual coordinate descent method for HiSVM

```
Start with $\boldsymbol{\alpha} = \mathbf{0} \in \mathbb{R}^{n_\alpha}$ ($n_\alpha = \sum_{i=1}^{n}|y_i||\overline{y}_i|$), and the corresponding $\boldsymbol{w}_i = \mathbf{0}$ ($i = 1, \cdots, k$)
while 1 do
  for $i = 1, \cdots, n$ and $(j,q) \in y_i \times \overline{y}_i$ do
    1. $G = (\boldsymbol{w}_p - \boldsymbol{w}_q)^T\boldsymbol{x}_i - 1$
    2. $PG = \begin{cases} G & \text{if } 0 < \alpha_{ipq} < \frac{C}{|y_i||\overline{y}_i|} \\ \min(0,G) & \text{if } \alpha_{ipq} = 0 \\ \max(0,G) & \text{if } \alpha_{ipq} = \frac{C}{|y_i||\overline{y}_i|} \end{cases}$
    3. If $|PG| \neq 0$,
       $\alpha_{ipq}^* \leftarrow \min\left(\max\left(\alpha_{ipq} - \frac{G}{2A_{ii}}, 0\right), \frac{C}{|y_i||\overline{y}_i|}\right)$
       $\boldsymbol{w}_p \leftarrow \boldsymbol{w}_p + (\alpha_{ipq}^* - \alpha_{ipq})(I + \lambda X^T L X)^{-1}\boldsymbol{x}_i$
       $\boldsymbol{w}_q \leftarrow \boldsymbol{w}_q - (\alpha_{ipq}^* - \alpha_{ipq})(I + \lambda X^T L X)^{-1}\boldsymbol{x}_i$
  end for
  if $\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\|/\|\boldsymbol{\alpha}\| < \epsilon$(e.g. $\epsilon = 0.01$) then
    Break
  end if
  $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$
end while
```

3.3 Predicting the Size of Label Set

So far we have only provided a label ranking algorithm. To identify the final labels of data, we need to design an appropriate threshold for each instance to determine the size of its corresponding label set. Here, we adopt the strategy proposed by Elisseeff and Weston [6], which treats threshold designing as a supervised learning problem. More concretely, for each instance $\boldsymbol{x}$, define a threshold function $h(\boldsymbol{x})$ and the size of label set $s(\boldsymbol{x}) = \|\{j \mid f_j(\boldsymbol{x}) > h(\boldsymbol{x}), j = 1, \cdots, k\}\|$. Our goal is to obtain $h(\boldsymbol{x})$ through a supervised learning method. For the training data $\boldsymbol{x}_i$, its label ranking values, $f_1(\boldsymbol{x}_i), \cdots, f_k(\boldsymbol{x}_i)$, can be given by the foregoing ranking algorithm, and its corresponding threshold $h(\boldsymbol{x}_i)$ is simply defined by

$$h(\boldsymbol{x}_i) = \frac{1}{2}(\min_{j \in y_i}\{f_j(\boldsymbol{x}_i)\} + \max_{j \in \overline{y}_i}\{f_j(\boldsymbol{x}_i)\})$$

Once the training data $(\boldsymbol{x}_1, h(\boldsymbol{x}_1)), \cdots, (\boldsymbol{x}_u, h(\boldsymbol{x}_u))$ are generated, we can use off-the-shelf learning methods to learn $h(\boldsymbol{x})$. In this paper, Linear Support Vector Regression [15] has been adopted to solve $h(\boldsymbol{x})$. We note there that all the label ranking based algorithms toward multi-label learning can use this postprocessing approach to predict the size of label set.

4. DATA DESCRIPTION

For experimental purpose, we created a data set consisting of 403 artists. For each artist, we include a representative song and also obtain the style and tag description.

4.1 Music Content Features

For each song, a single vector of 80 components is extracted. The single vector contains the following audio features:

1) Mel-Frequency Cepstral Coefficients (MFCC): Mel-Frequency Cepstral Coefficients (MFCC) is a feature set that is very popular in speech processing. MFCC is designed to capture short-term spectral based features. The features of MFCC are computed as follows: First, for each frame, the logarithm of the amplitude spectrum based on short-term Fourier transform is calculated, where the frequencies are divided into thirteen bins using the Mel-frequency scaling. Next, this vector is decorrelated using discrete cosine transform. The resulting vector is called the MFCC vector. In our experiments, we compute the mean and variance of each bin over the frame for the two vectors (before and after decorrelation). Thus, for each sample, MFCC occupies 52 components.

2) Short-Term Fourier Transform Features (STFT): This is a set of features related to timbral textures and is not captured using MFCC. It consists of the following types of features: Spectral Centroid, Spectral Rolloff, Spectral Flux, Zero Crossings, and Low Energy. More detailed descriptions of STFT can be found in [14]. In our experiments, we compute the mean for all types and the variance for all but zero crossings. STFT thus occupies 12 components.

3) Daubechies Wavelet Coefficient Histograms (DWCH): Daubechies wavelet filters are a set of filters that are popular in image retrieval (see [4]). The Daubechies Wavelet Coefficient Histograms, proposed in [8], are features extracted in the following manner: First, the Daubechies-8 (db8) filter with seven levels of decomposition (or subbands) is applied to 30 seconds of monaural audio signals. Then, the histogram of the wavelet coefficients is computed for each subband. Then the first three moments of each histogram, i.e., the average, the variance, and the skewness, are calculated from each subband. In addition, the subband energy, defined as the mean of the absolute value of the coefficients, is computed from each subband. More details of DWCH can be found in [8].

4.2 Music Tag Information

Music tags are descriptions given by visitors or music tag editors from the website to express their idea on the music artists. Tags can be as simple as a word or as complicated as a whole sentence. Popular tags are terms like “rock,” “black metal,” and “indie pop.” Long tags are like “I love you baby can I have some more.” The tags are not as formal as style description created by music experts, but they give us ideas of how large population music listeners think about the music artists. In our experiments, tag data was collected from the popular music website http://www.last.fm. In order to understand how important a tag is, and how accurately it reflects the characteristics of an artist, the frequencies of all the tags to describe the artists (tag counts) were also used in the experiments.

A total of 8,529 tags were collected. Each artist has at most 100 tags and at least 3 tags. On average, each artist is associated with 89.5 tags. Note that, each artist may be described by some tags for more than once, for example, Michael Jackson has been tagged with “80s” for 453 times.

4.3 Music Style Information

Style data were collected from All Music Guide (http://www.allmusic.com). These data are created by music experts to describe the characteristics of music artists. Style terms are nouns like Rock & Roll, Greek Folk, and Chinese Pop as well as adjectives like Joyous, Energetic, and New Romantic. Styles for each artist/track are different from the music tags described in the above, since each style name for one artist appears only once.

A total of 358 styles were found. Each artist has at most 12 and at least one style type. On average, every artist is associated with 4.7 style labels.

5. EXPERIMENTS

We performed experiments on HiSVM and four real-world multi-label learning models arising in text categorization, image classification, video indexing and gene function prediction. Comparisons are made with Binary-SVM and Rank-SVM [6].

5.1 Methods and Experimental Setup

The data set information we used to evaluate our proposed approach has been introduced in the previous section, where we use 70% of the data for training (282 pieces total), and the remaining 30% for testing (121 pieces total). Here, the four models used for multi-label learning are as follows:

- Binary-SVM. In this model, first, for each category, train a linear SVM classifier independently. Then, the labels for each test instance can be obtained by aggregating the classification results from all the binary classifiers. Here, we use LIBSVM [2] to train the linear SVM classifiers.
- Rank-SVM [6]. In this model, first, using Eq. (9), we implement Algorithm 1 ($\lambda = 0$) to train a linear label ranking system. We then apply the prediction method for the size of label set described in Section 3.3 to design the threshold model. Finally, for each test instance, we combine the label ranking and threshold models, thereby infer its labels.
- HiSVM. This is our proposed algorithm. The algorithm is composed of three steps: (1) we implement Algorithm 1 to achieve a linear label ranking system; (2) we apply the method in Section 3.3 to design the threshold model; (3) for each test instance, we combine the label ranking and threshold models to infer its labels.
- HSVM. HSVM is the style Hypergraph regularized SVM method, which is the same as the HiSVM method except that it only makes use of the style hypergraph and does not use the tag hypergraph.
- GSVM. GSVM is similar to HiSVM except we construct a traditional 2-graph where each vertex represents one training instance in GSVM rather than a hypergraph. In order to compute the Laplacian, the weight w_{ij} of the edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as follows

  $$w_{ij} = \exp(-\rho \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) \qquad (15)$$

  where ρ is a nonnegative constant. Apparently, the category correlation information is not used during the construction of 2-graph in GSVM.

Some details of HiSVM are in order. We use Eq. (5) to construct both the style hypergraph Laplacian L_s and the tag hypergraph Laplacian L_t, where the weight $w(e)$ of the hyperedge is calculated by

$$w(e) = \exp(-\nu \bar{d}_e) \qquad (16)$$

Here ν is a nonnegative constant, and $\bar{d}_e$ is the average intra-class distance (N.B. Each hyperedge corresponds to one specific style or tag):

$$\bar{d}_e = \frac{\sum_{u,v \in e} \|\boldsymbol{x}_u - \boldsymbol{x}_v\|^2}{\delta(e)(\delta(e)-1)} \qquad (17)$$

The smaller the average intra-class distance, the larger the corresponding hyperedge weight. Finally we combine L_s and L_t to obtain a unified hypergraph Laplacian L by

$$L = \frac{1}{2}(L_s + L_t)$$

Table 1. A contingency table

	YES is correct	NO is correct
Assigned YES	a	b
Assigned NO	c	d

which is used in the rest of the inferences and experiments.

In the above four models, it is necessary to identify the best value of model parameters such as C, λ and ν on the training data. Here, the grid search method with 5-fold cross validation is used to determine the best parameter values. For the penalty coefficient C in the Linear SVM, we tune it from the grid points $\{10^{-6}, 10^{-5}, \cdots, 10^0, 10^1, \cdots, 10^6\}$; for the tradeoff parameter λ, we tune it from the grid points $\{10^{-6}, 10^{-5}, \cdots, 10^0, 10^1, \cdots, 10^6\}$; the scale parameter ν and ρ are tuned from the grid points $\{2^{-6}, 2^{-5}, \cdots, 2^0, 2^1, \cdots, 2^6\}$.

5.2 Evaluation Metrics

We choose two measures, F_1 *Macro* and F_1 *Micro* [17], as the evaluation metrics for multi-label learning. Suppose there are a total of S style categories. Then for each category, we can construct a contingency table as follows: Let a (respectively, b) be the number of pieces that are correctly assigned (respectively, not correctly assigned) to this style category, and let c (respectively, d) be the number of pieces that are incorrectly rejected (respectively, correctly rejected) by this style category (see Table 1). Let $r = a/(a+c)$ and $p = a/(a+b)$, where the former is called the recall and the latter the precision. Then the F_1 score of this style category can be computed as

$$F_1 = \frac{2pr}{p+r} \qquad (18)$$

The F_1 *Macro* can be computed by first calculating the F_1 scores for the per-category contingency tables and then averaging these scores to compute the global means. F_1 *Micro* can be computed by first constructing a global contingency table, each of whose cell value is the sum of the corresponding cells in the per-category contingency tables, and then use this global contingency table to compute the Micro F_1 score.

5.3 Experimental Results

Table 2 illustrates the experimental results on our HiSVM algorithm along with the four other methods on the data set. The values in Table 2 are the F_1 Micro values and F_1 Macro values averaged over 50 independent runs together with their standard deviations. From the table we can clearly observe the following:

- Multi-label methods perform better than the simple Binary-SVM method.
- The consideration of label correlations is helpful for the final algorithm performance.

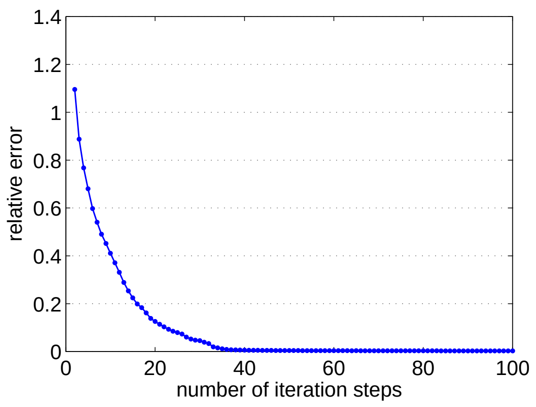


Figure 2. The relative error $\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\|/\|\boldsymbol{\alpha}\|$ vs. iteration step plot of our proposed dual coordinate descent algorithm for solving HiSVM.

Table 2. Performance comparisons of four models on the Last.fm dataset

Methods	F1 *Macro*	F1 *Micro*
Binary-SVM	0.4231 ± 0.0025	0.4317 ± 0.0103
Rank-SVM	0.4526 ± 0.0114	0.4733 ± 0.0036
GSVM	0.5018 ± 0.0054	0.5244 ± 0.0103
HSVM	0.5365 ± 0.0120	0.5509 ± 0.0072
HiSVM	$\mathbf{0.5613 \pm 0.0069}$	$\mathbf{0.5802 \pm 0.0116}$

- Hypergraph regularization is better than flat two-graph regularization because it can incorporate the high-order label relationships naturally.
- The incorporation of tag information is helpful for the final classification performance.

Figure 2 shows how the relative error $\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\|/\|\boldsymbol{\alpha}\|$ varies with the process of iteration using the dual coordinate descent method introduced in Algorithm 1. From the figure we clearly see that with the process of coordinate descent, the relative error will decrease and it takes approximately 30 steps to converge. This validates the correctness of our algorithm experimentally.

6. CONCLUSION

We propose a novel multi-label classification method called Hypergraph integrated SVM (HiSVM) for music style classification. Our method can not only take into account the music style correlations, but also the music tag correlations. We also propose an efficient dual coordinate descent algorithm to solve it, and finally experimental results on a real world data set are presented to show the effectiveness and correctness of our algorithm.

7. REFERENCES

[1] C. Berge. Graphs and Hypergraphs. Elsevier, 1973.

[2] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, 2001, software available at `http://www.csie.ntu.edu.tw/cjlin/libsvm`.

[3] F. R. K. Chung. The laplacian of a hypergraph. In Expanding Graphs, DIMACS Series, Vol. 10, AMS, 1993.

[4] I. Daubechies. Ten lectures on wavelets. SIAM, 1992.

[5] O. Dekel, C. D. Manning, and Y. Singer. Log-linear models for label ranking. In Proc. of NIPS, 2003.

[6] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In Proc. of NIPS, 2001.

[7] T. Li and M. Ogihara. Towards intelligent music information retrieval. IEEE Transactions on Multimedia 8(3):564-575, 2006.

[8] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In Proceedings of SIGIR, pages 282-289, 2003.

[9] U. von Luxburg. A tutorial on spectral clustering. Max Planck Institute for Biological Cybernetics, Tech. Rep., 2006.

[10] D. Qin and G. Z. Ma. Music style identification system based on mining technology. Computer Engineering and Design. 26, 3094-3096. 2005.

[11] S. Chen, F. Wang and C. Zhang: Simultaneous heterogeneous data clustering based on higher order relationships. ICDM Workshops 2007: 387-392.

[12] S. Rosenberg. The Laplacian on a Remannian manifold. London Math. Soc., 1997.

[13] L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multilabel classification. In Proc. KDD, 2008.

[14] G. Tzanetakis and P. Cook. Music genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5):293-302, 2002.

[15] V. Vapnik. The Nature of Statistical Learning Theory. Springer, 1995.

[16] F. Wang and C. Zhang. Label Propagation Through Linear Neighborhoods. In Proc. ICML, pages 985-992, 2006.

[17] Y. Yang. An evaluation of statistical approaches to text categorization. Information Retrieval 1:69–90, 1999.

[18] Y. B. Zhang and J. Zhou. A study on content-based music classification. In Proc. IEEE Signal Processing and Its Applications, 2003.

[19] Y. Zhou, T. Zhang, and J. Sun. Music style classification with a novel Bayesian model. In Proc. Advanced Data Mining and Appliations, Springer, 2006.

[20] X. Zhu. Semi-supervised learning literature survey. University of Wisconsin-Madision, Technical Report TR 1530, 2006.

EASY AS CBA: A SIMPLE PROBABILISTIC MODEL FOR TAGGING MUSIC

Matthew D. Hoffman
Dept. of Computer Science
Princeton University
mdhoffma at cs.princeton.edu

David M. Blei
Dept. of Computer Science
Princeton University
blei at cs.princeton.edu

Perry R. Cook
Dept. of Computer Science
Dept. of Music
Princeton University
prc at cs.princeton.edu

ABSTRACT

Many songs in large music databases are not labeled with semantic tags that could help users sort out the songs they want to listen to from those they do not. If the words that apply to a song can be predicted from audio, then those predictions can be used both to automatically annotate a song with tags, allowing users to get a sense of what qualities characterize a song at a glance. Automatic tag prediction can also drive retrieval by allowing users to search for the songs most strongly characterized by a particular word. We present a probabilistic model that learns to predict the probability that a word applies to a song from audio. Our model is simple to implement, fast to train, predicts tags for new songs quickly, and achieves state-of-the-art performance on annotation and retrieval tasks.

1. INTRODUCTION

It has been said that talking about music is like dancing about architecture, but people nonetheless use words to describe music. In this paper we will present a simple system that addresses tag prediction from audio—the problem of predicting what words people would be likely to use to describe a song.

Two direct applications of tag prediction are semantic annotation and retrieval. If we have an estimate of the probability that a tag applies to a song, then we can say what words in our vocabulary of tags best describe a given song (automatically annotating it) and what songs in our database a given word best describes (allowing us to retrieve songs from a text query).

We present the Codeword Bernoulli Average (CBA) model, a probabilistic model that attempts to predict the probability that a tag applies to a song based on a vector-quantized (VQ) representation of that song's audio. Our CBA-based approach to tag prediction

- Is easy to implement using a simple EM algorithm.
- Is fast to train.
- Makes predictions efficiently on unseen data.
- Performs as well as or better than state-of-the-art approaches.

2. DATA REPRESENTATION

2.1 The CAL500 data set

We train and test our method on the CAL500 dataset [1, 2]. CAL500 is a corpus of 500 tracks of Western popular music, each of which has been manually annotated by at least three human labelers. We used the "hard" annotations provided with CAL500, which give a binary value $y_{jw} \in \{0, 1\}$ for all songs j and tags w indicating whether tag w applies to song j.

CAL500 is distributed with a set of 10,000 39-dimensional Mel-Frequency Cepstral Coefficient Delta (MFCC-Delta) feature vectors for each song. Each Delta-MFCC vector summarizes the timbral evolution of three successive 23ms windows of a song. CAL500 provides these feature vectors in a random order, so no temporal information beyond a 69ms timescale is available.

Our goals are to use these features to predict which tags apply to a given song and which songs are characterized by a given tag. The first task yields an automatic annotation system, the second yields a semantic retrieval system.

2.2 A vector-quantized representation

Rather than work directly with the MFCC-Delta feature representation, we first vector quantize all of the feature vectors in the corpus, ignoring for the moment what feature vectors came from what songs. We:

1. Normalize the feature vectors so that they have mean 0 and standard deviation 1 in each dimension.
2. Run the k-means algorithm [3] on a subset of randomly selected feature vectors to find a set of K cluster centroids.
3. For each normalized feature vector f_{ji} in song j, assign that feature vector to the cluster k_{ji} with the smallest squared Euclidean distance to f_{ji}.

This vector quantization procedure allows us to represent each song j as a vector $\boldsymbol{n}_j$ of counts of a discrete set of codewords:

$$n_{jk} = \sum_{i=1}^{N_j} 1(k_{ji} = k) \tag{1}$$

where n_{jk} is the number of feature vectors assigned to codeword k, N_j is the total number of feature vectors in song j, and $1(a = b)$ is a function returning 1 if $a = b$ and 0 if $a \neq b$.

This discrete "bag-of-codewords" representation is less rich than the original continuous feature vector representation. However, it is effective. Such VQ codebook representations have produced state-of-the-art performance in image annotation and retrieval systems [4], as well as in systems for estimating timbral similarity between songs [5,6].

3. THE CODEWORD BERNOULLI AVERAGE MODEL

In order to predict what tags will apply to a song and what songs are characterized by a tag, we developed the Codeword Bernoulli Average model (CBA). CBA models the conditional probability of a tag w appearing in a song j conditioned on the empirical distribution $\boldsymbol{n}_j$ of codewords extracted from that song. One we have estimated CBA's hidden parameters from our training data, we will be able to quickly estimate this conditional probability for new songs.

3.1 Related work

One class of approaches treats audio tag prediction as a set of binary classification problems to which variants of standard classifiers such as the Support Vector Machine (SVM) [7,8] or AdaBoost [9] can be applied. Once a set of classifiers has been trained, the classifiers attempt to predict whether or not each tag applies to previously unseen songs. These predictions come with confidence scores that can be used to rank songs by relevance to a given tag (for retrieval), or tags by relevance to a given song (for annotation). Classifiers like SVMs or AdaBoost focus on binary classification accuracy rather than directly optimizing the continuous confidence scores that are used for retrieval tasks, which might lead to suboptimal results for those tasks.

Another approach is to fit a generative probabilistic model such as a Gaussian Mixture Model (GMM) for each tag to the audio feature data for all of the songs manifesting that tag [2]. The posterior likelihood $p(\text{tag}|\text{audio})$ of the feature data for a new song being generated from the model for a particular tag is then used to estimate the relevance of that tag to that song (and vice versa). Although this model tells us how to generate the audio feature data for a song conditioned on a *single* tag, it does not define a generative process for songs with *multiple* tags, and so heuristics are necessary to estimate the posterior likelihood of a set of tags.

Rather than assuming that the audio for a song depends on the tags associated with that song, we will assume that the tags depend on the audio data. This will yield a probabilistic model with a discriminative flavor, and a more coherent generative process than that in [2].

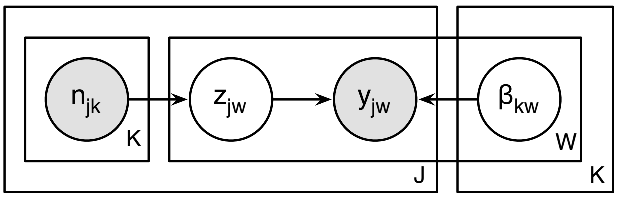


Figure 1. Graphical model representation of CBA. Shaded nodes represent observed variables, unshaded nodes represent hidden variables. A directed edge from node a to node b denotes that the variable b depends on the value of variable a. Plates (boxes) denote replication by the value in the lower right of the plate. J is the number of songs, K is the number of codewords, and W is the number of unique tags.

3.2 Generative process

CBA assumes a collection of binary random variables $\boldsymbol{y}$, with $y_{jw} \in \{0, 1\}$ determining whether or not tag w applies to song j. These variables are generated in two steps. First, a codeword $z_{jw} \in \{1, \ldots, K\}$ is selected with probability proportional to the number of times n_{jk} that that codeword appears in song j's feature data:

$$p(z_{jw} = k|\boldsymbol{n}_j, N_j) = \frac{n_{jk}}{N_j} \tag{2}$$

Then a value for y_{jw} is chosen from a Bernoulli distribution with parameter β_{kw}:

$$\begin{aligned} p(y_{jw} = 1|z_{jw}, \boldsymbol{\beta}) &= \beta_{z_{jw}w} \\ p(y_{jw} = 0|z_{jw}, \boldsymbol{\beta}) &= 1 - \beta_{z_{jw}w} \end{aligned} \tag{3}$$

The full joint distribution over $\boldsymbol{z}$ and $\boldsymbol{y}$ conditioned on the observed counts of codewords $\boldsymbol{n}$ is:

$$p(\boldsymbol{z}, \boldsymbol{y}|\boldsymbol{n}) = \prod_w \prod_j \frac{n_{jz_{jw}}}{N_j} \beta_{z_{jw}w} \tag{4}$$

The random variables in CBA and their dependencies are summarized in figure 1.

3.3 Inference using expectation-maximization

We fit CBA with maximum-likelihood (ML) estimation. Our goal is to estimate a set of values for our Bernoulli parameters $\boldsymbol{\beta}$ that will maximize the likelihood $p(\boldsymbol{y}|\boldsymbol{n}, \boldsymbol{\beta})$ of the observed tags $\boldsymbol{y}$ conditioned on the VQ codeword counts $\boldsymbol{n}$ and the parameters $\boldsymbol{\beta}$. Analytic ML estimates for $\boldsymbol{\beta}$ are not available because of the latent variables $\boldsymbol{z}$. We use the Expectation-Maximization (EM) algorithm, a widely used coordinate ascent algorithm for maximum-likelihood estimation in the presence of latent variables [10].

Each iteration of EM operates in two steps. In the expectation ("E") step, we compute the posterior of the latent variables $\boldsymbol{z}$ given our current estimates for the parameters $\boldsymbol{\beta}$. We define a set of expectation variables h_{jwk} corresponding to the posterior $p(z_{jw} = k|\boldsymbol{n}, \boldsymbol{y}, \boldsymbol{\beta})$:

$$h_{jwk} = p(z_{jw} = k|\boldsymbol{n}, \boldsymbol{y}, \boldsymbol{\beta}) \quad (5)$$

$$= \frac{p(y_{jw}|z_{jw} = k, \boldsymbol{\beta})p(z_{jw} = k|\boldsymbol{n})}{p(y_{jw}|\boldsymbol{n}, \boldsymbol{\beta})} \quad (6)$$

$$= \begin{cases} \frac{n_{jk}\beta_{kw}}{\sum_{i=1}^{K} n_{ji}\beta iw} & \text{if } y_{jw} = 1 \\ \frac{n_{jk}(1-\beta_{kw})}{\sum_{i=1}^{K} n_{ji}(1-\beta iw)} & \text{if } y_{jw} = 0 \end{cases} \quad (7)$$

In the maximization ("M") step, we find maximum-likelihood estimates of the parameters $\boldsymbol{\beta}$ given the expected posterior sufficient statistics:

$$\beta_{kw} \leftarrow \mathbb{E}[y_{jw}|z_{jw} = k, \boldsymbol{h}] \quad (8)$$

$$= \frac{\sum_j p(z_{jw} = k|\boldsymbol{h})y_{jw}}{\sum_j p(z_{jw} = k|\boldsymbol{h})} \quad (9)$$

$$= \frac{\sum_j h_{jwk}y_{jw}}{\sum_j h_{jwk}} \quad (10)$$

By iterating between computing $\boldsymbol{h}$ (using equation 7) and updating $\boldsymbol{\beta}$ (using equation 10), we find a set of values for $\boldsymbol{\beta}$ under which our training data become more likely.

3.4 Generalizing to new songs

Once we have inferred a set of Bernoulli parameters $\boldsymbol{\beta}$ from our training dataset, we can use them to infer the probability that a tag w will apply to a previously unseen song j based on the counts $\boldsymbol{n}_j$ of codewords for that song:

$$p(y_{jw}|\boldsymbol{n}_j, \boldsymbol{\beta}) = \sum_k p(z_{jw} = k|\boldsymbol{n}_j)p(y_{jw}|z_{jw} = k)$$

$$p(y_{jw} = 1|\boldsymbol{n}_j, \boldsymbol{\beta}) = \frac{1}{N_j}\sum_k n_{jk}\beta_{kw} \quad (11)$$

As a shorthand, we will refer to our inferred value of $p(y_{jw} = 1|\boldsymbol{n}_j, \boldsymbol{\beta})$ as s_{jw}.

Once we have inferred s_{jw} for all of our songs and tags, we can use these inferred probabilities both to retrieve the songs with the highest probability of having a particular tag and to annotate each song with a subset of our vocabulary of tags. In a retrieval system, we return the songs in descending order of s_{jw}. To do automatic tagging, we could annotate each song with the M most likely tags for that song. However, this may lead to our annotating many songs with common, uninformative tags such as "Not Bizarre/Weird" and a lack of diversity in our annotations. To compensate for this, we use a simple heuristic: we introduce a "diversity factor" d and discount each s_{jw} by d times the mean of the estimated probabilities $s_{\cdot w}$. A higher value of d will make less common tags more likely to appear in annotations, which may lead to less accurate but more informative annotations. The diversity factor d has no impact on retrieval.

The cost of computing each s_{jw} using equation 11 is linear in the number of codewords K, and the cost of vector quantizing new songs' feature data using the previously computed centroids obtained using k-means is linear in the number of features, the number of codewords K, and the length of the song. For practical values of K, the total cost of estimating the probability that a tag applies to a song is comparable to the cost of feature extraction. Our approach can therefore tag new songs efficiently, an important feature for large commercial music databases.

4. EVALUATION

We evaluated our model's performance on an annotation task and a retrieval task using the CAL500 data set. We compare our results on these tasks with two other sets of published results for these tasks on this corpus: those obtained by Turnbull et al. using mixture hierarchies estimation to learn the parameters to a set of mixture-of-Gaussians models [2], and those obtained by Bertin-Mahieux et al. using a discriminative approach based on the AdaBoost algorithm [9]. In the 2008 MIREX audio tag classification task, the approach in [2] was ranked either first or second according to all metrics measuring annotation or retrieval performance [11].

4.1 Annotation task

To evaluate our model's ability to automatically tag unlabeled songs, we measured its average per-word precision and recall on held-out data using tenfold cross-validation.

First, we vector quantized our data using k-means. We tested VQ codebook sizes from $K = 5$ to $K = 2500$. After finding a set of K centroids using k-means on a randomly chosen subset of 125,000 of the Delta-MFCC vectors (250 feature vectors per song), we labeled each Delta-MFCC vector in each song with the index of the cluster centroid whose squared Euclidean distance to that vector was smallest. Each song j was then represented as a K-dimensional vector $\boldsymbol{n}_j$, with n_{jk} giving the number of times label k appeared in song j, as described in equation 1.

We ran a tenfold cross-validation experiment modeled after the experiments in [2]. We split our data into 10 disjoint 50-song test sets at random, and for each test set

1. We iterated the EM algorithm described in section 3.3 on the remaining 450 songs to estimate the parameters $\boldsymbol{\beta}$. We stopped iterating once the negative log-likelihood of the training labels conditioned on $\boldsymbol{\beta}$ and $\boldsymbol{n}$ decreased by less than 0.1% per iteration.

2. Using equation 11, for each tag w and each song j in the test set we estimated $p(y_{jw}|\boldsymbol{n}_j, \boldsymbol{\beta})$, the probability of song j being characterized by tag w conditioned on $\boldsymbol{\beta}$ and the vector quantized feature data $\boldsymbol{n}_j$.

3. We subtracted $d = 1.25$ times the average conditional probability of tag w from our estimate of $p(y_{jw}|\boldsymbol{n}_j, \boldsymbol{\beta})$ for each song j to get a score s_{jw} for each song.

4. We annotated each song j with the ten tags with the highest scores s_{jw}.

Model	Precision	Recall	F-Score	AP	AROC
UpperBnd	0.712 (0.007)	0.375 (0.006)	0.491	1	1
Random	0.144 (0.004)	0.064 (0.002)	0.089	0.231 (0.004)	0.503 (0.004)
MixHier	0.265 (0.007)	0.158 (0.006)	0.198	0.390 (0.004)	0.710 (0.004)
Autotag (MFCC)	0.281	0.131	0.179	0.305	0.678
Autotag (afeats exp.)	**0.312**	0.153	0.205	0.385	0.674
CBA $K = 5$	0.198 (0.006)	0.107 (0.005)	0.139	0.328 (0.009)	0.707 (0.007)
CBA $K = 10$	0.214 (0.006)	0.111 (0.006)	0.146	0.336 (0.007)	0.715 (0.007)
CBA $K = 25$	0.247 (0.007)	0.134 (0.007)	0.174	0.352 (0.008)	0.734 (0.008)
CBA $K = 50$	0.257 (0.009)	0.145 (0.007)	0.185	0.366 (0.009)	0.746 (0.008)
CBA $K = 100$	0.263 (0.007)	0.149 (0.004)	0.190	0.372 (0.007)	0.748 (0.008)
CBA $K = 250$	0.279 (0.007)	0.153 (0.005)	0.198	0.385 (0.007)	0.760 (0.007)
CBA $K = 500$	0.286 (0.005)	**0.162 (0.004)**	**0.207**	0.390 (0.008)	0.759 (0.007)
CBA $K = 1000$	0.283 (0.008)	0.161 (0.006)	0.205	0.393 (0.008)	0.764 (0.006)
CBA $K = 2500$	0.282 (0.006)	**0.162 (0.004)**	0.206	**0.394 (0.008)**	**0.765 (0.007)**

Table 1. Summary of the performance of CBA (with a variety of VQ codebook sizes K), a mixture-of-Gaussians model (MixHier), and an AdaBoost-based model (Autotag) on an annotation task (evaluated using precision, recall, and F-score) and a retrieval task (evaluated using average precision (AP) and area under the receiver-operator curve (AROC)). Autotag (MFCC) used the same Delta-MFCC feature vectors and training set size of 450 songs as CBA and MixHier. Autotag (afeats exp.) used a larger set of features and a larger set of training songs. UpperBnd uses the optimal labeling for each evaluation metric, and shows the upper limit on what any system can achieve. Random is a baseline that annotates and ranks songs randomly.

To evaluate our system's annotation performance, we computed the average per-word precision, recall, and F-score. Per-word recall is defined as the average fraction of songs actually labeled w that our model annotates with label w. Per-word precision is defined as the average fraction of songs that our model annotates with label w that are actually labeled w. F-score is the harmonic mean of precision and recall, and is one metric of overall annotation performance.

Following [2], when our model does not annotate any songs with a label w we set the precision for that word to be the empirical probability that a word in the dataset is labeled w. This is the expected per-word precision for w if we annotate all songs randomly. If no songs in a test set are labeled w, then per-word precision and recall for w are undefined, so we ignore these words in our evaluation.

4.2 Retrieval task

To evaluate our system's retrieval performance, for each tag w we ranked each song j in the test set by the probability our model estimated of tag w applying to song j. We evaluated the average precision (AP) and area under the receiver-operator curve (AROC) for each ranking. AP is defined as the average of the precisions at each possible level of recall, and AROC is defined as the area under a curve plotting the percentage of true positives returned against the percentage of false positives returned. As in the annotation task, if no songs in a test set are labeled w then AP and AROC are undefined for that label, and we exclude it from our evaluation for that fold of cross-validation.

4.3 Annotation and retrieval results

Table 1 and figure 2 compare our CBA model's average performance under the five metrics described above with other published results on the same dataset. MixHier is Turnbull et al.'s system based on a mixture-of-Gaussians model [2], Autotag (MFCC) is Bertin-Mahieux's AdaBoost-based system using the same Delta-MFCC feature vectors as our model, and Autotag (afeats exp.) is Bertin-Mahieux's system trained using additional features and training data [9]. Random is a random baseline that retrieves songs in a random order and annotates songs randomly based on tags' empirical frequencies. UpperBnd shows the best performance possible under each metric. Random and UpperBnd were computed by Turnbull et al., and give a sense of the possible range for each metric.

We tested our model using a variety of codebook sizes K from 5 to 2500. Cross-validation performance improves as the codebook size increases until $K = 500$, at which point it levels off. Our model's performance does not depend strongly on fine tuning K, at least within a range of $500 \leq K \leq 2500$.

When using a codebook size of at least 500, our CBA model does at least as well as MixHier and Autotag under every metric except precision. Autotag gets significantly higher precision than CBA when it uses additional training data and features, but not when it uses the same features and training set as CBA.

Tables 2 and 3 give examples of annotations and retrieval results given by our model during cross-validation.

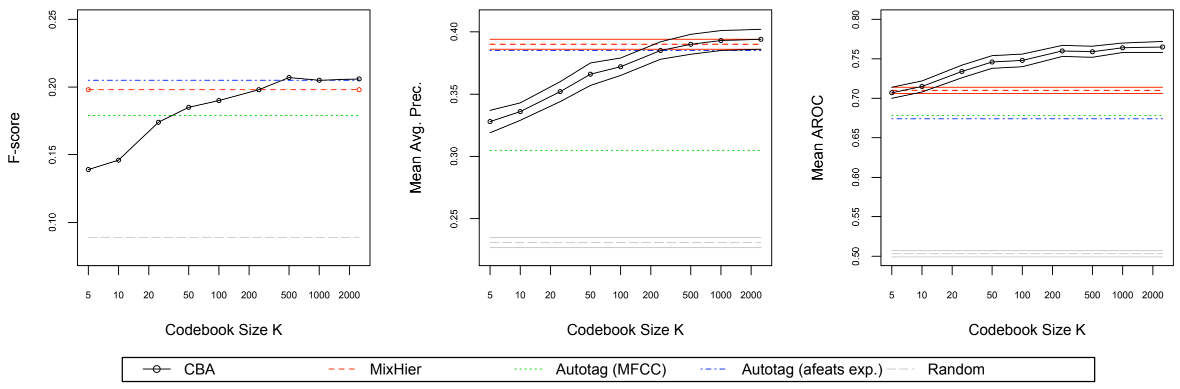


Figure 2. Visual comparison of the performance of several models evaluated using F-score, mean average precision, and area under receiver-operator curve (AROC).

4.4 Computational cost

We measured how long it took to estimate the parameters to CBA and to generalize to new songs. All experiments were conducted on one core of a server with a 2.2 GHz AMD Opteron 275 CPU and 16 GB of RAM running CentOS Linux.

Using a MATLAB implementation of the EM algorithm described in 3.3, it took 84.6 seconds to estimate CBA's parameters from 450 training songs vector-quantized using a 500-cluster codebook. In experiments with other codebook sizes K the training time scaled linearly with K. Once β had been estimated, it took less than a tenth of a millisecond to predict the probabilities of 174 labels for a new song.

We found that the vector-quantization process was the most expensive part of training and applying CBA. Finding a set of 500 cluster centroids from 125,000 39-dimensional Delta-MFCC vectors using a C++ implementation of k-means took 479 seconds, and finding the closest of 500 cluster centroids to the 10,000 feature vectors in a song took 0.454 seconds. Both of these figures scaled linearly with the size of the VQ codebook in other experiments.

5. DISCUSSION AND FUTURE WORK

We introduced the Codeword Bernoulli Average model, which predicts the probability that a tag will apply to a song based on counts of vector-quantized feature data extracted from that song. Our model is simple to implement, fast to train, generalizes to new songs efficiently, and yields state-of-the-art performance on annotation and semantic retrieval tasks.

We plan to explore several extensions to this model in the future. In place of the somewhat ad hoc diversity factor, one could use a weighting similar to TF-IDF to choose informative words for annotations. The vector quantization preprocessing stage could be replaced with a mixed-membership mixture-of-Gaussians model that could be fit simultaneously with β. Also, we hope to explore principled ways of incorporating song-level feature data describing information not captured by MFCCs, such as rhythm.

Query	Top 5 Retrieved Songs
Tender/Soft	John Lennon—Imagine Shira Kammen—Music of Waters Crosby Stills and Nash—Guinnevere Jewel—Enter From the East Yakshi—Chandra
Hip Hop	Tim Rayborn—Yedi Tekrar Solace—Laz 7 8 Eminem—My Fault Sir Mix-a-Lot—Baby Got Back 2-Pac—Trapped
Piano	Robert Johnson—Sweet Home Chicago Shira Kammen—Music of Waters Miles Davis—Blue in Green Guns n' Roses—November Rain Charlie Parker—Ornithology
Exercising	Tim Rayborn—Yedi Tekrar Monoide—Golden Key Introspekt—TBD Belief Systems—Skunk Werks Solace—Laz 7 8
Screaming	Nova Express—I'm Alive Rocket City Riot—Mine Tonite Seismic Anamoly—Wreckinball Pizzle—What's Wrong With My Footman Jackalopes—Rotgut

Table 3. Examples of semantic retrieval from the CAL500 data set. The left column shows a query word, and the right column shows the five songs in the dataset judged by our system to best match that word.

6. REFERENCES

[1] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the CAL500 data set. In *Proc. ACM SIGIR*, pages 439–446, 2007.

[2] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and

Give it Away the Red Hot Chili Peppers	*Fly Me to the Moon* Frank Sinatra	*Blue Monday* New Order	*Becoming* Pantera
Usage—At a party	Calming/Soothing	Very Danceable	NOT—Calming/Soothing
Heavy Beat	NOT—Fast Tempo	Usage—At a party	NOT—Tender/Soft
Drum Machine	NOT—High Energy	Heavy Beat	NOT—Laid-back/Mellow
Rapping	Laid-back/Mellow	Arousing/Awakening	Bass
Very Danceable	Tender/Soft	Fast Tempo	Genre—Alternative
Genre—Hip Hop/Rap	NOT—Arousing/Awakening	Drum Machine	Exciting/Thrilling
Genre (Best)—Hip Hop/Rap	Usage—Going to sleep	Texture Synthesized	Electric Guitar (distorted)
Texture Synthesized	Usage—Romancing	Sequencer	Genre—Rock
Arousing/Awakening	NOT—Powerful/Strong	Genre—Hip Hop/Rap	Texture Electric
Exciting/Thrilling	Sad	Synthesizer	High Energy

Table 2. Examples of semantic annotation from the CAL500 data set. The two columns show the top 10 words associated by our model with the songs *Give it Away, Fly Me to the Moon, Blue Monday,* and *Becoming*.

sound effects. *IEEE Transactions on Audio Speech and Language Processing*, 16(2), 2008.

[3] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, 1966.

[4] C. Wang, D. Blei, and L. Fei-Fei. Simultaneous image classification and annotation. In *Proc. IEEE CVPR*, 2009.

[5] M. Hoffman, D. Blei, and P. Cook. Content-based musical similarity computation using the hierarchical Dirichlet process. In *Proc. International Conference on Music Information Retrieval*, 2008.

[6] K. Seyerlehner, A. Linz, G. Widmer, and P. Knees. Frame level audio similarity—a codebook approach. In *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx08), Espoo, Finland, September*, 2008.

[7] M. Mandel and D. Ellis. LabROSA's audio classification submissions, mirex 2008 website. http://www.music-ir.org/mirex/2008/abs/AA_AG_AT_MM_CC_mandel.pdf.

[8] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[9] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: a model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):115–135, 2008.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[11] Mirex 2008 website. http://www.music-ir.org/mirex/2008/index.php/Main_Page.

USING ARTIST SIMILARITY TO PROPAGATE SEMANTIC INFORMATION

Joon Hee Kim, Brian Tomasik, Douglas Turnbull
Department of Computer Science,
Swarthmore College
{joonhee.kim@alum, btomasi1@alum, turnbull@cs}.swarthmore.edu

ABSTRACT

Tags are useful text-based labels that encode semantic information about music (instrumentation, genres, emotions, geographic origins). While there are a number of ways to collect and generate tags, there is generally a *data sparsity* problem in which very few songs and artists have been accurately annotated with a sufficiently large set of relevant tags. We explore the idea of *tag propagation* to help alleviate the data sparsity problem. Tag propagation, originally proposed by Sordo et al., involves annotating a novel artist with tags that have been frequently associated with other similar artists. In this paper, we explore four approaches for computing artists similarity based on different sources of music information (user preference data, social tags, web documents, and audio content). We compare these approaches in terms of their ability to accurately propagate three different types of tags (genres, acoustic descriptors, social tags). We find that the approach based on collaborative filtering performs best. This is somewhat surprising considering that it is the only approach that is not explicitly based on notions of semantic similarity. We also find that tag propagation based on content-based music analysis results in relatively poor performance.

1. INTRODUCTION

Tags, such as "hair metal", "afro-cuban influences", and "grrl power", are semantic labels that are useful for semantic music information retrieval (IR). That is, once we annotate (i.e., index) each artist (or song) in our music database with a sufficiently large set of tags, we can then retrieve (i.e., rank-order) the artists based on relevance to a text-based query.

The main problem with tag-based music IR is *data sparsity* (sometimes referred to as the *cold start* problem [1]). That is, in an ideal world, we would know the relevance (or lack thereof) between every artist and every tag. However, given that there are millions of songs and potentially thousands of useful tags, this is an enormous annotation problem. For example, Lamere [2] points out that Last.fm, a popular music-oriented social network, has a database containing over 150 millions songs each of which have been tagged with an average of 0.26 tags. This problem is made worse by *popularity bias* in which popular songs and artists tend to be annotated with a heavily disproportionate number of tags. This is illustrated by the fact that Lamere found only 7.5% of artists in his corpus of 280,000 artists had been annotated with one or more tags.

One potential solution to the *data sparsity* problem is to *propagate* tags between artists based on artist similarity. To annotate tags for an artist a, we find the most similar artists to a (referred to as neighbors) and transfer the most frequently occurring tags among the neighbors to artist a. Note that while we focus on artist annotation in this paper, our approach is general in that it could also be use to propagate tags between songs as well as other non-music related items such as movies and books.

Tag propagation has two potential uses. First, it allows us to index an unannotated artist if we can calculate the similarity between the artist and other annotated artists. Second, tag propagation allows us to augment and/or improve an existing annotation for an artist.

This idea was originally proposed by Sordo et al. who explore tag propagation of social tags based on acoustic similarity [3]. This content-based approach is compelling because we can automatically calculate artist similarity without relying on human input. However, as we will show in Section 5, the content-based tag propagation performs poorly relative to other music information sources.

In this paper, we extend their initial exploration by comparing alternative approaches to compute similarity: collaborative filtering of user preference data, similarity based on social tags, text-mining of web documents, and content-based analysis of music signals. In addition, we experiment with tag propagation on three different types of tags: acoustic descriptors, genres, and social tags.

While our focus is on the use of tag propagation for text-based music IR, we can also view our system as a way to evaluate artist similarity metric. That is, the approach that results in the best transfer of semantic information between artists may be considered a good approach for accessing artist similarity. Since artist similarity is often used for music recommendation, evaluating tag propagation performance is an automatic alternative to using labor-intensive human surveys when determining the quality of a music recommendation system.

1.1 Related Work

The importance of annotating music with tags is underscored by large investments that have been made by various companies in recent years. Companies like Pandora and AMG Allmusic employ dozens of professional music editors to manually annotate music with a small and structured vocabulary of tags. While this approach tends to produce accurate and complete characterizations of some songs, this labor-intensive approach does not scale with the rapidly increasing amount of available music online. For example, 50 Pandora experts annotate about 15,000 songs per month and would take over 83 years to annotate the 15 million songs that are currently in the AMG Allmusic database [1]

Last.fm and MyStrands use an alternative "crowdsourcing" approach in which millions of registered users are encouraged to label songs with any open-end free-text tags. As of September 2008, Last.fm had collected over 25 million song-tag annotations and 20 million artist-tag annotations using a vocabulary of 1.2 million unique tags (although only about 11% had been used more than 10 times) [4]. Each month, about 300 thousand unique users contribute more than 2.5 million new song-tag or artist-tag annotations. However, as mention above, a relatively small percentage of artists and songs have ever been tagged and even fewer have been thoroughly annotated.

Academic research has also focused on the music annotation problem in recent years. Turnbull et al. suggest that there are five general distinct approaches to annotating music with tags: conducting a survey (e.g., Pandora), harvesting social tags (e.g., Last.fm), playing annotation games [5, 6], text-mining web documents [7, 8], and analyzing audio content with signal processing and machine learning [9–11]. In some sense, tag propagation represents a sixth approach because it is based on the notions of artist similarity. That is, propagation can incorporate other forms of music information, such as user preference data, to generate tags for music. However, it cannot be used in isolation from these other approaches because it makes direct use of an initial set of annotated artists.

In the next section, we present the general tag propagation algorithm. We then introduce four different music information sources that are individually useful for calculating artist similarity. Section 4 describes the two evaluation metrics that we use to test our system with a database of 3,500 artists, four similarity metrics, and three types of tags. We discuss the results in Section 5, and conclude in Section 6.

2. TAG PROPAGATION

Compared with other automatic tagging algorithm, tag propagation is relatively straightforward. Suppose that we want to annotate a novel artist a. We find the most similar artists of a, combine existing annotations of them, and select the tags that appear frequently.

More formally, tag propagation requires two matrices: a *similarity* matrix $\mathbf{S}$ and a *tag* matrix $\mathbf{T}$. $\mathbf{S}$ is an artist-by-artist similarity matrix where $[\mathbf{S}]_{i,j}$ indicates similarity score between artist i and j. $\mathbf{T}$ is an artist-by-tag matrix where $[\mathbf{T}]_{a,t}$ represents the strength of association between artist a and tag t. In this paper, we consider the entries in $\mathbf{T}$ to be a binary number of 0 or 1, where 0 represents unknown or weak association, and 1 indicates a strong association. We call the a-th row of $\mathbf{T}$ the *tag annotation vector*, and denote as $\mathbf{t}_a$.

Once we have a similarity matrix $\mathbf{S}$ (as described in Section 3), we can use the standard k-Nearest Neighbor (kNN) algorithm to propagate tags. For the artist a in question, we find the k most similar artists (i.e., the neighbors), which we denote as $\mathcal{N}_a$. The neighbors are the columns corresponding to the k largest values in the a-th row of $\mathbf{S}$. We average the annotation vectors from $\mathbf{T}$ of $\mathcal{N}_a$ to estimate the annotation vector $\hat{\mathbf{t}}_a$ of a.

$$\hat{\mathbf{t}}_a = \frac{\sum_{i \in \mathcal{N}_a} \mathbf{t}_i}{k} \tag{1}$$

Based on an exponential grid search with $k \in \{2^i | 0 \leq i \leq 6\}$, we find that k between 8 and 64 results in comparable performance for each of our approaches. As such, we set $k = 32$ for each of our experiments in Section 5.

3. ARTIST SIMILARITY

In this section, we describe ways in which we can calculate artist similarity matrices from four different sources of music information. [2] In that our goal is to evaluate tag propagation, we primarily make use of existing music IR approaches [12–15].

3.1 Collaborative Filtering (CF)

Collaborative filtering (CF) is a popular commercial technique for calculating artist similarity [16] that is based on user preference data. The idea is that two artists are considered similar if there is a large number of users that listen to both artists. In this paper, we consider two forms of user preference data: *explicit* feedback and *implicit* feedback. Feedback is explicit if a user has indicated directly that he or she "likes" an artist. This information is often recorded by a user through a button on a music player interface. Implicit feedback is found by tracking user listening habits. For example, Last.fm monitors which songs each of their users listens to over a long period of time. Implicit feedback assumes that two artists are similar if many users listen to songs by both artists.

We aggregate user preference data from 400,000 Last.fm users, and build an artist similarity matrix, *CF-Explicit*, by counting the number of users who have explicitly indicated that they like both artists. We construct

[1] Pandora statistics are based on personal notes for a public talk by Pandora founder Tim Westergren. AMG statistics were found at http://www.allmusic.com.

[2] The data that we describe in this paper was collected from the Internet in April of 2008.

Table 1. Most similar pairs of artists based on CF (explicit) and their top social tags.

Tex Ritter	country	classic country	country roots	oldies	old timey
Red Foley	country	classic country	boogie	rock	american
Unwound	noise rock	post-hardcore	indie rock	math rock	post-rock
Young Widows	noise rock	post-hardcore	math rock	experimental	heavy
DLG	salsa	latin	dlg	bachata	spanish
Puerto Rican Power	salsa	latin	mambo	latino	cuba
Starkillers	dance	house	trance	electro house	electronica
Kid Dub	electro	electro house	electronic	dub	electro-house
Lynda Randle	gospel	female vocalists	christian	southern gospel	female vocalist
George Jones	country	classic country	americana	singer-songwriter	traditional country
An Albatross	experimental	grindcore	noisecore	hardcore	noise
See You Next Tuesday	grindcore	deathcore	mathcore	experimental	noisecore

a second similarity matrix, *CF-Implicit*, by counting the number of users who listen to both artists at least 1% of the time.

One issue that arises when using the raw co-occurrence counts is that the popular artists tend to occur frequently as a "most similarity" artist [16]. A standard solution is to *normalize by the popularity* of each artists:

$$[\mathbf{S}]_{i,j} = \frac{co(i,j)}{\sqrt{\sum_{k\in\mathcal{A}} co(i,k)}\sqrt{\sum_{k\in\mathcal{A}} co(k,j)}} \quad (2)$$

where $\mathcal{A}$ is the set of 3,500 artists, $co(i,j)$ is the number of users that have given feedback for both artist i and artist j (explicit or implicit depending on the matrix type). Note that this equation is equivalent to the cosine distance between two column vectors of a User-by-Item rating matrix if we assume that users give binary rating [16].

It could be the case that similarity based on CF is not strongly related to semantic similarity, and thus might not be useful for tag propagation. However, if we look at a couple of examples (see Table 1), we find that similar artists share a number of common tags. This is confirmed in Section 5.1, when we quantitatively compare the performance of tag propagation using *CF-Explicit* and *CF-Implicit*. We also report on the effect of popularity normalization for these two approaches.

3.2 Social Tags (ST)

As described in Section 1.1, social tags (ST) are socially generated semantic information about music. Lamere and Celma [13] show that computing artist similarity using social tags produces better performance for music recommendation than other approaches such as collaborative filtering, content-based analysis, or human expert recommendations.

Following their approach, we collect a set of social tags (represented as a tag annotation vector t_a) for each artist a from Last.fm. However, when collecting this data set, we found a total of about 30,000 unique tags for our 3,500 artists from Last.fm. Since Last.fm allows anyone to apply any tag, this vocabulary of tags contains many rare tags that seemed to be (inconsistently) applied to a small number of artists [1]. In an attempt to clean up the data, we choose to prune tags that are associated with less than .5% of the artists. This resulted in vocabulary of 949 unique tags.

The ST artist similarity matrix $\mathbf{S}$ is built by calculating cosine similarity between each annotation vector:

$$[\mathbf{S}]_{i,j} = \frac{\mathbf{t}_i \cdot \mathbf{t}_j}{|\mathbf{t}_i||\mathbf{t}_j|} \quad (3)$$

where each annotation vector $\mathbf{t}$ is a vector over 949 dimension.

3.3 Web Documents (WD)

Web documents represent a third source of music information that can be used to calculate music similarity. For each artist a, we collect 50 documents from the Google Search Engine[3] with query `''artist name'' music`. We combine the top 50 results into a single document and then represent that document as a bag-of-words. This bag-of-words is converted into the term-frequency-inverse-document-frequency (TF-IDF) *document* vector $\mathbf{d}_a$ over a large vocabulary of words [17]. TF-IDF is a standard text-IR representation that places more emphasis on words that appear frequently in the given document and are less common in the entire set of documents.

We build the WD artist similarity matrix $\mathbf{S}$ by calculating cosine similarity score on each pair of TF-IDF document:

$$[\mathbf{S}]_{i,j} = \frac{\mathbf{d}_i \cdot \mathbf{d}_j}{|\mathbf{d}_i||\mathbf{d}_j|} \quad (4)$$

where i, j are artists.

3.4 Content-Based Analysis (CB)

Lastly, we explore two content-based (CB) approaches for calculating artist similarity that have performed well in various MIREX tasks [12,15,18] in recent years. For both approaches, we begin by extracting a bag of Mel-Frequency

[3] www.google.com

Cepstral Coefficients (MFCCs) feature vectors from one randomly selected song by each artist.

Our first approach, which was proposed by Mandel and Ellis [12] (referred to as *CB-Acoustic*), models the bag-of-MFCCs with a single Gaussian distribution over the MFCC feature space. To calculate the similarity between two artists, we calculate the symmetric KL divergence between the two Gaussian distributions for the songs by the two artists. For this approach, we use the first 20 MFCCs and estimate the Gaussian distribution using a full covariance matrix. This approach is chosen because it is fast, easy to compute, and a popular baseline within the music-IR community.

The second approach, proposed by Barrington et al. [15] (referred to as *CB-Semantic*), involves estimating the KL-divergence between the two *Semantic Multinomial* distributions corresponding to the selected songs for each pair of artists. A semantic multinomial is a (normalized) vector of probabilities over a vocabulary of tags. To calculate the semantic multinomial, we first learn one Gaussian Mixture Model (GMM) over the MFCC feature space for each tag in our vocabulary. The GMMs are estimated using training data (e.g., songs that are known to be associated with each tag) in a supervised learning framework. We then take a novel song and calculate its likelihood under each of the GMMs to produce a vector of unnormalized probabilities. When normalized, this vector can be interpreted as a multinomal distribution over a semantic space of tags. We choose a vocabulary of 512 genres and acoustic tags and use 39-dimensional MFCC+Delta feature vectors. MFCC+Delta vectors include the first 13 MFCCs plus each of their 1st and 2nd instantaneous derivatives. This approach is chosen because it is based on a top performing approach in the 2007 MIREX audio similarity task and is based on a top performing approach in the 2008 MIREX audio tag classification task.

4. EXPERIMENTAL SETUP

4.1 Data

Our data set consists of 3,500 artists with music that spans 19 top-level genres (e.g., Rock, Classical, Electronic) and 123 subgenre (e.g., Grunge, Romantic Period Opera, Trance). Each artist is associated with 1 or more genre and 1 or more subgenres. The set of 142 genres and subgenres make up our initial *Genre* vocabulary.

For each artist, we collect a set of acoustic tags for songs by the artist from Pandora's Music Genome Project. This *Acoustic* tag vocabulary consists of 891 unique tags like "dominant bass riff", "gravelly male vocalist", and "acoustic sonority". In general, these acoustic tags are thought to be *objective* in that two trained experts can annotate a song using the same tags with high probability [19]. Lastly, we collect social tags for each artist using the Last.fm public API as discussed in Section 3.2. After pruning, the *Social* tag vocabulary, it consists of 949 unique tags.

In all three cases, we construct a binary ground truth tag matrix $\mathbf{T}$ where $[\mathbf{T}]_{s,a} = 1$ if the tag is present for the artists (or in one of the songs by the artists), and 0 otherwise.

4.2 Evaluation Metrics

We use leave-one-out cross-validation to test our system. For each artist a, we hold out the ground truth tag annotation vector $\mathbf{t}_a$ and calculate the estimated vector $\hat{\mathbf{t}}_a$ by kNN algorithm. In the *artist annotation* test, we test how well we can propagate relevant tags to a novel artist by comparing the estimated vector with the ground truth.

In the *tag-based retrieval* test, we generate a ranked list of the artists for each tag based on their association strength to a tag. Then we evaluate how high the relevant artists are placed on the ranked list. Each test is described in detail below.

One of our artist similarity metric is based on the similarity of socially generated tags as discussed in Section 3.2. We use tags generated by Last.fm users as our data source because it provides the largest data set of social tags. Unfortunately, we evaluate our system on the same data as well. Therefore, we use 10-fold cross-validation to evaluate the propagation of $social$ tags based on the similarity of $social$ tags. That is, for each of 10 folds, we use 90% of the tags to estimate a song similarity matrix. This similarity matrix is used to propagate the other 10% of the tags. We can combine the 10 estimated annotation tag vectors from each of the 10 folds into one complete annotation vector.

4.2.1 Artist Annotation

For each artist a, we evaluate the relevance of the estimated annotation vector $\hat{\mathbf{t}}_a$ by comparing it to the ground truth $\mathbf{t}_a$. As described earlier, the ground truth data is in binary format. We transform the estimated annotation vector into the same binary vector by setting each value that is above a threshold to 1, and zero otherwise.

By doing so, we move from the estimation problem to the standard retrieval problem [17]. That is, we predict a set of relevant tags to describe the artist. We can then calculate precision, recall and f-measure for the given threshold. By varying threshold, we compute a precision-recall curve as shown in Figure **??**.

4.2.2 Tag-Based Retrieval

In this experiment, we evaluate the performance of tag-based retrieval of relevant artists. For each tag, we generate a ranked-list of 3,500 artists. The rank is based on the association score of the tag in each artist's estimated annotation vector. Using the ground truth annotations, we calculate R-precision, 10-Precision, MAP (mean average precision) and AUC (area under the ROC curve) for each tag [17]. We then average the performance of the tags in each of our three tag vocabularies: Pandora Genre, Pandora Acoustic, and Last.fm Social.

Table 2. Exploring variants of collaborative filtering (CF): We report the average f-measure / area under the ROC curve (AUC) for explicit or implicit user preference information when we have either normalized or not normalized for popularity. Each evaluation metric is the average value over the three tag vocabularies.

	Unnormalized	Normalized
Explicit	.438 / .867	.495 / .885
Implicit	.410 / .824	**.502 / .891**

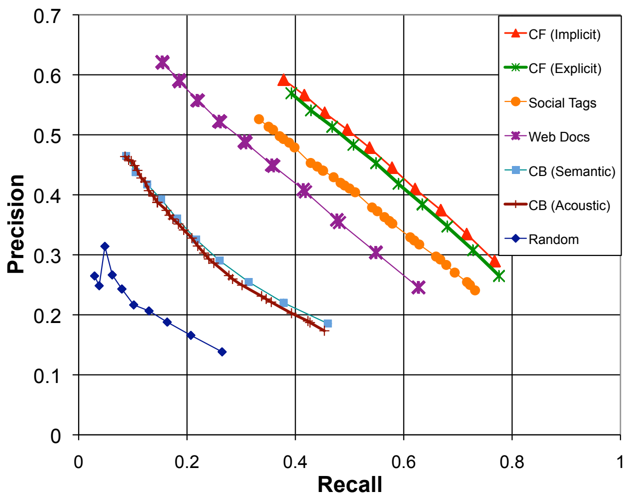


Figure 1. Semantic annotation and retrieval model diagram.

5. RESULTS

5.1 CF Comparison

The collaborative filtering approach has four variants with two sets of varying conditions. First, we compare using explicit and the implicit user preference data. Second, the similarity matrix **S** was generated with and without the popularity-normalization. We evaluate the performance of each variant by comparing f-measure from the *artist annotation* test and area under the ROC curve (AUC) from the *tag-based retrieval* test.

The result of each test is illustrated in Table 2. In our experiments, we observe no significant difference between the explicit and the implicit user preference data. However, in both cases, the normalization improves the performance. It is interesting that the normalization boosts the performance of the implicit data more significantly than the explicit data. This could be due to the fact that implicit data may be more prone to the popularity bias since Last.fm radio playlists tend to recommend music from *popular* artists [16].

5.2 Artist Annotation

The precision-recall curves for artist annotation are plotted in Figure **??**. For each test, we varied the threshold from 0.1 to 0.4 with the interval of 0.01 and calculated precision, recall, and f-measure. The baseline *Random* performance is calculated by estimating each annotation vector with $k = 32$ distinct random neighbors. Except for the random baseline, the f-measure was maximized at around a threshold of 0.3.

In general, the two variants of the collaborative filtering (CF) approach perform best, with the *implicit* feedback approach performing slightly better. This is surprising because the collaborative filtering approach does not explicitly encode semantic information whereas social tag, web documents, and CB-Semantic are based on the similarity of semantic information. This suggests that collaborative filtering is useful for determining semantic similarity as well as music recommendation.

5.3 Tag-based Retrieval

We evaluate tag-based music retrieval based on tag propagation using seven approaches to computing music similarity. We report the performance for three vocabularies of tags ($Genre$, $Acoustic$, and $Social$) in Table 3.

As was the case with artist annotation, both CF-Implicit and CF-Explicit show strong performance for all four metrics and all three vocabularies. However, ST has the best performance for R-Precision, 10-Precision, and MAP when propagating social tags.

Since area under the ROC curve (AUC) is an evaluation metric that is not biased by the prior probability of relevant artists for a given tag, we can safely compare average AUC values across the different tag vocabularies. Based on this metric, we see that all of the approaches (except for the CB-Acoustic) have higher AUC values in the order of $Genre$, $Acoustic$, and $Social$ tag sets. This suggest that it may be easiest to propagate genres and hardest to propagate social tags to novel artists.

Both CB approaches show relatively poor performance (though much better than random), which is disappointing since all of the other methods require additional human input to calculate music similarity for a novel artist. That is, if either CB approached showed better performance, we could remedy the data sparsity problem for novel artists with a fully automatic tag propagation approach.

6. CONCLUSION

In this paper, we have explored tag propagation as a technique for annotating artists with tags. We explored alternative ways to calculate artist similarity by taking advantage of the existing sources of music information such as user preference data (CF), social tags (ST), web documents (WD), and audio content (CB). Each similarity metric was tested on three distinct tag sets: $genre$, $acoustic$, and $social$. Both *artist annotation*, and *tag-based retrieval tests* show that CF generally performs the best, followed by ST, WD, and CB. This result is somewhat surprising because collaborative filtering (CF) is solely based on the aggregate trends of listening habits and user preferences, rather than explicitly representing music semantics. It confirms the idea that CF similarity (e.g., user behavior) can be

Table 3. Tag-based music retrieval performance. Each evaluation metric is averaged over all tags for each of the three vocabularies. R-precision for a tag is the precision (the ratio of correctly-labelled artists to the total number of retrieved artists) when R documents are retrieved, where R is the number of relevant artists in the ground-truth. Similarly, 10-precision for a tag is the precision when 10 artists are retrieved (e.g., the "search engine metric"). Mean average precision (MAP) is found by moving down the ranked list of artists and averaging the precisions at every point where we correctly identify a relevant artist based on the ground truth. The last metric is the area under the receiver operating characteristic (ROC) curve (denoted AUC). The ROC curve compares the rate of correct detections to false alarms at each point in the ranking. A perfect ranking (i.e., all the relevant songs at the top) results in an AUC equal to 1.0. We expect the AUC to be 0.5 if we randomly rank songs. More details on these standard IR metrics can be found in Chapter 8 of [17].

Approach	Genre (142 tags)				Acoustic (891 tags)				Social (949 tags)			
	r-prec	10-prec	MAP	AUC	r-prec	10-prec	MAP	AUC	r-prec	10-prec	MAP	AUC
Random	0.012	0.015	0.017	0.499	0.025	0.023	0.029	0.495	0.030	0.029	0.033	0.498
CF (implicit)	**0.362**	0.381	**0.342**	**0.914**	0.281	**0.306**	**0.254**	**0.882**	0.409	0.543	0.394	**0.876**
CF (explicit)	**0.362**	**0.388**	0.329	0.909	**0.282**	0.304	0.246	0.878	0.410	0.562	0.396	0.869
ST	0.344	0.349	0.311	0.889	0.267	0.274	0.237	0.874	**0.428**	**0.584**	**0.413**	0.874
WD	0.321	0.393	0.282	0.861	0.244	0.300	0.200	0.814	0.318	0.478	0.286	0.797
CB (acoustic)	0.101	0.127	0.076	0.701	0.118	0.132	0.088	0.692	0.117	0.159	0.092	0.661
CB (semantic)	0.087	0.103	0.069	0.687	0.115	0.123	0.091	0.714	0.107	0.126	0.084	0.662

used to capture the semantic similarity (e.g., tags) among artists. We also found that two content-based approaches (CB) performed poorly in our experiments. This is unfortunate because content-based similarity can be calculated for novel artists without human intervention, and thus would have solved the data sparsity problem.

7. REFERENCES

[1] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. *ISMIR*, 2008.

[2] P. Lamere. Social tagging and music information retrieval. *JNMR*, 2008.

[3] M. Sordo, C. Lauier, and O. Celma. Annotating music collections: How content-based similarity helps to propagate labels. In *ISMIR*, 2007.

[4] P. Lamere and E. Pampalk. Social tags and music information retrieval. ISMIR Tutorial, 2008.

[5] D. Turnbull, R. Liu, L. Barrington, D. Torres, and G Lanckriet. Using games to collect semantic information about music. In *ISMIR '07*, 2007.

[6] E. Law and L. von Ahn. Input-agreement: A new mechanism for data collection using human computation games. *ACM CHI*, 2009.

[7] B. Whitman and D. Ellis. Automatic record reviews. *ISMIR*, 2004.

[8] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A music search engine built upon audio-based and web-based similarity measures. In *ACM SIGIR*, 2007.

[9] M. Mandel and D. Ellis. Multiple-instance learning for music information retrieval. In *ISMIR*, 2008.

[10] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, February 2008.

[11] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Neural Information Processing Systems Conference (NIPS)*, 2007.

[12] M.I. Mandel and D.P.W. Ellis. Song-level features and support vector machines for music classification. *ISMIR*, 2005.

[13] P. Lamere and O. Celma. Music recommendation tutorial notes. ISMIR Tutorial, September 2007.

[14] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A large-scale evalutation of acoustic and subjective music-similarity measures. *Computer Music Journal*, pages 63–76, 2004.

[15] L. Barrington, A. Chan, D. Turnbull, and G. Lanckriet. Audio information retrieval using semantic similarity. *ICASSP*, 2007.

[16] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[17] C.D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] S. J. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 2008.

[19] T. Westergren. Personal notes from Pandora get-together in San Diego, March 2007.

MUSIC MOOD REPRESENTATIONS FROM SOCIAL TAGS

Cyril Laurier, Mohamed Sordo, Joan Serrà, Perfecto Herrera
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{cyril.laurier,mohamed.sordo,joan.serraj,perfecto.herrera}@upf.edu

ABSTRACT

This paper presents findings about mood representations. We aim to analyze how do people tag music by mood, to create representations based on this data and to study the agreement between experts and a large community. For this purpose, we create a semantic mood space from *last.fm* tags using Latent Semantic Analysis. With an unsupervised clustering approach, we derive from this space an ideal categorical representation. We compare our community based semantic space with expert representations from Hevner and the clusters from the MIREX Audio Mood Classification task. Using dimensional reduction with a Self-Organizing Map, we obtain a 2D representation that we compare with the dimensional model from Russell. We present as well a tree diagram of the mood tags obtained with a hierarchical clustering approach. All these results show a consistency between the community and the experts as well as some limitations of current expert models. This study demonstrates a particular relevancy of the basic emotions model with four mood clusters that can be summarized as: *happy*, *sad*, *angry* and *tender*. This outcome can help to create better ground truth and to provide more realistic mood classification algorithms. Furthermore, this method can be applied to other types of representations to build better computational models.

1. INTRODUCTION

Music classification by mood [1] recently emerged as a topic of interest in the Music Information Retrieval (MIR) community. The first task to tackle this problem is to find a relevant representation of mood. In this work, we study mood representations with a bottom-up approach, from a community point of view.

Several works have shown a potential to model mood in music (like [3–5] , see [6] for an extensive review). Although this task is quite complex, satisfying results can be achieved, especially if we concentrate on the mood expressed by the music rather than the mood induced [6]. However, almost every work differs in the way that it represents emotions. Similarly to psychological studies, there is no real agreement on a common model. Comparing these different techniques is a very arduous task. With the objective to evaluate several algorithms within the same framework, MIREX (Music Information Retrieval Evaluation eXchange) [7] organized a task on this topic for the first time in 2007. To do so, it was decided to frame the problem into a classification task with 5 mutually exclusive categories. However, it was shown that these clusters might not be optimal as we suspect some semantic overlap between categories [8]. In a nutshell, finding the right mood representation is complex.

In this study, we want to address this problem using data collected in an ”everyday life” context (not in controlled laboratory settings like in psychological studies). From this data, we want to create a semantic space for mood. In [10], the authors studied the agreement between experts and a community (also based on *last.fm* tags) for genre classification. Levy in [11], studied how tags can be used for genre and artist similarity and proposed a visualization of certain words in an emotion space. Both studies inspired our approach of using social tags to compare the semantics of the wisdom of crowds with expert knowledge.

The goal of this paper is to create a semantic mood space where we can represent mood and compare it with existing representations. There are two main motivations for this study. First we aim to verify if the knowledge extracted from social tags and the knowledge from the experts (psychologists) converges. Then, we want to generate mood representations that can serve as a basis for further works like music mood classification. In Section 2, we expose the expert mood representations. In Section 3, we detail the dataset of tags and then, in Section 4, its transformation into a semantic space. In Section 5, we study the categorical representations. In Sections 6 and 7, we generate and analyze dimensional and hierarchical representations. Finally, Section 8 concludes and summarizes the main findings.

2. EXPERT REPRESENTATIONS

Two main types of representation coexist in the literature. The first one is the categorical model, using for instance basic emotions with around four or five categories including: *happiness*, *sadness*, *fear*, *anger* and *tenderness* [1]. Some works propose mood clusters like the eight clusters from Hevner [9] (see Figure 1) or the five clusters used in the MIREX Audio Mood Classification task, detailed

[1] In order to simplify the terminology, we will use the words emotion and mood independently for the same meaning: a particular feeling characterizing a state of mind

Clusters	Mood Adjectives
Cluster 1	passionate, rousing, confident, boisterous, rowdy
Cluster 2	rollicking, cheerful, fun, sweet, amiable/good natured
Cluster 3	literate, poignant, wistful, bittersweet, autumnal, brooding
Cluster 4	humorous, silly, campy, quirky, whimsical, witty, wry
Cluster 5	aggressive, fiery, tense/anxious, intense, volatile, visceral

Table 1. Clusters of mood adjectives used in the MIREX Audio Mood Classification task.

in Table 1. The second type of representation is the dimensional model, based originally on Russell's circumplex model of affect [2] (see Figure 2). The two dimensions mostly used are arousal and valence [2] .

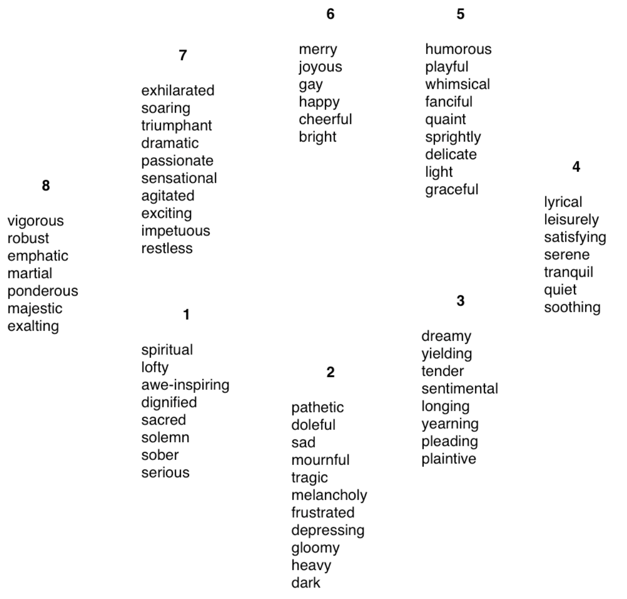


Figure 1. Hevner's [9] model with adjectives grouped into eight clusters.

3. DATASET

Our objective is to obtain a mood space based on social tags. In order to achieve this goal, we need two components: a list of mood words and social network data.

3.1 Mood list

For this study, we want to observe the way people use mood words in a social network. We selected words related to emotions based on the main articles in music and emotion research. We included words from different psychological studies like Hevner [9] or Russell [2]. We also added words representing basic emotions and other related adjectives [1]. Finally we aggregated the mood terms mostly used in MIR [6] and the ones selected for the MIREX task [8]. At the end of this process, we obtained a list of 120 mood words.

[2] In psychology, the term valence describes the attractiveness or aversiveness of an event, object or situation.

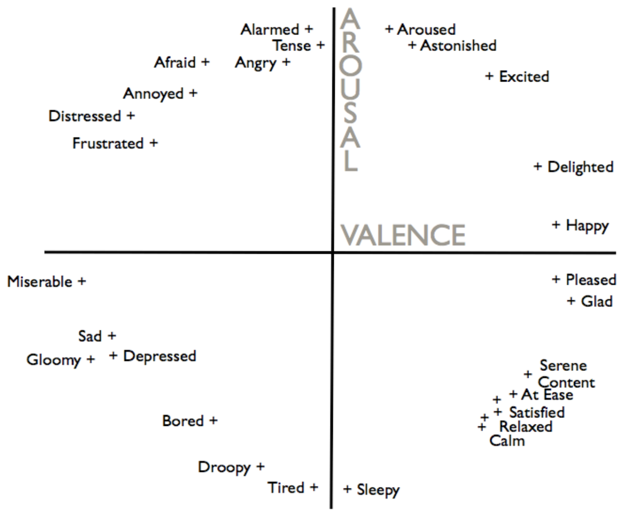


Figure 2. Russell's [2] circumplex model of affect with arousal and valence dimensions.

3.2 Social Tags

Last.fm [3] is a music recommendation website with a large community of users who are very active in associating tags with the music they listen to. With over 30 million users in more than 200 countries [4] , this social network is a good candidate to study how people tag their music. We crawled 6,814,068 tag annotations from 575,149 tracks in all main genres. From those, 492,634 tags were distinct. This huge dataset contains tags of any kind. From the original 120 mood words, 107 tags were found in our dataset. However some of them did not appear very often. We decided to keep only the tags that appeared at least 100 times, resulting in a list of 80 words. We also chose to keep the tracks were the same mood tag has been used by several users. This subset contains 61,080 tracks. We observe that the mood tags mostly used are *sad*, *fun*, *melancholy* and *happy*. For instance, the tag *sad* has been used 11,898 times in our dataset. On the contrary, the least used tags are *rollicking*, *solemn*, *rowdy* and *tense*, applied in less than 150 tracks. In average, a mood tag is used in 754 tracks.

4. SEMANTIC MOOD SPACE

We aim to compare mood terms by their co-occurences in tracks. Intuitively *happy* should co-occur more often with *fun* or *joy* than with *sad* or *depressed*. This co-occurence information included in the data we crawled from *last.fm* is embodied in a document-term matrix where the columns are track vectors representing tags.

The main problem we have when dealing with this matrix is its high dimensionality and its sparsity. Consequently, we applied a Latent Semantic Analysis (LSA) [12] to project the data into a space of a given lower dimensionality, while maintaining a good approximation of the distances between data points. This technique has been shown to be very efficient to capture tag representations for genre and artists

[3] http://www.last.fm
[4] http://blog.last.fm/2009/03/24/lastfm-radio-announcement

similarity [11]. LSA makes use of algebraic techniques such as Singular Values Decomposition (SVD) to reduce the dimensionality of the matrix. We decided to use a dimension of 100, which seems to be good trade-off for similarity tasks [11]. In the following experiments, we tried to change this dimension parameter (from 10 to 10 000 on a logarithmic scale), with no significant impact on the outcomes except less relevant results when selecting a too low or too high dimension. Once we have the data into this semantic space, we compute the distance between terms using the cosine distance. The distance values are included in the range $[0, 1]$. Here are some examples of distances between mood tags:

$d_{cos}(happy, sad) = 0.99$
$d_{cos}(cheerful, sleepy) = 0.97$
$d_{cos}(anger, aggressive) = 0.06$
$d_{cos}(calm, relaxed) = 0.03$

We observe that *happy* and *sad* are quite far from each other, as well as *cheerful* and *sleepy*. On the other hand, we note that *anger* is close to *aggressive* and that *calm* is similar to *relaxed*. Even if we show here some prototypical examples, values in the whole distance matrix intuitively make sense.

5. CATEGORICAL REPRESENTATIONS

To study the categorical mood representations, we first derive a folksonomy (community-based taxonomy) representation by means of unsupervised clustering from the social data. Then, we evaluate how the expert taxonomies fit into the semantic mood space.

5.1 Folksonomy representation

From our semantic space, we want to infer what would be the ideal categorical representation. To achieve this goal, we apply an unsupervised clustering method using the Expectation maximization (EM) algorithm. This algorithm and the implementation we used (WEKA) are described in [13]. The first important question to be answered is how many clusters should we consider. As we want this number to be inferred by the data itself, we used the *v-fold cross validation* algorithm. We divided the dataset in v folds, training on $v - 1$ folds and testing on the remaining one. We measure the log-likelihood computed for the observations in the testing samples. The results for the v replications are averaged to yield a single measure of the stability of the model. In Figure 3, we show the results of this process, displaying an average cost value (in our case 2 times the negative log-likelihood of the cross-validation data). Intuitively the lower is the value, the better is the cluster. To choose the ”right” number of clusters, we look at the cost value while increasing the number of clusters. Practically, we stop when the mean cost value stops decreasing and select the current number of clusters.

We observe that the cost rapidly decreases with the number of clusters until four clusters. After that, it is stable and even increases, meaning that the data is overfitted. Consequently, the optimal number of clusters is four. Using this number for the EM algorithm, we obtained the clusters exposed in Table 2.

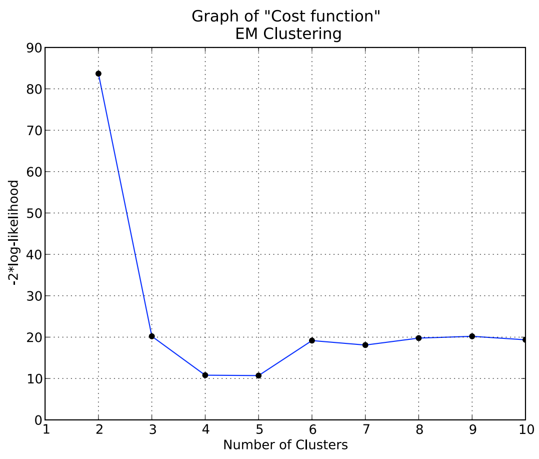


Figure 3. Plot of the cost values (2 times the negative log-likelihood) depending on the number of clusters.

Cluster 1	Cluster 2	Cluster 3	Cluster 4
angry	sad	tender	happy
aggressive	bittersweet	soothing	joyous
visceral	sentimental	sleepy	bright
rousing	tragic	tranquil	cheerful
intense	depressing	good natured	happiness
confident	sadness	quiet	humorous
anger	spooky	calm	gay
exciting	gloomy	serene	amiable
martial	sweet	relax	merry
tense	mysterious	dreamy	rollicking
anxious	mournful	delicate	campy
passionate	poignant	longing	light
quirky	lyrical	spiritual	silly
wry	miserable	wistful	boisterous
fiery	yearning	relaxed	fun

Table 2. Folksonomy representation. Clusters of mood tags obtained with the EM algorithm. For space and clarity reasons, we show only the first tags.

These four clusters are very similar to the categories posed by the main basic emotion theories [1]. Moreover, these clusters represents the four quadrants of the classical arousal-valence plane from Russell previously shown in Figure 2:

Cluster 1: angry (high arousal, low valence)
Cluster 2: sad, depressing (low valence, low arousal)
Cluster 3: tender, calm (high valence, low arousal)
Cluster 4: happy (high arousal, high valence)

To summarize, the semantic space we created is relevant and coherent with existing basic emotion approaches.

This result is very encouraging and assesses a certain quality of this semantic space. Moreover, it confirms that the community uses mood tags in a way that converges with the basic emotion theory from psychology.

5.2 Agreement between experts and community

In this section, we want to measure the agreement between experts and community representations. To do so, we performed a coarse-grained similarity, where we measured how *separable* the expert-defined mood clusters are in our semantic space. First, we computed the LSA cosine similarity among all moods within each cluster (intra-cluster similarity) and then we computed the dissimilarity among clusters, using the centroid of each cluster (inter-cluster dissimilarity). The expert representations we selected for this experiment are the eight clusters from Hevner (see Figure 1) where we could match more than 50% of the tags and the five clusters from the MIREX taxonomy (see Table 1) where all 31 tags were matched.

5.2.1 Intra-cluster similarity

For each cluster of the expert representations, we compute the mean cosine similarity between each mood tag in the cluster. The results for intra-cluster similarity are presented in Figure 4 for the Hevner representation and in Figure 5 for the MIREX clusters.

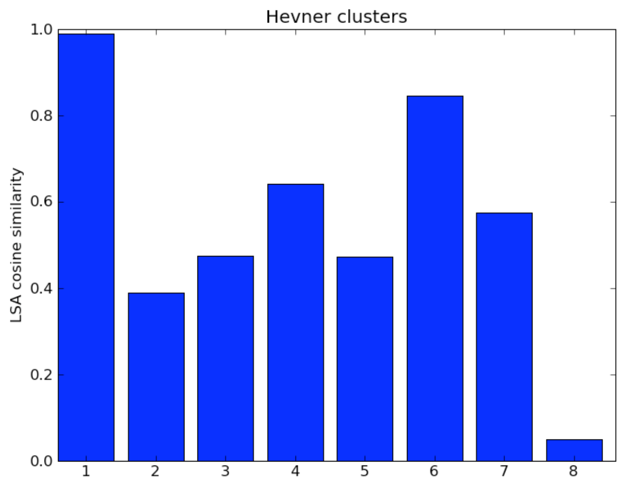


Figure 4. Intra-cluster cosine similarity for Hevner's representation.

In the results for the Hevner clusters, we note a high intra-cluster similarity value for cluster 1, which is the one including *spiritual* and *sacred* (please look at Figure 1 for the complete list). Cluster 6 performs also quite well (*joyous*, *bright*, *gay*, *cheerful*, *merry*). However we have poor intra-cluster similarity for cluster 8, which includes *vigorous*, *martial* and *majestic*. This might be because these words are also some of the less used in our dataset, but we hypothesize that they are less descriptive today than when the taxonomy was created (1936). Moreover, these words were selected for classical music which is not the main content of the *lasf.fm* music. The rest of the intra-cluster similarity values are in average quite low, meaning that this representation is not optimal in the semantic mood space.

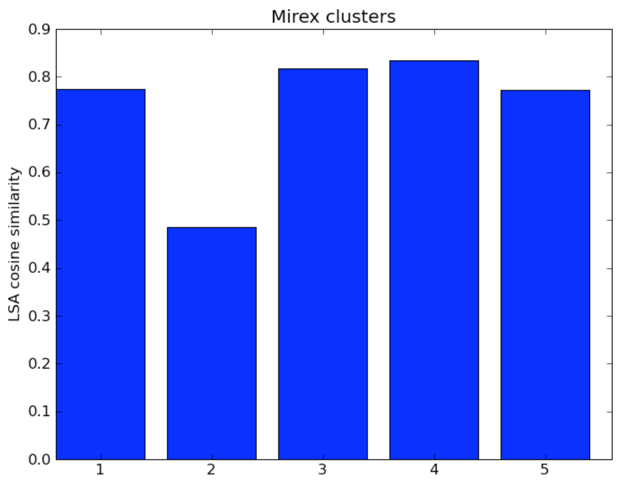


Figure 5. Intra-cluster cosine similarity for MIREX representation.

For the MIREX clusters, we remark that the lowest intra-cluster similarity is for cluster 2 (*sweet*, *good natured*, *cheerful*, *rollicking*, *amiable*, *fun*). Maybe is it quite clear that this category is about *happy* music, however the words used are not so common and may lower this value. In average, the intra-cluster similarity value is quite high for this representation. For comparison purpose, we note that the intra-cluster similarity of the folksonomy representation has an average intra-cluster similarity value of 0.82 (see Table 4). Obviously, as the folksonomy representation was made from the semantic space itself, it has better results than the other models.

In this part, we have looked at the consistency inside each cluster, however it is also crucial to look at the distances between clusters to evaluate the quality of the clustering representations.

5.2.2 Inter-cluster dissimilarity

To measure how *separable* are the different clusters, we compute the mean cosine distance from each cluster centroid to the other cluster centroids. If we look at our folksonomy representation clusters from Section 5.1, the cosine distance between centroids of clusters are all quite high (0.9 in average, see Table 4). This is not very suprising as the representation was designed with this data.

In Table 3, we show the confusion matrix of the inter-cluster dissimilarity for the MIREX clusters. We notice that the lowest value is between cluster 1 and cluster 5, meaning that these clusters are quite similar. This finding correlates with the results from the MIREX task, in which the confusion between these two classes was found significant [8]. However the confusion between clusters 2 and 4, also relevant in the MIREX results analysis, is not reflected here. Additionally, we observe that the most separated clusters (5 and 2), are also the less confusing in the MIREX results. Looking at the confusion matrix for the

	C1	C2	C3	C4	C5
C1	0	0.74	0.128	0.204	0.108*
C2	0.74	0	0.859	0.816	**0.876**
C3	0.128	0.859	0	0.319	0.265
C4	0.204	0.816	0.319	0	0.526
C5	0.108*	**0.876**	0.265	0.526	0

Table 3. Confusion matrix for the inter-cluster dissimilarity for the MIREX clusters (C1 means cluster 1, C2 cluster 2 and so on). The values marked with an asterisk are the most similar and in bold are the less similar values.

Hevner clusters (not shown here for space reasons), we remark that the highest values (dissimilarity above 0.95) are between clusters 7 and 8, and between clusters 1 and 2. On the contrary, the lowest value (0.09) is between clusters 1 and 4. Indeed both clusters have words than can appear similar like *spiritual* and *serene* for instance. We summarize the results of both intra and inter-cluster measures for the different taxonomies in Table 4.

Mood Taxonomy	Intra-cluster similarity	Inter-cluster dissimilarity
Hevner	0.55	0.70
MIREX	0.73	0.56
Folksonomy	0.82	0.9

Table 4. Intra-cluster similarity and inter-cluster dissimilarity means for each mood taxonomy.

In a nutshell, the Hevner clusters are less consistent but are more separated than the MIREX ones. Indeed, even if the latter has more intra-cluster similarity, it suffers from confusions between some categories as reflected in our results.

6. DIMENSIONAL REPRESENTATION

Dimensional representation is an important paradigm in emotion studies. To project our semantic mood space into a bi-dimensional space, we used the Self-Organizing Map algorithm (SOM). We decided to use SOM for its topology properties and because it stresses more on the local similarities and distinguishes groups within the data. Because less than half of the Russell's adjectives are present in our dataset, we prefer to compare qualitatively more that quantitatively the expert and the community models. We trained a SOM and mapped each tag onto its best-matching unit in the trained SOM. In Figure 6, we plot the resulting organization of mood tags (for clarity reasons, we show here a subset of 58 tags).

We observe in the 2D projection four main parts. At the top-left, terms related to *aggressive*, below *calm* and other similar words, at the top-right tags related to *sad* and below words close to *happiness*. We notice the four clusters corresponding to the basic emotions and our folksonomy representation mentioned in Section 5.1. This is somehow

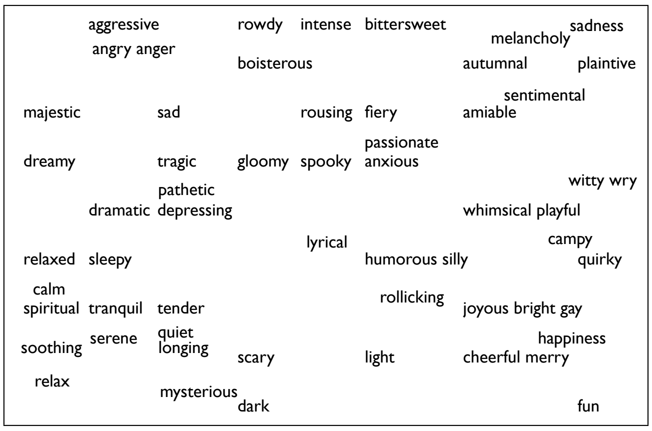


Figure 6. Self-Organizing Map of the mood tags in the semantic space.

expected as we already got these clusters from this data. However, having the same results with a second technique confirms our findings. Comparing with Russell's dimensions, we find that the diagonal from top-left to bottom-right is of high arousal. On the contrary, the diagonal from top-right to bottom-left is of low arousal. The vertical axis represents the valence dimension. Even though the 2D representation is not equal, there is a clear correlation between the community and the experts when framing the problem into two dimensions.

7. HIERARCHICAL REPRESENTATION

The semantic mood space can be visualized in many different ways. In this part we experimented hierarchical clustering techniques to produce a tree diagram (dendrogram). We applied a common agglomerative hierarchical clustering method with a complete linkage [14] and the cosine metric. We used the hcluster[5] implementation. With the 20 most used tags in our dataset, we computed the clustering and plot the resulting dendrogram in Figure 7 .

Although there exists some dendrogram representation of emotions in the psychology literature [1], the comparison is complex because many of the terms employed are not present in our dataset and also because finding the right metric to measure the similarity between both is not trivial. The hierarchical clustering starts with two branches. Looking at the tags of this first branching, we observe a very clear separation in arousal with *dreamy* and *calm* on the left and *angry* and *happy* on the right. Then the two following branching (resulting in four clusters) represents the four basic emotions also found as the best categorical representation in Section 5 (in order in the dendrogram: *calm*, *sad*, *angry* and *happy*). This confirms another time our findings about the relevancy of these four clusters. Moreover, we notice that the first separation is related to arousal, often considered as the most important dimension. The remaining branches group together similar terms like *angry* and *aggressive* or *sad* and *depressing*.

[5] http://code.google.com/p/scipy- cluster

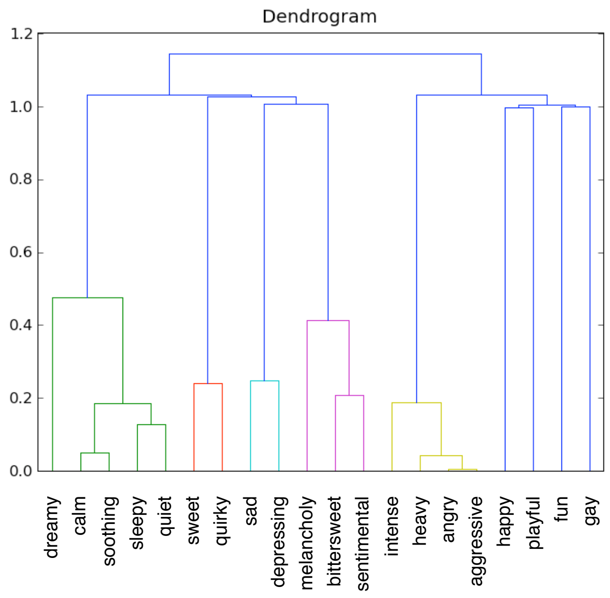


Figure 7. Dendrogram of the 20 most used tags.

8. CONCLUSIONS

This paper presented convergent evidence about mood representations. We created a semantic mood space based on a community of users from *last.fm*. We derived different representations from this data and compared them to the expert representations. We demonstrated that the basic emotions: *happy*, *sad*, *angry* and *tender*, are very relevant to the social network. We also found that the arousal and valence dimensions are pertinent. Moreover we have shown that both Hevner's and MIREX representations have advantages and limitations when evaluated in the semantic mood space. The former having better separated clusters and the latter having more consistent clusters. Observations on the confusion and similarity between MIREX clusters confirmed results from previous analysis. We also presented a dendrogram visualization validating again the basic emotion point of view and offering a new representation of the mood space. All these findings show the relevancy of using a mood semantic space derived from social tags. Folksonomy representations can be used in tasks like mood classification or regression to improve the quality of the audio content processing algorithms. We can also imagine a visualization of a user emotional states based on his listening habits or history. Moreover, one's musical library can be mapped and explored with a folksonomy representation derived from the whole social network or a particular subset. Finally this approach can be generalized to find other domain-specific representations.

9. ACKNOWLEDGMENTS

We want to thank the people from the Music Technology Group (Universitat Pompeu Fabra, Barcelona). Data from this work is available at: *http://mtg.upf.edu/people/claurier* This research has been partially funded by the EU Project PHAROS IST-2006-045035.

10. REFERENCES

[1] P. N. Juslin and J. A. Sloboda: *Music and Emotion: Theory and Research*, Oxford University Press, 2001.

[2] J. A. Russell: "A circumplex model of affect," *Journal of Personality and Social Psychology*, No. 39, pp. 1161, 1980.

[3] T. Li and M. Ogihara: "Detecting emotion in music," *Proceedings of ISMIR, Baltimore, MD, USA*, pp. 239–240, 2003.

[4] Y. H. Yang, Y. C. Lin, Y. F. Su, and H. H. Chen: "A regression approach to music emotion recognition," *IEEE Transactions on audio, speech, and language processing*, Vol. 16, No. 2, pp. 448–457, 2008.

[5] C. Laurier, O. Meyers, J. Serrà, M. Blech, P. Herrera: "Music Mood Annotator Design and Integration," *7th International Workshop on Content-Based Multimedia Indexing, Chania, Crete*, 2009.

[6] C. Laurier, P. Herrera: "Automatic Detection of Emotion in Music: Interaction with Emotionally Sensitive Machines," *Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence*,Chap. 2, pp. 9–32, IGI Global, 2009.

[7] J. S. Downie: "The music information retrieval evaluation exchange (2005-2007): A window into music information retrieval research," *Acoustical Science and Technology*, Vol. 29, No. 4, pp. 247–255, 2008.

[8] X. Hu, S. J. Downie, C. Laurier, M. Bay, and A. F. Ehmann: "The 2007 MIREX audio mood classification task: Lessons learned," *Proceedings of ISMIR, Philadelphia, PA, USA*, pp. 462–467, 2008.

[9] K. Hevner: "Experimental studies of the elements of expression im music," *The American Journal of Psychology*, Vol. 48, No. 2, pp. 246–268, 1936.

[10] M. Sordo, O. Celma, M. Blech, and E. Guaus: "The Quest for Musical Genres: Do the Experts and the Wisdom of Crowds Agree?," *Proceedings of ISMIR, Philadelphia, PA, USA*, pp. 255–260, 2008.

[11] M. Levy and M. Sandler: "A Semantic Space for Music Derived from Social Tags," *Proceedings of ISMIR, Vienna, Austria*, 2007.

[12] S. Deerwester, S. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman : "Indexing by latent semantic analysis.," *Journal of the Society for Information Science*, Vol. 14, pp. 391–407, 1990.

[13] I. H. Witten and E. Frank: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 1999.

[14] R. Xu and D. C. Wunsch: *Clustering*, IEEE Press, 2009

EVALUATION OF ALGORITHMS USING GAMES: THE CASE OF MUSIC TAGGING

Edith Law
CMU
edith@cmu.edu

Kris West
IMIRSEL/UIUC
kris.west@gmail.com

Michael Mandel
Columbia University
mim@ee.columbia.edu

Mert Bay J. Stephen Downie
IMIRSEL/UIUC
mertbay, jdownie@uiuc.edu

ABSTRACT

Search by keyword is an extremely popular method for retrieving music. To support this, novel algorithms that automatically tag music are being developed. The conventional way to evaluate audio tagging algorithms is to compute measures of agreement between the output and the ground truth set. In this work, we introduce a new method for evaluating audio tagging algorithms on a large scale by collecting set-level judgments from players of a human computation game called TagATune. We present the design and preliminary results of an experiment comparing five algorithms using this new evaluation metric, and contrast the results with those obtained by applying several conventional agreement-based evaluation metrics.

1. INTRODUCTION

There is a growing need for efficient methods to organize and search for multimedia content on the Web. This need is reflected in the recent addition of the audio tag classification (ATC) task at MIREX 2008, and the introduction of new music tagging algorithms [1, 2]. The conventional way to determine whether an algorithm is producing accurate tags for a piece of music is to compute the level of agreement between the output generated by the algorithm and the ground truth set. Agreement-based metrics, e.g. accuracy, precision, F-measure and ROC curve, have been long-time workhorses of evaluation, accelerating the development of new algorithms by providing an automated way to gauge performance.

The most serious drawback to using agreement-based metrics is that ground truth sets are never fully comprehensive [3]. First, there are exponentially many sets of suitable tags for a piece of music – creating all possible sets of tags and then choosing the best set of tags as the ground truth is difficult, if not impossible. Second, tags that are appropriate for a given piece of music can simply be missing in the ground truth set because they are less salient, worded differently (e.g. *baroque* versus *17th century classical*), or that they do not facilitate the objectives of the particular annotator. For example, a last.FM user who wants to showcase his expertise on jazz music may tag the music with highly obscure and technical terms. In output-agreement games such as MajorMiner [2] and the Listen Game [4], where the scoring depends on how often players' tags match with one another, players are motivated to enter (or select) tags that are common, thereby omitting tags that are rare or verbose. Furthermore, because an exhaustive set of negative tags is impossible to specify, when a tag is missing, it is impossible to know whether it is in fact inappropriate for a particular piece of music.

Agreement-based metrics also impose restrictions on the type of algorithms that can be evaluated. To be evaluated, tags generated by the algorithms must belong to the ground truth set. This means that audio tagging algorithms that are not trained on the ground truth set, e.g. those that use text corpora or knowledge bases to generate tags, cannot be evaluated using agreement-based metrics.

To be useful, tags generated by audio tagging algorithms must, from the perspective of the *end user*, accurately describe the music. However, because we do not yet fully understand the cognitive processes underlying the representation and categorization of music, it is often difficult to know what makes a tag "accurate" and what kinds of inaccuracies are tolerable. For example, it may be less disconcerting for users to receive a *folk* song when a *country* song is sought, than to receive a *sad, mellow* song when a *happy, up-beat* song is sought. Ideally, an evaluation metric should measure the quality of the algorithm by implicitly or explicitly capturing the users' differential tolerance of incorrect tags generated by the algorithms. The new evaluation metric we are proposing in this paper has exactly this desired property.

The problems highlighted above suggest that music tagging algorithms, especially those used to facilitate retrieval, would benefit enormously from evaluation by human users. Manual evaluation is, however, often too time-consuming or costly to be feasible. Human computation is a new area of research that studies how to build systems, such as simple casual games, to collect annotations from human users. In this work, we investigate the use of a human computation game called TagATune to collect evaluations of algorithm-generated music tags. In an off-season MIREX [5] evaluation task, we compared the performance of five audio tagging algorithms under the newly proposed metric, and present in this paper the preliminary findings.

2. TAGATUNE AS AN EVALUATION PLATFORM

TagATune [6] is a two-player online game that collects music tags from players. In each round of the game, two players are either given the same music clip or different music clips, and are asked to type in tags for their given music clip. After seeing each other's tags, players must then decide whether they were given the same music clip or not.

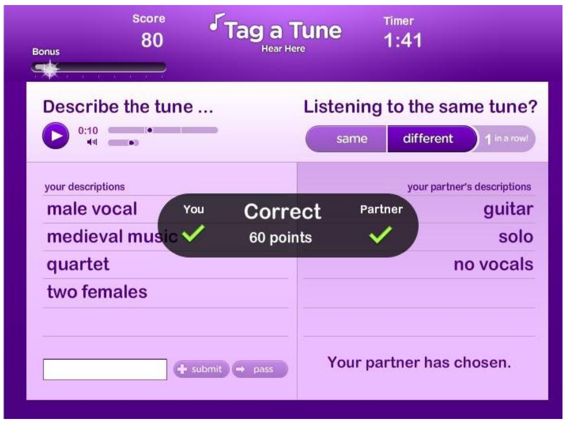


Figure 1. The TagATune interface

When a human partner is not available, a player is paired with a computer bot, which outputs tags that have been previously collected by the game for the particular music clip served in each round. This so-called *aggregate bot* serves tags that are essentially the ground truth, since they were provided by human players.

The key idea behind TagATune as an evaluation platform is that the aggregate bot can be replaced by an *algorithm bot*, which enters tags that were previously generated by an algorithm. An interesting by-product of playing against an algorithm bot is that by guessing same or different, the human player is essentially making a judgment on the appropriateness of the tags generated by the algorithm. Unlike the conventional evaluation metrics where a tag either matches or does not match a tag in the ground truth set, this evaluation method involves set-level judgments and can be applied to algorithms whose output vocabulary is ***arbitrarily different*** from that of the ground truth set.

2.1 Special TagATune Evaluation

To solicit submissions of audio tagging algorithms whose output can be used to construct the TagATune algorithm bots, a "Special TagATune Evaluation" was run off-season under the MIREX rubric. Participating algorithms were asked to provide two different types of outputs:

1. a binary classification decision as to whether each tag is relevant to each clip.

2. a real valued estimate of the 'affinity' of the clip for each tag. Larger values of the affinity score indicate that a tag is more likely to be applicable to the clip.

2.1.1 The Dataset

In the context of the off-season MIREX evaluation task, we trained the participating algorithms on a subset of the TagATune dataset, such that the tags they generated could be served by the algorithm bots in the game. The training and test sets comprise of 16289 and 100 music clips respectively. The test set was limited to 100 clips for both the human evaluation using TagATune and evaluation using the conventional agreement-based metrics, in order to facilitate direct comparisons of their results. Each clip is 29 seconds long, and the set of clips are associated with 6622 tracks, 517 albums and 270 artists. The dataset is split such that the clips in the training and test sets do not belong to the same artists. Genres include Classical, New Age, Electronica, Rock, Pop, World, Jazz, Blues, Metal, Punk etc. The tags used in the experiments are each associated with more than fifty clips, where each clip is associated only with tags that have been verified by more than two players independently.

2.1.2 Participating Algorithms

There were five submissions, which we will refer to as Mandel, Manzagol, Marsyas, Zhi and LabX[1] from this point on. A sixth algorithm we are using for comparison is called AggregateBot, which serves tags from a vocabulary pool of 146 tags collected by TagATune since deployment, 91 of which overlap with the 160 tags used for training the algorithms. The inclusion of AggregateBot demonstrates the utility of TagATune in evaluating algorithms that have different tag vocabulary.

2.1.3 Game-friendly Evaluation

An important requirement for using human computation games for evaluation is that the experiment does not significantly degrade the game experience. We describe here a few design strategies to maintain the enjoyability of the game despite the use of algorithm bots whose quality cannot be gauged ahead of time.

First, a TagATune round is randomly chosen to be used for evaluation with some small probability x. This prevents malicious attempts to artificially boost or degrade the evaluation of particular algorithms, which would be easy to do if players can recognize that they are playing against an algorithm bot. Second, while it may be acceptable to use half of the rounds in a game for evaluating good algorithms, one round may be one too many if the algorithm under evaluation always generates completely wrong tags. Since we do not know ahead of time the quality of the algorithms being evaluated, x must be small enough such that the effects of bad algorithms on the game will be minimized. Finally, using only a small portion of the game for evaluation ensures that a wide variety of music is served, which is especially important when the test set is small.

[1] The LabX submission was identified as having a bug which negatively impacted its performance, hence, the name of the participating laboratory has been obfuscated. Since LabX essentially behaves like an algorithm that randomly assigns tags, its performance establishes a lower bound for the TagATune metric.

Despite the small probability of using each round for evaluation, the game experience can be ruined by an algorithm that generates tags are contradictory (e.g. *slow* followed by *fast*, or *guitar* followed by *no guitar*) or redundant (e.g. *string*, *violins*, *violin*). Our experience shows that players are even less tolerant of a bot that appears "stupid" than of one that is wrong. Unfortunately, such errors occur quite frequently. Table 1 provides a summary of the number of tags generated (on average) by each algorithm for the clips in the test set, and how many of those are removed because they are contradictory or redundant.

Algorithm	Generated	Contradictory or Redundant
Mandel	36.47	16.23
Marsyas	9.03	3.47
Manzagol	2.82	0.55
Zhi	14.0	5.04
LabX	1.0	0.00

Table 1. Average number of tags generated by algorithms and contradictory/redundant ones among the generated tags

To alleviate this problem, we perform the following post-processing step on the output of the algorithms. First, we retain only tags that are considered relevant according to the binary outputs. Then, we rank the tags by affinity. Finally, for each tag, starting from the highest affinity, we remove lower affinity tags with which it is mutually exclusive. Although this reduces the number of tags available to the algorithm bots to serve in the game, we believe that this is a sensible post-processing step for any tag classification algorithms.

An alternative method of post-processing would be to first organize the "relevant" tags into categories (e.g. genre, volume, mood) and retain only the tag with the highest affinity score in each category, thereby introducing more variety in the tags to be emitted by the algorithm bots. We did not follow this approach because it may bias performance in an unpredictable fashion and favour the output of certain algorithms over others.

2.1.4 Evaluation Using The TagATune Metric

During an evaluation round, an algorithm is chosen to emit tags for a clip drawn from the test set. The game chooses the algorithm-clip pair in a round robin fashion but favors pairs that have been seen by the least number of unique human players. In addition, the game keeps track of which player has encountered which algorithm-clip pair, so that each evaluator for a given algorithm-clip pair is unique.

Suppose a set of algorithms $\mathcal{A} = \{a_i, \ldots, a_{|\mathcal{A}|}\}$ and a test set $\mathcal{S} = \{s_j, \ldots, s_{|\mathcal{S}|}\}$ of music clips. During each round of the game, a particular algorithm i is given a clip j from the test set and asked to generate a set of tags for that clip. To be a valid evaluation, we only use rounds where the clips given to the human player and the algorithm bot are the same. This is because if the clips are different, an algorithm can output the wrong tags for a clip and actually *help* the players guess correctly that the clips are different.

A human player's guess is denoted as $G = \{0, 1\}$ and the ground truth is denoted as $GT = \{0, 1\}$, where 0 means that the clips are the same and 1 means that the clips are different. The performance P of an algorithm i on clip j under TagATune metric is as follows:

$$P_{i,j} = \frac{1}{N}\sum_{n}^{N} \delta(G_{n,j} = GT_j) \quad (1)$$

where N represents the number of players who were presented with the tags generated by algorithm i on clip j, and $\delta(G_{n,j} = GT_j)$ is a Kronecker delta function which returns 1 if, for clip j, the guess from player n and the ground truth are the same, 0 otherwise. The overall score for an algorithm is averaged over the test set S:

$$P_i = \frac{1}{S}\sum_{j}^{S} P_{i,j} \quad (2)$$

2.1.5 Evaluation Using Agreement-Based Metrics

We have chosen to compute the performance of the participating algorithms using a variety of agreement-based metrics that were included in the 2008 MIREX ATC task, as a comparison against the TagATune metric. These metrics include precision, recall, F-measure [7], the Area Under the Receiver Operating Characteristic curve (AUC-ROC) and the accuracy of the positive and negative example sets for each tag. We omitted the "overall accuracy" metric, as it is a very biased statistics for evaluating tag classification models where there is a large negative to positive tag ratio.

As the TagATune game and metric necessarily focus on the first few tags returned by an algorithm (i.e. tags that have the highest affinity scores), we chose to also calculate the Precision-at-N (*P@N*) score for each algorithm. This additional set of statistics allows us to explore the effect of sampling the top few tags on the performance of the algorithms.

2.1.6 Statistical Significance

Friedman's ANOVA is a non-parametric test that can be used to determine whether the difference in performance between algorithms is statistically significant [5].

For each algorithm, a performance score is computed over the test set. Using the TagATune metric, this performance score is the percentage of unique players that correctly judged that the clips are the same or not using the tags emitted by the algorithm, computed using equation (1) and (2). For automated statistical evaluations, such as those performed during the MIREX ATC task, these may be the F-measure or *P@N* for the "relevant" tags generated for each clip, or the AUC-ROC for the "affinity" scores. These scores can be viewed as a rectangular matrix, with the different tagging algorithms represented as the columns and the clips (or the tags, in the case of F-measure aggregated over each tag) forming the rows.

To avoid having variance introduced by different tags affecting the scaling and distribution of the scores, Friedman's test replaces the performance scores with their ranks amongst the algorithms under comparison.

Algorithm	**TagATune metric**	+ve Example Accuracy	-ve Example Accuracy	Precision	Recall	**F-measure**
AggregateBot	**93.00%**	–	–	–	–	–
Mandel	**70.10%**	**73.13%**	80.29%	0.1850	**0.7313**	0.2954
Marsyas	68.60%	45.83%	96.82%	**0.4684**	0.4583	**0.4633**
Manzagol	67.50%	13.98%	98.99%	0.4574	0.1398	0.2141
Zhi	60.90%	40.30%	93.18%	0.2657	0.4030	0.3203
LabX	26.80%	0.33%	**99.36%**	0.03	0.0033	0.0059

Table 2. Evaluation statistics under the TagATune versus agreement-based metrics

Algorithm	Precision at N					Precision for ‘relevant’ tags	AUC-ROC
	3	6	9	12	15		
Mandel	0.6133	0.5083	0.4344	0.3883	0.3387	0.1850	0.8514
Marsyas	**0.7433**	**0.5900**	**0.4900**	**0.4308**	**0.3877**	**0.4684**	**0.9094**
Manzagol	0.4767	0.3833	0.3222	0.2833	0.2520	0.4574	0.7521
Zhi	0.3633	0.3383	0.3100	0.2775	0.2480	0.2657	0.6697
LabX	–	–	–	–	–	0.03	–

Table 3. Precision and AUC-ROC statistics collected for each algorithm

Friedman’s ANOVA is used to determine if there exists a significant difference in performance amongst a set of algorithms. If a difference is detected, then it is common to follow up with a Tukey-Kramer Honestly Significant Difference (TK-HSD) test to determine which pairs of algorithms are actually performing differently. This method does not suffer from the problem that multiple t-tests do where the probability of incorrectly rejecting the null hypothesis (i.e. that there is no difference in performance) increases in direct proportion to the number of pairwise comparisons conducted.

3. RESULTS

Tables 2 and 3 provide summaries of the evaluation statistics collected for each algorithm under the TagATune metric as well as agreement-based metrics. Each of the summary results was computed over the 100 clips in the test set, while the statistical significance tests were computed over the results for each individual clip. The following sections detail additional statistics that were collected by the TagATune evaluation.

3.1 Algorithm Ranking

According to the TK-HSD test on the TagATune metric results, AggregateBot’s performance is significantly better than all the others. A second group of equally performing algorithms consists of Mandel, Manzagol, Marsyas, and Zhi. LabX is the sole member of the worst performing group. Figure 2 highlights these TK-HSD performance groupings.

Several authors have speculated on the possibility of a “glass-ceiling” on the performance of current music classification and similarity estimation techniques. As identified by Aucouturier [8], many of these techniques are based on ‘bag-of-frames’ approaches to the comparison of the audio streams. Hence, the lack of a significant difference among the performances of the correctly functioning algorithms is not surprising.

The TK-HSD ordering of the algorithms using the F-measure scores (Table 2 and Figure 3) is different from that produced by the TagATune scores. Notably, the Marsyas algorithm significantly outperforms the other algorithms and the Zhi algorithm has improved its relative rank considerably.

These differences may be attributed to the fact that the performance of the Marsyas and Zhi algorithm is more balanced in terms of precision and recall than the Mandel algorithm (which exhibits high recall but low precision) and the Manzagol algorithm (which exhibits high precision but low recall). This conclusion is reinforced by the positive and negative accuracy scores, which demonstrate the tendency of the Mandel algorithm to over-estimate and Manzagol to under-estimate relevancy. Metrics that take into account the accuracies of all tags (e.g. F-measure) are particularly sensitive to these tendencies, while metrics that consider only the top N tags (e.g. the TagATune metric and $P@N$) are affected little by them.

These results suggest that the choice of an evaluation metric or experiment must take into account the intended application of the tagging algorithms. For example, the TagATune metric may be most suitable for evaluating retrieval algorithms that use only the highest ranked tags to

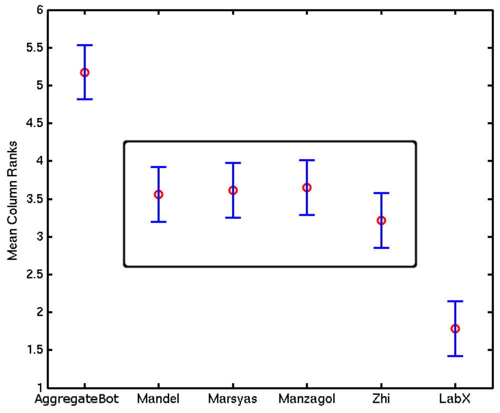


Figure 2. Tukey-Kramer HSD results based on the TagATune metric

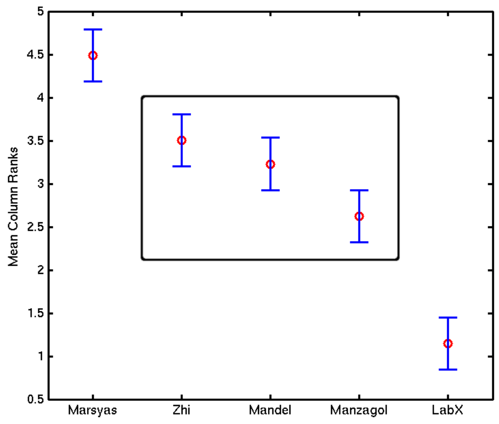


Figure 3. Tukey-Kramer HSD results based on the F-measure metric

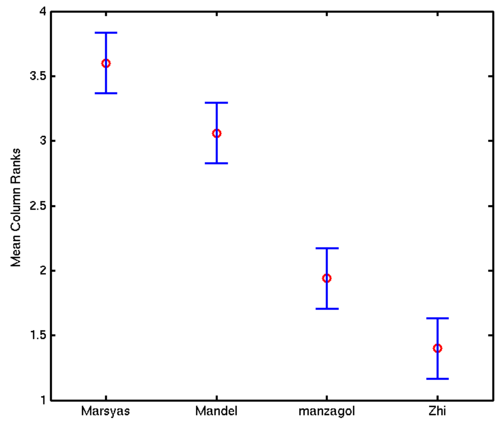


Figure 4. Tukey-Kramer HSD results based on the AUC-ROC metric

compute the degree of relevance of a song to a given query. However, for applications that consider the all relevant tags regardless of affinity, e.g. unweighted tag clouds generators, the TagATune metric is not necessarily providing an accurate indication of performance, in which case the F-measure might be a better candidate.

3.2 Game Statistics

In a TagATune round, the game selects a clip from the test set and serves the tags generated by a particular algorithm for that clip. For each of the 100 clips in the test set and for each algorithm, 10 unique players were elicited (unknowingly) by the game to provide evaluation judgments. This totals to 5000 judgments, collected over a one month period, involving approximately 2272 games and 657 unique players.

3.2.1 Number of tags reviewed

One complication with using TagATune for evaluation is that players are allowed to make the decision of guessing same or different at any point during a round. This means that the number of tags reviewed by the human player varies from clip to clip, algorithm to algorithm. As a by-product of game play, players are motivated to guess as soon as they believe that they have enough information to guess whether the clips are the same or different. Figure 5, which shows that players reviewed only a small portion of the generated tags before guessing, reflects this situation.

3.2.2 Correlation with precision

Figure 6 shows the average number of tags reviewed by players and how many of the reviewed tags are actually true positive tags (according to the ground truth) in success rounds (where the human player made the correct guess) versus failed rounds (where the human player made the wrong guess). Results show that generally the number of true positive tags reviewed is greater in success rounds than in failed rounds, suggesting that players are more likely to fail at guessing when there are more top-affinity tags that are wrong. Additionally, the average number of tags reviewed before guessing is fewer in the failed rounds than in the success rounds, with the exception of Mandel, possibly due to outliers and the much greater number of tags that this algorithm returns. This suggests that players make their guesses more hastily when algorithms make mistakes.

3.2.3 Detectable errors

A natural question to ask is whether one can detect from game statistics which of the reviewed tags actually caused players to guess incorrectly.

System	failed round	success round
Mandel	86.15%	49.00%
Marsyas	80.49%	45.00%
Manzagol	76.92%	33.33%
Zhi	84.38%	70.10%
LabX	100.0%	95.77%

Table 4. Percentage of the time that the last tag displayed before guessing is wrong in a failed round versus success round

To investigate this question, we consult the game statistics for the most frequent behavior of human players in terms of the number of tags reviewed before guessing, in the case when the guess is wrong. For example, we might find that most players make a wrong guess after reviewing n tags for a particular algorithm-clip pair. The hypothesis is that the last tag reviewed before guessing, i.e. the n^{th} tag, is the culprit.

Table 4 shows the percentage of times that the n^{th} tag is actually wrong in failed rounds, which is above 75% for all algorithms. In contrast, the probability of the last tag being wrong is much lower in success rounds, showing that using game statistics alone, one can detect problematic tags that cause most players to make the wrong guess in the game. This trend does not hold for LabX, possibly because players were left guessing randomly due to the lack of information (since this algorithm generated only one tag per clip).

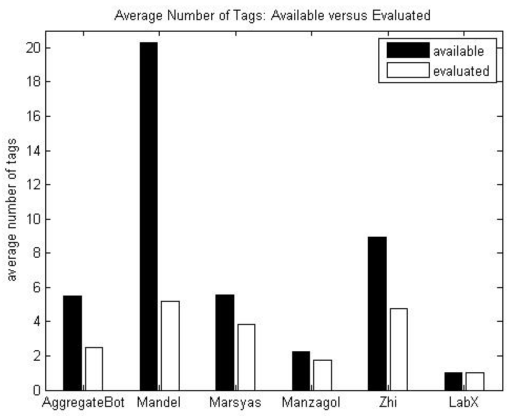


Figure 5. Number of tags available and reviewed by players before guessing

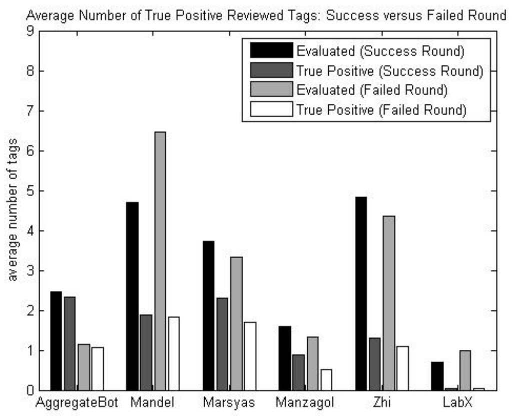


Figure 6. Number of overall and true positive tags evaluated in success and failed rounds

4. CONCLUSION

This paper introduces a new method for evaluating music tagging algorithms and presents the results of a proof-of-concept experiment using a human computation game as an evaluation platform for algorithms. This experiment has also been used to explore the behavior of conventional agreement-based metrics and has shown that averaged retrieval statistics, such as F-measure, can be sensitive to certain tendencies (e.g. imbalanced performance in terms of precision versus recall) that do not affect the TagATune metric, which considers the accuracies of only the top most relevant tags.

While there are many benchmarking competitions for algorithms, little is said about the level of performance that is acceptable for real world applications. In this work, we have shown that the use of aggregate data in the bot provides a performance level against which the algorithms can be judged. Specifically, human players can correctly guess that the music are the same 93% of the times when paired against the aggregate bot, while only approximately 70% of the times when paired against an algorithm bot.

Finally, our work has shown that TagATune is a feasible and cost-effective platform for collecting a large number of evaluations from human users in a timely fashion. This result is particularly encouraging for future research on using human computation games to evaluate algorithms in other domains, such as object recognition and machine translation.

5. ACKNOWLEDGMENT

MIREX has received considerable financial support from both the Andrew W. Mellon Foundation and the National Science Foundation (NSF) under grants NSF IIS-0328471 and NSF IIS-0327371. Edith Law is generously funded by a Microsoft Graduate Research Fellowship. Many thanks to those who participated in the MIREX Special TagATune Evaluation.

6. REFERENCES

[1] Doug Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic-description using the cal500 data set. In *SIGIR*, pages 439–446, 2007.

[2] Michael I. Mandel and Daniel P. W. Ellis. A web-based game for collecting music metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

[3] Edith Law. The problem of accuracy as an evaluation criterion. *ICML Workshop on Evaluation Methods in Machine Learning*, 2008.

[4] Douglas Turnbull, Ruoran Liu, Luke Barrington, and Gert Lanckriet. A game-based approach for collecting semantic annotations of music. In *ISMIR*, pages 535–538, 2007.

[5] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[6] Edith Law and Luis von Ahn. Input-agreement: A new mechanism for data collection using human computation games. *CHI*, pages 1197–1206, 2009.

[7] Keith Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.

[8] Jean-Julien Aucouturier. *Ten experiments on the modelling of polyphonic timbre*. PhD thesis, University of Paris 6, France, June 2006.

AUTOMATIC IDENTIFICATION FOR SINGING STYLE BASED ON SUNG MELODIC CONTOUR CHARACTERIZED IN PHASE PLANE

Tatsuya Kako†, Yasunori Ohishi‡, Hirokazu Kameoka‡, Kunio Kashino‡, Kazuya Takeda†

†Graduate School of Information Science, Nagoya University

‡NTT Communication Science Laboratories, NTT Corporation

kako@sp.m.is.nagoya-u.ac.jp, ohishi@cs.brl.ntt.co.jp, kameoka@eye.brl.ntt.co.jp
kunio@eye.brl.ntt.co.jp, kazuya.takeda@nagoya-u.jp

ABSTRACT

A stochastic representation of singing styles is proposed. The dynamic property of melodic contour, i.e., fundamental frequency (F_0) sequence, is assumed to be the main cue for singing styles because it can characterize such typical ornamentations as *vibrato* . F_0 signal trajectories in the phase plane are used as the basic representation. By fitting Gaussian mixture models to the observed F_0 trajectories in the phase plane, a parametric representation is obtained by a set of GMM parameters. The effectiveness of our proposed method is confirmed through experimental evaluation where 94.1% accuracy for singer-class discrimination was obtained.

1. INTRODUCTION

Although no firm definition has yet been established for "singing style" in musical information processing research, several studies have reported the relationship between singing styles and such signal features as singing formant [1, 2] and singing ornamentations. Various research efforts have been made to characterize ornamentations by the acoustical property of the sung melody, i.e., *vibrato* [3–11], overshoot [12], and fine fluctuation [13]. The importance of such melodic features for perceiving singer individuality was also reported in [14] based on psycho-acoustic experiments. They concluded that the average spectrum and the dynamical property of the F_0 sequence affect the perception of the individuality. Those studies suggest that singing style is related to the local dynamics of a sung melody that does not contain any musical information. Therefore, in this study, we focus on the local dynamics of the F_0 sequence, i.e., the melodic contour, as a cue of singing style and propose a parametric representation as a model for singing styles.

On the other hand, very few application systems have been reported that use the local dynamics of a sung melody. [15] reported a singer recognition experiment using *vibrato* . [16] reported a method for evaluating singing skill through the spectrum analysis of the F_0 contour. Although these studies try to use the local dynamics of melodic contour as a cue for ornamentation, no systematic method has been proposed for characterizing singing styles. A lag system model for typical ornamentations was reported in [14,17–19]; however, variation of singing styles was not discussed.

In this paper, we propose a stochastic phase plane as a graphical representation of singing styles and show its effectiveness for singing style discrimination. One merit of this representation to characterize singing style is that since neither an explicit detection function for ornamentation like *vibrato* nor estimation of the target note is required, it is robust to sung melodies.

In a previous paper [20], we applied this graphical representation of the F_0 contour in the phase plane to a query-by-hamming system and neutralized the local dynamics of the F_0 sequence so that only musical information was utilized for the query. In contrast, in this study, we use the local dynamics of the F_0 sequence for modeling singing styles and disregard the musical information because musical information and singing style are in a dual relation.

In this paper, we also evaluate the proposed representation through a singer-class discrimination experiment in which we show that our proposed model can extract the dynamic properties of sung melodies shared by a group of singers.

In the next section, we propose stochastic phase plane (SPP) as a stochastic representation of the melodic contour and show how singing ornamentations are modeled by the proposed SPP. In Section 3, we experimentally show the effectiveness of our proposed method through singer class discrimination experiments. Section 4 discusses the obtained results and concludes this paper.

2. STOCHASTIC REPRESENTATION OF THE DYNAMICAL PROPERTY OF MELODIC CONTOUR

2.1 F_0 signal in the Phase Plane

Such ornamental expressions in singing as *vibrato* are characterized by the dynamical property of their F_0 signal. Since the F_0 signal is a controlled output of the human speech production system, its basic dynamical characteristics can be related to a differential equation. Therefore, we can use the phase plane, which is the joint plot of a variable and its time derivative, i.e., $(x, \dot{x})$, to depict its dynamical property.

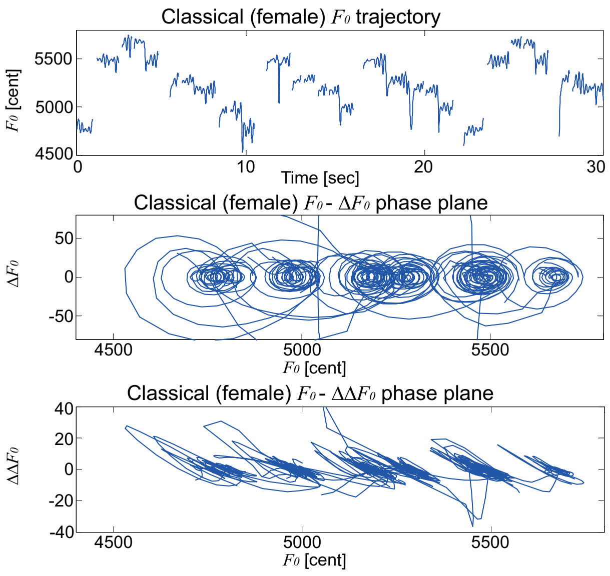


Figure 1. Melodic contour (top) and corresponding phase planes for F_0-ΔF_0 (middle) and F_0-$\Delta\Delta F_0$ (bottom)

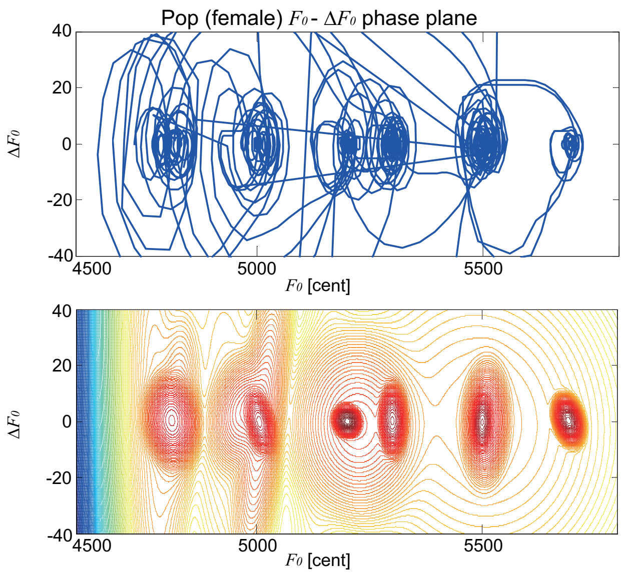


Figure 2. Gaussian mixture model fitted to F_0 contour in phase plane

Although the signal sequence is not given as an explicit function of time, $F_0(t)$, but as a sequence of numbers, $\{F_0(n)\}_{n=1,\cdots,N}$, we can estimate the time derivative using the *delta*- coefficient given by

$$\Delta F_0(n) = \frac{\sum_{k=-K}^{K} k \cdot F_0(n+k)}{\sum_{k=-K}^{K} k^2}, \tag{1}$$

where $2K$ is the window length for calculating the dynamics. Changing the window length extracts different aspects of the signal property.

An example of such a plot for a given melodic contour is shown in Fig. 1. Here, the F_0 signal (top), the phase plane (middle), and the second order phase plane, which is given by the joint plot of F_0 and $\Delta\Delta F_0$ (bottom), are plotted. The singing ornamentations are depicted as the local behavior of the trajectory around the centroids that commonly represent target musical notes. *Vibrato* in singing, for example, is shown as circular trajectories centered at target notes. In the second order plane, the trajectories appear as lines with a slope of -45 degrees. This shows that the relationship between F_0 and $\Delta\Delta F_0$ is given as

$$\Delta\Delta F_0 = -F_0. \tag{2}$$

Hence, the sinusoidal component is imposed in the given signal. Over/under-shoots to the target note are represented as spiral patterns around the note.

2.2 Stochastic representation of Phase Plane

Once a singing style is represented as a phase plane trajectory, parameterizing the representation becomes an issue for further engineering applications. Since the F_0 signal is not deterministic, i.e., it varies across singing behaviors, a stochastic model must be defined for the parameterization. By fitting a parametric probability density function to the trajectories in the phase plane, we can build a stochastic phase plane (SPP) and use it for characterizing the melodic contour. A common feature of the trajectories in the phase plane is that most of their segments are distributed around the target note, and therefore the distribution's histogram is multimodal, but each mode can be represented by a simple symmetric 2d or 3d-pdf. Therefore, Gaussian mixture model (GMM),

$$\sum_{m=1}^{M} \lambda_m \mathcal{N}(\mathbf{f}_0(n); \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{3}$$

where

$$\mathbf{f}_0(n) = \left[F_0(n), \Delta F_0(n), \Delta\Delta F_0(n)\right]^{\mathrm{T}}, \tag{4}$$

is adopted for the modeling. $\mathcal{N}(\cdot)$ is a Gaussian distribution, and

$$\Theta = \{\lambda_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1,\cdots,M}, \tag{5}$$

are parameters of the model, each of which represents the relative frequency, the mean vector, and the covariance matrix of each Gaussian.

A GMM trained for F_0 contours in the phase plane is depicted in Fig. 2. A smooth surface is trained through model fitting. The horizontal deviations of each Gaussian represent the stability of the melodic contour around the target note, but the vertical deviations represent the *vibrato* depth. In this manner, singing styles can be modeled by set of parameters Θ of the stochastic phase plane.

2.3 Examples of Stochastic Phase Plane

In Fig. 3, the F_0 signals of three female singers are plotted: professional classical, professional pop, and an amateur. A deep *vibrato* is observed as a large vertical deviation in the Gaussians in the professional classical singer's plot. On the other hand, the amateur's plot is characterized by large horizontal deviations. Although deep *vibrato* is not observed in the plot for the professional pop singer, its smaller horizontal deviation shows that she accurately sang the melody.

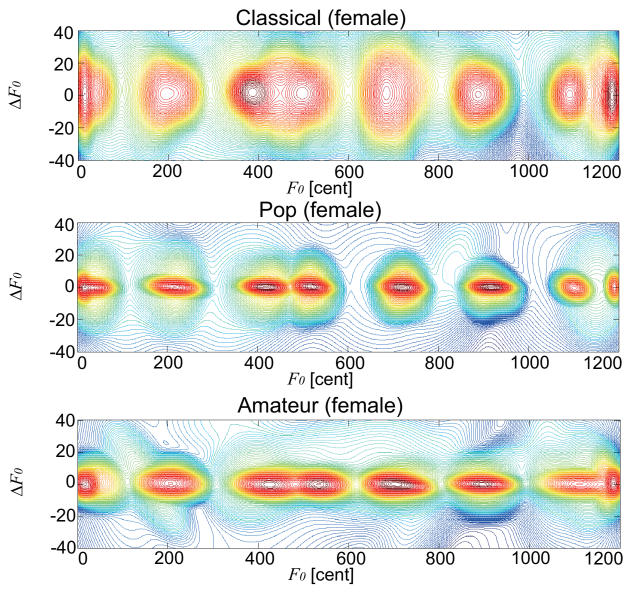


Figure 3. Stochastic phase plane models for professional classical (top), professional pop (middle), and amateur (bottom)

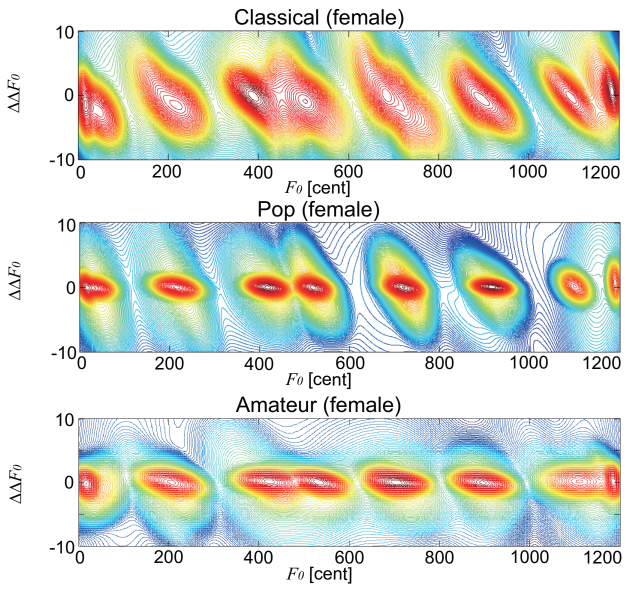


Figure 4. 2nd order stochastic phase plane models for professional classical (top), professional pop (middle), and amateur (bottom)

Table 1. Signal analysis conditions for F_0 estimation. Harmonical PSD pattern matching [21] is used with these parameters.

Signal sampling freq.	16 kHz
F_0 estimation window length	64 ms
Window function	Hanning window
Window shift	10 ms
F_0 contour smoothing	50 ms MA filter
Δ coefficient calculation	$K = 2$

The stochastic representations of the second order phase plane are also shown in Fig. 4. Strong negative correlations between F_0 and $\Delta\Delta F_0$ can be found only in the plot for the professional classical singer that also indicates deep *vibrato* in the singing style.

3. EXPERIMENTAL EVALUATION

The effectiveness of using SPP to discriminate different singing styles is evaluated experimentally.

3.1 Experimental set up

The following singing signals of six singers were used: one of each gender in the categories of professional classical, professional pop, and amateur. With/without musical accompaniment, each subject sang songs with Japanese lyrics and hummed. The songs were "*Twinkle, Twinkle, Little Star*", and "*Ode to Joy*" and five etudes. A total of 102 song signals was recorded.

The F_0 contour was estimated using [21]. The signal processing conditions for calculating F_0, ΔF_0, and the $\Delta\Delta F_0$ contours are listed in Table 1.

Since the absolute pitch of the song signals differ across singers, we normalized them so that only the singing style of each singer is used in the experiment. Normalization was done in the procedure below. First, the F_0 frequency in `[Hz]` is converted to `[cent]` by

$$1200 \times \log_2 \frac{F_0}{440 \times 2^{3/12-5}} \quad \texttt{[cent]}. \qquad (6)$$

Then the local deviations from the tempered clavier are calculated by the residue operation mod$(\cdot)$:

$$\text{mod } (F_0 + 50, 100). \qquad (7)$$

Obviously, after this conversion, the F_0 value is limited to $(0, 100)$ in `[cent]`.

3.2 Discrimination Experiment

The discrimination of three singer classes, i.e., professional classical, professional pop, and amateur, was performed based on the maximum *a posteriori* probability (MAP) decision:

$$\begin{aligned}\hat{s} &= \arg\max_s \left[p(s|\{F_0, \Delta F_0, \Delta\Delta F_0\})\right] \\ &= \arg\max_s \left[\frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{f}_0(n)|\Theta_s) + \log p(s)\right] \quad (8)\end{aligned}$$

where s is the singer-class id and Θ_s is the model parameters of the s^{th} singer-class. We used "*Twinkle-Twinkle, Little Star*" and five etudes sung by singers from each singer class for training and "*Ode to Joy*" sung by the same singers for testing. Therefore the results are independent from sung melodies but closed in singers. N is the length of the signal in the samples. Since we assumed an equal *a priori* probability for singer-class distribution $p(s)$, the above MAP decision is equivalent to the Maximum Likelihood decision.

3.3 Results

Fig. 5 shows the accuracy of the singer-class discrimination. The best is attained for a 13-second input sig-

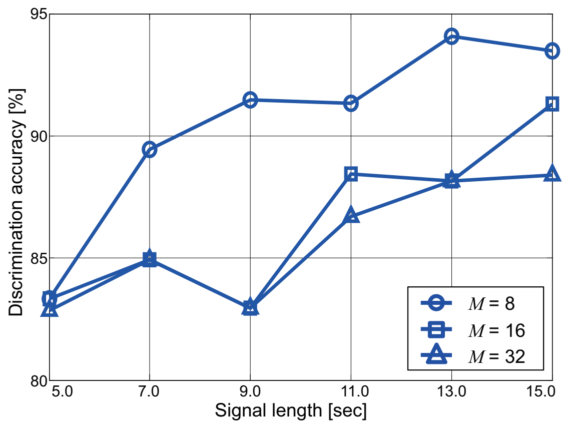


Figure 5. Accuracy in discriminating three singer classes

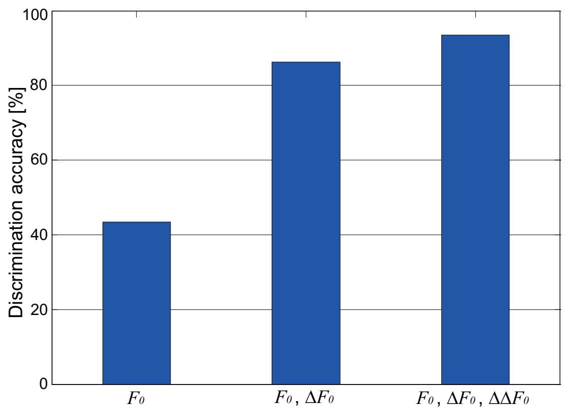


Figure 6. Comparing accuracy in discriminating singer classes

nal. The accuracy increases with the length of the test signal and 94.1% is attained with an 8-mixture GMM for singer-class models, when a 13-second signal is available for the test input. No significant improvement in accuracy was found for the longer test input because more song-dependent information contaminated the test signal. Fig. 6 compares the accuracy of singer-class discriminations using the three sets of features: F_0 only, (F_0, ΔF_0), and (F_0, ΔF_0, $\Delta\Delta F_0$). As shown in the figure, by combining F_0 and ΔF_0, the discrimination error rate becomes half of the error when only using F_0. Combining second order derivative $\Delta\Delta F_0$ further reduces the error but not as much as the case of ΔF_0. These results show that the proposed stochastic representation of the phase plane effectively characterizes the singing styles of the three singer classes.

4. DISCUSSION

Our proposed method for representing and parameterizing the F_0 contour effectively discriminates the three typical singer classes, i.e., professional classical and pop, and amateurs. To confirm that the method models the singing styles (and not singer individuality), we compared our proposed representation with MFCC under two conditions. As a closed condition, we trained three MFCC-GMMs using "*Twinkle-Twinkle, Little Star*" and five etudes sung by six (male and female professional classic, professional pop, and amateur) singers and used "*Ode to Joy*" sung by

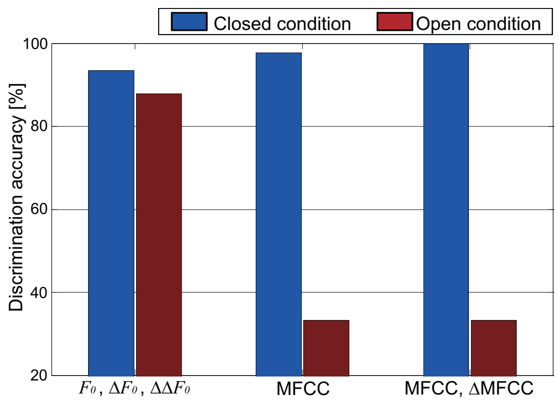


Figure 7. Comparing proposed representation with MFCC under two conditions

the same singers for testing. On the other hand, as an open condition, we evaluated the MFCC-GMMs through a singer independent manner where singer-class models (GMMs) were trained by female singer data and tested by male singer data. As shown in Fig. 7, the performances of the MFCC-GMM and the proposed method are almost identical (95.0%) in the closed condition. However, in the new (unseen) singer experiment, the result of the MFCC-GMM system significantly degraded to 33.3%, but the proposed method attained 87.9% accuracy. These results suggest that the MFCC-GMM system does not model the singing style but discriminates singer individuality. However, since SPP-GMM can correctly classify even an unseen singer's data, our proposed representation models the F_0 dynamic characteristics common within a singer class better than singer individuality.

5. SUMMARY

In this paper, we proposed a model for singing styles based on the stochastic graphical representation of the local dynamical property of the F_0 sequence. Since various singing ornamentations are related to signal production systems described by differential equations, phase plane is a reasonable space for depicting singing styles. Furthermore, the Gaussian mixture model effectively parameterizes the graphical representation; therefore, more than 90% accuracy can be achieved in discriminating the three classes of singers.

Since the scale of the experiments was small, increasing the number of singers and singer classes is critical future work. Evaluating the robustness of the proposed method to noisy F_0 sequences estimated under such realistic singing conditions as "karaoke" is also an inevitable step for building real-world application systems.

6. REFERENCES

[1] J. Sundberg, *The Science of the Singing*. Northern Illinois University Press, 1987.

[2] J. Sundberg, "Singing and timbre," *Music room acoustics*, vol. 17, pp. 57–81, 1977.

[3] C. E. Seashore, “A musical ornament, the vibrato,” in *Proc. Psychology of Music*. McGraw-Hill Book Company, 1938, pp. 33–52.

[4] J. Large and S. Iwata, “Aerodynamic study of vibrato and voluntary ”straight tone” pairs in singing,” *J. Acoust. Soc. Am.*, vol. 49, no. 1A, p. 137, 1971.

[5] H. B. Rothman and A. A. Arroyo, “Acoustic variability in vibrato and its perceptual significance,” *J. Voice*, vol. 1, no. 2, pp. 123–141, 1987.

[6] D. Myers and J. Michel, “Vibrato and pitch transitions,” *J. Voice*, vol. 1, no. 2, pp. 157–161, 1987.

[7] J. Hakes, T. Shipp, and E. T. Doherty, “Acoustic characteristics of vocal oscillations: Vibrato, exaggerated vibrato, trill, and trillo,” *J. Voice*, vol. 1, no. 4, pp. 326–331, 1988.

[8] C. D’Alessandro and M. Castellengo, “The pitch of short-duration vibrato tones,” *J. Acoust. Soc. Am.*, vol. 95, no. 3, pp. 1617–1630, 1994.

[9] D. Gerhard, “Pitch track target deviation in natural singing,” in *Proc. ISMIR*, 2005, pp. 514–519.

[10] K. Kojima, M. Yanagida, and I. Nakayama, “Variability of vibrato -a comparative study between japanese traditional singing and bel canto-,” in *Proc. Speech Prosody*, 2004, pp. 151–154.

[11] I. Nakayama, “Comparative studies on vocal expressions in japanese traditional and western classical-style singing, using a common verse,” in *Proc. ICA*, 2004, pp. 1295–1296.

[12] G. de Krom and G. Bloothooft, “Timing and accuracy of fundamental frequency changes in singing,” in *Proc. ICPhS*, 1995, pp. 206–209.

[13] M. Akagi and H. Kitakaze, “Perception of synthesized singing voices with fine fluctuations in their fundamental frequency contours,” in *Proc. ICSLP*, 2000, pp. 458–461.

[14] T. Saitou, M. Goto, M. Unoki, and M. Akagi, “Speech-To-Singing synthesis: Converting speaking voices to singing voices by controlling acoustic features unique to singing voices,” in *Proc. WASPAA*, 2007, pp. 215–218.

[15] T. L. Nwe and H. Li, “Exploring vibrato-motivated acoustic features for singer identification,” *IEEE Transactions on Audio, Speech, and Language processing*, pp. 519–530, 2007.

[16] T. Nakano, M. Goto, and Y. Hiraga, “An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features,” in *Proc. Interspeech*, 2006, pp. 1706–1709.

[17] H. Mori, W. Odagiri, and H. Kasuya, “F0 dynamics in singing: Evidence from the data of a baritone singer,” *IEICE Trans. Inf. and Syst.*, vol. E87-D, no. 5, pp. 1086–1092, 2004.

[18] N. Minematsu, B. Matsuoka, and K. Hirose, “Prosodic modeling of nagauta singing and its evaluation,” in *Proc. SpeechProsody*, 2004, pp. 487–490.

[19] L. Reqnier and G. Peeters, “Singing voice detection in music tracks using direct voice vibrato,” in *Proc. ICASP*, 2009, pp. 1658–1688.

[20] Y. Ohishi, M. Goto, K. Itou, and K. Takeda., “A stochastic representation of the dynamics of sung melody,” in *Proc. ISMIR*, 2007, pp. 371–372.

[21] M. Goto, K. Itou, and S. Hayamizu, “A real-time filled pause detection system for spontaneous speech recognition,” in *Proc. Eurospeech*, 1999, pp. 227–230.

AUTOMATIC IDENTIFICATION OF INSTRUMENT CLASSES IN POLYPHONIC AND POLY-INSTRUMENT AUDIO

Philippe Hamel, Sean Wood and Douglas Eck
Université de Montréal
Département d'informatique et de recherche opérationnelle
{hamelphi,woodsean,eckdoug}@iro.umontreal.ca

ABSTRACT

We present and compare several models for automatic identification of instrument classes in polyphonic and poly-instrument audio. The goal is to be able to identify which categories of instrument (Strings, Woodwind, Guitar, Piano, etc.) are present in a given audio example. We use a machine learning approach to solve this task. We constructed a system to generate a large database of musically relevant poly-instrument audio. Our database is generated from hundreds of instruments classified in 7 categories. Musical audio examples are generated by mixing multi-track MIDI files with thousands of instrument combinations. We compare three different classifiers : a Support Vector Machine (SVM), a Multilayer Perceptron (MLP) and a Deep Belief Network (DBN). We show that the DBN tends to outperform both the SVM and the MLP in most cases.

1. INTRODUCTION

Thanks in part to the vast amount of music available online, much research has been done on the automatic extraction of descriptors for music audio, such as genre, artist, mood and instrumentation. Because the majority of this research has focused on commercial recorded music, where ground truth is lacking, relatively little work has been done in identifying which instruments are playing in music audio. Solving this problem would give rise to better description of commercial audio collections. It could also form a part of a system able to synthesize music with timbres that match the instruments found in a particular audio file (e.g. "generate music that sound like this Sex Pistols mp3"). Such a system may be useful in applications such as user-content-guided video game music generation.

In this paper, our focus is on constructing a model able to determine which classes of musical instrument are present in a given musical audio example, without access to any information other than the audio itself. In order to obtain sufficient labeled training examples for good generalization, we generated our own database of audio. Our goal was to have enough variability in the set of instruments so as to allow us to generalize to instruments not used in the training set. An overview of our system is illustrated in Figure 1.

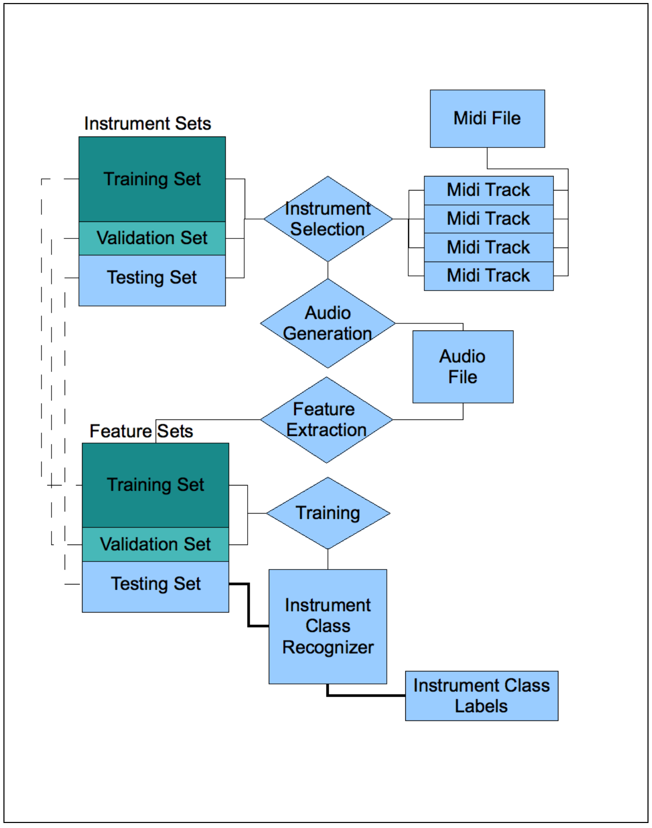


Figure 1. Overview of our automatic instrument class recognizer model

We compared three different classifiers to solve this task : a Support Vector Machine (SVM), a Multilayer Perceptron (MLP) and a Deep Belief Network (DBN).

The main contribution of this paper is the introduction of the DBN model to the instrument recognition task. Until recently, deep neural networks (i.e. networks having many hidden layers) were not used in practice because they are hard to train using random initialization and gradient descent alone. Recent developments have made training such networks possible [3, 15]. DBNs have since shown potential in many fields such as image and speech recognition.

Deep networks aim at learning higher-level features at each layer from the features of the layer below. Learning such high-level features allows a model to construct an abstract representation of the inputs. This is similar to how the human brain transforms raw sensory inputs to abstract features. An in-depth description and justification of the use of deep architectures for learning can be found in [2].

The paper is organized as follows : In Section 2, we describe previous research in the domain of instrument recognition. In Section 3 we describe the system used to generate our audio database. In Section 4, we discuss the features extracted from the audio. In Section 5, we describe in detail the three classification models we employed. We then discuss our results in Section 6.

2. PREVIOUS WORK

The problem of automatic instrument classification has been tackled from several different angles in the past decades. Psycho-acoustic studies have been conducted to build "timbre spaces" in which the distance between two sounds represents their degree of similarity [24, 30]. From the topology of these spaces, we can outline some important features of the sounds that are important when studying timbre (e.g. spectral centroid, spectral flux, etc.). A lot of work on instrument recognition has been done on isolated instrument sounds and monophonic audio [1,8–11,17,22,23,27]. An overview of previous approaches to automatic instrument classification is described in [13]. Recent work deals with more complex and musically relevant sounds such as duets and polyphonies [6,7,16,18–20]. There has also been much work done on the related task of predicting genre- and instrument-related tags from audio [4, 21, 26] for music recommentation.

In a polyphonic context, notes are not easily separable. Pitch tracking and source separation techniques can be useful to address this problem, but these are still unsolved problems in polyphonic audio that attracts a lot of research activity. The problem of instrument recognition becomes even more complex when we consider multi-instrument audio. Many different machine learning models have been tried to solve this task. In [6], missing feature theory and a Gaussian-mixture model (GMM) classifier was used to identify instruments in monophonic and polyphonic audio. In [20], a process using linear discriminant analysis (LDA) for instrument recognition for solo and duet performances is presented. A Support Vector Machine (SVM) and a hierarchical classification scheme are used on polyphonic music in [7]. [16] presents a classification model using LDA and feature weighting using polyphonic audio and musical context information. In [18], instrument recognition in polyphonic audio is done by applying a post-process on a mid-level harmonic atoms representation. [19] compares the performance of three classifiers : SVMs, Extra Trees and K-Nearest Neighbors.

Unfortunately, the relative performances of these different approaches are difficult to measure. Since each research team uses a different test database and a different classification taxonomy, it would be unfair to compare the reported classification accuracies.We take a small step towards addressing this by publishing our entire instrument database. See [12].

Most previous work on instrument recognition in polyphonic audio has focused on recognizing specific instruments from a small set of instruments. These models do not attempt to deal with instruments they have never heard before. In this paper we address this limitation by introducing a model capable of recognizing *classes* of instruments instead of specific ones. We argue that this behavior is better suited to large, rapidly-evolving commercial audio databases.

3. DATABASE GENERATION

To solve a difficult task such as instrument class recognition in poly-instrument audio, we require a large database with a lot of variability in the data. To address this challenge, we constructed our own database with a wide range of sampled and synthesized instruments. Using MIDI files to control our instruments, we were able to easily generate musically plausible examples with a wide range of velocities, harmonization, and note lengths. We believe this will help our classifier to better generalize.

3.1 Instrument bank

We used the instrument sounds from the commercial sampler "Kontakt 3" from "Native Instruments" to generate our database. The advantage of using a sampler instead of banks of isolated sounds or recorded performances is that we can generate musically relevant audio files from any MIDI file. We selected 172 different physical instruments. For 23 of these physical instruments, we treated different dynamics as individual instruments for a total of 320 instruments. We separated our instruments into 7 classes : Piano, Guitar, Bass, Organ, Woodwind, Brass and Strings. Each class contained from 9 to 94 instruments. To test for generalization we divided our instrument bank into three independent sets : 50% of the instruments were placed in a training set, 20% in a validation set and the remaining 30% in a test set.

3.2 Audio Generation

We built a system to automatically generate a large quantity of audio files using MIDI files and a bank of instrument samples. We generated two audio corpuses using solo instrument and poly-instrument MIDI files. We composed and pre-mixed six to seven 30-second midi files per experiment. The first corpus was obtained by generating audio from the solo instrument MIDI files, while the second was generated with multi-track MIDI files. We separated the MIDI files into individual tracks, each file having between two and six tracks. For each track, we randomly selected an instrument with a compatible range. By "compatible range", we mean that the chosen instrument has a sample set with a wide enough range (e.g. C4–C6) to actually play all of the notes in the file. We then mixed the tracks, making sure that all instruments in a mix were from the same data set (train, valid or test). We generated audio from each MIDI file with hundreds of different instruments mixes.

Examples of our generated audio and corresponding model predictions are available at the website :
`http://www.iro.umontreal.ca/~gamme/ismir_2009/`

4. FEATURE EXTRACTION

The selection of features is a crucial aspect of any instrument classifier. One of the most widely used features for timbre analysis is the Mel-Frequency Cepstral Coefficients (MFCCs). We used the 20 first MFCCs as well as their first and second derivatives (dMFCCs and ddMFCCs). The MFCCs were calculated on 32 ms windows with a window step size of 10 ms.

We also used a set of spectral features : centroid, spread, skewness, kurtosis, decrease, slope, flux and roll-off. The mathematical definitions of these features are described in [25].

We divided the audio files into 1 second frames, and calculated the mean and the standard deviation of each feature for each frame, yielding two values for each feature. In total, the feature vectors contain 136 values : 40 MFCCS, 40 dMFCCs, 40 ddMFCCs and 16 spectral features.

5. MODELS

We tested three different classifiers : a Mulitlayer Perceptron (MLP), a Support Vector Machine (SVM) and a Deep Belief Network (DBN).

5.1 Multilayer Perceptron

The first model is a single hidden layer feed-forward neural network, also know as Multilayer Perceptron. An advantage of such models is that they are very fast to use, once trained, making them good candidates for a real-time application. We used the neural network implementation from the publicly available PLearn library [28].We used a $tanh$ activation function for the hidden layer, and a logistic sigmoid function for the output layer. We used cross-entropy as the cost function to optimize. To avoid overfitting, we used an L2 norm regularization on the weights as well as an early stopping condition. We also used conjugate gradient descent to accelerate training.

5.2 Support Vector Machine

We also tested a Support Vector Machine (SVM) with a radial basis kernel. SVMs are widely used large margin classifiers. The implementation of a SVM is quite complex, but publicly available ready-to-use libraries make them rather simple to use [5]. SVMs have been used for the task of instrument recognition with a good degree of success [19]. SVMs have the advantage of having fewer hyper-parameters to optimize than neural networks. We used cross-validation to optimize the hyper-parameters.

5.3 Deep Belief Network

A deep network is constructed by superposing many layers of neurons. It is essentially an MLP with many hidden layers. The main difference comes from the initialization of the weights of the connections between neurons. In the single hidden layer case, a random initialization is generally sufficient for the gradient descent to work. However, with random initialization on many hidden layers, the solutions obtained appear to correspond to poor solutions that perform worse than the solutions obtained for networks with 1 or 2 hidden layers [2, 3]. To circumvent this problem, the DBN learning procedure consists of a greedy layer-wise unsupervised pre-training phase, followed by a supervised gradient descent fine-tuning phase. The pre-training phase configures the network such that it may efficiently represent the input data. The pre-training phase is typically done with layers of Restricted Boltzmann Machines (RBMs) [15] or autoencoders [14, 29]. In this work, we used RBMs. RBMs are constituted of two layers of neurons : a visible layer and a hidden layer. Each neuron is connected to every neuron of the other layer, but have no connection with neurons of the same layer. The RBMs have a simple and fast learning algorithm that basically try to minimize the reconstruction error using an algorithm called contrastive divergence [15]. We can stack many RBMs on top of each other, where the visible layer of the top RBMs is the hidden unit of the RBM below, to obtain a DBN. The pre-training phase then consists of training each RBM sequentially, starting from the input layer up to the output layer. Once this is completed, the model is further "fine tuned" for a specific supervised learning task. This fine tuning is done using the same gradient descent learning asn an MLP : given a cost function to optimize, the gradient is propagated through the network, and weights are updated accordingly. As for our MLP model, we used a cross-entropy cost function. One problem with DBNs is the large number of hyper-parameters : number of layers, number of units per layer, pre-training learning rate, gradient descent learning rate, weight regularization constant, number of pre-training epochs. This makes the hyper-parameter search tedious.

6. EXPERIMENTS

6.1 Experimental Setup

As in [19], we used weak labels as targets for training, i.e. targets for every frame in a given song are the same. If a song contains a string instrument and a guitar, every frame of that song will be labelled as containing 'strings' and 'guitar', even though there is no guarantee that there is a string instrument and a guitar in every frame.

The task of instrument class recognition in poly-instrument audio is a multi-label classification task, i.e. each instrument class may be present or not, and the classifier is unaware of how many classes are present.

In order to compare the three different models, we used the F-Score as a performance measure. The F-Score is a measure that balances precision and recall. The precision and recall are defined as

$$\text{Precision} = \frac{tp}{tp + fp}, \text{Recall} = \frac{tp}{tp + fn} \quad (1)$$

where tp, fp and fn are the number of 'true positives', 'false positives' and 'false negatives' examples. A true positive is a positive example that was correctly labeled as positive by the model. A false positive is a negative example that was mislabeled as positive. A false negative is a positive example that was mislabeled as negative. The F-Score (F) is defined as the harmonic mean of the precision and the recall

$$F = \frac{2(\text{precision} * \text{recall})}{\text{precision} + \text{recall}} \quad (2)$$

This can be simplified to

$$F = \frac{2tp}{2tp + fp + fn}. \quad (3)$$

To obtain F-Scores for each instrument, we calculated the F-Scores independently as for 7 independent classification tasks. In order to get a global F-score that represents the overall performance of the models, we took the sum of tp, fp and fn over all the instruments.

The neural networks (MLP, DBN) output a probability $\in [0, 1]$for each instrument, representing the network's belief that the instrument is present in the given input frame. If the probability for a given instrument is higher than a given threshold, we classify this class as being present. Lowering the threshold improves the recall, but lowers the precision, while increasing the threshold has the opposite effect. To label a whole song, we take the mean of the probabilities from each frame and apply a threshold to decide whether or not each instrument class is present. We optimized the threshold to maximize the global F-score.

The output of our SVM model is binary (0 or 1) for each class. We used a similar technique as the neural network to label a song, except that we have binary votes instead of probabilities.

6.2 Results and Discussion

6.2.1 Feature sets

To confirm that the features we extracted from the audio were useful for training our models, we compared the results of training with subsets of our feature sets on the solo instrument audio corpus. The mean F-score for each subset using our three models are shown in Table 1. We see a tendency that using more features helps the SVM and the DBN, but the MLP doesn't show improvement with the full set of features compared to using only 20 MFCCs. Another result that is remarkable is that the DBN performs surprisingly well compared to the two other models with only the spectral features as inputs. For the following experiments, we will always use our full set of features.

	SVM	MLP	DBN
Spectral Features (16)	0.51	0.74	0.81
12 MFCCs (72)	0.75	0.85	0.85
20 MFCCs (120)	0.81	0.86	0.87
All Features (136)	0.84	0.84	0.88

Table 1. Global F-score for different features subsets (features vector length in parenthesis)

6.2.2 Solo instrument audio

Our first audio corpus contains solo performances from all the instruments. For this experiment, we generated a total of 2735 song examples generated from 7 different MIDI files. We used 1984 of these for training and validation, for a total of 62434 1-second frames. The results are shown in Table 2.

	SVM	MLP	DBN	%
Bass	0.88	0.88	0.88	13.85%
Brass	0.87	0.88	0.91	22.37%
Guitar	0.0	0.0	**0.21**	2.13%
Organ	0.96	0.89	0.96	7.46%
Piano	0.45	0.43	**0.57**	6.39%
Strings	0.94	0.95	0.97	9.59%
Woodwind	0.82	0.85	0.89	29.83%
Global	0.84	0.84	0.88	

Table 2. F-score for solo instrument audio. The results that clearly outperforms the other models are highlighted in bold. The percentage of positive examples in the training set for each instrument is shown in the rightmost column

We see that the DBN tends to perform better than both the SVM and the MLP in this experiment. Moreover, the DBN seems to perform significantly better when the quantity of positive training example is smaller. Note that both the SVM and the MLP were unable to recognize the guitar instrument class. This is probably related to the fact that only a small fraction of the data set contained positive guitar examples.

The DBN that gave the best validation F-score had 5 layers of 50 units each. Only 3 epochs of pre-training over the training set were necessary to achieve the best generalization performance. The best MLP model had 40 hidden units.

6.2.3 Poly-instrument audio

Our second audio corpus is constructed from mixes of instruments. Each song is generated from one of 6 MIDI files containing between 2 and 6 tracks, and thus each example contains from 1 to 6 classes (many instruments from the same class are allowed). The data set is constituted of 3654 training and validation examples divided in 186532 frames. Results are shown in Table 3.

Again, in this experiment, the DBN seems to perform slightly better than the SVM and the MLP. In three cases (brass, guitar and woodwind), the performance difference was important. The DBN with the best generalization performance in this experiment had 4 layers of 100 units and

	SVM	MLP	DBN	%
Bass	0.86	0.83	0.85	50.00%
Brass	0.38	0.45	**0.63**	25.90%
Guitar	0.05	0.15	**0.28**	11.94%
Organ	0.84	0.84	0.85	62.99%
Piano	0.83	0.80	0.83	64.44%
Strings	0.37	0.37	0.36	18.82%
Woodwind	0.31	0.41	**0.52**	31.81%
Global	0.72	0.72	0.74	

Table 3. F-score for poly-instrument audio. The results that clearly outperforms the other models are highlighted in bold. The percentage of positive examples in the training set for each instrument is shown in the rightmost column

required 4 epochs of pre-training. The best MLP was constructed with 60 hidden units.

6.3 Discussion

In all 3 experiments, the DBN generally performed better than the 2 other models, although the difference is not always important. The DBN tends to perform better especially in cases where the quantity of positive examples is small. This could indicate that the DBN was able to learn higher-level features to discriminate instrument classes. In other words, it was able to use what it learned from other instrument classes to discriminate instruments that were less frequent.

Although the results seem to show that the DBN performed better than the SVM and MLP, we cannot draw any hard conclusion with these results because of the similarity of the results and the lack of confidence intervals. The F-Score may not be the best measure to get such confidence intervals. However, these results clearly show that DBNs can be useful for the task of instrument recognition. These results also motivate more experiments to confirm the tendency shown. In future work, these experiments should be run using cross-fold testing and measuring the classification error in order to obtain a reliable confidence measure.

When generating our labeled examples, we tried to stay as close to real music as possible. The MIDI format is good to reproduce some features of real music such as harmonization and timing. However, it is harder to represent musical features such as expressiveness and instrument dynamics variations in MIDI. Also, our system used a rather simple fixed mixing of the instruments in a given song, which gave rise to small variability in the relative volume of the instruments. The limited number of midi files we used is also a limitation of our model. In future work, we would like to add more variability to the music generation by using more songs and by diversifying the mixing between instruments.

Another aspect that could improve the performance of the three models would be to learn an independent decision threshold for each instrument class. We used only one decision threshold that was optimized on the validation set global F-Score. This may be related to the fact that the SVM and the MLP were unable to recognize the guitar class in the solo instrument experiment.

7. CONCLUSION AND FUTURE WORK

In this work, we have introduced the DBN model for instrument recognition. We have shown that DBNs perform at least as well as SVMs and MLPs for this task. We have also shown that the DBN tends to outperform these models when the feature set is limited, and when the number of positive examples for a class is limited. These results motivate the application of deep networks in music information retrieval tasks.

As seen in Section 4, adding more relevant features seems to improve the performance of the classifiers. In future work, it would be interesting to consider extracting a wider variety of features from the audio. In this study, we avoided harmonic features that rely on the identification of a single fundamental frequencey for a frame of audio because this is ill-defined in the polyphonic case. In future work, it would be interesting to test if extracting simple harmonic features (e.g. odd to even harmonics ratio) from mixed instruments using an estimate of the most salient frequency could help for this task. We suppose that there is useful information in such features.

We also plan to add more variability to our data set by adding reverb and background noise to our audio examples. We hypothesize that this would add robustness to our trained models.

Finally, it would be interesting to test our model on real music. This is something that we plan for the near future. To test our model on commercial music, we would need to train a wider range of instruments, such as drums, distorted guitars, vocals, etc.

8. ACKNOWLEDGMENTS

Philippe Hamel was supported financially by FQRNT. Douglas Eck and Sean Wood are financed by NSERC Discovery Grants. Special thanks to Nathanael Lecaude, Pascal Lamblin, Simon Lemieux, Olivier Delalleau, and all the people at the GAMME and LISA labs for helpful discussions.

9. REFERENCES

[1] G. Agostini, M. Longari, and E. Pollastri. Musical instrument timbres classification with spectral features. *EURASIP J. Appl. Signal Process.*, 2003 :5–14, 2003.

[2] Y Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, to appear, 2009.

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[4] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger : A model for predicting social tags from

acoustic features on large music databases. *Journal of New Music Research"*, 37(2) :115–135, 2008.

[5] C.-C. Chang and C.-J. Lin. *LIBSVM : a library for support vector machines*, 2001.

[6] J. Eggink and G.J. Brown. Application of missing feature theory to the recognition of musical instruments in polyphonic audio. In *International Symposium on Music Information Retrieval (ISMIR '03)*, 2003.

[7] S. Essid, G. Richard, and B. David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions On Audio, Speech And Language Processing*, 14 :68–80, 2006.

[8] S. Essid, G. Richard, and B. David. Musical instrument recognition by pairwise classification strategies. *IEEE Transactions On Audio, Speech And Language Processing*, 14 :1401–1412, 2006.

[9] A. Fraser and I. Fujinaga. Toward realtime recognition of acoustic musical instruments. In *Proceedings of the International Computer Music Conference. 175–7.*, 1999.

[10] I. Fujinaga. Machine recognition of timbre using steady-state tone of acoustic musical instruments. In *Proceedings of the International Computer Music Conference. 207-10.*, 1998.

[11] I. Fujinaga and K. MacMillan. Realtime recognition of orchestral instruments. In *Proceedings of the International Computer Music Conference. 141–3.*, 2000.

[12] P. Hamel, S. Wood, S. Lemieux, and D. Eck. The GAMME Poly-Instrument Audio Database, 2009. `http://www.iro.umontreal.ca/~gamme/instrument_data/`.

[13] P. Herrera, A. Klapuri, and M. Davy. *Signal Processing Methods for Music Transcription*, chapter Automatic Classification of Pitched Musical Instrument Sounds, pages 163–200. Springer US, 2006.

[14] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313 :504–507, 2006.

[15] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006.

[16] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Instrument identification in polyphonic music : feature weighting to minimize influence of sound overlaps. *EURASIP J. Appl. Signal Process.*, 2007(1) :155–155, 2007.

[17] A. G. Krishna and T. V. Sreenivas. Music instrument recognition : from isolated notes to solo phrases. In *in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '04)*, volume 4, pages 265–268, Montreal, Quebec, Canada, May 2004.

[18] P. Leveau, D. Sodoyer, and Daudet L. Automatic instrument recognition in a polyphonic mixture using sparse representations. In *ISMIR*, 2007.

[19] D. Little and B. Pardo. Learning musical instruments from mixtures of audio with weak labels. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008.

[20] A. Livshin and X. Rodet. Musical instrument identification in continuous recordings. In *Proc. of the 7th Int. Conference on Digital Audio Effects (DAFX-04), Naples, Italy*, 2004.

[21] M. I. Mandel and D. P.W. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 594–599, 2005.

[22] J. Marques and P. J. Moreno. A study of musical instrument classification using gaussian mixture models and support vec- tor machines. Crl technical report series crl/4, Cambridge Research Laboratory, Cambridge, Mass, USA, 1999.

[23] K Martin. *Sound-source recognition : A theory and computational model*. PhD thesis, MIT, 1999.

[24] S. McAdams. Psychological constraints on form-bearing dimensions in music. *Contemporary Music Review*, 1989.

[25] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, IRCAM, 2004.

[26] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech And Language Processing*, 16(2) :467–476, 2008.

[27] E. Vincent and X. Rodet. Instrument identification in solo and ensemble music using independent subspace analysis. In *ISMIR*, 2004.

[28] P. Vincent, Y. Bengio, N. Chapados, et al. Plearn. `http://plearn.berlios.de/`.

[29] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML '08 : Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, New York, NY, USA, 2008. ACM.

[30] D. L. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, Vol 3 No 2, 1979.

USING REGRESSION TO COMBINE DATA SOURCES FOR SEMANTIC MUSIC DISCOVERY

Brian Tomasik, Joon Hee Kim, Margaret Ladlow, Malcolm Augat,
Derek Tingle, Richard Wicentowski, Douglas Turnbull
Department of Computer Science, Swarthmore College, Swarthmore PA 19081
{btomasi1, joonhee.kim, mladlow1, maugat1}@alum.swarthmore.edu
{dt, richardw, turnbull}@cs.swarthmore.edu

ABSTRACT

In the process of automatically annotating songs with descriptive labels, multiple types of input information can be used. These include keyword appearances in web documents, acoustic features of the song's audio content, and similarity with other tagged songs. Given these individual data sources, we explore the question of how to aggregate them. We find that fixed-combination approaches like sum and max perform well but that trained linear regression models work better. Retrieval performance improves with more data sources. On the other hand, for large numbers of training songs, Bayesian hierarchical models that aim to share information across individual tag regressions offer no advantage.

1. INTRODUCTION

We are interested in developing a *semantic music discovery engine* in which users enter text queries and receive a ranked list of relevant songs. This task requires a *semantic music index*, i.e., a mapping between songs and associated tags. A *tag*, such as "afro-cuban roots," "heavy metal," or "steel-string guitar," is a short text token which describes some meaningful aspect of the music (e.g., genre, instrumentation, emotion, geographical origins). In this paper, our goal will be to compute a real-valued score $\widehat{y}_{st}$ that expresses how strongly tag t applies to song s.

There are a number of ways to collect semantic annotations of music. [1] compare five such approaches: surveys, social tagging, games, web documents, and audio content. Each of these data sources offers a different perspective, and each has its own strengths and weaknesses (e.g., scalability, popularity bias, accuracy), so we may wish to collect information from several of them. The question then becomes how to combine that information into a single score for use in our semantic index.

In Section 2, we describe three sources of music information that we have collected: text mining web documents, content-based audio analysis, and collaborative filtering. Section 3 describes various approaches for combining these sources, including simple fixed rules, as well as a trained *regression* model in which combination weights depend on the quality and sparsity of the input data. We explore both ordinary linear and logistic regression, as well as Bayesian hierarchical models that aim to share information across tags. Section 4 describes our experimental setup, which includes a ground-truth corpus of 10,870 songs for two vocabularies (71 Genre tags and 151 Acoustic tags) collected from Pandora's Music Genome Project. [1] Section 6 concludes.

2. MUSIC INFORMATION SOURCES

We collect semantic-annotation information from three sources: web documents (WD), content-based audio analysis (CB), and collaborative filtering (CF). For each song s and tag t, we use these sources to generate scores—denoted x_{st}^{WD}, x_{st}^{CB}, and x_{st}^{CF}, respectively—indicating how well t describes s.

2.1 Web Documents

Tags that appropriately describe a song will tend to appear in association with the song's name in natural-language text documents. We exploit this fact by downloading from the web pages that describe the song and counting how often the proposed tag appears within them.

Given a song s, we generate a database D_s of documents by querying Google for "song name" "artist name" in lower-case (e.g., "enjoy the silence" "depeche mode"). We download all hits in the top 10 and clean the HTML files into raw text. This was done for a total of 9,359 songs. Then, for each tag t, we compute

$$x_{st}^{\text{WD}} = \sum_{d \in D_s} \frac{n_{td}}{N_{td}},$$

where n_{td} is a measure that roughly expresses how many times t actually appeared in document d, and N_{td} is the number of times t *could have* appeared in d. N_{td} is just

[1] See http://www.pandora.com/mgp.shtml

$|d| \,/\, |t|$, the number of words in d divided by the number of words in t. n_{td} is a bit more complicated. For long tags, such as "call and answer vocal harmony (antiphony)," positional searches for the entire phrase would not work well. On the other hand, searching for the appearance of any of the words in t would yield too many hits. We compromise by computing n_{td} as the minimum number of hits for any word, taken over all words in t. In the case when the words in t appear in d only in the correct order, n_{td} will in fact be equal to the number of occurrences of the full phrase t.

2.2 Content-Based Audio Analysis

A second potential source of semantic information about a song is the audio content itself. For this purpose we use the supervised multiclass labeling (SML) model recently proposed by [2].

The audio track of a song is represented as a bag of feature vectors $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where each $\mathbf{x}_i$ is a feature vector that represents a short-time segment of audio, and T depends on the length of the song. We use the expectation maximization (EM) algorithm to learn a *song-specific* Gaussian mixture model (GMM) distribution over each $\mathcal{X}$. Then, for each tag in our vocabulary, we learn a *tag-specific* GMM using the Mixture Hierarchies EM algorithm [3]. This algorithm combines the set of song-specific GMMs for all the songs that have been associated with the tag. Given a novel song s, we compute the likelihood that its bag of feature vectors $\mathcal{X}_s$ would have been generated by each of the tag GMMs. Normalizing these likelihoods using the technique described in [2] yields our set of scores x_{st}^{CB}, which can be interpreted as the parameters of a multinomial distribution over the vocabulary of tags.

We use the popular Mel frequency cepstral coefficients (MFCCs) as our audio feature representation since it was incorporated into all of the top performing autotagging systems in the 2008 MIREX tag classification task [2, 4–6]. MFCCs are loosely associated with the musical notion of timbre ("color") of the music because they are a low-dimensional representation of the frequency spectrum of a a short-time audio sample. For each monaural song in the data set, sampled at 22,050 Hz, we compute the first 13 MFCCs for each half-overlapping short-time ($\sim$23 msec) window from 6 five-second clips spaced at uniform intervals over the length of the song. Over the time series of audio segments, we calculate the first and second instantaneous derivatives (referred to as *deltas*) for each MFCC. This results in about 5,000 39-dimensional MFCC+delta feature vectors per 30 seconds of audio content. We summarize an entire song by modeling the distribution of its MFCC+delta features with a 4-component GMM. We model each tag with an 8-component GMM.

2.3 Collaborative Filtering

One additional source of semantic information is user playlists: If two songs appear together in a large number of listener collections, one possible reason is that the songs share certain attributes (say, "punk influences") that the listeners enjoy. This suggests the idea of *tag propagation*: Find songs that tend to co-occur in playlists, and transfer tags from one of them to the other. A more robust approach is to find the collection of k songs ($k = 32$ here) that have the strongest co-occurence score with a given song s. For each tag t, we take the association x_{st}^{CF} of s with t to be the fraction of those 32 songs to which t applies. We set this number to 0 if the fraction is below a threshold of 0.3. The reasons for these choices, as well as further details on the entire data-collection process and choice of tag sets, appear in [7].

Our data consist of 400,000 user music libraries from last.fm, where a *library* is taken to be the set of items that a user listens to at least 1% of the time. It turns out that data at the song level is too sparse to generate meaningful co-occurence statistics, so we instead work at the artist level. We say that a tag applies to an artist if the tag applies to any of that artist's songs. At the end of the propagation process, we transfer an artist's score for a tag to each of its songs. We find the 32 closest artists using the following similarity score. Between artists i and j, we take

$$\mathrm{sim}(i, j) = \frac{p(i, j)}{\sqrt{p(i)p(j)}},$$

where $p(i, j)$ is the fraction of all artist co-occurrences represented by artists i and j, and $p(i)$ is the fraction of all co-occurrences containing artist i.

3. COMBINING METHODS

Given the data sources described in Section 2, how can we aggregate them? This general question has been well studied and is known variously as *combining expert judgments* (e.g., [8, 9]), *multi-sensor data fusion* (e.g., [10]), *information fusion* (e.g., [11]), or *combining classifiers* (e.g., [12, 13]). Rather than reviewing the entire body of literature on the subject, we focus on two of the most basic approaches: Fixed-combination rules and trained combiners, specifically regression.

3.1 Fixed Combiners

Fixed combining rules take the output score $\widehat{y}_{st}$ to be a simple function of the input scores: e.g., max, min, median, sum, or product [14, sec. 3]. Usually the input scores x_{st}^{i}, with $i \in \{\mathrm{WD}, \mathrm{CB}, \mathrm{CF}\}$, are calibrated so that they correspond to confidences or probabilities p_{st}^{i} that t applies to s given the source. This can be done, for instance, by standardizing the input scores to have mean 0 and variance 1 and then taking

$$p_{st}^{i} = \frac{1}{1 + \exp\left(-\alpha x_{st}^{i}\right)}$$

for some α [14, sec. 4.1]. We use $\alpha = 1$ in this paper.

A disadvantage of this technique, however, is that each source is treated on equal footing, when in fact, one of our sources may be far more trustworthy or better informed [14, sec. 1]. One method that overcomes this limitation is Bayesian Model Averaging (BMA) (e.g., [15]), which assumes that one of the data sources is the "correct" source and takes the final probability to be a weighted combination of the input probabilities:

$$p_{st}^{\text{all sources}} = \sum_{i \in \{\text{WD,CB,CF}\}} p_{st}^i p_i,$$

where p_i is the probability that source i is correct. As [16, sec. 1] point out, this assumption is often unrealistic, as the truth about whether a tag applies to a song needn't be captured by exactly one of our data sources. Still, the idea of taking our final score $\widehat{y}_{st}$ to be a weighted combination of the input scores—

$$\widehat{y}_{st} = \sum_{i \in \{\text{WD,CB,CF}\}} \beta_t^i x_{st}^i \tag{1}$$

for some weights β_t^i—does seem like a natural way to account for the differential predictive value of different inputs. The question is how to determine the weights.

3.2 Trained Combiners

If we have training data for a subset of songs, [2] the obvious answer is to use supervised learning. This is the *trained combiners* approach advocated in [14]. Indeed, (1) has the form of a linear-regression model, and we can determine the weights of the sources just by treating them as input features and computing their regression coefficients.

We try both linear and logistic regression, predicting the ground truth $y_{st} \in \{0, 1\}$ by the individual scores x_{st}^i, as well as an intercept and possibly other features of interest (see Section 3.4). We take our predicted values $\widehat{y}_{st}$ to be real-valued so that we can more finely rank-order songs than with 0/1 labels. Regression is a convenient combination approach because it potentially allows us to use a number of standard statistical tools: p-values for the significance of regression coefficients, prediction intervals for our output scores, model selection based on residual sum of squares, and many more advanced techniques.

3.3 Hierarchical Regression Models

One such technique is borrowing of information across tags. Each tag has its own regression model, but we might suspect that these models share significant structure: For instance, if collaborative filtering tends to be a highly predictive source, we would expect its coefficient to be consistently large. And the linear combination of sources that best predicts the tag "traditional country" is probably similar to the one that best predicts "contemporary country."

One way to capture this intuition is with a Bayesian hierarchical linear model (e.g., [17]). We'll illustrate this concept in the case of a single regression coefficient β_t for a single data source x_{st} without an intercept, but similar equations apply in the multivariate setting. Independent regression across the T tags assumes

$$y_{st} = \beta_t x_{st} + \epsilon_{st}, \;\; \epsilon_{st} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_t^2), \;\; t = 1, \ldots, T \tag{2}$$

for some variances σ_t^2, with no relationship among the β_t values. We call this the *Independent Linear* model. The *Independent Logistic* model is the same, except that y_{st} is replaced by the log-odds $\ln\left(\frac{p_{st}}{1-p_{st}}\right)$, where p_{st} is the probability that $y_{st} = 1$.

In a hierarchical model, we assume in addition to (2) that the β_t's share a common structure:

$$\beta_t = \overline{\beta} + v_t, \quad v_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2), \quad t = 1, \ldots, T. \tag{3}$$

For instance, if we had three tags with independent regression coefficients of 0.1, 0.2, and 0.3, it might be reasonable to suppose that $\overline{\beta} \approx 0.2$ with $\sigma \approx 0.1$. We can further assume a prior over $\overline{\beta}$ and perform Bayesian inference to estimate the parameters. The multivariate version of this model we call *Hierarchical Linear*, and the corresponding version in which y_{st} is replaced by the log-odds that $y_{st} = 1$ we call *Hierarchical Logistic*.

We might also assume that v_t in (3), rather than being normally distributed, is drawn from a mixture of normal distributions. For instance, perhaps the web-document source is much better at predicting genre labels than acoustic ones, so that its β_t values for genre tags cluster around 0.2, say, while its β_t values for acoustic tags cluster around 0.05. In that case, β_t could be modeled by taking $\overline{\beta} = 0.05$, with v_t having peaks at 0 and 0.15. We call this model *Mixture Linear*$_k$, where k is the number of centers. [3]

3.4 Regression Models

Equation (1) suggests the basic regression model to use, although in practice we include an intercept, which we find always to be highly statistically significant. We can also regress on just one or two of the main sources at a time.

A nice aspect of using regression is that we can include extra features in our model (assuming we expect they'll contribute useful information rather than meaningless noise that will lead us to overfit). In particular, we include *scrobble counts* from last.fm as a measure of the popularity of the artist who wrote the given song. If we suspected that more popular songs had more nonzero y_{st} values in our ground-truth, we would expect this popularity term to have a high positive regression coefficient. Including the term could be seen as a way of controlling for popularity bias if we omit the popularity feature when we predict $\widehat{y}_{st}$ for novel songs. We can also include terms for the interaction of data sources with popularity. A positive interaction coefficient would indicate that the data source gives a more confident prediction that a tag applies to a song when the song's artist is popular.

[2] Another possible scenario is that, rather than having ground-truth labels for a subset of our songs, we have data that applies to all of our songs but is weakly labeled, i.e., not every song that applies for a given tag is labeled as such. If our input data sources are less sparse, we can use them to "fill in zeros" in the ground truth while preserving the labels that the ground truth had already.

[3] See Chapters 3 and 5 of [18] for details on each of these three hierarchical models in a more general setting.

4. EXPERIMENTAL SETUP

4.1 Data Set

Our data set consists of 10,870 songs representing 19 top-level genres (e.g., rock, classical, electronic) and 180 subgenres (e.g., grunge, romantic period opera, trance). We have approximately 60 songs per subgenre. Each song is associated with one or more genres and one or more subgenres. For each song, we also attempt to collect between 2 and 10 acoustic tags from Pandora's Music Genome Project vocabulary. This vocabulary consists of over 1,000 unique tags like "dominant bass riff," "gravelly male vocalist," and "acoustic sonority." These acoustic tags can be thought to be *objective* in that two trained experts can annotate a song using the same tags with high probability [19].

4.2 Cross-Validation Setup

We evaluate the retrieval performance of our combined scores using five-fold cross-validation on the Pandora data set. Ordinarily, this would involve training our regression model on 4/5 of the data and testing on the remaining 1/5. However, we need to be careful here, because our content-based data source also trains on the Pandora data set. The danger is that the content-based system may overfit the training data, and because our regression model would be using the same training data, the model might overweight the content-based source. [14, sec. 5] notes this problem and suggests that it be addressed by dividing the training set into two parts, which we do as follows.

We divide the songs into five partitions, each with roughly 2,000 songs. We apply an *artist filter* to the partitions, with all of the songs by an artist appearing in a single fold, to avoid overfitting our model to the particular artists that appear in our training set. On three of the partitions we train the content-based system, using it to then obtain predictions for the songs in the remaining two. We use one of those partitions (roughly 2,000 songs) to train our regression model, which then makes its predictions on the final partition. We then cycle this process five times. The reason for the uneven split between the two training sets is that the content-based system needs to learn many more parameters than our regression model, which typically has at most five coefficients.

4.3 Tag Pruning

Some tags are labeled with too few songs to be useful for training when we divide the songs into five partitions, so we prune them. In particular, the content-based training considers only tags that have at least 20 positive instances in the ground truth over each possible set of three partitions on which to train. In addition, our regression model requires that each single partition have at least one positive ground-truth song (since it would be trivial to train a model when the y_{st}'s are all 0) and at least one positive song in each of the three main data sources. After pruning we are left with 71 Genre tags and 151 Acoustic tags.

4.4 Implementation Details

Regression works best when the features are roughly normally distributed, so we transform some of the input scores for this purpose. For popularity counts, which range anywhere from 1 to over 15 million, we apply a log transformation. For the web-document source, which is based on count data, we apply a square-root transformation [20, p. 84]. We then standardize each data source by subtracting the mean and dividing by the standard deviation for a given tag. The x^i_{st}'s referred to in Section 3.1 are these standardized values.

For a small number of tags, β^i_t was estimated as negative for one or two of the input data sources. Because we believe that our main three data sources, while potentially unhelpful, should not be anti-predictive of the ground truth, we eliminate negative coefficients by setting them to 0 when they occur. (Making this adjustment results in a small but statistically significant improvement in mean average precision and area under the ROC curve for both Genre and Acoustic tags.) We do allow popularity to have a negative coefficient, and we remove this restriction entirely when considering models with interaction terms.

4.5 Regression Types

We implement Independent Linear and Independent Logistic regression using the basic `lm` and `glm` functions of the `R` language. For the hierarchical regressions, we use the `bayesm` package [21], specifically the `rhierLinearModel`, `rhierBinLogit`, and `rhierLinearMixture` functions for Hierarchical Linear, Hierarchical Logistic, and Mixture Linear$_k$, respectively, with all optional parameters set to their default values. These methods use Markov chain Monte Carlo to sample the entire posterior distribution for the β^i_t's given the data, but we simply take our β^i_t estimate to be the average of these draws. Performance is good with as few as a few hundred samples, but we find that area under the ROC curve does not level off completely until 5,000 to 10,000 draws. For the results in this paper, we sample 15,000 draws, which takes on the order of 30 minutes with roughly 100 tags and 2,000 songs. A parameter sweep of the number k of means in the Gaussian-mixture prior showed no appreciable differences over the range 2 to 50, so we use $k = 2$ as the default.

5. RESULTS AND DISCUSSION

We assess performance using the four standard information-retrieval metrics listed in Table 1 (see [22, sec. 8.4] for explanation of each). We have also made available [4] a list of the top 5 predicted songs for each tag for purposes of qualitative evaluation.

[4] See `http://www.sccs.swarthmore.edu/users/09/btomasi1/combiner/`

Table 1. Area under the ROC curve, mean average precision, R-precision, and 10-precision for various settings described further in the text. Rows are ordered by average AUC for Genre tags. Means and standard errors are taken over the tags, applied to the averages of five-fold cross-validation. (To compute standard errors with respect to each individual CV fold, divide the reported standard errors by a further $\sqrt{5}$.) The data-source abbreviations are web documents (WD), collaborative filtering (CF), content-based analysis (CB), popularity (P), all three main sources in the model (All3), and interactions with each of the three main sources (I).

Regression Model								
	71 Genre Tags				151 Acoustic Tags			
	AUC	MAP	R-Prec	10-Prec	AUC	MAP	R-Prec	10-Prec
Random	0.502±0.003	0.09±0.01	0.08±0.01	0.08±0.02	0.508±0.003	0.032±0.003	0.030±0.003	0.03±0.00
WD	0.666±0.010	0.25±0.02	0.29±0.02	0.47±0.03	0.616±0.006	0.135±0.007	0.181±0.008	0.29±0.02
CF	0.732±0.010	0.45±0.02	0.45±0.02	0.72±0.04	0.641±0.008	0.154±0.010	0.213±0.011	0.25±0.02
CB	0.781±0.014	0.23±0.02	0.25±0.02	0.38±0.03	0.836±0.008	0.141±0.007	0.161±0.008	0.19±0.01
WD&CF	0.789±0.010	0.50±0.02	0.50±0.02	**0.74±0.04**	0.724±0.007	0.231±0.010	0.280±0.011	0.40±0.02
CB&WD	0.819±0.010	0.32±0.02	0.34±0.02	0.53±0.03	0.870±0.006	0.220±0.009	0.246±0.009	0.36±0.02
CB&CF	0.853±0.009	0.49±0.02	0.48±0.02	0.73±0.04	0.861±0.007	0.213±0.010	0.244±0.010	0.29±0.01
All3&P&I	0.856±0.007	**0.52±0.02**	0.50±0.02	**0.74±0.04**	0.860±0.006	0.262±0.010	0.288±0.010	0.40±0.02
All3	0.871±0.007	**0.52±0.02**	0.50±0.02	**0.74±0.04**	**0.888±0.006**	0.276±0.010	0.298±0.010	**0.42±0.02**
All3&P	**0.876±0.007**	**0.52±0.02**	**0.51±0.02**	**0.74±0.04**	0.887±0.006	**0.277±0.010**	**0.299±0.010**	**0.42±0.02**

Combination Method								
	71 Genre Tags				151 Acoustic Tags			
	AUC	MAP	R-Prec	10-Prec	AUC	MAP	R-Prec	10-Prec
Min	0.658±0.015	0.27±0.02	0.27±0.02	0.60±0.04	0.654±0.009	0.121±0.006	0.161±0.008	0.26±0.01
Product	0.826±0.009	0.42±0.03	0.41±0.02	0.67±0.04	0.814±0.006	0.197±0.008	0.232±0.009	0.32±0.01
Median	0.826±0.009	0.43±0.02	0.43±0.02	0.68±0.04	0.820±0.006	0.219±0.009	0.261±0.009	0.35±0.02
Sum	0.851±0.007	0.44±0.03	0.44±0.02	0.69±0.04	0.847±0.006	0.220±0.009	0.252±0.009	0.34±0.01
Max	0.856±0.007	0.46±0.02	0.48±0.02	0.59±0.03	0.859±0.006	0.239±0.009	0.274±0.009	0.34±0.01
Ind Log	0.866±0.006	0.51±0.03	0.50±0.02	0.72±0.04	0.875±0.005	0.266±0.010	0.293±0.010	0.40±0.02
Hier Log	0.872±0.006	0.51±0.03	0.50±0.02	0.73±0.04	0.883±0.006	0.272±0.010	0.296±0.010	0.40±0.02
Hier Mix	**0.876±0.007**	**0.52±0.02**	**0.51±0.02**	**0.74±0.04**	**0.887±0.006**	**0.277±0.010**	**0.299±0.010**	**0.42±0.02**
Hier Lin	**0.876±0.007**	**0.52±0.02**	**0.51±0.02**	**0.74±0.04**	**0.887±0.006**	**0.277±0.010**	**0.299±0.010**	**0.42±0.02**
Ind Lin	**0.876±0.007**	**0.52±0.02**	**0.51±0.02**	**0.74±0.04**	**0.887±0.006**	**0.277±0.010**	**0.299±0.010**	**0.42±0.02**

5.1 Regression Models

The top half of Table 1 reports the performance of the Independent Linear model on subsets of the data sources, as well as models that include popularity information. The Random method is a regression model in which all sources have coefficients of 0, so that the final ranking of songs is the same as the (randomized) order in which they were initially seen. Each source alone clearly performs better than random, and each addition of a new source results in a statistically significant improvement in AUC.[5] This is consistent with the fact that the data sources are relatively uncorrelated, having correlation coefficients typically less than 0.3 and often less than 0.1, depending on the tag.

According to the AUC measure, CB is the individually most predictive source, while according to precision, CF is. We suspect this reflects the fact that CB's input representation is dense, providing nonzero scores for 91.2% of songs for each tag, while CF's input contains mostly zeros, with scores for only an average across tags of 2.4% of songs. (WD falls in the middle, with nonzero scores for an across-tag average of 13.7% of songs.) When CF has a nonzero value, it really means something, so that CF's top results are very precise. Toward the later end of the ranked results list, however, CF is essentially random, while CB still provides useful information.

It is interesting to observe that CB's advantage over CF in terms of AUC is larger in the case of acoustic tags than genre tags, perhaps because acoustic tags are inherently more predictable by audio content alone.

Popularity data was not especially helpful. While its addition to the three main sources did result in a statistically significant AUC improvement for Genre tags (p-value 0.007), it did not for Acoustic tags (p-value 0.4), and the magnitude of difference was relatively small. In some sense, this is a welcome result, since it suggests that the Pandora labels are not biased very much by whether an artist is well-known. The interaction model contained too many features and tended to overfit, which is unsurprising given the modest usefulness of the main popularity term.

5.2 Coefficient Magnitudes

Our default regression model was Independent Linear with the three main data sources, popularity, and an intercept. Averaging the β_t^i's over all of the tags t gives the following prediction equation for Genre tags (the one for Acoustic tags is similar):

$$\widehat{y}_{st} = 0.08 + 0.02x_{st}^{\mathrm{WD}} + 0.02x_{st}^{\mathrm{CB}} + 0.09x_{st}^{\mathrm{CF}} + 0.02x_{s}^{\mathrm{pop}}.$$

[5] This is usually apparent from inspection of standard errors, but we verify it by checking that p-values are less than 0.05 for paired t-tests on the per-tag AUC values. In fact, the only pairs between which this fails to hold are (1) CB and WD&CF for Genre tags, (2) All3 and All3&P for Acoustic tags, and (3) CB&CF and All3&P&I for both tag types. If we apply a conservative Bonferroni correction for the $\frac{10\cdot 9}{2}$ pairs of tests, a few more pairs become not significant, including the transition from CB&CF to All3 for Genre tags.

Because the x_{st}^i's represent the transformed and standardized input values (see Section 4.4), the standard error for each β_{st}^i is roughly the same for a given tag, [6] so that the t-statistic of each coefficient is roughly proportional to the coefficient's magnitude. It's worth noting, though, that statistical significance of a coefficient as different from zero is not identical with usefulness as a data source. Indeed, we saw in Section 5.1 that CB was individually more predictive than CF, at least as measured by AUC, while CB's coefficient is 0.02 instead of 0.09. The reason may again be that CB provides a denser input representation than CF; CF can afford to have a large β_{st}^{CF} because in the rare cases when its values are nonzero, they're strongly informative.

5.3 Regression Types

The bottom half of Table 1 shows various combination techniques. The regression approaches use the model All3&P, while the fixed-combination approaches use just the three main sources. All trained regression models outperform all fixed-combining methods. [7] This result contrasts with the finding by [11] that the simple sum rule outperformed supervised linear-discriminant analysis (similar to logistic regression) and decision trees. Still, Sum and especially Max do not fare badly and would not be unreasonable choices for a simple combining system. That Max is close to Independent Logistic regression is perhaps unsurprising, because the fixed-combining methods apply the same sigmoid transformation to the input data that logistic regression uses.

While Hierarchical Logistic regression did slightly outperform Independent Logistic, the hierarchical and mixture models showed no apparent effect for linear regression. We suspect this is because the number of observations (songs) is so large (over 2,100 on average) that the Bayesian prior terms in those models wash out. To confirm this, we tried artificially restricting ourselves to 250 songs, and in that case, the hierarchical methods did slightly outperform their independent counterparts.

6. CONCLUSIONS

We have shown that combining different sources of song-tag annotation information improves retrieval performance. Fixed-combining methods like Sum and Max do a fine job for simple systems, but retrieval improves when we use a trained combining method like linear or logistic regression. In settings where large numbers of songs are available, basic Independent Linear regression on each tag separately gives results just as good as more sophisticated hierarchical models, while allowing for easier implementation, faster computation, and greater parallelizability.

[6] This is only "roughly" because of small inter-feature correlations.

[7] Paired t-tests on the AUC values for individual tags give p-values less than 0.05 for all pairs except between (1) Product and Median and (2) Sum and Max for Genre tags, and (3) all three of Hier Mix, Hier Lin, and Ind Lin for both tag types. For Genre tags, five more pairs fail to reject the null hypothesis if we apply a Bonferroni correction on the $\frac{10 \cdot 9}{2}$ pairs of tests, including Sum vs. Independent Logistic (p-value 0.01).

7. REFERENCES

[1] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. *ISMIR*, 2008.

[2] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 2008.

[3] N. Vasconcelos. Image indexing with mixture hierarchies. *IEEE CVPR*, pages 3–10, 2001.

[4] S. J. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 2008.

[5] M. Mandel and D. Ellis. Multiple-instance learning for music information retrieval. In *ISMIR*, 2008.

[6] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *NIPS*, 2007.

[7] J. Kim, B. Tomasik, and D. Turnbull. Using artist similarity to propagate semantic information. *ISMIR*, 2009.

[8] P. A. Morris. Combining expert judgments: A Bayesian approach. *Management Science*, pages 679–693, 1977.

[9] R. A. Jacobs. Methods for combining experts' probability assessments. *Neural computation*, 7(5):867–888, 1995.

[10] H. B. Mitchell. *Multi-Sensor Data Fusion: An Introduction*. Springer, 1 edition, September 2007.

[11] A. Ross and A. Jain. Information fusion in biometrics. *Pattern Recognition Letters*, 24(13):2115–2125, 2003.

[12] J. Kittler. Combining classifiers: A theoretical framework. *Pattern Analysis & Applications*, 1(1):18–27, 1998.

[13] C. De Stefano, C. D'Elia, A. Marcelli, and A. S. di Freca. Using bayesian network for combining classifiers. In *ICIAP*, pages 73–80, 2007.

[14] R. Duin. The combining classifier: To train or not to train? In *ICPR*, volume 16, pages 765–770, 2002.

[15] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: a tutorial. *Statistical science*, pages 382–401, 1999.

[16] Z. Ghahramani and H. C. Kim. Bayesian classifier combination. *Gatsby Computational Neuroscience Unit Tech Report*, 2003.

[17] D. V. Lindley and A. F. M. Smith. Bayes estimates for the linear model. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–41, 1972.

[18] P. E. Rossi and G. M. Allenby. Bayesian statistics and marketing. *Marketing Science*, pages 304–328, 2003.

[19] T. Westergren. Personal notes from Pandora get-together in San Diego, March 2007.

[20] J. J. Faraway. *Practical Regression and ANOVA using R*. July 2002. `http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf`.

[21] P. E. Rossi and R. McCulloch. bayesm. R package ver. 2.2-2, 2008-06-09, `http://faculty.chicagobooth.edu/peter.rossi/research/bsm.html`.

[22] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

LYRIC TEXT MINING IN MUSIC MOOD CLASSIFICATION

Xiao Hu **J. Stephen Downie** **Andreas F. Ehmann**

International Music Information Retrieval Systems Evaluation Laboratory
University of Illinois at Urbana-Champaign

xiaohu@illinois.edu jdownie@illinois.edu aehmann@illinois.edu

ABSTRACT

This research examines the role lyric text can play in improving audio music mood classification. A new method is proposed to build a large ground truth set of 5,585 songs and 18 mood categories based on social tags so as to reflect a realistic, user-centered perspective. A relatively complete set of lyric features and representation models were investigated. The best performing lyric feature set was also compared to a leading audio-based system. In combining lyric and audio sources, hybrid feature sets built with three different feature selection methods were also examined. The results show patterns at odds with findings in previous studies: audio features do not always outperform lyrics features, and combining lyrics and audio features can improve performance in many mood categories, but not all of them.

1. INTRODUCTION

There is a growing interest in developing and evaluating Music Information Retrieval (MIR) systems that can provide automated access to the mood dimension of music. Twenty-two systems have been evaluated between 2007 and 2008[1] in the Audio Mood Classification (AMC) task of the Music Information Retrieval Evaluation eXchange (MIREX). However, during these evaluations, several important issues have emerged and resolving these issues will greatly facilitate further progress on this topic.

1.1 Difficulties Creating Ground Truth Data

Due to the inherent subjectivity of music perception, there are no generally accepted standard mood categories. Music psychologists have created many different mood models but these have been criticized for missing the social context of music listening [1]. Some MIR researchers have exploited professionally assigned mood labels (e.g. AMG, MoodLogic[2]) [2,3], but none of these taxonomies has gained general acceptance. Professionally created labels have been criticized for not capturing the users' perspectives on mood.

[1] http://www.music-ir.org/mirex/2007/index.php/AMC
[2] http://en.wikipedia.org/wiki/MoodLogic

To date, the AMC dataset is the only ground truth set that has been used to evaluate mood classification systems developed by multiple labs. However, this dataset contains only 600 30 sec. song clips. In fact, reported experiments are seldom evaluated against datasets of more than 1,000 music pieces. The subjective nature of music makes it very difficult to achieve cross assessor agreements on music mood labels. A post-hoc analysis of the 2007 AMC task revealed discrepancies among human judgments on about 30% of the audio excerpts [4]. To overcome the limitation, one could recruit more assessors to assess more candidate tracks. Unfortunately this would require too much human labor to be realistic for most projects. Thus, it is clear that a scalable and efficient method is sorely needed for building ground truth sets for music mood classification experimentation and evaluation.

1.2 Need for Multimodal Mood Classification

The seminal work of Aucouturier and Pachet [5] revealed a "glass ceiling" in spectral-based MIR, due to the fact that many high-level (e.g., semantic) music features simply are not discernable using spectral-only techniques. Thus, researchers started to supplement audio with lyrics and have reported improvements in such tasks as genre classification and artist identification [6,[7]. However, very few studies have combined audio and text for music mood classification [8], and their limitations (see below) call for more studies to investigate whether and how lyrics might help improve classification performance.

1.3 Related Work

Hu et al. [11] derived a set of three primitive mood categories using social tags on last.fm. They collected social tags of single adjective words on a publicly available audio dataset, USPOP [12], and manually selected 19 mood related terms of the highest popularity which then reduced to three latent mood categories using multidimensional scaling. This set was not adopted by others because three categories were seen as a domain oversimplification.

Yang and Lee [8] performed early work on supplementing audio mood classification with lyric text analysis. They combined a lyric bag-of-words (BOW) approach with 182 psychological features proposed in the General Inquirer [13] to disambiguate categories that audio-based classifiers found confusing and the overall classification accuracy was improved by 2.1%. However, their dataset was too small (145 songs) to draw any reliable conclusions. Laurier et al. [9] also combined audio and BOW lyric features. They conducted binary classification experiments on 1,000 songs in four categories and experimental results showed that audio + lyrics combined features improved classification accuracies in all four categories. Yang et al. [10] evaluated both unigram and bigram BOW lyric features as well as three methods for fusing lyric and audio sources on 1,240 songs in four categories. In these studies, the set of four mood categories was most likely oversimplified, the datasets were relatively small and the lyric text features, namely BOW in tf-idf representation, were very limited.

In this paper, we describe a novel method of building a large-scale ground truth dataset with 5,585 songs in 18 mood categories. We then report experiments on a relatively complete set of lyric text features, including function words, POS features and the effect of stemming. Finally, we examine the impact of lyric features on music mood classification by comparing 1) lyric features; 2) audio features; 3) hybrid (lyric + audio features without feature selection; and, 4) hybrid features generated by three feature selection methods.

2. BUILDING A GROUND TRUTH SET

2.1 Data Collection

We began with an in-house collection of about 21,000 audio tracks. Social tags on these songs were then collected from last.fm. 12,066 of the pieces had at least one last.fm tag. Simultaneously, song lyrics were gathered from online lyrics databases. Lyricwiki.org was the major resource because of its broad coverage and standardized format. To ensure data quality, our crawlers used song title, artist and album information to identify the correct lyrics. In total, 8,839 songs had both tags and lyrics. A language identification program[3] was then run against the lyrics, and 55 songs were identified and manually confirmed as non-English, leaving lyrics for 8,784 songs. Table 1 presents the composition of the collection.

Collection	Avg. length (sec.)	Unique	Have tags	Have English Lyrics
USPOP	253.6	8,271	7,301	6,948
USCRAP	243.5	2,553	456	237
American music	183.2	5,049	2,209	790
Metal music	311.8	105	105	104
Beatles	163.8	163	162	161
Magnatune	253.9	4,204	1,261	19
Assorted pop	233.8	600	572	525
Total	(Avg.) 234.8	20,945	12,066	8,784

Table 1. Descriptions and statistics of the collection.

2.2 Identifying Mood Categories

Social tag data are noisy. We employed a linguistic resource, WordNet-Affect [14], to filter out junk tags and tags with little or no affective meanings. WordNet-Affect is an extension of WordNet where affective labels are assigned to concepts representing emotions, moods, or emotional responses. There were 1,586 unique words in the latest version of WordNet-Affect and 348 of them exactly matched the 61,849 unique tags collected from last.fm. However, these 348 words were not all mood related in the music domain. We turned to human expertise to clean up these words. Two human experts were consulted for this project. Both are MIR researchers with a music background and native English speakers. They first identified and removed tags with music meanings that did not involve an affective aspect (e.g., “trance” and “beat”). Second, judgmental tags such as “bad”, “poor”, “good” and “great” were removed. Third, some words have ambiguous meanings and there was not enough information to determine the intentions of the users when they applied the tags. For example, does “love” mean the song is about love or the user loves the song? To ensure the quality of the labels, these ambiguous words were removed. 186 words remained and 4,197 songs were tagged with at least one of the words.

Not all the 186 words represent distinguishable meanings. In fact, many of them are synonyms and should be grouped together [3]. WordNet is a natural resource for synonym identification, because it organizes words into *synsets*. Words in a synset are synonyms from the linguistic point of view. WordNet-Affect goes one step further by linking each non-noun synset (verb, adjective and adverb) with the noun synset from which it is derived. For instance, the synset of “sorrowful” is marked as derived from the synset of “sorrow”. Hence, for the 186 words, those belonging to and being derived from the same synset in WordNet-Affect were grouped together. As a result, the tags were merged into 49 groups.

Several tag groups were further merged if they were deemed musically similar by the experts. For instance, the group of (“cheer up”, “cheerful”) was merged with (“jolly”, “rejoice”); (“melancholic”, “melancholy”) was merged with (“sad”, “sadness”). This resulted in 34 tag groups, each representing a mood category for this dataset. Using the linguistic resources allowed this process to proceed quickly and minimized the workload of the human experts.

For the classification experiments, each category should have enough samples to build classification models. Thus, categories with fewer than 20 songs were dropped resulting in 18 mood categories containing 135 tags. These categories and their member tags were then validated for reasonableness by a number of native English speakers. Table 2 lists the categories, a subset of their member tags and number of songs in each category (after the filtering step described below)[4].

2.3 Selecting the Songs

A song was not selected for a category if its title or artist contained the same terms within that category. For example, all but six songs tagged with “disturbed” were songs by the artist “Disturbed.” In this case, the taggers may simply have used the tag to restate the artist instead of describing the mood of the song. In order to ensure enough data for lyric-based experiments, we only selected those songs with lyrics whose word count was greater than 100 (after unfolding repetitions as explained

[3] http://search.cpan.org/search%3fmodule=Lingua::Ident

[4] Due to space limit, the complete tag list can be found at http://www.music-ir.org/archive/figs/18moodcat.htm

in Section 3.2). After these filtering criteria were applied, we were left with 2,829 unique songs.

Multi-label classification is relatively new in MIR, but in the mood dimension, it is more realistic than single-label classification: A music piece may be "happy and calm" or "aggressive and depressed," etc. This is evident in our dataset as we have many songs that are members of more than one mood category. Table 3 shows the distribution of songs belonging to multiple categories. We adopted a binary classification approach for each of the 18 mood categories, and the 2,829 songs formed the positive example set.

Categories	# of tags	#of songs
calm, comfort, quiet, serene, mellow, chill out,…	25	1,394
sad, sadness, unhappy, melancholic, melancholy	8	916
happy, happiness, happy songs, happy music, …	6	472
romantic, romantic music	2	447
upbeat, gleeful, high spirits, zest, enthusiastic, …	8	321
depressed, blue, dark, depressive, dreary,	11	288
anger, angry, choleric, fury, outraged, rage, …	7	156
grief, heartbreak, mournful, sorrow, sorry, …	14	112
dreamy	1	85
cheerful, cheer up, festive, jolly, jovial, merry, …	13	76
brooding, contemplative, meditative, reflective, …	8	69
aggression, aggressive	2	53
confident, encouraging, encouragement, optimism	5	43
angst, anxiety, anxious, jumpy, nervous, angsty	6	36
earnest, heartfelt	2	34
desire, hope, hopeful, mood: hopeful	4	28
pessimism, cynical, pessimistic, weltschmerz,…	5	27
excitement, exciting, exhilarating, thrill, ardor,…	8	20
TOTAL	135	4,578

Table 2. Mood categories and song distributions.

# of categories	1	2	3	4	5	6
# of songs	1,625	788	305	91	17	2

Table 3. Distribution of songs with multiple labels.

In a binary classification task, each category needs negative samples as well. To create our negative sample set for a given category, we chose songs that were not tagged with any of the terms found within that category but are heavily tagged with many other terms. Since there were plenty of negative samples for each category, we randomly selected songs tagged with at least 15 other terms including mood terms in other categories. Hence, some negative samples of one category are positive samples of another category. In order to make samples of various categories as diverse as possible, we set a constraint that no negative samples were members of more than one category. Similar to positive samples, all negative samples have at least 100 words in their unfolded lyric transcripts. We balanced equally the positive and negative set sizes for each category. Our final dataset comprised 5,585 unique songs.

3. EXPERIMENTS

3.1 Evaluation Measures and Classifiers

This study uses classification accuracy as the performance measure. For each category, accuracy was averaged over a 10-fold cross validation. For each feature set, the accuracies across categories were averaged in a macro manner, giving equal importance to all categories regardless of the size of the categories. To determine if performances differed significantly, we chose the non-parametric Friedman's ANOVA test because the accuracy data are rarely normally distributed.

Support Vector Machines (SVM) were chosen as our classifier because of their strong performances in text categorization and MIR tasks. We used the LIBSVM [15] implementation of SVM and chose a linear kernel as trial runs with polynomial kernels did not yield better results. Parameters were tuned using the grid search tool in LIBSVM, and the default parameters performed best for most cases. Thus, the default parameters were used for all the experiments.

3.2 Lyric Preprocessing

Lyric text has unique structures and characteristics requiring special preprocessing techniques. First, most lyrics consist of such sections as *intro*, *interlude*, *verse*, *pre-chorus*, *chorus* and *outro*, many with annotations on these segments. Second, repetitions of words and sections are extremely common. However, very few available lyric texts were found as verbatim transcripts. Instead, repetitions were annotated as instructions like *[repeat chorus 2x]*, *(x5)*, etc. Third, many lyrics contain notes about the song (e.g., *"written by"*), instrumentation (e.g., *"(SOLO PIANO),"* and/or the performing artists. In building a preprocessing program that took these characteristics into consideration, we manually identified about 50 repetition patterns and 25 annotation patterns. The program converted repetition instructions into the actual repeated segments for the indicated number of times while recognizing and removing other annotations.

3.3 Lyrics Features

Lyrics are a very rich resource and many types of textual features can be extracted from them. This work compares some of the feature types most commonly used in related text classification tasks.

3.3.1 Bag-of-Words (BOW)

Bag-of-words (BOW) are collections of unordered words. Each word is assigned a value that can represent, among others, the frequency of the word, tf-idf weight, normalized frequency or a Boolean value indicating presence or absence. Among these variations, tf-idf weighting is the most widely used in text analysis and MIR, but some studies in text sentiment analysis also reported other representations outperformed tf-idf weighting [16]. These four representations were compared in our experiments.

Selecting the set of words to comprise the BOW set is an important consideration. Stemming is a process of merging words with the same morphological roots, and has shown mixed effects in text classification. Thus, we experimented with both options. We used the Snowball stemmer[5] supplemented with irregular nouns and verbs[6] as this stemmer cannot handle irregular words. Function words (see below) were removed for both the stemming and not stemming cases.

3.3.2 Part-of-Speech (POS)

Part-of-Speech (POS) is a popular feature type in text sentiment analysis [17] and text style analysis [18]. Other MIR studies on lyrics have also used POS features [6,19]. We used the Stanford POS tagger[7] which tags each word with one of 36 unique POS tags.

3.3.3 Function Words

Function words (e.g. *the*, *a*, etc.) carry little meaning. However, function words have been shown to be effective in text style analysis [18]. To evaluate the usefulness of function words in mood classification, the same list of 435 function words found in [18] were used as an independent feature set.

3.4 Audio Processing and Features

Studies in other MIR tasks have generally found lyrics alone are not as informative as audio [6,7]. To find out whether this is true in music mood classification, our best performing lyrics feature set was compared to Marsyas[8], the best performing audio system evaluated in the MIREX 2007 AMC task. Marsyas uses 63 spectral features: means and variances of Spectral Centroid, Rolloff, Flux, Mel-Frequency Cepstral Coefficients, etc. It also uses LIBSVM with a linear kernel for classification. Every audio track in the dataset was converted to 44.1KHz stereo .wav files and fed into Marsyas. The extracted spectral features were subsequently processed by SVM classifiers.

3.5 Hybrid Features and Feature Selection

Previous MIR studies suggest that combining lyric and audio features improves classification performance. Thus, we concatenated our best performing lyrics features and the spectral features to see whether and how much the hybrid features could improve classification accuracies.

In text categorization with BOW features, the dimensionality of document vectors is usually high. Thus, feature selection is often used for the sake of good generalizability and efficient computation. In this study, we compared three methods in selecting the most salient lyric features:

1. Select features with high F-scores. F-score measures the discrimination power of a feature between two sets [20]. The higher a feature's F-score is, the more likely it is to be discriminative. F-score is a generic feature reduction technique independent of classification task and method.

2. Select features using language model differences (LMD) proposed in [9], where the top 100 terms with largest LMD were combined with audio features and showed improved classification accuracies. We wish to find out if this method works in this study with more categories.

3. Select features based on the SVM itself. A trained decision function in a linear SVM contains weights for each feature indicating the relevance of the feature to the classifier. [16] has shown that trimming off features with lower weights improved SVM performance in literature sentimentalism classification. This study investigates if it works for music mood classification.

4. RESULTS

4.1 Best Lyrics Features

Table 4 shows the average accuracies across all 18 categories for the considered lyrics features and representations.

Representation	Boolean	term frequency (tf)	normalized tf	tf-idf weighting
BOW-Stemming	0.5748	0.5819	0.5796	**0.6043**
BOW-Not Stemming	0.5817	0.5829	0.5840	0.5923
POS	0.5277	0.5768	0.5691	0.5571
Function Words	0.5653	0.5733	0.5692	0.5723

Table 4. Average accuracies for lyric features.

The best text feature type is BOW with stemming and tf-idf weighting (BSTI). The difference between stemming options is not significant at $p < 0.05$. The four representations of BOW features do not differ significantly in average performances.

For POS features, the Boolean representation is not as good as others. This is not unexpected because presumably, most lyrics would contain most POS types. In general, POS features and function words are not as good as BOW features. This confirms the heuristic that content words are more useful for mood classification.

4.2 Combining Audio and All Text Features

Three feature sets were compared: spectral features, BSTI, and direct concatenation of both. Their accuracies are shown as part of Table 5. Although their difference is not significant (at $p < 0.05$) on average, BSTI was significantly better than spectral features in these five categories: *romantic*, *grief*, *aggression*, *angst*, and *exciting*. This observation is different from findings in [9] where lyrics features alone did not outperformed audio features in any category.

The accuracies in individual categories are shown in Figure 1 where categories are ordered by decreasing number of samples.

[5] http://snowball.tartarus.org/

[6] The irregular verb list was obtained from http://www.englishpage.com/irregularverbs/irregularverbs.html, and the irregular noun list was obtained from http://www.esldesk.com/esl-quizzes/irregular-nouns/irregular-nouns.htm

[7] http://nlp.stanford.edu/software/tagger.shtml

[8] http://www.music-ir.org/mirex/2007/abs/AI_CC_GC_MC_AS_tzanetakis.pdf

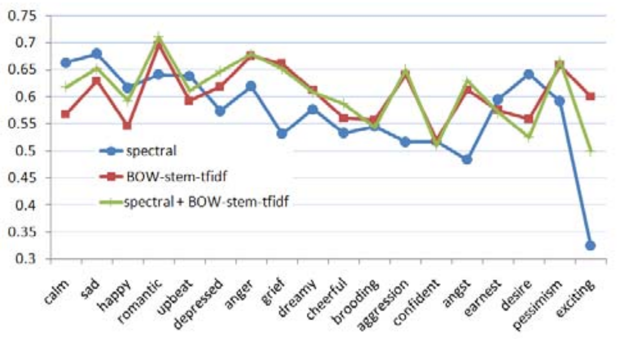


Figure 1. Accuracies of three systems in all categories.

As shown in Figure 1, system performances on different categories vary greatly, and no feature set performs best for all categories. It appears that spectral features are better for larger sized categories, while lyric features are better for middle sized categories. Data sparseness may be less an issue for text features because the samples were chosen to have a certain length of lyrics. From a semantic point of view, the categories where spectral features are significantly better than text features ("upbeat", "happy" and "calm") may have typical auditory characteristics that can be captured by audio spectral features. On the other hand, there may be certain lyrics words that connect well to the semantics of some categories like "grief", "romantic" and "anger." Thus, lyrics features possibly work better because of this connection. For example, the following stemmed words are ranked high for these categories by all of the three aforementioned feature selection methods:

grief: singl, scare, confus, heart, cri, sorry, lone, oooh,…
romantic: endless, love, promis, ador, whisper, lady,…
anger: fuck, man, dead, thumb, girl, bitch, kill,…

It is also clear from Figure 1 that the performances of the combined feature set closely follow the trend of the lyrics-only features. This is probably inevitable given the fact that there are several orders of magnitude more lyric features than spectral features in the combined set. This also demonstrates the necessity of feature selection.

We note that there is a general trend in terms of average accuracy decreasing with smaller sample sizes, sometimes even achieving lower than baseline (50%) performance. These cases also show the highest variance in terms of accuracies across folds. This is a somewhat expected result as the lengths of the feature vectors far outweigh the number of training instances. Therefore, it is difficult to make broad generalizations about these extremely sparsely represented mood categories.

4.3 Combining Audio and Selected Text Features

Using each of the three feature selection methods, we selected the top *n* BSTI features and combined them with the 63 spectral features. We first varied *n* from 63 to 500 (63, 100, 200,…, 500) for all categories. Since the number of features varies per category, we also varied *n* based on the number of features available in each category, from 10% to 90%. The results show that the best *n* varies across the three feature selection methods. Table 5 shows accuracies of the feature sets with the best average performances among each feature selection method.

Category	Spec + F-score n=80%	Spec + LMD n=63	Spec + SVM n=70%	Spec + BSTI	BSTI	Spectral
calm	*0.6112*	**0.6664**	*0.6054*	*0.6176*	*0.5674*	0.6635
sad	*0.6496*	**0.6976**	0.6573	0.6524	*0.6295*	0.6796
happy	0.5965	0.6147	0.5784	0.5922	*0.5455*	**0.6168**
romantic	*0.7014*	*0.7124*	***0.7127***	*0.7104*	*0.6959*	0.6407
upbeat	0.6232	*0.6075*	0.6013	0.6107	*0.5920*	**0.6389**
depressed	0.6318	0.6448	***0.6613***	0.6475	0.6183	0.5741
anger	0.6692	**0.6827**	0.6721	0.6787	0.6754	0.6194
grief	*0.6477*	*0.6386*	*0.6511*	*0.6511*	***0.6610***	0.5314
dreamy	**0.6396**	0.6326	0.6354	0.6083	0.6118	0.5771
cheerful	0.5661	0.5732	**0.5929**	0.5866	0.5598	0.5330
brooding	0.5583	0.5071	**0.5726**	0.5440	0.5571	0.5452
aggression	***0.6683***	0.5667	*0.6300*	*0.6500*	*0.6400*	0.5167
confident	0.6417	**0.7208**	0.5050	0.5100	0.5200	0.5175
angst	*0.4750*	*0.5875*	***0.6292***	***0.6292***	*0.6125*	0.4833
earnest	0.5667	**0.6250**	0.5833	0.5708	0.5750	0.5958
desire	0.5083	*0.4250*	0.5417	0.5250	0.5583	**0.6417**
pessimism	**0.6833**	0.5333	0.6667	0.6667	0.6583	0.5917
exciting	0.5750	0.4250	0.5750	0.5000	***0.6000***	0.3250
AVERAGE	0.6118	0.6033	**0.6151**	0.6084	0.6043	0.5717

Table 5. Accuracies of feature sets for individual categories. (bold font denotes the best for that category, italic indicates significant difference from spectral features at $p < 0.05$.)

The results show that not all categories can be improved by combining lyric features with spectral features. Audio-only and lyric-only features outperform *all* combined feature sets in five of the 18 categories. Each of the combined feature sets outperforms lyric and audio features in at most nine categories. This is different from findings in previous studies [8,9] where combined features were best for all experimented categories.

In particular, the language model difference method with 63 lyric features (Spec + LMD n = 63) shows an interesting pattern: it improves accuracies in six of the 12 categories where lyric features outperform spectral features and three of the six categories where spectral features beat lyric features. This indicates that, with the same dimensionality, lyrics and audio have indeed a similar impact on combined features.

In combining lyrics and audio features, feature selection often yields better results because many text features are either redundant or noisy. In terms of average accuracies, features selected by SVM models work slightly better for SVM classifiers than the other two feature selection methods. However, it is interesting to see that (Spec + LMD n = 63) outperforms lyric and audio features in nine categories which are the most among all combined feature sets. It also outperforms all others in five mood categories and achieves significantly better results than spectral features in three other mood categories. Similar patterns are observed for the F-score method with 63 lyric features. This suggests that in hybrid feature sets, lyric features can be and should be aggressively reduced.

5. CONCLUSIONS AND FUTURE WORK

This paper investigates the usefulness of text features in music mood classification on 18 mood categories derived from user tags. Compared to Part-of-Speech and function words, Bag-of-Words are still the most useful feature type. However, there is no significant difference between the choice of stemming or not stemming, or among the four text representations (e.g. tf-idf, Boolean, etc) on average accuracies across all categories.

Our comparisons of lyric, audio and combined features discover patterns at odds with previous studies. In particular lyric features alone *can* outperform audio features in categories where samples are more sparse or when semantic meanings taken from lyrics tie well to the mood category. Also, combining lyrics and audio features improves performances on most, but not all, categories. Experiments on three different feature selection methods demonstrated that too many text features are indeed redundant or noisy and combining audio with the most salient text features may lead to higher accuracies for most mood categories.

Future work includes investigation of other text features, such as text statistics and affective words provided by domain lexicons. It would also be interesting to take a close look at individual categories and find out why lyrics features do or do not help. Moreover, more sophisticated feature and model combination techniques besides naïve feature vector concatenation are worth investigating.

6. ACKNOWLEDGEMENT

We thank the Andrew W. Mellon Foundation for their financial support.

7. REFERENCES

[1] P. N. Juslin and P. Laukka: "Expression, Perception, and Induction of Musical Emotions: A Review and A Questionnaire Study of Everyday Listening," *Journal of New Music Research*, Vol. 33, No. 3, pp. 217-238, 2004.

[2] M. Mandel, G. Poliner, and D. Ellis: "Support Vector Machine Active Learning for Music Retrieval," *Multimedia Systems*, Vol. 12, No. 1, pp. 3-13, 2006.

[3] X. Hu and J. S. Downie: "Exploring Mood Metadata: Relationships with Genre, Artist and Usage Metadata," *Proceedings of the International Conference on Music Information Retrieval*, 2007.

[4] X. Hu, J. S. Downie, C. Laurier, M. Bay, and A. Ehmann: "The 2007 MIREX Audio Music Classification Task: Lessons Learned," *Proceedings of the International Conference on Music Information Retrieval*, 2008.

[5] J-J. Aucouturier and F. Pachet: "Improving Timbre Similarity: How High Is the Sky?" *Journal of Negative. Results in Speech and Audio Sciences,* Vol.1, No.1, 2004.

[6] T. Li and M. Ogihara: "Semi-Supervised Learning from Different Information Sources," *Knowledge and Information Systems*, Vol.7, No.3, pp.289-309, 2004

[7] R. Neumayer and A. Rauber: "Integration of Text and Audio Features for Genre Classification in Music Information Retrieval," *Proceedings of the European Conference on Information Retrieval,* pp.724 – 727, 2007.

[8] D. Yang, and W. Lee: "Disambiguating music Emotion Using Software Agents," In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04),* 2004.

[9] C. Laurier, J. Grivolla and P. Herrera: "Multimodal Music Mood Classification Using Audio and Lyrics," *Proceedings of the International Conference on Machine Learning and Applications,* 2008.

[10] Y.-H. Yang, Y.-C. Lin, H.-T. Cheng, I.-B. Liao, Y.-C. Ho, and H. H. Chen: "Toward multi-modal music emotion classification", Proceedings of Pacific Rim Conference on Multimedia (PCM'08), 2008.

[11] X. Hu, M. Bay, and J. S. Downie: "Creating a Simplified Music Mood Classification Groundtruth Set," *Proceedings of the 8th International Conference on Music Information Retrieval,* 2007.

[12] D. Ellis, A. Berenzweig, and B. Whitman: "The USPOP2002 Pop Music Data Set," Retrieved from http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html, 2003.

[13] Stone, P. J. *General Inquirer: a Computer Approach to Content Analysis.* Cambridge: M.I.T. Press, 1966.

[14] C. Strapparava and A. Valitutti: "WordNet-Affect: an Affective Extension of WordNet," *Proceedings of the International Conference on Language Resources and Evaluation,* pp. 1083-1086, 2004.

[15] C. Chang and C. Lin: *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[16] B. Yu: "An Evaluation of Text Classification Methods for Literary Study," *Literary and Linguistic Computing* Vol. 23, No. 3, pp. 327-343, 2008.

[17] B. Pang and L. Lee: "Opinion Mining and Sentiment Analysis," *Foundations and Trends in Information Retrieval,* Vol.2 No.1-2, pp. 1–135, 2008

[18] S. Argamon, M. Saric and S. S. Stein: "Style Mining of Electronic Messages for Multiple Authorship Discrimination: First Results," *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining,* pp. 475-480, 2003.

[19] R. Mayer, R. Neumayer, and A. Rauber: "Rhyme and Style Features for Musical Genre Categorisation by Song Lyrics," *Proceedings of the International Conference on Music Information Retrieval,* 2008.

[20] Y.-W. Chen and C.-J. Lin: "Combining SVMs with Various Feature Selection Strategies," In *Feature Extraction, Foundations and Applications*, Springer, 2006.

ROBUST AND FAST LYRIC SEARCH BASED ON PHONETIC CONFUSION MATRIX

Xin Xu, Masaki Naito, Tsuneo Kato
KDDI R&D Laboratories, Inc.
sh-jo, naito, tkato@kddilabs.jp

Hisashi Kawai
National Institute of Information and Communications Technology
hisashi.kawai@nict.go.jp

ABSTRACT

This paper proposes a robust and fast lyric search method for music information retrieval. Current lyric search systems by normal text retrieval techniques are severely deteriorated in the case that the queries of lyric phrases contain incorrect parts due to mishearing and misremembering. To solve this problem, the authors apply acoustic distance, which is computed based on a confusion matrix of an ASR experiment, into DP-based phonetic string matching. The experimental results show that the search accuracy is increased by more than 40% compared with the normal text retrieval method; and by 2% $\sim$4% compared with the conventional phonetic string matching method. Considering the high computation complexity of DP matching, the authors propose a novel two-pass search strategy to shorten the processing time. By pre-selecting the probable candidates by a rapid index-based search for the first pass and executing a DP-based search among these candidates during the second pass, the proposed method reduces processing time by 85.8% and keeps search accuracy at the same level as that of a complete search by DP matching with all lyrics.

1. INTRODUCTION

An easy-to-use music information retrieval (MIR) system plays an essential role in realizing satisfactory music distribution services. Current commercial MIR systems accept diverse queries by text, humming, singing, and acoustic music signals. Among these types of queries, text queries of lyric phrases are commonly used (lyric search) [1]. As many MIR systems apply full-text search engines to search lyric, the issue of lyric search has been widely accepted as a solved issue by state-of-art text retrieval techniques. However, the authors' preliminary investigations on real world queries suggested that, users are likely to input incorrect lyric phrases (incorrect queries in this paper) into MIR systems resulting in a failed lyric search. The incorrect lyric phrases are due to unreliable human memory or mishearing, as users remember the lyric phrases when they are impressed by hearing a part of a song without a lyric sheet. The analysis found that incorrect queries which replaces a word with another word of a similar pronunciation reaches 19%. This phenomenon is called "acoustic confusion" here.

In text retrieval field, some fuzzy algorithms, such as Latent Semantic Indexing (LSI) and partial matching, were used by major commercial Web search engines [2] to improve the robustness against incorrect queries. However, Xu's research verified that these algorithms were not helpful for acoustic confusion [3].

To solve this problem peculiar to lyric search, a search method is expected to be able to identify a lyric containing a part that is most similar in acoustic respect to the query. Phonetic string matching, which is used in such applications as name retrieval [4], was considered to be closest to the expected method. It uses edit distance between phoneme strings to search words with similar sound. However, edit distance is the minimum number of operations needed to transform one string into the other [5]. It does not present the degree of acoustic confusability between phonemes. For example, /aki/ is easily misheard as /agi/ as opposed to /aoi/, though the edit distances are identical. This is because the phonemes, "k" and "g", tend to be confused mutually, compared with "k" and "o".

In order to take the degree of acoustic confusability between phonemes into account for string matching, the authors apply a new distance, called acoustic distance, to phonetic string matching. Acoustic distance is obtained by DP matching with the costs derived from phonetic confusion probabilities between the phoneme strings of a query and a lyric. It is motivated by the ideas in Spoken Document Retrieval (SDR) and Spoken Utterance Retrieval (SUR) [6, 7]. Phonetic confusion probabilities are derived from a phonetic confusion matrix that is obtained from a preliminary automatic speech recognition (ASR) experiment.

As it is found by authors' preliminary investigations that queries of lyric phrases are not segmented by word or sentence boundary, edge-free DP matching between two phoneme strings of a query and a lyric is used to calculate acoustic distance. The computation complexity of DP matching $O\{DP\}$ cannot be ignored here because lyrics always contain long phoneme strings. Conventional phonetic string matching applied a complete search by DP matching with all lyrics, so the computation complexity is regarded as $O\{DP\} * I_t$, where I_t is the number of lyrics to search. Since commercial MIR systems usually provide hundreds of thousands of lyrics, the computation complexity is too high to realize a real time search.

Therefore, a lyric search method with a two-pass search strategy is proposed to speed up the search process. In the first pass, the proposed method pre-selects the probable lyric candidates by a rapid approximate search based on the accumulation of pre-computed and indexed partial acoustic distances. Then, a complete search by DP matching with the remaining lyrics is carried out during the second pass, which decreases the value of I_t contributing to the computation complexity.

The experimental results show that the application of phonetic confusion probability improves search accuracy in lyric search. Moreover, the processing time is greatly reduced by using two-pass search strategy.

The remainder of this paper is organized as follows: the analysis of real world queries is described in Section 2. The definition of acoustic distance and the proposed method are introduced in Section 3. The experiments are carried out to evaluate the proposed method in Section 4. The paper is summarized in Section 5.

2. ANALYSIS OF REAL WORLD LYRIC QUERIES

To analyze the queries of lyric phrases for MIR in the real world, the authors investigated some question & answer community web sites, where many questions were found that used lyric phrases to request the names of songs and singers. As 1140 queries of lyric phrases were collected, the authors compared each query with its corresponding lyric to distinguish whether lyric phrases in the query are correct or not (correct query or incorrect query) and how they were mistaken. The lyrics and queries are written in Japanese or English, or a mixture of both.

Figure 1 shows the distribution of incorrect queries in the different types and correct queries within the collected data. The incorrect queries, which occupy around 79%, are classified into the following types:

- Confusion of notations: Chinese characters in the queries are substituted for syllabary characters (Hiragana or Katakana in Japanese), or vice versa.
- Function-word-error: Only the function words, which have little lexical meaning, such as prepositions, pronouns, auxiliary verbs, are mistaken in the queries.
- Content-word-error: The content words such as a noun, verb, or adjective, that have a stable lexical meaning, are mistaken in the queries.

In the current full-text search methods, function-word-error and confusion of notations can be handled using a stop word list to filter out the function words [8], and a hybrid index of words and syllables [9].

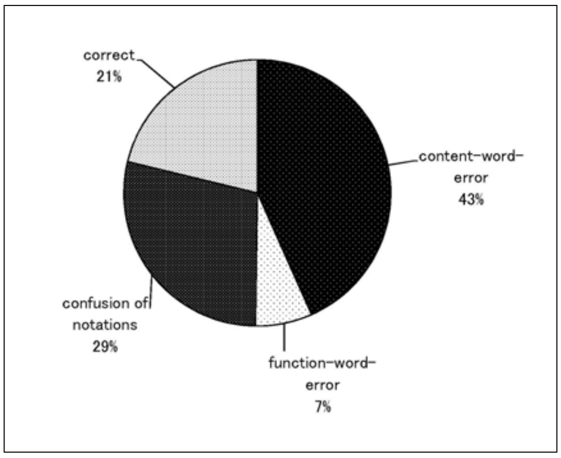


Figure 1. The distribution of incorrect queries in the different types and correct queries within the collected queries

On the other hand, as the content words play more important roles in determining the search intension [8], content-word-error queries were further categorized into three subtypes by the authors, viz., namely “acoustic confusion”, “meaning confusion” and “others”. The percentages and examples are listed in Table 1. The mistaken parts are marked in bold.

Acoustic confusion is defined as a replacement of a word with that of a similar pronunciation; or a replacement of the unknown-spelling words with syllable strings of a similar pronunciation. For the first example of acoustic confusion queries in Table 1, “/kotoganai/” and “/kotobawanani/” have similar pronunciations while the text strings have no common parts. In the second example, the Japanese syllable string is used as a query whose pronunciation is similar to the English phrase, “You’ve been out riding fences for so long now” in the target lyric. It was supposed to happen when users were not able to spell the foreign words that they heard in a song.

Meaning confusion is defined as a replacement of a word with its synonym or near-synonym. As shown in Table 1 the first example of meaning confusion queries, “/anata/” is mistaken for “/kimi/”. Both of the terms refer to the same meaning “you” in Japanese. For the second example, “/tsuki/” and “/hoshi/”, which mean “moon” and “star”, are confused.

The type of “others” contains word insertion, word deletion and other errors in the queries. From the analysis of collected examples, it is known that mistakes in “others” type are derived from arbitrary reasons, which include individual experiences or memories, special environments, and other reasons. The analysis did not find a relationship between the mistakes and the lyrics.

As the acoustic confusion queries occupy about 45% of content-word-error queries (19% of the collected queries), it remains an important issue for lyric search. Based on the description above, identifying a lyric containing a part that is most similar in the acoustic aspect of the query is a better solution for acoustic confusion than focusing on the textual or semantic aspects.

Types of queries	Percentage			Examples	
				Correct lyric	Mistaken queries
Acoustic confusion	45%	Ex.1	Text (Japanese):	好きな**事がない**	好きな**言葉は何**
			Pronunciation:	/sukina**kotoganai**/	/sukina**kotobawanani** /
			Meaning:	There is **nothing** I like.	What are you favorite **words**?
		Ex.2	Text:	**You've been out riding fences for so long now**	**ユーベーナウプラウドゥンシーンクスソーセングナウ**
			Pronunciation:	**/yuubiibiiNNautoraidiNNgufeNNshizufoosooroNNgunau /**	**/yuubeenaupurauduNNshiiNNkususooseNNgunau/**
			Meaning:	You've been out riding fences for so long now	* no actual meaning
Meaning confusion	17%	Ex.1	Text (Japanese):	**君**には何でも話せるよと	**あなた**には何でも話せるよと
			Pronunciation:	/**kimi**niwanaNNdemohanaseruyoto/	/**anata**niwanaNNdemohanaseruyoto/
			Meaning:	I can say anything to **you**	I can say anything to **you**
		Ex.2	Text (Japanese):	**月**に願いを	**星**に願いを
			Pronunciation:	/**tsuki**ninegaio/	/**hoshi**ninegaio/
			Meaning:	pray to the **moon**	pray to the **star**
Others	38%	Ex.1	Text (Japanese):	星**から来た**子の見る夢は	星の子**チョビン**の見る夢は
			Pronunciation:	/hoshi**karakita**konomiruyumewa /	/hoshinoko**chobiNN**nomiruyumewa /
			Meaning:	The dream that the child who **came from** the star has	The dream that child **Chobin** of the star has

Table 1. The distribution of mistaken types within content-word-error cases

3. AN EFFICIENT SEARCH METHOD BASED ON ACOUSTIC DISTANCE

3.1 Introduction of Acoustic Distance

The authors introduce acoustic distance to quantify the degree of acoustic confusion. Acoustic distance is calculated by DP matching with cost values derived from phonetic confusion probabilities, instead of the constant cost values used for edit distance.

First, a phonetic confusion matrix is obtained by running a phoneme speech recognizer over training data and by aligning the recognition results of phoneme strings with reference phoneme strings.

For the elements of the confusion matrix, $n(p, q)$ means the number of phoneme q obtained as recognition results by the actual utterances of phoneme p. As "ϕ" represents a null, $n(\phi, p)$ means the number of the misrecognized phoneme p (insertion) and $n(p, \phi)$ means the number of the deleted phoneme p (deletion). M represents the set of phonemes including null.

For each phoneme p, the phonetic confusion probabilities of an insertion $P_{ins}(p)$, deletion $P_{del}(p)$ and substitution for phoneme q $P_{sub}(p, q)$ are calculated on the basis of the confusion matrix elements, by Eq.1∼3.

$$P_{ins}(p) = \frac{n(\phi, p)}{\sum_{k \in M} n(k, p)} \tag{1}$$

$$P_{del}(p) = \frac{n(p, \phi)}{\sum_{k \in M} n(p, k)} \tag{2}$$

$$P_{sub}(p, q) = \frac{n(p, q)}{\sum_{k \in M} n(p, k)} \tag{3}$$

As a large value of $P_{ins}(p)$ represents a high confusability for an insertion of p, it corresponds to a low cost of an insertion operation for p in string matching based on DP. Therefore the value of insertion cost $C_{ins}(p)$, is calculated by Eq.4. In the same way, the value of deletion cost $C_{del}(p)$ and substitution cost $C_{sub}(p, q)$, are calculated from the corresponding phonetic confusion probabilities by Eq.5 and Eq.6.

$$C_{ins}(p) = 1 - P_{ins}(p) \tag{4}$$

$$C_{del}(p) = 1 - P_{del}(p) \tag{5}$$

$$C_{sub}(p, q) = 1 - P_{sub}(p, q) \tag{6}$$

Second, with the calculated cost values, edge-free DP matching between the phoneme strings S_1, S_2 is carried out by Eq.7∼9. Here, $S[x]$ is xth phoneme of phoneme string S and $len(S)$ means the length of S ($S_1, S_2 \in S$). $D(i, j)$ designates the minimum distance from the starting point to the lattice point (i, j). D_{S_1,S_2} is the accumulated cost of DP matching between S_1 and S_2, which is defined as the acoustic distance. It reflects acoustic confusion probability for each phoneme.

1. Initialization:

$$D(0, j) = 0 (0 \leq j \leq len(S_2)); \tag{7}$$

2. Transition:

$$D(i, j) = \min \begin{cases} D(i, j-1) + C_{ins}(S_2[j]) \\ D(i-1, j-1) + C_{sub}(S_1[i], S_2[j]) \\ D(i-1, j-1), (S_1[i] = S_2[j]) \\ D(i-1, j) + C_{del}(S_1[i]) \end{cases} \tag{8}$$

3. Determination

$$D_{S_1,S_2} = min\{D(len(S_1), j)\}(0 < j \leq len(S_2)); \tag{9}$$

3.2 Searching Method based on Acoustic Distance by DP matching

Based on the criterion that a lyric containing a part that has the minimum acoustic distance from the query should be the user's target, a method with a complete search by DP matching with all lyrics is described as follows:

1. The lyrics L_{I_t} are converted into syllable strings using a morphological analysis tool such as Mecab [10]. The syllable strings are converted into phoneme strings by referring to a syllable-to-phoneme translation table. Consequently, a phoneme string $S_{L(k)}$ represents a lyric $L(k)$ $(L(k) \in L_{I_t})$.
2. Once a query Q is provided, it is converted into a phoneme string S_Q in the same way as step 1. By Eq.7∼9, the acoustic distance $D_{S_Q,S_{L(k)}}$ between the query and all lyrics L_{I_t} is calculated.
3. Lyrics L_{I_t} are ranked in the order of the acoustic distance $D_{S_Q,S_{L(k)}}$, and then the lyrics with lower distance values are provided as search results.

3.3 Searching Method based on Acoustic Distance by Two-pass Searching

Considering that the complete search by DP matching with all lyrics requires high computation complexity, a method with two-pass search strategy is proposed and realized with following steps:

- Preliminary indexing: An inverted index construction is preliminarily incorporated for the first pass search. A list of linguistically existing units of N successive syllables (syllable N-gram) $A_1 \cdots A_n$ are collected from the text corpus. The units are organized as index units for fast access, as shown in Table 2. The acoustic distance $D_{S_{A_n},S_{L(k)}}$ between the phoneme strings of A_n and $L(k)$ are pre-computed by Eq.7∼9 and stored in the index matrix. It can be regarded as an index of acoustic confusion.
- First pass search: By accessing the index described above, a fast search is realized by using the four steps below, and the flowchart is illustrated in Figure 2:
    1. The input query Q is converted into a syllable string v by Macab.
    2. By Eq.10 the syllable string is converted into syllable N-gram sets, $V_1, \cdots, V_m, \cdots, V_M$. Here, $v[m]$ is the mth syllable of v.

    $$V_m = \{v[m], v[m+1], \cdots, v[m+N-1]\}; \tag{10}$$

    3. $V_1, \ldots, V_m, \ldots, V_M$ are matched with the index units $A_1, \ldots, A_n, \ldots$. By accumulating the pre-computed and indexed distance values $D_{S_{A_n},S_{L(k)}}$, the approximate acoustic distance $R(k)$ is calculated by Eq.11.

    $$R(k) = \sum_{m=1,\cdots,M} D_{S_{A_n},S_{L(k)}}, (V_m = A_n) \tag{11}$$

    4. To narrow the search space of lyrics, $L(k)$ with higher $R(k)$ is pruned off, and a lyric set L_{I_c} containing I_c $(I_c < I_t)$ best lyric candidates is preserved for the second pass.
- Second pass search: A complete search by DP matching with the lyrics in L_{I_c} is carried out.

Based on the processes above, the computing complexity of the proposed method is reduced to $O\{FPS\}+O\{DP\}* I_c$. As $O\{FPS\}$ is the computing complexity of the first pass search which is much less than $O\{DP\}$ and I_c is much less than I_t, it provides a faster response for a real-time MIR system.

4. EVALUATION OF SEARCH ACCURACY AND PROCESSING TIME

4.1 Experimental Set Up and Test Set

To evaluate the search performance of the proposed search method, experiments were carried out. A database of 10000 lyrics was collected containing both Japanese and English lyrics. The test set consisted of 220 incorrect queries that were mistaken in acoustic confusion. They were from the collected queries mentioned in Section 2. The lyrics corresponding to the queries were included in the database. The results of the experiment were obtained using a personal computer (Intel Core2Duo CPU 3.0GHz, 4G RAM). Four methods described as follows, were compared.

- "Baseline": A normal partial matching method using full-text retrieval engine "Lucene", which is based on inverted index construction [11]. In this method, as the query of lyric phrases is divided into N successive character substrings, the lyric containing more substrings is regarded as a more suitable candidate.
- "Method based on Edit Distance of Phoneme (EDP)": The search method based on the edit distance of phoneme strings, which is described in [4].
- "Method based on Acoustic Distance by DP matching (ADDP)": The search method using a complete search described in Section 3.2. The phonetic confusion matrix for calculating acoustic distance is obtained using the same speech recognition experiment as in [12]. Although the confusion matrix should be based on singing voice, a huge amount of singing data is not available. In this research, telephone speech data of Japanese phonetically balanced sentences were

syllable 3-gram \ Lyric No.	`L(1)`	$\cdots$	`L(k)`	$\cdots$
A_1 `[a-i-u]`	0.34	$\cdots$	0.23	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
A_n `[na-ko-to]`	0.88	$\cdots$	$D_{S_{A_n},S_{L(k)}}$	$\cdots$

Table 2. A sample of index item by syllable 3-gram

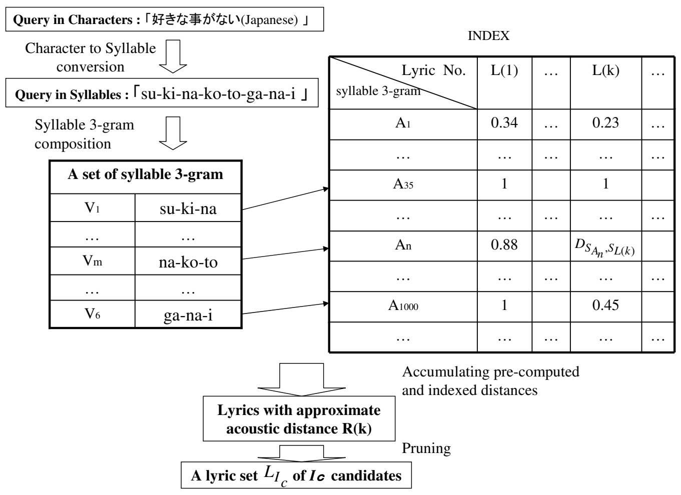

Query in Characters :「好きな事がない(Japanese) 」
Character to Syllable conversion
Query in Syllables :「su-ki-na-ko-to-ga-na-i 」
Syllable 3-gram composition
A set of syllable 3-gram
V1 su-ki-na
Vm na-ko-to
V6 ga-na-i
INDEX
Lyric No.
syllable 3-gram
L(1)
L(k)
A1 0.34 0.23
A35 1 1
An 0.88 $D_{S_{A_n},S_{L(k)}}$
A1000 1 0.45
Accumulating pre-computed and indexed distances
Lyrics with approximate acoustic distance R(k)
Pruning
A lyric set L_{I_c} of I_c candidates

Figure 2. Flowchart of the first pass search

used as the training data for the acoustic models of ASR.

- "Method based on Acoustic Distance by Two-pass Searching (ADTS)": The proposed method using the two-pass search strategy as described in Section 3.3. Considering the balance of index size and search accuracy, here N of syllable N-gram index is set to 3. The syllable 3-grams are collected from the lyric and newspaper corpus. 100,000 entries of syllable 3-grams, which cover 90% of all syllable 3-grams in the collected text corpus, are prepared in the index. As all the syllable 3-grams which exist in the queries are prepared, no search errors come from out-of-vocabulary syllable 3-grams in the experiments.

4.2 Improvements of Search Accuracy by Applying Acoustic Distance

The comparison of the results between "EDP" and "ADDP" are shown in Figure 3. The vertical axis means the hit rate, while the horizontal axis shows the top T candidates of the ranked lyrics, which is called T-best. Here, the hit rate of T-best is defined as the rate of the total number of hits within top T candidates to the total number of search accesses. "ADDP" improves the hit rates by 2% $\sim$ 4%, as the T of T-best is ranged from 1 to 100. It indicates that the proposed acoustic distance gives better effects than edit distance.

4.3 Evaluation of Search Accuracy and Time Complexity

The search accuracy and time complexity of four methods are shown in Figure 4 and Table 3 respectively.

Note that, the value of I_c for "ADTS" is determined by a preliminary experiment based on the same test set. It only uses the first pass search of "ADTS" for lyric search to investigate the relationship between hit rates and T-best to chose the best threshold value for I_c. Because the hit rates almost saturated when T is larger than 800, I_c is set to 800 in this paper.

With a well-designed data structure, "Baseline" achieved the fastest response among four methods. However, since the normal text retrieval techniques cannot solve the acoustic confusion problem in lyric search, other three methods based on phonetic string matching achieved higher search accuracy than "Baseline" by more than 40%. On the other hand, though "ADDP" and "EDP" provide high performances of search accuracy, the processing times for one query are over 9 seconds, which are not practical in real world search. By applying a fast search in the first pass to narrow the search space, "ADTS" shortens the processing time into 1.85 seconds, which is 14.2% of "ADDP", with only 0.5% $\sim$ 5% deterioration of search accuracy due to the loss happened in the index-based pruning of "ADTS".

Attributing to the application of acoustic distance, "ADTS" keep almost the same hit rates as "EDP" and achieves 2% improvement when T is larger than 20, by using only 19% time of "EDP".

5. CONCLUSION

This paper proposed a robust and fast lyric search method based on the introduced acoustic distance and a two-pass search strategy using an index-based approximate preselection for the first pass and a DP-based string matching in the second pass. In the case of incorrect queries caused by acoustic confusion, the proposed method achieved significantly higher search accuracy than the normal text retrieval method by more than 40%. An improvement by 2% ∼4% is also achieved compared with the conventional phonetic string matching method. Furthermore, the proposed method realized a real time operation by reducing 85.8% processing time with a slight loss in search accuracy compared with a complete search by DP matching with all lyrics. It is proved to be the most practical solution for acoustic confusion queries on the balance of high search accuracy and light computation complexity.

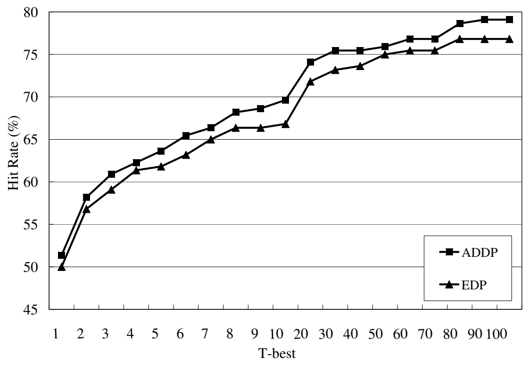


Figure 3. The improvement of search accuracy by acoustic distance

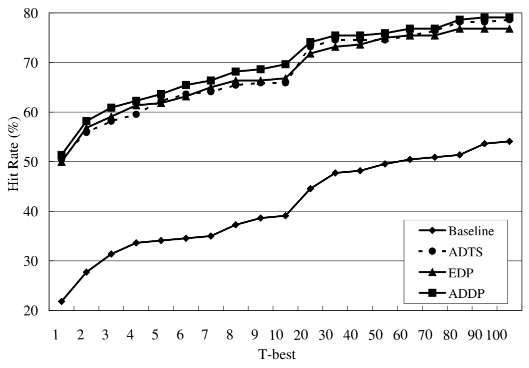


Figure 4. Search accuracy of four search methods

Methods	Baseline	ADTS	EDP	ADDP
Time (second)	0.006	1.85	9.72	13.01

Table 3. Average processing times of one query

6. REFERENCES

[1] Downie and Cunningham: "Toward a theory of music information retrieval queries: System design implications," *Proceedings of the Third International Conference on Music Information Retrieval (ISMIR 2002)*, pp. 299–300, 2002.

[2] Denys Poshyvanyk et al.: "Combining Probabilistic Ranking and Latent Semantic Indexing for Feature Identification," *the 14th IEEE International Conference on Program Comprehension*, pp. 137–148, 2006.

[3] Xin Xu, Masaki Naito, Tsuneo Kato, Hisashi Kawai: "An Introduction of a Fuzzy Text Retrieval System For Music Information Retrieval," *Information Processing Society of Japan SIG Notes*, No.127, pp. 41–46, 2008.

[4] J. Zobel and P. Dart: "Phonetic string matching: Lessons from information retrieval," *Proceedings of the 19th International Conference on Research and Development in Information Retrieval*, pp. 166–172, 1996.

[5] E. Ristad and P. Yianilos: "Learning string edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, pp. 522–532, 1998.

[6] Ville T. Turunen, Mikko Kurimo: "Indexing confusion networks for morph-based spoken document retrieval," *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 631–638, 2007.

[7] Takaaki Hori et.al: "Open-Vocabulary Spoken Utterance Retrieval Using Confusion Networks," *Proceedings of the 2007 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 73–76, 2007.

[8] Christopher Fox: "A stop list for general text," *ACM SIGIR Forum*, Vol. 24, No. 1–2, pp. 19-21, Fall 1989/Winter 1990.

[9] Nina Kummer, Christa Womser-Hacker and Noriko Kando: "MIMOR@NTCIR 5: A Fusion-based Approach to Japanese Information Retrieval," *Proceedings of NTCIR-5 Workshop Meeting, Tokyo, Japan*, 2005.

[10] http://mecab.sourceforge.net

[11] Erik Hatcher, Otis Gospodnetic: *Lucene In Action*, Manning Publications Co., 2004.

[12] Makoto Yamada, Tsuneo Kato, Masaki Naito and Hisashi Kawai: "Improvement of Rejection Performance of Keyword Spotting Using Anti-Keywords Derived from Large Vocabulary Considering Acoustical Similarity to Keywords," *Proceedings of INTERSPEECH*, pp. 1445–1448, 2005.

USING HARMONIC AND MELODIC ANALYSES TO AUTOMATE THE INITIAL STAGES OF SCHENKERIAN ANALYSIS

Phillip B. Kirlin
Department of Computer Science, University of Massachusetts Amherst
pkirlin@cs.umass.edu

ABSTRACT

Structural music analysis is used to reveal the inner workings of a musical composition by recursively applying reductions to the music, resulting in a series of successively more abstract views of the composition. Schenkerian analysis is the most well-developed type of structural analysis, and while there is a wide body of research on the theory, there is no well-defined algorithm to perform such an analysis. A automated algorithm for Schenkerian analysis would be extremely useful to music scholars and researchers studying music from a computational standpoint. The first major step in producing a Schenkerian analysis involves selecting notes from the composition in question for the primary soprano and bass parts of the analysis. We present an algorithm for this that uses harmonic and melodic analyses to accomplish this task.

1. INTRODUCTION

Numerous tasks in music information retrieval could be accomplished more effectively if information about musical structure were readily available. For example, in the task of retrieving musical passages that are similar to a given passage, having structural analyses available would allow similarity metrics to be based on the underlying musical structure of a composition as well as on the musical surface. An algorithm for structural analysis of music would therefore be an indispensable resource in music information research.

Schenkerian analysis [1] is a type of music analysis that emphasizes finding structural relationships among the notes of a composition. Developed by the Austrian music theorist Heinrich Schenker, Schenkerian analysis differs from other types of analysis that focus on a single aspect of music, such as the harmony or melody, to the exclusion of other aspects. Schenkerian analysis harnesses all aspects of a piece together to create an analysis that explains how various notes in the piece function in relation to others.

Of particular importance in Schenkerian analysis is the identification of *structural dependences* among groups of notes. If the way in which a note X functions in a musical passage is due to the presence of another note or group of notes Y, then X is said to be dependent upon Y, and Y is said to be at a higher structural level than X. The process of finding structural dependences proceeds recursively during an analysis. The final set of dependences can be depicted as a tree, with the surface-level notes as the leaves. With each structural dependence located, the more structurally important notes are elevated to higher levels.

Though a tree theoretically can show all the hierarchical levels of a Schenkerian analysis, typically analyses are illustrated through a sequence of *Schenker graphs*. These graphs are visual depictions of a few contiguous levels of the note hierarchy, using staves with notes as in common music notation, but using other notation symbols such as stems, beams, and slurs to show relationships among notes rather than timing or phrasing information.

Because Schenkerian analysis primarily focuses on the main melodic line and the main harmonic bass line of the music in question, Schenker graphs are often presented on two staves, with the primary melodic line on the upper staff and the supporting bass harmony tones on the lower staff. Notes of inner voices are occasionally shown on the graphs, but are sometimes omitted when they serve to only fill out the harmony. We focus on *foreground graphs*, the graphs that show the structural levels closest to the musical surface. A foreground graph is usually the first graph constructed when completing a Schenkerian analysis; all subsequent graphs are based — directly or indirectly — on the foreground graph. Therefore, it is critical to choose the correct set of notes to appear in the foreground graph. We will call a foreground graph, after notes have been selected for its staves but before any reductions have been applied, a *preliminary foreground graph*. Consider the first eight measures of Schubert's *Impromptu No. 2 in A-flat major*, shown in Figure 1. A preliminary foreground Schenker graph for the *Impromptu*, with appropriate notes in the soprano and bass parts, would look like Figure 2.

In this paper, we present and analyze an algorithm, FOREGRAPH, for identifying which notes in a score should belong on the soprano and bass staves of a preliminary foreground Schenker graph, based on analyses of harmony and voice leading. We build on the work of Kirlin and Utgoff [2], whose IVI system requires, as a first step, isolation of the primary soprano and bass parts prior to analysis. Automating Schenkerian analysis has been studied recently by Marsden [3–5] and Marsden and Wiggins [6]. These

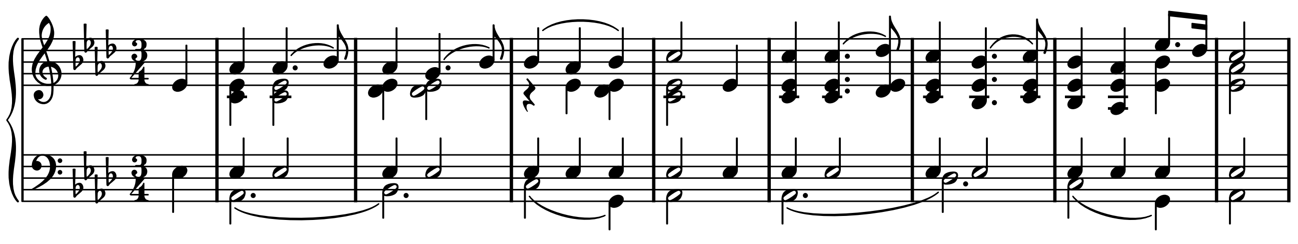


Figure 1. An excerpt from *Schubert's Impromptu No. 2 in A-flat major*.

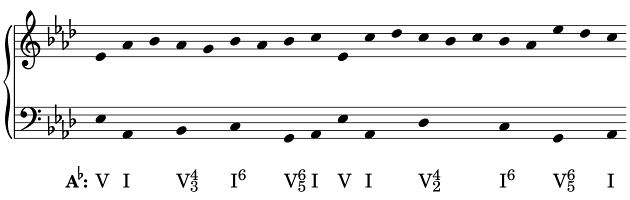


Figure 2. A preliminary foreground graph constructed by hand from the *Impromptu*.

lines of work are promising, but they have only been tested on short, sometimes synthetic, musical phrases. Lerdahl and Jackendoff developed a grammatical approach to musical structure in [7], which Hamanaka, et. al. [8] turned into an algorithm. Their system, however, requires manual adjustment of many parameters that differ for each musical composition. Older work by Kassler [9] and Smoliar [10] demonstrated understanding of the principles involved in automating analysis, but did not provide any algorithms.

2. COMPUTATIONAL METHODS FOR HARMONIC AND MELODIC ANALYSIS

Schenkerian analysis is based on the principles of harmony and voice leading. These two aspects of a composition must be examined prior to beginning an analysis. Since we desire a fully-automated system for producing foreground graphs, we must examine various algorithms for determining the harmony at various points in a composition, and the voice leading possibilities for any note in a piece.

We have chosen MusicXML as our representation of choice. MusicXML is a file format that represents common Western music notation by encoding the pitches and durations of notes. Though the MIDI representation is more widely used than MusicXML, the latter format encodes an additional wealth of information that the former does not supply, such as key and time signatures, stem and beaming information, and slurs and phrase marks.

One can look at harmonic analysis as occurring in two phases. First, a chord-labeling component assigns chord labels (such as "C Major") to segments of a composition. A second pass then uses the chord labels to assign functional Roman numerals to segments.

2.1 Chord Labeling

A chord labeling component must divide a composition temporally into segments, where each segment corresponds to a single harmony. We use a variant of Pardo and Birmingham's HARMAN algorithm [11] to accomplish this.

HARMAN uses two separate algorithms to perform chord labeling. The *labeling algorithm* is concerned with determining the best chord label for a given segment of music (a segment being an interval of time with fixed starting and ending times), and the *segmentation algorithm* determines the points in the music where the harmony changes. A harmony can change at a *partition point*: any place in the music where a note starts or stops.

While HARMAN does a very good job "out of the box," we use a modified version of the algorithm and detail our changes below.

- **Meter** — HARMAN does not take the meter of the piece into account, and sometimes it chooses a partition point in a metrically weak position that is adjacent to a metrically stronger one. Because it is preferable to have a change of harmony in an analysis at a metrically strong position [12], we force HARMAN choose each measure boundary as a partition point.

- **Octave doubling** — Because HARMAN analyzes each note in a segment independently, often notes that are doubled at the octave exert too much of an influence over the chord labeling algorithm. Therefore, when analyzing a segment, we consider multiple instances of notes with the same pitch class and duration as a single note. For example, in Figure 1, the notes of the melody in measures 5–7 that are doubled at the octave will not be counted twice.

- **Neighbor tones** — Our version of HARMAN ignores "obvious" neighbor tones within segments. An obvious neighbor tone is a note Y that occurs in a note sequence $X - Y - Z$ where X and Z have the same pitch, and are separated from Y by a half-step. Without this correction, HARMAN has trouble distinguishing between chord tones and non-chord tones in heavily figurated contexts.

2.2 Assignment of Roman Numerals

Given a chord labeling, the remaining task in harmonic analysis is mapping the chord labels (such as "C Major") to Roman numeral labels (such as "V^6"). While we have investigated algorithms for computing the key of a composition and the locations of any modulations and tonicizations, we restrict ourselves for the remainder of this discussion to non-modulating pieces whose key is encoded correctly in their MusicXML representation. As FOREGRAPH, the foreground Schenker graph generation algorithm presented in the next section, relies on a correct Roman numeral analysis, placing this restriction on the input music makes us more certain that we are supplying correct Roman numerals to FOREGRAPH.

The second phase detects tonicizations by looking for consecutive chords where the first chord functions functions as a temporary dominant to the second. For example, in the key of C major, this would detect the chord sequence "D Major – G Major" and change the harmonic analysis of "II – V" to "V/V – V." We stipulate that the first chord cannot occur normally in the original key, to eliminate the possibility of the common "I – IV" chord sequence being reinterpreted as a tonicization of the IV chord.

2.3 Voice Leading Analysis

A voice leading analysis determines, for every note in the piece, which notes could logically follow from that note, according to the principles of voice leading [12]. Algorithms for determining voice leading, however, can differ in their interpretations of *implicit polyphony* [13]. For example, given the four notes in the first measure of Figure 3, some algorithms would determine that all four notes belong to a single voice, whereas others would find two voices and interpret the four notes as standing for the triads shown in the second measure. The second interpretation is an example of implicit polyphony.

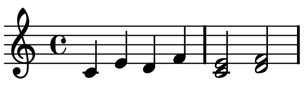


Figure 3. An example where the voice leading is ambiguous.

Schenkerian analysis, as it gives primary consideration to the linear connections in music [14], requires a voice leading analysis that uncovers implicit polyphony. A reasonable way to handle this is to permit a voice-leading connection between two notes only if the motion between them is stepwise.

If one takes this stance, it is easy to construct an algorithm for determining the voice leading for a given composition. For a note n in a piece, we examine the set of notes that begin at times later than the ending of n (there cannot be a voice-leading connection between two notes that overlap in time). Note n may have up to three voice leading connections: (1) a step-down connection, (2) a step-up connection, and (3) a same-pitch connection. For each type of connection, we find the earliest note that satisfies the criteria for that kind of connection. We also require that if n has a same-pitch voice-leading connection to a note m, then n may not have any stepwise voice-leading connections to notes that begin later than m. This is because voice-leading connections between notes of identical pitch are typically stronger than stepwise connections.

3. PRODUCING PRELIMINARY FOREGROUND GRAPHS

Recall that our goal is to produce preliminary foreground Schenker graphs like the one in Figure 2. Since the purpose of a foreground graph is to capture the primary soprano and bass tones of the piece, constructing such a graph reduces to selecting notes for the soprano and bass parts.

In most circumstances, the primary melody (soprano) tone is the highest one heard at any point in time, and the primary bass tone is the lowest. Therefore, FOREGRAPH is based on the idea of selecting the highest pitch for the soprano line and the lowest for the bass line. However, complications arise in situations *where the primary bass or soprano tones persist in time even though they may have stopped sounding.* Consider an Alberti bass line, such as in Figure 4. Because this figure is outlining a chord, only the lowest note of the chord belongs to the primary bass line (the other notes belong to inner voices). The low Cs, though they are only represented on the page as eighth notes, persist in the musical mind through the entire measure as if they were sounding constantly; the true bass line does not skip between the notes of the chord. This is the reason why we require a voice leading algorithm that can detect cases of implicit polyphony, not just in cases of arpeggiation, but in any case where the bass or soprano part may move between voices.

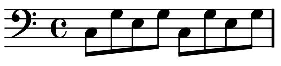


Figure 4. An Alberti bass line.

Still, there are cases where voices start and stop mid-composition, and an algorithm that blindly follows the initial bass and soprano lines stepwise from the start of the piece to the end would not suffice in cases, for example, of register transfer. Therefore, FOREGRAPH chooses appropriate bass and soprano tones for each harmonic segment defined by the harmonic analysis algorithm, and then follows the tones via voice-leading connections to fill out the segment; pseudocode is given in Figure 5. For each harmonic segment in a composition, FOREGRAPH finds the lowest and highest pitched notes that belong to the current harmony; these notes are added to the primary bass and soprano parts. The FILLRANGE procedure then adds additional notes by following voice-leading connections from the initial notes added in the segment; connections are followed both backwards and forwards in time, and notes are only added if they do not overlap in time with any other notes already added to the segment.

The EXTENDVOICE procedure then allows the musical line in a harmonic segment to be extended into following segments, stopping only upon reaching a note that is consonant in the prevailing harmony for the segment. Because the primary notes are determined independently for each harmonic segment, it is possible that the soprano or bass lines fleshed out by FILLRANGE will not connect musically over a segment break. EXTENDVOICE permits each line to be followed to a logical conclusion without adding too many notes of what may develop into an irrelevant inner voice. Because EXTENDVOICE halts upon adding a consonant note in the prevailing harmony, leaps are possible in the computed musical lines over segment breaks.

After choosing the notes for the soprano and bass lines, they are displayed as noteheads on staves as a preliminary foreground graph. FOREGRAPH produced the output shown in Figure 6 for the Schubert Impromptu in Figure 1.

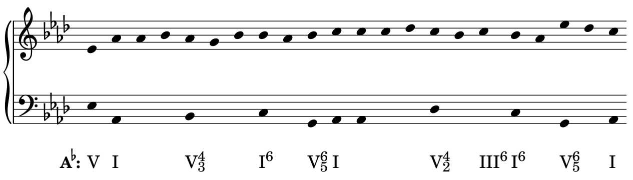


Figure 6. A preliminary foreground graph produced by FOREGRAPH.

If one compares the hand-constructed graph in Figure 2 to the one produced by FOREGRAPH in Figure 6, only a few differences are apparent. One is that the computer-constructed graph contains instances of adjacent notes of identical pitch. FOREGRAPH does not reduce these cases to single notes because although this occurs frequently in foreground graphs, it is not always done consistently.

The only other differences in the computer-generated analysis are the omitted “V” chord near the middle of the analysis, and the added “III^6” chord. Both of these differences derive from the harmonic analysis component used as a preliminary step to FOREGRAPH. The V chord in the hand-constructed graph was not generated in the computer analysis as it was absorbed into the I chords on either side. Similarly, the first-inversion III chord arises from a misinterpretation of chord tones and non-chord tones.

4. EVALUATION AND ANALYSIS

In order to evaluate the correctness of FOREGRAPH, we require a set of input music scores and correct foreground graphs for them. We turned to a standard Schenkerian analysis textbook [14], and encoded the first twelve musical examples that had correct analyses provided, and whose analyses contained soprano and bass parts (two of the examples were monophonic, and so were omitted). The examples are all multi-measure excerpts from common practice period works.

Our method of evaluation is based on the standard metrics of precision and recall. If one views each note in a composition as an individual document, then constructing a preliminary foreground graph is equivalent to executing two queries: one query to retrieve all notes belonging to the soprano part, and a second query to retrieve all notes belonging to the bass part. We also need to define what it means for a note to be “relevant” and “retrieved” to compute precision and recall. We consider a note “retrieved” for a query if it appears in the corresponding part (soprano or bass) for the computer-constructed foreground graph. Defining “relevant” is complicated because the foreground graphs as they appear in the textbook (1) often contain pertinent pitches of inner voices along with the primary soprano and bass parts, and (2) already have had some reductions applied in most cases, which removes some notes from the ground-truth that would appear in the computer-generated graphs.

Therefore, we have two notions of “relevant” and compute statistics based on each definition. In our first set of calculations, we consider a note to be relevant for the soprano (bass) query if it is present on the upper (lower) staff of the Schenker graph in the textbook analysis. This definition, however, considers many notes as relevant that will not be present in the computer-generated analyses as they belong to inner voices. To remedy this, our second definition considers a note to be relevant for the soprano (bass) query if it is present on the upper (lower) staff in the Schenker graph in the textbook analysis, and has a stem pointing up (down). If it is clear that stem direction in a graph is *not* being used to indicate to which voice a note belongs (and the direction is only determined by aesthetics), the restriction on stem direction is ignored, and only the presence of the stem is considered. Stems in graphs are indications of structural importance, and therefore these are notes that we are particularly interested in having FOREGRAPH identifying correctly.

We ran the FOREGRAPH algorithm on each example and compared the resulting graphs to the textbook’s graphs. For each example, and for each part (soprano and bass), we computed precision (the fraction of retrieved notes that were also relevant) and recall (the fraction of relevant notes that were also retrieved). To provide a baseline for comparison with FOREGRAPH, we evaluated a second foreground graph creation algorithm, RANDOM, that selects notes for the soprano and bass parts from the input music randomly. RANDOM always chooses the same number of notes for the soprano and bass parts for each example as were selected by FOREGRAPH for the same example. We calculated average precision and recall for RANDOM over 500 runs. All of the precision and recall statistics for FOREGRAPH and RANDOM are displayed in Table 1. To show more clearly the improvement of FOREGRAPH over RANDOM, Figure 7 compares the F1 measure (harmonic mean of precision and recall) for each musical example for the two algorithms.

One of the excerpts deserves special mention. The excerpt from Schubert’s *Symphony in B minor* confused FOREGRAPH as the accompaniment part is pitched higher than the primary melody. The analysis constructed by FOREGRAPH contained a harmony line in the soprano part, and the true melody was not present at all. Because this single example distorted the statistics for the soprano part, Table 1 contains entries for the aggregate precision and recall with and without the *Symphony* included.

Overall, we are encouraged by the results of the evaluation. We are especially pleased with the recall values for the stemmed notes definition of relevance; disregarding the Schubert *Symphony*, FOREGRAPH retrieved almost 90% of the relevant bass notes, and almost 80% of the relevant soprano notes. Figure 7 clearly indicates that FOREGRAPH

```
procedure ForeGraph
    Let $V(x, y)$ be true if there is a voice leading connection between notes $x$ and $y$.
    Let $S$ be a set of notes for the primary soprano part.
    Let $B$ be a set of notes for the primary bass part.
    for each harmonic segment $H$ in the composition do
        Let $n$ be the lowest pitched note in $H$ that is a member of $H$'s harmony.
        Add $n$ to $B$
        FillRange$(n, B, H)$
        ExtendVoice$(B, H)$
        Let $n$ be the highest pitched note in $H$ that is a member of $H$'s harmony.
        Add $n$ to $S$
        FillRange$(n, S, H)$
        ExtendVoice$(S, H)$
procedure FillRange(note $n$, part $P$, harmonic segment $H$)
    Initialize queue $Q$ to contain just $n$
    while $Q$ is not empty do
        Remove the top note from the queue, call it $m$
        Let $N$ be the set of all notes such that if $x \in N$, then either $V(m, x)$ or $V(x, m)$, and $x$ is in $H$.
        Sort $N$ by increasing length of time between $m$ and each note in $N$
        if $N$ is empty, then return
        for each note $x \in N$ do
            if $x$ does not conflict with any notes in $P$ then add $x$ to $P$ and add $x$ to $Q$
procedure ExtendVoice(part $P$, harmonic segment $H$)
    Let curr be the last note in $H$ that is also in $P$
    while curr is not consonant in $H$'s harmony do
        Let $N$ be the set of all notes such that if $x \in N$, then $V(m, x)$
        if $N$ is empty, then return
        Let $n$ be the note in $N$ with the minimum length of time to curr
        Add $n$ to $P$
        curr $\leftarrow n$
```

Figure 5. The ForeGraph algorithm.

is a large improvement over choosing notes randomly.

The two issues mentioned earlier that complicated choosing an appropriate definition of relevance cause the precision and recall values to not represent the true quality of the graphs produced by ForeGraph. The first issue is that many of the ground-truth analyses contain notes of inner voices on the upper and lower staves, as well as notes from the primary soprano and bass parts. The bass part of the Chopin *Nocturne*, for example, contains arpeggiated chords. ForeGraph only included the lowest note of each chord in the primary bass part, while the textbook included all of the notes of each chord, with all but the lowest given as inner voices. This lowered the recall value for all bass notes in this example to 23.5%.

The second issue is that many of the textbook's graphs have already had simple reductions applied to the musical surface; repeated notes in the textbook's graphs have also been removed in many cases. Because ForeGraph only *selects* notes for the foreground graphs and does not perform any reductions, many of the precision values are lower than they would be if those reductions had not been done in the textbook's graphs. For example, in the *French Suite*; ForeGraph placed many notes in the soprano part that were not present in the textbook's graph because reductions had already been applied to them.

We are confident that ForeGraph is ready to be used as a precursor to an actual Schenkerian reduction algorithm. Because we are only selecting notes to be placed in the soprano and bass parts, the output of ForeGraph is ready for processing to search for reductions, and any low precision statistics should not be alarming.

5. REFERENCES

[1] Heinrich Schenker. *Der Freie Satz*. Universal Edition, Vienna, 1935. Published in English as *Free Composition*, translated and edited by E. Oster, Longman, 1979.

[2] Phillip B. Kirlin and Paul E. Utgoff. A framework for automated Schenkerian analysis. In *Proceedings of the Ninth International Conference on Music Information Retrieval*, pages 363–368, Philadelphia, September 2008.

[3] Alan Marsden. Automatic derivation of musical structure: A tool for research on Schenkerian analysis. In *Proceedings of the Eighth International Conference on Music Information Retrieval*, pages 55–58, 2007. Extended Version.

[4] Alan Marsden. Generative structural representation

	All notes				Stemmed notes			
	Precision		Recall		Precision		Recall	
Excerpt	Sop.	Bass	Sop.	Bass	Sop.	Bass	Sop.	Bass
J. S. Bach, *Aria variata* (BWV 989)	0.320	0.438	0.889	0.700	0.200	0.250	1.000	0.667
Beethoven, *Ninth Symphony*, III	0.818	0.800	0.818	1.000	0.455	0.300	0.714	1.000
Haydn, *Symphony in D major, No. 104*	0.667	0.538	0.706	0.875	0.389	0.462	0.875	0.857
Chopin, *Prelude in A major*, Op. 28/7	0.636	0.333	0.636	1.000	0.091	0.333	0.500	1.000
Haydn, *Divertimento in B-flat*, II	0.643	0.615	1.000	1.000	0.286	0.308	1.000	1.000
Schubert, *Standchen* from *Schwanengesang*	0.778	0.571	0.280	1.000	0.556	0.429	0.556	1.000
J. S. Bach, Chorale No. 149	0.800	0.857	1.000	0.857	0.600	0.429	1.000	0.750
J. S. Bach, *French Suite in C minor*, Sarabande	0.550	0.750	0.786	0.900	0.250	0.333	0.714	0.800
Schubert, *Symphony in B minor, No. 8*, I	0.000	0.667	0.000	1.000	0.000	0.667	0.000	1.000
Mozart, *Symphony in C major*, K. 425, IV	0.636	0.174	0.609	0.500	0.273	0.130	1.000	1.000
Chopin, *Nocturne in D-flat major*, Op. 27/2	0.889	0.400	0.533	0.235	0.333	0.400	0.500	1.000
Brahms, *Rhapsody in E-flat major*, Op. 119/4	1.000	1.000	0.917	1.000	0.364	0.636	1.000	1.000
FOREGRAPH, all excerpts	0.568	0.557	0.568	0.772	0.279	0.364	0.729	0.911
FOREGRAPH, all except for Schubert's *Symphony*	0.650	0.547	0.675	0.753	0.319	0.336	0.797	0.896
RANDOM, all excerpts	0.255	0.154	0.255	0.213	0.104	0.083	0.273	0.208
Improvement of FOREGRAPH over RANDOM	0.313	0.403	0.313	0.559	0.175	0.281	0.456	0.703

Table 1. Precision and recall for evaluation.

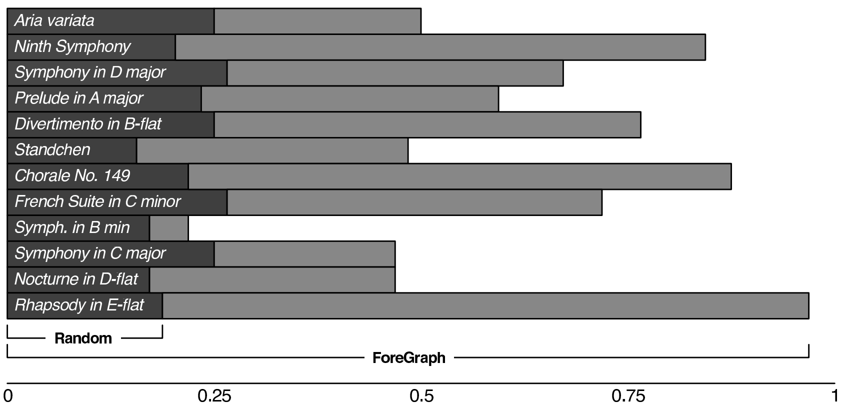


Figure 7. A graph comparing the F1 measures for RANDOM (dark bars) and for FOREGRAPH (light bars).

of tonal music. *Journal of New Music Research*, 34(4):409–428, December 2005.

[5] Alan Marsden. Extending a network-of-elaborations representation to polyphonic music: Schenker and species counterpoint. In *Proceedings of the First Sound and Music Computing Conference*, pages 57–63, 2004.

[6] Alan Marsden and Geraint A. Wiggins. Schenkerian reduction as search. In *Proceedings of the Fourth Conference on Interdisciplinary Musicology*, Thessaloniki, Greece, July 2008.

[7] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, Massachusetts, 1983.

[8] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. ATTA: Implementing GTTM on a computer. In *Proceedings of the Eighth International Conference on Music Information Retrieval*, pages 285–286, 2007.

[9] Michael Kassler. APL applied in music theory. *APL Quote Quad*, 18(2):209–214, 1987.

[10] Stephen W. Smoliar. A computer aid for Schenkerian analysis. *Computer Music Journal*, 2(4):41–59, 1980.

[11] Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[12] Edward Aldwell and Carl Schachter. *Harmony and Voice Leading*. Harcourt Brace & Company, Fort Worth, Texas, second edition, 1989.

[13] David Temperley. *The Cognition of Basic Musical Structures*. MIT Press, Cambridge, Massachusetts, 2001.

[14] Allen Forte and Steven E. Gilbert. *Introduction to Schenkerian Analysis*. W. W. Norton and Company, New York, 1982.

Hierarchical Sequential Memory for Music: A Cognitive Model

James B. Maxwell
Simon Fraser University
jbmaxwel@sfu.ca

Philippe Pasquier
Simon Fraser University
pasquier@sfu.ca

Arne Eigenfeldt
Simon Fraser University
arne_e@sfu.ca

ABSTRACT

We propose a new machine-learning framework called the Hierarchical Sequential Memory for Music, or HSMM. The HSMM is an adaptation of the Hierarchical Temporal Memory (HTM) framework, designed to make it better suited to musical applications. The HSMM is an online learner, capable of recognition, generation, continuation, and completion of musical structures.

1. INTRODUCTION

In our previous work on the MusicDB [10] we outlined a system inspired by David Cope's notion of "music recombinance" [1]. The design used Cope's "SPEAC" system of structural analysis [1] to build hierarchies of musical objects. It was similar to existing music representation models [7, 9, 13], in that it emphasized the construction of hierarchies in which the objects at each consecutively higher level demonstrated increasing "temporal invariance" [5]—i.e., an "S" phrase in SPEAC analysis, and a "head" in the Generative Theory of Tonal Music [9], both use singular names at higher levels to represent *sequences* of musical events at lower levels.

Other approaches to learning musical structure include neural network models [8], recurrent neural network models (RNNs) [11], RNNs with Long Short-Term Memory [3], Markov-based models [12, 14], and statistical models [2]. Many of these approaches have achieved high degrees of success, particularly in modeling melodic and/or homophonic music. With the HSMM we hope to extend such approaches by enabling a single system to represent melody, harmony, homophony, and various contrapuntal formations, with little or no explicit *a priori* modeling of musical "rules"—the HSMM will learn only by observing musical input. Further, because the HSMM is a cognitive model, it can be used to exploit musical knowledge, in real time, in a variety of interesting and interactive ways.

2. BACKGROUND: THE HTM FRAMEWORK

In his book "On Intelligence", Jeff Hawkins proposes a "top-down" model of the human neocortex, called the "Memory Prediction Framework" (MPF) [6]. The model is founded on the notion that intelligence arises through the interaction of perceptions and predictions; the perception of sensory phenomena leads to the formation of predictions, which in turn guide action. When predictions fail to match learned expectations, new predictions are formed, resulting in revised action. The MPF, as realized computationally in the HTM [4, 5], operates under the assumption of two fundamental ideas: 1) that memories are hierarchically structured, and 2) that higher levels of this structure show increasing temporal invariance.

The HTM is a type of Bayesian network, and is best described as a memory system that can be used to discover or infer "causes" in the world, to make predictions, and to direct action. Each node has two main processing modules, a Spatial Pooler (SP) for storing unique "spatial patterns" (discrete data representations expressed as single vectors) and a Temporal Pooler (TP) for storing temporal groupings of such patterns.

The processing in an HTM occurs in two phases: a "bottom-up" classification phase, and a "top-down" recognition, prediction, and/or generation phase. Learning is a bottom-up process, involving the storage of discrete vector representations in the SP, and the clustering of such vectors into "temporal groups" [4], or variable-order Markov chains, in the TP. A node's learned Markov chains thus represent temporal structure in the training data. As information flows up the hierarchy, beliefs about the identity of the discrete input representations are formed in each node's SP, and beliefs about the membership of those representations in each of the stored Markov chains are formed in the TP. Since the model is hierarchical, higher-level nodes store invariant representations of lower-level states, leading to the formation of high-level spatio-temporal abstractions, or "concepts."

A simplified representation of HTM processing is given in Figure 1. Here we see a 2-level hierarchy with two

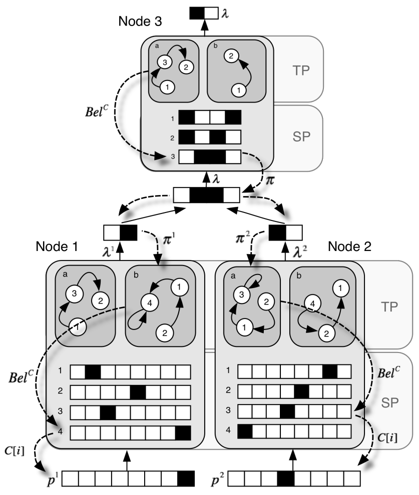


Figure 1. Simplified HTM processing.

nodes at L1 and one node at L2. This HTM has already received some training, so that each L1 node has stored four spatial patterns and two Markov chains, while the L2 node has stored three spatial patterns and two Markov chains. There are two input patterns, p^1 and p^2. It can be seen that p^1 corresponds to pattern 4 of Node 1, and that pattern 4 of Node 1 is a member of Markov chain *b*. When presented with p^1, the node identifies pattern 4 as the stored pattern most similar to p^1, calculates the membership of pattern 4 in each of the stored Markov chains, and outputs the vector [0, 1], indicating the absence of belief that p^1 is a member of Markov chain *a*, and the certainty that p^1 is a member of Markov chain *b*.

It can also be seen from Figure 1 that the outputs of the children in hierarchy are concatenated to form the inputs to the parent. The SP of node 3 thus treats the concatenated outputs of nodes 1 and 2 as a discrete representation of their temporal state at a given moment—i.e., time is 'frozen' by the parent node's SP. Node 3 then handles its internal processing in essentially the same manner as nodes 1 and 2.

The dotted lines indicate the top-down processes by which discrete state representations can be extracted or inferred from the stored Markov chains, and passed down the network. Top-down processing can be used to support the recognition of inputs patterns, to make predictions, or to generate output.

3. MOTIVATIONS BEHIND THE HSMM

Our interest in the HTM as a model for representing musical knowledge derives from its potential to build spatio-temporal hierarchies. The current HTM implementation from Numenta Inc., however, is focused primarily on visual pattern recognition [4, 5], and is currently incapable of learning the sort of high-level temporal structure found in music. This structure depends not only on the temporal proximity of input patterns, but also on the specific *sequential order* in which those patterns arrive. The HSMM treats sequential order explicitly, and can thus build detailed temporal hierarchies.

Another motivation behind the HSMM lies in the fact that the HTM is strictly an offline learner. For compositional applications, we are interested in a system that can acquire new knowledge during interaction, and exploit that knowledge in the compositional process. We have thus designed the HSMM with four primary functions in mind:

1) **Recognition:** The system should have a representation of the hierarchical structure of the music at any given time in a performance.
2) **Generation:** The system should be capable of generating stylistically integrated musical output.
3) **Continuation:** If a performance is stopped at a given point, the system should continue in a stylistically appropriate manner.
4) **Pattern Completion:** Given a degraded, or partial input representation, the system should provide a plausible completion of that input (i.e., by adding a missing note to a chord).

4. MUSIC REPRESENTATION

For the current study, we are working with standard MIDI files from which note data is extracted and formatted into three 10-member vectors: one for pitch data, one for rhythmic data, and one for velocity data. The incoming music is first pooled into structures similar to Cope's "Groups" [1]—vertical 'slices' of music containing the total set of unique pitches at a given moment. A new Group is created every time the harmonic structure changes, as shown in Figure 2. The Groups are preprocessed using a simple voice-separation algorithm, which divides polyphonic material across the 10 available voices in the vector representation. Group pitches are first sorted in ascending order, after which the voice separation routine follows one simple rule: tied (sustained) notes must not switch voices.

Pitch material is represented using inter-pitch ratio [16], calculated by converting the MIDI notes to hertz,

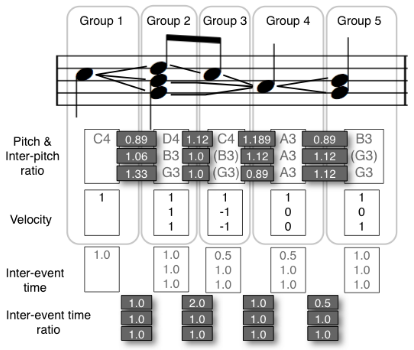


Figure 2. Music representation for the HSMM.

and dividing the pitch at time *t*-1 by the pitch at time *t.* In order to avoid misleading values, as a result of calculating the ratio between a rest (pitch value 0.0) and a pitch, rests are omitted from the pitch representation, and the velocity representation is used to indicate when notes are active or inactive (see Figure 2).

It will be noted that velocity is not given using conventional MIDI values, but is rather used as a flag to indicate the state of a given voice in the Group. Positive values indicate onsets, negative values indicate sustained notes, and zeros indicate offsets. We have simplified the non-zero values to 1 and -1 in order to avoid attributing too much weight to note velocity in the training and inference process.

The rhythmic values used represent the times at which each voice undergoes a transition either from one pitch to another, or from a note-on to a note-off. We use the inter-event time between such changes, and calculate the ratio between consecutive inter-event times, for each voice *n*, according to the following:

$$interEventRatio^{t}[n] = \frac{interEventTime^{t-1}[n]}{interEventTime^{t}[n]} \quad (1)$$

The final representation for the HSMM will thus consist of one 10-member inter-pitch ratio vector, one 10-member inter-event time ratio vector, and one 10-member velocity flag vector.

5. HSMM LEARNING AND INFERENCE

Figure 3 shows a four-level HSMM hierarchy with inputs for pitch, rhythm, and velocity information, an "association" node (L2) for making correlations between the L1 outputs, and two upper-level nodes for learning higher-ordered temporal structure. The association node at L2 provides the necessary connectivity between pitch, rhythm, and velocity elements required for the identification of musical "motives." The upper-level nodes at L3 and L4 are used to learn high-order musical structure from the motives learned at L2.

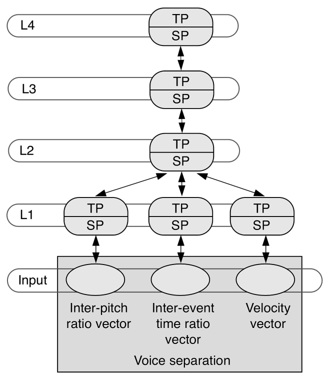


Figure 3. A four-level HSMM hierarchy.

5.1 Online Learning in the HSMM

Whereas the TP in the HTM builds Markov chains during its training phase, in the HSMM we focus simply on constructing discrete *sequences* from the series of patterns input to the node. As in the HTM, the patterns stored in the SP will be referred to as "coincidences." The sequences stored by the TP will be referred to simply as "sequences."

5.1.1 Learning and Inference in the Spatial Pooler

The objective of SP learning is to store unique input patterns as coincidences, while the objective of SP inference is to classify input patterns according to the stored coincidences. The algorithm used is given in Figure 4. As in the HTM, when a new input is received the SP checks the input against all stored coincidences, *C*. The result is an SP output vector y^t, calculated according to:

$$y^t[i]=e^{-d(p,C[i])^2/\sigma^2},\ for\ i=0\ to\ |C|-1 \tag{2}$$

where $d(p,C[i])$ is the Euclidean distance from input p to coincidence $C[i]$, and σ is a constant of the SP. The constant σ is used to account for noise in the input, and is useful for handling rhythm vectors, where frequent fluctuations of timing accuracy are expected. The output y^t is a belief distribution over coincidences, in which a higher value indicates greater similarity between input pattern p and stored coincidence $C[i]$, and thus greater evidence that p should be classified as $C[i]$. If the similarity of p to all stored coincidences is less than the minumum allowed similarity, *simThreshold*, p is added as a new coincidence.

In the event that a new coincidence is added, the algorithm uses the value of *maxSimilarity*—i.e., the belief in the coincidence most similar to input p—as the initial belief value when adding the new coincidence. It then normalizes y^t in order to scale the new belief according to the belief over all stored coincidences. In order to decide whether a new coincidence is required at higher levels, we start by first determining whether the input pattern λ (see Figure 1) should be stored as a new coincidence. This is simply a matter of checking the length of the λ vector at time t against the length at time t-1. If the length has increased, we know that at least one of the children has learned a new representation in the current time step, and that a new coincidence must be added in order to account for the additional information. For each new coincidence stored by the SP, a histogram called the *counts* vector is updated. In the HTM, the update is an integer incrementation—a count of how many times the coincidence has been seen during training. However, because the HSMM is an online learner, an integer incrementation is not appropriate, as it would lead to counts of vanishingly small proportions being assigned to new coincidences if the system were left running for long periods of time. Thus, in the HSMM, the *counts* vector is updated according to the following:

$$inc=|C|\times 0.01 \tag{3}$$

$$counts^t[topCoinc^t]=counts^{t-1}[topCoinc^t]+inc \tag{4}$$

$$counts^t[i]=\frac{counts^t[i]}{1+inc},\ for\ i=0\ to\ |counts|-1 \tag{5}$$

where C is the number of learned coincidences, *inc* is the incrementation value, and $topCoinc^t$ is the coincidence that rated as having the *maxSimilarity* (Figure 4) to the input. Because *counts* is regularly normalized, it represents a time-limited histogram in the HSMM.

SP inference above L1 is calculated as in the HTM, but we outline it here for the sake of clarity. At higher levels we want to calculate the *probability* that the new input λ should be classified as one of the stored coincidences. When the node has more than one child, we consider each child's contribution to the overall probability separately:

$$\begin{gathered} C=C^1\cup\ldots\cup C^M \\ \lambda=\lambda^1\cup\ldots\cup\lambda^M \\ y^t[i]=\prod_{j=1}^{M}\max_k(C^j[k,i])\times\lambda^j[k],\ for\ i=0\ to\ |C|-1 \end{gathered} \tag{6}$$

where M is the number of child nodes, C^j is the portion of coincidence vector k attributed[1] to child j, and λ^j is the portion of λ attributed to child j. Figure 5 shows an example calculation for a hypothetical SP with two stored coincidences.

p	The current input pattern
C	The table of stored coincidences
maxSimilarity	The maximum similarity value found
simThreshold	The minimum degree of similarity between input *p* and coincidence *C*[*i*] required for *p* to be classified as *C*[*i*]
unmatchedCoinc	A count of the number of times input *p* was found to be insufficiently similar to all coincidences in *C*

```
Set maxSimilarity to 0
For each stored coincidence C[i] in C
  Calculate $y^t[i]$, given input p, according to Equation 2
  If $y^t[i]$ > maxSimilarity
    Set maxSimilarity to $y^t[i]$
  If $y^t[i]$ < simThreshold
    Increment unmatchedCoinc count

If unmatchedCoinc count = size of C
  add input p to stored coincidences C
  append maxSimilarity to end of $y^t$ vector
  normalize $y^t$ vector
```

Figure 4. Online SP learning and inference.

[1]Recall that when the node has more than one child, each coincidence will be a concatenation of child outputs.

5.1.2 Learning in the Temporal Pooler

The objective of TP learning is to construct sequences from the series of belief vectors (y) received from the SP. When a new input to the TP is received, the TP first calculates the winning coincidence of y^t:

$$topCoinc^t = \underset{i}{\operatorname{argmax}}\,(y^t[i]) \tag{7}$$

It then determines whether this coincidence has changed since the previous time step—i.e., whether $topCoinc^t$ equals $topCoinc^{t-1}$—and stores the result in a flag called *change*.

The next step is to determine whether the transition from $topCoinc^{t-1} \rightarrow topCoinc^t$ exists among the TP's stored sequences. To do this, we depart from the HTM entirely, and use an algorithm we refer to as the Sequencer algorithm. In the Sequencer algorithm, we consider two aspects of the relationship between $topCoinc^t$ and a given stored sequence, Seq^n: 1) the *position* of $topCoinc^t$ in Seq^n (zero if $topCoinc^t \notin Seq^n$), referred to as the "sequencer state", and 2) the cumulative slope formed by the history of sequencer states for Seq^n. Thus, if Seq^n is four coincidences in length, and each successive $topCoinc^t$ matches each coincidence in Seq^n, then the history of sequencer states will be the series: {1, 2, 3, 4}, with each transition having a slope of 1.0. We use a vector called *seqState* to store the sequencer states, and a vector called *seqSlope* to store the cumulative slope for each sequence, formed by the history of sequencer states. The slope is calculated as follows:

$$seqSlope^t[i] = \frac{1}{seqState^t[i] - seqState^{t-1}[i]} \tag{8}$$

$$seqSlope^t[i] = \begin{cases} seqSlope^{t-1}[i] - seqSlope^t[i], & i = 1.0 \\ seqSlope^{t-1}[i] - |seqSlope^t[i]|, & i \neq 1.0 \end{cases} \tag{9}$$

$$seqSlope^t[i] = \frac{2}{1 + e^{-seqSlope^t[i]}} - 1 \tag{10}$$

where $seqState^t[i]$ indicates the position of $topCoinc^t$ in sequence i (zero if non-member). The sigmoid scaling performed in Equation 10 helps to constrain the cumulative slope values. Figure 6 shows an example of using cumulative sequence slopes to reveal the best sequence.

At levels above L1, we only update the *seqSlope* vector when *change* = 1, in order to help the TP learn at a time scale appropriate to its level in the hierarchy. A node parameter, *slopeThresh*, is used to determine the minimum slope required for the TP to pass onto the inference stage *without* adding a new sequence or extending an existing sequence. If the maximum value in *seqSlope* does not exceed the value of *slopeThresh*, then either a new sequence is created, or an existing sequence extended.

Generally, we allow only one occurrence of any given coincidence in a single sequence at all levels above L1, though any number of sequences may share that coincidence. This is done to avoid building long sequences at the bottom of the hierarchy, thus dividing the construction of longer sequences across the different levels. We allow consecutive repetitions of coincidences at L1, but do not allow non-consecutive repetitions. This is a musical consideration, given the frequent use of repetitions in musical language.

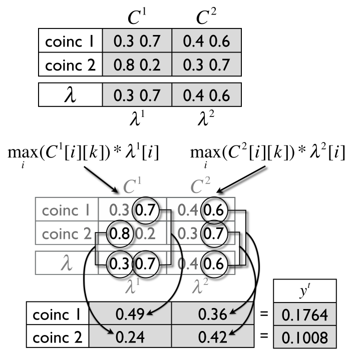
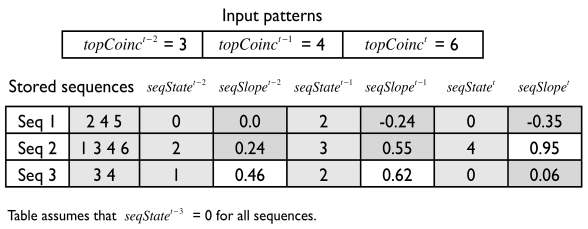


Figure 5. SP inference calculations above L1.



Input patterns

$topCoinc^{t-2}$ = 3	$topCoinc^{t-1}$ = 4	$topCoinc^{t}$ = 6

Stored sequences		$seqState^{t-2}$	$seqSlope^{t-2}$	$seqState^{t-1}$	$seqSlope^{t-1}$	$seqState^{t}$	$seqSlope^{t}$
Seq 1	2 4 5	0	0.0	2	-0.24	0	-0.35
Seq 2	1 3 4 6	2	0.24	3	0.55	4	0.95
Seq 3	3 4	1	0.46	2	0.62	0	0.06

Table assumes that $seqState^{t-3}$ = 0 for all sequences.

Figure 6. Using *seqSlope* to find the best sequence.

5.1.3 Inference in the Temporal Pooler

The objective of TP inference is to determine the likelihood that $topCoinc^t$ is a member of a given stored sequence. At each time step, the TP uses the *counts* vector, from the SP, to update a Conditional Probability Table, called the *weights* matrix, which indicates the probability of a specific coincidence occurring in a given sequence. The *weights* matrix is calculated as:

$$weights[i,j] = counts[j] \times I_{i,j} \,/\, (\sum_{i=1}^{k} counts[k] \times I_{i,j}) \tag{11}$$

$$I_{i,j} = \begin{cases} 1, & C[j] \in S[i] \\ 0, & C[j] \notin S[i] \end{cases}$$

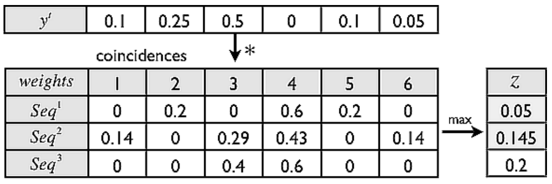
where $C[j]$ is the j^{th} stored coincidence and $S[i]$ is the i^{th} stored sequence.

The probabilities stored by the *weights* matrix are used during TP inference, and also when forming top-down beliefs in the hierarchy, as introduced in Section 2. It is a row-normalized matrix where rows represent sequences and columns represent coincidences. Because the *counts* vector maintains its histogram of $topCoinc^t$ occurrences over a limited temporal window, the *weights* matrix in the HSMM is able to act as a form of short-term memory for the node.

The output of TP inference is the bottom-up belief vector z, which indicates the degree of membership of $topCoinc^t$ in each of the stored sequences. The argmax of z thus identifies the sequence most strongly believed to be active, given $topCoinc^t$. To calculate z, we use a variant of the "sumProp" and "maxProp" algorithms used in the HTM [6], which we refer to as *p*MaxProp. The algorithm uses the *weights* matrix to calculate a belief distribution over sequences, as follows:

$$z[i] = \max_{j=1}^{i}(weights[i,j] \times y[j]) \tag{12}$$

An example run of the *p*MaxProp algorithm is given in Figure 7, using the coincidences and sequences from Figure 6. Because the *weights* matrix in the HSMM is a



y^t	0.1	0.25	0.5	0	0.1	0.05

coincidences ↓*

weights	1	2	3	4	5	6
Seq^1	0	0.2	0	0.6	0.2	0
Seq^2	0.14	0	0.29	0.43	0	0.14
Seq^3	0	0	0.4	0.6	0	0

max →

z
0.05
0.145
0.2

Figure 7. The *p*MaxProp algorithm calculations.

short-term memory, and the *p*MaxProp algorithm is a "one-shot" inference, with no consideration of the previous time step, we combine the results of *p*MaxProp with the results of the Sequencer algorithm, to yield the final bottom-up belief vector:

$$z^t[i]=\frac{z^t[i]+seqSlope^t[i]}{2} \qquad (13)$$

5.2 Belief Formation in an HSMM Node

The final belief vector to be calculated, a belief distribution over coincidences called Bel^C, represents the combination of the node's classification of a given input, and its prediction regarding that input in the current temporal context. Thus, for every bottom-up input there is a top-down, feedback response. Bottom-up vector representations passing between nodes are denoted with λ, while top-down, feedback representations are denoted with π[2]. A schematic of node processing can be seen in Figure 8. The top-down, feedback calculations used in the HSMM are the same as those used in the HTM, but we outline them here for completeness.

The first step in processing the top-down message is to divide the top-down parent belief *π* by the node's bottom-up belief λ (at the top of the hierarchy, the bottom-up belief *z* is used for the top-down calculations):

$$\pi'[i]=\pi[i]\,/\,\lambda[i] \qquad (14)$$

Next, the *π*' vector is used to calculate the top-down belief distribution over stored coincidences as:

$$y^{\downarrow}[i]=\max_{Seq_i\in S}(weights^T[i,j]\times\pi'[j]) \quad for\ i=0\ to\ |C|-1 \qquad (15)$$

where $weights^T[i,j]$ is the transposed *weights* matrix, and $y^{\downarrow}$ is the top-down belief over coincidences, and *S* is the table of stored sequences. Figure 9 gives an example, assuming the coincidences and sequences from Figure 7.

The Bel^C vector is then calculated as the product of the top-down ($y^{\downarrow}$) and bottom-up ($y^{\uparrow}$) belief distributions over coincidences:

$$Bel^C[i]=y^{\downarrow}[i]\times y^{\uparrow}[i] \qquad (16)$$

This calculation ensures that the belief of the node is always based on the available evidence both from *above* and *below* the node's position in the hierarchy. At all levels above L1, the top-down output of the node (the message sent to the children) is calculated using the Bel^C vector and the table of stored coincidences *C*:

$$\pi[i]=\operatorname*{argmax}_{C[i]\in C}(C[i]\times Bel^C[j]) \quad for\ i=0\ to\ |C|-1 \qquad (17)$$

[2]At the node level, the λ and *z* vectors are equivalent. The naming is intended to distinguish the between-node processes from the within-node processes.

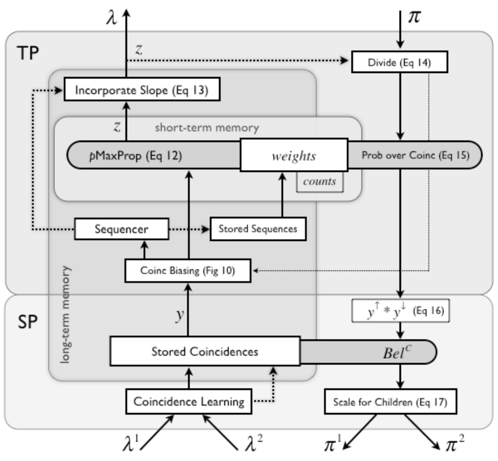
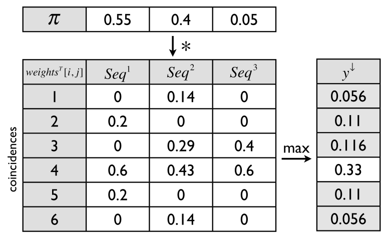


Figure 8. HSMM node processing.

This calculation ensures that each child portion of the top-down output is proportional to the belief in the node. In cases where the parent node has two or more children, the *π* vector is divided into segments of length equal to the length of each child's *λ* vector (i.e., reversing the concatenation of child messages used during bottom-up processing). The various stages of top-down processing are illustrated on the right side of Figure 8.

One extra step, in support of TP inference in the HSMM, is added that is not present in the HTM. In accordance with the ideas of the MPF, it seemed intuitively clear to us that predictions could be used locally in the TP to support the learning process by disambiguating SP inputs whenever possible. With this in mind we added a calculation to the TP inference algorithm that biases the SP belief vector, *y*, according to the state of the *change* flag, and the current top-down inference over sequences. In cases where the sequence inferred by top-down processing at time *t*-1 contains $topCoinc^{t-1}$, and *change* = 0, the belief value for $topCoinc^{t-1}$ is strengthened. However, when *change* = 1, belief in the *next* coincidence in the inferred sequence is strengthened. The algorithm is given in Figure 10. Thus, when the state of the node appears to be changing, belief is biased slightly toward what is most *likely* to occur, whereas when the state appears to be stable, the most recent belief is assumed to be correct.



π	0.55	0.4	0.05

↓*

coincidences	$weights^T[i,j]$	Seq^1	Seq^2	Seq^3
	1	0	0.14	0
	2	0.2	0	0
	3	0	0.29	0.4
	4	0.6	0.43	0.6
	5	0.2	0	0
	6	0	0.14	0

max →

$y^{\downarrow}$
0.056
0.11
0.116
0.33
0.11
0.056

Figure 9. Using the *weights* matrix to calculate the top-down belief over coincidences.

6. DISCUSSION AND CONCLUSION

The strength of the HSMM lies in its balancing of hierarchical interdependence with node-level independence. Each node learns in the context of the hierarchy as a whole, but also forms its own representations and beliefs over a particular level of musical structure. At L1, simple motivic patterns can be recognized and/or generated, and at the higher levels, larger musical structures like phrases, melodies, and sections can also be learned, classified, and generated. Further, since nodes above L1 all process information in an identical manner, and only require a single child, additional high-level nodes could be added, enabling the learning of higher levels of formal structure —songs, movements, compositions. Each node can be monitored independently, and its state exploited for compositional, analytical, or musicological purposes. Composition tools could be developed, offering various levels of interactivity, while maintaining stylistic continuity with the user's musical language. In the area of classic Music Information Retrieval, low levels of the HSMM could be used to identify common motivic gestures among a given set of works, while higher levels could be used to recognize the music of individual composers, or to cluster a number of works by stylistic similarity.

topSeq^{t-1}	The sequence inferred by top-down processing
predCoinc	The predicted coincidence

```
For each coincidence $c$ in $topSeq^{t-1}$
  If $topSeq^{t-1}[c]$ equals $topCoinc^{t-1}$
    Set $predCoinc$ to $topSeq^{t-1}[c+1]$

If $change = 0$
  $y[topCoinc^{t-1}] = y[topCoinc^{t-1}] * 1.1$
else if $change = 1$
  $y[predCoinc] = y[predCoinc] * 1.1$
```

Figure 10. Biasing the predicted coincidence.

The HSMM exploits short-term and long-term memory structures, and uses an explicit sequencing model to build its temporal hierarchies, thus giving it the capacity to learn high-level temporal structure without the tree-like topologies required by HTM networks.

Tremendous progress has been made in the cognitive sciences and cognitive modeling, but such work has remained largely unexplored by the computer music community, which has focused more on pure computer science and signal processing. The HSMM offers a first step toward the development and exploitation of a realistic cognitive model for the representation of musical knowledge, and opens up a myriad of areas for exploration with regard to the associated cognitive behavior.

7. FUTURE WORK

A working prototype of the HSMM has been implemented, and initial tests have shown great promise. A future paper will cover the evaluation in detail, with an emphasis is on exploiting the strengths offered by the cognitive model.

We are interested in exploring alternative distance metrics for the L1 nodes—particularly the pitch and rhythm nodes, where more musically-grounded metrics may be effective. We are also interested in exploring different topologies for the hierarchy, in particular, topologies that isolate individual voices and allow the system to learn both independent monophonic hierarchies and associative polyphonic hierarchies. Along similar lines, we would like to explore the possibilities offered by state-based gating of individual nodes in more complex hierarchies, in order to simulate the cognitive phenomenon of attention direction.

8. REFERENCES

[1] D. Cope: *Computer Models of Musical Creativity.* 87-123, 226-242, MIT Press, Cambridge, MA, 2005.

[2] S. Dubnov, G. Assayag, and O. Lartillot: “Using Machine-Learning Methods for Musical Style Modelling,” *Computer*, 36/10, 2003.

[3] D. Eck and J. Schmidhuber: “Learning the Long-Term Structure of the Blues,” *Lecture Notes in Computer Science*, Vol. 2415, Springer, Berlin, 2002.

[4] D. George. “How the Brain Might Work: A Hierarchical and Temporal Model for Learning and Recognition,” PhD Thesis, Stanford University, Palo Alto, CA, 2008.

[5] D. George and J. Hawkins: “A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex,” Redwood Neuroscience Institute, Menlo Park, CA, 2004.

[6] J. Hawkins and S. Blakeslee: *On Intelligence*, Times Books, New York, NY, 2004.

[7] K. Hirata and T. Aoyagi: “Computational Music Representation Based on the Generative Theory of Tonal Music and the Deductive Object-Oriented Database,” *Computer Music Journal*, 27/3, 2003.

[8] D. Hörnel and W. Menzel: “Learning Music Structure and Style with Neural Networks,” *Computer Music Journal*, 22/4, 1998.

[9] F. Lerdhal and R. Jackendoff: *A Generative Theory of Tonal Music*, MIT Press, Cambridge, MA, 1983.

[10] J. Maxwell and A. Eigenfeldt: “The MusicDB: A Music Database Query System for Recombinance-based Composition in Max/MSP,” *Proceedings of the 2008 International Computer Music Conference*, Belfast, Ireland, 2008.

[11] M.C. Mozer: “Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing,” *Connection Science*, 6/2-3, 1994.

[12] E. Pollastri and G. Simoncelli: “Classification of Melodies by Composer with Hidden Markov Models,” *Proceedings of the First International Conference on WEB Delivering of Music*, 2001.

[13] Y. Uwabu, H. Katayose and S. Inokuchi: “A Structural Analysis Tool for Expressive Performance,” *Proceedings of the International Computer Music Conference,* San Fransisco, 1997.

[14] K. Verburgt, M Dinolfo and M. Fayer: “Extracting Patterns in Music for Composition via Markov Chains,” *Lecture Notes in Computer Science*, Springer, Berlin, 2004

[15] G. Wilder: “Adaptive Melodic Segmentation and Motivic Identification,” *Proceedings of the International Computer Music Conference*, Belfast, Ireland, 2008.

ADDITIONS AND IMPROVEMENTS TO THE ACE 2.0 MUSIC CLASSIFIER

Jessica Thompson
Music Technology
McGill University
jessica.thompson@mail.mcgill.ca

Cory McKay
CIRMMT
McGill University
cory.mckay@mail.mcgill.ca

John Ashley Burgoyne
CIRMMT
McGill University
ashley@music.mcgill.ca

Ichiro Fujinaga
CIRMMT
McGill University
ich@music.mcgill.ca

ABSTRACT

This paper presents additions and improvements to the Autonomous Classification Engine (ACE), a framework for using and optimizing classifiers. Given a set of feature values, ACE experiments with a variety of classifiers, classifier parameters, classifier ensembles and dimensionality-reduction techniques in order to arrive at a configuration that is well-suited to a given problem. Changes and additions have been made to ACE in order to increase its functionality as well as to make it easier to use and incorporate into other software frameworks. Details are provided on ACE's remodeled class structure and associated API, the improved command line and graphical user interfaces, a new ACE XML 2.0 ZIP file format and expanded statistical reporting associated with cross validation. The resulting improved processing and methods of operation are also discussed.

1. INTRODUCTION

Automatic classification techniques play an essential role in many music information retrieval (MIR) research areas. These include genre classification, mood classification, music recommendation, performer identification, composer identification, and instrument identification, to name just a few. Classification software especially adapted to MIR can be of significant benefit. Some important work has already been done in this area, as noted in Section 2. ACE, which is part of the jMIR software suite described below, is a framework that builds upon these systems and adds additional functionality that is generally lacking in both systems designed specifically for music classification as well as general classification systems. ACE 2.0, which is presented in this paper, has been significantly improved since the release in 2005 of ACE 1.1 [1]. Improvements include an entirely restructured and simplified API, a significantly improved command-line interface, a new GUI, new file formats, improved processing and significantly expanded statistical reporting.

1.1 jMIR

jMIR [2] is a suite of software tools developed for use in MIR research. The jMIR components can be used either independently or as an integrated suite. Although the components can read and write to common file formats such as Weka ARFF, jMIR also uses its own ACE XML file formats that offer a number of advantages over alternative data-mining formats [1, 3].

jMIR was designed to provide:

- a flexible set of tools that can easily be applied to a wide variety of MIR-oriented research tasks;
- a platform that can be used to combine research on symbolic, audio and/or cultural data;
- easy-to-use and accessible software with a minimal learning curve that can be used by researchers with little or no technological training;
- a modular and extensible framework for iteratively developing and sharing new feature extraction and classification technologies; and
- software that encourages collaboration between different research centers by facilitating the sharing of research data using powerful and flexible file formats [4].

jMIR is the only existing software suite that combines a meta-learning component (ACE) into an integrated framework with three different types of musical feature extractors, a metadata correction tool, and ground truth data. jMIR is also the only unified MIR research framework that combines all three of symbolic, audio, and cultural features.

1.2 ACE XML

ACE XML [1, 3] is a set of file formats developed to enable communication between the various jMIR software components, including ACE. These file formats have been designed to be very flexible and expressive. It is hoped that the MIR research community will eventually adopt them as multi-purpose standardized formats, beyond the limited scope of jMIR. ACE XML has recently been significantly revised and expanded in order to help make this possible [3].

1.3 ACE

ACE is a meta-learning classification system that can automatically experiment with a variety of different dimensionality-reduction and machine-learning algorithms in order to evaluate which ones are best suited to particular problems. ACE can also be used as a simple automatic classification system. ACE is open source and available for free. It is implemented entirely in Java in order to maximize portability.

ACE is built on the standardized Weka machine-learning infrastructure [5] and makes direct use of a variety of algorithms and data structures distributed with Weka. This means not only that new algorithms produced by the very active Weka community can be incorporated into ACE immediately, but also that new algorithms specifically designed for MIR research can be developed using the Weka framework. ACE can read features stored in either ACE XML or Weka ARFF files.

Two Weka data structures of particular interest that are used by ACE and referred to in this paper are the Weka `Instances` object, which stores a set of instances in a representation similar to the Weka ARFF file, and the Weka `Classifer` object, which classifies a set of Weka `Instances` with a specified classification algorithm.

2. RELATED WORK

There are a number of existing software packages that are often used for machine learning, including Weka [5], PRTools [6] and several other MATLAB [7] toolboxes. There are also several systems that offer meta-learning functionality, including RapidMiner (formerly Yale) [8] and METAL [9]. All of these are general purpose systems, however, and do not meet some of the special needs of MIR, as discussed in [1]. ACE and ACE XML make it possible to represent and use types of information that are particularly relevant to MIR but are not expressible or usable in most alternative systems. For example, jMIR and ACE XML have the ability to:

- maintain logical groupings between multi-dimensional features;
- represent class labels and feature values for potentially overlapping sub-sections of instances as well as for instances as a whole;
- represent structured class ontologies; and
- associate multiple classes with a single instance.

There are also several high-quality toolsets that have been designed specifically for MIR, but they tend to offer less sophisticated processing specifically with respect to machine learning. MIRtoolbox [10] is a powerful modular MATLAB toolbox for designing and extracting audio features. The well-known CLAM [11] and Marsyas [12] focus on audio-related tasks. The M2K [13] graphical patching interface can be used to connect a range of different MIR processing components in ways that can take advantage of distributed processing.

3. IMPROVEMENTS NEW TO ACE 2.0

3.1 Architectural Restructuring

ACE's class structure has been redesigned to be more flexible, extensible, and easy to understand. This redesign is intended to facilitate integration with other software. ACE's main functionality is accessed through an interface class called `Coordinator`. Figure 1 illustrates this organization: the GUI, command-line interface, and external software all only directly access this new `Coordinator` class, which then communicates with ACE's processing classes. This organization ensures that all processing is performed identically, regardless of the source of the request, and makes ACE easier to use. New users wishing to use the ACE API need only understand the `Coordinator` class in order to be able to use all of ACE's functionality. The remainder of this section presents ACE's main classes.

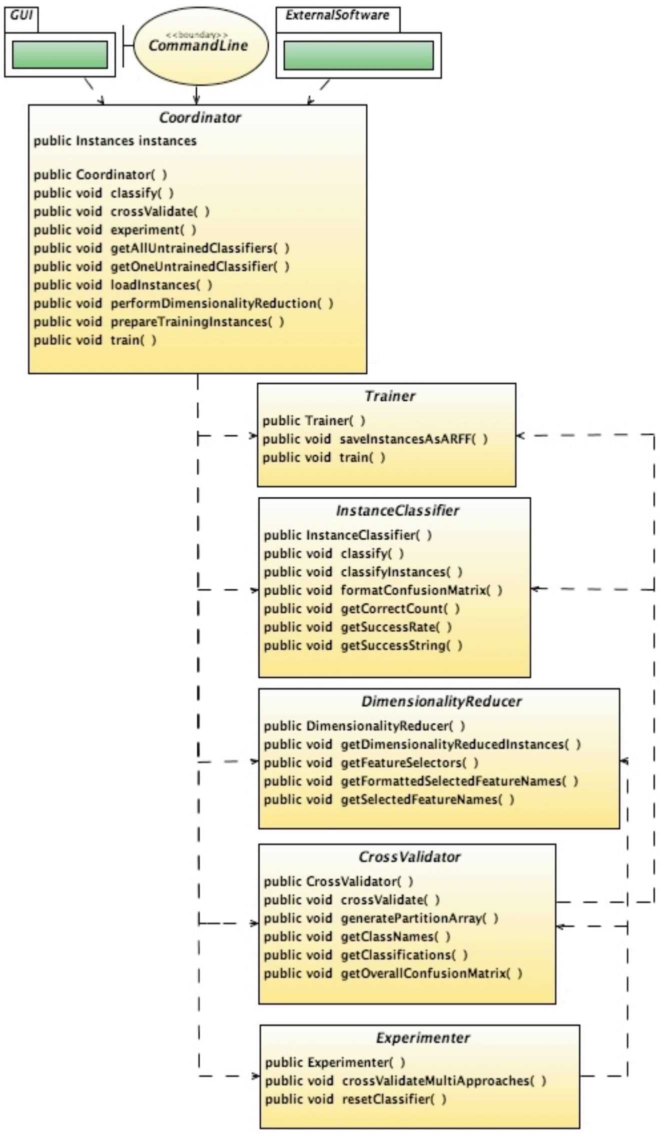

GUI
<<boundary>>
CommandLine
ExternalSoftware
Coordinator
public Instances instances
public Coordinator( )
public void classify( )
public void crossValidate( )
public void experiment( )
public void getAllUntrainedClassifiers( )
public void getOneUntrainedClassifier( )
public void loadInstances( )
public void performDimensionalityReduction( )
public void prepareTrainingInstances( )
public void train( )
Trainer
public Trainer( )
public void saveInstancesAsARFF( )
public void train( )
InstanceClassifier
public InstanceClassifier( )
public void classify( )
public void classifyInstances( )
public void formatConfusionMatrix( )
public void getCorrectCount( )
public void getSuccessRate( )
public void getSuccessString( )
DimensionalityReducer
public DimensionalityReducer( )
public void getDimensionalityReducedInstances( )
public void getFeatureSelectors( )
public void getFormattedSelectedFeatureNames( )
public void getSelectedFeatureNames( )
CrossValidator
public CrossValidator( )
public void crossValidate( )
public void generatePartitionArray( )
public void getClassNames( )
public void getClassifications( )
public void getOverallConfusionMatrix( )
Experimenter
public Experimenter( )
public void crossValidateMultiApproaches( )
public void resetClassifier( )

Figure 1. Structure of ACE's main processing classes. The `Coordinator` class provides an exclusive interface through which ACE's main functionality can be accessed. Arrows indicate interactions between classes. All public methods are listed, but parameters are omitted from method declarations to save space.

- `Coordinator`: The class provides the interface through which ACE's training, classification, cross validation, and experimentation functionality can be accessed. Only `Coordinator` calls the classes listed below; all other sources need only call the appropriate methods in `Coordinator` to access the functionality of all other processing classes. Loading and preparation of instances as well as dimensionality reduction is performed in this class prior to passing instances to processing classes.
- `Trainer`: Trains a specified type of Weka `Classifier` based on the given training instances. The trained Weka `Classifier` is stored and saved in an ACE `TrainedModel` object.
- `InstanceClassifier`: Classifies a set of instances using a trained Weka `Classifier`. This class reads the `TrainedModel` object from a specified file and uses it to classify the given instances. In the context of cross validation, a classified Weka `Instances` object is returned. Classifications can be written to a Weka ARFF file or an ACE XML Instance Label file.
- `DimensionalityReducer`: Reduces the dimensionality of the features extracted from a set of instances. This class is called by the `Coordinator` class to reduce the dimensionality of the training data prior to training and cross-validation in order to help avoid the "curse of dimensionality." This class is also called by the `Experimenter` class to create an array of multiple dimensionality-reduced versions of an original set of instances.
- `CrossValidator`: Cross-validates the given instances with the specified type of Weka `Classifier` using the specified number of partitions. Instances are partitioned randomly into training and testing data for each fold. `CrossValidator` makes calls to the `Trainer` and `InstanceClassifier` classes to evaluate the performance of a specific classification approach. The specified type of Weka `Classifier` is trained on the remaining training data and tested on the testing data for each partition. Statistics are stored for each partition and used to generate performance reports that provide much more statistical detail than Weka itself provides.
- `Experimenter`: Tests to find the best performing classification approach by making repeated calls to the `CrossValidator` class using different parameters each time. Different types of classifiers are tested with different types of dimensionality reduction. `Experimenter` calls `DimensionalityReducer` to get an array of Weka `Instances` objects, wherein each cell contains a different dimensionality-reduced version of the original instances.

  Each type of classifier is cross-validated with each set of dimensionality-reduced instances. A summary of the results for each cross-validation experiment for each dimensionality-reduction experiment is generated, as well as more detailed results when requested by the user. After the best classification methodology has been selected, validation is performed using a publication set put aside at the beginning of the experiment. A new Weka `Classifier` of the chosen type is created and trained on the chosen type of dimensionality-reduced instances (all instances are now available for use as training data, except for the publication set). The newly trained Weka `Classifier` is tested on the publication set and the results are saved.

3.2 Redesigned Cross Validation

ACE performs full meta-learning with training, testing, and publication data sets. Previously, cross-validation was performed using the Weka API, but now, ACE implements its own cross-validation that improves upon Weka's. This new implementation, contained in the `CrossValidator` class, includes output of additional statistics and more transparent data processing. Whereas previously Weka's cross-validation only allowed access to overall correctness statistics and confusion matrices, ACE's new implementation includes variances across partitions, individual instance classification results for each partition, confusion matrices for each partition, and data on running times.

3.3 ACE XML 2.0 ZIP and Project Files

ACE XML, the file format used to transmit information between the jMIR components, consists of four different file types for storing, respectively, extracted feature values, feature metadata, labeled instances and class ontologies. Although the separation of this data into four different types of files does have significant advantages [4], large projects consisting of multiple files can become unwieldy.

The new ACE XML 2.0 Project and ZIP files present solutions to this problem. The Project file allows users to associate ACE XML files together so that they may be automatically saved or loaded together, and the ZIP format makes it possible to package all files referred to in a Project file into a single compressed ZIP file.

The ACE XML ZIP file is implemented using an ACE XML Project file and a hidden text file with the extension ".sp". This file contains only one line of text that specifies the name of the single ACE XML Project file compressed within the ACE XML ZIP file. When ACE parses an ACE XML ZIP file, it looks for the .sp file first, and then parses the associated ACE XML Project file so that the other contents of the ACE XML ZIP file can be properly interpreted.

ACE includes utilities for creating, accessing, and managing ACE XML ZIP files. When ACE unzips an ACE XML ZIP file, it rewrites the ACE XML Project file to reflect the new path names of the newly unzipped files. ACE can also extract or add a single file from/to an ACE XML ZIP file. An ACE XML ZIP file can be used to load or save an ACE project via the ACE command-line interface, GUI or API.

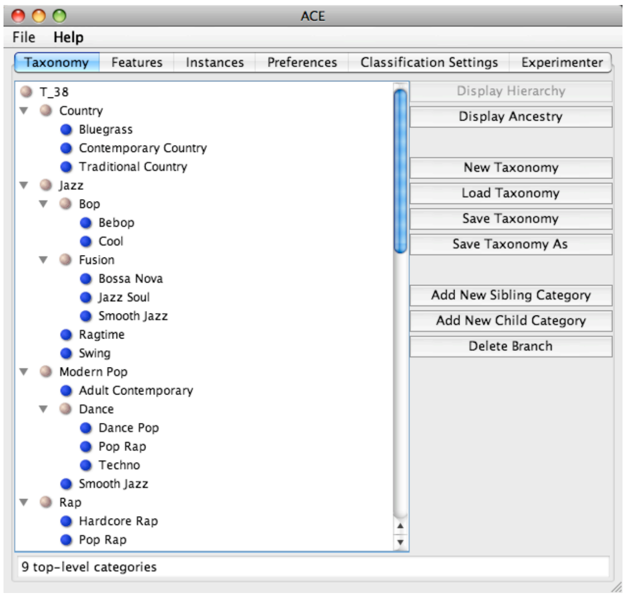
3.4 Improved Command-Line Interface

The previous ACE command-line interface has been entirely redesigned with clearer and more intuitive commands. The command-line interface of software such as ACE is particularly important, as it is often needed to perform batch processing that can last days or weeks. Running ACE from the command line has become easier with the addition of new functionality such as the ability to load an ACE project from an ACE XML Project file or an ACE ZIP file. The user can also now specify the type of classifier or dimensionality-reduction algorithm to be used as well as other options related to the distribution of datasets (e.g., randomization, maximum class membership, and maximum class spread). With the *verbose* option, the user also has the option of printing a more detailed report of the performed processing. These improvements to the command-line interface not only make ACE easier to use, but also provide more precise control of ACE's processing.

3.5 Graphical User Interface

ACE also now includes functionality for GUI-based viewing, editing, and saving of ACE XML files. This functionality is divided between three panes: the *Taxonomy* pane, which displays the contents of an ACE XML Class Ontology file; the *Features* pane, which displays the contents of an ACE XML Feature Description file; and the *Instances* pane, which displays the combined contents of both ACE XML Feature Value files and ACE XML Instance Label files.

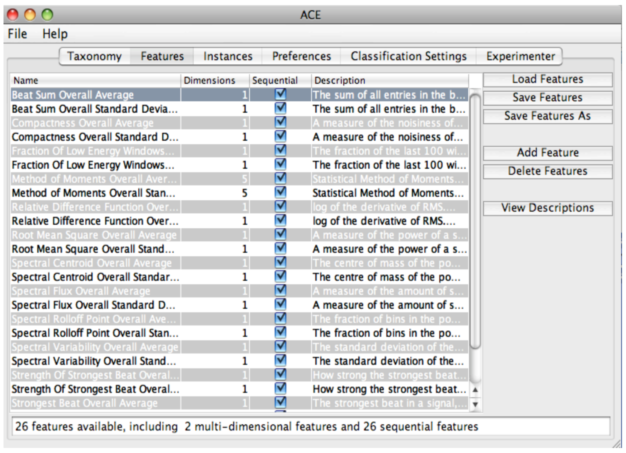
A screen shot of the *Taxonomy* pane is shown in Figure 2. The displayed structure indicates a genre taxonomy for use in an automatic genre classification task. If an ACE XML Instance Label file is loaded without explicitly specifying such a class ontology either manually or with an ACE XML Class Ontology file, then a flat ontology is automatically generated based on the labels used in the ACE XML Instance Label file, and is displayed in the *Taxonomy* pane. Figure 3 shows a *Features* pane displaying a list of audio features. If an ACE XML Feature Descriptions file is not loaded here prior to loading an ACE XML Feature Values file, feature descriptions are generated automatically based on the features present in the ACE XML Feature Values file. Figure 4 shows how the *Instances* pane can be used to display class labels that have been associated with particular instances.



- T_38
  - Country
    - Bluegrass
    - Contemporary Country
    - Traditional Country
  - Jazz
    - Bop
      - Bebop
      - Cool
    - Fusion
      - Bossa Nova
      - Jazz Soul
      - Smooth Jazz
    - Ragtime
    - Swing
  - Modern Pop
    - Adult Contemporary
    - Dance
      - Dance Pop
      - Pop Rap
      - Techno
    - Smooth Jazz
  - Rap
    - Hardcore Rap
    - Pop Rap

9 top-level categories

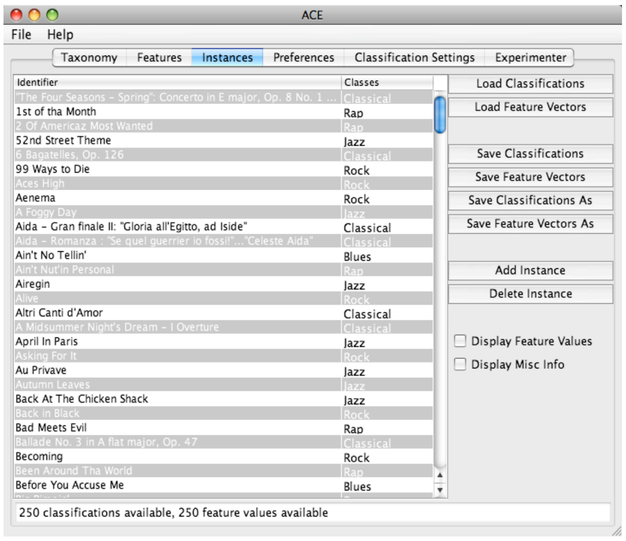
Figure 2. A sample genre ontology displayed in the *Taxonomy* pane of the ACE GUI.



Name	Dimensions	Sequential	Description
Beat Sum Overall Average	1	✓	The sum of all entries in the b...
Beat Sum Overall Standard Devia...	1	✓	The sum of all entries in the b...
Compactness Overall Average	1	✓	A measure of the noisiness of...
Compactness Overall Standard D...	1	✓	A measure of the noisiness of...
Fraction Of Low Energy Windows...	1	✓	The fraction of the last 100 wi...
Fraction Of Low Energy Windows...	1	✓	The fraction of the last 100 wi...
Method of Moments Overall Aver...	5	✓	Statistical Method of Moments...
Method of Moments Overall Stan...	5	✓	Statistical Method of Moments...
Relative Difference Function Over...	1	✓	log of the derivative of RMS....
Relative Difference Function Over...	1	✓	log of the derivative of RMS....
Root Mean Square Overall Average	1	✓	A measure of the power of a s...
Root Mean Square Overall Stand...	1	✓	A measure of the power of a s...
Spectral Centroid Overall Average	1	✓	The centre of mass of the po...
Spectral Centroid Overall Standar...	1	✓	The centre of mass of the po...
Spectral Flux Overall Average	1	✓	A measure of the amount of s...
Spectral Flux Overall Standard D...	1	✓	A measure of the amount of s...
Spectral Rolloff Point Overall Ave...	1	✓	The fraction of bins in the po...
Spectral Rolloff Point Overall Stan...	1	✓	The fraction of bins in the po...
Spectral Variability Overall Average	1	✓	The standard deviation of the...
Spectral Variability Overall Stand...	1	✓	The standard deviation of the...
Strength Of Strongest Beat Overal...	1	✓	How strong the strongest beat...
Strength Of Strongest Beat Overal...	1	✓	How strong the strongest beat...
Strongest Beat Overall Average	1	✓	The strongest beat in a signal,...

26 features available, including 2 multi-dimensional features and 26 sequential features

Figure 3. The *Features* pane of the ACE GUI displaying metadata about a set of audio features.



Identifier	Classes
"The Four Seasons – Spring": Concerto in E major, Op. 8 No. 1 ...	Classical
1st of tha Month	Rap
2 Of Americaz Most Wanted	Rap
52nd Street Theme	Jazz
6 Bagatelles, Op. 126	Classical
99 Ways to Die	Rock
Aces High	Rock
Aenema	Rock
A Foggy Day	Jazz
Aida – Gran finale II: "Gloria all'Egitto, ad Iside"	Classical
Aida – Romanza : "Se quel guerrier io fossi!"..."Celeste Aida"	Classical
Ain't No Tellin'	Blues
Ain't Nut'in Personal	Rap
Airegin	Jazz
Alive	Rock
Altri Canti d'Amor	Classical
A Midsummer Night's Dream – I Overture	Classical
April In Paris	Jazz
Asking For It	Rock
Au Privave	Jazz
Autumn Leaves	Jazz
Back At The Chicken Shack	Jazz
Back in Black	Rock
Bad Meets Evil	Rap
Ballade No. 3 in A flat major, Op. 47	Classical
Becoming	Rock
Been Around Tha World	Rap
Before You Accuse Me	Blues

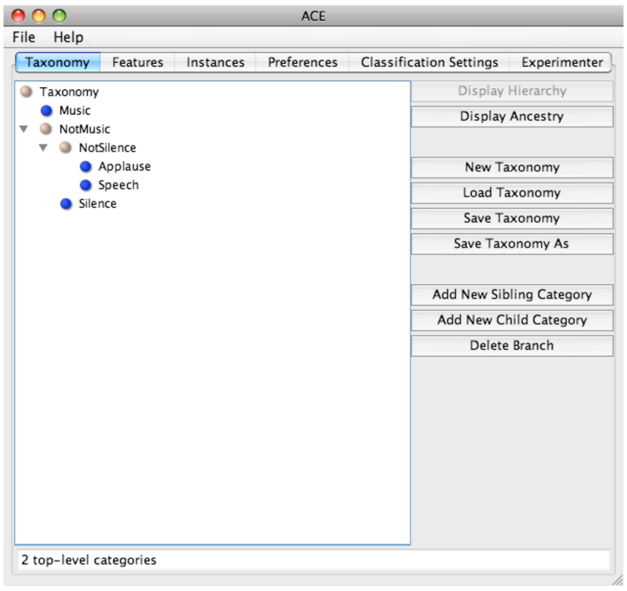
250 classifications available, 250 feature values available

Figure 4. The *Instances* pane of the ACE GUI. Song titles are associated in this example with particular genre

labels drawn from the genre ontology shown in Figure 2. Note that neither the *Display Feature Values* nor the *Display Misc Info* checkboxes are checked, so only instance identifiers and classes are displayed in this particular example.

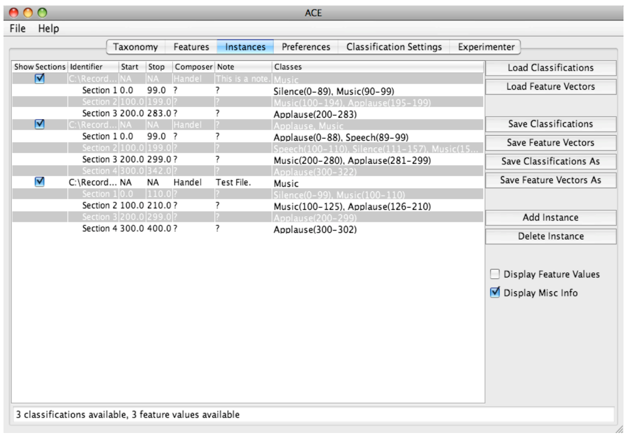
Figures 5, 6, and 7 illustrate a more complex example. Figure 5 establishes a new class ontology, in this case a hierarchical music–speech–applause–silence discriminator. Figure 6 shows how the *Instances* pane can display not only class labels, but also miscellaneous metadata. This figure also demonstrates that instances can be broken into separately-labeled subsections and that each instance or subsection may be associated with multiple class labels. Start and stop times indicate the boundaries of the subsections. Figure 7 demonstrates how feature values can also be displayed using the *Instances* pane. It shows the same data as Figure 6, except that feature values are displayed and miscellaneous metadata is not.

If feature arrays and class labels are loaded for the same subsection, all information for that instance is presented in one row. The specific time of the overlap of class labels within a subsection is indicated within parentheses after the class name. Subsection rows for any overall instance can be hidden by unchecking the *Show Sections* checkbox. The *Composer* and *Note* columns are metadata loaded from the particular Instance Label ACE XML file associated with this example and can be hidden by clicking on the *Display Misc Info* checkbox.



2 top-level categories

Figure 5. Another class ontology displayed in the *Taxonomy* pane of the ACE GUI.

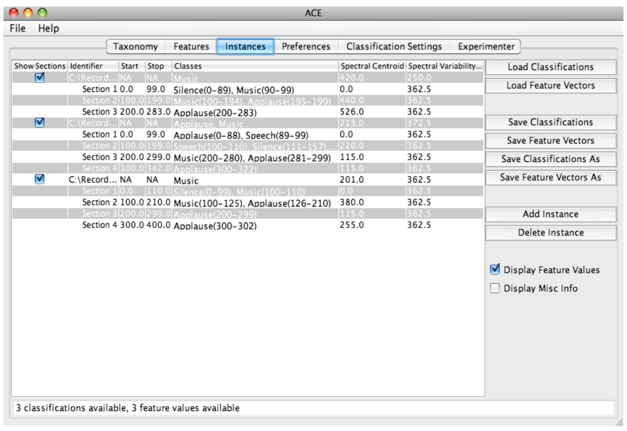


Show Sections	Identifier	Start	Stop	Composer	Note	Classes
✓	C:\Record...	NA	NA	Handel	This is a note.	Music
	Section 1	0.0	99.0	?	?	Silence(0-89), Music(90-99)
	Section 2	100.0	199.0	?	?	Music(100-194), Applause(195-199)
	Section 3	200.0	283.0	?	?	Applause(200-283)
✓	C:\Record...	NA	NA	Handel	?	Applause, Music
	Section 1	0.0	99.0	?	?	Applause(0-88), Speech(89-99)
	Section 2	100.0	199.0	?	?	Speech(100-110), Silence(111-157), Music(15...
	Section 3	200.0	299.0	?	?	Music(200-280), Applause(281-299)
	Section 4	300.0	342.0	?	?	Applause(300-322)
✓	C:\Record...	NA	NA	Handel	Test File.	Music
	Section 1	0.0	110.0	?	?	Silence(0-99), Music(100-110)
	Section 2	100.0	210.0	?	?	Music(100-125), Applause(126-210)
	Section 3	200.0	299.0	?	?	Applause(200-299)
	Section 4	300.0	400.0	?	?	Applause(300-302)

☐ Display Feature Values
☑ Display Misc Info

3 classifications available, 3 feature values available

Figure 6. Instances with subsections and metadata displayed in the *Instances* pane of the ACE GUI.



Show Sections	Identifier	Start	Stop	Classes	Spectral Centroid	Spectral Variability...
✓	C:\Record...	NA	NA	Music	420.0	250.0
	Section 1	0.0	99.0	Silence(0-89), Music(90-99)	0.0	362.5
	Section 2	100.0	199.0	Music(100-194), Applause(195-199)	440.0	362.5
	Section 3	200.0	283.0	Applause(200-283)	526.0	362.5
✓	C:\Record...	NA	NA	Applause, Music	213.0	372.5
	Section 1	0.0	99.0	Applause(0-88), Speech(89-99)	0.0	362.5
	Section 2	100.0	199.0	Speech(100-110), Silence(111-157),...	220.0	362.5
	Section 3	200.0	299.0	Music(200-280), Applause(281-299)	115.0	362.5
	Section 4	300.0	342.0	Applause(300-322)	115.0	362.5
✓	C:\Record...	NA	NA	Music	201.0	362.5
	Section 1	0.0	110.0	Silence(0-99), Music(100-110)	0.0	362.5
	Section 2	100.0	210.0	Music(100-125), Applause(126-210)	380.0	362.5
	Section 3	200.0	299.0	Applause(200-299)	115.0	362.5
	Section 4	300.0	400.0	Applause(300-302)	255.0	362.5

☑ Display Feature Values
☐ Display Misc Info

3 classifications available, 3 feature values available

Figure 7. Instances with subsections and feature values displayed in the *Instances* pane of the ACE GUI.

4. CONCLUSION AND FUTURE WORK

Improvements have been made to ACE since its original publication, including new capabilities that make it a more complete and easy-to-use meta-learning classification framework. ACE is an ongoing project, and further improvements will continue to be made.

4.1 Fully Functional GUI

The ACE GUI currently serves as a tool for viewing and editing ACE XML files. It will eventually be possible to also use the GUI to perform experiments on data sets, as can currently only be done with the command-line interface or API. This functionality will be accessible from two currently unfinished panes: the *Experimenter* pane and the *Preferences* pane. The *Experimenter* pane will allow full access to all of ACE's machine learning functionality. Several sub-panes will be used to display the same output that is printed or saved to files when running experiments from the ACE command-line interface. The *Preferences* pane of the ACE GUI will allow users to specify preferences related to both interface settings and machine learning parameters. User studies will also be performed in order to validate and improve the GUI design.

4.2 Distributed Work Load

Functionality is being built into ACE to allow it to run trials on multiple computers in parallel in order to reducte execution times. Once the distributed aspect of the system is complete, a server-based subsystem will be designed that contains a coordination system and database. Although not necessary for using ACE, users will be able to choose to dedicate computers to this server, allowing ACE to run continually. The server will keep a record of the performances of all ACE operations run on a particular user's cluster and generate statistics for self-evaluation and improvement. ACE will then make use of any idle time to attempt to improve solutions to previously encountered but currently inactive problems. Ultimately, the user would only be required to specify the total time available (typically days or weeks) for ACE to run its experiments and everything else, including the choice of learning algorithms and their parameters, would be automatically determined by ACE.

4.3 Expanded Machine Learning Algorithms

In the future, ACE will include learning schemes important to MIR that are currently missing from the Weka distribution, such as hidden Markov models and recurrent neural networks. Support for Weka's unsupervised learning functionality will also be incorporated. It would also be beneficial to include tools for constructing blackboard systems, in particular those that can integrate knowledge sources based on expert heuristics. Another potentially beneficial addition would be to implement modules for facilitating post-processing. All of these extensions would add to ACE's flexibility and breadth of processing.

5. ACKNOWLEDGEMENTS

The authors would like to thank the *Andrew W. Mellon Foundation* and the *Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)* for their generous financial support, as well as the members of the *Networked Environment for Music Analysis (NEMA)* group for their valuable critiques and suggestions.

6. REFERENCES

[1] McKay, C., R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga. 2005. ACE: A framework for optimizing music classification. *Proceedings of the International Conference on Music Information Retrieval.* 42–9.

[2] McKay, C., and I. Fujinaga. 2009. jMIR: Tools for automatic music classification. *Proceedings of the International Computer Music Conference*

[3] McKay, C., J. A. Burgoyne, J. Thompson, and I. Fujinaga. 2009. Using the ACE XML 2.0 file formats to store and share music classification data. *Proceedings of the International Conference on Music Information Retrieval.*

[4] McKay, C., and I. Fujinaga. 2008. Combining features extracted from audio, symbolic and cultural sources. *Proceedings of the International Conference on Music Information Retrieval*, 597–602.

[5] Witten, I., and E. Frank. 2005. *Data mining: Practical machine learning tools and techniques with Java implementations*. San Francisco: Morgan Kaufmann.

[6] van der Heijden, F., R. P. W. Duin, D. de Ridder, and D. M. J. Tax. 2004. *Classification, parameter estimation and state estimation: An engineering approach using MATLAB.* New York: Wiley.

[7] MathWorks 2008. *MATLAB version 7.6.0.* Natick, Massachusetts: The MathWorks.

[8] Mierswa, I., M. Wrust, R. Klinkenberg, M. Scholz, and T. Euler. 2006. YALE: Rapid prototyping for complex data mining tasks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

[9] Brazdil, P., C. Soares, and J. Costa. 2003. Ranking learning algorithms. *Machine Learning* 50 (3): 251–77.

[10] Lartillot, O., P. Toiviainen, and T. Eerola. 2008. A Matlab toolbox for music information retrieval. In *Data analysis, machine learning and applications,* ed. C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker. New York: Springer. 261–8.

[11] Arumi, P., and X. Amatriain. 2005. CLAM: An object oriented framework for audio and music. *Proceedings of the International Linux Audio Conference.*

[12] Tzanetakis, G., and P. Cook. 1999. MARSYAS: A framework for audio analysis. *Organized Sound* 4 (3): 169–75.

[13] Downie, S., A. Ehmann, and D. Tcheng. 2005. Music-to-knowledge (M2K): A prototyping and evaluation environment for music information retrieval research. *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 676.

A PROBABILISTIC TOPIC MODEL FOR UNSUPERVISED LEARNING OF MUSICAL KEY-PROFILES

Diane J. Hu and Lawrence K. Saul
Department of Computer Science and Engineering
University of California, San Diego
{dhu,saul}@cs.ucsd.edu

ABSTRACT

We describe a probabilistic model for learning musical key-profiles from symbolic files of polyphonic, classical music. Our model is based on Latent Dirichlet Allocation (LDA), a statistical approach for discovering hidden topics in large corpora of text. In our adaptation of LDA, symbolic music files play the role of text documents, groups of musical notes play the role of words, and musical key-profiles play the role of topics. The topics are discovered as significant, recurring distributions over twelve neutral pitch-classes. Though discovered automatically, these distributions closely resemble the traditional key-profiles used to indicate the stability and importance of neutral pitch-classes in the major and minor keys of western music. Unlike earlier approaches based on human judgement, our model learns key-profiles in an unsupervised manner, inferring them automatically from a large musical corpus that contains no key annotations. We show how these learned key-profiles can be used to determine the key of a musical piece and track its harmonic modulations. We also show how the model's inferences can be used to compare musical pieces based on their harmonic structure.

1. INTRODUCTION

Musical composition can be studied as both an artistic and theoretical endeavor. Though music can express a vast range of human emotions, ideas, and stories, composers generally work within a theoretical framework that is highly structured and organized. In western tonal music, two important concepts in this framework are the *key* and the *tonic*. The key of a musical piece identifies the principal set of pitches that the composer uses to build its melodies and harmonies. The key also defines the tonic, or the most stable pitch, and its relationship to all of the other pitches in the key's pitch set. Though each musical piece is characterized by one overall key, the key can be shifted within a piece by a compositional technique known as *modulation*. Notwithstanding the infinite number of variations possible in music, most pieces can be analyzed in these terms.

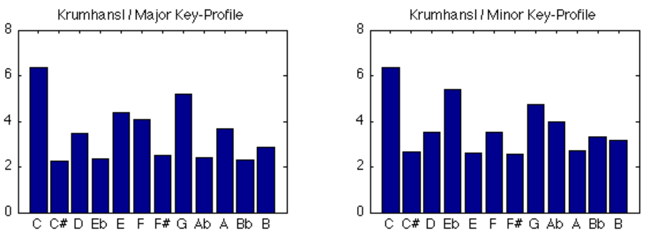


Figure 1. C major (left) and C minor (right) key-profiles proposed by Krumhansl-Kessler (KK), used in the Krumhansl-Schmukler (KS) key-finding algorithm.

Musical pieces are most commonly studied by analyzing their melodies and harmonies. In any such analysis, the first step is to determine the key. While the key is in principle determined by elements of music theory, individual pieces and passages can exhibit complex variations on these elements. In practice, considerable expertise is required to resolve ambiguities.

Many researchers have proposed rule-based systems for automatic key-finding in symbolic music [2,10,12]. In particular, Krumhansl and Schmuckler (KS) [8] introduced a model based on "key-profiles". A key-profile is a twelve-dimensional vector in which each element indicates the stability of a neutral pitch-class relative to the given key. There are 24 key-profiles in total, one for each major and minor key. Using these key profiles, KS proposed a simple method to determine the key of a musical piece or shorter passages within a piece: first, accumulate a twelve-dimensional vector whose elements store the total duration of each pitch-class in a song; second, compute the key-profile that has the highest correlation with this vector. The KS model uses key-profiles derived from probe tone studies conducted by Krumhansl and Kessler (KK) [9]. Figure 1 shows the KK key profiles for C major and C minor; profiles for other keys are obtained by transposition. In recent work [14, 15], these key-profiles have been modified to achieve better performance in automatic key-finding.

In this paper, we show how to learn musical key-profiles automatically from the statistics of large music collections. Unlike previous studies, we take a purely data-driven approach that does not depend on extensive prior knowledge of music or supervision by domain experts. Based on a model of *unsupervised* learning, our approach bypasses the need for manually key-annotated musical pieces, a process that is both expensive and prone to error. As an additional benefit, it can also discover correlations in the data of which the designers of rule-based approaches are unaware. Since we do not rely on prior knowledge, our model can also be applied in a straightforward way to other, non-western genres of music with different tonal systems.

Our approach is based on Latent Dirichlet Allocation (LDA) [1], a popular probabilistic model for discovering latent semantic topics in large collections of text documents. In LDA, each document is described as a mixture of topics, and each topic is characterized by its own particular distribution over words. LDA for text is based on the premise that documents about similar topics contain similar words. Beyond document modeling, LDA has also been adapted to settings such as image segmentation [5], part-of-speech tagging [6], and collaborative filtering [11].

Our variant of LDA for unsupervised learning of key-profiles is based on the premise that musical pieces in the same key use similar sets of pitches. Roughly speaking, our model treats each song as a "document" and the notes in each beat or half-measure as a "word". The goal of learning is to infer harmonic "topics" from the sets of pitches that commonly co-occur in musical pieces. These harmonic topics, which we interpret as key-profiles, are expressed as distributions over the twelve neutral pitch-classes.

We show how to use these key-profiles for automatic key-finding and similarity ranking of musical pieces. We note, however, that our use of key-profiles differs from that of the KS model. For key-finding, the KS model consists of two steps: 1) derive key-profiles and 2) predict keys using key-profiles. In our model, these steps are naturally integrated by the Expectation-Maximization (EM) algorithm [3]. We do not need further heuristics to make key-finding predictions from our key-profiles as the EM algorithm yields the former along with the latter.

2. MODEL

This section describes our probabilistic topic model, first developing the form of its joint distribution, then sketching out the problems of inference and parameter estimation. We use the following terminology and notation throughout the rest of the paper:

1. A *note* $u \in \{\mathrm{A}, \mathrm{A}\sharp, \mathrm{B}, \ldots, \mathrm{G}\sharp\}$ is the most basic unit of data. It is an element from the set of neutral pitch-classes. For easy reference, we map these pitch-classes to integer note values 0 through 11. We refer to $V=12$ as the vocabulary size of our model.

2. A *segment* is a basic unit of time in a song (e.g., a measure). We denote the notes in the nth segment by $\mathbf{u}_n = \{u_{n1}, \ldots, u_{nL}\}$, where $u_{n\ell}$ is the ℓth note in the segment. Discarding the ordering of the notes, we can also describe each segment simply by the number of time each note occurs. We use x_n to denote the V-dimensional vector whose jth element x_n^j counts the number of times that the jth note appears in the nth segment.

3. A *song* s is a sequence of notes in N segments: $s = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$. Discarding the ordering of notes within segments, we can also describe a song by the sequence of count vectors $X = (x_1, \ldots, x_N)$.

4. A music *corpus* is a collection of M songs denoted by $\mathcal{S} = \{s_1, \ldots, s_M\}$.

5. A *topic* z is a probability distribution over the vocabulary of $V = 12$ pitch-classes. Topics model particular groups of notes that frequently occur together within individual segments. In practice, these groupings should contain the principle set of pitches for a particular musical key. Thus, we interpret each topic's distribution over twelve pitch-classes as the key-profile for a musical key. We imagine that each segment in a song has its own topic (or key), and we use $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ to denote the sequence of topics across all segments. In western tonal music, prior knowledge suggests to look for $K = 24$ topics corresponding to the major and minor scales in each pitch-class. Section 2.3 describes how we identity the topics with these traditional key-profiles.

With this terminology, we can describe our probabilistic model for songs in a musical corpus. Note that we do not attempt to model the order of note sequences within a segment or the order of segments within a song. Just as LDA for topic modeling in text treats each document as a "bag of words", our probabilistic model treats each song as a "bag of segments" and each segment as a "bag of notes".

2.1 Generative process

Our approach for automatic key-profiling in music is based on the generative model of LDA for discovering topics in text. However, instead of predicting words in documents, we predict notes in songs. Our model imagines a simple, stochastic procedure in which observed notes and key-profiles are generated as random variables. In addition, we model the key-profiles as latent variables whose values must be inferred by conditioning on observed notes and using Bayes rule.

We begin by describing the process for generating a song in the corpus. First, we draw a topic weight vector that determines which topics (or keys) are likely to appear in the song. The topic weight vector is modeled as a Dirichlet random variable. Next, for each segment of the song, we sample from the topic weight vector to determine the key (e.g., A minor) of that segment. Finally, we repeatedly draw notes from the key-profile until we have generated all the notes in the segment. More formally, we can describe this generative process as follows:

1. For each song in the corpus, choose a K-dimensional topic weight vector θ from the Dirichlet distribution:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i \theta^{\alpha_i - 1}. \tag{1}$$

Note that α is a K-dimensional corpus-level parameter that determines which topics are likely to co-occur in individual songs. The topic weight vector satisfies $\theta_i \geq 0$ and $\sum_k \theta_k = 1$.

2. For each segment indexed by $n \in \{1, \ldots, N\}$ in a song, choose the topic $z_n \in \{1, 2, \ldots, K\}$ from the multinomial distribution $p(z_n = k|\theta) = \theta_k$.
3. For each note indexed by $\ell \in \{1, \ldots, L\}$ in the nth measure, choose a pitch-class from the multinomial distribution $p(u_{n\ell} = i|z_n = j, \beta) = \beta_{ij}$. The β parameter is a $V \times K$ matrix that encodes each topic as a distribution over $V = 12$ neutral pitch-classes. Section 2.3 describes how we identify these distributions as key-profiles for particular musical keys.

This generative process specifies the joint distribution over observed and latent variables for each song in the corpus. In particular, the joint distribution is given by:

$$p(\theta, \mathbf{z}, s|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) \prod_{l=1}^{L_n} p(u_{nl}|z_n, \beta). \quad (2)$$

Figure 2(a) depicts the graphical model for the joint distribution over all songs in the corpus. As in LDA [1], we use plate notation to represent independently, identically distributed random variables within the model. Whereas LDA for text describes each document as a "bag of words", we model each song as a "bag of segments", and each segment as a "bag of notes". As a result, the graphical model in Figure 2(a) contains an additional plate beyond the graphical model of LDA for text.

2.2 Inference and learning

The model in eq. (2) is fully specified by the Dirichlet parameter α and the musical key-profiles β. Suppose that these parameters are known. Then we can use probabilistic inference to analyze songs in terms of their observed notes. In particular, we can infer the main key-profile for each song as a whole, or for individual segments. Inferences are made by computing the posterior distribution

$$p(\theta, \mathbf{z}|s, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, s|\alpha, \beta)}{p(s|\alpha, \beta)} \quad (3)$$

following Bayes rule. The denominator in eq. (3) is the marginal distribution, or likelihood, of a song:

$$p(s|\alpha, \beta) = \int p(\theta|\alpha) \prod_{n=1}^{N} \sum_{z_n=1}^{K} p(z_n|\theta) \prod_{l=1}^{L_n} p(u_{nl}|z_n, \beta)\, d\theta. \quad (4)$$

The problem of learning in our model is to choose the parameters α and β that maximize the log-likelihood of all songs in the corpus, $\mathcal{L}(\alpha, \beta) = \sum_m \log p(s_m|\alpha, \beta)$. Learning is unsupervised because we require no training set with key annotations or labels.

In latent variable models such as ours, the simplest approach to learning is maximum likelihood estimation using the Expectation-Maximization (EM) algorithm [3]. The EM algorithm iteratively updates parameters by computing expected values of the latent variables under the posterior distribution in eq. (3). In our case, the algorithm iteratively alternates between an E-step, which represents each song in the corpus as a random mixture of 24 key-profiles, and an M-step, which re-estimates the weights of the pitch classes for each key-profile. Unfortunately, these expected values cannot be analytically computed; therefore, we must resort to a strategy for approximate probabilistic inference. We have developed a variational approximation for our model based on [7] that substitutes a tractable distribution for the intractable one in eq. (3). Appendix A describes the problems of inference and learning in this approximation in more detail.

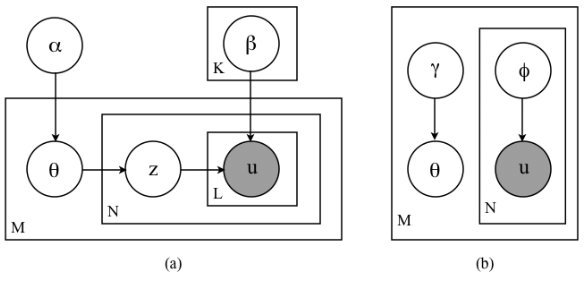


Figure 2. (a) Graphical representation of our model and (b) the variational approximation for the posterior distribution in eq. (3). See Appendix A for details.

2.3 Identifying Topics as Keys

Recall from section 2.1 that the estimated parameter β expresses each topic as a distribution over $V = 12$ neutral pitch-classes. While this distribution can itself be viewed as a key-profile, an additional assumption is required to learn topics that can be identified with particular musical keys. Specifically, we assume that key-profiles for different keys are related by simple transposition: e.g., the profile for C♯ is obtained by transposing the profile for C up by one half-step. This assumption is the full extent to which our approach incorporates prior knowledge of music.

The above assumption adds a simple constraint to our learning procedure: instead of learning $V \times K$ independent elements in the β matrix, we tie diagonal elements across different keys of the same mode (major or minor). Enforcing this constraint, we find that the topic distributions learned by the EM algorithm (see section 3) can be unambiguously identified with the $K = 24$ major and minor modes of classical western music. For example, one topic distribution places its highest seven weights on the pitches C, D, E, F, G, A, and B; since these are precisely the notes of the C major scale, we can unambiguously identify this topic distribution with the key-profile for C major.

3. RESULTS

We estimated our model from a collection of 235 MIDI files compiled from http://www.classicalmusicmidipage.com.

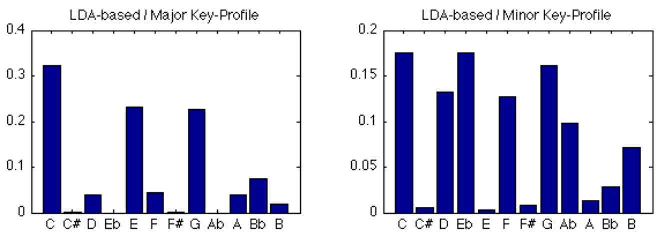


Figure 3. The C major and C minor key-profiles learned by our model, as encoded by the β matrix.

The collection included works by Bach, Vivaldi, Mozart, Beethoven, Chopin, and Rachmaninoff. These composers were chosen to span the baroque through romantic periods of western, classical music.

We experimented with different segment lengths and different ways of compiling note counts. Though measures define natural segments for music, we also experimented with half-measures and quarter-beats. All these choices led to similar musical key-profiles. We also experimented with two ways of compiling note counts within segments. The first method sets the counts proportional to the cumulative duration of notes across the segment; the second method sets the counts proportional to the number of distinct times each note is struck. We found that the second method worked best for key-finding, and we report results for this method below.

3.1 Learning Key-Profiles

Recall that each column of the estimated β matrix encodes a musical key as a distribution over $V = 12$ neutral pitch-classes. Fig. 3 shows the two columns that we identified as belonging to the keys of C major and C minor. These key-profiles have the same general shape as those of KK, though the weights for each pitch-class are not directly comparable. (In our model, these weights denote actual probabilities.) Note that in both major and minor modes, the largest weight occurs on the tonic (C), while the second and third largest weights occur on the remaining degrees of the triad (G, E for C major; G, E♭ for C minor). Our key-profiles differ only in the relatively larger weight given to the minor 7th (B♭) of C major and major 7th (B) of C minor. Otherwise, the remaining degrees of the diatonic scale (D, F, A for C major; D, F, A♭ for C minor) are given larger weights than the remaining chromatics. Profiles for other keys can be found by transposing.

3.2 Symbolic Key-Finding

From the posterior distribution in eq. (3), we can infer hidden variables θ and $\mathbf{z}$ that identify dominant keys in whole songs or segments within a song. In particular, we can identify the overall key of a song from the largest weight of the topic vector θ that maximizes eq. (3). Likewise, we can identify the key of particular segments from the most probable values of the topic latent variables z_n.

We first show results at the song-level, using our model to determine the overall key of the 235 musical pieces in

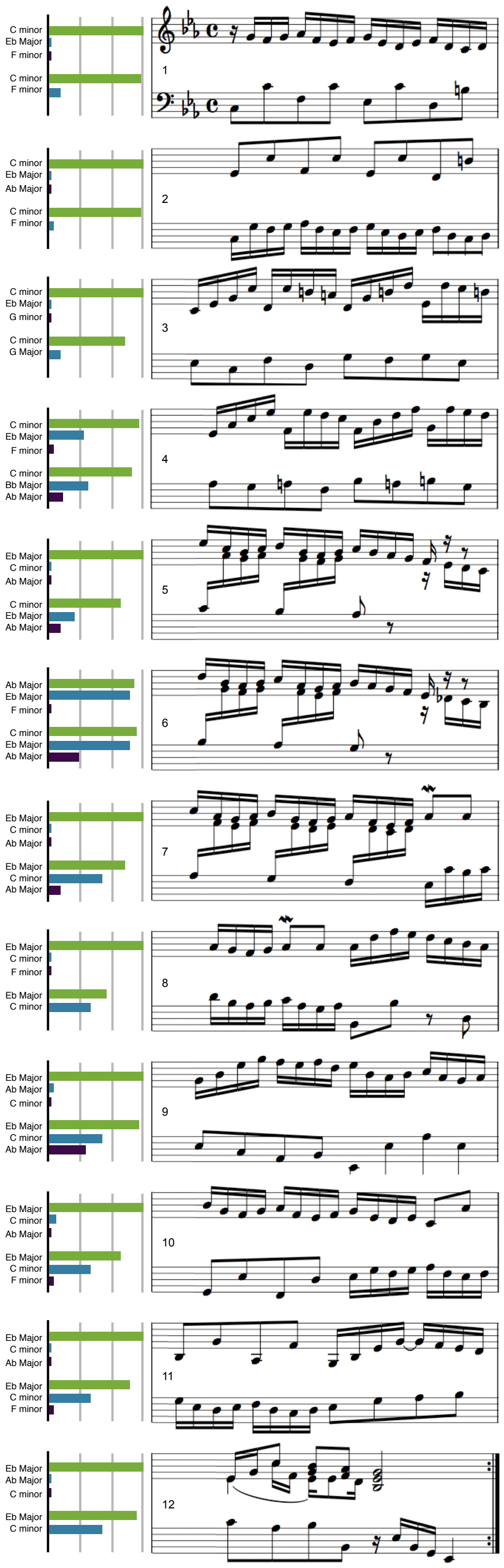


Figure 4. Key judgments for the first 12 measures of Bach's Prelude in C minor, WTC-II. Annotations for each measure show the top three keys (and relative strengths) chosen for each measure. The top set of three annotations are judgments from our LDA-based model; the bottom set of three are from human expert judgments [8].

Song Length	All	20 beats	8 beats	4 beats
LDA	86%	77%	74%	67%
KS	80%	71%	67%	66%

Table 1. Key-finding accuracy of our LDA model and the KS model on 235 classical music pieces. Song length indicates how much of each piece was included for analysis.

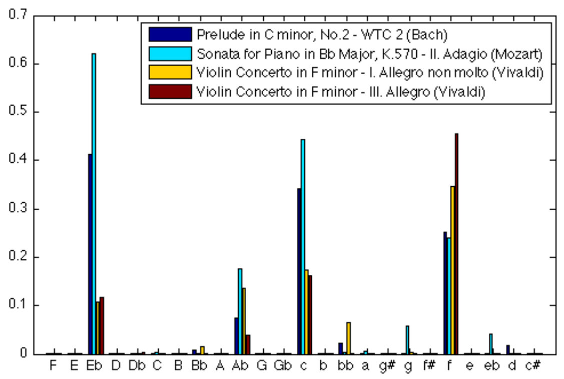


Figure 5. Songs represented as distributions over key-profiles. The first set of bars shows keys used in the query song; the remaining sets of bars show keys used in the three songs of the corpus judged to be most similar. Note how all songs modulate between the keys of E♭ M, A♭ M, C m, and F m.

our corpus. We tested our model against a publicly available implementation of the KS model [4] that uses normalized KK key-profiles and weighted note durations. Table 1 compares the results when various lengths of each piece are included for analysis. In this experiment, we found that our model performed better across all song lengths.

We also compared our model to three other publicly available key-finding algorithms [13]. We were only able to run these algorithms on a subset of 107 pieces in our corpus, so for these comparisons we only report results on this subset. These other algorithms used key-profiles from another implementation of the KS model [8] and from empirical analyses of key-annotated music [14, 15]. Analyzing whole songs, these other algorithms achieved accuracies between 62%–67%. Interestingly, though these models obtained their key-profiles using rule-based or supervised methods, our unsupervised model yielded significantly better results, identifying the correct key for 79% of the songs in this subset of the corpus.

Next, we show results from our model at the segment level. Fig. 4 shows how our model analyzes the first twelve measures of Bach's Prelude in C minor from Book II of the *Well-Tempered Clavier* (WTC-II). Results are compared to annotations by a music theory expert [8]. We see that the top choice of key from our model differs from the expert judgment in only two measures (5 and 6).

3.3 Measuring Harmonic Similarity

To track key modulations within a piece, we examine its $K = 24$ topic weights. These weights indicate the proportion of time that the song spends in each key. They also provide a low-dimensional description of each song's harmonic structure. We used a symmetrized Kullback-Leibler (KL) divergence to compute a measure of dissimilarity between songs based on their topic weights. Fig. 5 shows several songs as distributions over key-profiles. (Note that previous graphs showed key-profiles as distributions over pitches.) The first set of bars show the topic weights for the same Bach Prelude analyzed in the previous section; the remaining sets of bars show the topic weights for the three songs in the corpus judged to be most similar (as measured by the symmetrized KL divergence). From the topic weight vectors, we see that all songs modulate primarily between the keys of E♭ M, A♭ M, C m, and F m.

4. CONCLUSION

In this paper, we have described a probabilistic model for the unsupervised learning of musical key-profiles. Unlike previous work, our approach does not require key-annotated music or make use of expert domain knowledge. Extending LDA from text to music, our model discovers latent topics that can be readily identified as the $K = 24$ primary key-profiles of western classical music. Our model can also be used to analyze songs in interesting ways: to determine the overall key, to track harmonic modulations, and to provide a low-dimensional descriptor for similarity-based ranking. Finally, though the learning in our model is unsupervised, experimental results show that it works very well compared to existing methods.

5. REFERENCES

[1] D. M. Blei, A. Y. Ng, M.I. Jordan: "Latent Dirichlet allocation," *Journal of Machine Learning Research*, 3:993-1022, 2003.

[2] E. Chew: "The Spiral Array: An Algorithm for Determining Key Boundaries," *Proc. of the Second Int. Conf. on Music and Artificial Intelligence*, 18-31, 2002.

[3] A. Dempster, N. Laird, D. Rubin: "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, 39(1):1-38, 1977.

[4] T. Eerola, P. Toiviainen: "MIDI Toolbox: MATLAB Tools for Music Research," *University of Jyväskylä, Jyväskylä, Finland*, Available: http://www.jyu.fi/musica/miditoolbox/, 2004.

[5] L. Fei-Fei, P. Perona: "A Bayesian hierarchical model for learning natural scene categories," *CVPR*, 524-531, 2005.

[6] T. Griffiths, M. Steyvers, D. Blei, J. Tenenbaum: "Integrating topics and syntax," In L. Saul, Y. Weiss, and L. Bottou, editors, *NIPS*, 537-544, 2005.

[7] M. I. Jordan, Z. Ghahramani, T. Jaakkola, L. Saul: "Introduction to variational methods for graphical models," *Machine Learning*, 37:183-233, 1999.

[8] C. Krumhansl: *Cognitive Foundations of Musical Pitch*, Oxford University Press, Oxford, 1990.

[9] C. Krumhansl, E. J. Kessler: "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys," *Psychological Review*, 89:334-68, 1982.

[10] H. C. Longuet-Higgins, M. J. Steedman: "On interpreting Bach," *Machine Intelligence*, 6:221-41, 1971.

[11] B. Marlin: "Modeling user rating profiles for collaborative filtering," In S. Thrun and L. Saul and B. Schölkopf, editors, *NIPS*, 2003.

[12] D. Rizo: "Tree model of symbolic music for tonality guessing," *Proc. of the Int. Conf. on Artificial Intelligence and Applications*, 299-304, 2006.

[13] D. Sleator, D. Temperley: "The Melisma Music Analyzer," Available: http://www.link.cs.cmu.edu/music-analysis/, 2001.

[14] D. Temperley: *The Cognition of Basic Musical Structure*, MIT Press, 2001.

[15] D. Temperley: "A Bayesian approach to key-finding," *Lecture Notes in Computer Science*, 2445:195-206, 2002.

A. VARIATIONAL APPROXIMATION

This appendix describes our variational approximation for inference and learning mentioned in section 2. It is similar to the approximation originally developed for LDA [1].

A.1 Variational Inference

The variational approximation for our model substitutes a tractable distribution for the intractable posterior distribution that appears in eq. (3). At a high level, the approximation consists of two steps. First, we constrain the tractable distribution to belong to a parameterized family of distributions whose statistics are easy to compute. Next, we attempt to select the particular member of this family that best approximates the true posterior distribution.

Figure 2(b) illustrates the graphical model for the approximating family of tractable distributions. The tractable model $q(\theta, \mathbf{z}|\gamma, \phi)$ drops edges that make the original model intractable. It has the simple, factorial form:

$$q(\theta, \mathbf{z}|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n) \tag{5}$$

We assume that the distribution $q(\theta|\gamma)$ is Dirichlet with variational parameter γ, while the distributions $q(z_n|\phi_n)$ are multinomial with variational parameters ϕ_n. For each song, we seek a factorial distribution of the form in eq. (5) to approximate the true posterior distribution in eq. (3). Or more specifically, for each song s_m, we seek the variational parameters γ_m and ϕ_m such that $q(\theta, \mathbf{z}|\gamma_m, \phi_m)$ best matches $p(\theta, \mathbf{z}|s_m, \alpha, \beta)$.

Though it is intractable to compute the statistics of the true posterior distribution $p(\theta, \mathbf{z}|\alpha, \beta)$ in eq. (3), it is possible to compute the Kullback-Leibler (KL) divergence

$$\mathrm{KL}(q, p) = \sum_{\mathbf{z}} \int d\theta q(\theta, \mathbf{z}|\gamma, \phi) \log \frac{q(\theta, \mathbf{z}|\gamma, \phi)}{p(\theta, \mathbf{z}|s, \alpha, \beta)} \tag{6}$$

up to a constant term that does not depend on γ and ϕ. Note that the KL divergence measures the quality of the variational approximation. Thus, the best approximation is obtained by minimizing the KL divergence in eq. (6) with respect to the variational parameters γ and ϕ_n. To derive update rules for these parameters, we simply differentiate the KL divergence and set its partial derivatives equal to zero. The update rule for γ_m is analogous to the one in the LDA model for text documents [1]. The update rule for the multinomial parameters ϕ_{ni} is given by:

$$\phi_{ni} \propto \prod_{j=1}^{V} \beta_{ij}^{x_n^j} \exp[\Psi(\gamma_i)], \tag{7}$$

where $\Psi(\cdot)$ denotes the digamma function and x_n^j denotes the count of the jth pitch class in the nth segment of the song. We omit the details of this derivation, but refer the reader to the original work on LDA [1] for more detail.

A.2 Variational Learning

The variational approximation in eq. (5) can also be used to derive a lower bound on the log-likelihood $\log p(s|\alpha, \beta)$ of a song s. Summing these lower bounds over all songs in the corpus, we obtain a lower bound $\ell(\alpha, \beta, \gamma, \phi)$ on the total log-likelihood $\mathcal{L}(\alpha, \beta) = \sum_m \log p(s_m|\alpha, \beta)$. Note that the bound $\ell(\alpha, \beta, \gamma, \phi) \leq \mathcal{L}(\alpha, \beta)$ depends on the model parameters α and β as well as the variational parameters γ and ϕ across all songs in the corpus.

The variational EM algorithm for our model estimates the parameters α and β to maximize this lower bound. It alternates between two steps:

1. (E-step) Fix the current model parameters α and β, compute variational parameters $\{\gamma_m, \phi_m\}$ for each song s_m by minimizing the KL divergence in eq. (6).
2. (M-step) Fix the current variational parameters γ and ϕ across all songs from the E-step, maximize the lower bound $\ell(\alpha, \beta, \gamma, \phi)$ with respect to α and β.

These two steps are repeated until the lower bound on the log likelihood converges to a desired accuracy. The updates for α and β in the M-step are straightforward to derive. The update rule for β is given by:

$$\beta_{ij} \propto \sum_{m=1}^{M} \sum_{n=1}^{N} \phi_{mn}^{i} x_{mn}^{j}. \tag{8}$$

While the count x_{mn}^j in eq. (8) may be greater than one, this update is otherwise identical to its counterpart in the LDA model for text documents. The update rule for α also has the same form.

PUBLISHING MUSIC SIMILARITY FEATURES ON THE SEMANTIC WEB

Dan Tidhar, György Fazekas, Sefki Kolozali, Mark Sandler
Centre for Digital Music
Queen Mary, University of London
Mile End Road, London E1, UK
{dan.tidhar, gyorgy.fazekas, sefki.kolozali, mark.sandler}@elec.qmul.ac.uk

ABSTRACT

We describe the process of collecting, organising and publishing a large set of music similarity features produced by the SoundBite [10] playlist generator tool. These data can be a valuable asset in the development and evaluation of new Music Information Retrieval algorithms. They can also be used in Web-based music search and retrieval applications. For this reason, we make a database of features available on the *Semantic Web* via a SPARQL end-point, which can be used in *Linked Data* services. We provide examples of using the data in a research tool, as well as in a simple web application which responds to audio queries and finds a set of similar tracks in our database.

1. INTRODUCTION

Similarity-based retrieval is an important subject area in music information research. Yet, researchers working in this field are often limited by the unavailability of large audio collections, copyright restrictions, and even more often, unreliable metadata associated with songs in a particular music database or personal library. This paper describes a system for collecting and publishing music similarity features from a large user base coupled with valuable editorial metadata. Metadata are verified against MusicBrainz, [1] a large public database of editorial information on the Web, and published together with the matching similarity features on the *Semantic Web* [1]. We explore some research opportunities opened by the system, and describe SAWA [2] recommender, a sample web application which demonstrates how the published data can be used. Rather than describing a music recommender in detail, our primary motivation is in making high quality data available for similarity and recommendation research in a standardised way.

[1] http://www.MusicBrainz.org/

[2] SAWA stands for Sonic Annotator Web Application. A search and recommendation system built on SAWA and the SoundBite data set is available at: http://www.isophonics.net/sawa/rec

The heart of the data collection system is SoundBite [10] [15], a tool for similarity-based automatic playlist generation. Soundbite is available as an iTunes plugin, and is currently being implemented as a plugin for other audio players as well. Once installed, it extracts features from the user's entire audio collection and stores them for future similarity calculations. It can then generate playlists consisting of n most similar tracks to any given seed track specified by the user. The similarity data currently consists of 40 values per track, based on the distribution of Mel-Fequency Cepstral Coefficients (MFCC) as described in [10]. The extracted features are also reported to a central server, where they become part of the so called *Isophone* database. This database is used for aggregating information from SoundBite clients, consisting of editorial metadata and similarity features for each audio track. The entire system may therefore be regarded as a distributed framework for similarity feature extraction. The accumulated data can be valuable to the research community, and may also be used by other audio similarity and recommendation systems. In order to facilitate such usage, we publish a cleaned-up portion of the data on the Semantic Web.

The rest of the paper is organised as follows: In section two, we provide brief explanations of some of the key terms relevant to the technologies we use. In section three, we describe the published data set, the collection system architecture, the data clean-up process, and the way researchers as well as Semantic Web applications can access the data using a *SPARQL end-point* [3]. Finally, in section four, we describe our prototype recommender, a publicly accessible web application based on this data set.

2. LINKED DATA AND THE SEMANTIC WEB

Building the Semantic Web involves creating a machine-interpretable web of data in parallel to the existing web of documents [1]. By uniformly integrating diverse data and services, it aims to enable applications which would be difficult, if not impossible, to build using prevailing incompatible interfaces and representation formats. An example application from the world of music would interlink content providers (music labels, music sellers, online radio stations), meta-databases holding musical and artists infor-

[3] A web resource that responds to queries using the SPARQL Protocol and RDF Query Language, an SQL-like language for accessing RDF [9] data bases.

mation, semantic audio tools and music identification services, and perhaps even music collections held on personal computers. This could revolutionise the way we access or discover new music. However, creating such a distributed network requires that all data sources speak the same language, i.e., are governed by a common schema.

Because of the diverse and unbounded nature of information on the general Web (and we believe that musical information is just as diverse), a major challenge was set forth to Semantic Web developers: How to design a standard, extensible schema for representing information encompassing a wide range of human knowledge? The Semantic Web's answer to this apparently complex and circular problem is in specifying how information is published, rather than trying to arrange everything into rigid data structures.

2.1 Semantic Web Technologies

The key concepts and technologies enabling the development of the Semantic Web are the Resource Description Framework (RDF) [9], Semantic Web ontologies, and RDF query languages.

RDF is a conceptual data model. It provides the flexibility and modularity required for publishing diverse semi-structured data — that is, just about anything on the Semantic Web. It is based on the simple idea of expressing statements in the form of *subject — predicate — object*. Elements of these statements are *literals*, and *resources* named by Uniform Resource Identifiers (URI). This provides the model with an unambiguous way of referring to things, and – through the HTTP dereferencing mechanism – access to additional information a resource may hold. Simple RDF statements, however, are not sufficient for expressing things unambiguously. In order to be precise in our statements, we need to be able to define, and later refer to concepts and relationships pertinent to a domain or application. Ontologies are the tools for establishing these necessary elements.

Semantic Web ontologies are built on the same conceptual model that is used for expressing data. However, additional vocabularies were created for expressing formal ontologies. RDF is the basis for a hierarchy of languages recommended by the W3C [4] . This includes the *RDF Schema Language* (RDFS) for defining classes and properties of RDF resources and the *OWL Web Ontology Language* for making RDF semantics more explicit. [5]

Besides a standard way of representing information, access to data also needs to be standardised. The *SPARQL Protocol and RDF Query Language [6]* is a recent recommendation by the W3C for accessing RDF data stores. A Web interface which accepts and executes these queries is commonly referred to as a *SPARQL end-point*.

SPARQL allows access to information in a multitude of ways. In the simplest case, it is used in a similar manner to querying a relational database using SQL [6] . A query – consisting of a set of triple patterns – is matched against the database. Results are then composed of variable bindings of matching statements, based on a *select* clause specified by the user. This can be used to retrieve information about a particular resource. More complex SPARQL queries are frequently used to aggregate information in a particular way. For example, a user agent may interpret a query and aggregate data from various sources on the fly. The standardisation and increasing support of the SPARQL query language strongly promotes the adoption of RDF as a prevailing metadata model and language.

2.2 Linked vs. Structured Data

There are already a large number of services exposing structured data on the Web. Examples include Google, Yahoo, OpenSearch, Amazon, Geonames, and the MediaWiki APIs. Music-related data providers include the Magnatune and Jamendo labels, and the MusicBrainz database. Most of these services use proprietary XML-based data formats. This is sufficient for structuring data for a given application, yet, because of the fairly ad-hoc definition of concepts in XML schema, these formats do not provide the means for transparent access to a variety of services. The *Linked Data* community [7] offers standardised access to some information exposed by the previously listed services, as well as other related data sets. In Linked Data services, the reliance on diverse interfaces and result formats is reduced by using RDF as a common representation. This also provides the means for making data available on the Semantic Web.

Most existing metadata formats for expressing audio features are also based on XML. MPEG-7 [7] and ACE-XML [11] are perhaps the most prominent examples. The structural and syntactical requirements for expressing elements and schemes in MPEG-7 are fulfilled by using an extended XML schema language. Although this allows the production of machine-parsable data, it does not provide a machine-interpretable representation of the semantics associated with MPEG-7 metadata elements. The same problem arises with the ACE-XML format developed for the jMIR package, linking components such as jAudio for feature extraction, and the ACE classification engine. A common problem can be recognised in using XML for standardised syntax, while the data model remains disjoint and often arbitrary, with ad-hoc definition of terms, and without the ability to define meta-level relationships such as the equivalence of certain concepts. This hinders the ability to integrate services expressing metadata in these formats, or the reuse of any of the defined terms in other domains. Our data, on the other hand, is expressed using a flexible RDF and Web Ontology based data model. It is compatible with the Music Ontology [12], which is already widely used in Linked Data applications.

[4] The World Wide Web Consortium: `http://www.w3.org/`

[5] For example, OWL-DL (description logic) can impose restrictions on the range and domain types of properties, or constraints on cardinality.

[6] Structured Query Language

[7] "Linking open data on the semantic web", `http://linkeddata.org/`

2.3 Ontologies

As mentioned in section 2.1, only a conceptual model is provided by RDF. Ontologies are used for the actual definition of pertinent terms and relationships. Recent efforts [13] toward integrating music-related web services and data sources have led to the creation of the Music Ontology [12]. It serves as a standard base ontology which can be readily used for describing a wide range of concepts. These include high-level editorial data about songs or artists, production data about musical recordings, and detailed structural information about music using events and timelines. The ontology provides the basis for numerous extensions, including the Audio Features Ontology [14]. The music similarity features published and used by the services described in this paper are expressed using these ontologies.

3. THE SOUNDBITE DATASET

The SoundBite dataset consists of MFCC features and MusicBrainz identifiers for a cleaned-up subset of the data reported back to the central server by the different instances of the SoundBite client application. Currently, the database includes metadata for 152,410 tracks produced by 6,938 unique artists. These numbers are expected to grow as the number of SoundBite users grows, and the data clean-up procedure is refined. We believe that this dataset can be especially valuable because of its scope and diversity. Furthermore, it originates from real-world users, and therefore reflects at least a part of the users' community interests and relevant needs. It is not susceptible to any biases which might be implicit in datasets which are artificially-created for research purposes. We currently do not collect personal data about SoundBite users, although this information might be of interest for other studies. However, at the time of writing this paper, the growing user community already seems sufficiently large and varied for the dataset to cover the most popular genres. The dataset coverage is expected to further improve as a direct result of user base growth and further clean-up.

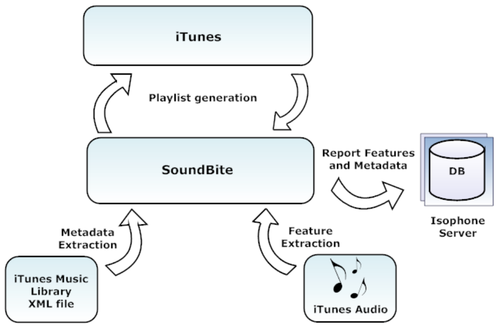


Figure 1. Simplified SoundBite Architecture.

As mentioned in section 1, the features extracted by each instance of the SoundBite client application are reported back to a central server, where they are stored in a database alongside the relevant textual metadata. Figure 1 illustrates the interaction between the iTunes application, the SoundBite plugin, and the Isophone server. The relevant resources on the client side are iTunes music library and the corresponding XML file which describes the collection. Since textual metadata contained in this XML file, such as title and artist, are often inserted or altered by the users themselves, we cannot rely on their accuracy. They certainly cannot be used as unique identifiers which are necessary for facilitating public usage of the dataset. Prior to publishing, the data need to undergo a clean-up process, as described in following sections. Using the MFCC data for automatic playlist creation, as done by the Soundbite plugin, requires similarity metrics to be defined on the data. These are not provided as part of the dataset, but are rather considered part of an algorithm which utilizes the data for a particular application, namely, playlist creation. The published data facilitate the exploration of further similarity algorithms and applications.

3.1 Data filtering and publishing

Since the audio tracks to which the features relate reside in end-users' audio collections, they are inaccessible to us and we obviously cannot provide them as part of the dataset. It is therefore of crucial importance that we do provide unique identifiers to the audio material, without which the provided features can be of very little use. As a source for such unique identifiers, and as an aid in metadata-based filtering, we use the MusicBrainz database.

MusicBrainz is a comprehensive public community music meta-database. It can be used to identify songs or CDs, and provides valuable data about tracks, albums, artists and other related information. MusicBrainz can be accessed either through their web site or by using client applications via an application programming interface (API). We use the MusicBrainz service as metadata reference in the filtering process, and use MusicBrianz ID's as unique identifiers which are published together with the MFCC's.

The editorial metadata reported back to the server by SoundBite (as depicted in figure 1) include the entire content of the iTunes Music Library XML file. The data clean-up procedure currently uses the following metadata items:

- Track Title
- Main Artist
- Album Title
- Track Duration
- File Format
- Bit Rate

In the first stage of the clean-up process, title, artist, and album are matched against the MusicBrainz database. The track's duration is used for resolving ambiguities, as well as for sanity check (a large difference between the reported duration value and the duration retrieved from MusicBrainz may indicate that the other fields are erroneously or maliciously wrong). Each matching track is assigned an

ID provided by the MusicBrainz database, which serves as unique identifier. We found that about 28% of the entries in our database had exact matches (artist, title, album, and approximate duration) in the MusicBrainz database. The remaining 72% are stored for possible future use, but do not currently qualify for publishing. The relatively small proportion of tracks that do qualify can be regarded as an indication of the poor reliability of textual metadata in end users' audio collection.

As indicated in [16], MFCC features are more robust at higher bit rates. Therefore, in the second stage the data is further filtered according to maximum bit rate and best quality audio file type (e.g. keeping AACs as opposed to MP3s), in order to preserve the highest quality features for each track. Since these parameters are included in the metadata reported to the server, this doesn't require access to the audio files themselves.

Once cleaned-up and filtered as described above, the MFCC features and the obtained MusicBrainz ID's are exported from the database as RDF's using the D2R Mapping [2], with the appropriate linking to the Audio Features [14] and SoundBite ontologies (see figure 2). They are then made available via a SPARQL end-point on our server[8].

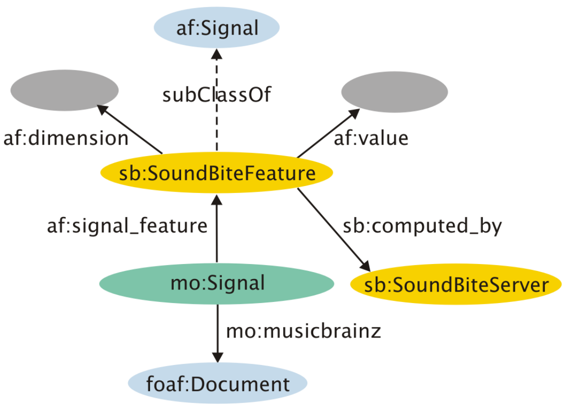


Figure 2. Accessing the SPARQL endpoint using the SoundBite ontology.

4. APPLICATIONS

In this section we describe how our data set can be used as basis for the development of new music similarity and music recommendation algorithms. Additionally, we provide an example of a prototype audio search engine. The service uses our database to find tracks similar to an audio query and returns editorial metadata about the found set obtained from external web-services.

4.1 Research Platform

There has recently been a significant amount of research on music similarity and audio-based genre classification. Both fields use content-based descriptors extracted from audio signals. Apart form being computationally expensive, audio-similarity features coupled with matching textual metadata are not easily obtainable in large quantities. The published Isophone data provide an excellent opportunity for further research based on a reliable music collection with readily-available MFCC features. Obviously, since the available features are calculated prior to being published, the dataset does not accommodate changes to the algorithms which produced them in the first place. There is, however, plenty of room for experimentation with the way the different features are combined to form similarity metrics, and the way they are used on the application level. We use the dataset in a research platform, which facilitates such experiments. We are currently exploring different similarity metrics based on the published features, as well as different ways to combine the features with other relevant data, e.g. in the context of hybrid recommender systems (see, for exmple, [5]). As a proof of concept, and to demonstrate how the research community could use the published data, we have implemented a tool which queries the SPARQL endpoint to obtain MFCC's for given tracks, to facilitate the above mentioned research activities.

4.2 SAWA-recommender

SAWA-recommender[9] is a simple *query by example* search service made available on the Web. Its main goal is to demonstrate an application of the published music similarity features. In this section, we outline the use and construction of this service.

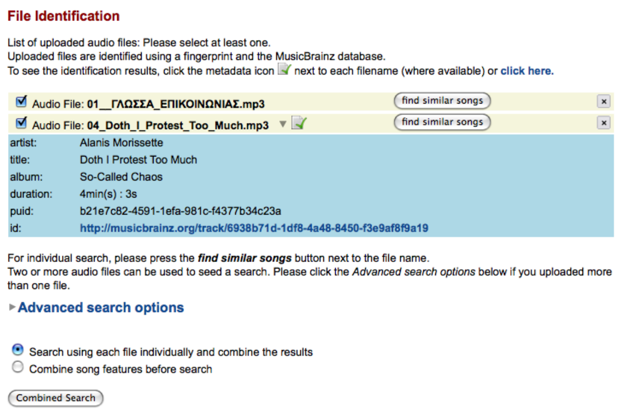
A query to SAWA-recommender is formed by one or more audio files uploaded by the user. It is typically based on single file, however, uploading multiple audio files is also allowed. In the latter case, a small set of songs forms the basis of the query, either by considering similarity to any of the uploaded songs (and ranking the results appropriately), or formulating a single common query by jointly calculating the features of the query songs. The calculated query is matched against the Isophone database holding similarity features and MusicBrainz identifiers associated with each song in this database. Finally, the MusicBrainz web API is used to obtain metadata about songs in the result set. These are displayed to the user. The metadata consist of basic information such as song title, album title and the main artist's name associated with each song. We also provide direct links to MusicBrainz, as well as Linked Data services such as BBC Music[10] artist pages.

For each uploaded file, the system also attempts to identify the audio by calculating a MusicDNS[11] fingerprint and associated identifier. This identifier is matched against the MusicBrainz database to obtain editorial data, hence one can also use the service to find more information about an audio file (see figure 3).

The architecture of the web application is depicted in figure 4. The system is built on software components de-

[8] `http://dbtune.org/iso/`

[9] `http://isophonics.net/sawa/rec`
[10] `http://www.bbc.co.uk/music/`
[11] `http://www.musicdns.com/`


File Identification

List of uploaded audio files: Please select at least one.
Uploaded files are identified using a fingerprint and the MusicBrainz database.
To see the identification results, click the metadata icon next to each filename (where available) or **click here.**

[✓] Audio File: **01__ΓΛΩΣΣΑ_ΕΠΙΚΟΙΝΩΝΙΑΣ.mp3** — find similar songs
[✓] Audio File: **04_Doth_I_Protest_Too_Much.mp3** — find similar songs

artist:	Alanis Morissette
title:	Doth I Protest Too Much
album:	So-Called Chaos
duration:	4min(s) : 3s
puid:	b21e7c82-4591-1efa-981c-f4377b34c23a
id:	http://musicbrainz.org/track/6938b71d-1df8-4a48-8450-f3e9af8f9a19

For individual search, please press the ***find similar songs*** button next to the file name.
Two or more audio files can be used to seed a search. Please click the *Advanced search options* below if you uploaded more than one file.

▸ **Advanced search options**

[✓ Search using each file individually and combine the results]
Combine song features before search

Combined Search

Figure 3. File Identification and Selection Interface.

veloped in the OMRAS2 project [12] and a small set of common open-source libraries.

The signal processing back-end of the service is provided by Sonic Annotator [13] together with Vamp audio analysis plugins [3]. These plugins use an application programming interface (API) designed for audio feature extraction. They take audio input and return structured numerical results. While Vamp plugins perform the feature extraction step (implemented in efficient C++ code), Sonic Annotator is the host application that reads audio data and applies plugins to one or more files in batch. This program accepts configuration data and returns audio features in RDF according to specific ontologies [14] [4]. For the purpose of this present search system, we configure Sonic Annotator and a suitable Vamp plugin to extract audio similarity features based on MFCCs [10].

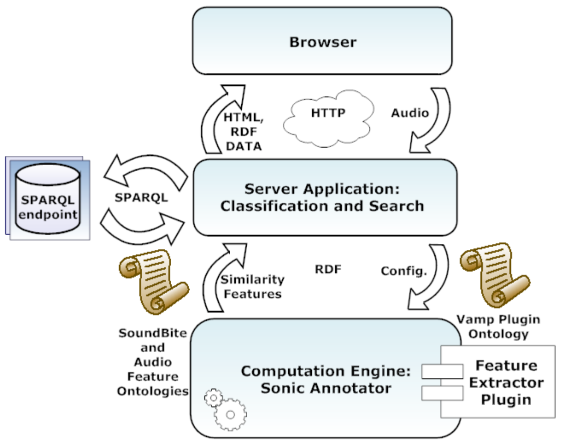


Figure 4. Search Engine System Architecture.

The core of the search system is a Python application which provides a Web interface and a basic search and classification engine. It also manages user sessions and uploaded files. Since users may upload copyrighted material, user sessions are fully isolated, and all audio files are automatically deleted as the user leaves the service.

[12] `http://www.omras2.org/`
[13] Available at: `http://omras2.org/SonicAnnotator`

The Web interface is built using the Cherrypy [14] Python library. This allows the implementation of HTTP request handlers as ordinary methods defined within a web application class. Using this library, it is straightforward to accept audio files as well as publishing data received from other system components using dynamically generated web pages.

Query processing and database search is performed in three steps. First, we extract features from the uploaded audio files. For optimised search, the query features are matched against a model trained on the whole database. Finally, a selected group of songs are ranked based on their similarity to the query and the results are displayed to the user.

Although simple linear search was suggested for personal collections, [10] the size of our current database is over 150.000 tracks and it is expected to grow. For this reason, we partition the data space by similarity to form self-similar groups of songs. These groups or clusters can then be used to index the database. We can limit the search space by choosing the best matching cluster based on its proximity to the query song. Hence, the number of direct similarity calculations is greatly reduced. Since our goal is search optimisation rather than classification, we choose an unsupervised learning algorithm using a self-organising model, similar to a Self Organising Map [8]. The details of this exceed the scope of our current discussion. However, it is important to note that using the symmetrised Kullback-Leibler (KL) divergence as basis for training and classification, we could verify the scarcity of hubs reported in [10] using a 100-times larger database of features. The songs are roughly equally distributed among the nodes. Only 4% of the nodes became hubs (containing a large set of songs) and 3% of them contain fewer songs. We also found that this phenomenon is largely independent of the size of the model (the number of nodes). The fact that the collection can be partitioned automatically by grouping similar songs - without obtaining too many over-populated clusters (hubs) - shows that the database is well balanced and justifies the choice of metrics and learning algorithm. This is also favourable for the search application, since we can limit the number of songs where the similarity has to be explicitly calculated and compute the divergence only within a single class without significantly modifying the results set. In our current implementation, a local copy of the partitioned database is used for searching, however, the model is trained on the data available at the SPARQL end-point. This is achieved by an appropriate SPARQL query, generated and issued in each training iteration. This way, the model can easily be adjusted if the database is expanded in the future. For producing the final results, a limited set of similar songs is collected and ranked by similarity to the query song(s) using the KL divergence described in [10]. Finally, the metadata are obtained from MusicBrainz and displayed to the user.

Since our similarity assessment follows the same principles applied in SoundBite, these results can be seen as

[14] Available at: `http://www.cherrypy.org/`

content-based recommendations. However, given the size of the database they might be useful for identifying unknown songs or song segments. In a commercial situation, our service might be useful in finding an alternative for a song, where a copyright agreement for its use can not be obtained.

5. CONCLUSION

We described the SoundBite dataset and its publication on the Semantic Web. We believe that due to its scope and diversity (which are expected to grow even further), it is a valuable resource for researchers as well as application developers. We provided some examples of applying the data in research and prototyping web applications. These initial examples strengthen our beliefs regarding the value and potential of this dataset, and we therefore intend to continue to follow our policy of publishing accumulated data on the Semantic Web. We intend to further develop this particular dataset by collecting more raw data and refining the filtering process, and to continue developing applications which utilize the data for research purposes and public use.

6. ACKNOWLEDGEMENTS

The authors acknowledge the support of the School of Electronic Engineering and Computer Science, Queen Mary, University of London. This work has been carried out at the Centre for Digital Music and supported by the EPSRC-funded ICT project OMRAS- 2 (EP/E017614/1).

7. REFERENCES

[1] T. Berners-Lee, J. Handler, and O. Lassila, "The semantic web", *Scientific American*, pp. 34–43, May 2001.

[2] C. Bizer and R. Cyganiak, "D2R Server-Publishing Relational Databases on the Semantic Web", *5th International Semantic Web Conference,* Athens, GA, USA, 2006.

[3] C. Cannam, "The Vamp Audio Analysis Plugin API: A Programmer's Guide", `http://vamp-plugins.org/guide.pdf` Last accessed: July, 2009.

[4] C. Cannam, "The Vamp Plugin Ontology", `http://omras2.org/VampOntology`, Last accessed: July, 2009.

[5] J. Donaldson,, "A hybrid social-acoustic recommendation system for popular music", *In proceedings of the 2007 ACM conference on Recommender systems*, Minneapolis, MN, USA, 2007.

[6] K. Grant Clark, L. Feigenbaum, E. Torres, (eds.) "SPARQL Protocol for RDF" *W3C Recommendation*, 15. January 2008 `http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/` Last accessed: July, 2009.

[7] H. Kim, N. Moreau, T. Sikora, "MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval." *Wiley and Sons*, October 2005.

[8] T. Kohonen, "Self-Organizing Maps", *Springer*, Berlin, 1995.

[9] O. Lassila, R. Swick, "Resource description framework model and syntax specification", 1998. `http://citeseer.ist.psu.edu/article/lassila98resource.html` Last accessed: March, 2009.

[10] M. Levy and M. Sandler, "Lightweight measures for timbral similarity of musical audio", *In Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, (Santa Barbara, California, USA, October 27, 2006). AMCMM '06. ACM, New York, NY, 27-36.

[11] D. McEnnis, C. McKay, I. Fujinaga, P. Depalle, "jAudio: A Feature Extraction Library", *in Proc. of the International Conference on Music Information Retrieval,*London, UK, 2005.

[12] Y. Raimond., S. Abdallah, and M. Sandler. "The Music Ontology", *Proceeedings of the 8th International Conference on Music Information Retrieval*, Vienna, Austria, 2007.

[13] Y. Raimond. and M. Sandler, "A Web of Musical Information", *Proceeedings of the 9th International Conference on Music Information Retrieval*, Philadelphia, USA September 14-18, 2008.

[14] Y. Raimond, "Audio Features Ontology Specification", `http://motools.sourceforge.net/doc/audio_features.html` Last accessed: March, 2009.

[15] M. Sandler, M. Levy, "Signal-based Music Searching and Browsing", *ICCE 2007. International Conference on Consumer Electronics*, 10-14 Jan. 2007.

[16] S. Sigurdsson, K. Brandt Petersen, T. Lehn-Schiler, "Mel Frequency Cepstral Coefficients: An Evaluation of Robustness of MP3 Encoded Music," *ISMIR 2006 7th International Conference on Music Information Retrieval.*

GENRE CLASSIFICATION USING BASS-RELATED HIGH-LEVEL FEATURES AND PLAYING STYLES

Jakob Abeßer, Hanna Lukashevich, Christian Dittmar, Gerald Schuller
Fraunhofer IDMT, {abr, lkh, dmr, shl}@idmt.fraunhofer.de

ABSTRACT

Considering its mediation role between the poles of rhythm, harmony, and melody, the bass plays a crucial role in most music genres. This paper introduces a novel set of transcription-based high-level features that characterize the bass and its interaction with other participating instruments. Furthermore, a new method to model and automatically retrieve different genre-specific bass playing styles is presented. A genre classification task is used as benchmark to compare common machine learning algorithms based on the presented high-level features with a classification algorithm solely based on detected bass playing styles.

1. INTRODUCTION

After prolonged series of publications focusing on low- and mid-level features, many works within the MIR community nowadays emphasize the importance of musical high-level features. Their application is expected to significantly increase the precision in automatic music classification and similarity search tasks that have limits using conventional modeling paradigms [2]. Various automatic transcription techniques allow the extraction of score parameters like note pitch, velocity (volume), onset time and duration from polyphonic mixtures. These parameters embody the prior foundation for a subsequent feature extraction. Due to their close relation to musicological expressions, high-level features can be easily understood by musicologists. Thus, they offer a promising opportunity to translate existing musicological knowledge into automatically retrievable properties of analyzed music.
The remainder of this paper is organized as follows. In Sec. 2, we illustrate the goals of this publication and give an overview over related work in the subsequent section. We present both novel transcription-based high-level features and a new framework to model concepts and classes for the purpose of music classification in Sec. 4. Evaluation results from different scenarios are presented and discussed in Sec. 5 and a final conclusion is given in the last section.

2. GOALS & CHALLENGES

Our goal is to design transcription-based high-level features that enable a better characterization of the bass track in different songs. Furthermore, we aim to develop a general method to translate musicological knowledge into rules on feature values that can be easily evaluated. This approach is intended to facilitate the design of an instrument-related classifier that is trained by musicological knowledge – similar to an expert system. When analyzing real audio data, the strong dependence of a well-performing transcription system still remains the biggest challenge.

3. PREVIOUS APPROACHES

Various bass transcription algorithms have been proposed so far in [13], [11], [6], and [18]. They extract the score parameters of a bass track in polyphonic audio recordings. Still, transcription errors related to pitch and onset values appear due to the high complexity of overlapping instrument spectra. These errors affect the accuracy of the deduced high-level features. As shown in [16], high-level features can be derived from different music domains like instrumentation, texture, rhythm, dynamics, pitch statistics, melody, and chords. Offering a direct access to the relevant score parameters, symbolic audio data like MIDI receives preferential treatment in many publications.
The authors of [4] applied several statistical methods to derive high-level features from note onsets, pitches, and intervals. The versatility of complexity-based descriptors based on entropy, compression, and prediction has been shown in [15]. A set of musical features derived from the bass part was introduced in [19]. The authors restricted themselves to pitch-related features and distinguished between features characterizing the pitch variability and the pitch motion. Rhythmical aspects like the swing or syncopations have been investigated in various publications as for instance in [12] and [9]. In [16], [4], [19], and [1], genre classification solely on high-level features was covered.

4. NEW APPROACHES

4.1 High-level features

High-level features allow to model and quantify musical properties that are directly observable by experienced musicologists. These are for instance the key, the time signature or measure of the harmonic consonance in a piece of music. They can be deduced from the pitch, the onset time, and the duration values of all notes.

Melody-related features
By analyzing the course of the *absolute pitch* p_A, we derive features from the incidence rate of typical pitch progressions, such as *notes with constant pitch* or *chromatic note sequences* related to the overall number of notes and the overall *pitch range* in halftones. With reference to the simultaneously sounding chords of the harmony track, we derive a feature from the ratio of *chord notes* within the bass line. Besides, we convert the absolute pitch of each note into its *functional pitch* $p_{A,F}$. It represents the interval type between each bass note and the root note of the simultaneously sounding chord. We consider all interval types from primes to sevenths ($p_{A,F} \in [1,7]$), bigger intervals are mapped into this interval range. The incidence rates of all possible values of $p_{A,F}$ are used as features that provide key-independent information about the frequency of occurrence of different interval types related to the harmony accompaniment.
The prior use of root notes, octaves, and fifths of the current chord within a bass line does not allow a conclusive differentiation between major and minor based chords by exclusively investigating the bass accompaniment. Thus, a measure of *harmonic ambiguity* is calculated proportional to the occurrence rate of primes and fifths and inversely proportional to the occurrence rate of thirds as $F_{HA} = P(p_{A,F} = 1) + P(p_{A,F} = 5) - P(p_{A,F} = 3)$.
We use a simple bar-wise distance measure combining rhythmic and melodic similarity to detect the *dominant bass pattern*. Therefore, we compute a square matrix D_τ containing the similarity between the notes in each pair of bars. We use $D_\tau(k,m) = 0.5[(1 - N_{k,m}/N_k) + (1 - N_{m,k}/N_m)]$ where N_i denotes the number of notes in bar i and $N_{i,j}$ denotes the number of notes of bar i that have a note equivalent in bar j with the same pitch (p_A) and onset [mod$(\tau,1)$]. We choose the notes of bar $n_{dom} = n$ that minimizes $\sum_i D_{i,n}$ as the dominant pattern since this bar has the lowest overall distance to the other bars. Subsequently, measures of *tonal* and *rhythmic variation* are derived from the mean distance between all bars to bar n_{dom}. For the rhythmical variation, only the aforementioned onset condition of the note equivalent is taken into account.
The interval progression of the bass line is characterized by three different representations, namely the relative pitch $p_R \in [-12, 12]$ (mapped down to a two octave range), the relative pitch mapped to functional intervals $p_{R,F} \in [-7, 7]$ (to provide a representation independent of the key-type as described above), and the interval direction $p_{R,D} \in [-1, 1]$. Subsequently, several statistical properties such as entropy and relative number of non-zero elements of the probabilities of all parameter values are extracted as features. The measures of *constant direction* F_{CD} & *dominant direction* F_{DD} furthermore quantify the temporal ratio of note passages with constant interval direction and characterize the dominant direction. Thus, they measure to what extend a melody appears to be fluent. We use $F_{CD} = N\left[p_{R,D}(i) \equiv p_{R,D}(i+1)\right]/N_{Intervals}$ and $F_{DD} = N(p_{R,I} = 1)/N_{Intervals}$.

Rhythm-related features
The *beat grid* contains the temporal positions and indices of all beats corresponding to the current time signature. After its extraction, all note onset t and duration values Δt are mapped from seconds to certain multiples of the corresponding bar lengths (resulting in τ and $\Delta\tau$). This allows a tempo-independent extraction of rhythm-related features. We applied a similar approach as described in [12] to derive the *swing ratio* related to the 8th- and the 16th-note grid.
A measure of *syncopation* related to the both aforementioned temporal grids is derived by retrieving binary patterns (like for instance “1001” in an 16th-note grid representing two notes whereas the first one is played on a downbeat and the other one on the adjacent off-beat related to the 8th-note grid).
Based on the *dominant bass pattern* and its dynamic progression, we take the number of bass notes within each bar with a velocity above 60% of the maximum occuring bass note velocity as the measure of *accent sparsity*. Percussionists often use the bass-drum to “double” the main accents of the bass line. We measure the ratio of the number of notes that both instruments played rhythmically *in unison* to the sum of all notes played by the bass and the bass drum individually.

Structure-related features
In addition, features characterizing repeating melodic and rhythmic segments are derived. Therefore, we apply a simple pattern search algorithm (*Correlative Matrix Approach* [14]) on character strings derived from the aforementioned score parameters p_A, τ, and $\Delta\tau$.
We use the statistical properties mean, median, standard deviation, minimum, and maximum from each of the pattern parameters length, incidence rate, and mean distance between similar patters as features. Overall, all single- and multidimensional high-level features result in an 154-dimensional feature vector.

4.2 Concept-based framework

To improve genre classification, we aim at modeling common bass playing styles that are typical for certain music genres. Therefore, we apply a generic framework to translate known musicological properties into explicit restrictions on feature values. The assignment of weighting factors furthermore allows to take the importance of each property into account. In the following subsections, we introduce the terms *concept*, *class*, and *property* as the major components of the framework. Afterwards we explain how *relevance values* for both properties and classes are derived to measure their significance to the investigated piece of music and close with a detailed example. Hereafter, multidimensional variables are denoted in bold print.

Concepts & classes
The term *concept* represents a general approach to categorize music. Each concept is defined by a set of *classes* as

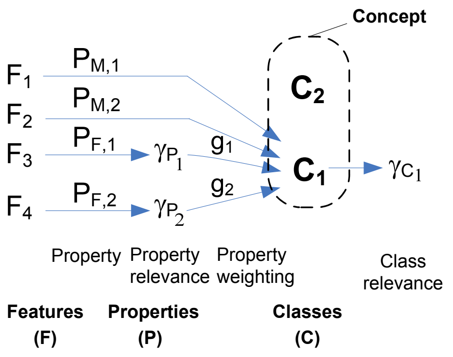


Figure 1. Concept-based framework

shown in Fig. 1. In this paper, we apply the concepts *BassPlayingStyle* (denoted as SB), which represents a common way of playing the bass in a specific music genre and *Genre* (denoted as G), which is a common category for musicologists. One well-known example is the bass playing style walking bass (defined as class *WalkingBass*), which is widely applied by bass players in different music genres related to *Swing*. It is covered as an example in the end of this section. The assignment between classes of both concepts is shown in Fig. 2.

Properties

Each class is defined by a number of *properties* $\boldsymbol{P}$. They translate its musicological description into explicit restrictions on the values of certain *features* $\boldsymbol{F}$.

We discern *mandatory properties* (M) and *frequent properties* (F). Mandatory properties are strictly need to be fulfilled whereas frequent properties are not mandatory for a certain class. A *weighting factor* $0 \leq g_i \leq 1$ is assigned to each frequent property. g_i is proportional to the importance of the corresponding property with regard to the current class.

Furthermore, properties are either *omnipresent* (O) or *conditional* (C). Omnipresent properties are constantly valid, whereas the validity of conditional properties depends on a certain condition. This may for instance be the presence of an instrument that a feature and thus a property is related to. Only if the condition is fulfilled, the corresponding property needs to be considered. Generally, the indices of P imply the corresponding property type. Examples are given in the end of this section. We derived the weighting factors and thresholds of all properties used in this paper from experiments with development data samples, which did not belong to the evaluation set.

Relevance values

The *property relevance value* γ_P measures to what extent a property P is fulfilled ($\gamma_P = 1$) or not ($\gamma_P = 0$). It is derived from the corresponding feature value F by using a *rating function* $r(F)$. This function depends on the type of restriction on the feature value F that is defined by P. For instance, we use $\gamma_P = r(F) = 0.5[\text{sgn}(F - V) + 1]$ to match the property $P \rightarrow F$ `isBiggerThan` V. The

A frequent use of chord tones is mandatory.
$P_{1,MO} \rightarrow F_{ChordToneRatio}$ `isBiggerThan` 0.3
2) The melodic direction is often constant within each bar. (important property - weighting factor $g_2 = 0.7$)
$P_{2,FO} \rightarrow F_{ConstantDirection}$ `isBiggerThan` 0.7
3) If quarter notes are primarily used (such as in slow and mid-tempo Jazz songs), there is a high swing factor related to the eighth note grid. (important property - weighting factor $g_3 = 0.8$)
`if` *Condition* ($F_{DominantRhythmicalGrid}$ `is` 4) $P_{3,FC} \rightarrow F_{SwingFactor,8}$ `isBiggerThan` 0.7
4) If eighth notes are primarily used (such as in up-tempo Jazz songs), there is a high swing factor related to the sixteenth note grid. (important property - weighting factor $g_4 = 0.8$)
`if` *Condition* ($F_{DominantRhythmicalGrid}$ `is` 8) $P_{4,FC} \rightarrow F_{SwingFactor,16}$ `isBiggerThan` 0.7
5) Chromatic note passages are occasionally used. (less important property - weighting factor $g_5 = 0.3$)
$P_{5,FO} \rightarrow F_{Chromatics}$ `isRelativelyHigh`

Table 1. Properties of the class *WalkingBass* (concept *BassPlayingStyle*)

rating function is designed in such a way that $0 \leq \gamma_P \leq 1$ is assured.

Subsequently, the *class relevance value* γ_C is derived for each class C from its corresponding property relevance values. γ_C quantifies to what extend a certain class is relevant for the musicological description of an analyzed piece of music.

We suggest the following algorithm to comply with the different property types. If all mandatory properties are given to be true, γ_C is calculated as a weighted sum over all frequent properties $\boldsymbol{\gamma_{P_F}}$ according to their normalized weighting factors $\widehat{\boldsymbol{g}}$ ($\sum \widehat{g}_i = 1$). Otherwise it is set to zero. This algorithm can be summarized as follows:

$$\gamma_C = \begin{cases} \sum_i \widehat{g}_i \gamma_{P_{F,i}} & \text{if } \gamma_{P_{M,j}} = 1 \;\forall\; P_{M,j} \in \boldsymbol{P_M}, \\ 0 & \text{else} \end{cases} \tag{1}$$

Example

As shown in Table 1, the class *WalkingBass* of the concept *BassPlayingStyle* is defined by 5 feature-related properties that are derived from musicological properties of this style.

5. EVALUATION

We use two data sets consisting of symbolic (MIDI) and real audio (AUDIO) each with 50 respectively 40 excerpts from each of the genres *PopRock* (POP), *Swing* (SWI), *Latin* (LAT), *Funk* (FUN), *Blues* (BLU), and *MetalHardRock* (MHR). All excerpts are derived from instrumental solo parts of the melody instruments between 20 and 35 seconds of length. Fig. 3 depicts all processing steps that precede the evaluation.

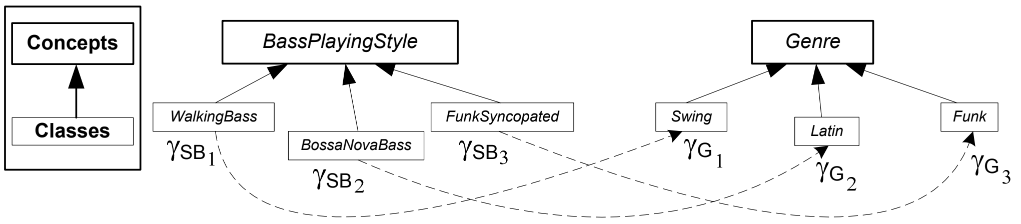


Figure 2. Assignment between the classes of the concepts *BassPlayingStyle* and *Genre*

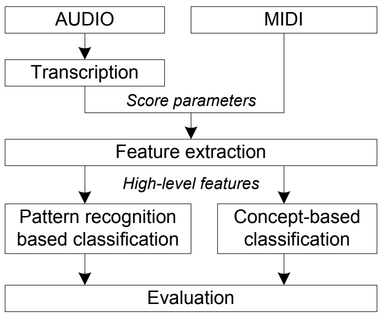


Figure 3. Processing flow-chart

5.1 Transcription & Pre-processing

We used the Transcription Toolbox [6] to extract the score parameters from real audio data. It provides algorithms to extract the bass, melody, harmony, and drum part as well as the beat grid information. As explained in Sec. 4.1, our aim is to focus on the bass and its interaction with the participating instruments. Concerning symbolic audio data, score parameters are directly extracted.

5.2 Feature Selection (FS) and Feature Space Transformation (FST)

The following feature selection and feature space transformation techniques have been utilized to reduce the dimensionality of the feature space.

Inertia Ratio Maximization using Feature Space Projection (IRMFSP).
IRMFSP was proposed in [17]. This FS algorithm is motivated by the ideas similar to Fisher's discriminant analysis. During each iteration of the algorithm, we look for the feature maximizing the ratio of between-class inertia to the total-class inertia. To avoid the next chosen feature to provide the same information on the next iteration, all features are orthogonalized to the selected one. In this evaluation we use the ISMFSP algorithms with the modifications proposed in [8].

Linear Discriminant Analysis (LDA)
LDA is one of the most often used supervised FST methods [10]. It is successfully applied as a pre-processing for audio signal classification. Original feature vectors are linearly mapped into new feature space guaranteeing a maximal linear separability by maximization of the ratio of between-class variance to the within-class variance. This mapping is conducted by multiplying the original $K \times N$ dimension feature matrix $\mathbf{X}$ with the transformation matrix $\mathbf{T}$. Reducing the dimension of the transformed feature vector from N to $D \leq N$ is achieved by considering only the first D column vectors of $\mathbf{T}$ for multiplication.

Generalized Discriminant Analysis (GDA)
Real-world classification routines often have to deal with non-linear problems, thus linear discrimination in the original feature space is often not possible. The idea of the FST technique GDA [3] is to map the features into higher dimensional (sometimes infinity dimensional) space, where the linear discrimination is possible. Dealing with a high dimensional space leads to an increase of the computation effort. To overcome this problem, the so called *kernel trick* is applied. The key idea of the kernel trick is to replace the dot product in a high-dimensional space with a kernel function in the original feature space.

5.3 Classification

We applied 4 known methods (SVM, GMM, NB, and kNN) as well as a novel concept-based approach for the purpose of classification.

Support Vector Machines
A Support Vector Machine (SVM) is a discriminative classifier, attempting to generate an optimal decision plane between feature vectors of the training classes [20]. Commonly for real-world applications, classification with linear separation planes is not possible in the original feature space. The transformation to the higher dimensional space is done using above mentioned kernel trick (we applied the RBF kernel in this paper). Transformed into a high-dimensional space, non-linear classification problems can become linearly solvable.

Gaussian Mixture Models
Gaussian Mixture Models (GMM) are commonly used generative classifiers.Single data samples of the class are interpreted as being generated from various sources and each source is modeled by a single multivariate Gaussian. The probability density function (PDF) is estimated as a weighted sum of the multivariate normal distributions. The parameters of a GMM can be estimated using the Expectation-Maximization algorithm [5].

Naive Bayes Classifier
Naive Bayes classifier (NB) is a simple probabilistic classifier. NB uses a strong assumption of feature dimensions being statistically independent and thus takes into account only means and variances over the feature dimensions for all training data of the class. Recently, applicability and efficiency of NB classifiers were discussed in detail in [21].

k-Nearest Neighbor
With k-Nearest Neighbor (kNN), the classification is based on the class assignment of the closest training examples in the feature space [7]. We used the Euclidean distance here. This type of discriminative classifier is also referred as instance based learning. The level of generalization of kNN can be tuned by adjusting the number of nearest neighbors k taken into account.

Novel approach: concept-based classifier
Using Eq. 1, we derive a class relevance value $\gamma_{SB_i} \equiv \gamma_{C_i}$ for each class of the concept *BassPlayingStyle*. We defined one common bass playing style for each of the 6 genres that were considered in the evaluation (see Sec. 5), namely *WalkingBass* (SWI), *BluesShuffle* (BLU), *FunkSyncopated* (FUN), *SteadyRiff* (MHR), *BossaNovaBass* (LAT), and *ChordRootAccompaniment* (POP). For our experiments, we used 5 different properties for each class.Using the assignment between the classes of both concepts as depicted in Fig. 2, the concept-based classifier estimates the genre $\widehat{G} = G_j$ that is assigned to the bass playing style SB_i with the highest class relevance value γ_{SB_i}. In case two or more bass playing styles related to different genres obtain the same class relevance values, the classification is considered to be correct if at least one of the candidates is related to the correct genre and false if not. As a proof of concept, we performed the evaluation experiment using the concept-based classifier on the MIDI data set.

6. RESULTS

Table 3 gives an overview over the classification scores for different FS / FST combination. For each combination, the parametrization with the best results is depicted. Further evaluation parameters such as the number of gaussians for the GMM classifiers, k for the kNN classifiers, and the number of dimensions after IRMFSP are given in brackets. We performed a 25-fold cross validation to derive mean classification scores and their standard deviations (given in brackets below) for each classifier. As shown there, best mean classification accuracies for the MIDI and AUDIO data set of 81.47% and 46.85% have been achieved applying a combined IRMFSP - GDA preprocessing for both data sets. Above all, we expect transcription errors affecting note pitch values, onset values and beat grid information to cause significantly lower classification scores for real audio data. For both data sets, the application of feature selection and feature space transformation algorithms clearly increases the accuracy values of the subsequent classifiers.

	BLU	FUN	LAT	MHR	POP	SWI
BLU	**68.0**	-	4.0	-	-	28.0
FUN	28.0	**46.0**	4.0	4.0	4.0	14.0
LAT	16.0	-	**70.0**	-	2.0	12.0
MHR	34.0	8.0	6.0	**34.0**	2.0	16.0
POP	36.0	-	20.0	2.0	**6.0**	36.0
SWI	36.0	-	22.0	-	-	**42.0**

Table 2. Confusion matrix of the concept-based classifier (MIDI data set) in %

As depicted in Table 2, the concept-based classifier achieved a mean classification accuracy of 44.3% varying in a strong way for different genres. Best results have been obtained for *Latin* (70.0%) and *Blues* (68.0%). The low results for *Pop* (6.0%) and *MetalHardRock* (34.0%) lead to the assumption, that modeling only one bass playing style per genre is not sufficient due to the high variability in the applied data set. Further steps include the evaluation based on a larger database.

7. CONCLUSIONS & FUTURE WORK

In this paper, we introduced a novel set of transcription-based high-level features related to the rhythmic, melodic, harmonic, and structural description of bass lines. Furthermore, we presented a new approach to model musical knowledge of musical styles as properties related to the values of transcription-based high-level features. The main advantage of concept-based classification approach is that significantly fewer features are necessary to model each class as in common machine learning approaches. Future steps include modeling additional genre-specific bass playing styles as well as transferring the proposed method onto other frequently used instruments like the guitar.

8. ACKNOWLEDGMENTS

This work has been partly supported by the german research project *GlobalMusic2One*[1] funded by the Federal Ministry of Education and Research (BMBF-FKZ: 01/S08039B). The authors would like to thank Paul Bräuer for the fruitful discussions.

9. REFERENCES

[1] J. Abeßer, C. Dittmar, and H. Großmann. Automatic genre and artist classification by analyzing improvised solo parts from musical recordings. In *Proceedings of the Audio Mostly*, 2008.

[2] J.-J. Aucouturier, B. Defreville, and F. Pachet. The bag-of-frame approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. *Journal of the Acoustical Society of America*, 122(2):881–891, 2007.

[3] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

[4] P. J. Ponce de Léon and J. M. Iñesta. Pattern recognition approach for music style identification using shallow statistical

[1] `http://www.globalmusic2one.de`

Dataset	FS / FST	Dim.	SVM	GMM(2)	GMM(3)	GMM(5)	GMM(10)	NB	kNN(1)	kNN(5)	kNN(10)
MIDI	-	154	69.13 (8.34)	67.45 (9.08)	66.28 (9.45)	59.52 (6.21)	60.31 (5.97)	60.03 (7.27)	66.88 (6.51)	64.17 (5.61)	62.69 (7.39)
	LDA	5	63.06 (7.20)	59.24 (5.10)	61.22 (6.69)	53.44 (7.43)	59.23 (14.43)	60.50 (7.33)	60.14 (6.97)	62.15 (7.66)	62.38 (8.53)
	GDA($\gamma = 2^{-7}$)	5	77.60 (7.65)	77.60 (7.65)	77.60 (7.65)	44.04 (8.31)	18.73 (9.54)	18.37 (6.27)	77.60 (7.65)	77.60 (7.65)	77.60 (7.65)
	IRMFSP(20)	20	73.82 (7.89)	58.70 (8.08)	65.07 (8.23)	63.75 (10.20)	64.21 (7.08)	57.99 (6.38)	**78.06** (7.31)	71.70 (6.84)	68.85 (8.72)
	IRMFSP(80) + LDA	5	72.15 (8.93)	69.87 (10.67)	69.34 (7.60)	67.95 (9.81)	65.20 (11.70)	69.65 (8.52)	69.45 (9.02)	70.48 (7.93)	69.66 (6.65)
	IRMFSP(40) + GDA($\gamma = 2^{-5}$)	5	76.99 (13.88)	19.32 (5.07)	20.15 (9.26)	13.30 (5.60)	16.09 (2.85)	18.37 (6.27)	81.10 (6.39)	**81.47** (6.20)	**81.47** (6.20)
AUDIO	-	154	41.33 (8.33)	33.45 (8.59)	34.95 (8.98)	33.73 (10.33)	33.80 (10.62)	27.24 (8.33)	36.54 (10.35)	31.42 (11.61)	32.25 (9.66)
	LDA	5	32.98 (7.39)	32.82 (7.46)	30.16 (6.88)	28.90 (7.95)	28.76 (8.33)	34.25 (7.58)	31.16 (9.00)	33.84 (8.66)	34.10 (8.28)
	GDA($\gamma = 2^{-9}$)	5	42.74 (11.53)	42.74 (11.53)	42.74 (11.53)	27.09 (15.08)	15.02 (6.90)	12.79 (6.63)	42.74 (11.53)	42.74 (11.53)	42.74 (11.53)
	IRMFSP(40)	40	43.26 (11.76)	39.19 (10.83)	38.31 (10.56)	39.83 (12.93)	36.62 (9.86)	26.69 (8.87)	45.23 (11.70)	42.04 (12.06)	37.83 (10.67)
	IRMFSP(20) + LDA	5	43.80 (10.61)	41.66 (11.16)	42.28 (10.68)	41.32 (11.50)	40.58 (13.52)	**43.90** (12.09)	35.29 (9.10)	40.69 (10.75)	41.48 (10.24)
	IRMFSP(40) + GDA($\gamma = 2^{-5}$)	5	**46.85** (9.73)	**46.85** (9.73)	**46.85** (9.73)	26.59 (13.75)	16.64 (7.10)	12.79 (6.63)	**46.85** (9.73)	**46.85** (9.73)	**46.85** (9.73)

Table 3. Mean classification accuracy [%] for the MIDI and AUDIO data set (standard deviation [%] given in brackets)

descriptors. *IEEE Transactions on System, Man and Cybernetics - Part C : Applications and Reviews*, 37(2):248–257, March 2007.

[5] A. P. Dempster, N. M. Laird, and D. B. Rdin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[6] C. Dittmar, K. Dressler, and K. Rosenbauer. A toolbox for automatic transcription of polyphonic music. In *Proceedings of the Audio Mostly*, 2007.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2nd edition, November 2000.

[8] S. Essid. *Classification automatique des signaux audio-fréquences : reconnaissance des instruments de musique*. PhD thesis, Université Pierre et Marie Curie, Paris, France, December 2005.

[9] P. Flanagan. Quantifying metrical ambiguity. In *Proceedings of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 635–640, 2008.

[10] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, September 1990.

[11] M. Goto. A real-time music-scene-description system - predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43:311–329, 2004.

[12] F. Gouyon. *A computational approach to rhythm description - audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing*. PhD thesis, University Pompeu Fabra, 2005.

[13] S. W. Hainsworth and M. D. Macleod. Automatic bass line transcription from polyphonic audio. In *Proceedings of the Int. Computer Music Conf. (ICMC)*, 2001.

[14] J.-L. Hsu, C.-C. Liu, and A. L. P. Chen. Discovering nontrivial repeating patterns in music data. In *IEEE Transactions on Multimedia*, volume 3 of *IEEE Transactions on Multimedia*, pages 311–324, September 2001.

[15] S. T. Madsen and G. Widmer. A complexity-based approach to melody track identification in midi files. In *Proceedings of the Int. Workshop on Artificial Intelligence and Music (MUSIC-AI)*, January 2007.

[16] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proceedings of the Int. Conf. in Music Information Retrieval (ISMIR)*, pages 525–530, 2004.

[17] G. Peeters and X. Rodet. Hierarchical gaussian tree with inertia ratio maximization for the classification of large musical instruments databases. In *Proceedings of the 6th Int. Conf. on Digital Audio Effects (DAFx)*, London, UK, 2003.

[18] M. P. Ryynänen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32:72–86, 2008.

[19] Y. Tsuchihashi, T. Kitahara, and H. Katayose. Using bass-line features for content-based mir. In *Proceedings of the Int. Conf. on Music Information Retrieval (ISMIR)*, pages 620–625, 2008.

[20] V. N. Vapnik. *Statistical learning theory*. Wiley New York, 1998.

[21] H. Zhang. The optimality of naive bayes. In V. Barr and Z. Markov, editors, *Proceedings of the FLAIRS Conf.* AAAI Press, 2004.

FROM MULTI-LABELING TO MULTI-DOMAIN-LABELING: A NOVEL TWO-DIMENSIONAL APPROACH TO MUSIC GENRE CLASSIFICATION

Hanna Lukashevich, Jakob Abeßer, Christian Dittmar, Holger Grossmann
Fraunhofer Institute for Digital Media Technologies, Ilmenau, Germany
{lkh, abr, dmr, grn}@idmt.fraunhofer.de

ABSTRACT

In this publication we describe a novel two-dimensional approach for automatic music genre classification. Although the subject poses a well studied task in Music Information Retrieval, some fundamental issues of genre classification have not been covered so far. Especially many modern genres are influenced by manifold musical styles. Most of all, this holds true for the broad category "World Music", which comprises many different regional styles and a mutual mix up thereof. A common approach to tackle this issue in manual categorization is to assign multiple genre labels to a single recording. However, for commonly used automatic classification algorithms, multi-labeling poses a problem due to its ambiguities. Thus, we propose to break down multi-label genre annotations into single-label annotations within given time segments and musical domains. A corresponding multi-stage evaluation based on a representative set of items from a global music taxonomy is performed and discussed accordingly. Therefore, we conduct 3 different experiments that cover multi-labeling, multi-labeling with time segmentation and the proposed multi-domain labeling.

1. INTRODUCTION

In the field of Music Information Retrieval, automatic genre classification has been covered in numerous publications. Although genre labels as being used in online music stores or music journals mostly represent marketing terms, genre itself embodies both a culturally relevant term and an intuitive concept to categorize music. Single genre labels usually reflect some sort of stylistic elements inherent to a piece of music. Especially nowadays, music is influenced by an increasing amount of different musical styles. This leads to the necessity of describing single recordings with multiple genre labels. At the same time, this increases ambiguity in case a distinct genre classification result is intended. We stumbled across this problem while attempting to train supervised classifiers for a given sub-genre classification taxonomy of global music content. It is obvious that the broad term "World Music" is one of the most ill-defined tags when being used to lump all "exotic genres" together. It lacks justification because this category comprises such a huge variety of different regional styles, influences, and a mutual mix up thereof. On the one hand, retaining the strict classification paradigm for such a high variety of musical styles inevitably limits the precision and expressiveness of a classification system that shall be applied to a world-wide genre taxonomy. On the other hand, multi-labeling is not straight forward to deploy for automatic supervised classification since data sets with multiple class assignments are not well suited as training data due to their inherent ambiguity. To tackle these issues, we considered to break down the multi-label genre classification problem into a set of single-label genre classification tasks, where each classifier can be trained and optimized using well-defined data. The novelty of the proposed 2-dimensional approach for multi-label genre classification consists in the combination of segment-wise and domain-specific genre classifications. The term "domain" refers to the perceived semantic dimensions of music in which the classification is performed, in our case timbre, rhythm and tonality, which represents melody and harmony. We call the introduced approach "multi-domain labeling". In this paper we evaluate and discuss the potential of a more detailed approach directly, compared to multi-label genre classification. The rest of this paper is organized as follows. We give an overview over related work to this topic in the subsequent section. Then, after explaining our novel approach in section 3, we give an overview over the utilized databased as well as the manual genre annotations corresponding to the proposed method in section 4. In the following section, we describe the 3 evaluation experiments that we performed. Details on feature extraction, feature selection, feature space transformation as well as on the applied classification algorithms are presented in section 6. After discussing the results of the experiments in section 7, we conclude our work and provide perspectives for future directions in section 8.

2. RELATED WORK

Various classification schemes for automatic genre classification have been proposed during the last years. [17] provides a comprehensive overview over existing publications in the domain. There different approaches related to expert systems, unsupervised classification, and supervised classification systems have been covered. Considering the general confusion between similar genres, relaxing the strict classification paradigm and allowing for multiple-genre classification seemed to be a reasonable future direction to the authors to implement a more realistic classification system. The earlier work of [3] gave a more pessimistic outlook by considering the term genre to be intrinsically ill-defined and hardly grounded in timbre characteristics. Already in one of the basic works on music genre classification by Tzanetakis [21], separate feature sets representing timbre, rhythm, and tonality were introduced that allowed for different types of similarity measures. However, separate domain-specific genre models have not been proposed there. [17] provides also an overview of different classifier approaches applied in genre classification, such as Support Vector Machines (SVM), Hidden Markov Models (HMM) or Artificial Neural Networks (ANN). Other publications such as [21] utilized Gaussian mixture models (GMM) for this purpose. Among others, the authors of [18] used ensemble-based decision approaches namely a one-against all and a round-robin algorithm to combine binary classifiers. Different feature-space transformation methods such as Linear Discriminant Analysis are applied to increase discrimination between the classes resulting in better classifications scores [17]. Novel musically motivated low- and mid-level features such as the Octave-based Modulation Spectral Contrast [11] or multiscale spectro-temporal modulation features [15] were reported to outperform conventional features such as Mel-Frequency Cepstral Coefficients (MFCCs). Moreover, an increasing amount of publications focused on high-level features that are supposed to better characterize musicological properties as described for instance in [14], [16], and [1]. Further relevant publications regarding feature design are referenced in Section 6.1.

While most research has been conducted using western popular music, only a few works were related to more diverse global music content. A study on the applicability of different classifiers for automatic genre classification of traditional Malaysian music was conducted in [7]. The general issue of multi-label annotations has been addressed only in a few publications so far. In [13], the authors experimented with SVM-based "binary relevance" multi-label genre classification in conjunction with MARSYAS-based features [21]. This approach was continued in [23], where the authors modified a k-Nearest Neighbors classifier in order to handle multi-label data directly. In [20], automatic mood estimation was modeled as a multi-label classification task where every item may belong to more than one class. To the current knowledge of the authors, no publication so far discussed an approach similar to multi-domain labeling, that will be explained in detail in the following section.

3. MULTI-DOMAIN-LABELING

As explained in Section 1, while dealing with musical content from various regional music genres (often referred to as "World music"), the problem frequently arises that songs cannot solely be labeled with one single genre label. Instead, various rhythmic, melodic and harmonic influences conflate into multi-layered mixtures. Common classifier approaches fail because of their immanent assumption that for all song segments, one dominant genre exists and thus is retrievable.

To overcome these problems, we introduce a novel approach called "multi-domain labeling". We aim at breaking down multi-label annotations towards single-label annotations within different musical domains, namely *timbre*, *rhythm*, and *melody / harmony* that are well-known aspects of perceivable music similarity. Furthermore, a separate annotation of each temporal segment of the overall song is enabled. This leads to a more meaningful and realistic two-dimensional description of multi-layered musical content. In addition, the approach facilitates a more precise training of a classifier by avoiding fuzzy multi-labeled data samples.

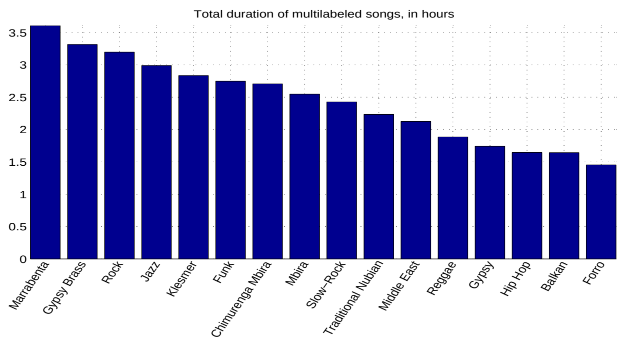


Figure 1. Structure of the database

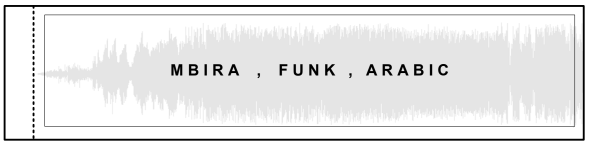
4. DATABASE & ANNOTATIONS

The music collection that we used for our investigations consists of 430 full-length tracks from the 16 world music genres. For each genre, the database includes approximately two hours of music on average (see Fig. 1 for details). This music data collection was provided by the content partner of the research project *GlobalMusic2One*[1]. The research project involves educated musicologists working with a world music label and being in regular contact with musicians associated with the applied genres. Annotations were manually made by using an annotation software allowing to label music genres in different domains with regard to an arbitrary amount of time segments. This annotation software includes automatic segmentation algorithm, which makes the first suggestion in order to speed up the annotation process. The experts had a fully freedom to modify borders and assessments of the segments in each of domains.

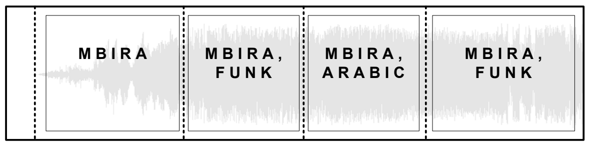
In this paper, we applied a flat taxonomy with all aforementioned genres considered to be situated at the same hierarchical level. Above all, for our experiments we selected tracks that have been annotated with multi-labels

[1] `http://www.globalmusic2one.net`

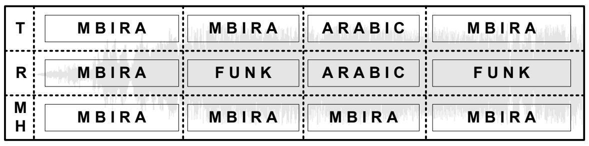
in at least one time segment. To evaluate our new annotation approach, the data set was annotated following the principles of multi-domain-labeling as described in Sec. 3. Music experts were allowed to assign up to 4 different genre concepts for each segment - a global genre, a timbre-related genre, a rhythm-related genre, and a genre related to the melodic and harmonic content. The three domain specific annotations were not mandatory. If there were multiple genre influences audible in a single segment, the experts were only allowed to assign one genre label for each domain. This proceeding ensured single-label annotations within each segment and domain. One observation that we made was that these domain-specific genre influences seem to be stable for each segment. The resulting label cardinality (average number of labels per track) of multi-labeled songs per genre was between 1.1 and 2.0 for the selected genres, with 1.0 being a genre that has never been assigned in conjunction with another genre. The label cardinality appeared to be different depending on the music genres.



(a) Experiment 1: Multi-labeling



(b) Experiment 2: Multi-labeling with time segmentation



(c) Experiment 3: Multi-domain labeling (**T**imbre, **R**hythm, **M**elody/**H**armony)

Figure 2. Evaluation experiments

5. THREE EVALUATION EXPERIMENTS

To evaluate the improvement of the classifier performance, we perform three different experiments as depicted in Fig. 2(a) - 2(c). Therefore, we are moving stepwise from the fuzzy case of multi-labeled songs towards single-labeled segments within different musical domains as described in the previous section.

Multi-labeling (Exp.1)
In the first experiment, all multi-labeled songs are generally used to train multiple classifiers, more precisely all classifier related to the annotated genres.

Multi-labeling with time segmentation (Exp.2)
Bearing the temporal structure of music in mind, we furthermore consider single segments in the second experiment. Multi-labeled segments are repeatedly used as class instances according to their assigned genre labels.

Multi-domain-labeling with time segmentation (Exp.3)
In the third experiment, we are using temporal segments to train three different domain-related classifiers. Therefore, we restricted ourselves to features that can be semantically assigned towards the particular musical domain, as will be detailed in 6.1.

6. SYSTEM WORK-FLOW

6.1 Feature extraction

For the experiments conducted in this paper, we utilize a broad palette of features commonly reported in the literature (see Sec. 2). Besides low-level acoustic features, several mid-level representations [4] are extracted. These measures are computed from excerpts of approximately 5 seconds duration by deriving specialized descriptive measures (including musical knowledge) from the observed evolution of low-level features. Besides indifferent usage of all features (in Exp. 1 and Exp. 2), groups of features are assigned to the aforementioned domains in the following manner.

Timbre
In addition to common features, such as Mel-Frequency Cepstral Coefficients (MFCC), Audio Spectrum Centroid (ASC), Spectral Crest Factor (SCF) or Spectral Flatness Measurement (SFM), modulation spectral features [2] have proved to be extremely useful to capture short term dynamics of the low-level features. We applied a cepstral low-pass filtering to the modulation coefficients to reduce their dimensionality and decorrelate them as described in [6].

Rhythm
All rhythmic features used in the current setup are derived from excerpts of the different bands of the Audio Spectrum Envelope (ASE) feature. Part of the measures, such as the Percussiveness [22] and the Envelope Cross-Correlation, are based on the envelope signals. The other part is derived from the Auto Correlation Function (ACF) domain. Besides the measures described in [6], the log-lag ACF and its descriptive statistics are extracted according to [10].

Tonality
Tonality descriptors are computed from a Chromagram based on Enhanced Pitch Class Profiles (EPCP) [12], [19]. The EPCP undergoes a statistical tuning estimation and correction to account for tunings deviating from the equal tempered scale. Most important, the so-called symmetry model, a pitch-space representations as described in [9] are derived from the Chromagram as mid-level features. The model provides an analytic description of aspects of musical consonance and dissonance, as well as functional relationships between probable notes.

6.2 Dimension Reduction

MIR systems usually use a multitude of low-level and mid-level acoustic features. Each feature is designed to correlate with one of the aspects of perceptual similarity, e.g. timbre, tempo, loudness or harmony. The distinct acoustical features are joined together into so called acoustical feature vectors. While temporal changes in one feature often correspond to temporal changes in the other feature (for instance, timbre is changing along with loudness), the individual dimensions of the feature vectors can often be strongly correlated and cause information redundancy. These raw feature vectors could cause various problems on classification stage. One of the usual ways to suppress redundant information in the feature matrix is to utilize dimension reduction techniques. Their purpose is to decrease the feature dimension N while keeping or even revealing the most characteristic data properties. Generally, all dimension reduction methods can be divided into supervised and unsupervised ones. Among the unsupervised approaches the one most often used is *Principal Component Analysis* (PCA). The key idea of PCA [8] is to find a subspace whose basis vectors correspond to the maximum-variance directions in the original feature space. Dimension reduction is obtained then by simply discarding those column vectors with the smallest eigenvalues.

6.3 Classification

In this section we shortly describe the applied classifier and bring the architecture details regarding all three experiments.

Gaussian Mixture Models
Gaussian Mixture Models (GMM) is a commonly used generative classifier. Single data samples of the class are thought of as generated from various sources and each source is modeled by a single multivariate Gaussian. The probability density function (PDF) of the feature frames is estimated as a weighted sum of the multivariate normal distributions. Each single i-th mixture is characterized by its mean vector μ_i and covariance matrix Σ_i. Thus, a GMM is parametrized in $\Theta = \{\omega_i, \mu_i, \Sigma_i\}$, $i = \overline{1, M}$, where ω_i is the weight of the i-th mixtures and $\sum_i \omega_i = 1$. The generalization properties of the model can be adjusted by choosing the number of Gaussian mixtures M. The parameters of the GMM can be estimated using the Expectation-Maximization algorithm [5].

Classifier architecture for three experiments
On the classification stage for each data frame the likelihoods of all class models are calculated. We do not use prior distribution information. The classification decision is therefore made using maximum likelihood rule. In a case of Exp. 1 and Exp. 2 the same data samples may belong to multiple data classes. To tackle the problem, here the classification task is reduced to a set a binary classification decisions, where every binary classifier H_c is trained to make a binary decision (if the data sample belong to a class c or not). These decisions of binary classifiers are joined together to form the multi-label classification. In a case of Exp. 3 as described above only single labels are used within each domain and time segment. Thus for each domain we train one GMM classifier. On the classification stage firstly each domain is classified and post-processed (see Sec. 6.4 for details) independently, and later the results for all domains are joint together.

6.4 Post-processing

Classification with GMM results in class decision for each frame of the feature vector. Thus we apply the following post-processing procedure to reduce frame-level classification to the full-track multi-labels. For Exp. 1 and Exp. 2 the procedure is identical. For all frames of the track for each of the genres we sum up the number of frames associated to these genres. Then be build the normalized histogram of these data. The maximum of this histogram is pointing out the most probable genre for this track. As we are expecting more then single label per track, probably, we also have to accept the second maximum of the normalized histogram. This decision is made by a simple thresholding of the normalized histogram. The track is considered to be associated to those genres, where the values of the normalized histogram are above the threshold. As the histogram is normalized, the threshold is set to $(0, \ldots, 1)$. The choice of the threshold crucially influences the performance of the system. For instance, too low threshold causes high recall values, but might lead to poor precision. Thus the threshold values have to be optimized for each of the experiments. In a case of Exp. 3 we first perform the thresholding for each of domains independently, and then joint the results.

6.5 Evaluation Measures

In multi-label classification each data sample (in our case each song or song segment) is associated with a set of labels $Y \subseteq L$, where L is a full set of labels. Let D be a multi-label dataset, consisting of $|D|$ multi-label examples (X_i, Y_i), $i = 1 \ldots |D|$, $Y_i \subseteq L$, where X_i is a feature matrix of the data example i and Y_i is a set of (ground-truth) labels associated to the data example i. The label cardinality of D is defined as follows:

$$LC(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} |Y_i| \,. \tag{1}$$

Given the multi-label classifier H, the estimated set of labels for sample i is $Z_i = H(X_i)$. The traditional information retrieval evaluation measures for multi-label case are written as:

$$Precision(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|}, \tag{2}$$

$$Recall(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i|}, \tag{3}$$

$$F\_measure(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{2 * |Y_i \cap Z_i|}{|Y_i| + |Z_i|}. \tag{4}$$

7. RESULTS

First of all we detail the settings of the system. The feature extraction procedure results in 233 dimensions for timbre features, 768 dimension for rhythmic features, and 187 dimensions for tonal features. All in all it leads to 1188 dimensions of the feature vector. It is well known that GMMs are sensitive to the curse of dimensionality. As the available annotated database is relatively small, we applied PCA to reduce the dimensionality of feature vectors within each of domains to 100 dimensions. The PCA algorithm has been trained on the randomly chosen training set (70% of the database) and then applied to the test set (30% of the database). This PCA-transformed data have been used in all 3 experiments. The GMMs have been trained with 1, 5, 20, and 50 mixtures, only diagonal covariance matrices have been used. The threshold for the post-processing (as described in Sec. 6.4) has been varied within a range of 0 and 1 for Exp. 1, Exp. 2, and for each of three domains in Exp. 3. Figure 3 depicts the dependency of the F-measure on the thresholding for all above mentioned cases. It is interesting to note, that for Exp. 1 and Exp. 2 achieved F-measure significantly differs depending on the amount of mixtures in the GMM, while in all domains for Exp. 3 the values of F-measure become comparable. Using 5 mixtures results into highest F-measure values for all experiments. The optimal thresholds values are within a range of 0.15 and 0.25.

Within Exp. 3 we found out, that the optimal thresholds for each of domains separately do not form the optimal combination of the thresholds leading to the best F-measure performance when the domains are joined together. Thus, in a case of GMM1 (using only one gaussian to model the class) the optimal thresholds in all domains are found within a range of 0.20 and 0.25, while in a case of GMM20 the optimal thresholds lies within a range of 0.30 and 0.35. Figure 4 depicts the F-measure performance for all three experiments. The F-measure values for each number of mixtures in GMM are increased for Exp. 3 in comparison to Exp. 1 and Exp. 2. The best performance is achived in Exp. 1 for GMM with 5 mixtures reaching the F-measure of 0.61. The significant performance raise of about 10% is observed for the case of using only one gaussian to model the class information. It can be explain with a fact, that in a case of Exp. 3 the classes are less overlapped and easier to model then in a case of the set of binary classifiers (as in Exp. 1 and Exp. 2).

Note that for Exp. 3 the involved GMMs include about two times less free parameters than in a case of Exp. 1 and Exp. 2. As we used only diagonal covariance matrices, the number of free parameters for each GMM can be approximated as $m \cdot (2d + 1)$, where m is a number of mixtures and d is the dimensionality of the feature vector. Thus for Exp. 3 the number of all free parameters comprises $3 \cdot k \cdot m\,(2d' + 1)$, where k is a number of classes and d' is the features dimensionality within one domain; GMMs are trained within each of three domains. Whilst in a case of Exp. 1 and Exp. 2 the amount of free parameters for the set of binary classifiers reaches $2k \cdot m\,(2 \cdot 3d' + 1)$.

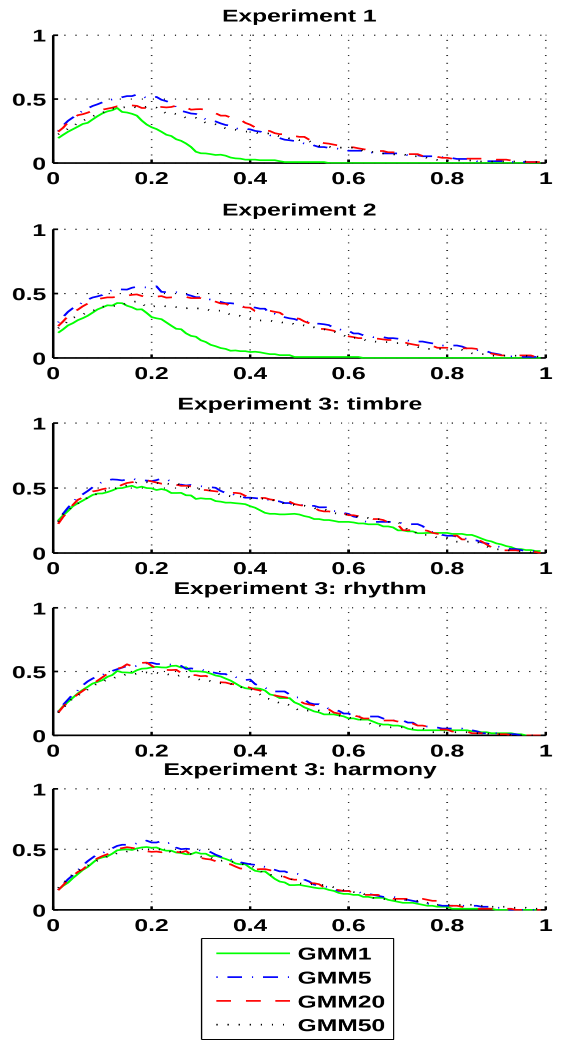


Figure 3. Dependency of F-measure on the thresholding while post-processing as described in Sec. 6.4. For Exp. 1 and Exp. 2 the F-measure performance strongly depends on the number if mixtures in GMM.

8. CONCLUSIONS & FUTURE WORK

The paper presented a novel two dimensional approach to music genre classification. It allows to decompose the multi-label classification problem into multiple single-class classification problems by breaking it down in two dimensions. First results demonstrate high potential of the proposed approach. Future work will be directed towards applying Support Vector Machines as alternative classification technique, as it has been proved to perform better than GMM for binary classification. In a case of multi-domain classification we shall make use of supervised feature selection and feature space transformation methods, which can not be utilized in a case of multi-label classification. Furthermore, in the context of the research project *GlobalMusic2One*, we are going to use *vocals* and *instrumentation* as additional domains. We believe the presented approach to be extensible to other music genres as the semantic partitioning of music into different musical domains is universal for most of the world's regional music styles.

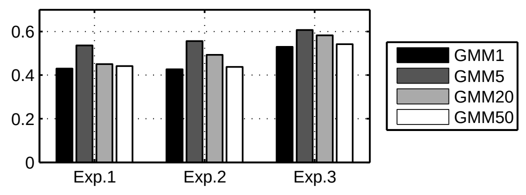


Figure 4. F-measures for all three experiments

9. ACKNOWLEDGMENTS

This work is part of the German research project *Global-Music2One* funded by the German Ministry of Education and Research (BMBF-FKZ: 01/S08039).

10. REFERENCES

[1] J. Abeßer, C. Dittmar, and H. Großmann. Automatic genre and artist classification by analyzing improvised solo parts from musical recordings. In *Proc. of the Audio Mostly Conference (AMC)*, Piteå, Sweden, 2008.

[2] L. Atlas and S. A. Shamma. Joint acoustic and modulation frequency. *EURASIP Journal on Applied Signal Processing*, 2003:668–675, 2003.

[3] J.-J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32:83–93, 2003.

[4] J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proc. of Intl. Conf. on Music Information Retrieval (ISMIR)*, London, UK, 2005.

[5] A.P. Dempster, N. M. Laird, and D. B. Rdin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[6] C. Dittmar, C. Bastuck, and M. Gruhne. Novel mid-level audio features for music similarity. In *Proc. of the Intl. Conf. on Music Communication Science (ICOMCS)*, Sydney, Australia, 2007.

[7] S. Doraisamy, S. Golzari, N. M. Norowi, M. N. B. Sulaiman, and N. I. Udzir. A study on feature selection and classification techniques for automatic genre classification of traditional Malay music. In *Proc. of Intl. Conf. on Music Information Retrieval (ISMIR)*, Philadelphia, Pennsylvania, USA, 2008.

[8] K. Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition (Computer Science and Scientific Computing Series)*. Academic Press, September 1990.

[9] G. Gatzsche, M. Mehnert, D. Gatzsche, and K. Brandenburg. A symmetry based approach for musical tonality analysis. In *Proc. of Intl. Conf. on Music Information Retrieval (ISMIR)*, Vienna, Austria, 2007.

[10] M. Gruhne and C. Dittmar. Improving rhythmic pattern features based on logarithmic preprocessing. In *Proc. of the 126th AES Convention, Munich, Germany*, 2009.

[11] C.-H. Lee, J.-L. Shih, K.-M. Yu, and J.-M. Su. Automatic music genre classification using modulation spectral contrast feature. In *Proc. of the IEEE Intl. Conf. on Multimedia and Expo (ICME)*, 2007.

[12] K. Lee. Automatic chord recognition from audio using enhanced pitch class profile. In *Proc. Intl. Computer Music Conf. (ICMC), New Orleans, USA*, 2006.

[13] T. Li and M. Ogihara. Detecting emotion in music. *Proceedings of the Fifth International Symposium on Music Information Retrieval*, pages 239–240, 2003.

[14] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proc. of Intl. Conf. on Music Information Retrieval (ISMIR)*, 2004.

[15] I. Panagakis, E. Benetos, and C. Kotropoulos. Music genre classification: A multilinear approach. In *Proc. of Intl. Conf. on Music Information Retrieval (ISMIR)*, 2008.

[16] C. Pérez-Sancho, P. J. Ponce de León, and J. M. Iñesta. A comparison of statistical approaches to symbolic genre recognition. In *Proc. of the Intl. Computer Music Conf. (ICMC)*, pages 545–550, 2006.

[17] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine*, 23:133–141, 2006.

[18] C. N. Silla, C. A. A. Kaestnerlso, and A. L. Koerich. Automatic music genre classification using ensemble of classifiers. In *Proc. of IEEE Intl. Conf. on Systems, Man, and Cybernetics*, pages 1687–1692, October 2007.

[19] M. Stein, B. M. Schubert, M. Gruhne, G. Gatzsche, and M. Mehnert. Evaluation and comparison of audio chroma feature extraction methods. In *Proc. of the 126th AES Convention, Munich, Germany*, 2009.

[20] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proc. of Intl. Conf. on Music Information Retrieval (ISMIR)*, Philadelphia, Pennsylvania, USA, 2008.

[21] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[22] C. Uhle, C. Dittmar, and T. Sporer. Extraction of drum tracks from polyphonic music using independent subspace analysis. In *Proc. of the 4th Intl. Symposium on Independent Component Analysis*, 2003.

[23] A. Wieczorkowska, P. Synak, and Z. W. Ras. *Multi-Label Classification of Emotions in Music*. Springer Berlin Heidelberg, 2006.

21ST CENTURY ELECTRONICA: MIR TECHNIQUES FOR CLASSIFICATION AND PERFORMANCE

Dimitri Diakopoulos, Owen Vallis, Jordan Hochenbaum, Jim Murphy, Ajay Kapur

California Institute of the Arts
Valencia, CA USA
{ddiakopoulos,jamesmurphy}
@alum.calarts.edu

New Zealand School of Music
Wellington, NZ
{ajay.kapur,vallisowen,hochenjord}
@nzsm.ac.nz

ABSTRACT

The performance of electronica by Disc Jockys (DJs) presents a unique opportunity to develop interactions between performer and music. Through recent research in the MIR field, new tools for expanding DJ performance are emerging. The use of spectral, loudness, and temporal descriptors for the classification of electronica is explored. Our research also introduces the use of a multi-touch interface to drive a performance-oriented DJ application utilizing the feature set. Furthermore, we present that a multi-touch surface provides an extensible and collaborative interface for browsing and manipulating MIR-related data in real time.

Keywords: Electronica, Electronic Dance Music, Genre Classification, User Interfaces, DJ, Multi-touch.

1. INTRODUCTION

Electronic dance music, often referred to as Electronica, is an overarching collection of genres that focus predominately on rhythmic motifs & repeating loops. A task of the electronica DJ is to compile a set-list of music for performance. Additionally, DJs are always looking for ways to expand the interactivity of their performances through the use of new tools. The primary goal of this work is to give the modern, digital DJ access to a wider range of performance options using MIR techniques such as feature extraction, genre classification, and clustering. Combined with advances in tabletop computing, these techniques have made it possible to add a layer of interactivity to automatic playlist generation.

In the following section we detail related work on music features and electronic performance interfaces including recent work in tabletop computing. In the remainder of the paper we discuss our feature extractors, genre classification results and the interface that we developed to enable DJs to interact with those results to create set-lists. We conclude the paper with a discussion of future work in interactive MIR powered DJ applications and tabletop computing.

2. RELATED WORK

Our work draws on a wide array of related research ranging from musical descriptors and novel performance interfaces to recent applications in tabletop computing. By synthesizing these related but disparate areas of research, we enable new performance experiences for individual and group DJs to create and modify set-lists in real time.

Genre classification can be accomplished using a range of signal features and algorithms. For electronica in particular, features and patterns such as rhythm, tempo, periodicity, and even use of panning have been explored in the literature [1-3].

For DJs specifically, the use of interfaces to retrieve musically relevant material in performance has included query-by-beat-boxing [4], and query-by-humming [5]. Retrieval using both traditional and non-traditional instruments and interfaces has been explored by [6]. Other research in the academic arena for enabling DJ performance includes AudioPad [7], and Mixxx [8]. Although we take influence in these interfaces for retrieval, our work wishes to explore a browsing paradigm using similar creative interfaces.

In the commercial sector, Stanton's Final Scratch[1] enables DJs to use a physical controller to manipulate and mix digital music, while Native Instruments' Traktor[2] is a software-only solution for DJ performance. Ableton's flagship software, Live[3], has been increasingly used to enable DJs to use their own pre-composed music in live performance through the synchronized playback of different audio loops, known as clips.

A multitude of literature on tabletop computing & interfaces exists. The Reactable team was one of the first groups to directly apply both tangible and multi-touch interaction to the performance of music [9], followed by others including the earlier referenced *AudioPad,* which is also a tangible interface. More recently*, MarGrid,* a UI for the browsing of a digital music collection using Self-Organizing Maps has been examined using a tabletop interface [10]. The use of Self Organizing Maps (SOM) for visualizing feature data has also been previously covered by [11], [12]. In addition, although not performance-oriented, *MusicSim* presents an interesting combination of audio analysis and music browsing in an interactive computer-based interface [13].

Our aim here is to expand on these efforts by introducing the use of a multi-touch surface in a way that is both intuitive and collaborative. The use of Self Organizing

[1] http://www.stantondj.com/
[2] http://www.native-instruments.com/
[3] http://www.ableton.com/

Maps represents a useful way of organizing features for visualization, on-top of which many real-time interactive applications are possible.

3. DATA COLLECTION

For our experiments, six genres across the spectrum of electronic music were selected for their diverse characteristics and wide-spread popularity.

One hundred 2 to 8 minute prototypical tracks were sliced at random into single 30-second chunks for each genre. Our dataset contains at least 20 distinct artists in each genre; tracks were not chosen on the perceived genre of the composing artist, but a human baseline analysis by the authors. In total, there are 600 30-second clips, each in a stereo 44.1 kHz PCM-encoded file format. All files were normalized before experimentation.

3.1 Genre Definitions

Many subgenres fall beneath the umbrella term of electronica—this paper examines six of the most broad & popular genres commonly played by DJs: intelligent dance music (IDM), house, techno drum and bass (DnB), trance, and downtempo. A brief description of them is as follows:

IDM distinguishes itself by its heavy use of complex meter, sophisticated and often sporadic percussive elements, and varying use of syncopation. IDM carries with it a rich harmonic and melodic palate borrowed from many genres. Tempos typically range from 150-180 BPM. Notable artists in the genre are Aphex Twin, Squarepusher, and Autechre. IDM may sometimes be referred to as Glitch music.

House music makes use of the common 'four-on-the-floor' rhythm pattern consisting of a steady kick drum on each downbeat in a 4/4 meter. Defining characteristics involve offbeat open hi-hat patterns and snare or claps on the two and four of every bar. Harmonic content and instrumentation is often borrowed from Disco genres. Tempos usually range from 115 to 135 BPM. Daft Punk, Thomas Bangalter, and Alan Braxe are popular artists in the genre.

Techno uses minimal melodic ornamentation, relying more on bass riffs and polyrhythmic drums layered over a common four-on-the-flour kick drum. The rhythmic elements in techno are often the defining features of the song, with percussive grooves and riffs taking precedence over more traditional melodic and harmonic structure. Significant artists include Derrick May, Richie Hawtin, and Robert Hood.

DnB makes heavy use of "break beat chopping,"—the re-sequencing of drum hits from other previously recorded material. DnB is often composed above 160 BPM, with characteristic bass lines moving at half the tempo. Goldie and Pendulum are both well-known artists.

Trance distinguishes itself by employing thick, complex harmonic components, leaving little room for the complex rhythmic structures found in other similar genres. Trance often makes use of arpeggios, drum rolls, and long crescendos of synthesizers. The genre is composed around 140 BPM. DJ Tiesto, Ferry Corsten, and Sasha are popular artists within the Trance genre.

Downtempo employs lush harmonic textures and groove-oriented percussion. Tempos are characteristically low, ranging from 60 to 90 BPM. Boards of Canada, Air, and Bonobo are well-known artists within the genre.

4. AUDIO ANALYSIS AND CLASSIFICATION

Audio analysis was performed using the ChucK audio programming language [14]. Our results are based on a two-second (88200 sample) Hann window, resulting in 15 8-dimensional vectors for each audio clip. In addition to being written to disk for further analysis, the raw data was also sent over networked protocol (OSC[1]) into Processing[2], a visuals-oriented programming language. The process of visualizing the data using Processing is later presented in Section 5. Before application development could begin, a central concern was to uncover a feature-set that could accurately classify electronica. We follow with a description of the eight features used in our experiments.

4.1 Spectral Features

- Centroid, the centre of mass of the spectrum;
- Flux, the change in spectral energy across successive frames;
- Rolloff, the frequency below which resides 85 percent of a spectrum's energy.

4.2 Loudness Features

- RMS, the amplitude of a window;
- Panning, a coefficient used to describe the weight of the signal in either the left or right channels [3];
- Panning Delta, change in the panning coefficient across successive windows [3].

4.3 Temporal Features

- Number of Bass Onsets, an integer representing the number of peaks ('Beats') detected in a window;
- Average Inter-onset Time, a basic feature to describe the periodicity of the beats across a window.

4.4 Classification

Four separate classifiers were run on all six classes, and also on a smaller set of four classes. All experiments were performed utilizing a 10-fold cross-validation method in the Weka machine learning environment [15].

A k-Nearest Neighbour classifier (IBk) gave the best overall result, resting at a 75.2% classification rate across the six classes (16.7% baseline accuracy). Table 1 shows the confusion matrix for this experiment.

[1] http://opensoundcontrol.org/
[2] http://www.processing.org/

As Table 1 illustrates, the k-NN classifier had trouble distinguishing between IDM & DnB, and House & Techno. This is most likely attributed the sporadic percussive elements found in IDM & DnB, and very similar tempos found in House & Techno. Another experiment was run omitting IDM and House, resulting in a superior 87.0% classification rate. In the context of real-time performance and playlist generation, the omission was the result of IDM and House being considerable similar to genres already being classified. In favor of omitting any single pair of the confused genres, one of each was left out. The confusion matrix of this experiment is shown in Table 2.

Other classifiers used in testing were a C4 Decision-Tree (J48), a backpropagation Artificial Neural Network (MultiLayerPerceptron), and a Support Vector Machine (SMO). More details on these classifiers can be found in [15, 16]. Table 3 lists the accuracy of the four different classifiers using both the six-class and four-class datasets.

The exclusion of the two panning features and average inter-onset for the 6 and 4-class datasets using k-NN reduced classification accuracy by 7.70% and 4.85% respectively, indicating that both temporal and panning features moderately improved classification. Given the distinct tempos and production values between electronica genres, higher-level features using both tempo and panning should be considered an important facet of future classification experiments.

	Idm	**Tno**	**Dnb**	**Hse**	**Trn**	**Dtm**
Idm	**0.60**	0.02	0.16	0.07	0.02	0.12
Tno	0.02	**0.84**	0.03	0.07	0.04	0.01
Dnb	0.10	0.03	**0.72**	0.05	0.05	0.04
Hse	0.04	0.05	0.06	**0.78**	0.04	0.03
Trn	0.01	0.04	0.05	0.05	**0.82**	0.02
Dtm	0.13	0.01	0.06	0.05	0.02	**0.74**

Table 1 Confusion matrix, in percent, for the 6-class k-NN classifier

	Techno	**Dnb**	**Trance**	**Dtempo**
Techno	**0.90**	0.05	0.05	0.01
Dnb	0.04	**0.84**	0.06	0.06
Trance	0.05	0.06	**0.87**	0.02
Dtempo	0.01	0.10	0.02	**0.87**

Table 2 Confusion matrix, in percent, for the 4-class k-NN classifier

	IBk	**J48**	**MLPercept.**	**SMO**
6 Class	0.75	0.66	0.60	0.58
4 Class	0.87	0.82	0.81	0.79

Table 3 Accuracy, in percent, among the four classifiers

5. APPLICATIONS

A large portion of our work consisted of prototyping and testing potentially useful tools for the DJ. By sorting our dataset through the use of Self Organizing Maps, DJs will be able to generate groupings of musical material that immediately work well together. This data organization will provide not only obvious song clusters, but also interesting musical associations that may otherwise be overlooked.

5.1 Bricktable

A multi-touch surface called Bricktable [17] was chosen as the interface for visualizing and interacting with the SOMs. Multi-touch screens add a certain physicality to the data for the user, additionally supplying a modular software platform on which to expand the performance capabilities of these tools, especially between multiple potential users.

5.2 Self Organizing Maps

Our first application visualized the data in Processing using a SOM. The ability to effectively reduce dimensionality using a standard k-NN algorithm and the ease of visualization made a SOM an appealing choice to display the data as well as create a basic platform for playlist generation. The use of SOMs for playlist generation has been previously researched extensively by M. Dittenbach *et. al.* using their PlaySOM system [18].

Individual songs consist of 15 8-dimensional feature vectors. During our feature extraction stage, ChucK sends the features over Open Sound Control into Processing along with file name and path. The features are then ordered in a hierarchal manner, and superimposed over an RGB vector. The color vectors are then used to visualize unique songs on a 2D map. Once the map is populated and sorted, users can access individual songs by touching a coloured circle. This will recall the filename and begin playing the song, allowing users to quickly compare neighbouring music.

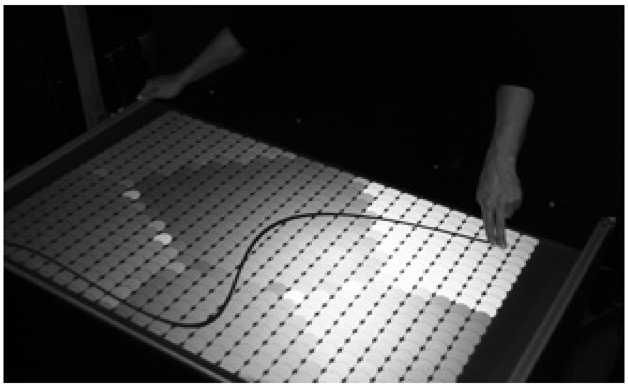


Figure 1 The SOM being displayed on the Bricktable, with the *DJen* interface minimized

5.3 The *DJen* Performance Application

Although the use of multi-touch for selecting songs directly on the SOM provides an engaging way for browsing a music collection, the application can be pushed further within the multi-touch paradigm. With this in mind, we present the *DJen* ('D-Gen') application, a tool to facilitate automatic set-list generation by enabling effective navigation of large libraries of music.

A critical skill among successful DJs is the ability to navigate seamlessly between many different songs, sometimes from varying genres. The key to this task is having the songs share a relationship in some way, usually through tempo. Via our SOM visualization, DJs already have access to musical groupings based off similarities, even if the genre is misclassified; however, *DJen* allows DJs to gesture a path through this map creating a dynamic playlist that can be used as source material for a performance. Due to the similarities between neighbours on the map, any arbitrary path will automatically generate a list of songs that share a strong relationship. Additionally, multiple DJs can create paths simultaneously, and *DJen* can interpolate a single path equidistant from all other paths. This will create a set list that represents the mean vectors between the original *DJen* paths. Finally, paths may be modified in real time for fine-tuning. Through this process we hope to enable the grouping of material in ways that a DJ may find inspiring. This path-based system is reminiscent of research conducted by R. Gulik and F. Vignoli in [19].

Figure 2 demonstrates the DJen GUI with the two primary UI elements shown: the playlist editor, and the 'now-playing' bar. Without a path set, a DJ can drag individual circles into the playlist editor to create a set list. When a path is drawn, the editor is automatically populated. If working collaboratively, another DJ may reshape the path and the playlist editor will automatically regenerate a set list.

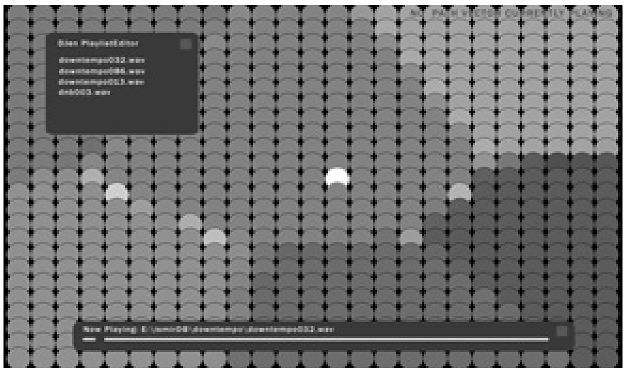


Figure 2 The SOM with the *DJen* GUI.

6. CONCLUSIONS & FUTURE WORK

Through the use of MIR tools we have shown strong potential for categorizing Electronica by genre. The k-NN algorithm provided an effective way of processing the sample data, making the *DJen* application possible through the use of a SOM. Finally, coupling this work with a multi-touch surface opened new avenues for DJs to interact with their music.

The *DJen* application represents a motivating step toward 'intelligent', MIR-powered tools for DJs. Its strength is revealed through the interactive user interface and visualization techniques. In the future, more rhythmic features to categorize electronica may be explored to create a system whereby *DJen* can perform automatic transitions between songs. This will allow the DJ to concentrate on other expressive areas such as sampling, looping, and effects processing.

The continuous growth of multi-touch necessitates development of further applications to explore both single and multi-user performance paradigms. Although *DJen* may be used by one or more users, extensive collaboration options may be enabled by allowing one DJ to oversee transitioning, while another manages multiple set-lists stemming from the original path chosen across the map.

Looking into the future, we hope *DJen* and other MIR-powered applications of its type will in the future enable any DJ to create expressive performances for their audiences.

7. ACKNOWLEDGEMENTS

We would like to thank Nick Diakopoulos for his helpful ideas and revisions. We would also like to thank George Tzanetakis for his inspiration and training.

8. REFERENCES

1. Gouyon, F. and S. Dixon. *Dance Music Classification: a tempo-based approach*. in *Proceedings of the 5th International Conference on Music Information Retrieval*. 2004. Barcelona, Spain.
2. Gouyon, F., et al. *Evaluating rhythmic descriptors for musical genre classification*. in *Proceedings of the 25th International AES Conference*. 2004. London, UK.
3. Tzanetakis, G., R. Jones, and K. McNally. *Stereo Panning Features for Classifying Recording Production Style*. in *Proceedings of the 8th International Conference on Music Information Retrieval*. 2007. Vienna, Austria.
4. Kapur, A., R. McWalter, and G. Tzanetakis. *Query-by-Beat-Boxing: Music Retrieval For The DJ*. in *Proceedings of the 5th International Conference on Music Information Retrieval*. 2004. Barcelona, Spain.
5. Birmingham, W., R. Dannenberg, and B. Pardo, *Query by humming with the VocalSearch system*. Commun. ACM, 2006. **49**(8): p. 49-52.
6. Kapur, A., R. McWalter, and G. Tzanetakis. *New Music Interfaces for Rhythm-Based Retrieval*. in *Proceedings of the 6th International Conference on Music Information Retrieval*. 2005. London, England.

7. Patten, J., B. Recht, and H. Ishii. *Interaction techniques for musical performance with table-top tangible interfaces*. in *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*. 2006. Hollywood, California.
8. Andersen, T.H. *Mixxx: Towards Novel DJ Interfaces*. in *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*. 2003. Montreal, Canada.
9. Jorda, S., et al. *The reacTable*. in *Proceedings of the International Computer Music Conference*. 2004. Barcelona, Spain.
10. Hichner, S., J. Murdoch, and G. Tzanetakis. *Music Browsing Using a Tabletop Display*. in *Proceedings of the 8th International Conference on Music Information Retrieval*. 2007. Vienna, Austria.
11. Pampalk, E., G. Widmer, and A. Chan, *A new approach to hierarchical clustering and structuring of data with Self-Organizing Maps.* Intell. Data Anal., 2004. **8**(2): p. 131-149.
12. Cooper, M., et al., *Visualization in Audio-Based Music Information Retrieval.* Comput. Music Journal, 2006. **30**(2): p. 42-62.
13. Chen, Y.-X. and A. Butz, *Musicsim: integrating audio analysis and user feedback in an interactive music browsing ui*, in *Proceedings of the 13th International Conference on Intelligent User Interfaces*. 2009: Florida, USA.
14. Fiebrink, R., G. Wang, and P. Cook. *Support For MIR Prototyping and Real-Time Applications in the ChucK Programming Language*. in *Proceedings of the 9th International Conference on Music Information Retrieval.* 2008. Philadelphia, USA.
15. Witten, I.H. and E. Frank, *Data Mining: Practical machine learning tools and techniques*. 2nd ed. 2005, San Francisco: Morgan Kaufmann.
16. Duda, R.O., P.E. Hart, and D.G. Stork, *Pattern Classification*. 2nd ed. 2000, New York: John Wiley & Sons.
17. Hochenbaum, J. and O. Vallis. *Bricktable: A Musical Tangible Multi-Touch Interface*. in *Proceedings of the Berlin Open*. 2009. Germany.
18. Dittenbach, M., R. Neumayer, and A. Rauber. *PlaySOM: An Alternative Approach to Track Selection and Playlist Generation in Large Music Collections*. in *Proc. 1st Intl. Workshop on Audio-Visual Content and Information Visualization in Digital Libraries*. 2005. Cortona, Italy.
19. Guelik, R.V. and F. Vignoli. *Visual Playlist Generation on the Artist Map*. in *Proceedings of the 6th International Conference on Music Information Retrieval*. 2005. London, England.

RELATIONSHIPS BETWEEN LYRICS AND MELODY IN POPULAR MUSIC

Eric Nichols[1], Dan Morris[2], Sumit Basu[2], and Christopher Raphael[1]

[1]Indiana University
Bloomington, IN, USA
{epnichol,craphael}@indiana.edu

[2]Microsoft Research
Redmond, WA, USA
{dan,sumitb}@microsoft.com

ABSTRACT

Composers of popular music weave lyrics, melody, and instrumentation together to create a consistent and compelling emotional scene. The relationships among these elements are critical to musical communication, and understanding the statistics behind these relationships can contribute to numerous problems in music information retrieval and creativity support. In this paper, we present the results of an observational study on a large symbolic database of popular music; our results identify several patterns in the relationship between lyrics and melody.

1. INTRODUCTION

Popular music uses several streams of information to create an emotionally engaging experience for the listener. Lyrics, melody, chords, dynamics, instrumentation, and other aspects of a song operate in tandem to produce a compelling musical percept. Extensive previous work has explored each of these elements in isolation, and certain relationships among these components – for example, the relationship between melody and chords – have also been addressed in the research community. However, despite their salience and central role in music cognition, lyrics have not been addressed by computational analysis to the same degree as other aspects of popular music.

In this study, we examine the relationship between lyrics and melody in popular music. Specifically, we investigate the assumption that songwriters tend to align low-level features of a song's text with musical features. Composer Stephen Sondheim, for example, has commented that he selects rhythms in music to match the natural inflections of speech [1], and popular books on songwriting suggest considering the natural rhythms of speech when writing melodies [2]. With this qualitative evidence in mind, we quantitatively examine relationships between text and music using a corpus of several hundred popular songs. Specifically, we investigate the general hypothesis that textual salience is correlated with musical salience, by extracting features representative of each and exploring correlations among those features.

This study contributes fundamental statistics to musicology and music-cognition research, and makes the following specific contributions to the music information retrieval community:

1) We establish new features in the hybrid space of lyrics and melody, which may contribute to musical information and genre analysis as well as music recommendation.
2) We demonstrate a quantitative correlation between lyrical and melodic features, motivating their use in composition-support tools which help composers work with music and text.
3) We strengthen the connection between MIR and speech research; the features presented here are closely related to natural patterns in speech rhythm and prosody.
4) We make analysis of lyrics and melody in popular music more accessible to the community, by releasing the parsing and preprocessing code developed for this work.

2. RELATED WORK

Previous work in the linguistics and speech communities has demonstrated that inherent rhythms are present even in non-musical speech (e.g. [3,4]). Additional work has shown that the rhythms inherent to a composer's native language can influence instrumental melodic composition. Patel and Daniele [5] show a significant influence of native language (either English or French) on composers' choice of rhythmic patterns, and Patel et al. [6] extend this work to show a similar influence of native language on the selection of pitch intervals. This work does not involve text per se, only the latent effect of language on instrumental classical music. Beyond the rhythmic aspects of speech, additional work has demonstrated that vowels have different intrinsic pitches [7], and even that phonemes present in musical lyrics can influence a listener's perception of pitch intervals [8]. This work supports our claim that there is a strong connection between not only rhythmic aspects of speech and music, but also between linguistic, phonemic, pitch, and timbral aspects of speech and music.

In addition to these explorations into the fundamental properties of speech and lyrics, preliminary applications of the statistics of lyrics have begun to emerge for both creativity support tools and problems in music information retrieval and analysis. Proposing a creativity support tool to explore alignments of melodies and lyrics, [9] uses a series of hand-coded heuristics to align a known set of lyrics to the rhythm of a known melody. Oliveira et al. [10] develop a preliminary system that addresses the problem of generating text to match a known rhythm; this works also includes a preliminary analysis of a small database to qualitatively validate the authors' assumptions.

Wang et al. [11] and Iskandar et al. [12] use higher-level properties of lyrical structure to improve the automatic alignment of recordings with corresponding lyrics. Lee and Cremer [13] take a similar approach to match high-level segments of lyrics to corresponding segments in a recording. Recent work in the music information retrieval community has also applied lyric analysis to problems in topic detection [14], music database browsing [15], genre classification [16], style identification [17], and emotion estimation [18]. This work motivates the present study and suggests the breadth of applications that will benefit from a deeper, quantitative understanding of the relationship between lyrics and melody.

3. METHODS

3.1 Data Sources and Preprocessing

Our database consisted of 679 popular music lead sheets in MusicXML format. 229 of our lead sheets came from a private collection; the remaining 450 came from Wikifonia.org, an online lead sheet repository. Our data spans a variety of popular genres, including pop, rock, R&B, country, Latin, and jazz, with a small sampling of folk music.

Each lead sheet in our database contains a melody, lyrics, and chords for a single song (chords were not used in the present analysis). Lyrics are bound to individual notes; i.e., no alignment step was necessary to assign lyrics to their corresponding notes. Word boundaries were provided in the MusicXML data so it was possible to determine which syllables were joined to make whole words without consulting a dictionary. Key and time signature information was also provided for each song (including any changes within a song). For all analyses presented in this paper, we ignored measures of music with a time signature other than 4/4. Lead sheets were processed to build a flat table of notes (pitch and duration) and their corresponding syllables, with repeats flattened (expanded and rewritten without repeats) to allow more straightforward analysis.

3.2 Computed Musical Features

This section describes the three features that were computed for each note in our melody data.

3.2.1 Metric Position

For each note, the "Metric Position" feature was assigned to one of five possible values based on the timing of the note's onset: **downbeat** (for notes beginning on beat 1), **half-beat** (for notes beginning on beat 3), **quarter beat** (beginning on beats 2 or 4), **eighth beat** (beginning on the "and" of any quarter beat), and **other**.

3.2.2 Melodic Peak

The "Melodic Peak" feature is set to **True** for any note with a higher pitch than the preceding and subsequent notes. It is set to **False** otherwise (including notes at the beginning and end of a song). We selected this feature because previous research has connected melodic contours to a number of features in instrumental music [19].

3.2.3 Relative Duration

For a note in song *s*, the "Relative Duration" feature is computed by calculating the mean duration (in beats) for all notes in *s* and then dividing each note's duration by the mean. Thus "Relative Duration" values greater than 1 indicate notes longer than mean duration for the associated song.

3.3 Computed Lyrical Features

This section describes the three features that were computed for each syllable in our lyric data, based on the syllable itself and/or the containing word.

We determined the pronunciation of each syllable by looking up the containing word in the CMU Pronouncing Dictionary [20], a public-domain, machine-readable English dictionary that provides phoneme and stress level information for each syllable in a word. In cases where the dictionary provided alternate pronunciations, we selected the first one with the correct number of syllables. Unknown words and words whose associated set of notes in our MusicXML data did not correspond in number to the number of syllables specified by the dictionary were removed from the data. Note that this dictionary provides pronunciation for isolated words. Stress patterns can change based on the surrounding context, so this pronunciation data is only an approximation of natural speech.

3.3.1 Syllable Stress

The CMU dictionary gives a stress level according to the following ordinal scale: **Unstressed**, **Secondary Stress**, and **Primary Stress**; each syllable was assigned one of these three values for the "Syllable Stress" feature. Secondary stress is typically assigned in words with more than two syllables, where one syllable receives some stress but is not the primary accent. For example, in the word "letterhead", the first syllable is assigned a primary stress, the second is unstressed, and the third is assigned a secondary stress.

3.3.2 Stopwords

Stopwords are very common words that carry little semantic information, such as "a", "the", and "of". Stopwords are generally ignored as "noise" in text processing

CMU Vowel	IPA (Pan-English)	Example
AH	ə	hut
UH	ʊ	hood
IH	ɪ	it
ER	ɜ	hurt
EH	ɛ	Ed
AE	æ	at
AA	ɑ:	odd
IY	i:	eat
UW	u:	two
AY	aɪ	hide
AO	ɔ:	ought
OW	oʊ	oat
EY	eɪ	ate
AW	aʊ	cow
OY	ɔɪ	toy

Table 1. Vowels used in our analysis (sorted by increasing average associated relative note duration – see section 4.4). In order to classify vowels as short, long, or diphthong, vowels from the CMU dictionary were translated to Pan-English IPA (International Phonetic Alphabet) symbols according to [23]. Symbols ending in a colon (:) represent long vowels; symbols containing two characters (e.g. oʊ) represent diphthongs. As is further elaborated in Section 4, we highlight that when sorted by average musical note duration, short vowels are correlated with shorter durations than long vowels and diphthongs in all cases, and with the exception of one long vowel (AO, or ɔ:), diphthongs are assigned longer durations than long vowels.

systems such as search engines. There is no definitive or absolutely correct list of English stopwords; we use the monosyllabic subset of the online adaption [21] of the fairly canonical stopword list originally presented by van Rijsbergen [22]. We specifically choose the monosyllabic subset so that we are conservative in our identification of stopwords; we consider words such as "never", while perhaps too common for certain applications, to be semantically rich enough to merit treatment as non-stopwords. The "Stopword" feature is set to **True** or **False** for each monosyllabic word, and is undefined for multisyllable words.

3.3.3 Vowels

Each syllable in the dictionary may include multiple consonants, but only one vowel. We extract the vowel for each syllable; this categorical feature can take on one of 15 possible values, enumerated in Table 1.

4. RESULTS

Having established a set of features in both the melodic and lyrical spaces, we now turn our attention to exploring correlations among those features.

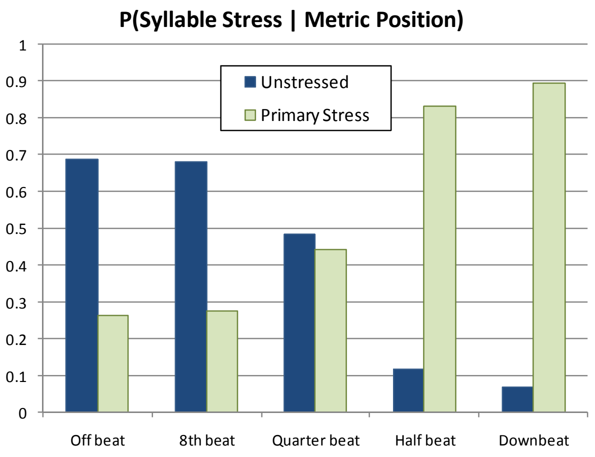


Figure 1. *P(syllable stress | metric position).* The stronger a note's metric position, the more likely it is that the associated syllable has a primary stress. Secondary stresses are rare overall and were omitted from this graph.

4.1 Syllable Stress

Based on our general hypothesis that musical salience is frequently associated with lyrical salience, we hypothesized that stressed syllables would tend to be associated with musically accented notes. We thus explored correlations between the "Syllable Stress" feature and each of our melodic features. Each analysis in this subsection was performed only using note data associated with polysyllabic words, so that stress values are meaningful.

4.1.1 Syllable Stress and Metric Position

A stronger syllable stress is associated with a stronger metric position, as we see in Figures 1 and 2. These give two different views of the data, based on conditioning first by either *metric position* or *syllable stress*.

Figure 1 demonstrates that the half beat and downbeat positions strongly favor stressed syllables, and are rarely associated with unstressed syllables. For comparison, stressed and unstressed syllables occur with approximately equal a priori probabilities (P(primary stress) = 0.46

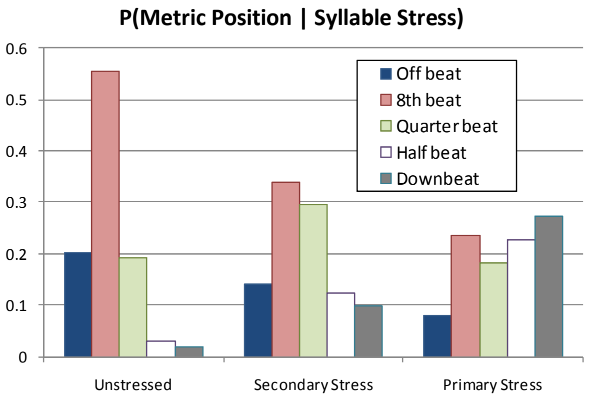


Figure 2. *P(metric position | syllable stress).* Unstressed syllables are very unlikely to show up on a downbeat, but very likely at an 8th beat position. Primary stresses rarely occur on off-beats.

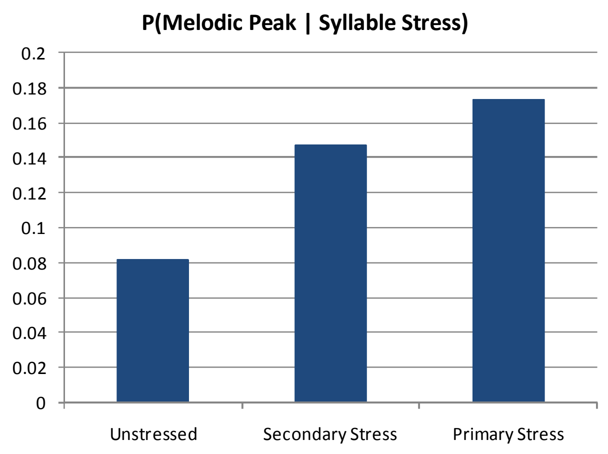


Figure 3. *P(melodic peak | syllable stress)*. The probability of a melodic peak increases with increasing syllable stress.

and P(unstressed) = 0.48). Figure 2 similarly shows that unstressed syllables are very unlikely to show up on a downbeat, but very likely at an 8th-beat position, and that primary stresses rarely occur on off-beats. Pearson's Chi-Square test confirms a significant relationship between these features ($p < 0.0001$).

4.1.2 Syllable Stress and Melodic Peaks

Figure 3 shows that stronger syllable stress is also strongly associated with the occurrence of melodic peaks. This relationship holds in both directions: the probability of a primary stress is significantly higher at syllables corresponding to melodic peaks than at non-peaks, and the probability of a melodic peak is much higher at stressed syllables than non-stressed syllables. Pearson's Chi-Square test confirms a significant relationship between these features (p < 0.0001).

4.1.3 Syllable Stress and Note Duration

In Figure 4, the "Relative Duration" feature has been discretized into two values: "Short" (Relative Duration ≤ 1, i.e. notes shorter than the mean duration within a song), and "Long" (Relative Duration > 1). Figure 4 shows that long notes are more likely to associated with stressed syllables than unstressed syllables, and short notes are more likely to be associated with unstressed syllables. The inverse relationship is true as well; most notes (55%) associated with unstressed syllables are short, and most notes (55%) associated with primary-stress syllables are long. Pearson's Chi-Square test confirms a significant relationship between these features (p < 0.0001).

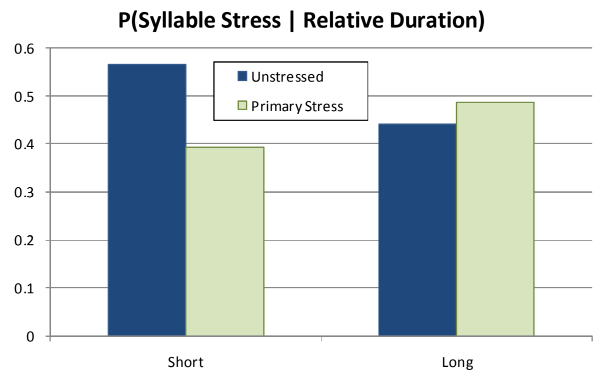


Figure 4. *P(syllable stress | relative duration)*. The "Relative Duration" feature was discretized into two values: "Short" (Relative Duration ≤ 1, i.e. notes shorter than the mean duration within a song), and "Long" (Relative Duration > 1). Shorter note durations are more likely to be associated with unstressed syllables; longer durations are more likely to be associated with stressed syllables.

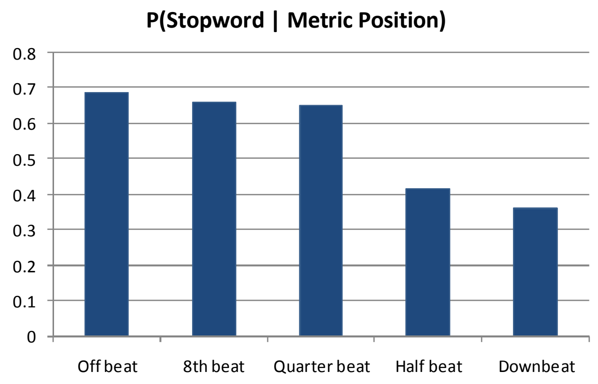


Figure 5. *P(stopword | metric position)*. This graph shows metric positions moving from weak (left) to strong (right), and the corresponding decrease in the probability of stopwords at corresponding syllables.

4.2 Stopwords

Based on our general hypothesis that musical salience is frequently associated with lyrical salience, we hypothesized that semantically meaningful words would tend to be associated with musically salient notes, and consequently that stopwords – which carry little semantic information – would be associated with musically non-salient notes. In this subsection, only notes associated with monosyllabic words are used in the analysis, since our list of stopwords includes only monosyllabic words.

4.2.1 Stopwords and Metric Position

Figure 5 shows the probability of finding a stopword at each metric position. The stronger the metric position, the less likely the corresponding word is to be a stopword. The overall probability of a stopword (across all metric positions) is 0.59. However, the half-beat and downbeat positions favor non-stopwords. Pearson's Chi-Square test confirms a significant relationship between these features (p < 0.0001).

4.2.2 Stopwords and Melodic Peaks

Figure 6 shows that melodic peaks are more frequently associated with non-stopwords than with stopwords. The inverse relationship holds as well: the probability of observing a stopword at a melodic peak is lower than at a non-peak. Pearson's Chi-Square test confirms a significant relationship between these features (p < 0.0001).

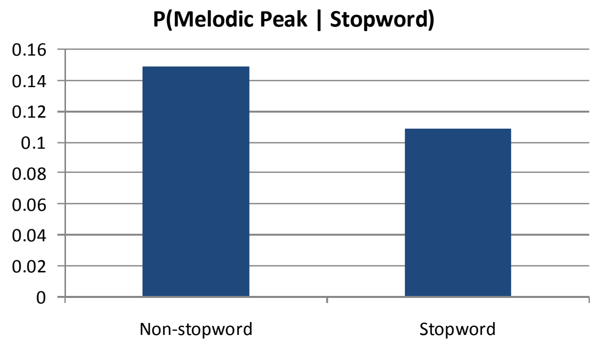


Figure 6. *P(melodic peak | stopword).* Melodic peaks are significantly more likely to coincide with non-stopwords than with stopwords.

4.3 Vowels

We hypothesized that vowel sounds would vary reliably with note durations, reflecting both the aesthetic properties of different vowel types and the impact of different vowel types on a singer's performance. We thus looked at correlations between the "phonetic length" of vowels (short, long, or diphthong) and the average durations of corresponding notes. We assign phonetic length to vowel length according to the IPA convention for Pan-English interpretation of phonemes (Table 1).

4.3.1 Vowels and Relative Duration

Figure 7 is a sorted plot of mean relative duration of notes for each vowel type. In general agreement with our hypothesis, the shorter vowels all have mean relative duration less than 1 (i.e. short vowels have shorter duration than average in a song); long vowels and diphthongs have mean relative duration greater than 1 (i.e. long vowels have longer duration than average). We highlight that short vowels are correlated with shorter durations than long vowels and diphthongs in all cases, and with the exception of one long vowel (AO, or ɔ:), diphthongs are assigned longer durations than long vowels.

If we generate a Boolean feature indicating whether a vowel is long (including diphthongs) or short, and we similarly use the Boolean version of the "Relative Duration" feature (see Figure 5), we can proceed as in previous sections and correlate vowel length with relative duration. Figure 8 shows that longer notes are more likely to be associated with long vowels, and short notes with short vowels. Pearson's Chi-Square test confirms the significance of this relationship ($p < 0.0001$).

5. DISCUSSION

5.1 Summary of Findings

We have introduced an approach for analyzing relationships between lyrics and melody in popular music. Here we summarize the relationships presented in Section 4:

1) Level of syllabic stress is strongly correlated with strength of metric position.

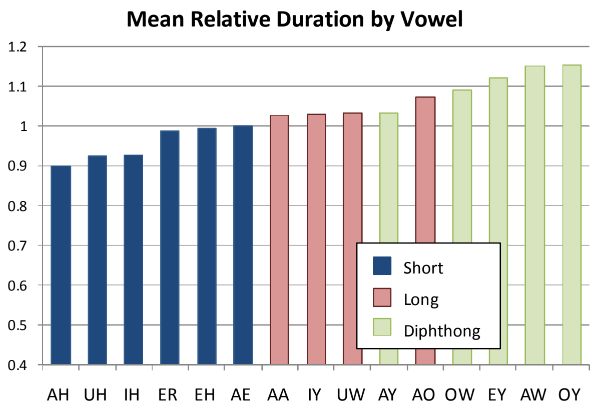


Figure 7. Mean relative duration of notes associated with each vowel, sorted form short notes (left) to long (right). The resulting partitioning of similar vowel types shows that short vowels are correlated with shorter durations than long vowels and diphthongs in all cases, and with the exception of one long vowel (AO), diphthongs are correlated with longer durations than long vowels.

2) Level of syllabic stress is strongly correlated with the probability of melodic peaks.
3) Level of syllabic stress is strongly correlated with note duration.
4) Stopwords (which carry little semantic weight) are strongly correlated with weak metric positions.
5) Stopwords are much less likely to coincide with melodic peaks than non-stopwords.
6) Short vowels tend to be associated with shorter notes than long vowels, which tend to be associated with shorter notes than diphthongs.

These findings support our highest-level hypothesis: songwriters tend to align salient notes with salient lyrics. The strength of these relationships – and our ability to find them using intuitive features in both lyrics and melody – suggests the short-term potential to apply these relationships to both MIR and creativity support tools.

5.2 Applications and Future Work

The analysis presented here used features that were easily accessible in our database of symbolic popular music. Future work will explore similar relationships among more

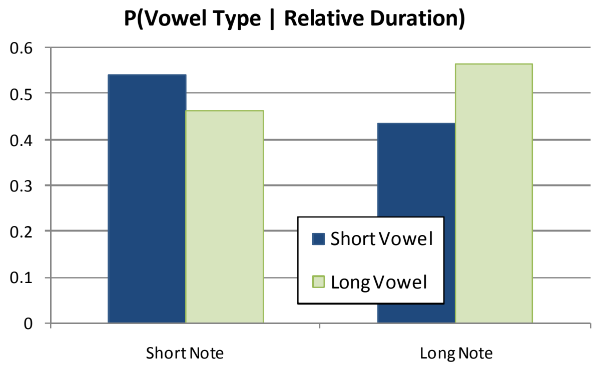


Figure 8. *P(vowel type | relative duration).* Short notes are more frequently associated with short vowels, and long notes with long vowels.

complex features of both lyrics (e.g. valence, parts of speech) and music (e.g. tone and timbre, dynamics, and pronunciation data extracted from vocal performances).

Understanding the statistics of lyrics alone will contribute to the many of the same applications that will benefit from our understanding of the relationship between lyrics and music. Therefore, future work will also include a large-scale study that more deeply explores the statistics and grammatical patterns inherent to popular lyrics, as compared to non-musical text corpora.

Most importantly, future work will explore applications of a quantitative understanding of the relationship between lyrics and melody. For example, these relationships can provide priors for lyric transcription and lyric alignment to audio recordings. Similarly, strengthening the connection between music and lyrics will allow us to more easily borrow techniques from the speech community for problems such as artist identification and score-following for popular music.

Furthermore, a quantitative understanding of the relationship between lyrics and melody has applications in tools that support the creative process. Composers and novices alike may benefit from systems that can suggest lyrics to match a given melody or vice versa, and understanding the relationships presented in this paper is an important first step in this direction. One might similarly imagine a "grammar checker" for popular composition, which provides suggestions or identifies anomalies not in text, but in the relationship between melody and lyrics.

6. PREPROCESSING TOOLKIT

In order to stimulate research in this area and allow replication of our experiments, we provide the preprocessing components of our analysis toolkit to the community at:

http://www.music.informatics.indiana.edu/code/musicxml

The archive posted at this location does not include our database (for copyright reasons), but we provide instructions for downloading the Wikifonia data set.

7. ACKNOWLEDGEMENTS

Data sets were provided by Wikifonia and Scott Switzer.

8. REFERENCES

[1] M. Secrest: *Stephen Sondheim, A Life*. New York, Alfred A. Knopf, 1998.

[2] J. Peterik, D. Austin, and M. Bickford: *Songwriting for Dummies*. Hoboken, Wiley, 2002.

[3] F. Cummins: Speech Rhythm and Rhythmic Taxonomy. *Proc Speech Prosody*, April 2002.

[4] M. Brady, R. Port. Quantifying Vowel Onset Periodicity in Japanese. *Proc 16th Intl Congress of Phonetic Sciences*, Aug 2007.

[5] A.D. Patel and J.R. Daniele: An empirical comparison of rhythm in language and music. *Cognition*, v87, p35-45, 2002.

[6] A.D. Patel, J.R. Iversen, and J.C. Rosenberg: Comparing the rhythm and melody of speech and music: The case of British English and French. *J. Acoustic Soc. Am*, 119(5), May 2006.

[7] S. Sapir: The intrinsic pitch of vowels: Theoretical, physiological and clinical considerations. *Journal of Voice* (3) 44-51, 1998.

[8] F. Russo, D. Vuvan, and W. Thompson: Setting words to music: Effects of phoneme on the experience of interval size. *Proc 9th Intl Conf on Music Perception and Cognition (ICMPC)*, 2006.

[9] E. Nichols: Lyric-Based Rhythm Suggestion. To appear in *Proc Intl Comp Music Conf (ICMC) 2009.*

[10] H. Oliveira, A. Cardoso, F.C. Pereira: Tra-la-Lyrics: An approach to generate text based on rhythm. *4th Intl Joint Workshop on Comp Creativity*, 2007.

[11] Y. Wang, M.-Y. Kan, T.L. Nwe, A. Shenoy, and J. Yin: LyricAlly: Automatic Synchronization of Acoustic Musical Signals and Textual Lyrics. *Proc ACM Multimedia*, Oct 2004.

[12] D. Iskandar, Y. Wang, M.-Y. Kan, H. Li: Syllabic Level Automatic Synchronization of Music Signals and Text Lyrics. *Proc ACM Multimedia*, Oct 2006.

[13] K. Lee and M. Cremer: Segmentation-Based Lyrics-Audio Alignment Using Dynamic Programming. *Proc ISMIR 2008.*

[14] F. Kleedorfer, P. Knees, and T. Pohle: Oh Oh Oh Whoah! Towards Automatic Topic Detection in Song Lyrics. *Proc ISMIR 2008.*

[15] H. Fujihara, M. Goto, and J. Ogata: Hyperlinking Lyrics: A Method for Creating Hyperlinks Between Phrases in Song Lyrics. *Proc ISMIR 2008.*

[16] R. Mayer, R. Neumayer, and A. Rauber: Rhyme and Style Features for Musical Genre Classification by Song Lyrics. *Proc ISMIR 2008.*

[17] T. Li and M. Ogihara: Music artist style identification by semi-supervised learning from both lyrics and content. *Proc ACM Multimedia*, Oct 2004.

[18] D. Wu., J.-S. Chang, C.-Y. Chi, C.-D. Chiu, R. Tsai, and J. Hsu: Music and Lyrics: Can Lyrics Improve Emotion Estimation for Music? *Proc ISMIR 2008.*

[19] Z. Eitan: *Highpoints: A Study of Melodic Peaks.* Philadelphia, Univ of Pennsylvania Press, 1997.

[20] http://speech.cs.cmu.edu/cgi-bin/cmudict Downloaded on May 20, 2009.

[21] http://dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words. Retrieved on May 15, 2009.

[22] C.J. van Rijsbergen: *Information Retrieval* (2nd edition). London, Butterworths, 1979.

[23] http://en.wikipedia.org/wiki/IPA_chart_for_English_dialects. Retrieved on May 15, 2009.

RHYTHMIXEARCH: SEARCHING FOR UNKNOWN MUSIC BY MIXING KNOWN MUSIC

Makoto P. Kato
Department of Social Informatics, Graduate School of Informatics
Kyoto University, Kyoto, Japan
kato@dl.kuis.kyoto-u.ac.jp

ABSTRACT

We present a novel method for searching for unknown music. *RhythMiXearch* is a music search system we developed that can accept two music inputs and mix those inputs to search for music that could reasonably be a result of the mixture. This approach expands the ability of *Query-by-Example* and allows greater flexibility for users in finding unknown music. Each music piece stored by our system is characterized by text data written by users, i.e., review data. We used *Latent Dirichlet Allocation* (LDA) to capture semantics from the reviews that were then used to characterize the music by *Hevner's* eight impression categories. *RhythMiXearch* mixes two music inputs in accordance with a probabilistic mixture model and finds music that is the most likely product of the mixture. Our experimental results indicate that the proposed method is comparable to human in searching for music by multiple examples.

1. INTRODUCTION

Much music content has become available, and music analysis and retrieval systems have recently been rapidly developing. To make finding music easy, many prototype systems for searching for music pieces by using content-based IR techniques have been proposed [17] [6]. They enable users to find music by inputting an audio file as a query, called *Query-by-Example* (QBE), in particular inputting by humming, i.e., *Query-by-Humming* (QBH) [4]. Based on the input audio signals, QBE systems retrieve music by calculating the similarity between the queried music piece and stored music and then return the results in the order of similarity to the query. Searching by example is helpful for obtaining new music similar to music that you have or that you have heard.

However, these content-based IR methods are not able to meet the specific needs of users wanting to find music they have never heard. A common situation is that you want to find a certain piece of music which you imagine in your mind, but have neither the keywords related to it, music similar to it, nor the ability to sing it. In addition, content-based approaches rank at the top only music similar to what you know well, so you cannot find music very different from yours; the opportunity to discover new music is lost. This is caused by the lack of flexibility in inputting queries. As the amount of digital music content increases, finding the precise music you want requires higher expressiveness of queries.

We present a novel approach to searching for unknown music. *RhythMiXearch* is a music search system we developed that can accept two or more music inputs. By mixing the input music, it searches for music that could reasonably be a result of the mixture. This approach expands the ability of *Query-by-Example* and allows greater flexibility for users in finding unknown music. For example, intuitively, *RhythMiXearch* can introduce music similar to *The Beatles' Let It Be + Coldplay's Viva la Vida* to you.

Stored music in *RhythMiXearch* is characterized on the basis of users' impressions. We retrieved review data on *Amazon.com*, analyzed the text data by using *Latent Dirichlet Allocation* (LDA) [2], and determined the impressions that users received from the music.

There is a strong reason that users' impressions were used as a feature of the music and were extracted from the review data rather than the features of the music itself being used. Consider music that users do not know but want to find. The mood or impression the music will give users is more important than the timbre [1] or rhythm [5] [12] it has. Users are likely not able to imagine the details of the wanted music, such as the timbre and rhythm; they only feel the sense of the music they want, such as the mood and impression. In addition, in our approach to mixing input music, picturing mixtures of timbre and rhythm would be difficult, and for users, the result may not be what is expected or wanted. For detecting the impression given by music pieces, seeing review text written by humans about the music would be more effective than analyzing the music itself. Mood detection by signal analysis has been proposed [15] [14]. However, the final feeling we get from listening to music is a product of knowing the title and artist, listening to the melody, understanding the lyrics, and so on; simply analyzing the timbre and rhythm of a piece is not enough for estimating what listeners will feel. In contrast, reviews are provided by music listeners, so analyzing review text rather than the music itself would be helpful for determining the impression given by the music.

Input music is combined based on the features of the music represented by the estimated impression, and our system ranks its stored music pieces by their likelihood of reasonably being a result of the combined input music. Situations in which multiple examples could be used include the following: searching for music that has all the features of multiple music inputs, and searching with multiple inputs of your favorite music. For these situations, we developed a method to combine two music inputs in one query. We named the multiple input query *Query-by-Mixture-of-Examples*.

2. RELATED WORK

Characterizing music by using text data has been reported recently. Knees et al. used Web documents to develop a system that searches for music pieces through natural language queries [11] and also presented a method to combine signal-centered features with document-centered ones [9]. They characterized music pieces by using a conventional IR approach, which is the *Vector Space Model* with tf-idf method. In addition to searching for music, artist classification [10] was done by the same text-based approach with the SVM. Pohle et al. [13] describe artists by common topics or aspects extracted from Web documents. A browser application they presented enables users to formulate a query to search for desired artists by simply adjusting slider positions.

Turnbull et al. [16] focused on natural language queries such as "*female lead vocals*" , called *Query-by-Semantic-Description* (QBSD). In their approach, the *Computer Audition Lab 500-Song* (CAL500) data set was used to learn a word-level distribution over an audio feature space. QBSD can search for music pieces unfamiliar to users, which is the same aim as ours. Terms used as queries to illustrate music, however, are limited with regard to amount and cannot capture subtle nuances to search for wanted music.

Music Mosaics [18] is a concept for creating a new query by concatenating short segments of other music pieces. It applies the signal analysis technique to characterize music and represents pieces of the music by thumbnails. Querying with multiple music pieces in music mosaics is quite similar to our method, but as mentioned above, making a query by assembling pieces of signal information to find unfamiliar music is difficult.

Similar to our approach, MusicSense [3] is a music recommendation system for users reading Web documents such as Weblogs. It adopted a generative model called *Emotional Allocation Modeling* to detect emotions of documents and music with the text. In this model, a collection of terms is considered as generated over a mixture of emotions, like the LDA approach.

3. METHODOLOGY

At first, we propose a framework of our approach. Then, we explain song characterization with reviews by using LDA, probabilistic mixture model for combining input music pieces, and ranking music pieces by the similarity.

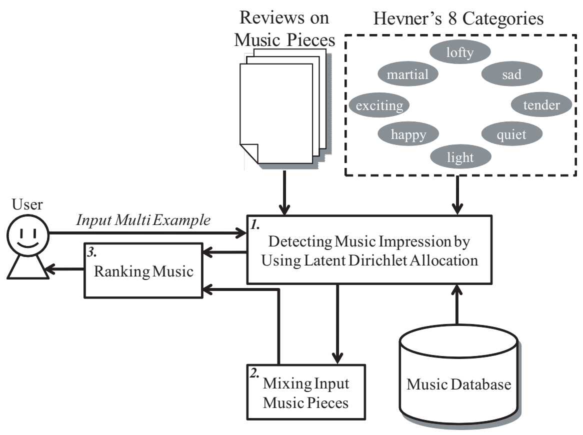


Figure 1. Framework of our approach.

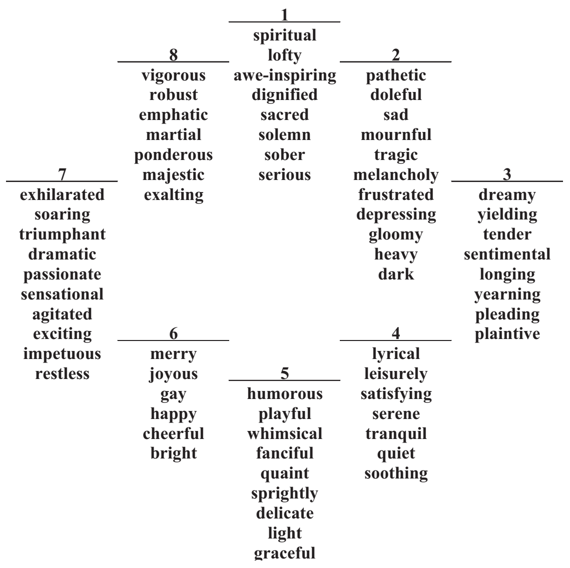


Figure 2. 8 sets of impression words proposed by Hevner. Adjacent sets are similar impressions, and opposite ones are counter-impressions.

3.1 Framework of Our Approach

The framework of our approach is shown in Fig. 1. It consists of three steps: (1) detecting impressions of music pieces by using LDA from music reviews, (2) mixing input music pieces on the basis of the impressions, and (3) ranking stored music pieces by their likelihood of being the result of the mixture.

For extracting impressions from music reviews, we used a generative model named LDA, in which it is assumed that terms in a document are generated by a mixture of topics, i.e., multinomial distributions over topics. The assumption enables us to conjecture the fundamental meanings of documents, and the meanings are represented by the topic distribution for each document.

The sets of impression words for music proposed by Hevner [8] are shown in Fig. 2. The impression words are used to find which impression a review gives in the generative model , in intuitive terms, by calculating the similarity between reviews and the impression words, where we regard the sets of impression words as documents. We obtain the probability that each distribution over topics for a document would generate a set of impression words if only Hevner's sets of impression words were provided.

Given multiple music inputs, we mix on the basis of the impression probability. Different mixture models are proposed for different situations. Finally, the results from the stored music pieces are returned, ranked by the similarity to the mixture of multiple examples. One easy method is the similarity-based ranking between stored music and the virtual music created as a result of a mixture. We apply this method to our system and introduce a prototype system based on the framework.

3.2 Characterizing Songs by Reviews

First, we introduce a method to characterize songs by analyzing text review data with LDA. In the LDA analysis, terms in a document are assumed to be generated from a topic and topics allocated to words are chosen from multinomial distributions for the documents. Each multinomial distribution is selected from the Dirichlet distribution, which is often adopted as a prior distribution for a multinomial distribution.

The LDA generative process consists of choosing parameters for each document $\mathbf{w}$ as follows.

1. Choose $\theta \sim$ Dirichlet(α).
2. For each i^{th} word w_i in document $\mathbf{w}$,
   (a) choose a topic $z_i \sim$ multinomial(θ), and
   (b) choose a word w_i from $p(w_i|z_i, \beta)$, a multinomial distribution conditioned on the topic z_i,

where α and β are hyper-parameters for a corpus that was assumed to be previously fixed in this paper, θ is determined for a document, and w_i and z_i for a word.

The probability over the i^{th} word for a multinomial distribution θ is given by

$$p(w_i|\theta, \beta) = \sum_{z_i} p(w_i|z_i, \beta)p(z_i|\theta). \tag{1}$$

The probability $p(z_i|\theta)$ characterizes a document by the topics, which have lower dimensions K than the words. Each topic is represented by the word-occurrence $p(w_i|z_i, \beta)$.

With multiplication of all the N words in a document $\mathbf{w}$ and integration over θ, the occurrence distribution of a document $\mathbf{w}$ is computed as

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{i=1}^{N} \sum_{z_i} p(w_i|z_i, \beta)p(z_i|\theta) \right) d\theta. \tag{2}$$

Taking the product of all the documents in a corpus, we obtain the occurrence probability of the corpus. We use the Gibbs sampling technique [7] to estimate the parameters for the probability of the corpus and obtain the approximate distribution $p(w_i|z_i, \beta)$ and the parameter θ, which is allocated to each document.

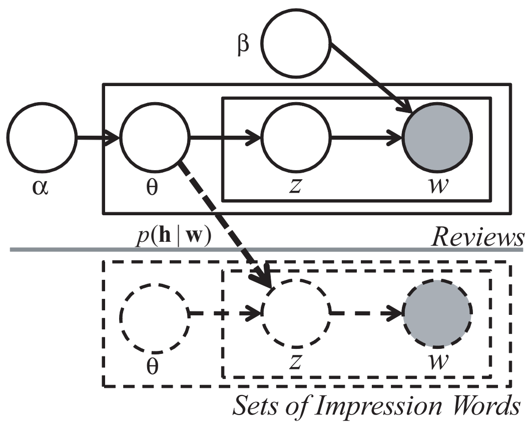


Figure 3. Graphical model representation of Latent Dirichlet Allocation and of detecting impressions given by music reviews. The upper outer rectangle represents reviews, and the inner rectangle represents the chosen topics and words in a review. The bottom outer rectangle represents sets of impression words. We estimate impressions of music by calculating the probability $p(\mathbf{h}|\mathbf{w})$ that a multinomial distribution for a review $\mathbf{w}$ generates a set of impression words $\mathbf{h}$.

After analyzing a corpus, we calculate the probability that a topic distribution for a document would generate a set of impression words. The distribution is denoted by $p(\mathbf{h}|\mathbf{w})$, where $\mathbf{h}$ is a variable for Hevner's sets of impression words H and is one of the sets. A graphical model representation of LDA and of detecting impressions given by documents is shown in Fig. 3. Through Bayes' theorem, $p(\mathbf{h}|\mathbf{w})$ is represented by only the product $p(\mathbf{w}|\mathbf{h})p(\mathbf{h})$:

$$p(\mathbf{h}|\mathbf{w}) = \frac{p(\mathbf{w}|\mathbf{h})p(\mathbf{h})}{\sum_{\mathbf{h}} p(\mathbf{w}|\mathbf{h})p(\mathbf{h})}, \tag{3}$$

where parameter β is omitted and $p(\mathbf{h})$ is assumed to be the same for all $\mathbf{h}$ in H.

The probability $p(\mathbf{w}|\mathbf{h})$ is divided by the latent parameters or the topics:

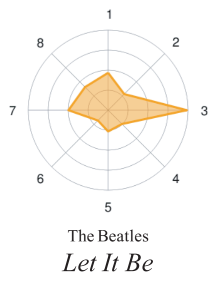
$$p(\mathbf{w}|\mathbf{h}) = \prod_{i=1}^{N} \sum_{z_i} p(w_i|z_i)p(z_i|\theta_{\mathbf{h}}). \tag{4}$$

$\theta_{\mathbf{h}}$ is a parameter of a multinomial distribution for a set of impression words $\mathbf{h}$, which is estimated regarding the set as a document.

Finally, summing up over all documents for a music piece, i.e., reviews, we obtain the probability $p(\mathbf{h}|m)$ that a music piece m generates an impression $\mathbf{h}$:

$$p(\mathbf{h}|m) = \sum_{\mathbf{w} \in D_m} p(\mathbf{h}|\mathbf{w})p(\mathbf{w}|m), \tag{5}$$

where D_m is a collection of reviews for music piece m and we assume the same distribution for $p(\mathbf{w}|m)$, i.e., $1/\left|D_m\right|$. The probability $p(\mathbf{h}|m)$ can be explained as an impression represented by a set of words $\mathbf{h}$ at the probability $p(\mathbf{h}|m)$ obtained by a user listening to music m.



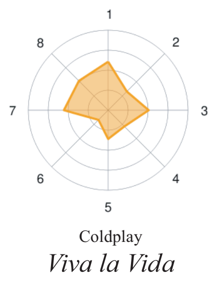


Figure 4. The left chart represents *Let It Be* by *The Beatles*. The right chart represents *Viva la Vida* by *Coldplay*. The numbers correspond to those in Fig. 2.

Examples are shown in Fig. 4. The reviews for the two pieces of music were downloaded from *Amazon.com*, and the probability $p(\mathbf{h}|m)$ was visualized by *Google Chart API*[1] .

There are two reasons we put the topic distributions into eight impression categories. First, to measure the similarity between music pieces effectively, we should select the most suitable topic, i.e., give weight to topics that strongly represent the music features and reduce the weight of those that do not relate to the features. This is because all the topics do not necessarily represent features of the music, e.g., a topic may simply indicate that a music piece is expensive. Second, to convey to users why the specific results were returned, the music must be visualized in some way. This is important particularly in a situation when a user wants to find unknown music.

3.3 Probabilistic Mixture Model

In the previous subsection, we characterized music pieces by $p(\mathbf{h}|m)$, which is the probability that the music m gives an impression $\mathbf{h}$ represented by some adjectives. On the basis of this probability, two music pieces input by users are combined and a new probability for the result of the mixture is generated. A basic method is to compute the average of two given distributions $p(\mathbf{h}|m_x)$ and $p(\mathbf{h}|m_y)$, i.e., $\{p(\mathbf{h}|m_x) + p(\mathbf{h}|m_y)\}/2$. However, this is likely to provide a flattened distribution whose probabilities are similar. An ordinary average operation has a potential problem: a remarkable feature on the distribution may be ignored in the result of the combination. Thus, we propose two mixing operations for two input distributions that can be used in different situations.

3.3.1 Feature-preserved Mixture

To combine two music pieces while preserving their features, we suppose the following probabilistic process.

1. Choose one of two input music pieces at a $1/2$ probability.

2. Repeat two impression extractions from the chosen music until the extracted impressions converge.

3. Adopt the concurrent impression as the result of the mixture for the two music pieces.

The process is given by the following equation:

$$p(\mathbf{h}|m_z) = \frac{1}{2}\left\{\frac{p(\mathbf{h}|m_x)^2}{\sum_{\mathbf{h}} p(\mathbf{h}|m_x)^2} + \frac{p(\mathbf{h}|m_y)^2}{\sum_{\mathbf{h}} p(\mathbf{h}|m_y)^2}\right\}, \quad (6)$$

where $p(\mathbf{h}|m_x)$ and $p(\mathbf{h}|m_y)$ are the distributions over the impressions for input music m_x and m_y, respectively, and $p(\mathbf{h}|m_z)$ is that for virtual music m_z assumed to be the result of the mixture.

The operation to adopt the concurrent impression enhances the outstanding probability in each distribution. This method to combine two music pieces is suitable for a situation where users want music that has the remarkable features of both pieces.

3.3.2 Product Mixture

The second approach to mix two music pieces effectively is to accentuate the features common to both music inputs. This is achieved by the formula

$$p(\mathbf{h}|m_z) = \frac{p(\mathbf{h}|m_x)p(\mathbf{h}|m_y)}{\sum_{\mathbf{h}} p(\mathbf{h}|m_x)p(\mathbf{h}|m_y)}. \quad (7)$$

This operation corresponds to the following process.

1. Repeat extractions of the impression from each music piece until the extracted impressions converge.

2. Adopt the concurrent impression as the result of the mixture for the two music pieces.

This method is suitable for a situation where users want music that has a remarkable feature common to input music m_x and m_y. It can be applied for recommending music by using multiple music pieces listened to by users as a query.

3.4 Ranking by Similarity between Music Pieces

The virtual music resulting from the combination of two music inputs is characterized by a distribution $p(\mathbf{h}|m_z)$, and the music in a system is ranked by closest similarity and returned as a search result. Here, defining the similarity between two music pieces is necessary.

Generally, the *Kullback-Leibler divergence* $D_{KL}(p||q)$ is used for the similarity of probabilistic distributions p and q. This function is not symmetric, thus we take the average of the two versions and define the similarity between two music pieces m_x and m_y, letting $p = p(\mathbf{h}|m_x)$ and $q = p(\mathbf{h}|m_y)$:

$$\mathrm{Sim}(m_x, m_y) = \exp\left[-\frac{1}{2}\{D_{KL}(p||q) + D_{KL}(q||p)\}\right]. \quad (8)$$

Given the distribution $p(\mathbf{h}|m_z)$ for a virtual music piece, each music piece $m \in M$ in a system is returned on the basis of the similarity $\mathrm{Sim}(m_z, m)$.

[1] http://code.google.com/intl/en/apis/chart/

Number of impression neighbors	Average percentage of songs in same genre
1	0.579
5	0.523
10	0.488
20	0.454
50	0.407
100	0.359
All	0.152

Table 1. Average percentage of most similar songs in same genre

4. IMPLEMENTATION

We collected music pieces and reviews from Amazon.com with Amazon Web Services [2], querying by artist names that are listed in CAL500 [16]. We obtained 86,050 pieces, for which 879,666 reviews were written; the average number of reviews per artist was about 10.2. The obtained reviews were analyzed by GibbsLDA++ [3], which is an implementation of Gibbs sampling for LDA. As parameters in LDA, we fixed the number of topics $K = 100$ and hyper-parameters $\alpha = 50/K$ and $\beta = 0.1$. We then conducted 1000 iterations of Gibbs sampling for the parameter estimation.

5. EVALUATION

5.1 Evaluation of Characterization

Before evaluating our system, the performance of characterization by impressions must be clarified. We evaluated our method in accordance with the objective evaluation by Aucouturier et al. [1]. We calculated the correlation between impression and genre similarity by using the songs in our system. Because Amazon.com has multiple labels on songs, only 356 songs that had only 1 label and more than 20 reviews were used in our evaluation , and the top 11 genres used in our experiment were R&B, country, rap and hip-hop, classic rock, classical, jazz, blues, pop, alternative rock, world music, and soundtracks.

The results can be seen in Table 1 and Fig. 5. In Table 1, the closest k songs for a song were retrieved, the percentage of the same genre was calculated, and the average was taken for all the songs. There was a low correlation between impression and genres. As indicated in the study on timbre and genre [1], this approach cannot measure the performance correctly because two songs in the same genre do not always give similar impressions. However, comparing the results with those of timbre similarity, we could show the effectiveness of review-centered characterization.

A similarity matrix for each genre is shown in Fig. 5. Each cell represents the average of the similarity between songs in two genres. We could see a difference between songs in the same genre and different genres except in the alternative rock genre.

[2] http://aws.amazon.com/

[3] http://gibbslda.sourceforge.net/

	R&HH	Country	Classic Rock	WM	R&B	Jazz	Classical	AR	Sound-tracks	Blues	Pop
Rap&Hip-Hop	0.879	0.824	0.841	0.838	0.836	0.531	0.678	0.776	0.811	0.614	0.827
Country	0.824	0.871	0.821	0.806	0.827	0.505	0.719	0.791	0.827	0.555	0.837
Classic Rock	0.841	0.821	0.859	0.837	0.821	0.498	0.687	0.781	0.797	0.699	0.829
World Music	0.838	0.806	0.837	0.852	0.836	0.526	0.679	0.759	0.781	0.691	0.817
R&B	0.836	0.827	0.821	0.836	0.850	0.506	0.691	0.765	0.791	0.619	0.828
Jazz	0.531	0.505	0.498	0.526	0.506	0.766	0.579	0.479	0.541	0.343	0.502
Classical	0.678	0.719	0.687	0.679	0.691	0.579	0.735	0.682	0.734	0.470	0.712
Alternative Rock	0.776	0.791	0.781	0.759	0.765	0.479	0.682	0.750	0.770	0.568	0.788
Soundtracks	0.811	0.827	0.797	0.781	0.791	0.541	0.734	0.770	0.833	0.539	0.806
Blues	0.614	0.555	0.699	0.691	0.619	0.343	0.470	0.568	0.539	0.878	0.613
Pop	0.827	0.837	0.829	0.817	0.828	0.502	0.712	0.788	0.806	0.613	0.841

Figure 5. Similarity matrix for 11 genres. Each cell represents the average of similarity between songs in two genres. The black cells represent the maximum similarity in each row, and the gray cells represent the 2nd and 3rd maximum similarity within 10% of the maximum in each row.

5.2 Evaluation of Query-by-Mixture-of-Example

Comparing with results returned by a human, we investigated the performance of our proposed method to search with a query by mixture of example. We asked a student who knows music pieces well to choose reasonable songs as mixture of input queries listed in Table 2. Then, we asked 5 persons to listen input music and output music including both music recommended by human and returned by *RhythMiXearch*, and to evaluate relevance of the outputs in five levels. The result is shown in Fig. 6, where the average scores were taken for each question, and the question numbers correspond to those in Table 2.

In some questions, music recommended by human were considered more relevant than the results returned by *RhythMiXearch*. Our system is inferior to human in performance, however, the result by human should be regarded as the upper bound in the evaluation. In the questions 4 and 5, the results by *RhythMiXearch* obtained higher scores, whereas in the questions 2 and 3, our method failed to return relevant results for mixture of example. The result may show that a human can recommend music only for similar two music like the inputs seen in the question 2 and 3, on the one hand, our system can search for music even for different types of music like the inputs used in the question 4 and 5.

6. CONCLUSION

We presented a novel method for searching for unknown music and also presented our developed system *RhythMiXearch*, which can accept two music inputs and mix those inputs to search for music that could reasonably be a result

#	Input A	Input B	Human	Feature-Preserved Mixture	Product Mixture
1	The Beatles, Let It Be	Coldplay, Viva La Vida	Bob Dylan, Blowin' In the Wind	Kiss, Dynasty	The Black Crowes, Lions
2	Michael Jackson, Thriller	Madonna, Like a Virgin	Jamiroquai, Cosmic Girl	Jimi Hendrix, The Jimi Hendrix Experience	*
3	Eminem, The Eminem Show	Britney Spears, Britney	TLC, Silly Ho	Green Day, Nimrod	*
4	Eric Clapton, 461 Ocean Boulevard	John Lennon, Imagine	Eagles, New Kid in Town	Eric Clapton, Me and Mr. Johnson	Cream, Disraeli Gears
5	The Cardigans, First Band on the Moon	Whitney Houston, Whitney	Janis Joplin, Half Moon	Christina Aguilera, Stripped	*

Table 2. 5 set of inputs and outputs for evaluation of Query-by-Mixture-of-Example. (* means the same result as Feature-preserved Mixture.)

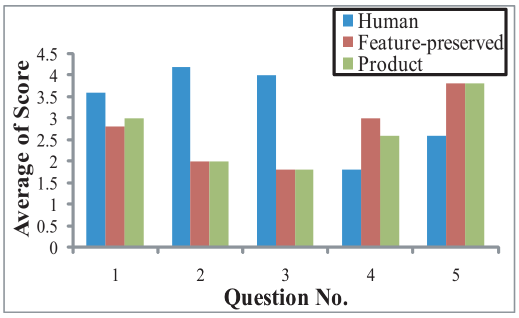


Figure 6. Average of scores for each question

of the mixture. Our first contribution was to characterize music pieces by reviews with LDA and to evaluate the performance of the representation of the music pieces. The second contribution was to propose a probabilistic mixture model for processing multiple example queries. We believe that *Query-by-Mixture-of-Examples* is an important concept for searching for new music pieces.

7. ACKNOWLEDGEMENTS

This research was supported by Exploratory IT Human Resources Project (MITOH Program Youth 2008). I express my sincere gratitude to Professor Michiaki Yasumura, Keio University, for comments on my work. Thanks to GOGA, Inc. for supporting the project. Thanks to members of Tanaka laboratory, especially Yusuke Yamamoto, for helping the user experiments and discussion on my work.

8. REFERENCES

[1] J.J. Aucouturier and F. Pachet. Music similarity measures: Whatfs the use. In *Proc. of the 3rd ISMIR*, pages 157–163, 2002.

[2] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[3] R. Cai, C. Zhang, C. Wang, L. Zhang, and W.Y. Ma. MusicSense: contextual music recommendation using emotional allocation modeling. In *Proc. of the 15th Multimedia*, pages 553–556, 2007.

[4] R.B. Dannenberg and N. Hu. Understanding search performance in query-by-humming systems. In *Proc. of the 5th ISMIR*, pages 232–237, 2004.

[5] J. Foote, M. Cooper, and U. Nam. Audio retrieval by rhythmic similarity. In *Proc. of the 3rd ISMIR*, pages 265–266, 2002.

[6] M. Goto and K. Hirata. Recent studies on music information processing. *Acoustical Science and Technology*, 25(6):419–425, 2004.

[7] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proc. of the National Academy of Sciences*, 101(90001):5228–5235, 2004.

[8] K. Hevner. Experimental studies of the elements of expression in music. *The American Journal of Psychology*, pages 246–268, 1936.

[9] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A music search engine built upon audio-based and web-based similarity measures. In *Proc. of the 30th SIGIR*, pages 447–454, 2007.

[10] Peter Knees, Elias Pampalk, and Gerhard Widmer. Artist classification with web-based data. In *Proc. of the 5th ISMIR*, pages 517–524, 2004.

[11] Peter Knees, Tim Pohle, Markus Schedl, Dominik Schnitzer, and Klaus Seyerlehner. A document-centered approach to a natural language music search engine. In *Proc. of the 30th ECIR*, pages 627–631, 2008.

[12] J. Paulus and A. Klapuri. Measuring the similarity of rhythmic patterns. In *Proc. of the 3rd ISMIR*, pages 150–156, 2002.

[13] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Meaningfully Browsing Music Services. In *Proc. of the 8th ISMIR*, pages 23–30, 2007.

[14] M. Tolos, R. Tato, and T. Kemp. Mood-based navigation through large collections of musical data. In *Proc. of the IEEE 2nd CCNC*, pages 71–75, 2005.

[15] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proc. of the 9th ISMIR*, pages 325–330, 2008.

[16] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the CAL500 data set. In *Proc. of the 30th SIGIR*, pages 439–446, 2007.

[17] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. A survey of music information retrieval systems. In *Proc. of the 6th ISMIR*, pages 153–160, 2005.

[18] G. Tzanetakis, A. Ermolinskyi, and P. Cook. Beyond the query-by-example paradigm: New query interfaces for music information retrieval. In *Proc. of the 2002 ICMC*, pages 177–183, 2002.

MUSICAL STRUCTURE RETRIEVAL BY ALIGNING SELF-SIMILARITY MATRICES

Benjamin Martin, Matthias Robine and Pierre Hanna
LaBRI - University of Bordeaux
351, cours de la Libération
33405 TALENCE Cedex - FRANCE
firstname.name@labri.fr

ABSTRACT

We propose a new retrieval system based on musical structure using symbolic structural queries. The aim is to compare musical form in audio files without extracting explicitly the underlying audio structure. From a given or arbitrary segmentation, an audio file is segmented. Irrespective of the audio feature choice, we then compute a self-similarity matrix whose coefficients correspond to the estimation of the similarity between entire parts, obtained by local alignment. Finally, we compute a binary matrix from the symbolic structural query and compare it to the audio segmented matrix, which provides a structural similarity score. We perform experiments using large databases of audio files, and prove robustness to possible imprecisions in the structural query.

1. INTRODUCTION

Content-based search on very large audio files databases is an important issue in music information retrieval. New browsing tools propose to compare audio songs according to music properties such as style, rhythm, melody, timbre, etc. Among all of these properties, taking into account information about structure may be very useful for discriminating songs, since it may be closely linked to music style or music composer. In this paper, we propose to focus on these structural properties.

Musical structure has been of major concern over the last years. This field aims to retrieve and compare human-recognizable musical structure within an audio piece. To this end, Foote proposed in 1999 [1] a self-similarity matrix-shaped representation whose coefficients carry structural information over the musical piece. This matrix is obtained analyzing repeated sections within the piece; it describes both a global structure and different local structures.

Existing works about music structure generally focus only on structural analysis in audio files. Foote *et al.* proposed a music summarization method by summing scores in its self-similarity matrix representation [2]. Dannenberg tested several transcription methods by adapting them to the nature of the given audio file, in order to explicitly retrieve the underlying structure [3]. Müller *et al.* proposed a method to extract relevant paths in a self-similarity matrix, deducing the precise structure of the music piece [9]. Bartsch developed an automatic thumbnailing system that retrieve relevant parts from audio [7], and Goto focused later in chorus extraction over songs taking into account possible modulations or variable durations of their occurrences in the audio musical piece [6]. Peeters used [5] a representation in terms of ”states” of music and proposed an algorithm that computes a structural summarization in several passes over the audio file. Bartsch *et al.* worked [4] on a summarization centered on choruses using chroma features. Paulus *et al.* used [8] several audio features to build a probability space in order to analyze the best probable underlying structure.

Once the structure is correctly analyzed, a retrieval system may be directly developed by comparing the sequences of structure. The problem is the correctness of the structure estimated. Although several systems have been proposed and evaluated, analysis errors significantly limit the accuracy of a retrieval system based on such approaches. Lately, Izumitani and Kashino proposed a method that estimates the structural similarity between two excerpts by directly comparing the self-similarity matrices computed [12]. This method is applied to cover song detection.

The method we propose trails after the same principle: comparing musical form without extracting explicitly the underlying audio structure. In this paper, we bring a new retrieval system based on musical structure using symbolic queries. In Section 2, we describe the system proposed. In Section 3, we present different experiments on real pop music databases. Finally, we conclude and open perspectives in Section 4.

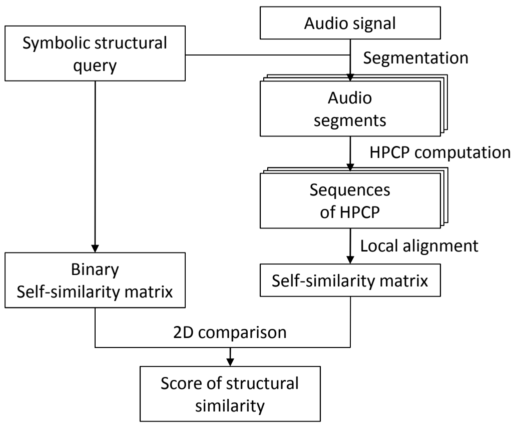


Figure 1. Overview of the query-by-structure method proposed.

2. METHOD

The proposed method searches the database for the musical piece that best matches a given symbolic query. See Figure 1 for the method global schematic view.

2.1 Audio Self-Similarity matrix computation

First, the system splits an audio file into n audio segments according to a given segmentation. The applied segmentation used in our retrieval system will be described later.

2.1.1 Features

In the proposed method, the comparison is based on musical structures, not directly on audio features. Therefore, it is fundamental to keep in mind that the chosen audio features can be changed, regardless of the further steps of the method.

As a feature set, we use *Harmonic Pitch Class Profiles* (HPCP) [11]. HPCPs, which provide tonal information, are robust to noise, timbre, dynamics, tuning or loudness variations, and ensure then an accurate tonal description. The different tonality vectors are extracted the same way as in [11] or [10], frame by frame.

This analysis transforms an input audio file into a sequence $H = (\overrightarrow{h_i})_{1 \leq i \leq n}$ of N B-dimensional vectors $\overrightarrow{h_i}$, where N denotes the number of frames in the musical piece, and B denotes the chosen chroma bin resolution (generally 12, 24 or 36). Our system settles for a 12 bins resolution.

The method proposed compares the signal to itself in order to retrieve structural information (see Section 2.1.2). That's why an adapted measure that enables the comparison between two HPCP vectors is needed. We choose the binary local alignment technique described in [10] to this purpose. On top of being adapted to HPCPs, this measure computes the optimal transposition index between two chromas to provide a similarity score, which allows our detection to be robust to key changes within the same audio musical piece. This comparison measure is able to compute from two features sequences H_1 and H_2 a similarity score by local alignment.

2.1.2 Segmented self-similarity matrix of an audio file

As explained before, we use self-similarity matrices in order to represent the repeated sections. In our model, self-similarity matrices contain elements, whose range is $[0, 1]$, that stand for the likeliness between two parts of a structure. Horizontal and vertical axis of the matrix represent time, that runs from left to right as well as from top to bottom.

In classic uses of self-similarity matrices, each i, j coefficient is computed by comparing the feature corresponding to the time i of the musical piece with the one corresponding to the time j of the same musical piece. However, in our model, contrary to the general self-similarity computing process, time does not run uniformly on both axis. Indeed, each element i, j of the matrix corresponds to the similarity measure between two entire parts P_i and P_j of the musical piece. Thus, each coefficient corresponds to the evaluation of the similarity between two sets of feature vectors, not directly between two vectors.

Based on the n audio file segments computed from the audio signal, n sequences $H_1, H_2, \ldots, H_n$ of HPCP vectors are computed, each one corresponding to an audio segment. Therefore, comparing two HPCP sequences means comparing two segmented parts of the audio signal. Thus, for each couple of HPCP sequences H_1 and H_2, we compute a similarity score using the binary local alignment by dynamic programming technique inspired from Gomez's work and described in 2.1.1. This provides a self-similarity matrix R, whose coefficients stand for the comparison of two parts within the musical piece:

$$R = (\text{alignment}(H_i, H_j))_{1 \leq i \leq n, 1 \leq j \leq n} \quad (1)$$

where alignment() denotes the binary local alignment by dynamic programming technique used to compare two HPCP sets. An example of reference segmented self-similarity matrix is shown on Figure 2 (right).

2.2 Query matrix computation

In a first approach, the query used for comparisons is a binary self-simlarity matrix computed from a symbolic structural query, that can be arbitrarily defined or taken from a ground truth.

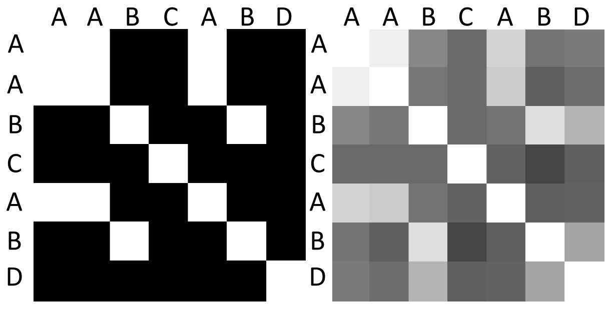


Figure 2. Left: Symbolic query self-similarity matrix of *The Beatles - All My Loving* musical piece. Right: Segmented reference self-similarity matrix of the same musical piece. Letters show the different recognizable patterns; white pixels stand for repeated sections, and black pixels stand for distinct sections. The left reference matrix is segmented according to the ground truth for this musical piece (exact structure).

2.2.1 Symbolic structural query

Our model uses symbolic structural queries, which are a symbolic representation of the underlying structure of a musical piece. A symbolic query can be seen as a sequence of symbols that represent a particular musical form. Each part within a symbolic structural query has its own duration. It can represent a simple note as well as an entire excerpt.

A symbolic structural query can be seen as a sequence of symbols, for instance 'aabca', combined or not to symbols representing duration information. Two identical symbols within the sequence indicate a similarity between the two corresponding parts, and two different symbols indicate a dissimilarity.

2.2.2 Self-similarity matrix of a symbolic query

Symbolic queries are comparable to pattern sequences that impose two kinds of constraints : similarities, *i.e.* remarkable repetitions, and dissimilarities.

The symbolic query self-similarity matrix is created by analyzing the provided patterns sequence. Assuming that $(s_k)_{1 \leq k \leq n}$ represents a symbols sequence of length n (*e.g.* $s =$ '$ababc$'), the query self-similarity matrix Q is defined as follows:

$$\forall (i,j) \in \{1 \ldots n\}^2,\ Q_{i,j} = \begin{cases} 1 \ if \ s_i = s_j \\ 0 \ if \ s_i \neq s_j \end{cases} \quad (2)$$

The resulting matrix is binary, and stands for 2 types of constraints: similarity and dissimilarity. An example of a self-similarity matrix created from a symbolic query can be viewed in Figure 2 (left).

2.3 Matrices comparison

In order to assign a similarity score between the matrices we compute, we use three different algorithms that have different properties. These three algorithms provide a normalized similarity score according to a binary query matrix and a reference matrix. From now on, the two matrices compared are denoted as Q and R.

The first approach consists in computing the similarity between matrices using a pixel-to-pixel algorithm, based on an euclidean distance algorithm. However, we consider more sophisticated algorithms that show different characteristics.

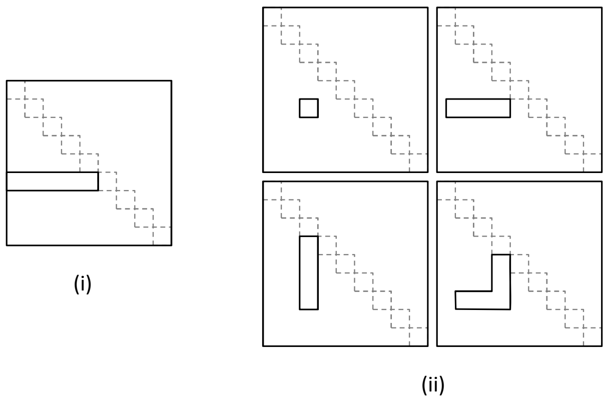


Figure 3. Possible compared patterns taken into account in local alignment with (i) Izumitani's algorithm and (ii) Lecroq's algorithm.

An approach to the comparison problem can be yielded by local alignment algorithms. In our context, this principle is extended to matrices comparison; possible operations in order to transform the first matrix into the second work on symbols that can be seen as clusters of the compared matrices.

In some cases, it can be very relevant to consider pixel-wise deletions or insertions in the matrices comparison. Moreover, if the query binary matrix is smaller than the compared reference matrix (structure excerpts search), local alignment techniques are able to cope with the dimension difference and to provide a local similarity score.

2.3.1 *Izumitani and Kashino's algorithm*

The first local alignment algorithm tested is the one presented in [12]. This algorithm takes as an input two self-similarity matrices. Since these matrices are symmetrical and have constant maximum diagonal, this algorithm only works on the lower triangles. It is based on a dynamic programming method that searches the diagonal direction of the reference self-similarity matrix. Indeed, the Izumitani and Kashino's algorithm compares each entire line of the reference matrix with each entire line of the query matrix. Thus, for each comparison, it allows the 3 different operations (insertion/deletion/substitution) on a unique pattern: entire lines of the lower triangle of the compared matrices. This pattern can be seen on Figure 3 (left).

Furthermore, the dynamic programming matching method is based on "matched element indices sets" recursively computed, which means that at each step n, matching elements between two compared lines are deduced from the $n-1$ step (see [12], 2.3, p.612). In other words, if an element does not match the compared line at a step, the whole following column elements will not be taken into account in any further comparison. This represents a limitation of this algorithm.

2.3.2 *Lecroq et al.'s algorithm*

In order to improve the alignment comparison, we chose to consider a different method. Indeed, reducing considered patterns for the comparisons to lines only seemed to be rather limited, which led us to evaluate a new method.

The second local alignment algorithm studied is adapted from [13]. It was developed and used in order to compare symbolic dialog annotations, and is particularly specialized in aligning 2-dimensional patterns. Lecroq *et al.*'s algorithm browses the matrices element by element, and allows the 3 typical operations on different patterns: a single element (pixel), a part of a line or an entire line, a part of a column or an entire column, and a part of a line and a part of a column simultaneously. These different patterns are represented on Figure 3 (right).

Our adaptation consisted in not comparing text entries but patterns included in self-similarity matrices, taking into account their properties and adapting the comparison to a non-binary similarity measure.

2.4 Retrieval system specificities

2.4.1 *Query-based segmentation*

Symbolic structural queries can be combined or not to informations about the duration of each pattern.
If the query indicates absolute time informations (in frames or seconds), these can be used to split the musical piece in segments.
If the query indicates relative time informations, the global duration of the piece can be used to split the musical piece.
If no time information is provided in the symbolic query, segmentation is arbitrary: for instance, it can be uniformly processed, each part having the same duration than the others.

2.4.2 *Pre-processing*

Before evaluating a similarity score, the reference self-similarity matrix must be pre-processed. Indeed, since the reference matrix contains the similarity scores between the different parts of the musical piece, the distribution of the values of its coefficients is likely to vary according to the considered musical piece. For some audio files, the tonal distinction between two parts that are supposed to be different, e.g. a chorus and a verse, will be very clear, whereas it will turn out to be vague in some other cases.

Let μ and σ be respectively the average and the standard deviation values of the coefficients' distribution in the reference matrix R. The normalized reference self-similarity matrix $\hat{R}$ is computed as follows:

$$\forall (i,j) \in \{1 \dots n\}^2, \hat{R_{i,j}} = \frac{(R_{i,j} - \mu)}{\sigma} \cdot \hat{\sigma} + \mu \qquad (3)$$

where $\hat{\sigma}$ is a constant corresponding to the new standard deviation to apply. It must be adapted to the pixel-to-pixel comparison constants (see 2.4.3). In our model, we used $\hat{\sigma} = 0,31$.

2.4.3 *Adapted euclidean distance*

In a first approach, we compare matrices by computing an euclidean score, based on pixel-to-pixel comparisons. As explained before, the binary query matrix contains similarity and dissimilarity constraints. However, our method does not give these two constraints the same importance.

Actually, we can reasonably hypothesize that two parts that are supposed to be similar (*i.e.* that are represented by the same symbol in the query) present two close tonal descriptions, whereas two parts that are supposed to be dissimilar can be either close, or different in their tonal descriptions: there is no gradation on the dissimilarity notion. Therefore, it is necessary to make an distinction between the strong similarity constraints and the weak dissimilarity constraints that imposes the symbolic query.

Our adapted euclidean distance computes then two different scores, which takes into account this distinction: A similarity score $s^{=} \in [0,1]$, that is established only with similarity constraints imposed by the query,
A dissimilarity score $s^{\neq} \in [0,1]$, that is established only with the dissimilarity constraints of the query.
Let Q and R denote the query and reference compared matrices, respectively. We introduce the set

$$R^{=} = \{(i,j) | Q_{i,j} = 1, i < j\}$$

that denotes the reference matrix indices that correspond to a similarity (white pixel, value 1) in the query. In the same way, we introduce the set

$$R^{\neq} = \{(i,j) | Q_{i,j} = 0, i < j\}$$

that denotes the reference matrix indices that correspond to a dissimilarity (black pixel, value 0) in the query. $s^=$ is computed comparing each similarity element of the query with the corresponding element in the reference the following way:

$$s^= = \frac{1}{|R^=|} \sum_{(i,j) \in R^=} ||1 - R_{i,j}|| \qquad (4)$$

where $||.||$ denotes the classical euclidean norm. $s_{\neq}$ is computed comparing each dissimilarity element of the query with the corresponding element in the reference the following way:

$$s^{\neq} = \frac{1}{|R^{\neq}|} \sum_{(i,j) \in R^{\neq}} f(R_{i,j}) \qquad (5)$$

where f denotes an exponential shaped function that was empirically determined :

$$f(x) = \frac{1}{e^8 - 1} \cdot (e^{8 \cdot x} + 1)$$

3. EXPERIMENTS

In order to evaluate our method in the most systematic way, we established a serie of experiments that underline its different characteristics.

3.1 Databases establishment

We based our research on a structural ground truth corpus created by M. Levy, K. Noland and G. Peeters [1]. It includes 60 accurate XML annotations for western pop songs, and numbers for each musical piece every detected section with its duration. It was found to be one of the most used corpora in this field, in many prior studies, such as [14] or [15].

We used two different audio files databases in order to validate our model.
- The ground-truth corresponding audio files database $\mathcal{D}_g$, that includes the 60 musical pieces annotated in the corpus;
- A noise database $\mathcal{D}_m$, that contains 200 audio western pop music audio files from different authors.
Obviously, we respected the following inclusion:
$\mathcal{D}_g \subset \mathcal{D}_m$. Audio files were taken from commercial CD versions.

We analyzed signals using HPCP features (see 2.1.1) with an overlap of 50% and a window size of 744 milliseconds.

3.2 Exact structural queries

The first experiment consists in searching musical pieces through a large database with the prior knowledge of their exact structure and time segmentation. Thus, we generated queries according to the symbolic representation and segmentation provided by our ground truth corpus. To compute the reference matrices, we segmented the $\mathcal{D}_m$ files according to each available ground truth data given in $\mathcal{D}_g$ (60 files). Segmentation was carried out with relative time information provided by the queries.

We then computed similarity scores between each query and the 200 well-segmented references, and checked whether the best matching was made on the file corresponding to the query. To do so, since the applied segmentation and symbolic query were supposed to be exact, we used the adapted euclidian distance described in Section 2.4.3. Because of the high accuracy of the used symbolic queries and segmentations, this simple algorithm was as efficient as local alignment techniques for this experiment.

[1] http://www.elec.qmul.ac.uk/digitalmusic/downloads/index.html#segment

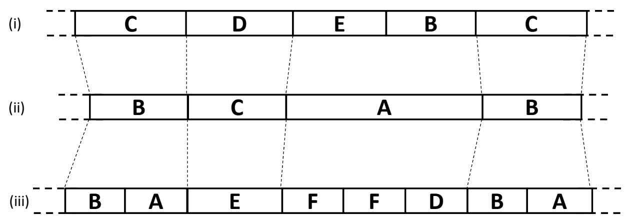


Figure 4. Structural excerpts concordance between the three best results matching with *All My Loving* excerpt. (i) *Stop the rock*, (ii) *All My Loving* - the queried structural excerpt, and (iii) *A Day in the Life*. Rectangles widths are commensurate with pattern durations; Dashed lines show structural concordances.

For all of the 60 cases tested, the experiment was conclusive: the corresponding audio file was best matched. Therefore, the system is able to retrieve exactly a musical piece providing its exact structure and segmentation.

3.3 Structure excerpts queries

The second experiment consists in searching musical pieces through a database with the prior knowledge of a part of their exact structure and time segmentation. In other words, knowing an excerpt of a structure and the segmentation of a musical piece, we now aim at retreiving the entire original musical piece as well as any piece that contains the same given structural excerpt.

As a symbolic query, we chose a structural excerpt from the *All My Loving* by *The Beatles* musical piece:
'VCBV' (Verse - Chorus - Bridge - Verse).
We assume that the time segmentation of this structure is known, *i.e.* we know how long each part lasts.

Since the symbolic query matrix and the reference matrix do not have the same size, the pixel-to-pixel distance is irrelevant here. Thus, we tested the alignment of the subrequest matrix on every musical piece of $\mathcal{D}_g$ segmented according to the *The Beatles* piece with Izumitani et al.'s and Lecroq et al.'s algorithms.

With each of both algorithms, the best matching was obtained on the original Beatles piece. However, the second best matched result differs from one algorithm to the other. Figure 4 shows the matched structure excerpt on the best matched pieces: the original *The Beatles* piece (ii), that was best matched on both of the algorithms, the second best matched piece with Izumitani's algorithm (i), and the second best matched piece with Lecroq's algorithm (iii). The indicated patterns correspond to ground truth data relative to each audio musical piece. Here are the full symbolic structures of these pieces :
All My Loving: 'A A B C B A B D'
A Day in the Life: 'A B B B C D E B C F G'
Stop the Rock: 'A A A A A B C D B A E F F D B A'

By segmenting musical pieces according to the *All My Loving* ground truth, the structures of pieces (i) and (iii) were re-segmented, exhibiting a new structure that highly matched the queried structure excerpt (ii). Dashed lines in Figure 4 show the high pattern matching: pattern similarities and dissimilarities are nearly identical between each of the three best matched pieces.

3.4 Time robustness

In the experiments described above, we processed exact segmentations taken from the ground truth. However, our query-by-structure method aims at getting rid of time constraints, and at being able to retrieve correctly providing exclusively a symbolic query as an entry.

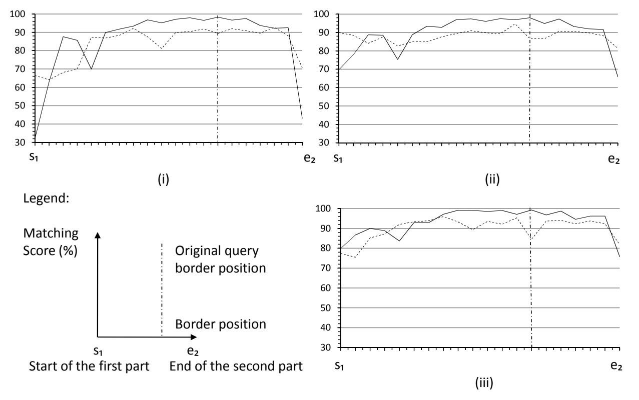


Figure 5. Altering a single time border between two parts of a structure: consequences on similarity scores. (i): Izumitani and Kashino alignment algorithm, (ii): Lecroq alignment algorithm, (iii): Adapted pixel-to-pixel distance. Plain line stand for the *All My Loving, The Beatles* musical piece, while dashed line stands for the second best matching score. Horizontal axis show the border position, from the start of the first part to the end of the second, in seconds. Its length is about 15 seconds.

Therefore, it is necessary to test our algorithms on time constraints, *i.e.* to change more or less the prior exact segmentation in order to evaluate their robustness towards this criterion.

As explained before, a segmentation combined to a symbolic query provides a series of symbolic parts indexed in time. From now on, we will denote as a border of two consecutive parts the exact point in time when ends the first part and starts the second one. The following time robustness experiments consist in changing the position of one or several borders in order to observe the impact on the matching scores.

3.4.1 Changing one segmentation border

The first time robustness test consists in changing one border over a given segmentation. From the ground truth corpus, we chose a musical piece (*All My Loving*), and modified its exact segmentation.

Here is the symbolic structure of this musical piece :
Verse - Verse - Chorus - Bridge - Verse - Chorus - Outro
We chose to work on the first Chorus / Bridge border, for it generally demarcates two distinct parts in their tonal description.

Let s_1, e_1, s_2 and e_2 denote respectively the beginning and the end of the first chorus, and the beginning and the end of the bridge. In order to estimate time robustness on our method, we generated a series of 20 different segmentations changing the border position from s_1 to e_2, keeping the constraint that $e_1 = s_2$. Thus, the only difference between two generated segmentations is the position of this border.

The experiment results are shown on Figure 5. The plain line identifies the scores for the considered musical piece, whereas the dashed line indicates the second best matching score over the database $\mathcal{D}_g$. In the x-axis, the tested position of the border varies from s_1 to e_2, the vertical dashed and dotted line indicating the original position of the border in the ground truth.

For the 3 comparison methods, we can see that the scores obtained for the musical piece (plain lines) vary slightly in an interval that corresponds to ± 6 seconds around the original position of the border. Above this value, the border seems to alter the score in such a way that the tested musical piece does not best match the symbolic query matrix.

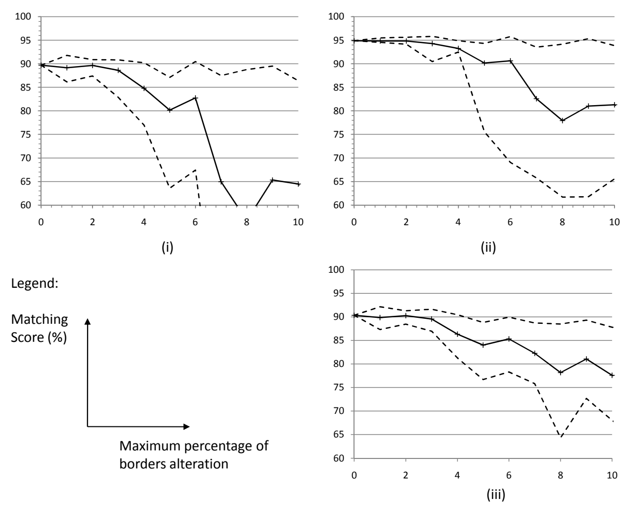


Figure 6. Altering every time borders between parts in a structure: consequences on similarity scores. (i) Izumitani and Kashino alignment algorithm, (ii) Lecroq et al. alignment algorithm, (iii) Adapted pixel-to-pixel distance. Plain line stand for the average over the 10 draws, and dashed lines stand for the greatest and lowest obtained values. The matching scores correspond to the output computed by the different algorithms.

3.4.2 Changing every segmentation borders

After changing a single border in a segmentation, we estimated time robustness on our algorithms by changing every borders in the same segmentation. Let d be the total considered musical piece duration. We introduce p_{max}, expressed as a percentage of the total musical piece duration, that corresponds to the maximum variation of each border in the segmentation (in percent). In other words, introducing $t_{max} = p_{max} \cdot d$, each border in the segmentation may be changed to a new time that does not exceed the old one $\pm t_{max}$. We chose 10 different values for p_{max} from 1% to 10% of the total piece duration, and generated a random segmentation whose borders were changed according to the above principle. This operation was repeated 10 times for each value of p_{max}.

Results can be seen in Figure 6. The figures show the average scores (plain lines) as well as the minimum and maximum scores (dashed lines) obtained over the 10 draws. They show the evolution of similarity scores between the symbolic query and the randomly altered segmentations according to the maximum alteration factor. Considering the second best matching scores obtained on the same symbolic query, the three algorithms seem to be robust to a maximum variation of the borders of 6% of the total piece duration. Above this value, the best matching is realized on a different piece.

3.5 Query-by-Structure

We now consider timeless queries that only impose a symbolic structure. From now on, no information about time segmentation is given: the principle is to search for a musical piece in the database providing exclusively its symbolic structure.

In order to realize this experiment, we focused on a few pieces that make part of the ground truth. This way, we could get the annotated structure of each musical piece and use it as the input symbolic query of our method. However, borders timings were not retrieved from the annotation files, which yielded queries without timing indications. Lacking any indication about time in the segmentation, our system assumes that the different symbols are

		Rank		
#	Musical Piece	Eucl	Izum	Lecroq
1	AllMyLoving	7	6	**4**
2	DevilinHerHeart	2	**1**	2
3	Drive	**1**	**1**	3
4	ItWon'tBeLong	**1**	**1**	**1**
5	Lonestar	84	111	**26**
6	Misery	**1**	**1**	**1**
7	NotaSecondTime	**1**	**1**	**1**
8	TakeOnMe	29	**13**	55
9	Thubthumpning	**3**	15	**3**
10	Wannabe	13	12	**8**
11	WhenI'mSixtyFour	100	90	**54**
12	WithaLittleHelp..	**1**	**1**	**1**
13	Words	**2**	9	10
14	YouReallyGot..	**1**	**1**	**1**
	Average Rank	17.57	18.79	12.14
	Median Rank	2	3.5	3
	MRR	0.544	0.537	0.480

Table 1. Query-by-Structure results with 3 different algorithms: euclidean (Eucl), Izumitani (Izum) and Lecroq.

uniformly distributed over the excerpt. In other words, it assigns the same duration to each symbol in the query.

Results can be viewed in Table 1. We tested 14 different symbolic structural queries with no time informations compared to each file of $\mathcal{D}_m$. The result table shows the retrieval ranks obtained for every musical pieces and for each of the 3 algorithms.

As shown by the two previous experiments, the algorithms used are able to deal with an inexact segmentation over the audio files. However, we saw limitations on the maximum time variation applied on borders. After analysing the ground truth relative to the different tested musical pieces, we could split the test set in two classes:
- Regular structured pieces, whose recognizable patterns have close durations (less than 10% of the duration of the piece). This class contains pieces No. 1, 2, 3, 4, 6, 7, 9, 12, 13, 14 in the table;
- Irregular structured pieces, whose recognizable patterns durations may vary significantly (possibly more than 10% of the duration of the piece). This class contains pieces No. 5, 8, 10, 11.

As we can see, irregular structured excerpts ranks are rather high with the three algorithms, which shows the limitations of our system. However, regular structured pieces are much more precisely retrieved.

At comparing the different algorithms used, we can see that the three algorihms seem to be efficient, euclidian adapted distance providing the best Mean Reciprocal Rank (MRR).

4. CONCLUSION AND FUTURE WORK

We have proposed a music retrieval system based on structural similarity. Considering a symbolic representation of the underlying structure of a musical piece, an audio file can be segmented and compared using self-similarity representation on HPCP features, providing a similarity score that indicates the structural likeliness. We used three different algorithms to compare matrices. We proved that not only the algorithm works exactly on accurate symbolic queries, but it presents also a significant robustness to slight time variations, which even allow searching for structures ignoring timing informations.

An interesting idea as a perspective is to work on a query-by-example oriented system. This time, the query will not be a symbolic structural information but an audio file, arbitrarily segmented. Then, the retrieval will be focused on finding the closest structures to the underlying structure of the input audio file. In our first tests, we managed to get from an input audio file a very structurally similar musical piece.

Finally, our system is based on a tonal representation of the signal. Nevertheless, since the retrieval operation works on structural data, this parameter could be changed to any other one. We should consequently test our system on other features, such as rhythm representations or timbre analysis.

This work is part of the SIMBALS project (JC07-188930), funded by the French National Research Agency.

5. REFERENCES

[1] J. Foote: "Visualizing Audio Segmentation using Self-Similarity," *Proc. of ACM Multimedia*, 77-80, 1999.

[2] J. Foote and M. Cooper: "Automatic Music Summarization via Similarity Analysis," *ISMIR02*, 2002.

[3] R. B. Dannenberg and N. Hu: "Pattern Discovery Techniques for Music Audio," *ISMIR02*, 2002.

[4] M. A. Bartsch and G. H. Wakefield: "Audio Thumbnailing of Popular Music Using Chroma-Based Representations," *IEEE Transactions on multimedia, Vol. 7*, 2005.

[5] G. Peeters, A. L. Burthe and X. Rodet: "Toward Automatic Music Audio Summary Generation From Signal Analysis," *Proc. of ISMIR*, 2002.

[6] M. Goto: "A Chorus-Section Detecting Method for Musical Audio Signals," *Proc. of ICASSP*, 2003.

[7] M. A. Bartsch and G. H. Wakefiels: "To catch a chorus: using chroma-based representations for audio thumbnailing," *Proc. of WASPAA*, 2001.

[8] J. Paulus and A. Klapuri: "Music Structure Analysis Using a Probabilistic Fitness Measure and an Integrated Musicological Model" *Proc. of ISMIR*, 2008.

[9] M. Müller and F. Kurth: "Towards Structural Analysis of Audio Recordings in the Presence of Musical Variations," *EURASIP Journal on Advances in Signal Processing*, Vol. 07, 2007.

[10] J. Serra, E. Gomez, P. Herrera, and X. Serra: "Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification," *IEEE Transactions on Audio, Speech and Language Processing*, 2008.

[11] E. Gomez: "Tonal description of music audio signals," *Ph.D. dissertation*,, MTG, Universitat Pompeu Fabra, Barcelona, Spain, 2006.

[12] T. Izumitani and K. Kashino: "A Robust Musical Audio Search Method Based on Diagonal Dynamic Programming Matching of Self-Similarity Matrices," *Proc. of ISMIR*, 2008.

[13] E. Chanoni, T. Lecroq and A. Pauchet: "Une nouvelle heuristique pour l'alignement de motifs 2D par programmation dynamique (in french)," *JFPDA08 (Planification, Decision and Learning French-Speaking Days)*, Metz, France, pp.83-92, 2008.

[14] E. Peiszer, T. Lidy and A. Rauber: "Automatic Audio Segmentation: Segment Boundary and Structure Detection in Popular Music," *Proc. of LSAS*, Paris, France, 2008.

[15] M. Casey and M. Slaney: "The Importance of Sequences in Musical Similarity," *Proc. of ICASSP06*, Vol. 5, Toulouse, France, 2006.

EXPLORING AFRICAN TONE SCALES

Dirk Moelants
Ghent University
Dirk.Moelants@UGent.be

Olmo Cornelis
University College Ghent
Olmo.Cornelis@hogent.be

Marc Leman
Ghent University
Marc.Leman@UGent.be

ABSTRACT

Key-finding is a central topic in Western music analysis and development of MIR tools. However, most approaches rely on the Western 12-tone scale, which is not universally used. African music does not follow a fixed tone scale. In order to classify and study African tone scales, we developed a system in which the pitch is first analyzed on a continuous scale. Peak analysis is then applied on these data to extract the actual scale used. This system has been applied to a selection of African music, it allows us to look for similarities using cross-correlation. Thus it provides an interesting tool for query-by-example and database management in collections of ethnic music which can not be simply classified according to keys. Next to this the data can be used for ethnomusicological research. The study of the intervals used in this collection, e.g., gives us evidence for Western influence, with recent recordings having a tendency to use more regular intervals.

1. INTRODUCTION

Scale recognition has a long tradition in the analysis of Western music. Already in medieval music theory, determining the mode and classifying pieces according to their mode was a central topic. Also in the music theories of the Middle-East and India, classification of music according to the scale (often connected to a certain 'mood') is an important topic. Not surprisingly, with the advent of computational methods, researchers started to design systems to perform the process of scale recognition automatically [1]. In recent years the focus has shifted from symbolic approaches, based on MIDI or score representations, to the analysis of musical audio files (e.g. [2 - 6]). Various systems have reached a reasonable level of success in labeling music according to the keys of the Western tonal system (cf. MIREX 2005 [7]).

Automatic analysis and classification of scales in music that is not organized according to the Western tonal system is much less developed. Some efforts that have been done to extract the scales of e.g. Australian aboriginal didjeridu music [8] or Indian classical music [9] use a reduction to Western pitch classes, thus avoiding the problems raised by irregular temperaments. Although this approach can be efficient to a certain extent, it seems limited to music with a pitch organisation that has a certain resemblance to the Western system and is problematic in terms of culture specific information. In some music the pitch-set used is as such not very important, but rather the musical gestures associated with playing or moving from one tone to another are the most characteristic aspects. This has been used in the study of Chinese guqin music [10] and Carnatic (South-Indian classical) music [11], using prototypical gestural patterns or melodic atoms to describe the melodic content of music in which the pitch is seldom stable.

Some work has been done on the scale analysis of music of the Middle-East, more precisely on Turkish [12] and Persian [13] modes. This music is characterized by the occurrence of intervals based on (roughly) a quarter tone scale. Therefore, an analysis based on a chromatic (half-tone) division of the octave can not be used. Therefore the pitch is analysed on a more continuous scale, then transformed to pitch histograms, which can be attributed to schemata that represent the modes used in this specific repertoire.

Pitch organisation in the music of Sub-Saharan Africa does not rely on a fixed theoretical framework. Ethnomusicological research has shown that a large variety of scales is in use. Often these scales use intervals that do not conform to the European chromatic scale, e.g. the use of intervals around 240 cents in (roughly) equidistant pentatonic scales [14]. However, standardized tuning systems or culture-specific classification systems do not exist. In this paper we will propose a system to explore African scales with applications in Music Information Retrieval and ethnomusicology. First we will present the collection on which the scale-detection system is applied, as well as the test-set which will be analyzed in detail. In the next chapter we give a brief description of the pitch detection and peak extraction systems used to analyze the music and how the output can be coupled with the metadata associated with the original sound files.

2. BACKGROUND

The audio set which has been used in this research, is a selection from the audio archive of the RMCA (Royal Museum for Central Africa) in Belgium. It is one of the largest collections worldwide of music from Central Africa. The audio collection consists of about 50,000 sound recordings (with a total of 3,000 hours of music), dating from the early 20th century up to now. Aiming for durable conservation and enhanced accessibility, the audio archive has been digitized entirely. Not only the audio but also accompanying metadata and contextual information have been digitized. A database and website were developed containing complete descriptions and fragments of the audio. The results of this project can be accessed on the website http://music.africamuseum.be.

For this study a selection of 901 audio files was used. In order to be sure to get a selection that consists only of music that uses a relatively fixed tone scale (and not e.g. music for percussion ensemble), we extracted music using four common types of musical instruments: musical bow (N = 132), zither (N = 134), flute (N = 385) and lamellophone (thumb piano) (N = 250). The selection was limited to music described as solo performances, mostly they contain only the sound of the instrument, in some cases the performer also sings, accompanying him/herself on the instrument.

3. ANALYSIS

3.1 Pitch detection

The pitch algorithm used in this paper has originally been designed to perform automated transcription of sung audio into a sequence of pitch classes and their duration [15]. Original goal of this tool was the development of a query-by-humming system for retrieving pieces of music from a digitized musical library. In this original system, the acoustic signal is turned into a parametric representation of the time-frequency information. A note is assigned to the segment by identifying the highest peak in the histogram of the frame-level pitch frequencies found in the segment, and by computing the average of the pitches lying in that bin. The pitch is then converted to a MIDI note rounding the computed annotation to the closest note frequency.

For the application of pitch recognition to the study of African scales, some important adaptations had to be made. First, the time segmentation, necessary to create melodies, was not of importance for building the pitch scales and was left out. Second, the quantization of the annotations into MIDI notes was unwanted, as we want to describe music that does not necessarily follows the equally-tempered scale. Therefore, the actual frequencies were used as pitch annotations. The output of this pitch algorithm consists of a list in which every line represents 10 ms, listing six potential frequencies, each with a probability. This allows extension to polyphonic textures.

In this case however, we choose to work with largely monophonic music. Therefore only the pitch with the highest probability was retained for every 10 ms, at least if this probability was higher than a minimal threshold (in this case 0.5). Then the frequencies were transformed to a cents scale, setting C0 to zero (cents). For comparing histograms, all listed values were reduced to one octave, generating a chromavector of 1200 values, representing the scale of the piece. A typical example of the graphical representation generated by the pitch detection system is shown in Figure 1.

3.2 Peak Extraction

The pitch analysis gives us a precise representation of the pitch content used in every piece. To extract the scale, a peak analysis was performed on the histograms. As a first step, the 1200 integers are ranked by their value. Starting from the highest value, peaks are assigned. A peak is accepted if it meets all parameters. Parameters for the selection are width of the area around the peak in which another peak cannot be assigned, the size (volume) of the peak and height relative to average height. Parameters were manually optimized for this data set by trial and error. As the histograms show a large variance (small and wide peaks, high and low peaks, high and low noise levels), a mean of the best individual settings was chosen as final parameter settings (see Table 1). The analysis gives us the number of peaks, average height and a precise description of the location and size of the individual peaks (Table 2).

Parameter	**Definition**	**Value**
Places	number of lines in the input file	1200
Peakradius	width of the peaks	30
Overlap	tolerated overlap between peaks	0.25
Accept	maximal proportion: volume peak without overlap/volume peak with overlap	25
Volfact	minimal volume of a peak: volfact*(average height of histogram)*(1+2*peakradius)	1
Heightfact	minimal height of a peak: heightfact*(average height of the histogram)	1

Table 1. Parameters used in the peak detection, with the settings used in the current analysis in the Reith column.

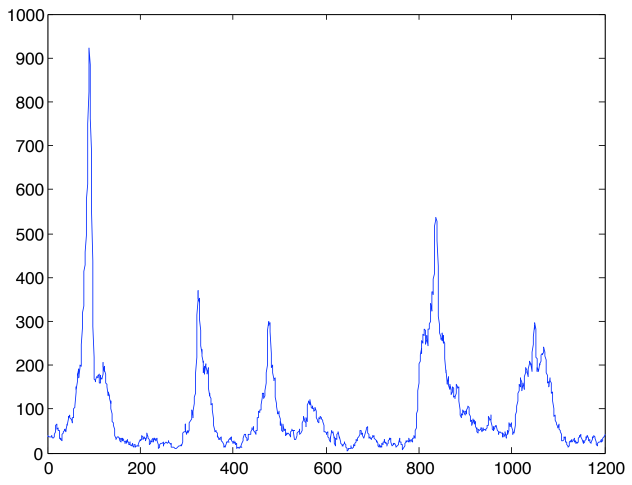


Figure 1. Example of the graphical output of the pitch analysis.

Peaks (cents)	volume	height	left side	% peak height	right side	% peak height
91	20441	922	61	0,15	121	0,21
837	17953	538	807	0,47	867	0,29
325	9603	371	295	0,13	355	0,29
476	8305	301	446	0,17	506	0,16
1050	12313	296	1020	0,57	1080	0,52

Table 2. Example of the output of the peak analysis for the piece shown in Figure 1, showing the pitches, together with information on the size of the peak.

3.3 Metadata

All the meta-data that were originally associated with the collection were digitized. Thus we get a large number of data fields from different categories: identification (number/id, original carrier, reproduction rights, collector, date of recording, duration), geographic information (country, province, region, people, language), and musical content (function, participants, instrumentation). Unfortunately, not for every recording all fields are available and often these data cannot be traced, as a large part of the collection is made up of unique historical sources.

The results of the pitch and scale analysis can be coupled with existing meta-data such as instrumentation, geographical information or date of recording. This can give us valuable information on the use of certain scales, such as geographical spread or evolution through time. As the current selection of pieces is relatively small, we used broad categories for the geographical origin (West-Africa, Southern Africa,…) and the recording time (before 1960, between 1960 and 1975, after 1975). An example of such a coupling is given in Figure 2. It gives the amount of peaks for each piece for each of the three time periods. This shows that in recent recordings, hexatonic and heptatonic scale become relatively more important while the importance of pentatonic and tetratonic scales diminishes.

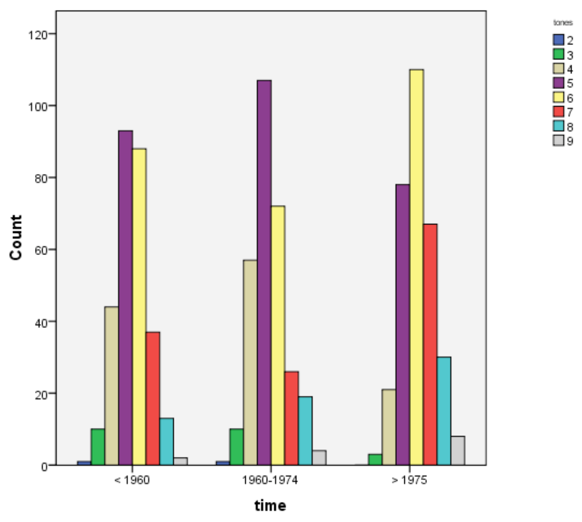


Figure 2. Bar chart representing the amount of peaks (2-9), for each the three categories of recording time: before 1960 (n = 288), between 1960 and 1975 (n = 296) and after 1975 (n = 317).

4. APPLICATIONS

The analyses made can be applied in different areas. First we will show the application of the pitch detection for data-mining applications, using cross-correlation of the pitch profiles to look for similarities. Next we will show an application of the techniques for ethnomusicological research, studying the intervals used in African scales.

4.1 Correlation analysis

The chromavectors given by the pitch analysis can be cross-correlated with each other in order to search for similar scales. As African music doesn't have a standardized tuning pitch, we need to allow a shift of pitch. Therefore, the cross-correlation technique uses every permutation of the original chromavector and returns the highest correlation from a list of 1200 correlations together with amount of cents it had to be shifted (Figure 4). Thus this method can be used for a query-by-example in which the output is a list of pieces with similar scales. This application allows to retrieve a song from a database without knowing any concrete fact about it, which is an important element for the usability of a search engine in a database of largely unknown music.

Next to this, the method can be useful for database management. The technique allows to check whether some songs are already present in their archive (so called double listed items), looking for perfect correlations without pitch shift. It can also help to establish groups of pieces with a similar origin, detecting possible links between recordings from different origins (cf. Figure 3). This could eventually lead to determination of missing meta-data.

Although the results of this analysis are promising, still some optimizations have to be done. Thus e.g. noisy pitch profiles with broad peaks, indicating less stable pitches (e.g. from singing) (cf. Figure 3) are more likely to generate high correlations compared to pieces with very clearly defined pitches. Similarly the larger the number of peaks the more difficult it gets to obtain high correlations. Some mechanisms to deal with these differences should still be developed.

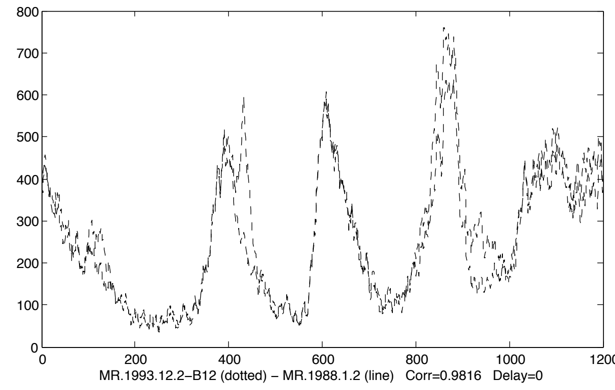


Figure 3. Graphical representation of a query-by-example, in this case the correlation is very high (r = .98) and no shift in pitch is necessary, which could point to a similar origin, despite the different sources.

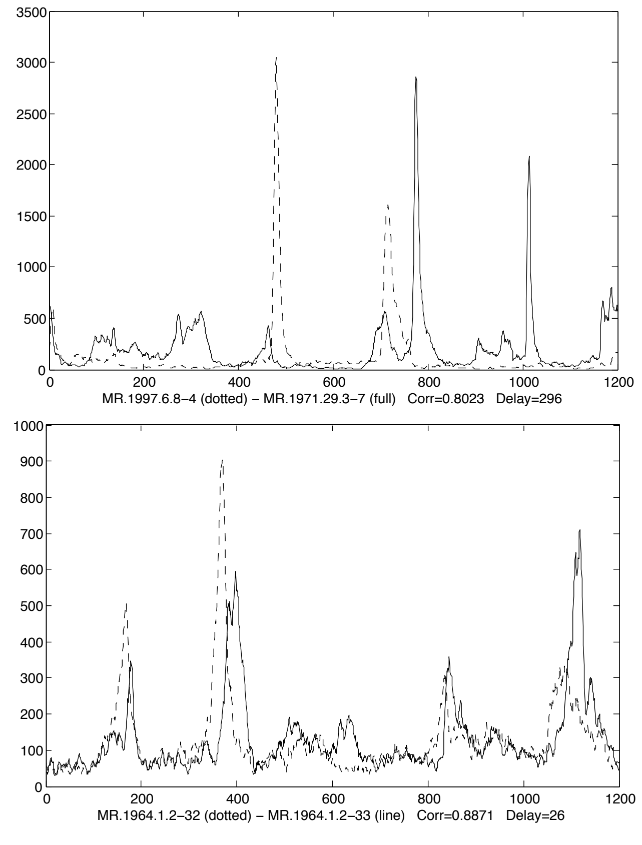


Figure 4. Two examples of a cross-correlation analysis, where the optimal correlation is found through a pitch shift. In the upper example a relatively large shift of 296 cents (about a minor third) reveals the highest similarity (r = .80), while in the lower example only a small shift of 26 is necessary to obtain the maximum (r = .89).

4.2 Interval analysis

In the analysis of 20th century Western classical music, so-called 'interval vectors' are used to express the intervallic content of a pitch-class set [16]. Using a Western chromatic scale, interval vectors are limited to an array of six numbers, expressing the amount of occurrences of each possible pitch interval (from a minor second to a tritone). With the variety of intervals found in African scales, this reduction to six numbers is not possible. Nevertheless, creating a global view on the intervals that constitute the scales can give us interesting insights in the pitch structure of the music. Are there for example any specific intervals that occur often, can we see regional differences or is there an evolution through time.

For this analysis the scales obtained from the peak analysis are transformed to an array of all possible intervals that can be built with this scale. As we work with scales reduced to one octave, the distinction between rising and falling intervals can not be made. Therefore the maximum interval size is set at 600 cents (a tritone or half an octave). For the analysis presented here, the intervals were grouped in bins of 5 cents, which gives us interval vectors of 120 elements.

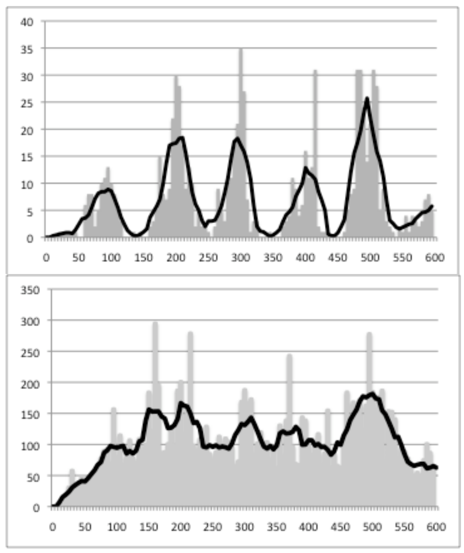


Figure 5. Comparison of all the pitch intervals found in the scale analysis from (above) the 42 pieces from the J.S. Bach's six cello suite and (below) our collection of 901 pieces of African music.

First we can make a global analysis of the intervals. In figure 5, a comparison is made between our complete collection of 901 pieces and a small sample of Western tonal music (Johann Sebastian Bach's six cello suites, played by Mstislav Rostropovich, a collection of 42 movements). In the interval analysis of the Western music we clearly see peaks corresponding to the standard intervals of 100 cents. For the African music the situation is much less clear. One similarity could be the importance of the 500 cents intervals (corresponding to the pure fourth/fifth), but the other peaks are much less well-defined and in some cases sharp peaks appear at odd intervals (e.g. 160, 370 cents).

Now we can also couple the interval vectors with the meta-data. As an example we can look if we can find some differences in interval content between the three time periods. The analysis of the meta-data already revealed that tone scales with larger number of pitches became more important in recent recordings (cf. supra). Do we also find an influence on the pitch content? All three interval profiles are very irregular and show peaks at unexpected places, as seen in the global analysis. An interesting evolution is seen if we look at the relative share of the intervals corresponding to the Western equally-tempered scales. Counting the relative share of the 5 relevant intervals by taking the two bins around the correct interval (e.g. 95-105 cents for the minor second), we see that the share of these intervals almost doubles in the recent recordings (Table 3). Only for the minor third we don't see an increase, and the change is especially remarkable for the major seconds (also containg the minor sixths) and the pure fourths/fifths. A detailed view on the area in which pure fourths and fifths are found reveals an interesting evolution (Figure 6). The main peak seems to shift from 530 cents in the earliest recordings to 515 cents in the middle period to end up at 500 cents in the most recent recordings. This possibly also indicates a gradual evolution to a Western pure-fifth based tuning.

Interval	**< 1960**	**1960-1975**	**> 1975**
min. 2nd	1,46	1,87	2,20
maj. 2nd	1,57	1,71	5,20
min. 3rd	2,26	3,39	2,84
maj. 3rd	1,25	1,28	2,78
4th/5th	2,55	2,56	5,31
sum	**9,10**	**10,81**	**18,33**

Table 3. Relative share (in %) of pitches in an area of 10 cents around the Western equally-tempered intervals, for three recording time periods.

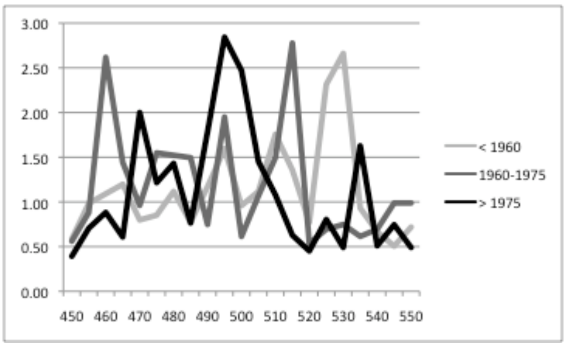


Figure 5. Relative share (in %) of pitches in bins of 5 cents between 450 and 550 cents for three recording time periods.

A detailed analysis of regional differences goes beyond the scope of this paper. Yet, we can see some interesting elements in relation to the global analysis of intervals. We see for example that the peak at 160 cents is present in every region. This shows that it is not a feature of a particular culture, but a 'pan-African' characteristic. Further ethnomusicological work is necessary to find a possible explanation for the importance of this interval. Interestingly similar interval sizes are found in the music of the Middle-East, where they are classified as 'neutral seconds' (neither minor nor major, but in between). The pitch system there however is organized according to completely different principles, so it is not clear if a direct link can be established.

5. DISCUSSION AND CONCLUSIONS

We proposed a number of methods to deal with non-standardized tone-scales, as they are found in African music. Avoiding working with a priori determined categories (such as the pitches of the chromatic scale), allows a representation and analysis of a large variety of tone scales. This was illustrated by a sample of solo-music on four different instrument types taken from the archive of the Belgian Royal Museum of Central-Africa. The results are promising, both for data-mining application and as a starting point for ethnomusicological research. Before we can expand these techniques to the whole database, several problems still have to be solved. Some important obstacles for are the presence of unaccompanied vocal music, which usually has a fluctuating pitch. This makes it very hard to extract the exact scale automatically, without applying a kind of pitch correction first. Also there is a problem with the use of percussion. The presence of percussive sounds tends to obscure the actual pitch scale used and to generate one large peak associated with the pitch of the percussion instrument. Therefore a system to suppress these percussive sounds should also be developed.

Using this relatively small sample of 901 pieces, we could already develop some methods for ethnomusicological research, creating a more elaborate view on scales and temperaments in African music in an automated way. A global comparison between the intervals found in Western and in African scales, shows that African music does not conform to the fixed chromatic scale nor has another fixed scale, however in recent recordings there seems to be a tendency to the use of more elaborate, equally-tempered scales. Further research has to be done in these historical aspects as well as on the geographical aspects of African tone scales. These techniques lead to usable applications for query-by-example, database management and classification.

6. REFERENCES

[1] H. C. Longuet-Higgins & M. J. Steedman: "On interpreting Bach," *Machine Intelligence,* vol. 6, pp. 221–241, 1971.

[2] M. Leman: *Music and Schema Theory*, Berlin, Springer Verlag, 1995.

[3] S. Pauws: "Extracting the Key from Music," in W. Verhaegh, E. Aarts & J. Korst (eds.) *Intelligent Algorithms in Ambient and Biomedical Computing*, pp. 119–132, 2006.

[4] Ö. Izmirli: "Localized Key Finding from Audio Using Nonnegative Matrix Factorization for Segmentation," in *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR2007),* 2007.

[5] C. Chuan & E. Chew: "Audio key finding: considerations in system design and case studies on Chopin's 24 preludes," *EURASIP Journal on Applied Signal Processing*, Vol. 2007, No. 1, 15 pp., 2007.

[6] E. Gomez: *Tonal Description of Music Audio Signals*, Ph.D. Thesis, Universitat Pompeu Fabra, Barcelona, 2006.

[7] J. S. Downie, K. West, A. Ehmann & E. Vincent: "The 2005 Music Information Retrieval Evaluation eXchange (MIREX 2005): Preliminary Overview", in *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005),* pp. 320-323, 2005.

[8] A. Nesbit, L. Hollenberg & A. Senyard: "Towards Automatic Transcription of Australian Aboriginal Music," in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pp. 326-330, 2004.

[9] P. Chordia, M. Godfrey & A. Rae: "Extending Content-Based Recommendation: the Case of Indian Classical Music," *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR2008),* pp. 571-576, 2008.

[10] H. Li & M. Leman: "A Gesture-baseed Typology of Sliding-tones in Guqin Music," *Journal of New Music Research*, Vol. 36, pp. 61-82, 2007.

[11] A. Krishnaswamy: "Multi-dimensional Musical Atoms in South-Indian Classical Music," *Proceedings of the 8th International Conference on Music Perception and Cognition (ICMPC8)*, 2004.

[12] B. Bozkurt: "An Automatic Pitch Analysis Method for Turkish Maqam Music," *Journal of New Music Research*, Vol. 37, pp. 1-13, 2008.

[13] P. Heydarian, L. Jones & A. Seago: "The Analysis and Determination of the Tuning System in Audio Musical Signals," *Paper presented at the 123rd convention of the Audio Engineering Society*, 5 pp., 2007.

[14] G. Kubik: *Theory of African Music*, Willemshaven, F. Noetzel, 1994.

[15] L.P. Clarisse, J.P. Martens, M. Lesaffre, B. De Baets, H. De Meyer & M. Leman: "An Auditory Model Based Transcriber of Singing Sequences, " *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR2002),* pp. 116-123, 2002.

[16] A. Forte: *The Structure of Atonal Music*, Yale University Press, 1973.

A DISCRETE FILTER BANK APPROACH TO AUDIO TO SCORE MATCHING FOR POLYPHONIC MUSIC

Nicola Montecchio, Nicola Orio
Department of Information Engineering
University of Padova
{nicola.montecchio,nicola.orio}@dei.unipd.it

ABSTRACT

This paper presents a system for tracking the position of a polyphonic music performance in a symbolic score, possibly in real time. The system, based on Hidden Markov Models, is briefly presented, focusing on specific aspects such as observation modeling based on discrete filterbanks, in contrast with traditional FFT-based approaches, and describing the approaches to decoding. Experimental results are provided to assess the validity of the presented model. Proof-of-concept applications are shown, which effectively employ the described approach beyond the traditional automatic accompaniment system.

1. INTRODUCTION

The concept of *audio to score alignment* refers to the ability of a system to align a digital audio signal recorded from a music performance with its score. More precisely, given a recording of a music performance and its score, the aim of such alignment system is to match each sample of the audio stream with the musical event it belongs to. There are a number of possible applications of such technology, ranging from the “automatic accompanist”, a software allowing solo players to practice their part while the computer plays the orchestral accompaniment, to tools for musicological analysis or augmented audio access.

Most systems currently used for audio to score alignment are based on statistical models. In particular Hidden Markov Models (HMMs) [1, 5], possibly with hybrid approaches that make use of Bayesian networks and HMMs [8] or Hidden Hybrid Markov / semi-Markov chains [3].

In this paper we propose an HMM-based system that focuses on handling highly polyphonic music through the use of a filterbank approach.

2. MODEL DESCRIPTION

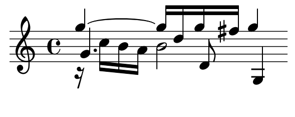
The main idea of the proposed approach is that the most relevant acoustic features of a music performance can be modeled statistically as observations of a Hidden Markov Model (HMM). The process of performing a music work can be regarded as stochastic because of the freedom of interpretation, yet the knowledge of the work that can be obtained from the score can be exploited to model the possible performances. In the presented system, a HMM is built according to the data contained in the music score. The incoming audio signal is divided into frames of fixed length, with every frame corresponding to one time step of the HMM; the HMM performs a transition every time a new audio frame is observed and the advancement of the performance in the score is tracked by performing the decoding of the HMM. The crucial point is the definition of the graph topology and the observation modeling while decoding is performed with well-known algorithms.

2.1 Score Graph Modeling

The score modeling step aims at obtaining a graph structure representing the music content of the score. In particular, a score is represented as a sequence of events, implying that it can be transformed into a simple graph where states are connected as in a chain. Two levels of abstraction can be distinguished in the resulting graph: a *score level* modeling the macro-structure of the piece, that is the sequence of music events, and an *event level* dealing with the structure of each music event; the distinction between the two reflects the conceptual separation between different sources of mismatch: the former deals with possible errors both by the musicians and in the score, while the latter models the duration and the acoustic features of each event, which vary depending on interpretation, instrumentation, recording conditions and other factors.

2.1.1 Score Parsing

The first step in building the HMM graph is the transformation of the symbolic score into a sequence of events. In the case of a monophonic score, all the notes and explicit rests correspond to an event, while events in a polyphonic score are bounded by any single onset and offset of all the notes that are played by the various instruments/voices (see Figure 1). Due to the large availability of already transcribed music, MIDI has been used as the score representation format although, being provided by end users, most of the MIDI files contain transcription errors that may influence the alignment effectiveness.



(a) Original score

event	0	1	2	3	4	5	6	7	8
notes	G5	G5	G5	G5	G5	D5	G5	F#5	G5
		C5	B4	A4	B4	B4	B4	B4	B4
	G4	G4	G4	G4	G4	G4	D4	D4	G3
duration	1	1	1	1	1	1	1	1	4

(b) Event sequence

Figure 1. Score representation

2.1.2 *Graph Topology – Score Level*

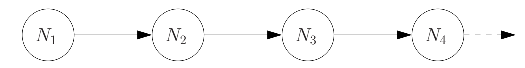
In its simplest form, the topology of the score level graph directly represents the succession of events: the states, each corresponding to a single music event, form a linear chain, as seen in Figure 2(a). This approach has no explicit model for local differences between the score representation and the actual performance that has to be aligned, thus the overall alignment can be affected by local mismatches. For instance, a skipped event, which should create only a local misalignment, can extend its effect also when subsequent correct events are played resulting in larger differences in the alignment.

In order to overcome these problems, a special type of states was introduced, namely *ghost states* – as opposed to *event states*, which correspond to real events in the music work. Ghost states were proposed in [4]. The basic graph topology is modified so that each event state can perform a transition to an associated ghost state, which in turn can perform either a self-transition or a forward transition to subsequent event states. The final representation is made of two parallel chains of nodes, as shown in Figure 2(b). This approach can model local differences between the score and the performance, because in this case the most probable path can pass through one or more ghost states during the mismatch and realign on the lower chain when the performance matches again the score. The transition probabilities from event states to corresponding ghost states are typically fixed, while the transition probabilities from a ghost state to subsequent event states follow a decreasing function of distance: this resembles the idea of *locality* of a mismatch due to an error.

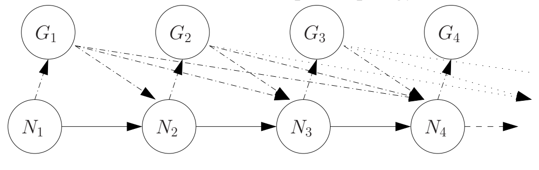
2.1.3 *Graph Topology – Event Level*

The event level models the expected acoustic features of the incoming audio signal. Every state of this level is modeled as a chain of n *sustain states*, each having a self-loop probability p, possibly followed by a *rest state*, as shown in Figure 2(c). Sustain states model the features of the sustained part of an event, while rest states model the possible presence of silence at the end of each event that can be due to effects such as staccato playing style. As described in [7], the probability of having a segment duration d is modeled by a negative binomial distribution, with expected value $\mu = \frac{n}{1-p}$ and variance $\sigma^2 = \frac{np}{(1-p)^2}$. The duration of an event is modeled by setting the values of n and p accordingly; in particular μ is set equal to the event duration in the score.

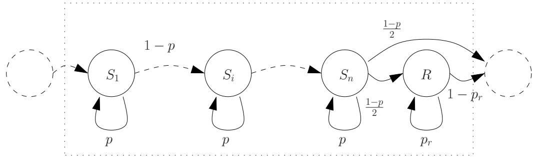
Two cases can be distinguished depending on the choice of having n fixed or variable. In the former case event duration is modeled by self-loop probability. This approach is easy to implement and with a small n the total number of states in the graph is relatively small and proportional to the number of events; on the other hand the variance of the distribution changes with events duration. The latter case allows for a more precise modeling of event duration. It is reasonable to compute n and p in order to have $\sigma^2 = k\mu$, where p is constant for all the events, and the only parameter responsible for the event duration is the number of sustain states, of which the total number is thus proportional to the duration of the score.



(a) Score level (simplest topology)



(b) Score level, with ghost states



(c) Event level

Figure 2. Graph topologies

2.2 Modeling the Observations

The fundamental assumption of the model is that states of the event level emit the expected acoustic features of the incoming signal. Because polyphonic pitch detection is still unreliable, the signal itself is not analyzed, instead its harmonic features are compared to the expected features of the emissions of the HMMs.

2.2.1 *Sustain States*

The core feature used by the observation modeling of sustain states is the similarity, for each audio frame, between the spectrum of the incoming signal and an ideal spectrum of the sustain state that is being considered. Sophisticated techniques have been proposed making use of specific knowledge of instrument timbre [2]. Although very effective in specific situations, such as contemporary music performances where the instruments can be sampled, this kind of approach is not suitable for the general case where the instrument cannot be known in advance from the score.

Typically, spectrum analysis is done via the Fast Fourier Transform: the energies for the various frequency bands are computed by summing the energies in the appropriate FFT bins. The problem with this approach is that the linear frequency resolution of the FFT leads to a significant loss of precision in the lower frequency range. While the situation is partially compensated by upper harmonics a different strategy can nevertheless improve the performances of a system.

In our approach, the frequency resolution problem is handled using a bank of discrete filters. In particular, each one is a second order filter of the form

$$H_i(z) = \frac{(1-r_i)\sqrt{1-2r_i\cos(2\theta_i)+r_i^2}}{(1-r_ie^{-j\theta_i}z^{-1})(1-r_ie^{j\theta_i}z^{-1})} \qquad (1)$$

which has unit gain at θ_i (the normalized nominal frequency of the i-th note), and allows, by changing the pole radius r_i, to set the filter bandwidth; each filter output is then routed to a delay line in order to compensate for the different group delays: assuming that each filter has the same bandwidth in semitones, the filters corresponding to the lowest notes have a much higher group delay than the highest ones. We assume that this delay, which can be removed off-line or compensated in real time applications, is to be preferred to a lack of frequency resolution for lower notes. A comparison of FFT and Filterbank analysis is presented in Section 3.3.

The observation probability of a note is computed by partitioning the spectrum into frequency bands, with each band corresponding to a note in the music scale. Let E_i^f be the energy of the i-th filter output signal in the current frame, i.e. $E_i^f = \sum_t y_i^2(t)$; the energy E_i^n corresponding to the i-th note can be defined as

$$E_i^n = \sum_j E_{i+h(j)}^f \qquad (2)$$

where $w_j = 1$ and $h(j)$ is a simple map between the index of a harmonic and the corresponding note index. In this very simple instrument model, the energy for the note C3 is computed as the sum of the energies for the filters corresponding to the notes C3, C4, G4, C5, E5, and so on.

The observation probability for the i-th sustain state is computed as

$$b_i^{(s)} = F(\frac{E_i^n}{E_{\text{tot}}}) \qquad (3)$$

where E_i is the energy in the expected frequency bands and E_{tot} is the total energy of the audio frame. $F(\cdot)$ is the unilateral exponential probability density function

$$F(x) = \frac{e^\lambda}{e^\lambda - 1}\lambda e^{\lambda(x-1)} \qquad 0 \le x \le 1 \qquad (4)$$

Other similarity functions can be applied with similar results, in particular the cosine distance between the vector representations of the simple instrument model used to compute E_i and the filter output energies.

While the above approach is robust enough for monophonic alignment, the complexity of polyphony makes it preferable to apply a different weighting of the harmonics in the instrument model. A possible solution is to modify Equation 2 by adding decreasing weights to the note harmonics to reflect a more realistic instrument model. When filters overlap for some harmonics of different notes, the weight assigned to that harmonic in the instrument model can be either the sum or the maximum of the individual weights; the latter solution seems to perform better, and the intuitive explanation is that typical scores do not contain precise information about the loudness of each note/part, so a simpler model is more general.

2.2.2 *Rest States*

The observation probability for the i-th rest state is computed as a decreasing function of the ratio of the current audio frame energy over a reference threshold representing the maximum signal energy.

$$b_i^{(r)} = F(\frac{E_{\text{tot}}}{E_{\text{thres}}}) \qquad (5)$$

The threshold is adaptive, to compensate for possible differences in the overall recording volume of different input streams.

2.2.3 *Ghost States*

A simple approach for modeling the observations of ghost states is to assign a fixed value to the observation probabilities, because these states are meant to provide a sort of "emergency exit" for local matches. The approach can be improved by computing the observation probability for the i-th ghost state as:

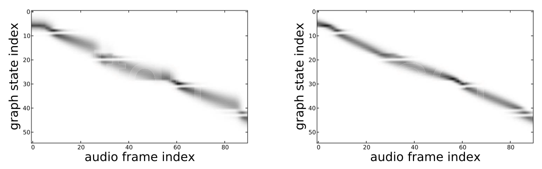
$$b_i^{(g)} = \sum_{j=i}^{i+k} w_i(j) b_j^{(s)} \qquad (6)$$

that is, a weighted sum of the sustain observation probabilities of the following event states, where $w_i(\cdot)$ is a decreasing discrete distribution function and its presence is motivated by the fact that, intuitively, in case of wrong or skipped notes, the notes actually played would probably be close to the expected ones. In case of errors in the score, the weighting function induces the system to quickly realign on near notes.

2.3 Decoding Strategies

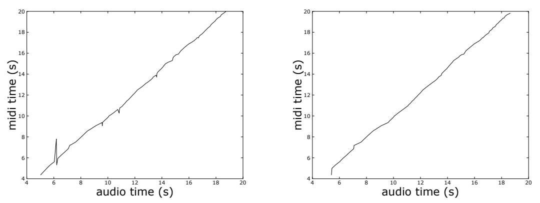
The proposed system exploits the decoding algorithms described in [6], depending on the application context, namely *forward decoding* and *forward-backward decoding*. These strategies determine, at each time interval, the *most probable state*, without forcing the decoded sequence of states to actually be the *most probable sequence* of states as is the case for *Viterbi decoding*. Preliminary tests showed that the system recovers more quickly, because the decoded sequence does not need to be a feasible state sequence.

Figure 3 compares a typical evolution of the state probabilities for the forward and forward-backward decoding algorithms. The latter is characterized by a more precise evolution, a highly desirable behavior in the case of subsequent events with the same set of harmonics: if no modeling of a note attack is employed – as is the case with the current version of the system – and the rest states at the end of the lower level event chain do not help discriminating the events, the evolution of forward-backward decoding automatically assigns to the events a duration in the alignment which is proportional to the duration in the score.



(a) Forward decoding (b) Forward-backward decoding

Figure 3. Evolution of state probabilities



(a) Forward decoding (b) Forward-backward decoding

Figure 4. Typical alignment evolution

3. EXPERIMENTAL RESULTS

The evaluation of an audio to score alignment system is a difficult task, mainly because of the lack of a manually aligned test collection of polyphonic music. For instance, the MIREX test collection is not publicly available because of copyright reasons and it contains mainly monophonic recordings. For this reason, two test collections have been prepared, the former made up of single-instrument polyphonic pieces and chamber music and the latter comprising excerpts of more complex orchestral works. A experimental comparison of the FFT and Filterbank analysis approaches is presented using recordings of tuba and cello music, characterized by a low frequency content.

3.1 Single Instrument and Chamber Music Collection

The audio collection is made up of excerpts from well known piano, violin, and chamber music works [1] extracted from CD and home recordings; the MIDI files were downloaded from the Internet. The files in the collection have been chosen so that the complexity of their polyphony is representative of pieces which could be realistically used in a typical automatic accompaniment system, with real time requirements. The resulting alignments were manually checked, visually inspecting the mismatches and aurally verifying them by listening to a stereo recording containing the original piece and a synthesized version generated from the alignment data on different channels.

Out of 20 test recordings, none caused the system to get lost, but in one case the alignment was very unstable (it was always in proximity of the “true” alignment but never precise) so its contribution will not be considered. For the other recordings the mismatches were classified according to their duration as either brief (shorter than two seconds) or long (larger time intervals, although never more than 10 seconds); the former type of mismatches occurred 41 times while the latter 10 times, mostly on complex passages of polyphonic material. Example alignments can be viewed and heard in the authors' home pages [2], where more detailed statistics can also be found.

Because of the real time requirements, the forward decoding algorithm was used to compute the alignments. If real time is not a constraint, usually forward-backward decoding gives better results, in which many of the glitches in the forward-decoded alignment are eliminated. Such an example is shown in Figure 4.

All the alignments have been performed using the same model parameters; further experiments showed that some improvements can be obtained by assigning different weights to the harmonics in Equation 2 for piano and string works. Essentially, the different weighting reflects the suitability of a more refined instrument model, in particular the piano model is characterized by more rapidly decaying overtones than the string model.

3.2 Orchestral Music Collection

The orchestral music collection comprises 48 excerpts of 40 seconds from CD recordings of symphonic works [3] ; the MIDI scores are generally much less accurate than the ones used in the chamber music collection.

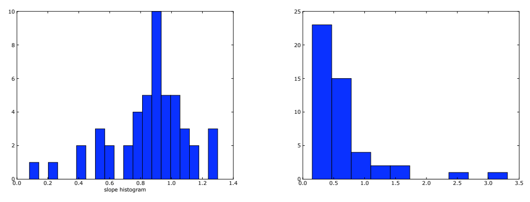
A simple evaluation methodology was devised in order to present results for this collection. The output of the alignment system for a single performance/score couple is a list of value pairs in the form [*audiotime*,*miditime*]. Once all the performances in a collection are aligned to their corresponding score, these alignments are analyzed to extract a measure of precision based on the average deviation of the alignment data from the best fitting line. This measure is based on the hypothesis that an orchestra plays more or less *a tempo*, at least in short time intervals, thus a graphic representation of the alignment should follow a straight line. While this is clearly a potentially incorrect assumption, the suitability of the particular performances in the test collection was verified by the authors. The best fitting line computed from the alignment data is thus assumed to be the correct alignment; Δ_{avg} is defined as the average deviation of the alignment data point from the best fitting line. Under the assumption of a performance characterized by a steady tempo, the lower is Δ_{avg} the higher is the alignment accuracy. This evaluation methodology was not used for the chamber music collection because the tempo was not steady enough.

Figure 5 shows the histograms of the slope and Δ_{avg} distributions for the best fitting lines obtained from the alignments. The tempo of the recorded performances and of the respective MIDI files are roughly comparable, so

[1] Bach: Italian Concerto, Goldberg Variations, Chaconne from the Violin Partita in D minor; Beethoven: Piano Sonata op. 13, String Quartet op. 18 n. 1; Mozart: Piano Sonata KV333; Ravel: String Quartet; Schubert: Quartettsatz D703; Schumann: Waldszenen op. 82.

[2] `http://www.dei.unipd.it/~montecc2/ismir09/`

[3] Beethoven: Symphonies n. 3, 7, 9; Haydn: Symphony n. 104; Mendelssohn: Symphony n. 4; Mozart: Symphonies and Serenades K136, K412, K525, K550; Vivaldi: The Four Seasons.



(a) Slopes histogram (b) Δ_{avg} histogram

Figure 5. Orchestral collection alignment results

the expected histogram of the slopes should be centered around 1; an alignment can thus be safely considered incorrect when slope values are outside the interval $(0.6, 2)$. This simple assumption allows to quickly interpret the graphical results and deduce that the performance of the system with orchestral music is, as expected, clearly worse than the case for single instrument or chamber music, in which all alignments were essentially correct. Manual inspection of the results showed that the correct alignments were 36; for those, the average Δ_{avg} was 0.47.

A closer analysis pointed out that in the correct and incorrect sets of alignment the elements are homogeneous with respect to the music work, e.g. all Vivaldi's and most of Mozart's music was correctly aligned while most of Beethoven's were not. The reason for this was found out to be the fact that in the recordings of Beethoven's works the reference pitch was slightly higher than the standard 440 Hz for A4; correcting this setting considerably improved the results for Beethoven's music. This situation is a clear example of how a single set of parameters is not suitable for all the possible situations, but this is typically not a requirement: in the offline case multiple alignments can be performed and only the best one, according to the simple heuristics discussed above, can then be presented to the user, while when real time is required, it is reasonable to assume that the system parameters can be adjusted using previous rehearsals as reference.

In the above results, the forward decoding algorithm was used to compute the alignments; the reason is that the forward-backward algorithm turned out to be less robust for aligning performances where an alignment computed with forward decoding was not precise.

3.3 Comparison of FFT and Filterbank analysis

Several experiments were performed on a small collection of recordings of tuba and cello music, to show the advantages of discrete Filterbank analysis over traditional FFT for observation modeling on music characterized by a low frequency content. The recordings were aligned manually in order to count the number of wrongly recognized or skipped notes. Of 105 total events, the FFT based system did not recognize 12 and skipped 1, while the Filterbank based system did not recognize only 4 notes and skipped none. It should be noted that in almost all cases of not recognized notes both system realigned on the correct note immediately, and that the parameters of the systems were not tuned for this particular situation, so that better performances can be expected; forward decoding was used to simulate a real-time operation. The alignments of the worst performing recording are shown in Figure 6.

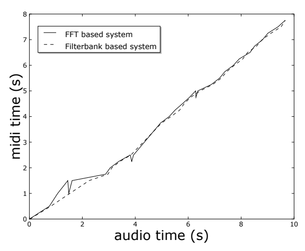


Figure 6. Comparison of FFT and Filterbank approaches

4. APPLICATIONS

Two applications are presented that make use of audio to score alignment technology for music analysis tasks.

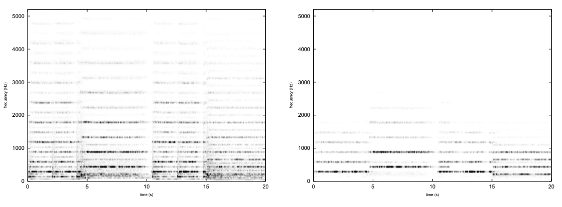
4.1 AudioZoom

AudioZoom is a software for the auditory highlight of single instruments in a complex polyphony. The basic idea is that the alignment can help dividing a polyphonic music performance into its individual components: the general problem is known as *source separation*, which is usually defined *blind* when it is assumed that almost no information is available about the role of each source. In our case, having the score as a reference, the system has a complete knowledge about the notes played, at each instant, by all the instruments. The user, typically a teacher who may exploit this tool to highlight particular instruments or passages to students that are not able to follow a complex score, can select one or more instruments, one or more particular musical themes or patterns, or any combination, and the system can selectively amplify the chosen elements.

The final effect is to put on the front, or zooming, the interested elements. A prototype of *AudioZoom* has been developed, based on a bank of bandpass filters centered around the harmonics of a selected instrument, using an approach similar to the instrument model described in Section 2.2.1. The user selects one channel from the MIDI file that represents the score, and the system aligns the different filterbanks with the audio recording. An example of the effect of AudioZoom, applied to the viola part of the beginning of Haydn's Symphony n. 104, is shown in the sonograms of Figure 7.

4.2 Interpretation Analysis

Analyzing different interpretations of a music work is a central activity of musicological analysis. Of all the features that characterize a personal rendition, tempo is probably the most perceivable one. The alignment of two audio performances allows to compare the relative tempos, but neither can be considered as a reference since no interpretation can be neutral. It can be noted that the concept of neutral interpretation is itself not well defined.



(a) Original excerpt (b) Zoomed excerpt

Figure 7. Effect of *AudioZoom*

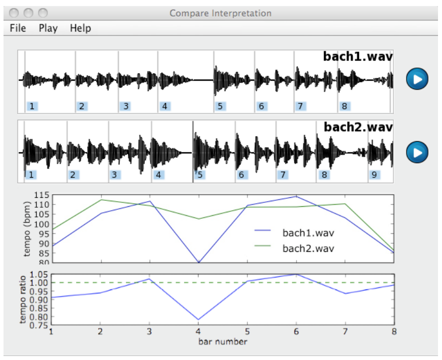


Figure 8. Different interpretations of the same piece

The alignment of two interpretations to the score allows a musicologist to draw some considerations on the different interpretations, for instance by comparing the instantaneous tempo at each bar. Figure 8 shows an early prototype of a tool for the comparison of different performances, in which two interpretations of the beginning of J. S. Bach's Italian Concerto are juxtaposed using the measures in the score as a reference. Clearly, the prototype can be extended by representing the differences in loudness, the use of *accelerandi* and *rallentandi* or more complex features related to timbre perception.

5. CONCLUSIONS AND FUTURE WORK

A system is proposed for the alignment of an audio performance with a score. The system is based on the use of filterbanks to extract pitch related information from the performances. Comparative evaluations with previous versions of the system showed that observation modeling based on discrete filterbanks has some advantages with respect to the simpler FFT approach, resulting in higher effectiveness. In general, evaluation showed that the approach can be effectively applied to real application scenarios; many areas however can be improved, and below we propose some research directions which seem the most promising.

A clear priority is the creation of a collection which comprises precise manual alignments, in order to properly evaluate the effectiveness of the approach but also to train the model parameters in a rigorous way. This is a very time-consuming task, requiring music experts and specific annotation tools for properly marking the matches between the events in the scores and the corresponding time instants in the recordings. The only viable solution in our opinion is to involve other research teams in building a shared collection of reasonable size; such collaborative effort would also help in devising appropriate data and evaluation methodologies for alignment system. A good starting point is the collection used for the MIREX campaigns, which should be improved adding polyphonic scores and a clearer time reference for the alignment evaluation.

The introduction of a refined modeling for the attack of notes is desirable for many instruments with percussive attacks – in particular the piano – to better handle repeated notes, but with the appropriate decoding strategies this issue is not critical. Another improvement regards the modeling of complex events, such as trills or *glissandi*, which are hard to extract from MIDI files, resulting in potentially less effective models.

6. ACKNOWLEDGMENTS

The work has been partially supported by a grant of the University of Padova for the project "Analysis, design, and development of novel methodologies for the study and the dissemination of music works".

7. REFERENCES

[1] P. Cano, A. Loscos and J. Bonada. Score-Performance Matching using HMMs. In *Proceedings of the International Computer Music Conference*, pp. 441-444 1999.

[2] A. Cont. Realtime Audio to Score Alignment for Polyphonic Music Instruments Using Sparse Non-Negative Constraints and Hierarchical HMMs. In *IEEE International Conference in Acoustics and Speech Signal Processing*, pp. V245–V248, 2006.

[3] A. Cont. Modeling Musical Anticipation: From the Time of Music to the Music of Time. PhD. thesis, 2008.

[4] N. Montecchio and N. Orio. Automatic Alignment of Music Performances with Scores Aimed at Educational Applications. In *Proceedings of the International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution*, pp. 17–24, 2008.

[5] N. Orio and F. Déchelle. Score Following Using Spectral Analysis and Hidden Markov Models. In *Proceedings of the International Computer Music Conference (ICMC)*, pp. 125-129, 2001.

[6] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, 77(2):257-286, 1989.

[7] C. Raphael. Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):360–370, 1999.

[8] C. Raphael. Aligning Music Audio with Symbolic Scores using a Hybrid Graphical Model. *Machine Learning*, 65:2(389–409), 2006.

ACCELERATING NON-NEGATIVE MATRIX FACTORIZATION FOR AUDIO SOURCE SEPARATION ON MULTI-CORE AND MANY-CORE ARCHITECTURES

Eric Battenberg
Parallel Computing Laboratory
University of California, Berkeley
ericb@eecs.berkeley.edu

David Wessel
Center for New Music and Audio Technologies
University of California, Berkeley
wessel@cnmat.berkeley.edu

ABSTRACT

Non-negative matrix factorization (NMF) has been successfully used in audio source separation and parts-based analysis; however, iterative NMF algorithms are computationally intensive, and therefore, time to convergence is very slow on typical personal computers. In this paper, we describe high performance parallel implementations of NMF developed using OpenMP for shared-memory multi-core systems and CUDA for many-core graphics processors. For 20 seconds of audio, we decrease running time from 18.5 seconds to 2.6 seconds using OpenMP and 0.6 seconds using CUDA. These performance increases allow source separation to be carried out on entire songs in a number of seconds, a process which was previously impractical with respect to time. We give insight into how such significant speed gains were made and encourage the development and use of parallel music information retrieval software.

1. INTRODUCTION

Even though music information retrieval (MIR) research is growing in importance and popularity, we have yet to see widespread adoption of MIR techniques in end-user applications. Part of this may be due to the ubiquity of online music recommendation services such as Pandora and Last.fm that use hand-labeled data and collaborative filtering as a basis for their recommendations, but also, the overall computational complexity of many MIR techniques makes their use outside of powerful compute clusters infeasible. The rate of progress of MIR research could be greatly improved if the execution time of MIR techniques was reduced enough to allow for quicker evaluation and tuning of algorithm parameters and more frequent real-world usage.

An emphasis on creating fast implementations has seen some attention, though not nearly enough. Tzanetakis produced submissions to MIREX 2007 using the Marsyas audio processing framework that ran orders of magnitude faster than the submissions of competitors while producing comparable results [1]. For example, in the audio mood classification task, the multi-core Tzanetakis implementation completed in 2 minutes, while competing implementations took between 8 minutes and 3 hours. Even for research implementations, such large speed differences can significantly impact the usability of MIR software.

In this paper, we describe our efforts to speed up percussive source separation based on non-negative matrix factorization (NMF), an unsupervised learning technique that has been used in audio source separation and parts-based analysis [2] [3] [4] [5]. Since NMF dominates the computation time in such a source separation task, it is an important computational procedure to optimize.

The goal of this paper is to demonstrate the dramatic speedup that can be achieved by multi-core and many-core implementations of multimedia applications and to encourage MIR researchers to develop and reuse high performance parallel implementations of important MIR procedures.

In Section 2, we explain the importance of producing parallel MIR applications. Section 3 covers the practical considerations for audio source separation based on NMF. In Section 4, we introduce the OpenMP and CUDA parallel programming models. Section 5 details the design of our parallel implementations and gives insight into techniques important to parallelizing MIR applications. Finally, Section 6 concludes with suggestions on how MIR can most benefit from parallel computing.

2. PARALLELIZING MULTIMEDIA APPLICATIONS

Percussive source separation is a useful first step in such MIR tasks as drum transcription, rhythm summarization, and beat tracking. By extracting an audio signal containing only percussive instruments, the task of rhythmic analysis can be greatly simplified. Helen and Virtanen [6] use NMF along with a support vector machine (SVM) to accomplish this. The drum track extractor we use as a target for performance optimization is similar to that presented in [6] but includes additional complexity optimizations and percussive features introduced in [7].

Computation time in this system is dominated by NMF, which makes up about 80% of the CPU time (18.5 seconds of the 23.1 seconds total) in a Matlab implementation run on 20 seconds of audio. In order to increase throughput, the NMF step must be optimized.

Because single-core CPU performance increases have been hindered by power concerns, limits on memory speed, and diminishing returns on instruction level parallelism, the focus of computer science research has turned strongly towards parallel architectures and programming models [8]. Applications programmers can no longer develop a sequential implementation of their software and hope that future uniproccessor speedups will provide the necessary computing power to make their application useful. Instead, the exponentially increasing number of processing elements, or cores, in current architectures must be exploited to maximize performance.

Multi-core CPU architectures are already commonplace in workstations, servers, and laptops, so parallelizing code to utilize available cores will lead to significant performance increases for most users. In addition, the majority of personal computers today ship with many-core graphics processors contained on the system's video card. Current high-end graphics processors (GPUs) ship with tens of processors each capable of executing operations on large data vectors. The end result is a highly data-parallel architecture that can be used for general computation (not just graphics rendering) thanks to programming frameworks like OpenCL [9] and Nvidia's CUDA [10].

CUDA has been successfully used to achieve very high performance on a variety of applications that rely on signal processing and machine learning. Examples include a fast GPU-based support vector machine implementation that achieves up to 135× speedup over LIBSVM [11], a large vocabulary speech recognition engine with 10× speedup over sequential versions [12], and an image contour detector that achieves 114× speedup [13]. To help put these numbers in perspective, the 114× speedup represents a reduction in runtime from 4 minutes to 2 seconds.

We aim to achieve such dramatic performance gains with NMF-based source separation.

3. NON-NEGATIVE MATRIX FACTORIZATION FOR AUDIO SOURCE SEPARATION

Non-negative matrix factorization can be used for audio source separation by decomposing a spectrogram matrix into two matrices which contain source-wise spectral contributions and time-varying gains. NMF can be phrased as the optimization problem:

Given an $M \times N$ non-negative matrix $\mathbf{X} \in \mathbb{R}_+^{M\times N}$, find matrices $\mathbf{W} \in \mathbb{R}_+^{M\times K}$ and $\mathbf{H} \in \mathbb{R}_+^{K\times N}$ that minimize the cost function $f(\mathbf{X}, \mathbf{WH})$.

3.1 Cost Function

Rather than using the mean-squared error between $\mathbf{X}$ and the product $\mathbf{WH}$ as the cost function, we use a matrix version of the Kullback-Leibler divergence:

$$D(\mathbf{X}\|\mathbf{WH}) = \sum_{ij}\left(\mathbf{X}_{ij}\log\frac{\mathbf{X}_{ij}}{(\mathbf{WH})_{ij}} - \mathbf{X}_{ij} + (\mathbf{WH})_{ij}\right) \tag{1}$$

It has been shown in [3] that this divergence cost function achieves better audio source separation results than mean-squared error.

3.2 Multiplicative Updates

Lee and Seung [14] have proposed an algorithm based on gradient-based multiplicative updates for minimizing the above optimization problem. For the divergence cost function, we alternate between updates on the two matrices using the following expressions

$$\mathbf{H} \leftarrow \mathbf{H}.*\frac{\mathbf{W}^{\mathrm{T}}\frac{\mathbf{X}}{\mathbf{WH}}}{\mathbf{W}^{\mathrm{T}}\mathbf{1}}, \qquad \mathbf{W} \leftarrow \mathbf{W}.*\frac{\frac{\mathbf{X}}{\mathbf{WH}}\mathbf{H}^{\mathrm{T}}}{\mathbf{1}\mathbf{H}^{\mathrm{T}}} \tag{2}$$

Where division is carried out element-wise, ".∗" is element-wise multiplication, and $\mathbf{1}$ represents an $M \times N$ matrix of ones and is used to compute row and column sums.

It is important to note that, because the optimization problem is not convex in both $\mathbf{W}$ and $\mathbf{H}$, the above updates do not necessarily converge to a global minimum. To address this problem, researchers typically use multiple random initializations and choose the best result. Adding extra computation time by running multiple trials cannot be done without significant justification since time to convergence can be in the minutes when operating on just seconds of audio.

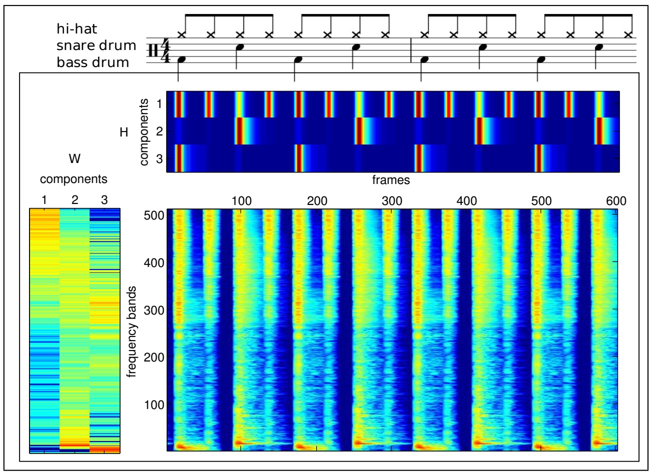


Figure 1. A spectrogram matrix for a basic rock beat surrounded by its factor matrices $\mathbf{W}$ and $\mathbf{H}$ computed using NMF. The component-wise gain matrix $\mathbf{H}$ has been aligned with the corresponding drum score.

3.3 Initialization

Other approaches use a deterministic initialization based on the structure or statistics of the matrix $\mathbf{X}$ or derived from knowledge about the domain. We use an approach based on the latter [7], which uses a subset of discrete cosine transform basis functions and typical drum spectra as the initial columns of $\mathbf{W}$. For our purposes, the initialization choice does not directly affect the speed with which the updates in eq. (2) are executed, but it can affect the overall number of iterations required for convergence. To eliminate this dependence, we will only focus on optimizing the speed of a set number of iterations rather than time to convergence.

3.4 Matrix Dimensions

An additional consideration that must be made is the dimensionality of the spectrogram matrix that is to be factorized. To adequately represent drum sounds in both time and frequency, a length 4096 Hann window is used to extract each analysis frame and a hop size of 256 is used to shift the window in time. For 20 seconds of audio sampled at 44.1kHz, this gives us a matrix of size 2049×3445 (number of positive frequency bins $\times$ number of analysis frames). Since such high frequency resolution ($\sim$10Hz) is not required at higher frequencies, we use a Bark-based perceptual dimensionality reduction [7] on the columns of $\mathbf{X}$ to arrive at a matrix of size 512×3445. After NMF is carried out on this smaller matrix, we can interpolate to return to the original frequency scale if necessary. Lastly, we choose an inner dimension for the factor matrices $\mathbf{W}$ and $\mathbf{H}$ of $K = 30$. This represents the number of sources involved in the separation.

Using these dimensions, our implementations require about 60MB of memory per minute of audio, making entire-song decomposition feasible from a memory standpoint.

Next we introduce the programming models that will be used to parallelize the NMF algorithm.

4. OPENMP AND CUDA

4.1 OpenMP

OpenMP is a standardized API that enables parallel execution on shared-memory multi-core machines [15]. OpenMP has been implemented for C, C++, and Fortran and is supported in Visual C++ 2005, the Intel compiler, and gcc 4.2 and above. The beauty of OpenMP lies in its ability to parallelize existing sequential code by annotating it with compiler directives. OpenMP automatically forks threads that execute on separate processors according to the directives.

OpenMP very conveniently parallelizes loops containing independent iterations using a single directive. The element-wise array multiplication shown below can be split amongst nt cores using a leading #pragma directive.

```
#pragma omp parallel for num_threads(nt)
    for(i=0;i<N;i++)
        c[i] = a[i]*b[i];
```

A reduction, which operates on multiple pieces of data and returns a single result, can be carried out using a *reduction* clause in the *for* pragma. In the example below, the reduction operator is addition, so we are returning the sum of an array. The first pragma creates a team of nt threads that are each assigned a chunk of the work in the for loop. After each thread completes its work, the values contained in each thread's private variable s are summed into a single final variable s.

```
    s = 0;
#pragma omp parallel num_threads(nt)
#pragma omp for reduction(+:s)
    for(i=0;i<N;i++)
        s += a[i];
```

4.2 CUDA

CUDA encompasses both the parallel device architecture used in newer Nvidia GPUs and the extensions to the C language used to program the CUDA architecture for general purpose computation. CUDA code compiled using Nvidia's *nvcc* is executed on the *host*, or CPU, which then issues instructions to the *device* or GPU. Host code typically contains control flow instructions and memory movement operations between host memory and device memory, while device code is made up of *kernels*, which are functions written to execute in a Single Program, Multiple Data (SPMD) fashion, i.e. each thread running on the device during kernel invocation executes the kernel code independently on whatever chunk of data is assigned to the thread.

Teams of threads can also share memory. As of CUDA 2.1, threads can be grouped into *thread blocks* of up to size 512. Threads within the same block are executed on the same processor and can all access special on-chip shared memory, which is necessary for inter-thread communication. Because separate thread blocks cannot share data, they can be executed independently on separate processors. Therefore, a kernel that uses a large number of thread blocks should scale well on future GPUs with more processors.

In the box below, we see a kernel that performs element-wise addition. Each thread runs the vecAdd function separately and computes an array index from its thread ID, block ID, and block size, and operates on the array elements located at that index. In the main function, the kernel is invoked with B thread blocks each containing N threads, so $B \times N$ should be equal to the size of the arrays.

```
// kernel definition
__global__ void vecAdd(float* a,
                       float* b, float* c){
    int i = threadIdx.x+blockIdx.x*blockDim.x;
    c[i] = a[i] + b[i];
}

int main(){
    . . .
    // kernel invocation
    vecAdd<<<B,N>>>(a,b,c);
}
```

Device kernels are physically executed in groups of 32 adjacent threads called *warps*. Warps are most efficient when the group of threads can be executed in a completely SIMD (Single Instruction, Multiple Data) manner, i.e. each thread in the warp does the exact same thing but to different data. Inserting control flow statements into a kernel that cause threads within the same warp to execute different code (this is referred to as a "divergent" warp) forces the affected threads to be run sequentially rather than concurrently. Double-precision hardware support is currently lacking in CUDA, which is why we focus on single-precision implementations in this work.

CUDA is designed to achieve high throughput on highly data-parallel computations. Luckily, most multimedia applications (especially music) exhibit a large amount of data parallelism.

5. PARALLEL IMPLEMENTATION

5.1 Important Kernels

To help organize our NMF implementation, we decompose the updates in eq. (2) into the most important computational kernels, including dense matrix multiplication, column and row sums, and element-wise vector arithmetic. Each of the kernels will be called sequentially, but individual kernels will be heavily parallelized and optimized.

The kernel that will do the most work in terms of floating point operations (flops) is the **S**ingle-precision **GE**neral **M**atrix **M**ultiply, or *SGEMM*. For the matrix dimensions listed at the end of Section 3.4, the four SGEMMs in eq. (2) require about 423 Mflops. The element-divides require about 3.6 Mflops, the sums about 0.1 Mflops, and the element-multiplies about 0.1 Mflops. To prevent dividing by zero, a small constant (called *EPS*) is added to every element in each divisor matrix, which produces a non-trivial amount of work (3.6 Mflops). Also, in order to check for convergence, we compute the divergence cost function (1) every 25 iterations, which computes the sum of 1.8×10^6 log-based values.

Even though the SGEMMs contain the vast majority of the work, other operations, namely the slow floating-point divides and the sums, can end up using a lot of compute time. Divides are inherently slow operations and can take tens of clock cycles on certain architectures. While the sums contain relatively few total operations, a parallelized sum will require inter-thread communication which can be very slow. Since a highly optimized SGEMM routine is available in most vendor BLAS libraries, our implementation goal was to tune the remaining kernels so that the SGEMMs dominate the overall computation time. Practically speaking, significantly outperforming our Matlab implementation (which takes 18.5 seconds to run 200 iterations on a Core 2 Duo T9300) was a more exciting goal.

5.2 OpenMP Implementation

As stated before, OpenMP makes it very easy to parallelize existing sequential code for a multi-core shared-memory machine. Using the two types of *for* pragmas from Section 4.1 we can parallelize the sums and element-wise arithmetic. Since the element divides are numerous, slow, and do not require inter-thread communication, it makes sense to parallelize their loop. The row and column sums, however, require a lot of communication for the amount of addition work done per core (since the partial sum computed by each core must be sent to another core), so parallelizing the reduction loop actually led to a slower kernel. The larger sum in the divergence cost function not only contains lots of addition but a slow log-based computation, so the work to communication ratio was befitting parallel speedup.

For the SGEMMs, we use Intel's Math Kernel Library (MKL) ver. 10.0.1.014, which is heavily optimized to take advantage of memory hierarchy and SIMD instructions. MKL uses OpenMP under the hood, so the number of threads used for the SGEMMs can be controlled in the same way as our parallel loops.

Performance results for the OpenMP implementation are shown in Figure 2 for a dual-socket Intel Core i7 920 machine which has 8 cores and 16 hardware threads. The best performance is seen at 14 threads and is about 4.3× faster than the single-threaded run. The most significant speed up is seen in the SGEMM since it has the highest work to communication ratio, but other time-consuming kernels benefit as well. Running this implementation on the Core 2 Duo T9300 with 2 threads takes 8.9 seconds, which is 2× faster than our optimized Matlab implementation using 2 threads.

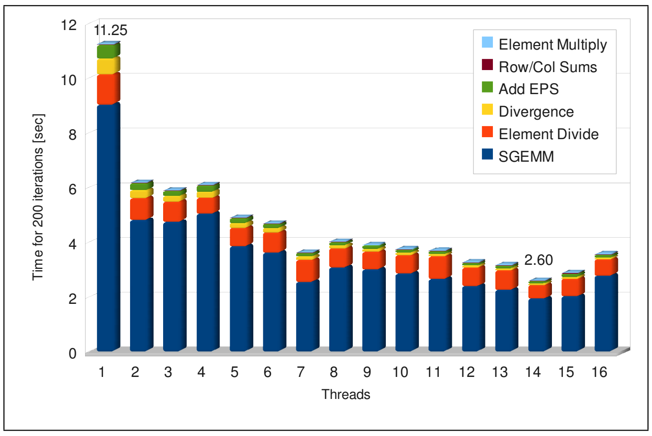


Figure 2. Performance results for the OpenMP implementation on a dual-socket Intel Core i7 920

5.3 CUDA Implementation

Writing a CUDA implementation takes a bit more thought. First, the matrices must be copied to GPU memory. Copies between CPU and GPU are relatively slow (ideally 3 GB/s over the PCI bus), and it's best to avoid them except during initialization or when returning results. This means that in our case it's better to perform all of the matrix computations on the GPU to avoid extra copies even if certain operations are better suited for the CPU.

Element-wise arithmetic is completely data-parallel and is easily accomplished with code similar to that in Section 4.2. Other kernels, including the SGEMMs and sums, require a bit of inter-thread communication and are not so trivially parallelized on CUDA.

5.3.1 SGEMM

Luckily, an optimized SGEMM routine is available in the CUBLAS 2.1 library that achieves 60% of theoretical peak performance for large matrices on current GPUs [17]. For the Geforce GTX 280, 60% of peak amounts to 373 Gflops/s. For our particular matrix multiplications of dimensions $[512 \times 30 \times 3445]$, $[30 \times 512 \times 3445]$, and $[512 \times 3445 \times 30]$, the CUBLAS SGEMM achieves 117, 147, and 104 Gflops/s respectively on this GPU. Even though these are relatively small SGEMMs, we should still be able to do better.

Upon inspection of the paper [17] that describes the methods used in the current CUBLAS SGEMM, we discovered that threads operate on matrix sub-blocks with dimensions 16 and 64. With this in mind, we tried zero padding our matrices to multiples of 16, 32, and 64. We found that simply padding the matrices to multiples of 32 resulted in an effective throughput (not counting operations on zero-padded areas) of 264, 196, and 85 Gflops/s for each SGEMM size. Since the NMF algorithm uses two SGEMMs of the first size, this results in an SGEMM running time reduction from 0.71 to 0.52 seconds for 200 iterations.

5.3.2 *Reduction*

Because parallel reductions, such as sums, mins, and maxes, are not included in standard libraries, we will have to write our own routines. A tutorial on optimizing reductions in CUDA is available in the CUDA SDK [18]. This overview presents optimization strategies that can be used to greatly improve the speed of large power-of-2-size reductions and shows how a 30× speedup can be achieved for a 4.2×10^6 length sum over a naive binary tree implementation.

A binary tree reduction can be constructed in various ways. Using the shared memory of a thread block, we can perform a series of two-element reductions. Two ways to organize the overall reduction are shown in Figure 3. In both versions, each thread in the thread block starts by reading an array element from global memory into shared memory. Then threads are assigned to carry out two-element sums.

The difference lies in which threads work on which array elements. Method 1 interleaves working and non-working threads which act on adjacent elements. Method 2 sequentially assigns working threads so there are contiguous blocks of working and non-working threads. This decreases the number of divergent warps. Also, the memory accesses are strided rather than adjacent to reduce the number of simultaneous memory bank accesses (since shared memory locations are cyclically assigned to memory banks) [16].

In addition to reorganizing the tree traversal, other optimizations –such as explicit loop unrolling and allowing each thread to read and sum multiple array elements into its shared memory location before the tree traversal begins– improve performance a bit. These techniques had to be adapted for non-power-of-2-size arrays, but they greatly improved the speed of the large 1.8×10^6 length divergence sum.

For the smaller 512 and 3445 length column and row sums, these techniques were not quite enough, and the CUDA kernel ran much slower than a sequential CPU version. In

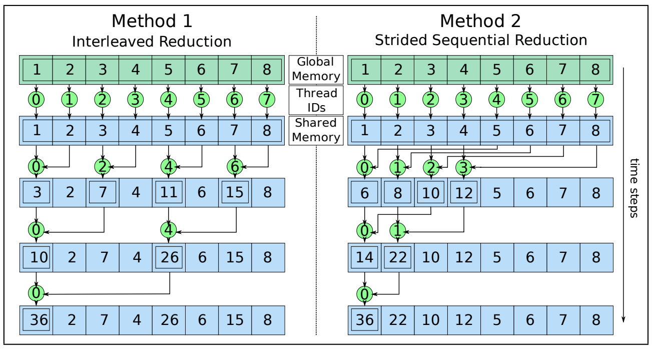


Figure 3. Two methods of shared memory reduction

order to produce more concurrent work (in terms of thread blocks), we can compute all 30 of the column or row sums simultaneously. This is accomplished by launching a 2D *grid* of thread blocks, in which the first dimension represents which of the 30 sums is being computed and the second dimension indexes the thread blocks within the individual sum. This final optimization produced staggering speedup for the 30 smaller sums as shown in Figure 4.

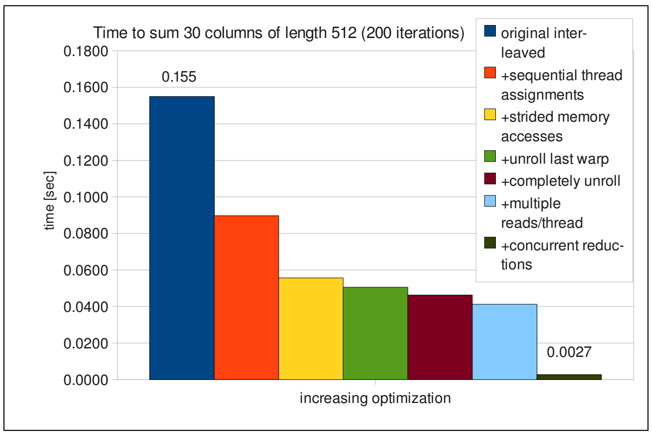


Figure 4. Cumulative effect of various optimizations on running time of 200 iterations of the 30 column sums

5.3.3 *CUDA Performance Results*

The results for the CUDA implementation compared to OpenMP and Matlab implementations are shown in Figure 5. The Matlab implementation is optimized for single-precision vector operations and uses the dimensionality reduction technique mentioned in Section 3.4. Our Matlab implementation runs about 3× faster than a naive Matlab implementation that doesn't use dimensionality reduction. The OpenMP version runs more than twice as fast as the Matlab version on the same machine, and shows significant speedup when using more threads on the Core i7; however, the non-linear speedup between 1 and 14 threads suggests that the OpenMP version will not scale well to more cores.

Our CUDA implementation shows great performance on the older Geforce 8600 GTS, which has 4 multiprocessors at 1.46 GHz. The newer Geforce GTX 280, with 30 multiprocessors at 1.3GHz, runs the CUDA implementation over 30× faster than the optimized Matlab implementation and 18× faster than the single-threaded OpenMP

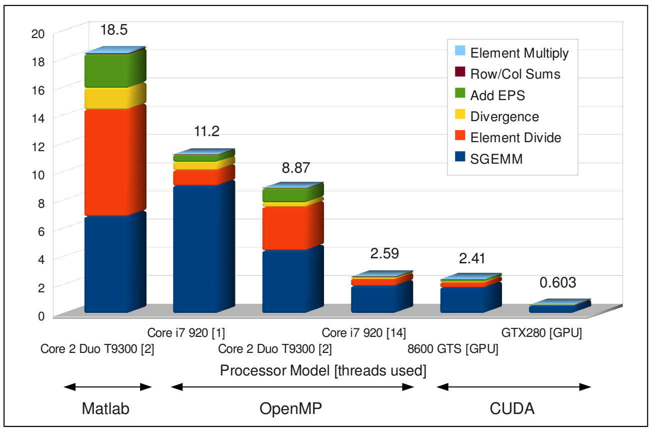


Figure 5. Running time comparison for 200 iterations of $512 \times 30 \times 3445$ NMF using optimized implementations in Matlab, OpenMP, and CUDA on different architectures

version on the Core i7 920. Both of these GPUs are marketed to consumers for desktop gaming and graphics so are quite affordable compared to many of the professional-grade cards.

Additional speedup is possible with future GPUs with more multiprocessors and greater memory bandwidth. As stated earlier, CUDA programs scale well if kernels have a large number of independent thread blocks. The relatively small size of the matrix operations doesn't guarantee strong scaling in the future, but in this case, additional speedup is not necessarily required. For audio source separation, the NMF already performs at 33× real-time on the GTX 280.

6. DISCUSSION AND FUTURE WORK

After achieving such significant speedup on the NMF step of percussive source separation, the next step would be to parallelize the remaining pieces of the complete source separation process. As with the bulk of signal processing and machine learning routines, these steps are all very data-parallel (since individual audio frames can be processed independently) so would benefit from parallelization.

When choosing between OpenMP and CUDA for programming MIR applications, it is important to note that while CUDA can achieve superior performance on newer GPUs, the programmer effort required is much greater than with OpenMP, which is a better starting point for those who already know how to program in C. We must also remember that parallel MIR applications do not necessarily have to be coded from scratch. Many MIR techniques can be assembled from basic building blocks that already have fast parallel implementations. In addition to standard libraries like MKL, fftw, and CUBLAS, many researchers have released parallel implementations of important routines.

We will be releasing Python modules for the implementations described in this paper so that other researchers can benefit from the speed gains. We feel that sharing high-performance, user-friendly tools in order to encourage more widespread use of parallel implementations within the MIR community is an important step in increasing the practicality of MIR techniques.

7. REFERENCES

[1] G. Tzanetakis: "Marsyas submissions to MIREX 2007", *MIREX 2007*, 2007. URL: http://www.music-ir.org/mirex/2008/abs/mirex2007.pdf

[2] D. Lee and H. Seung: "Learning the parts of objects by non-negative matrix factorization," *Nature*, Vol. 401, pp. 788–791, 1999.

[3] T. Virtanen: "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 3, pp. 1066–1074, 2007.

[4] P. Smaragdis and J. Brown: "Non-negative matrix factorization for polyphonic music transcription," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 177–180, 2003.

[5] A. Cont, S. Dubnov, D. Wessel: "Realtime Multiple-Pitch and Multiple-Instrument Recognition for Music Signals Using Sparse Non-Negative Constraints," *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2007.

[6] M. Helen and T. Virtanen: "Separation of drums from polyphonic music using nonnegative matrix factorization and support vector machine," *Proc. EUSIPCO*, 2005.

[7] E. Battenberg: "Improvements to Percussive Component Extraction Using Non-Negative Matrix Factorization and Support Vector Machines," Masters Thesis, University of California, Berkeley, December 2008. URL: http://cnmat.berkeley.edu/publications/author/Battenberg

[8] K. Asanovic, R. Bodik, et al.: "The landscape of parallel computing research: A view from Berkeley," *Electrical Engineering and Computer Sciences, University of California at Berkeley, Technical Report No. UCB/EECS-2006-183*, December, 2006.

[9] A. Munschi: "OpenCL: Parallel computing on the GPU and CPU," *SIGGRAPH08: ACM SIGGRAPH 2008 classes*, 2008.

[10] J. Nickolls, I. Buck, et al.: "CUDA: Scalable parallel programming," *ACM Queue*, April, 2008.

[11] B. Catanzaro, N. Sundaram, and K. Keutzer: "Fast support vector machine training and classification on graphics processors," *Proceedings of the 25th international conference on Machine learning*, pp. 104–111, 2008.

[12] J. Chong, Y. Yi, et al.: "Data-Parallel Large Vocabulary Continuous Speech Recognition on Graphics Processors," *Proceedings of the 1st Annual Workshop on Emerging Applications and Many Core Architecture (EAMA)*, pp. 23–25, 2008.

[13] B. Catanzaro, B. Su, et al.: "Efficient, high-quality image contour detection," *International Conference on Computer Vision*, 2009.

[14] D. Lee and H. Seung: "Algorithms for Non-negative Matrix Factorization'," *Advances In Neural Information Processing Systems*, pp. 556–562, 2001.

[15] Open MP Architecture Review Board: *OpenMP application programming interface*, Ver. 2.5, May 2005.

[16] "Nvidia CUDA Programming Guide," Ver. 2.1, URL: developer.download.nvidia.com/compute/cuda/2_1/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.1.pdf, 2008.

[17] V. Volkov and J. Demmel: "Benchmarking GPUs to tune dense linear algebra," *Supercomputing 08*, 2008.

[18] M. Harris: "Optimizing parallel reduction in CUDA," *Nvidia Cuda SDK 2.1*, URL: http://developer.download.nvidia.com/compute/cuda/sdk/website/projects/reduction/doc/reduction.pdf, 2008.

Research supported by Microsoft and Intel funding (Award #20080469) and by matching funding by U.C. Discovery (Award #DIG07-10227)

MUSICAL MODELS FOR FOLK-SONG MELODY ALIGNMENT

Peter van Kranenburg, Anja Volk, Frans Wiering, Remco C. Veltkamp
Utrecht University, Department of Information and Computing Sciences
{petervk,volk,fransw,Remco.Veltkamp}@cs.uu.nl

ABSTRACT

In this paper we show that the modeling of musical knowledge within alignment algorithms results in a successful similarity approach to melodies. The score of the alignment of two melodies is taken as a measure of similarity. We introduce a number of scoring functions that model the influence of different musical parameters. The evaluation of their retrieval performance on a well-annotated set of 360 folk-song melodies with various kinds of melodic variation, shows that a combination of pitch, rhythm and segmentation-based scoring functions performs best, with a mean average precision of 0.83.

1. INTRODUCTION

In this paper we use alignment algorithms to measure the similarity of melodies. Alignment algorithms are widely used for comparison of sequences of symbols. Creating an alignment is a way to relate two sequences with each other by finding the best corresponding parts. Especially in the field of computational biology, where they are used to find corresponding patterns in protein or nucleotide sequences, many algorithms that align sequences have been developed. Sequence alignment is also suitable for assessing musical similarity for several reasons. Firstly, music unfolds in time, therefore, a model of music as a one-dimensional sequence of events seems appropriate. Secondly, manual alignments have extensively been used in folk-song research to evaluate relations between melodies. Thirdly, structural alignment is a prominent model in cognitive science for human perception of similarity [3].

Most alignment algorithms use a dynamic programming approach. One of the earliest variants is the Levenshtein distance [8], which is an edit distance: it computes how many operations are needed to transform one sequence into another. Needleman and Wunsch [9] proposed an algorithm that finds an optimal alignment of two complete sequences. The quality of an alignment is measured by the alignment score, which is the sum of the alignment scores of the individual symbols. If we consider two sequences of symbols $\mathbf{x} : x_1, \ldots, x_i, \ldots, x_n$, and $\mathbf{y} : y_1, \ldots, y_j, \ldots, y_m$, then symbol x_i can either be aligned with a symbol from sequence $\mathbf{y}$ or with a gap. Both operations have a score, the substitution score and the gap score. The gap score is mostly expressed as penalty, i.e. a negative score. The optimal alignment and its score are found by filling a matrix D recursively according to:

$$D(i,j) = \max \begin{cases} D(i-1,j-1) + S(x_i, y_j) \\ D(i-1,j) - \gamma \\ D(i,j-1) - \gamma \end{cases}, \quad (1)$$

where $S(x_i, y_j)$ is the substitution scoring function, γ is the gap penalty, $D(0,0) = 0$, $D(i,0) = -i\gamma$, and $D(0,j) = -j\gamma$. $D(i,j)$ contains the score of the optimal alignment up to x_i and y_j and therefore, $D(m,n)$ contains the score of the optimal alignment of the complete sequences. We can obtain the alignment itself by tracing back from $D(m,n)$ to $D(0,0)$; the algorithm has both time and space complexity $O(nm)$. In our modeling, we use an extension of the algorithm proposed by Gotoh [5], which employs an affine gap penalty function without loss of efficiency. In this approach, the extension of a gap gets a lower penalty than its opening.

Mongeau and Sankoff [10] were among the first to adapt alignment algorithms to music. They used an extended version of the Needleman-Wunsch algorithm. Their scoring function takes both pitch and duration into account. Mongeau and Sankoff's approach has been quite influential, e.g. the search algorithm implemented in the search engine MELDEX [13] is based on this algorithm. Gómez et al. [4] successfully tested a modified version on a MIREX dataset. In general, alignment algorithms have often been used to match short melodic phrases against a larger database [1,4,7,13,14]. Typical tasks addressed with this approach are to find a tune in the database with QBH [1], different arrangements of a piece [14], or similar incipits given to the query [4]. We use the alignment between complete melodies in order to find melodies that belong to the same tune family. The similarity relations that have to be modeled originate in the oral transmission of folk-songs and differ from those in the previous tasks.

Contribution. In this paper we model various features of music as substitution scoring functions, which we incorporate in the Needleman-Wunsch-Gotoh algorithm. Using a set of melodies that are well-described regarding their different kinds of similarity relations, we evaluate the influence of these scoring functions on the retrieval performance. Our best scoring function combines several musical features and outperforms well-known approaches from literature.

2. DATA

2.1 The Annotated Corpus

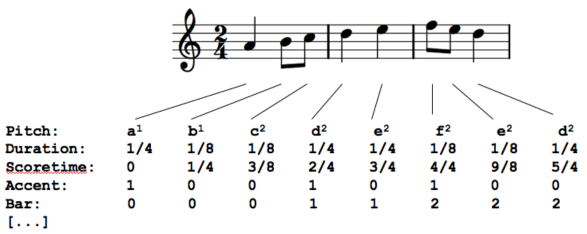
The set of melodies studied in this paper is part of a larger collection of over 6000 encoded Dutch folk-songs hosted by the Meertens Institute in Amsterdam. In the ongoing project of digitization at this institute, the melodies are encoded both from ethnomusicological transcriptions of field recordings and from written sources of folk-songs, delivering several formats (humdrum **kern, MIDI, Lilypond). [1] For a subset of 360 melodies, detailed annotations have been created in order to describe similarity relations between melodies [16], resulting in a well-documented set of songs, the Annotated Corpus.

The melodies are grouped in so-called tune families. [2] All melodies in one group are considered to be historically related through the process of oral transmission. Since the history of each tune family is not fully documented, it is often not known whether two melodies are historically related. Instead, musicological experts decide whether melodies belong to the same tune family by assessing their melodic and textual similarity. In order to make the experts' musical intuition behind the similarity assessments explicit, we developed an annotation system (described in [16]). For the Annotated Corpus (consisting of 26 tune families) several dimensions of perceived similarity (contour, rhythm, motives, text) were numerically rated by the musicologists such that the similarity between the most typical melody of a tune family (the reference melody) and all other members of the tune family was described. The 26 tune families were chosen from the larger collection by an expert such that this set contains a representative diversity of similarity relations between members of a tune family. Comparing the annotations to the retrieval performance of alignment algorithms allows a detailed understanding of the success or failure of the models based on musicological insights.

2.2 Representation of melodies

For applying alignment algorithms, a melody has to be presented as a sequence of symbols. In our representation, each symbol represents a note. A symbol has a number of attributes, including: pitch (in base40 encoding), duration (rational number), score time (rational number), time in bar (rational number), onset (integer), current bar number (integer), current phrase number (integer), upbeat (boolean), current meter (rational number), free meter (boolean), accented (boolean), and time position within phrase (real number in $[0, 1]$). These attributes are used to compute substitution scores or other attributes. Figure 1 shows an example with some of the attributes.

Based on the encoded time signature, two levels of accents are distinguished: either accented or not accented. The first note of any group of two in a double meter and the first beat in any group of three beats in a triple meter is considered accented. All other notes are unaccented. Thus, in songs in free meter, or in songs with additive or asymmetrical meters, [3] which are very uncommon in this corpus, all notes are unaccented. Furthermore, phrase boundaries have been annotated by the encoders.



```
Pitch:      a¹   b¹   c²   d²   e²   f²   e²   d²
Duration:   1/4  1/8  1/8  1/4  1/4  1/8  1/8  1/4
Scoretime:  0    1/4  3/8  2/4  3/4  4/4  9/8  5/4
Accent:     1    0    0    1    0    1    0    0
Bar:        0    0    0    1    1    2    2    2
[...]
```

Figure 1. Representation of melodies.

2.3 Rests

Most notated rests can be considered inessential. In particular at the end of phrases singers often take a breath, such that timing between the phrases is very variable. The exact encoding of rests as performed is therefore not reasonable. To make melodies more comparable, all rests have been replaced by a prolongation of the previous note.

2.4 Transposition Invariance

Since songs are notated in different keys, the similarity measure should be transposition invariant. To achieve this, a pitch histogram for both melodies is created that indicates for each pitch the total duration during the song. Then the shift at which the normalized histograms have maximal intersection is computed. Since the pitches are represented in base40 encoding, the shift of the histogram can be interpreted as the interval with which the one melody should be transposed in order to compare it to the other.

2.5 Normalization of Alignment Scores

Since the score of an alignment depends on the length of the sequences, normalization is needed to compare different alignment scores. Therefore, we divide the alignment score by the length of the shortest sequence.

3. SCORING FUNCTIONS

3.1 Single substitution scoring functions

In this section we introduce a number of substitution scoring functions for different musical dimensions. They determine substitution scores that are based on musicological knowledge. Each function takes two symbols of the melodic sequence as input. The output of each scoring function is in the interval $[-1, 1]$.

First, we introduce scoring functions that are based on pitch-related features. The simplest scoring function determines whether two pitches are the same or not. The score

[1] The full collection is browsable at: http://www.liederenbank.nl.

[2] At the Meertens Institute the concept of "melody norm" is used, see [16] for a more detailed explanation of this concept.

[3] Asymmetrical meters consist of stacked groupings of dissimilar metrical groups.

is either maximal or minimal:

$$S_{exactpitch}(x_i, y_j) = \begin{cases} 1 & \text{if } x_i = y_j \\ -1 & \text{if } x_i \neq y_j \end{cases} . \quad (2)$$

In oral transmission, slight changes of pitches are likely to occur, therefore, we allow substitution with pitches that are within a band with certain width:

$$S_{pitchb}(x_i, y_j) = \begin{cases} 1 - \frac{int(x_i,y_j)}{23} & \text{if } int(x_i, y_j) \leq 23 \\ -1 & \text{otherwise} \end{cases} . \quad (3)$$

We define $int(x_i, y_j) = |p(x_i) - p(y_j)| \bmod 40$, with $p(x)$ as the pitch of symbol x in base 40 encoding. A fifth is 23 in base 40 encoding. Thus, all intervals up to a fifth get a positive substitution score and all larger intervals are considered a bad match.

Another way to express the distance of two pitches is by their harmonic relation. The substitution of consonances gets a higher score than the substitution of dissonances:

$$S_{harm}(x_i, y_j) = \begin{cases} 1 & \text{prime} \\ 0.5 & \text{consonance} \\ 0.5 & \text{augmented prime} \\ -1 & \text{dissonance} \end{cases} . \quad (4)$$

The intervals are taken modulo octave. Consonances are minor and major third, perfect fourth, perfect fifth and minor and major sixth. The augmented prime gets a positive substitution score to favour alignments of songs that have a minor and a major variant.

Furthermore, we define two substitution functions that are based on melodic contour, taking either the contour of a phrase or of the entire melody into account:

$$S_{phrasecont}(x_i, y_j) = 1 - 2 * |p_{phr}(x_i) - p_{phr}(y_j)| \ . \quad (5)$$

$$S_{songcont}(x_i, y_j) = 1 - 2 * |p_{song}(x_i) - p_{song}(y_j)| \ . \quad (6)$$

Here $p_{phr}(x)\epsilon[0, 1]$ indicates the vertical position between the lowest and highest pitches of the phrase that x is part of, while $p_{song}(x)\epsilon[0, 1]$ indicates the vertical position between the lowest and highest pitches of the entire song. In determining the highest and lowest pitches, the notes in the upbeats of the phrases are disregarded, since these are very variable between variants of a song.

Next, we define three scoring schemes that are based on rhythmic features. In a simple approach based on note durations, the score is maximal if the durations are the same, and minimal otherwise:

$$S_{exactdur}(x_i, y_j) = \begin{cases} 1 & \text{if } d(x_i) = d(y_j) \\ -1 & \text{if } d(x_i) \neq d(y_j) \end{cases} , \quad (7)$$

in which $d(x)$ is the duration of the symbol x.

Metric accents derived from the notated time signature describe a further aspect of the rhythmic structure of melodies. We define a substitution function that uses these metric accents in the following way:

$$S_{accent}(x_i, y_j) = \begin{cases} 1 & \text{if } a(x_i) = a(y_j) \\ -1 & \text{if } a(x_i) \neq a(y_j) \end{cases} , \quad (8)$$

in which $a(x)$ indicates whether the symbol x is accented or not (for defining accents see section 2.2).

A more complex notion of metric accents based on the rhythmic structure of notes instead of the time signature is provided by Inner Metric Analysis (IMA) [15]. We define a scoring function that is determined by the metric weights of the notes, as computed by IMA:

$$S_{ima}(x_i, y_j) = 1 - 2 * |w(x_i) - w(y_j)| \ . \quad (9)$$

Here $w(x)$ denotes the metric weight of the symbol x scaled into the interval $[0, 1]$. For scaling, all weights were divided by the greatest weight in the song. The parameters for the IMA algorithm are the ones that are mostly used: $p = 2, l = 2$ (e.g., in [15]).

Furthermore, we want to use the information of phrase boundaries given in our data-set. We introduce a scoring function based on the horizontal position within the phrase:

$$S_{phr}(x_i, y_j) = 1 - 2 * |phr(x_i) - phr(y_j)| \ , \quad (10)$$

in which $phr(x)\epsilon[0, 1]$ indicates for the symbol x the horizontal position in its phrase. This substitution function helps to keep phrases together in alignments.

3.2 Combination

The single substitution scoring functions defined in section 3.1 model isolated aspects of melodies. In order to model several aspects within one function to get closer to the multidimensionality of melodies, we combine substitution functions. We want alignments in which the aligned symbols are similar in all dimensions, therefore, we multiply the individual scores:

$$S_{combination}(x_i, y_j) = \prod_{k=1}^{n} S_k(x_i, y_j) \ , \quad (11)$$

in which each $S_k(x_i, y_j)$ is scaled into the interval $[0, 1]$, and the final score is scaled into $[-1, 1]$ back again.

3.3 Gap penalty function

We use an affine gap penalty function in which the penalty for opening a gap is 1, and the penalty for extending a gap is 0.1. Thus, variants of songs in which e.g. a phrase is repeated can be better aligned, since these penalties result in one long gap instead of many short gaps. Furthermore, the use of an affine gap penalty function prevents gaps from being scattered all over the alignment.

4. EVALUATION OF SCORING FUNCTIONS

The scoring functions are evaluated by their respective retrieval performance on our Annotated Corpus as described in section 2. To evaluate a scoring scheme each melody is taken once as query and the other melodies are sorted according to the normalized score of the alignment with the query melody. At all ranks the average recall and average precision for all ranking lists is computed. These values are plotted in a diagram. The criterion for relevance is the membership of the same tune family.

4.1 Evaluation of Single Substitution Functions

First, we study the performance of the single pitch-based substitution functions introduced in section 3.1. Variation in pitch is considered an important element of oral transmission (see e.g. [6]). Nevertheless, aligning melodies using the exact pitch information with the simplest function $S_{exactpitch}$ results in a relatively good performance (see Figure 2). Allowing pitch variation within a small range using the pitch band function improves this performance only slightly.

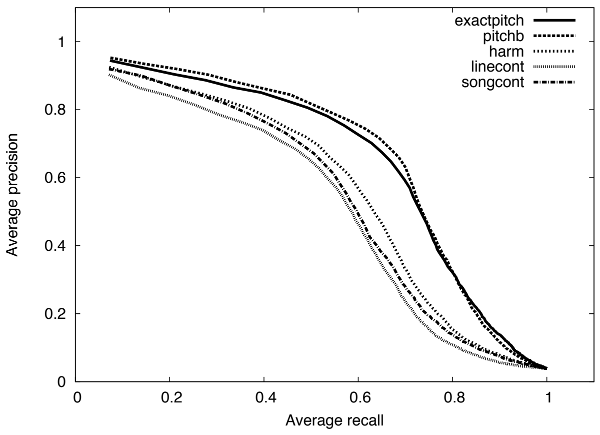


Figure 2. Retrieval performance of pitch-based substitution functions.

Both the harmonic and contour-based substitution functions perform worse than $S_{exactpitch}$. Considering the contour instead of the exact pitch sequence does not result in a better retrieval performance. Harmonic relations, which have otherwise successfully been used in models of melodic expectancy [11], do not improve the alignment of melodies of a tune family in comparison to exact pitch information.

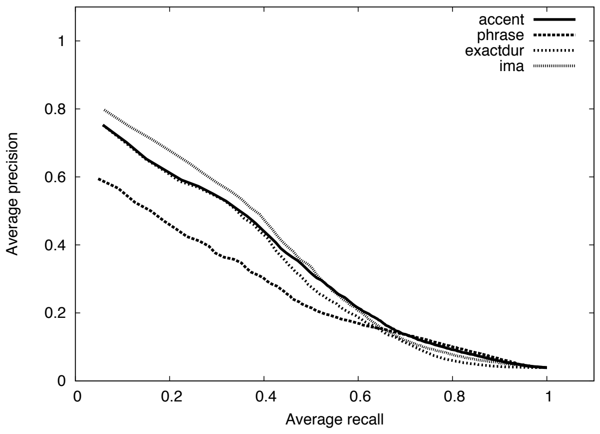


Figure 3. Retrieval performance of non-pitch-based substitution functions.

Figure 3 shows retrieval performance for the scoring functions that do not involve pitch information. Although rhythmic features have been considered quite stable within oral transmission (see [6]), all rhythm-related substitution functions perform worse than pitch-related functions. S_{ima} performs at the top of the ranking slightly better than S_{accent}, however S_{accent} performs slightly better in the low range. In general the difference between the two models is quite small, indicating that the accents of the notated bars are synchronous to the accents based on notes onsets.

4.2 Evaluation of Combinations of Single Substitution Functions

In a next step, we combine rhythmical, metrical and segmentation data. First, we combine the best of the pitch-related functions (S_{pitchb}) with rhythmical and phrase functions. Figures 2 and 3 show that the individual substitution functions perform worse than S_{pitchb}, but from the curves of $S_{pitchb-accent}$, $S_{pitchb-phrase}$, $S_{pitchb-exactdur}$, and $S_{pitchb-ima}$ in Figure 4 it appears that combinations yield better retrieval performance for all combinations but $S_{exactdur}$. Since $S_{exactdur}$ is binary and the combination is by multiplication, the pitch similarity for symbols with no exact correspondence in duration is lost.

Combination with the other two rhythmic functions (S_{ima} and S_{accent}) show equal improvement. The rather modest improvement when considering metric accents in comparison to the single substitution function S_{pitchb} contradicts the hypothesis that pitches among melodies of the same tune family are more stable on metrically accented notes than on metrically weak positions as assumed in [2]: obviously pitches on metrically weak positions also vary to only a small extent. The phrase information yields the greatest improvement.

Finally, we evaluate the retrieval performance of the combination of the best substitution functions. We choose S_{ima} as the metric scoring function, S_{pitchb} is the best pitch based scoring function. S_{phr} improved retrieval results by stimulating phrase boundaries to be aligned. The retrieval performance of the combination $S_{pitchb-ima-phr}$ shown in Figure 4 shows even better performance results than the combinations of two single substitution functions. If we average the precision of all relevant items for all queries, we get a mean average precision of 0.83 for this combination. Choosing S_{accent} instead of S_{ima} gives nearly the same retrieval performance.

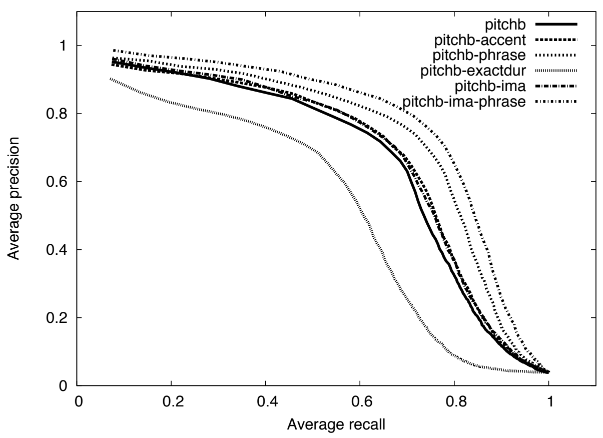


Figure 4. Retrieval performance of combinations of substitution functions.

To evaluate the scalability, we performed the same test with a data-set containing all 4863 classified songs and with the same 360 queries. The results yield a mean average precision of 0.67.

4.3 Comparison with Related Methods

Figure 5 shows comparisons of our best scoring scheme with alignment methods from literature. For the method of Mongeau and Sankoff [10] the parameters were taken as given by Mongeau and Sankoff. The normalization was done by dividing the alignment score by the sum of the durations of both sequences. DiffEd and rawEd were taken from the Simile alignment toolbox without change [12]. It appears that our $S_{pitchb-ima-phr}$ performs best.

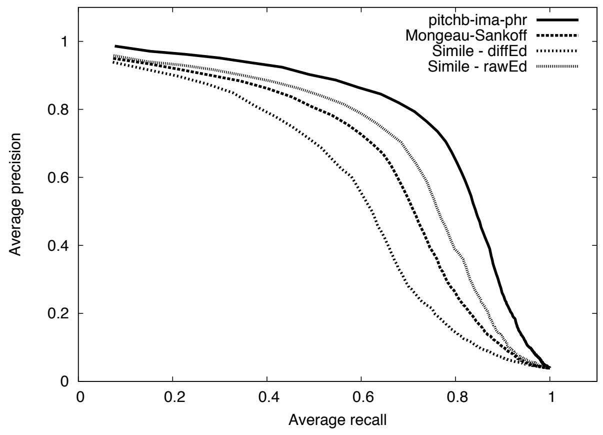


Figure 5. Comparison with related methods.

5. RETRIEVAL PERFORMANCE PER TUNE FAMILY

The classification of the melodies into tune families by musicological experts is based on a number of musical dimensions. The importance of the different dimensions and the form of interaction between them varies to a great extent among tune families. Therefore, finding a similarity model that performs well on all tune families is a challenging task. The retrieval performance of the scoring function $S_{pitchb-ima-phr}$ shows per tune family a considerably stable success, with average precision values ranging from 1 to 0.8 for 23 out of 26 tune families. Hence, this function works reasonably well on the majority of tune families. For three tune families the function shows only moderate retrieval performance, which are *Een lindeboom stond in het dal 1* (short: *Lindeboom*), *Daar reed een jonkheer 1* (short: *Jonkheer*) and *Heer Halewijn 4* (short: *Halewijn*), with respective average precision of 0.71, 0.67 and 0.65. Table 1 gives an overview of low ranking results extracted from rankings in which the reference melody of the tune family (see section 2.1) was used as the query. For *Lindeboom*, Figure 6 shows the first line of the query along along with a good match and the two melodies with ranks 77 and 140. The melodies on these low ranks are quite different from the query. This is reflected by the experts' annotations: for the two melodies on ranks 77 and 140 all global musical

Tune family	Relevant melodies at rank
Lindeboom	77 and 140
Jonkheer	78, 98 and 167
Halewijn	19, 40, 74 and 101

Table 1. Overview over low ranking results.

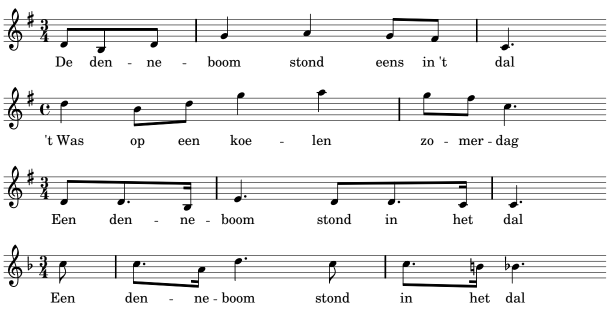

De den - ne - boom stond eens in 't dal
't Was op een koe - len zo - mer - dag
Een den - ne - boom stond in het dal
Een den - ne - boom stond in het dal

Figure 6. From top to bottom: incipits of reference melody (query), melodies on ranks 1, 77 and 140..

dimensions (rhythm, contour and importance of motives) are rated *somewhat similar*, while for most of the other melodies of *Lindeboom* at least two of these dimensions are rated *very similar*.

In the tune family *Jonkheer* the melody on rank 78 is notated with a different phrase structure than the query, such that two phrases correspond to one phrase of the query. Since phrase information is used for the alignment, this inconsistent phrase assignment introduces lower scores. The melody on rank 98 has a very different formal structure than the query: while the query has the form ABA', this melody has the form AAA'A". As a consequence, notes in phrases that are aligned with each other differ to a great extent. The melody on rank 167 is quite different from the query (see Figure 7), which is reflected by low ratings of the experts in both local and global musical dimensions.

The tune family *Halewijn* is according to the experts most of all characterized by a similar rhythmic organization. Melodies of this group differ considerably regarding pitch, such that the contour is mostly rated as only *somewhat similar*. The averaged annotation values for all musical dimensions in this tune family show a significant ($p = 0.02$) linear correlation with the distances obtained using the $S_{pitchb-ima-phr}$ rater— hence the lowest ranked melodies tend to receive low similarity scores in the annotations. For the melody on rank 74 the expert commented that it is possible that this melody does not belong to the tune family.

6. CONCLUSION AND FUTURE WORK

We have shown that the inclusion of musical knowledge in alignment algorithms improves the assessment of similarity among folk song melodies. By evaluating different substitution scoring functions, we found that our pitch-related functions lead to better recognition than rhythm-related functions. The use of phrase information improved

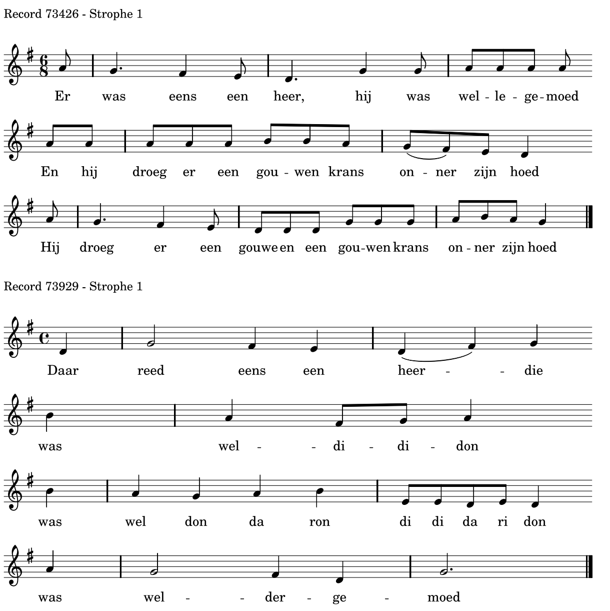

Record 73426 - Strophe 1
Er was eens een heer, hij was wel - le - ge-moed
En hij droeg er een gou - wen krans on - ner zijn hoed
Hij droeg er een gouween een gou-wenkrans on - ner zijn hoed
Record 73929 - Strophe 1
Daar reed eens een heer - - die
was wel - - di - di - don
was wel don da ron di di da ri don
was wel - - der - ge - moed

Figure 7. Reference melody of *Jonkheer* (top) and melody on rank 167 (bottom).

the retrieval results considerably. The best combination of functions, combining a pitch-based, a rhythm-based and a segmentation-based scoring function, outperforms related methods from literature.

Next, we will develop scoring functions that reflect more advanced musicological models. We will use the annotations to evaluate the results by means of the quality of alignments. Since the occurrence of similar motives in related melodies was considered important by the musicological experts, we will investigate how corresponding motives can be aligned. For testing the suitability of our approach to model similarity within oral transmission in general, we will evaluate its performance on different collections. These steps contribute to the development of a similarity rater adequate for oral transmission that is based on musically advanced models of melodic similarity.

Acknowledgments. This work was supported by the Netherlands Organization for Scientific Research within the WITCHCRAFT project NWO 640-003-501, which is part of the CATCH-program. We thank Jeroen Donkers for suggesting the alignment approach, Daniel Müllensiefen and Klaus Frieler for calculating distance matrices for rawEd and diffEd, Bas de Haas for his evaluation software, and Carlos Gómez for his implementation of the Mongeau-Sankoff algorithm.

7. REFERENCES

[1] N.H. Adams, M.A. Bartsch, J.B. Shifrin and G.H. Wakefield: "Time Series Alignment for Music Information Retrieval", *Proceedings of the 5th International Conference on Music Information Retrieval*, Barcelona, 2004.

[2] J. Garbers, A. Volk, P. van Kranenburg, F. Wiering, L. Grijp, and R.C. Veltkamp: "On pitch and chord stability in folk song variation retrieval", *Proceedings of the First International Conference of the Society for Mathematics and Computation in Music*, 2007.

[3] R.L. Goldstone: "Similarity, interactive activation, and mapping", *Journal of Experimental Psychology: Learning, Memory, and Cognition*, Vol. 20, pp. 3–28, 1994.

[4] C. Gómez, S. Abad-Mota, and E. Ruckhaus: "An Analysis of the Mongeau-Sankoff Algorithm for Music Information Retrieval", *Proceedings of the 8th International Conference on Music Information Retrieval*, Vienna, Austria, pp. 109–110, 2007.

[5] O. Gotoh: "An Improved Algorithm for Matching Biological Sequences", *Journal of Molecular Biology*, Vol. 162, pp. 705–708, 1982.

[6] E. Klusen, H. Moog, and W. Piel: "Experimente zur mündlichen Tradition von Melodien", *Jahrbuch für Volksliedforschung*, Vol. 23, pp. 11–32, 1978.

[7] K. Lemström and E. Ukkonen: "Including Interval Encoding into Edit Distance Based Music Comparison and Retrieval", *Proceedings of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, Birmingham, pp. 53–60, 2000.

[8] V. Levenshtein: "Binary Codes Capable of Correcting Deletions, Insertions and Reversals", *Soviet Physics Doklady*, Vol. 10, No. 8, pp. 707–710, 1966.

[9] S.B. Needleman and C.D. Wunsch: "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *Journal of Molecular Biology*, Vol. 48, No. 3, pp. 443-53, 1970.

[10] M. Mongeau and D. Sankoff: "Comparison of musical sequences", *Computers and the Humanities*, Vol. 24, pp. 161–175, 1990.

[11] E.H. Margulis: "A Model of Melodic Expectation". *Music Perception*, Vol. 22, No. 4, pp. 663–714, 2005.

[12] D. Müllensiefen and K. Frieler: "Cognitive Adequacy in the Measurement of Melodic Similarity: Algorithmic vs. Human Judgements," *Computing in Musicology*, Vol. 13, pp. 147–177, 2004.

[13] L.A. Smith, R.J. McNab, and I.H. Witten: "Sequence-Based Melodic Comparison: A Dynamic-Programming Approach", *Computing in Musicology*, Vol. 11, pp. 101–117, 1998.

[14] A. Uitdenbogerd and J. Zobel: "Melodic Matching Techniques for Large Music Databases", *The Seventh International Multimedia Conference*, Orlando, Florida, 1999.

[15] A. Volk: "Persistence and Change: Local and Global Components of Metre Induction using Inner Metric Analysis", *Journal of Mathematics and Computation in Music*, Vol. 2, No. 2, pp. 99–115, 2008.

[16] A. Volk, P. van Kranenburg, J. Garbers, F. Wiering, R.C. Veltkamp, L.P. Grijp: "A manual annotation method for melodic similarity and the study of melody feature sets", *Proceedings of the Ninth International Conference on Music Information Retrieval*, pp. 101–106, 2008.

HETEROGENEOUS EMBEDDING FOR SUBJECTIVE ARTIST SIMILARITY

Brian McFee
Computer Science and Engineering
University of California, San Diego
bmcfee@cs.ucsd.edu

Gert Lanckriet
Electrical and Computer Engineering
University of California, San Diego
gert@ece.ucsd.edu

ABSTRACT

We describe an artist recommendation system which integrates several heterogeneous data sources to form a holistic similarity space. Using social, semantic, and acoustic features, we learn a low-dimensional feature transformation which is optimized to reproduce human-derived measurements of subjective similarity between artists. By producing low-dimensional representations of artists, our system is suitable for visualization and recommendation tasks.

1. INTRODUCTION

A proper notion of similarity can dramatically impact performance in a variety of music applications, such as search and retrieval, content-based tagging engines, and song or artist recommendation. When designing such a system, practitioners must choose an appropriate measure of similarity for the task at hand. Often, this involves selecting among multiple heterogeneous feature types, which may not be directly comparable, e.g., social network connectivity and probabilistic models of keywords. Integration of diverse features must be conducted carefully to ensure that the resulting similarity measure sufficiently captures the qualities desired for the application.

In music applications, the problem of selecting an optimal similarity measure is exacerbated by *subjectivity*: people may not consistently agree upon whether or to what degree a pair of songs or artists are similar. Even more flexible notions of similarity, such as ranking, may suffer from the effects of inconsistency, which must be understood and counteracted.

In this work, our goal is to construct artist-level similarity measures, adhering to two key principles. First, a similarity measure should integrate heterogeneous features in a principled way, emphasizing relevant features while being robust against irrelevant features. Second, instead of relying solely on features, the measure should learn from people and be optimized for the task at hand, i.e., predicting human perception of similarity. Using recently developed algorithms, we demonstrate how to learn optimal metrics for subjective similarity while seamlessly integrating multiple feature modalities. We do not mean to imply that there exists a fully consistent *ground truth* in musical similarity. Rather, we seek to construct similarity measures which are maximally consistent with human perception.

1.1 Related work

There has been a considerable amount of research devoted to the topic of musical similarity, primarily in the realms of playlist generation and recommendation [3, 14, 18]. The present work is perhaps most similar to that of Slaney, et al. [21], in which convex optimization techniques were applied to learn metric embeddings optimized according to side information. Our work differs in that we focus on artist similarity, rather than classification, and we use direct measurements of human perception to guide the optimization.

Barrington, et al. applied multiple-kernel learning to a classification task [4]. Our approach uses a different formulation of multiple-kernel learning which allows greater flexibility in assigning weights to the features and training set, and produces a metric space instead of a linear separator.

Ellis, et al. and Berenzweig, et al. studied the issue of consistency in human perception of artist similarity, and evaluated several acoustic- and socially-driven similarity measures against human survey data [5, 9]. Their work focused on the comparison of existing measures of similarity (e.g., playlist co-occurrence), rather than learning an optimal measure.

2. EMBEDDING ALGORITHMS

Our approach to the artist recommendation task is to embed each artist from a set $\mathcal{X}$ into a Euclidean space so that distances correspond to human perception of dissimilarity. Although it has been documented that notions of similarity between artists can vary dramatically from person to person, *rankings* of similarity between pairs of artists are comparatively more robust [9].

One simple ranking method involves comparisons of artists j and k relative to a fixed reference artist i. This yields similarity *triplets* (i, j, k), indicating that the pair (i, j) are more similar to each-other than the pair (i, k). Data of this variety are becoming increasingly common

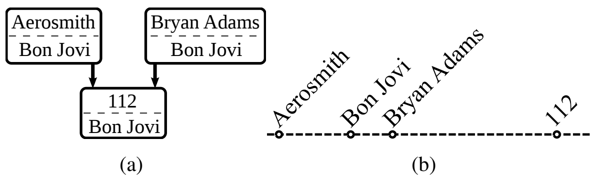


Figure 1. Similarity triplets can be interpreted as a directed graph over pairs of artists: an edge $(i, j) \to (i, k)$ indicates that i and j are more similar than i and k. (a) The graph representation of two triplets: *(Bon Jovi, Aerosmith, 112)* and *(Bon Jovi, Bryan Adams, 112)*. (b) An example of a 1-dimensional embedding that satisfies these triplets.

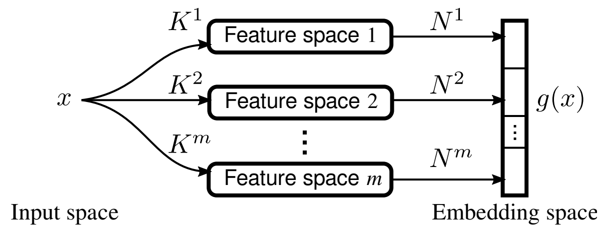


Figure 2. The embedding procedure first maps a point x into m different non-linear spaces (encoded by m different kernel matrices), and then learns a set of projections N^p ($p = 1 \ldots m$) which form the embedding space.

for general ranking and human perception modeling tasks, such as the Tag-a-Tune bonus round [12].

In this setting, we seek a Euclidean embedding function $g : \mathcal{X} \to \mathbb{R}^D$ such that each given triplet (i, j, k) yields

$$\|g(i) - g(j)\|^2 + 1 < \|g(i) - g(k)\|^2, \qquad (1)$$

where the unit margin is enforced for numerical stability. In other words, distance in the embedding space corresponds to perceived similarity. This framework eliminates the need to normalize quantitative similarity scores (as in multi-dimensional scaling), and does not over-simplify the description language to a binary problem (e.g., *same* versus *different*).

Several algorithms have been proposed to solve embedding problems in this framework [1, 16, 20]. Here, we briefly summarize the partial order embedding (POE) algorithm of [16].

2.1 Partial order constraints

A collection of similarity triplets can be equivalently represented as a directed graph in which each vertex represents a pair of artists, and a directed edge indicates a comparison of pairwise similarities (see Figure 1). Interpreting the similarity triplets as a graph allows us to simplify the embedding problem by pruning edges which may be redundant or inconsistent.

If the triplets give rise to a *directed acyclic graph* (DAG), this defines a partial order over distances, which implies the existence of some similarity space which is consistent with the measured triplets. If the graph contains cycles, then no similarity function can satisfy all of the triplets, and we say that the triplets are *inconsistent*. In practice, there are always inconsistencies in human similarity perception, but the graph representation provides a direct way to locate and quantify these inconsistencies. Section 4.1 describes an experiment and methodology to analyze inconsistencies in a collection of similarity measurements.

2.2 Multi-kernel embedding

Since our eventual goal is to recommend similar artists when presented with a previously unseen artist, we will need to provide a means to map unseen artists into the embedding space after training, without requiring any similarity measurements for the new artist. POE achieves this by restricting the choice of embedding functions to linear projections from a given feature space. This readily generalizes to non-linear embeddings through the use of *kernel functions* [19]. Artists are first mapped into a high-dimensional inner-product space by a feature transform defined by a kernel function $k(\cdot, \cdot)$. POE then learns a projection from this feature space into a low-dimensional Euclidean space. This leads to the parameterization

$$g(x) = NK_x,$$

where N is a linear projection matrix, and K_x is the vector formed by evaluating a kernel function $k(x, i)$ against all points i in the training set.

Since formulating the problem in terms of N would lead to a non-convex optimization problem — with perhaps infinitely many parameters — POE instead optimizes over a positive semidefinite matrix $W = N^{\mathsf{T}}N \succeq 0$ [6]. N may be infinite-dimensional (as is the case in Gaussian kernels), but an approximation to N can be recovered from W by spectral decomposition:

$$\begin{aligned} N^{\mathsf{T}}N = W = V\Lambda V^{\mathsf{T}} &= V\Lambda^{1/2}\Lambda^{1/2}V^{\mathsf{T}} \\ &= \left(\Lambda^{1/2}V^{\mathsf{T}}\right)^{\mathsf{T}}\left(\Lambda^{1/2}V^{\mathsf{T}}\right) = \tilde{N}^{\mathsf{T}}\tilde{N} \end{aligned} \qquad (2)$$

where V and Λ contain the eigenvectors and eigenvalues of W.

In MIR tasks, it is becoming common to combine data descriptions from multiple feature modalities, e.g., social tags and spectral features [4]. POE accomplishes this by learning a separate transformation N^p for each of m kernel matrices K^p ($p = 1 \ldots m$), and concatenating the resulting vectors (see Figure 2). This formulation allows (squared) distance computations in the embedding space to be decomposed as

$$d(i, j) = \sum_{p=1}^{m} \left(K_i^p - K_j^p\right)^{\mathsf{T}} W^p \left(K_i^p - K_j^p\right). \qquad (3)$$

The multi-kernel POE algorithm is given as Algorithm 1. The objective function has three components: the first term, $\sum_{i,j} d(i, j)$ maximizes the variance of the embedded points, which has been demonstrated to be effective for reducing dimensionality in manifold data [23]. In the present application, variance maximization diminishes erroneous recommendations by pushing all artists far away from each-other, except where prevented from doing so by similarity ordering constraints.

The second term, $-\beta \sum_{\mathcal{C}} \xi_{ijk}$, incurs hinge-loss penalties for violations of similarity constraints, scaled according to a free parameter β. The last term, $-\gamma \sum \mathrm{Tr}\left(W^p K^p\right)$, regularizes the solution and enforces sparsity in the solution, again scaled by a free parameter γ. Parameters β and γ are tuned by cross-validation, similar to the C parameter in support vector machines [7].

There are four types of constraints in Algorithm 1. The first, $d(i,j) \leq \Delta_{\mathcal{C}}$, bounds the diameter of the embedding to resolve scale invariance. [1] The second set of constraints, $d(i,j) + 1 - \xi_{ijk} \leq d(i,k)$ enforces consistency between the learned distances and similarity triplets in the training set, as in Equation 1. The slack terms $\xi_{ijk} \geq 0$ allow similarity constraints to be violated, provided it yields an overall increase in the value of the objective function. Finally, $W^p \succeq 0$ forces each W^p to be positive semidefinite, so that the N^p matrices can be recovered as in Equation 2.

The optimal solution $\left(W^1, W^2, \ldots, W^m\right)$ is computed by gradient ascent, and then each matrix is decomposed to produce the embedding function

$$g(x) = \left(N^p K_x^p\right)_{p=1}^m, \tag{4}$$

where $\left(N^p K_x^p\right)_{p=1}^m$ denotes the concatenation over all m vectors $N^p K_x^p$.

Algorithm 1 Multi-kernel partial order embedding [16]. $d(i,j)$ is defined as in Equation 3, and $\left(W^1, W^2, \ldots, W^m\right)$ are optimized by gradient ascent.

```
**Input:** kernel matrices $K^1, K^2, \ldots, K^m$,
triplets $\mathcal{C} = \{(i,j,k) : (i,j) \text{ more similar than } (i,k)\}$
**Output:** matrices $W^1, W^2, \ldots, W^m \succeq 0$.

$\max_{W^p,\xi} \sum_{i,j} d(i,j) - \beta \sum_{\mathcal{C}} \xi_{ijk} - \gamma \sum_p \mathrm{Tr}\left(W^p K^p\right)$
s.t.
    $\forall i,j \in \mathcal{X}$                 $d(i,j) \leq \Delta_{\mathcal{C}}$
    $\forall (i,j,k) \in \mathcal{C}$           $d(i,j) + 1 - \xi_{ijk} \leq d(i,k)$
                                      $\xi_{ijk} \geq 0$
    $\forall p \in 1,2,\ldots,m$              $W^p \succeq 0$
```

3. DATA

To evaluate our system, we designed experiments around the *aset400* data set of Ellis, et al [9]. The data consists of 412 popular artists, and similarity triplets collected with a web survey in 2002. We augmented the data set with several types of features, both human-derived (tags and text), and purely content-driven, as described below.

3.1 Text features

Our text-based features were collected from Last.FM between January and May of 2009. To standardize the list of artist names, we used the *search_artists* method of the Echo Nest API [17].

We then collected for each artist two types of textual features from Last.FM: biography summaries and the top 100 tags [11]. The tags were filtered by a small set of regular expressions to resolve common spelling variations. For example, *r-n-b*, *r&b*, *r-and-b* were all mapped to *rnb*, and the merged tag *rnb* received a score equal to the sum of scores of its constituent tags.

The tags and biographies were filtered by stop-word removal and stemming, resulting in dictionaries of 7737 unique tag words, and 16753 biography words. Each artist was summarized as two bags of words (one for tags and one for biographies), which were then re-weighted by TF-IDF. Finally, to compare similarity between artists, we constructed kernels K^{tag} and K^{bio} defined by the cosine similarity between word vectors.

3.2 Acoustic features

For each artist, we selected between one and ten songs at random (depending on availability), with an average of 3.8 songs per artist. From these songs, we extracted a variety of content-based features. Since content-based features relate to songs and not directly to artists, we do not expect them to perform as well the textual features described above. We are primarily interested in integrating heterogeneous features, and quantifying the improvements achieved by optimizing for artist similarity.

3.2.1 MFCC

Mel-frequency cepstral coefficients (MFCCs) have been demonstrated to capture timbral or textural qualities, and perform well in a variety of MIR applications [13, 15]. For each song, we compute the first 13 MFCCs for up to 10000 half-overlapping short-time segments (23 msec), along with the first and second instantaneous derivatives. This results in a collection of 39-dimensional delta-MFCC vectors for each song.

Each artist was summarized by modeling the distribution of delta-MFCC vectors in all songs belonging to that artist, using a Gaussian mixture model (GMM) of 8 components and diagonal covariances. Then, to compare models between artists, we construct a probability product kernel (PPK) between the GMMs:

$$K_{ij}^{\text{MFCC}} = \int \sqrt{p\left(x; \theta_i^{\text{MFCC}}\right) p\left(x; \theta_j^{\text{MFCC}}\right)}\, dx,$$

where θ_i^{MFCC} and θ_j^{MFCC} are the GMM model parameters for artists i and j [10]. Unlike kernels derived from Kullback Leibler divergence, PPK can be computed in closed-form for mixtures of Gaussians.

3.2.2 Chroma

For each song in our database, we modeled the distribution of spectral energy present in frequencies corresponding to the chromatic scale, resulting in a 12-dimensional vector for every 250 msec of audio. Although chroma features are not specifically suited to the artist similarity task, they

[1] $\Delta_{\mathcal{C}}$ is computed from the structure of the similarity triplets graph, and is not a free parameter. See [16] for details.

have been shown to work well in other applications when combined with other features, such as MFCCs [8]. We summarized each artist by collecting chroma features for each of the artist's songs, which were then modeled with a single full-covariance Gaussian distribution $\left(\theta^{\text{ch}}\right)$. From these chroma models, we construct an artist similarity kernel [2] from symmetrized KL-divergence:

$$D_{KL}(i,j) = \int p\left(x;\theta_i^{\text{ch}}\right) \log \frac{p\left(x;\theta_i^{\text{ch}}\right)}{p\left(x;\theta_j^{\text{ch}}\right)}\, dx$$

$$K_{ij}^{\text{ch}} = \exp\left(-\frac{D_{KL}(i,j) + D_{KL}(j,i)}{\mu}\right),$$

where μ is the mean KL-divergence over all pairs i, j. Since we are not using mixture models here, this can be computed in closed form.

3.2.3 Content-based auto-tagging

In contrast to the low-level acoustic features, we also evaluate high-level conceptual features which were automatically synthesized from audio content. To achieve this, we computed semantic multinomial distributions using the system described in [22]. For each song, the auto-tagger examines the acoustic content and produces a multinomial distribution over a vocabulary $\mathcal{V}$ of 149 words, e.g., *mellow*, *dance pop*, *horn section*, *etc*. The semantic model parameters θ_i^{SM} for an artist i were computed by averaging the parameters of each of that artist's song-level models. (We also tested a version using the point-wise maximum of song-level models, but it yielded little quantitative difference.) To compare models between artists, we construct a semantic multinomial kernel using the multinomial PPK:

$$K_{ij}^{\text{SM}} = \left(\sum_{x\in\mathcal{V}} \sqrt{p\left(x;\theta_i^{\text{SM}}\right)\, p\left(x;\theta_j^{\text{SM}}\right)}\right)^s.$$

This is equivalent to a homogeneous polynomial kernel of degree s over the model parameter vectors. For our experiments, setting $s = 75$ yielded reasonable results.

4. EXPERIMENTS

4.1 Quantifying inconsistency

The aset400 data set consists of 412 popular artists, and similarity triplets gathered from a web-based survey. In the survey, an informant was presented with a query artist i, and was asked to select, from a list of ten artists, the response j most similar to the query artist. Then, for each of the remaining responses k which were not selected, measurements (i, j, k) were recorded. Note that in a list of ten potential responses, there may be several "good" choices. Being forced to choose a single best response therefore results in numerous inconsistencies in the triplets, which we set out to quantify.

The survey data contains 98964 triplets, generated from 10997 queries to 713 human informants. We analyze the *filtered* version of the data, which has been reduced to 16385 measurements wherein the informant was likely to be familiar with the artists in question. Although this greatly reduces the amount of noise present in the full set, the filtered set still contains numerous inconsistencies.

[2] Symmetrized KL-divergence does not generally produce a PSD kernel matrix, but the POE algorithm is still correct for indefinite kernels.

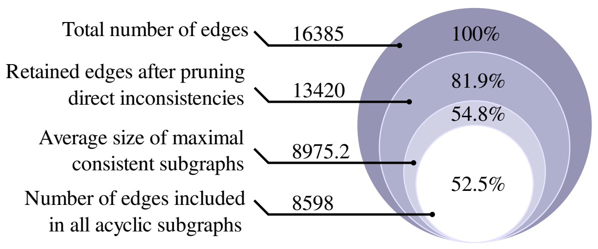


Figure 3. Quantitative summary of consistency within the aset400 filtered triplets. *Directly inconsistent* triplets are those where both (i, j, k) and (i, k, j) are present.

Consistency in the similarity measurements can be quantified by analyzing their graph representation. As a first step, we filter out all measurements (i, j, k) if (i, k, j) is also present, i.e., those artist-pairs which the informants could not consistently rank. We refer to these triplets as *directly inconsistent*. Removing these triplets decreases the number of edges by 18.1% to 13420.

However, simply removing all length-2 cycles from the graph does not ensure consistency: all cycles must be removed. Finding a maximum acyclic subgraph is NP-hard, but we can find an approximate solution by Algorithm 2. Since the algorithm is randomized, we repeat it several times to compute an estimate of the average maximal acyclic subgraph. With 10 trials, we find consistent subsets of average size 8975.2.

To evaluate the stability of these subgraphs, we count the number of edges present in all solutions, i.e., those measurements which are never pruned. Over 10 trials, 8598 edges (95.8%) were common to all solutions, leaving 4.2% variation across trials. Our results are summarized in Figure 3.

Algorithm 2 Approximate maximum acyclic subgraph

```
**Input**: Directed graph $G = (V, E)$
**Output**: Acyclic graph $G'$
$E' \leftarrow \emptyset$
**for** each $(u, v) \in E$ in random order **do**
    **if** $E' \cup \{(u, v)\}$ is acyclic **then**
        $E' \leftarrow E' \cup \{(u, v)\}$
    **end if**
**end for**
$G' \leftarrow (V, E')$
```

4.2 Order prediction

The goal of our system is to recommend similar artists in response to a query. To evaluate the system, we test its ability to predict for artists i, j and k (where i is unseen), the ordering of similarity between (i, j) and (i, k), i.e., which of the artists j or k is more similar to artist i.

4.2.1 *Experimental Setup*

We split the data for 10-fold cross validation, resulting in 370 training and 42 test artists for each fold. All directly inconsistent triplets were removed from both training and test sets, as described in Section 4.1. For each training set, we filtered the triplets to produce a maximal acyclic subgraph, retaining only those measurements which were included in all of 10 trials of Algorithm 2. The acyclic subgraphs were then pruned down to their *transitive reductions*, i.e., minimal graphs with equivalent transitivity properties [2]. This effectively removes the measurements which could be deduced from others, thereby reducing the complexity of the embedding problem with no loss of quality. The resulting training sets have an average of 6252.7 similarity measurements. The corresponding graphs have average diameter 30.2, indicating the longest contiguous chain of comparisons which can be followed in the training sets.

For each test set, we included only those triplets (i, j, k) where i is in the test set and j, k are in the training set, resulting in an average of 1149.6 triplets per test set. Aside from pruning directly inconsistent triplets, no further processing was done to enforce consistency in the test set. Therefore, we cannot expect 100% prediction accuracy on the test set. As shown in Figure 3, we can expect a lower-bound on the achievable accuracy of 67% ($8975.2/13420$). This is consistent with the upper-bound of 85% constructed in [9].

4.2.2 *Results*

We tested the embedding method on each of the kernels described in Sections 3.1 and 3.2 independently, and then combined. The free parameters β and γ were tuned by sweeping over $\beta \in [100, 10000]$ and $\gamma \in [100, 1000]$. After learning, performance was evaluated by counting the number of test-triplets correctly predicted by Euclidean distance in the embedding space.

Figure 5 illustrates two regions of an embedding produced by the combination of tags and biography features, including several query points which were mapped in after learning. The nearest neighbors of the query points provide reasonable recommendations, and the neighborhoods are generally consistent. Moreover, neighborhoods which are largely dissimilar (e.g., *female vocals* and *punk*) have been pushed to opposite extremes of the space by the variance maximization objective.

For comparison purposes, we also evaluated the prediction accuracy of distance-based ranking in the native feature spaces. Native multi-kernel results were computed by concatenating the kernels together to form feature vectors, which is equivalent to setting each $N^p = I$. This provides an intuitive and consistent way to compute distances to neighbors in one or more feature spaces.

Figure 4 lists the quantitative results of our experiments. In all cases, prediction accuracy improves significantly after learning the optimal embedding. Moreover, the improvement is more significant than it may at first seem,

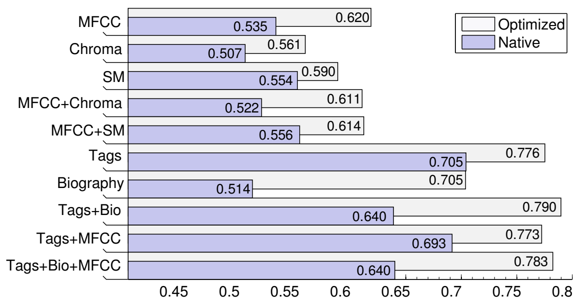


Figure 4. Triplet prediction accuracy for each feature and combinations, before and after learning.

since the maximum achievable performance is less than 100% due to inconsistencies in the test set.

It is not surprising that textual features give the best performance, and there are two main factors which explain this effect. First, only textual features were attributed directly to artists and not songs. Second, textual features derive from natural language, which is well-suited to describing subtle differences. We achieve significant improvements by optimizing the similarity metric, with gains of 7% for tags and 19% for biographies. Moreover, combining both types of textual features results in better performance than either feature on its own.

As expected, embeddings based on acoustic features perform significantly worse than those derived from text. We believe this is primarily due to the fact that acoustic features relate directly to songs, and variation across an artist's songs introduces noise to the artist-level models. Note that combining a kernel which performs poorly (e.g., chroma) does not significantly degrade the overall performance, indicating that the algorithm correctly selects the most relevant features available.

4.2.3 *Comparison*

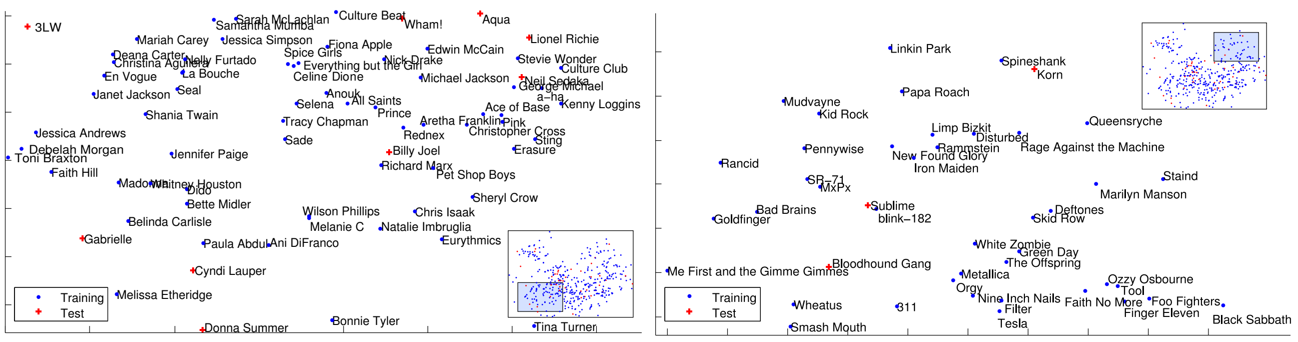
Our results can be directly compared to the "unweighted agreement" score measurements of [9]. Particularly of interest is the comparison of our biography-based embedding, which is analogous to the text-based measures in [9]. Our biography features natively achieve 51.4% accuracy, compared to 57.4% for the web documents in [9]. However, the optimized embedding improves prediction accuracy to 70.5%.

5. CONCLUSION

In this paper, we demonstrated a method for optimizing multi-modal musical similarity measures to match human perception data. We believe that the techniques illustrated here could be applicable in other subjective similarity tasks, particularly at the song level, and this will be the focus of future work.

6. ACKNOWLEDGEMENTS

We thank Luke Barrington and Douglas Turnbull for their advice. Gert Lanckriet is supported by NSF grant DMS-MSPA 0625409.



(a) Female vocalist/pop (b) Punk/metal/alternative

Figure 5. Two neighborhoods at opposite extremes of an optimized text-based embedding.

7. REFERENCES

[1] Sameer Agarwal, Joshua Wills, Lawrence Cayton, Gert Lanckriet, David Kriegman, and Serge Belongie. Generalized non-metric multi-dimensional scaling. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2007.

[2] A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

[3] Jean-Julien Aucouturier and François Pachet. Music similarity measures: What's the use? In *Inernational Symposium on Music Information Retrieval (ISMIR2002)*, pages 157–163, 2002.

[4] Luke Barrington, Mehrdad Yazdani, Douglas Turnbull, and Gert Lanckriet. Combining feature kernels for semantic music retrieval. In *International Symposium on Music Information Retrieval (ISMIR2008)*, pages 614–619, September 2008.

[5] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[8] D. Ellis. Classifying music audio with timbral and chroma features. In *International Symposium on Music Information Retrieval (ISMIR2007)*, pages 339–340, 2007.

[9] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proeedings of the International Symposium on Music Information Retrieval (ISMIR2002)*, pages 170–177, October 2002.

[10] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[11] Last.FM, 2009. http://www.last.fm/.

[12] Edith Law and Luis von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1197–1206, New York, NY, USA, 2009. ACM.

[13] B. Logan. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval (ISMIR2000)*, 2000.

[14] B. Logan. Music recommendation from song sets. In *International Symposium on Music Information Retrieval (ISMIR2004)*, 2004.

[15] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *International Symposium on Music Information Retrieval (ISMIR2005)*, pages 594–599, 2005.

[16] Brian McFee and Gert Lanckriet. Partial order embedding with multiple kernels. In *Proceedings of the Twenty-sixth International Conference on Machine Learning*, 2009.

[17] The Echo Nest, 2009. http://www.echonest.com/.

[18] E. Pampalk, A. Rauber, and D. Merkl. Content-based Organization and Visualization of Music Archives. In *Proceedings of the ACM Multimedia*, pages 570–579, Juan les Pins, France, December 1-6 2002. ACM.

[19] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[20] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[21] M. Slaney, K. Weinberger, and W. White. Learning a metric for music similarity. In *International Symposium on Music Information Retrieval (ISMIR2008)*, pages 313–318, September 2008.

[22] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

[23] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 839–846, 2004.

THE INTERSECTION OF COMPUTATIONAL ANALYSIS AND MUSIC MANUSCRIPTS: A NEW MODEL FOR BACH SOURCE STUDIES OF THE 21ST CENTURY

Masahiro Niitsuma† **Tsutomu Fujinami**‡ **Yo Tomita**†
†School of Muisc and Sonic Arts, Queen's University, Belfast
‡School of Knowledge Science, Japan Advanced Institute of Science and Technology (JAIST)
niizuma@nak.ics.keio.ac.jp, fuji@jaist.ac.jp, y.tomita@qub.ac.uk

ABSTRACT

This paper addresses the intersection of computational analysis and musicological source studies. In musicology, scholars often find themselves in the situation where their methodologies are inadequate to achieve their goals. Their problems appear to be twofold: (1) the lack of scientific objectivity and (2) the over-reliance on new source discoveries. We propose three stages to resolve these problems, a preliminary result of which is shown. The successful outcome of this work will have a huge impact not only on musicology but also on a wide range of subjects.

1. INTRODUCTION

Recent developments in computer and information technology have brought significant changes to the ways in which we conduct research in a wide range of domains, and musicology is not an exception.

Yet in historical musicology the majority of scholars still conduct their research without making full use of this technological advancement, thus creating huge potential for future advancement.

By nature, their research methods are less scientific, i.e. they tend not to, or find it impossible to disclose all the information they used in order to arrive at their conclusions, and hence it is often difficult to verify their findings regardless of whether or not there are elements of subjective judgment in them.

There is a separate problem in musicology in that the majority of source-based studies heavily rely on the rediscovery of new sources.[1] Thus, if a new source is not found, there is often little discussion to challenge the existing interpretation offered by scholars in the past. Is there really no way of improving the theories unless a new source is discovered? How can a computer assist musicologists in analysing the information contained in the known sources?

The main objective of this study is to solve such problems in historical musicology by addressing the following questions:

1. Can computational analysis offer the same conclusions as those arrived at by historical musicologists?
2. Are there any oversights in the musicologists' analysis of the sources?

To achieve our objectives, it is necessary to address the following issues:

1. How to define a data structure for storing Bach's manuscripts in digital format;
2. How to extract information from the digitised manuscripts;
3. How to analyse the extracted information.

This paper is structured as follows: Section 2 describes the relationship between the proposed methods and existing scholarly debates in the field; Section 3 discusses the research methods to be employed; Section 4 shows a preliminary result of the proposed method; Section 5 illustrates the contribution that the proposed research will make; and Section 6 offers concluding remarks.

2. PREVIOUS RESEARCH

There are numerous research projects dealing with computation in musicology and different kinds of data formats have been proposed to encode musical data [1–3]. However, all of them deal with limited musical information such as pitch or rhythm derived from printed scores, and the majority of previous research on computational music analysis [4–9] is based on those data formats.

There is also a certain amount of research related to automatic music analysis using the signal-processing technique with acoustic sources [10–14], which record musical performance from published scores. But if we investigate only published scores, rather than the original manuscripts, we miss important information that has been lost in the process of creating an edition.

[1] Sources refer to manuscript sources, that is written scores by hand. Before the invention of printing, music was preserved either by oral transmission or by MS copies.

Recent journal articles or proceedings of ISMIR [15–17] includes a considerable number of researches on the Optical Music Recognition (OMR). Most of them deal with staff removal algorithm, which eases the preprocessing of the digitised images of the manuscripts such as the music symbol recognition.

With regard to the research related to manuscript analysis, Tomita developed a database of variants and errors which supposedly lists all the extant manuscripts and early prints of the Well-Tempered Clavier II, a work well known for its complex history of compilation, revision and transmission [18]. The database contains all kinds of information extracted from manuscripts – not only musical variants but also notational errors and variants that may have been inherited from its model or may cause errors when fresh copies were made from it – giving us many insights into how the future database should be developed.

3. METHODOLOGY

There are three stages in this project:

1. To define of a data structure for storing Bach's manuscripts in digital format;
2. To develop a methodology to automatically extract data from the digitised images of music manuscripts;
3. To develop a methodology to analyse these data to find significant information for musicological study.

In the first instance, a data structure that is appropriate to be analysed by computers needs to be defined. This data structure should be designed in such a way that it can encode all the information extracted from manuscripts – not only musical aspects such as pitch or rhythm, but also the physical aspects of the manuscript which may account for the scribe's unintentional omissions, misplacement, superfluous symbols that were somehow caused by the appearance of its exemplar. This has been investigated with the collaboration of musicologists.

Secondly, a method will be developed to harvest the information useful for research from the digitised images of the manuscripts. At the moment, we consider primarily the visible information such as the direction of stems or the position of note-heads. The first task is the recognition of each music symbol such as staff line, bar line, note stem, note head and clef. The Gamera [19] framework will be used for this task.

Finally, a method to analyse the data will be proposed. In order to achieve this, powerful machine learning methods such as bagging [20], boosting [21], and random forest [22] will be adopted.

Figure 1 illustrates how the proposed method operates. First, a digitised image file is created by physically scanning the manuscripts. Secondly, symbolic data is extracted from the digitised image file. Thirdly, computational analysis is carried out using the symbolic data.

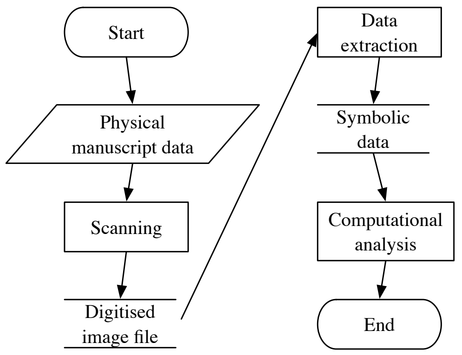


Figure 1. Flowchart of the proposed method

4. PRELIMINARY EXPERIMENT

4.1 An overview of the preliminary experiment

This sections presents a preliminary result of the third stage described under "3. Methodology". Currently, the first and second stages are conducted manually, while the program was developed for the third stage. To demonstrate the performance of the latter, the simplest example would be to examine the origin and authenticity of variants. Because WTC II was so popular among Bach's pupils and admirers during and after his lifetime, numerous manuscripts were made, copied and edited, which not only increased the number of errors or variant readings, but also resulted in introducing contamination to the texts in some sources [23, 24]. This program produces a source affiliation diagram showing how closely these sources were related, taking into account the differences that may be caused either by accident or on purpose while being copied.

In this paper, we focus on the sources of Viennese origin, which are considered to have been originated from a copy that was brought from Berlin to Vienna in 1777 by Gottfried van Sweieten (1734-1803). How the unique text of the Viennese sources evolved up has been the principal interest for musicologists, for this was the state of musical text which Mozart learned in 1782. In [25], Tomita investigated the Viennese sources, thereby proposing a source affiliation diagram of them, an excerpt of which is shown in Figure 2.

4.2 Preliminary result

We describe one approach to this task using the database developed by Tomita [24], an excerpt of which is shown in Figure 3, where S/N is the serial number given to each examination point; Bar indicates in which measure(s) the elements are examined; V, bt/pos stands for Voice, Beat and Position, respectively; Element specifies the target of enquiry; Spec. Loc gives graphic representation of information under examination; Classified suggests text-critical significance.

Firstly, the distance between two manuscripts should be defined. The simplest way is to count the number of different factors between two manuscripts.

In Figure 3, "Q11731" has no different factors from

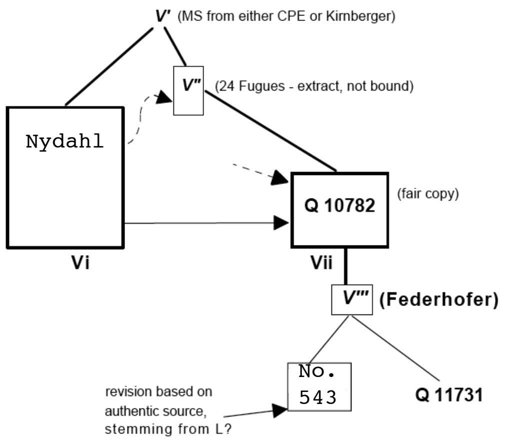
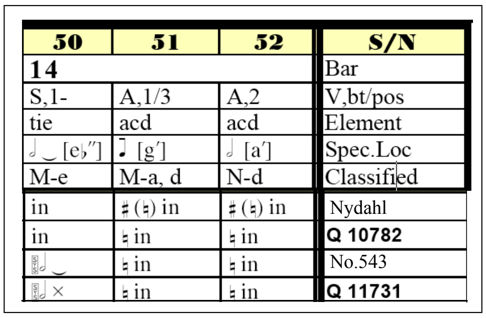

V' (MS from either CPE or Kirnberger)
V'' (24 Fugues - extract, not bound)
Nydahl
Q 10782
(fair copy)
Vi
Vii
V''' (Federhofer)
No. 543
Q 11731
revision based on authentic source, stemming from L?

Figure 2. Score affiliation diagram of the Well-tempered Clavier Book Ⅱ, generated by human analysis (excerpted from [18])

those of "No.543", thus the distance between "Q11731" and "No.543" is 0. On the other hand, "Q11731" has three factors which are different from those of "Nydahl", thus the distance between "Q11731" and "Nydahl" is 3. However, such observation dose not reflect the reality sufficiently. To improve the accuracy of observation, we should consider how easily each factor can change. For instance, notational factors such as the direction of the stem or position of the note-head are more likely to change than musical factors such as pitch or duration. Taking this into consideration, genealogical distance is defined by the following equation,

$$D(MSS1, MSS2) = \sum_{i=1}^{SN} \alpha_{Type_i} I(MSS1[i], MSS2[i]) \qquad (1)$$

where, $MSS1$ and $MSS2$ denote two different manuscripts, $MSS[i]$ denotes the ith content of MSS, α_{Type_i} is the weight considering the fluidity of each type of the content, and $I(x, y)$ is the indicator function which returns 0 if $x = y$ else 1. In this paper, all α_{Type_i} were equalized, leaving an adjustment of α_{Type_i} as a future task.



50	51	52	S/N
14			Bar
S,1-	A,1/3	A,2	V,bt/pos
tie	acd	acd	Element
𝅗𝅥 ‿ [e♭″]	♪ [g′]	𝅗𝅥 [a′]	Spec.Loc
M-e	M-a, d	N-d	Classified
in	♯ (♮) in	♯ (♮) in	Nydahl
in	♮ in	♮ in	**Q 10782**
𝅗𝅥 ‿	♮ in	♮ in	No.543
𝅗𝅥 ×	♮ in	♮ in	**Q 11731**

Figure 3. Database used for the experiment (excerpted from [18]).

Secondly, manuscripts are clustered by a hierarchical cluster analysis using a set of dissimilarities calculated on the basis of Equation (1). Initially, each manuscript is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster. At each stage distances between clusters are recomputed by the Lance-Williams dissimilarity update formula according to the complete linkage method.

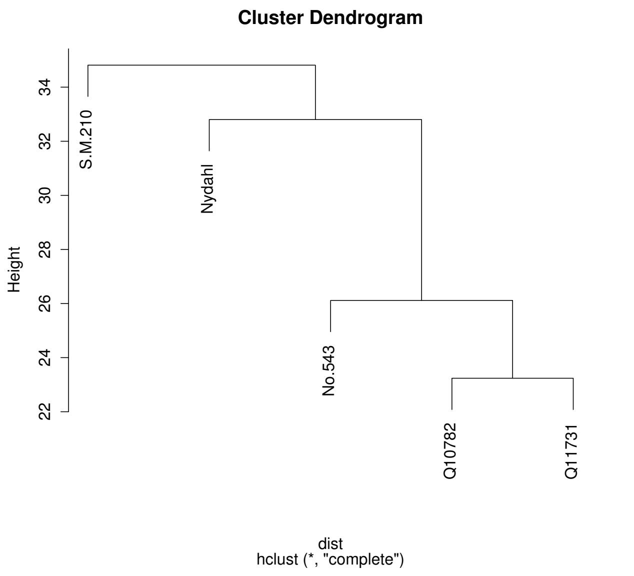


Figure 4. Score affiliation diagram of Fugue No.22 in B♭ minor from the Well-tempered Clavier Book Ⅱ, generated by computational analysis

Figure 4 illustrates an example of source affiliation diagram automatically generated by the proposed algorithm. Manuscripts of Fugues 10, 12 and 14 were used to calculate the distance between each manuscript. This result is almost consistent with that of human analysis, while the position of No.543 (Berea) is considered to be different. This result indicates that this database is sufficient to achieve a rough classification; but to achieve a more reliable classification or for further analysis, it is necessary to develop a new data structure that is suitable for a more detailed computational analysis. The manual weighting of α_{Type_i} can reflect the expert knowledge of musicologists; however it could also reflect their own subjectivity. To exclude it, a method for automatic weighting of these factors should be investigated.

There are numerous possibilities of using these databases for analysis and the potential is far-reaching. Figure 5 shows biplot of the result of the principle component analysis. This reveals that there exists a large gap between Add.35021 (Bach's autograph manuscript) and the Viennese sources. Figure 6 shows the result of the variable importance estimation for the classification of the manuscripts of Fugue 23 by random forest, where y-axis corresponds to S/N of the text critical database. This indicates that S/N 475, and 136 are important for computer to classify them. These analyses using appropriate databases are considered

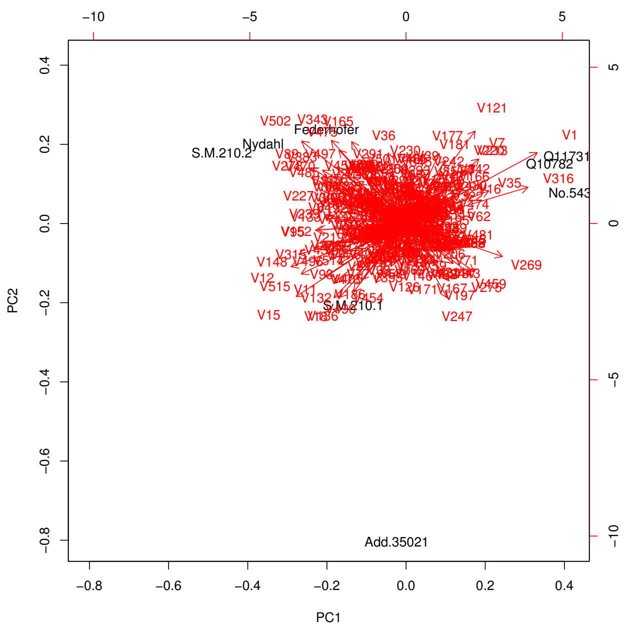


Figure 5. Biplot produced from the output of the principle component analysis of the text critical database of Fugue 23

to bring the objectivity and new findings to historical musicology.

Another area of investigation is an automatic handwriting analysis. The method for identifying handwriting in noisy document images [26] cannot directly be applied to music manuscripts. This is because handwriting identification needs not only visual information such as curvature (which represents the shape of the curves or bending angle) but also multifaceted information such as the purpose for which a manuscript was written, the scribe's habits, the conditions under which the manuscript was made, and so on. The proposed method is expected to overcome such difficulties by taking into account the multifaceted information with the appropriate database for computational analysis.

5. CONTRIBUTION

This research makes main contributions in the following areas:

1. The proposed method will provide a way to verify previous research in historical musicology;
2. It will be possible to offer new information about the sources from the already known sources;

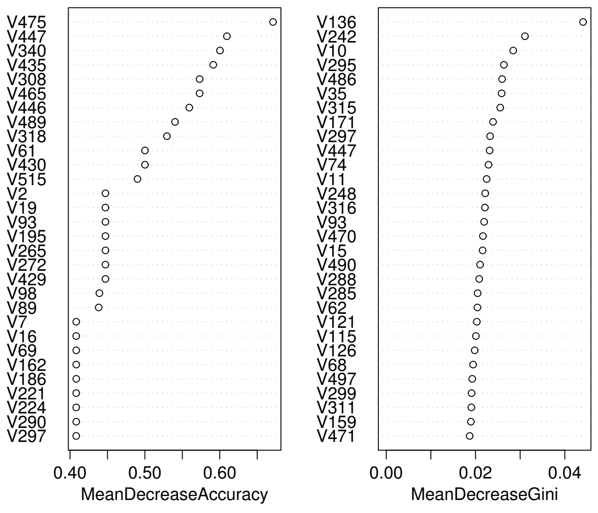


Figure 6. Result of variable importance estimation for the classification of Viennese sources by a random forest, where y-axis corresponds to S/N of Fugue 23 shown in [18]: for example, V475 is notation difference of rest in bar 89; V136 is the existence of accidental in bar32.

3. The proposed method can be a prototype of an empirical research method.

The result of the proposed research has a good potential for becoming a road map for musicological research of the future, and empirical research method would offer an alternative to the previous research methods often criticised for their inherent subjectivism. Consequently, it is hoped that the majority of previous research may be reworked by using the proposed methods. In this process, new discoveries can still be made that would shed new light on the musical works concerned without requiring the rediscovery of new sources. Moreover, the results of the proposed research may also serve as a prototype in other areas of research, such as archaeology, historical literature or other social science subjects that involve the study of historical sources.

6. CONCLUSION

In this paper, we have shown the necessity of using the computational approach in source studies. We also addressed the problems of subjective attitudes and its over-reliance on new source discoveries in traditional research methods in musicology. Three stages that may resolve these problems have been discussed. The outcome of this work should affect not only musicology but also a wide range of subjects.

7. REFERENCES

[1] Content-based unified interfaces and descriptors for audio/music databases available. http://www.cuidado.mu.

[2] Online music recognition and searching. http://www.elec.qmul.ac.uk/research/projects/ nsf_9905842_ omras.html.

[3] Web delivering of music. http://www.wedelmusic.org.

[4] Greg Aloupis, Thomas Fevens, Stefan Langerman, Tomomi Matsui, Antonio Mesa, Yurai Nuez, David Rappaport, Godfried, and Toussaint. Algorithms for computing geometric measures of melodic similarity. *Computer Music Journal*, 30(3):67–76, 2006.

[5] David Huron. Music information processing using the humdrum toolkit: Concepts, examples and lessons. *Computer Music Journal*, 26(2):11–26, 2002.

[6] Steven Jan. Meme hunting with the humdrum toolkit: Principles, problems and prospects. *Computer Music Journal*, 28(4):68–84, 2004.

[7] David Meredith. The ps13 pitch spelling algorithm. *Journal of New Music Research*, 35(2):121–159, 2006.

[8] Wai Man Szeto and Man Hon Wong. A graph-theoretical approach for pattern matching in post-tonal music analysis. *Journal of New Music Research*, 35(4):304–321, 2006.

[9] Heinrich Taube. Automatic tonal analysis: Toward the implementation of a music theory workbench. *Computer Music Journal*, 23(4):16–32, 1999.

[10] Lee Kyogu and Slaney Malcolm. Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio. *IEEE Transactions on audio, speech, and language processing*, 16(2):291–301, 2008.

[11] Olivier Lartillot. A musical pattern discovery system founded on a modeling of listening strategies. *Computer Music Journal*, 28(3):56–67, 2004.

[12] Pierre Leveau, Emmanuel Vincent, and Gal Richard. Instrument-specic harmonic atoms for mid-level music representation. *IEEE Transactions on audio, speech, and language processing*, 16(1):116–128, 2008.

[13] Rui Pedro Paiva, Teresa Mendes, and Amlcar Cardoso. Melody detection in polyphonic musical signals: Exploiting perceptual rules, note salience, and melodic smoothness. *Computer Music Journal*, 30(4):80–89, 2006.

[14] Li Yipeng and Wang DeLiang. Musical sound separation using pitch-based labeling and binary time-frequency masking. *Proceedings of International Conference on Acoustics Speech and Signal Processing*, pages 173–176, 2008.

[15] C. Dalitz, B. Czerwinski M. Droettboom, and I. Fujinaga. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):753–66, 2008.

[16] J. Hockman Pugin, L., J.A. Burgoyne, and I. Fujinaga. Gamera versus aruspix: Two optical music recognition approaches. *Proceedings of the International Conference on Music Information Retrieval*, pages 419–424, 2008.

[17] John Ashley Burgoyne, Johanna Devaney, Laurent Pugin, and Ichiro Fujinag. Enhanced bleedthrough correction for early music documents with recto-verso registration. *Proceedings of the International Conference on Music Information Retrieval*, pages 407–412, 2008.

[18] Yo Tomita. *J. S. Bach's 'Das Wohltemperierte Clavier II' A Critical Commentary. Volume II: All the Extant Manuscripts*. Leeds: Household World, 1995.

[19] The Gamera framework for building custom recognition systems. Michael droettboom and karl macmillan and ichiro fujinaga. *Proceedings of the Symposium on Document Image Understanding Technologies*, pages 275–86, 2003.

[20] L Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[21] Y Freud and R.E Schapire. Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[22] L Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[23] Yo Tomita. Breaking the limits: some consideration on an e-science approach to source studies. *Musicology and Globalization. Proceedings of the international Congress of the Musicological Society of Japan*, pages 233–237, 2002.

[24] Yo Tomita and Tsutomu Fujinami. Managing a large text-critical database of Johann Sebastian Bach's Well-Tempered Clavier II with XML and relational database. *Proceedings of the International Musicological Conference*, 2002.

[25] Yo Tomita. The Sources of J.S. Bach's Well-Tempered Clavier II in Vienna, 1777-1801. *BACH: Journal of the Riemenschneider Bach Institute*, 29(2):8–79, 1998.

[26] Yefeng Zheng, Huiping Li, and David Doermann. Machine printed text and handwriting identification in noisy document images. *Pattern Analysis and Machine Intelligence*, 26(3):337–353, 2004.

ON RHYTHM AND GENERAL MUSIC SIMILARITY

Tim Pohle[1]**, Dominik Schnitzer**[1,2]**, Markus Schedl**[1]**, Peter Knees**[1] **and Gerhard Widmer**[1,2]
[1]Department of Computational Perception, Johannes Kepler University, Linz, Austria
[2]Austrian Research Institute for Artificial Intelligence (OFAI), Wien, Austria
`music@jku.at`

ABSTRACT

The contribution of this paper is threefold:

First, we propose modifications to Fluctuation Patterns [14]. The resulting descriptors are evaluated in the task of rhythm similarity computation on the "Ballroom Dancers" collection.

Second, we show that by combining these rhythmic descriptors with a timbral component, results for rhythm similarity computation are improved beyond the level obtained when using the rhythm descriptor component alone.

Third, we present one "unified" algorithm with fixed parameter set. This algorithm is evaluated on three different music collections. We conclude from these evaluations that the computed similarities reflect relevant aspects both of rhythm similarity *and* of general music similarity. The performance can be improved by tuning parameters of the "unified" algorithm to the specific task (rhythm similarity / general music similarity) and the specific collection, respectively.

1 INTRODUCTION

Many of the rhythm descriptors proposed so far eventually reduce the rhythm to a representation that discards information about which frequency band the rhythmic feature originates from. We begin this paper by asking: *"Can the performance of rhythm descriptors be improved by adding frequency information?"* To this end, we follow two directions. First, we propose and evaluate descriptors that retain information about the frequency range in which a given rhythm feature (more precise: periodicity strength) was measured. Related work in this direction includes [10]. Second, we add frequency information in the form of a "timbral" component (cf. [3]).

The paper is organized as follows. In Section 2, we suggest a number of modifications to Fluctuation Patterns (FPs) [14]. Relative to our evaluation setting, the modified variant seems to capture rhythmic similarity better than the unmodified algorithm. In Section 3, we go on by adding frequency information to the proposed rhythm descriptors in the form of a "timbral" component, and find that in our evaluation setting, rhythm similarity computation is improved further this way. We consider this finding as complementary to the practice of using rhythm descriptors to improve the performance of (general) music similarity measures (e.g., [14]). Based on this observation, we design an algorithm that seems to perform well *both* in the task of rhythm similarity *and* in the task of general music similarity computation (Section 4). In our evaluation setting, this combined algorithm outperforms approaches that are specifically designed for the respective tasks.

2 GETTING THE RHYTHM

This section is dedicated to rhythm descriptors and their evaluation on the Ballroom Dancers collection.

2.1 Rhythm Descriptors

Below, the rhythm descriptors evaluated in this paper are described. These are the well-known Fluctuation Patterns, and our proposed extensions *Onset Patterns* (OPs) and *OnsetCoefficients* (OCs).

2.1.1 *Fluctuation Patterns (FPs)*

Fluctuation Patterns (FPs) [14] measure periodicities of the loudness in various frequency bands, considering a number of psychoacoustic findings. We use the implementation of the MA Toolbox [1] with the proposed parameter set, so that the frequency bands correspond to 20 critical bands. Details about the computation are given e.g. in [14]. An evaluation of the importance of the various psychoacoustic processing steps in FP calculation is given in [10].

2.1.2 *Onset Patterns (OPs)*

We suggest a number of changes to FPs (cf. [4, 17, 18]). To this end, a number of preliminary experiments was conducted. The most important changes to FPs are listed here, before the points are discussed in detail:

- Reduce the signal to the parts of increasing amplitude (i.e., likely onsets).
- Use semitone bands to detect onsets instead of fewer critical bands.
- Use Hanning window and zero padding before detecting periodicities with FFT.

[1] http://www.pampalk.at/ma/

- Represent periodicity in log scale instead of linear scale.

We only consider onsets (or increasing amplitudes) in a given frequency band. To detect such onsets, we use a cent-scale representation of the spectrum with 85 bands of 103.6 cent width, with frames being 15.5 ms apart. On each of these bands, an *unsharp-mask* like effect is applied by subtracting from each value the mean of the values over the last 0.25 sec in this frequency band, and half-wave rectifying the result. This aims to detect also slow-attack instrument onsets in melodies that have notes with only one (or few) semitones apart. Subsequently, values are transformed by taking the logarithm, and reducing the number of frequency bands from 85 to 38 which is closer to the number of critical bands.

As in the computation of FPs, segments of frames are analyzed for periodicities. We use segments of 2.63 sec length with a superimposed Hanning window, zero-padded to six seconds. Adjacent segments are 0.25 sec apart. Each of these segments is analyzed for periodicities in the range from $T_0 = 1.5$ sec up to about 13.3 Hz (40 to about 800 bpm), separately in each of the 38 frequency bands. A crucial point in this transformation is that we do not represent periodicities on a linear scale (as in FPs), but rather we use a log-representation. Thus, after taking the FFT on the six seconds of a given frequency band, a log filterbank is applied to represent the selected periodicity range in 25 log-scaled bins. In this representation, periodicity (measured in Hz) is doubled every 5.8 bins (i.e., going 6 bins to the right means measuring a periodicity about twice as fast). By using this log scale, all activations in an OP are shifted by the same amount in the x-direction when two pieces have the same onset structure but different tempi. While this representation is not blurred (as done in the computation of FPs), the applied techniques induce a smearing in the lower periodicity range (cf. Figure 1). After a segment is computed, each of the 25 periodicities is normalized to have the same response to a broadband noise modulated by a sine with the given periodicity. This is done to eliminate the filter effect of the onset detection step and the transformation to logarithmic scale.

To arrive at a description of an entire song, the values over all segments are combined by taking the mean of each value over all segments. We call the resulting representation of size $38 \cdot 25$ *Onset Patterns* (OPs). In this paper, the distance between OPs is calculated by taking the Euclidean distance between the OPs considered as column vectors.

2.1.3 OnsetCoefficients (OCs)

OnsetCoefficients are obtained from all OP segments of a song by applying the two-dimensional discrete cosine transformation (DCT) on each OP segment, and discarding higher-order coefficients in each dimension. The DCT leads to a certain abstraction from the actual tempo (cf. [5, 18]) *and* from the frequency spectrum (like in MFCCs). This is motivated by the notion that slightly changing rhythm and sounds does not have a big impact on the perceived characteristic of a rhythm, while the same rhythm played with a drastically different tempo may have a different perceived characteristic. For example, one can imagine that a slow and laid-back drum loop, used in a Drum'n'Bass track played back two or three times as fast, is perceived as cheerful.

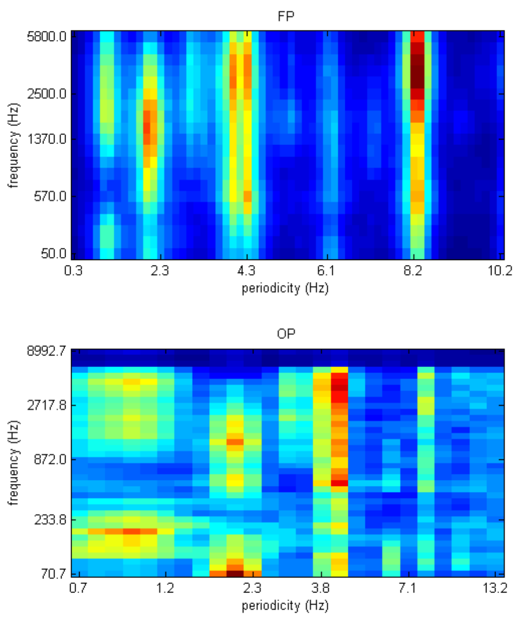


Figure 1. FP and OP of the same song. Doubling of periodicity appears evenly spaced in the OP. A bass drum plays at regular rate of about 2 Hz. The piece has a tap-along tempo of about 4 Hz, while the measured periodicities at about 8 Hz are likely caused by offbeats in between taps.

The number of DCT coefficients kept in each dimension (periodicity / frequency) is an important parameter. The selected coefficients are stacked into a vector. For example, keeping coefficients 0 to 7 in the periodicity dimension, and coefficients 0 to 2 in the frequency dimension yields a vector of length $8 \cdot 3 = 24$. We abbreviate this selection as 7×2. Based on the vectors for all segments, the mean and full covariance matrix (i.e, a single Gaussian) is calculated, which is the OC feature data for a song.

The OC distance D between two Songs (i.e., Gaussians) X and Y is calculated by the Jensen-Shannon (JS) divergence (cf. [11]).

$$D(X, Y) = H(M) - \frac{H(X) + H(Y)}{2} \qquad (1)$$

where H denotes the entropy, and M is the Gaussian resulting from merging X and Y. We calculate the merged Gaussian following [20]. We use the square root of this distance.

2.2 Setup for Rhythm Experiments

We evaluate the rhythm descriptors on the ballroom dance music set [2] previously used by other authors, e.g. [5, 4, 2,

[2] data from ballroomdancers.com

15, 7] and for the ISMIR'04 Dance Music Classification Contest[3]. This set consists of 698 tracks assigned to 8 different dance music styles ("genres"). The classification baseline is 15.9%.

The purpose of the descriptors discussed above is to measure *rhythmic similarity*. For evaluation, we assume that tracks that are in the same class have a similar rhythm. To facilitate comparison to previous work [5, 4], we use a 1-nearest-neighbor (1NN) stratified 10-Fold cross validation (averaged over 32 runs) in spite of a certain variance induced by the random selection of folds. We assume that the only information that is available is the audio signal. Using 1NN 10fold cross validation, [5] report up to 79.6% accuracy.

When using more sophisticated classification algorithms (and other features), higher accuracies are obtained. For example, [2] report a classification accuracy of up to 82% using only automatically computed features (i.e., without using correct tempo annotation or manually corrected first bar annotations). The highest classification accuracy we are aware of is 86.9%, obtained by kNN classification [7].

The mentioned accuracies are obtained when the audio signal is the only data source made available to the algorithms. It has to be noted that the algorithms yield higher accuracies when also the *correct* tempo annotation is given as feature data. In this case (which is not considered in this paper), an accuracy of 95.1% (or 96.0% when also human-corrected bar annotations are used [2]) have been obtained.

2.3 Results for Rhythm-Only Descriptors

FPs as implemented in the MA toolbox, compared by Euclidean distance, yield an accuracy of **75.0%**. OPs compared with Euclidean distance yield **86.7%**. The results for various settings of using only OnsetCoefficients for similarity estimation are shown in Figure 2. It can be seen that the highest values are obtained when keeping more than 16 coefficients in the periodicity dimension and when only keeping the 0th coefficient in the frequency dimension (which corresponds to averaging over all frequencies). In this range, values increase when including more periodicity coefficients, which seems consistent with the findings in [5]. In this range, we obtain an average value of **87.7%**[4].

3 ADDING "TIMBRE" INFORMATION

To examine how the discussed rhythmic descriptors can be used in conjunction with "bag of frames" audio similarity measures, we combine them with a "timbral" audio similarity measure. The used frame-based features are the well-known MFCCs (coefficients 0..15), Spectral Contrast Coefficients [9] (using the 2N approach [1], coefficients 0..15), and the descriptors *Harmonicness* and *Attackness*. The latter two describe the amount of harmonic and percussive elements (cf. [13]) in a cent-scaled spectrogram with frequency bands being 66 cent and frames being 46 ms apart. Percussive elements are detected by applying a 5×5 filter with the kernel (-0.14, -0.06, 0.2, 0, 0) replicated over five rows. The analogous filter to detect harmonic elements has the form $(-0.09, -0.01, 0.2, -0.01, -0.09)^T$, replicated over five columns. The Harmonicness value for a frame is the sum of the half-wave rectified responses of this filter centered at the frequency bins of the considered frame. The frame's Attackness value is calculated the same way but using the filter for percussive elements. Altogether, these are 34 descriptor values for a frame, which are combined over a song by taking their mean and full covariance matrix. Two songs are compared by taking the Jensen-Shannon divergence as described above.

We combine the discussed rhythm descriptors with this timbral component by simply summing up the two distance values (i.e., timbral and rhythm component are weighted $1 : 1$). For comparison, e.g., in the G1C algorithm [14], FP based features are weighted with 30%, and a MFCC component is weighted with 70%. Our weighting decision is not based on systematic evaluations but rather it is mainly based on impressions gained from non-representative listening experiments. To bring the two distances (rhythm based and timbre based) to a comparable magnitude, for each song the distances of this song to all other songs in the collection are normalized by mean removal and division by standard deviation[5]. Subsequently, the distances are symmetrized by summing up the distances between each pair of songs in both directions. This preprocessing step is done for each component (timbral and rhythm) independently before summing them up.

3.1 Combination Experiment

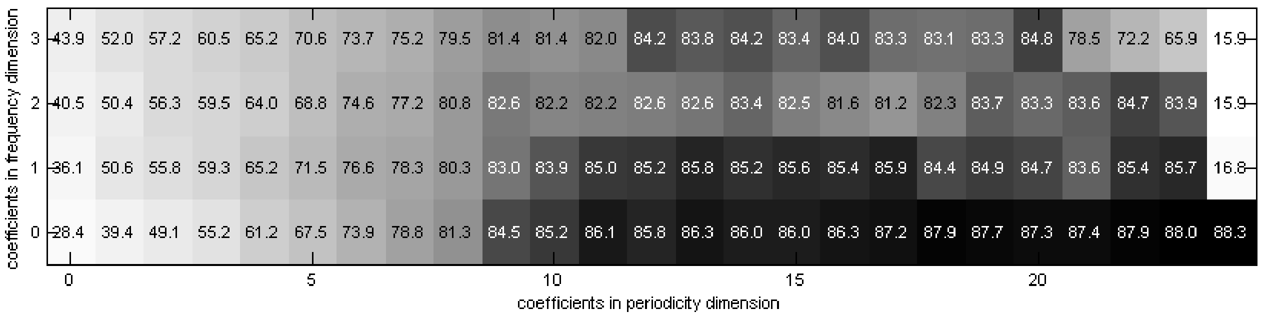
We repeat the experiment shown in Figure 2, but this time combining the rhythm descriptors with the timbral component as described. The 1NN 10fold cross validation accuracy is 54.0% when considering only the timbral component, 79.4% in combination with FPs, and 87.1% with OPs. From the results in Figure 3, it can be seen that classification results are improved when combining OCs with the timbral component. This time, average results of 90.2% are obtained over the parameter range discussed above (compared to 87.7% in the the first experiment, Figure 2). The highest obtained 1NN accuracy is 91.3%.

Results are summarized in Table 1. The results for the combined method are above the values obtained for each component (rhythm and timbre) alone. We think this is an indication that rhythm similarity computations can be improved by including timbre information. This is in line with [19] who reason that tempo can be detected better when considering timbre information. In a way, this is complementary to previous approaches where descriptors of rhythmical properties were added to timbre descriptors in order to improve music similarity computations (e.g. the

[3] http://mtg.upf.edu/ismir2004/contest/rhythmContest/

[4] We take the average rather than the maximum value as an indicator due to variances introduced by 10fold CV.

[5] This is done once before splitting up training and test sets for classification. No class labels are used in this step. We expect the impact of determining the normalization factors only on the respective (stratified) training set to be negligible.



3	43.9	52.0	57.2	60.5	65.2	70.6	73.7	75.2	79.5	81.4	81.4	82.0	84.2	83.8	84.2	83.4	84.0	83.3	83.1	83.3	84.8	78.5	72.2	65.9	15.9
2	40.5	50.4	56.3	59.5	64.0	68.8	74.6	77.2	80.8	82.6	82.2	82.2	82.6	82.6	83.4	82.5	81.6	81.2	82.3	83.7	83.3	83.6	84.7	83.9	15.9
1	36.1	50.6	55.8	59.3	65.2	71.5	76.6	78.3	80.3	83.0	83.9	85.0	85.2	85.8	85.2	85.6	85.4	85.9	84.4	84.9	84.7	83.6	85.4	85.7	16.8
0	28.4	39.4	49.1	55.2	61.2	67.5	73.9	78.8	81.3	84.5	85.2	86.1	85.8	86.3	86.0	86.0	86.3	87.2	87.9	87.7	87.3	87.4	87.9	88.0	88.3

Figure 2. Dance genre classification based on OnsetCoefficients; distances calculated according to Equation 1. 1NN 10fold CV accuracies obtained on ballroom dataset when including coefficients 0 up to the given number in the respective dimension. For example, including coefficients 0..17 in the periodicity dimension and coefficients 0..1 in frequency dimension (resulting in $18 \cdot 2 = 36$ dimensional feature data) yields an accuracy of 85.9%. Low results at right border are caused by numerical instabilites when calculating the determinant during entropy computation. For better visibility, gray shades indicate ranks instead of actual values.

Algorithm	1NN
Baseline	15.9%
FP	75.0%
OP	86.7%
OC	up to around 87.7%
Timbre	54.0%
Timbre+FP	79.4%
Timbre+OP	87.1%
Timbre+OC	up to around 90.2%

Table 1. Ballroom dataset: 10fold CV accuracies obtained by the evaluated methods. The methods below the line are combined by distance normalization and addition.

G1C algorithm [14]). This duality leads to the experiments presented next.

4 THE "UNIFIED" ALGORITHM

Encouraged by the experiments presented in the previous section, we examine the performance of this algorithm not only in the task of *rhythm* similarity computation, but also in the task of general music similarity. Our aim is to find a selection of OCs that perform well in both tasks, which eventually leads to a "unified" music retrieval algorithm that reflects both rhythm and timbre similarity.

4.1 Data Sets

Music similarity experiments are performed on the set from the ISMIR'04 genre classification contest (ISMIR'04)[6], and on the "Homburg" data set (HOMBURG) [8]. Like the ballroom set, these collections are available to the research community, which facilitates reproduction of experiments and gives a benchmark for comparing different algorithms. There are two variants of the ISMIR'04 collection. The first is the "training" set which consists of 729 tracks from six genres. The second consists of all the tracks in the "training" and "development" sets, which are 1458 tracks from six genres. We use the central two minutes from each track. The HOMBURG set consists of 1886 excerpts of 10 seconds length.

[6] http://ismir2004.ismir.net/genre_contest/index.htm

4.2 Combination Experiment

In this section, we conduct a similar experiment as in Section 3.1 on the ISMIR'04 *training* collection. The aim is to evaluate the impact of OCs on the performance in *general* music similarity computation (i.e., not limited to rhythm similarity). The results from these experiments are used to create the "unified" algorithm, which will then be evaluated on all three collections (including the HOMBURG collection).

Following previous work [1, 14], we take genre classification accuracy as an indicator of the algorithm's ability to find similar sounding music. We use the same evaluation methodology as before. The timbre component alone yields 83.8%. Combining it with FPs as described, accuracy drops to 83.6%. Using OPs instead, accuracy increases to 85.2%. With OCs, accuracy can be improved up to 87.8% in the parameter range shown in Figure 4. This figure shows an outlier for 19×0 OCs, for which unfortunately we did not find an obvious explanation such as outliers in the distance matrix or numerical instabilities. Comparing Figures 3 and 4, it seems that a good tradeoff between the two collections is found when using 16×1 OCs. This selection yields $17 \cdot 2 = 34$-dimensional feature data, i.e., the rhythm feature data consists of a mean vector of length 34 and a covariance matrix of size $34^2 = 1156$.

4.3 Final Evaluation and Optimization

In Table 2, 10fold CV results obtained with this setting are listed. For comparison to previous work, also the highest classification accuracies obtained so far that we are aware of are listed. These accuracies refer to methods only using audio descriptors without additional human-annotated clues. On all three collections, the results of the "unified" algorithm are above these previously reported results.

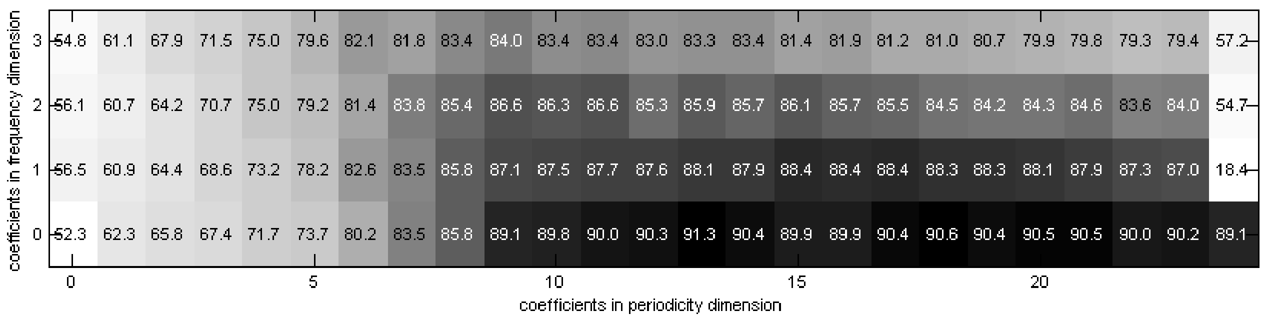


Figure 3. Combination of OCs with timbral component on the ballroom dancers collection, 1NN 10fold cross validation.

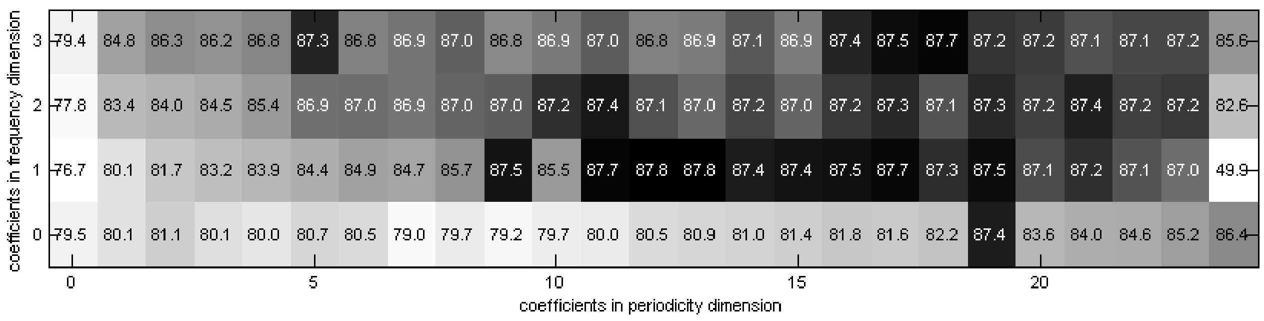


Figure 4. Combination of OCs with timbral component, ISMIR'04 training collection.

Collection	1NN	highest kNN (obtained at k)	Literature
Ballroom	88.4%	**89.2%** (k=5)	86.9% [7]
ISMIR'04 train	87.6%	**87.6%** (k=1)	84.0% [16]
ISMIR'04 1458	90.4%	**90.4%** (k=1)	83.5% [6]
HOMBURG	50.8%	**57.0%** (k=10)	55% [12]

Table 2. Accuracies obtained by the "unified" algorithm on the various collections.

While these results show that our "unified" algorithm outperforms the respective specialized approaches, we observe that when tuning to the particular collections, our techniques can be used to obtain even higher accuracies. For these experiments, we use leave-one-out evaluation for two reasons. First, doing 10fold cross validation (and repeating it several times for averaging) has a clearly longer runtime, as we evaluate a fixed matrix of pairwise distances. Second, in the 10fold cross validation experiments, we observe a certain variance between repeated experiments.

Our non-exhaustive tuning experiments indicate that even the normalization step used to combine two measures (Section 3) alone in some cases increases accuracy. On the Ballroom Dancers collection, a 3NN accuracy of 91.8% is obtained when including normalised OCs up to 24 × 0. Using only the normalised timbre component, on the ISMIR'04 training set a 1NN accuracy of $\mathbf{88.8}\%$, and on the full ISMIR'04 set an accuracy of $\mathbf{91.8}\%$ is reached. On the HOMBURG set, 11NN classification using only the normalised timbre component yields $\mathbf{58.4}\%$.

Common sense indicates that the "unified" algorithm is a better choice for similarity estimation than such tuned variants, as the tuned variants do not perform well on all collections. In particular, these experiments show that discarding the rhythm component and using the timbre component alone, higher accuracies than those of the "unified" algorithm are obtained both on the ISMIR'04 set and the HOMBURG set. But with this setting, accuracy decreases clearly on the "Ballroom Dancers" collection. This may indicate the existence of an *evaluation glass ceiling* in the sense that an improved general music similarity algorithm might even yield lower accuracies.

5 CONCLUSIONS

We have presented modifications of Fluctuation Patterns (FPs) that can be used to obtain higher classification accuracies on the audio signal of the "Ballroom Dancers collection" than FPs compared by Euclidean distance. By adding frequency information to these proposed rhythm descriptors in the form of a "timbral" component results are further improved.

Based on these results, we suggest a "unified" algorithm. The presented experiments indicate that the similarities computed by this algorithm *both* reflect aspects of rhythm similarity *and* aspects of general music similarity. In both respects, classification accuracies obtained in our test setting are at least comparable to those previously reported for algorithms specifically designed for the respective tasks.

Going beyond this, presented preliminary results show that by using different parameter settings (including selection of used OCs, and relative weighting of timbral and rhythm component) for different collections, the accuracies obtained with the "unified" algorithm can be further improved. As by doing so, one loses the generality of the algorithm, we refrain from further optimizations in this direction.

6 ACKNOWLEDGMENTS

This work is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung under project number L511-N15.

7 REFERENCES

[1] Jean-Julien Aucouturier and Francois Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[2] Simon Dixon, Fabien Gouyon, and Gerhard Widmer. Towards characterisation of music via rhythmic patterns. In *Proc. International Conference on Music Information Retrieval (ISMIR'04)*, 2004.

[3] A. Flexer, F. Gouyon, S. Dixon, and G. Widmer. Probabilistic combination of features for music classification. In *Proc. International Conference on Music Information Retrieval (ISMIR'06)*, 2006.

[4] Fabien Gouyon and Simon Dixon. Dance Music Classification: A Tempo-Based Approach. In *Proc. International Conference on Music Information Retrieval (ISMIR'04)*, 2004.

[5] Fabien Gouyon, Simon Dixon, Elias Pampalk, and Gerhard Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Proc. AES* 25th *International Conference*, 2004.

[6] A. Holzapfel and Y. Stylianou. Musical genre classification using nonnegative matrix factorization-based features. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):424–434, 2008.

[7] Andre Holzapfel and Yannis Stylianou. A scale transform based method for rhythmic similarity of music. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009.

[8] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A benchmark dataset for audio classification and clustering. In *Proc. International Conference on Music Information Retrieval (ISMIR'05)*, 2005.

[9] Dan-Ning Jiang Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2002.

[10] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proc. International Conference on Music Information Retrieval (ISMIR'05)*, 2005.

[11] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37:145–151, 1991.

[12] Fabian Moerchen, Ingo Mierswa, and Alfred Ultsch. Understandable models of music collections based on exhaustive feature generation with temporal statistics. In *Proc.*12th *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[13] Nobutaka Ono, Kenichi Miyamoto, Hirokazu Kameoka, and Shigeki Sagayama. A real-time equalizer of harmonic and percussive components in music signals. In *Proc. International Conference on Music Information Retrieval (ISMIR'08)*, 2008.

[14] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval.* Doctoral dissertation, Vienna University of Technology, 2006.

[15] Geoffroy Peeters. Rhythm classification using spectral rhythm patterns. In *Proc. International Conference on Music Information Retrieval (ISMIR'05)*, 2005.

[16] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Automatically adapting the structure of audio similarity spaces. In *Proc.* 1st *Workshop on Learning the Semantics of Audio Signals (LSAS 2006),* 1st *International Conference on Semantics and Digital Media Technology (SAMT 2006)*, 2006.

[17] Yuan-Yuan Shi, Xuan Zhu, Hyoung-Gook Kim, Ki-Wan Eom, and Kim Ji-Yeun. Log-scale modulation frequency coefficient: A tempo feature for music emotion classification. In *Proc.* 1st *Workshop on Learning the Semantics of Audio Signals (LSAS 2006),* 1st *International Conference on Semantics and Digital Media Technology (SAMT 2006)*, 2006.

[18] Kris West. *Novel techniques for Audio Music Classification and Search.* Doctoral dissertation, School of Computing Sciences, University of East Anglia, 2008.

[19] Linxing Xiao, Aibo Tian, Wen Li, and Jie Zhou. Using a Statistic Model to Capture the Association Between Timbre and Perceived Tempo. In *Proc. International Conference on Music Information Retrieval (ISMIR'08)*, 2008.

[20] Wei Xu, Jacques Duchateau, Kris Demuynck, and Ioannis Dologlou. A New Approach to Merging Gaussian Densities in Large Vocabulary Continuous Speech Recognition. In *Proc. IEEE Benelux Signal Processing Symposium*, 1998.

GROUPING RECORDED MUSIC BY STRUCTURAL SIMILARITY

Juan Pablo Bello
Music and Audio Research Lab (MARL), New York University
jpbello@nyu.edu

ABSTRACT

This paper introduces a method for the organization of recorded music according to structural similarity. It uses the Normalized Compression Distance (NCD) to measure the pairwise similarity between songs, represented using beat-synchronous self-similarity matrices. The approach is evaluated on its ability to cluster a collection into groups of performances of the same musical work. Tests are aimed at finding the combination of system parameters that improve clustering, and at highlighting the benefits and shortcomings of the proposed method. Results show that structural similarities can be well characterized by this approach, given consistency in beat tracking and overall song structure.

1. INTRODUCTION

Characterizing the temporal structure of music has been one of the main goals of the MIR community, with example applications including thumbnailing, long-term segmentation and synchronization between multiple recordings [1, 2]. Despite this focus, however, there has been little in terms of using structure as the main driver of audio-based retrieval and organization engines.

This paper proposes and evaluates a methodology for the characterization of structural similarity between musical recordings. The approach models similarity in terms of the information distance between music signals represented using self-similarity matrices. These matrices are well-known for their ability to characterize recurring patterns in structured data, and are thus widely used in MIR for the analysis of musical form. However, in retrieval applications they are mostly used as intermediate representations from which a final representation (e.g. beat spectrum, segment labels) is derived. In this paper we argue that self-similarity matrices can be used directly in the computational modeling of texture-, tempo- and key-invariant relationships between songs in a collection. Our approach is mainly inspired by the work in [3], which uses the same principle to compare the structure of protein sequences.

1.1 Background

The use of structure for audio-based MIR was first proposed in [4]. This approach is based on the idea that long-term structure can be characterized by patterns of dynamic variation in the signal. In this approach, song similarity is measured as the cost of DP-based pairwise alignment between sequences of local energy or magnitude spectral coefficients. Experimental results, albeit preliminary, show the potential of this idea for retrieval.

A similar concept is explored in [5], and more extensively in [6], where variations of spectral content are quantized into a symbolic sequence, obtained via vector quantization or HMMs. In these works, pairwise song similarity is measured using the edit distance or, more efficiently, locality sensitive hashing [6].

The mentioned sequences are not only able to represent the texture and harmony of musical pieces, but also structural patterns, from motifs and phrases to global form. Musical sequences sharing style, origin or functionality will be likely to show structural similarity, despite differences in actual sequence content. Hence, a change of key does not preclude listeners from identifying a 12-bar blues, and the relationship between different variations and renditions of a work remain close, despite changes of instrumentation, ornamentation, tempo, dynamics and recording conditions. Unfortunately, all representations discussed above are sensitive to one or more of these variables. As a result, their success at characterizing music similarity depends on their ability to marginalize those changes. Examples include the use of modified distance metrics and suboptimal feature transposition methods [2, 5].

Structure comparison has been extensively studied in other fields, such as bioinformatics. For protein sequences, for example, structures are usually characterized using *contact maps*, which are, simply put, binary self-similarity matrices where a 1 characterizes a contact (i.e. similarity higher than a certain threshold) and a 0 the lack of it. The problem of comparing protein topologies using contact maps is known as *maximum contact map overlap*, with many proposed solutions in the literature. In this paper we concentrate on the one proposed in [3], which uses an approximation of the information distance between two contact maps known as the *normalized compression distance* (NCD), to be discussed in more detail in section 2.2.

In music, the NCD has been used on raw MIDI data for clustering and classification based on genre, style and melody [7, 8]. More recently, it has been used on audio data for sound and music classification [9] and, with lim-

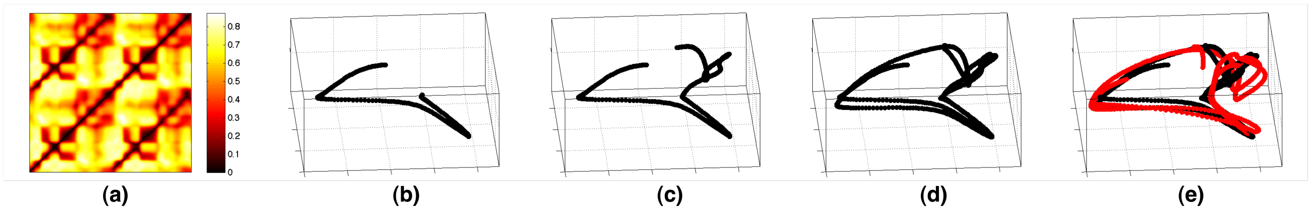


Figure 1. (a) Self similarity matrix of the first 248 bars of a performance of Beethoven 5^{th} Symphony; MDS projection of a quarter (b), half (c) and full matrix (d) to 3 dimensions; (e) comparison of two different performances.

ited success, in cover-song identification [10]. To the best of our knowledge this paper proposes the first use of NCD to characterize structural similarity between music audio recordings.

1.2 Example

Figure 1(a) shows a self-similarity matrix of the first 248 bars of the first movement of Beethoven's 5^{th} symphony. The recording is of a 2006 performance by the Russian National Orchestra conducted by Mikhail Pletnev. Figures 1(b-d) are the result of taking the distances in the matrix and projecting them into a 3-dimensional space using classical multidimensional scaling (MDS). The figures show the trajectory of the piece at a quarter, half and full segment length, respectively. Figures 1(b) and (c) depict the famous opening section of this symphony as a loop, while figure 1(d) shows the recapitulation as simply another, approximate instance of the same loop. This example clearly shows how self-similarity matrices are able to characterize primary (the trajectory itself) and, at least, secondary (local motifs such as the loop) structure in music. Figure 1(e) shows the full segment trajectory described above (in black), and a new trajectory, corresponding to a 1963 recording by the Berlin Philharmonic conducted by Herbert von Karajan (in red). The goal of our approach is to quantify the (dis)similarity of these representations, and to use the results to group related music together.

2. APPROACH

The proposed approach consists of three main parts: (a) representation, where a self-similarity matrix is generated from the analysis of the audio signal; (b) similarity, where the pairwise distance between the representations is computed using the NCD; and (c) clustering, where the matrix of NCDs is used for the grouping of songs. The details are explained in the following.

2.1 Representation

In our implementation we use a beat-synchronous feature set F, composed of either MFCC or chroma features. The first 20 MFCCs are calculated using a 36-band filterbank, frame size of 23.22ms and 50% overlap. The chroma features are computed via the constant-Q transform using a minimum frequency of 73.42 Hz, 36 bins per octave and a 3-octave span, on a signal downsampled to $f_s = 5512.5$ Hz. The resulting features are tuned and their dimensionality reduced to 12 with a weighted sum across each 3-bin pitch class neighborhood. For beat tracking we use the algorithm in [11], and average the extracted features between consecutive beats. Beat tracking is used to reduce the size of the self-similarity matrix and to minimize the effect of tempo-variations on the representation.

The feature set is smoothed using zero-phase forward-backward filtering with a second order Butterworth filter. Filter cutoff is at 1/128th of the feature rate. Finally, the features are standardized (separately for each song).

The computation of self-similarity matrices has been discussed extensively elsewhere in the literature and will not be discussed in any detail here. Suffices to say that for our tests we use both the euclidean and cosine distances. Once computed, matrices are normalized (per song) to the [-1,1] range, their upper triangular part extracted, and the values uniformly quantized and encoded into B bits. In our experiments B assumes the values 2, 3 and 4. It is worth noting that we have favored the notion of "fuzzy" rather than binary self-similarity, as it is not clear what an adequate definition of *contact* may be in the context of this work. For the same reason we have favored the use of uniform quantization over other possible partitions of the similarity range.

2.2 Similarity

We measure similarity using the normalized compression distance (NCD), which will be briefly introduced here (For a comprehensive discussion the reader is referred to [7]).

It can be shown that the information distance between two objects o_1 and o_2, up to a logarithmic additive term, is equivalent to:

$$ID(o_1, o_2) = max\{K(o_1|o_2), K(o_2|o_1)\} \quad (1)$$

where $K(.)$ denotes the Kolmogorov complexity. The conditional complexity $K(o_1|o_2)$ measures the resources needed by a universal machine to specify o_1 given o_2.

The information distance in Eq. 1 suffers from not considering the size of the input objects, and from the non-computability of $K(.)$. To solve the first problem, a normalized information distance can be defined as:

$$NID(o_1, o_2) = \frac{max\{K(o_1|o_2), K(o_2|o_1)\}}{max\{K(o_1), K(o_2)\}} \quad (2)$$

To solve the second problem, we can approximate $K(.)$ using $C(.)$, the size in bytes of an object when compressed using a standard compression algorithm. Using this principle, it can be shown that equation 2 can be approximated by the normalized compression distance:

$$NCD(o_1, o_2) = \frac{C(o_1 o_2) - min\{C(o_1), C(o_2)\}}{max\{C(o_1), C(o_2)\}} \quad (3)$$

where $C(o_1 o_2)$ is obtained by compressing the concatenation of objects o_1 and o_2 [7]. For our implementation the objects are the encoded self-similarity matrices for each song. We use the NCD implementation in the *CompLearn* toolkit [1] with the bzip2 and PPMd compression algorithms.

2.3 Clustering

We use an algorithm from Matlab's statistics toolbox that builds a hierarchical cluster tree using the complete linkage method [12]. The clusters are defined by finding the smallest height in the tree at which a cut across all branches will leave $MaxClust$ or less clusters. The output of the process is a vector containing the cluster number per item in the test set.

3. EXPERIMENTAL SET-UP

3.1 Test Data

We use two datasets in our experiments. The first set, which we call P56, consists of 56 recordings of piano music, including excerpts of 8 works by 3 composers (Beethoven, Chopin and Mozart), played by 25 famous pianists between 1946 and 1998. It was collected as part of the computational study of expressive music performance discussed in [13]. Each work has, at least, 3 associated renditions and at most 13, with audio file lengths in the range of 1 to 8 minutes.

The second set (S67, collected by the authors) includes 67 recordings of symphonic music, including one movement for each of 11 works by 7 composers (Beethoven, Berlioz, Brahms, Mahler, Mendelssohn, Mozart and Tchaikovsky). The set includes instances from 56 different recording sessions scattered between 1948 and 2008, featuring 34 conductors. Each work has 6 associated renditions, with the sole exception of the 3rd movement of Brahm's Symphony No. 1 in C minor, for which 7 performances are available. The duration of the recorded movements range from 3 to 10 minutes.

Classical music is used as, apart from the odd repetition of a motif or section, the structure of renditions can be expected to be the same. The two sets are composed of recordings using similar instrumentation (piano, orchestra), to emphasize the difference with timbre-base similarity approaches. Both sets, however, present significant variations in recording condition and interpretation (notably in dynamics and tempo). All files are 128 kb/s MP3s with sampling frequency of 44.1kHz.

[1] http://www.complearn.org

3.2 Methodology

Clustering methods are highly sensitive to both the number and relative size of partitions in a dataset. To account for variations of those factors and avoid overfitting, every test is performed I times, each using a random sample of size $N < M$, where M is the number of items in the dataset. For every test, we report the mean accuracy of clusters across the I subsets, measured as follows.

Given a partition of the dataset into R groups, $Q = \{q_1, ..., q_R\}$, produced by the clustering algorithm, and a target partition, $T = \{t_1, ..., t_P\}$, we can validate Q using the Hubert-Arabie Adjusted Rand (AR) index as:

$$AR = \frac{\binom{N}{2}(a+d) - [(a+b)(a+c) + (c+d)(b+d)]}{\binom{N}{2}^2 - [(a+b)(a+c) + (c+d)(b+d)]} \quad (4)$$

where $\binom{N}{2}$ is the total number of object pairs in our dataset. AR measures the correspondence between Q and T, as a function of the number of the following types of pairs: (a) pairs with objects in the same group both in Q and T; (b) objects in the same group in Q but not in T; (c) objects in the same group in T but not in Q; and (d) objects in different groups in both Q and T. The AR index accounts for chance assignments and does not require arbitrary assignment of cluster labels not $P = R$, as might be the case when using classification accuracy to validate clustering. Readers unfamiliar with the AR index might find the following guidelines useful: AR = 1 means perfect clustering, while values above 0.9, 0.8 and 0.65 reflect, respectively, excellent, good and moderate cluster recovery. Random partitions of the dataset result on AR $\rightarrow$ 0 (can also assume small negative values). For a detailed discussion of the properties and benefits of the AR index see [14].

4. RESULTS AND DISCUSSIONS

The main goal of our experiments is to test the capacity of the proposed approach in characterizing structural similarity. As similarity is an elusive concept which is not easily quantified, we test an approximate scenario: the task of clustering a music collection into groups of renditions of the same work. Thus, for example, a partition Q of S67, generated using the approach in section 2 with parameters θ, is validated using AR and a target partition T of 11 groups, where each group contains the 6 or 7 renditions of one of the works in the collection.

Specifically, our experiments seek to: (1) find the parameterization θ that maximizes AR, (2) assess the impact of the used clustering methodology, and (3) highlight the strengths and shortcomings of our approach.

4.1 Parameterization

In our experiments $\theta = \{F, d, B, C, MaxClust\}$, where F is the feature set (MFCC or chroma), d the distance metric used to compute the self-similarity matrix (euclidean or cosine), B the number of bits used to quantize the matrix (2, 3 or 4), C the compression method used for the computation of the NCD (bzip2 or PPMd), and $MaxClust$ the

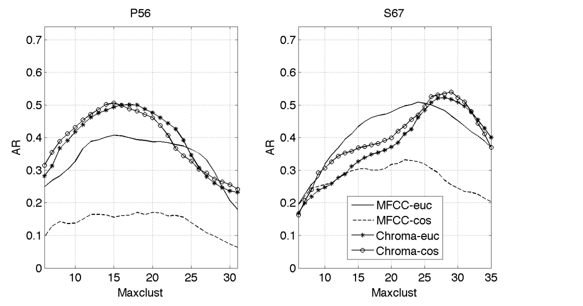


Figure 2. Comparison of mean AR results for all F, d combinations on sets P56 (left) and S67 (right).

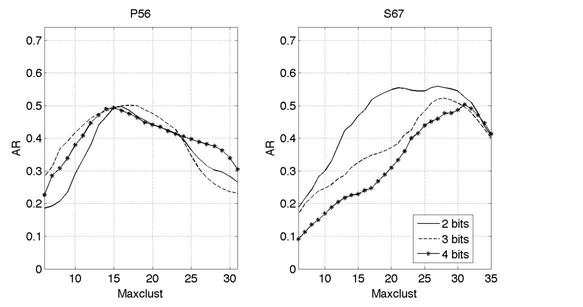


Figure 3. Comparison of mean AR results for $B = \{2, 3, 4\}$ on sets P56 (left) and S67 (right).

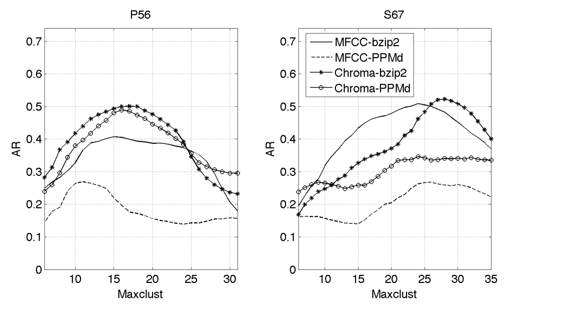


Figure 4. Comparison of mean AR results for $C = \{bzip2, PPMd\}$ on sets P56 (left) and S67 (right).

maximum number of clusters to be retrieved from the tree (between 6 and 35).

All possible combinations of θ are tested $I = 50$ times [2], using random samples of size $N = 0.75 \times M$ (42 for P56, 50 for S67). In all tests, both collections are tested independently.

Figure 2 shows results for all F, d combinations for $C = bzip2$ and $B = 3$. As with most figures in this section, it separately shows AR values for P56 (left) and S67 (right), across the range of $MaxClust$ values. For both datasets, chroma features outperform MFCCs, clearly for P56 and slightly for S67. This is consistent with the notion of harmonic content as a reliable indicator of structure in music, as has been repeatedly found in the segmentation literature [1,2]. The better performance of MFCCs in S67 compared to P56 is to be expected, as within-song timbre differences and dynamic changes are more pronounced in orchestral than in piano music. For chroma features, the use of euclidean or cosine distances in the computation of the self-similarity matrix makes little difference. For MFCCs, however, the euclidean distance results in significantly better performance, indicating that dynamics are as important as timbre changes in defining the structure of a piece.

[2] We tested $I = \{10, 20, 50, 100, 200, 500, 1000\}$ and found variations of mean AR to be minimal for $I \geq 50$.

Figure 3 illustrates the importance of the number of bits B used in the encoding and quantization of the self-similarity matrix, for $F = chroma$, $d = euclidean$ and $C = bzip2$. Apart from $B = 2$ giving the best results for S67, no clear trend is visible in these plots (at least not common to both sets). This hints at process independence from the choice of B. The good performance of $B = 2$, however, opens the door for a binary definition of contacts in music, although more extensive testing is necessary to define an appropriate threshold.

Finally, figure 4 compares two compression methods for the computation of NCD. In these plots, $F = chroma$, $d = euclidean$ and $B = 3$. In all cases $bzip2$ outperforms $PPMd$, which is unfortunate as the latter is much faster than the former. This result seems to contradict findings in the literature where the PPM family of compression methods usually works best for the NCD computation [7].

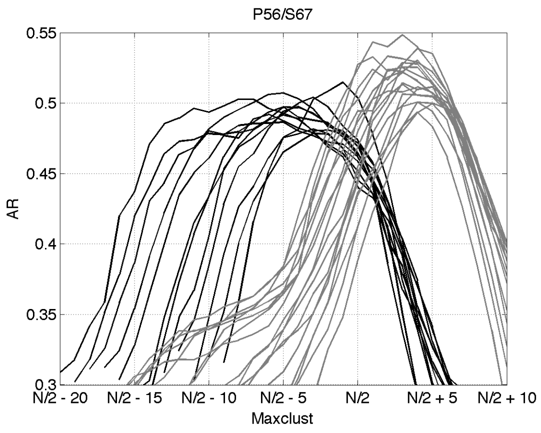


Figure 5. Variation of mean AR according to random sample size N (P56 in black, S67 in gray).

4.2 Clustering methods

On a separate experiment, we tested our system against variations of the random sample size N for both collections. N values ranged from 30 to 52 for P56, and 64 for S67. We used $F = chroma$, $d = euclidean$, $B = 3$ and $C = bzip2$. Figure 5 shows results for P56 (in black) and S64 (in gray, skewed towards the right), across a range of $MaxClust$ values ranging from $N/2 - 20$ to $N/2 + 10$. Each curve corresponds to a value of N. Variations of peak AR across N appear to be uniformly distributed in the depicted range for each test set. Their location within this range does not follow any obvious trend. For example, for P56, the minimum peak corresponds to the $N = 32$ curve, while the maximum peak is for $N = 30$ (closely followed by $N = 48$). All other peaks are randomly located in between.

Notably, the location of peaks appears to be a function of N, with most peaks in $(N/2 - 5) \pm 3$ for P56 and in $(N/2 + 3) \pm 2$ for S67. The difference between the sets, however, also indicates that the size of the collection M, the number of groups within that collection and the size of those groups have a hand in the results. While N and M are always known, it is unreasonable to expect the number and size of groups to be known, making the choice of value for the critical $MaxClust$ parameter a complex one. Our inability to define $MaxClust$ with prior information is a major shortcoming of the proposed approach.

As an alternative we have tested a different clustering algorithm, which operates by merging clusters whose separation, measured in their connecting node, is less than a pre-specified *Cutoff* value, ranging between 0 and 1. Notably, this method does not require any prior information about cluster numbers. Additionally, we test building the hierarchical cluster tree using single, average and weighted linkage in addition to the complete linkage method used in the rest of this paper [12]. Figure 6 shows the results of these tests using $F = chroma$, $d = euclidean$, $B = 2$ and $C = bzip2$. The AR = 0.63 result for weighted linkage and Cutoff = 0.85 in S67 is the highest obtained in our experiments, a significant increase on our previous best (visible in the "complete" curve of the same graph). It clearly shows that gains can be made by improving our clustering stage. However, this result is not indicative of a general trend, as illustrated by the low results obtained for the same method in the P56 dataset. An in-depth exploration of the space of clustering methods and their parameterizations will be the focus of future work.

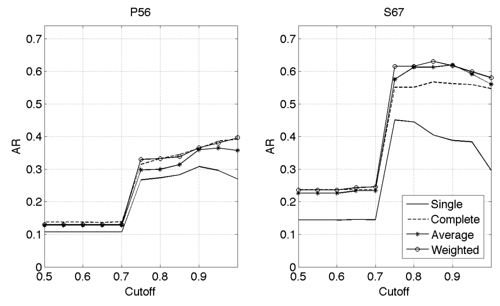


Figure 6. Test of cutoff clustering with 4 linkage methods.

4.3 An example tree

Figure 7 is generated using yet another linkage algorithm on the full S67 dataset, the quartet method described in [7], using $F = chroma$, $d = euclidean$, $B = 3$ and $C = bzip2$. Clustering on this tree using $MaxClust = 36$ results on $AR = 0.55$, which makes this graph representative of system performance using the best parameterization.

The tree branches out into 10 clusters, each corresponding to a work in the collection. Four of those clusters group all renditions of a given work. Figure 7(a) shows a detail of the tree exemplifying one such cluster, corresponding to the 7 renditions of the third movement of Brahm's Symphony No. 1 in C minor. Two clusters group 5 out of 6 performances, for example those for the third movement of Mozart's Symphony No. 4 in G minor k550 depicted in Figure 7(c). One cluster, for the second movement of Mahler's Symphony No. 1 in D major "Titan", groups 4 out of 6 performances as shown in Figure 7(b). The three remaining clusters group only 3 or 2 performances out of 6. Only one work results in no clusters of any kind. In total, 47 out of 67 recordings are correctly assigned to a group. Ungrouped recordings are located in the stem of the tree, which has been gray-shaded in the graph.

Figures 7(b) and (c) also help illustrate the effect of beat tracking accuracy on the proposed approach. The number of detected beats in the missing performance of Mozart's k550, visible in the stem of the tree in Fig. 7(b), is approximately twice as many as those detected in all other performances of the same piece. Octave errors act as filters on the feature set, which can result on a significant loss of detail in the corresponding self-similarity matrix and, as the tree shows, a poor characterization of structural similarity between the recordings. This is an important drawback of our approach as octave errors are common in beat tracking. Another example of the same problem are the two missing recordings in Mahler's Symphony 1 cluster in Fig. 7(b), which are located in the lower end of the stem of the tree. An informal analysis of the results shows that a good portion of overall clustering errors are associated to inconsistencies in beat tracking. It is worth noting that "inconsistency" is the right word in this case, as what is really important is not that beats are correctly tracked, but that their relation to the actual tempo of the piece is the same for all performances.

An additional observation relates to the six performances of the fourth movement of Berlioz's "Symphonie Fantastique". The score includes a repetition of the first 77 bars of this movement before entering its second half, roughly describing an AAB structure. Half of the performances in our dataset, however, ignore that repetition resulting on a shorter AB structure. Correspondingly, the cluster in the tree related to this piece groups only the latter, while the other three performances appear close together in the lower end of the tree. While in theory the common part of the structure should be enough to identify the similarity between all six recordings, in practice this is clearly not the case. This sensitivity to common structural changes, e.g. repetitions, raises questions about the potential use of NCD-based similarity in the modeling of the relationships that exist amongst variations, covers, remixes and other derivatives of a given work. Further research is now being conducted to fully explore this issue.

5. CONCLUSIONS AND FUTURE WORK

This paper presents a novel approach for the organization of recorded music according to structural similarity. It uses the Normalized Compression Distance (NCD) on self-similarity matrices extracted from audio signals, using stan-

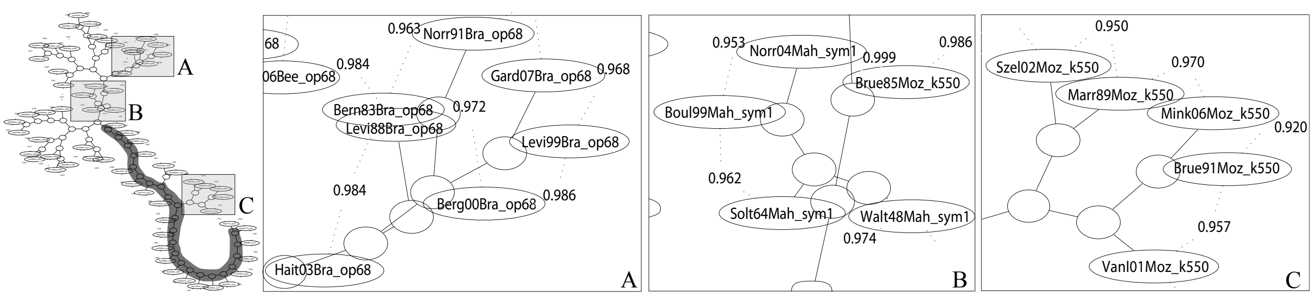


Figure 7. Uprooted binary tree of S67 using the quartet method. Details show a perfect cluster (A) and two partial clusters (B and C)

dard features and distance metrics. The approach is evaluated on its ability to facilitate the clustering of different performances of the same piece together. Experimental results on piano and orchestral music datasets show that the approach is able to successfully group the majority of performances in a collection, resulting on average AR values in the 0.5-0.6 range. Our tests show that best results are obtained for self-similarity matrices computed using chroma features and the euclidean distance, and encoded using 2-3 bits. They also show that the NCD works best when using the $bzip2$ compression algorithm. Preliminary results also indicate that further gains can be made by improving the clustering stage.

On the downside, the approach has shown sensitivity to octave errors in beat tracking and, predictably, to structural changes, which limit the potential application of the current implementation to the retrieval and organization of other types of musical variations. To address these issues, future work will concentrate on two main areas. First, the improvement of the self-similarity representation, along the lines of work in [2], to include transposition invariance, path following and the merging of matrices computed at 1/2, 1 and 2 times the tracked tempo. Second, we will explore alternatives to the use of NCD for the maximum contact map overlap problem. We plan to explore solutions based on the branch and cut approach (e.g. [15]) and adapt them to the specificities of music data.

6. ACKNOWLEDGEMENTS

The author would like to thank Gerhard Widmer and Werner Goebl for the P56 dataset, and Dan Ellis and the CompLearn team for free distribution of their code libraries. This material is based upon work supported by the NSF (grant IIS-0844654) and by the IMLS (grant LG-06-08-0073-08).

7. REFERENCES

[1] M. A. Bartsch and G. H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *WASPAA-01, NY, USA*, pages 15–18, 2001.

[2] M. Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[3] N. Krasnogor and D. A. Pelta. Measuring the similarity of protein structures by means of the universal similarity metric. *Bioinformatics*, 20(7):1015–1021, 2004.

[4] J. Foote. Arthur: Retrieving orchestral music by long-term structure. In *ISMIR*, 2000.

[5] J.-J. Aucouturier and M. Sandler. Using long-term structure to retrieve music: Representation and matching. In *ISMIR 2001, Bloomington, Indiana, USA*, 2001.

[6] M. Casey and M. Slaney. Song intersection by approximate nearest neighbour retrieval. In *ISMIR-06, Victoria, Canada*, 2006.

[7] R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[8] Ming Li and Ronan Sleep. Genre classification via an LZ78 string kernel. In *ISMIR-05, London, UK,*, 2005.

[9] M. Helén and T. Virtanen. A similarity measure for audio query by example based on perceptual coding and compression. In *DAFx-07, Bordeaux, France*, 2007.

[10] T. Ahonen and K. Lemström. Identifying cover songs using normalized compression distance. In *MML'08, Helsinki, Finland*, 2008.

[11] D. Ellis. Beat Tracking by Dynamic Programming. *Journal of New Music Research*, 36(1):51–60, March 2007.

[12] R. Xu and D. Wunsch. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[13] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic. In search of the Horowitz factor. *AI Mag.*, 24(3):111–130, 2003.

[14] D. Steinley. Properties of the Hubert-Arabie adjusted Rand index. *Psychological methods*, 9(3):386–396, September 2004.

[15] W. Xie. and N. V. Sahinidis. A branch-and-reduce algorithm for the contact map overlap problem. *Research in Computational Biology (RECOMB 2006), Lecture Notes in Bioinformatics*, 3909:516–529, 2006.

A FILTER-AND-REFINE INDEXING METHOD FOR FAST SIMILARITY SEARCH IN MILLIONS OF MUSIC TRACKS

Dominik Schnitzer
Austrian Research Institute
for Artificial Intelligence
dominik.schnitzer@ofai.at

Arthur Flexer
Austrian Research Institute
for Artificial Intelligence
arthur.flexer@ofai.at

Gerhard Widmer
Johannes Kepler University
Linz, Austria
gerhard.widmer@jku.at

ABSTRACT

We present a filter-and-refine method to speed up acoustic audio similarity queries which use the Kullback-Leibler divergence as similarity measure. The proposed method rescales the divergence and uses a modified FastMap [1] implementation to accelerate nearest-neighbor queries. The search for similar music pieces is accelerated by a factor of $10-30$ compared to a linear scan but still offers high recall values (relative to a linear scan) of $95-99\%$.

We show how the proposed method can be used to query several million songs for their acoustic neighbors very fast while producing almost the same results that a linear scan over the whole database would return. We present a working prototype implementation which is able to process similarity queries on a 2.5 million songs collection in about half a second on a standard CPU.

1. INTRODUCTION

Today an unprecedented amount of music is available online. As of April 2009, the Apple iTunes music store alone lists more than 10 million downloadable songs in its catalog. Other online music stores like Amazon MP3 still offer a 5 million songs catalog to choose from. With the catalog numbers constantly reaching new record highs, the need for intelligent music search algorithms that provide new ways to discover and navigate music is apparent.

Unfortunately many of the intelligent music processing algorithms that have been published do not easily scale to the millions of music pieces available in an online music store. In particular, this is true for music recommendation algorithms which compute acoustic music similarity using a Gaussian timbre representation and the Kullback-Leibler divergence, as in [2], [3] or [4].

Especially the Kullback-Leibler divergence, as it is used in the referenced works, poses multiple challenges when developing a large scale music recommendation system: (1) the divergence is very expensive to compute, (2) it is not a metric and thus makes building indexing structures around it very hard and (3) in addition, the extracted acoustic music similarity features have a very high degree of freedom, which too is a general problem for indexing solutions ("curse of dimensionality") [5].

But on the other hand, systems using this technique regularly rank in the very top places in the yearly MIREX Automatic Music Recommendation evaluations [1], which makes them a tempting but challenging target for broad usage in real applications.

1.1 Related Work

The idea of using FastMap-related techniques for computationally heavy non-metric similarity measures and nearest neighbor retrieval was first demonstrated by Athitsos [6]. They use BoostMap [7] to improve the speed of classifying handwritten digits. Cano et al. [8] use FastMap to map the high dimensional music timbre similarity space into a 2-dimensional space for visualization purposes.

Roy et al. [9] present a music recommendation system which uses a Monte-Carlo approximation of the Kullback-Leibler (KL) divergence as similarity measure. The Monte-Carlo approximation of the KL divergence is far more expensive to compute and less accurate than the closed form of the KL divergence which is used in our paper and recent music similarity algorithms. To speed up a similarity query, they narrow the number of nearest neighbor candidates by incrementally increasing the accuracy of the Monte-Carlo sampled divergence measure.

Another interesting approach, which was pursued by Garcia [10], is to compute computationally expensive similarity measures on modern graphics processors (GPUs). Modern GPUs offer high floating-point performance and parallelism. As an example Garcia shows how a linear brute force nearest neighbor scan using the Kullback-Leibler divergence can be accelerated on a GPU compared to computing it on a standard CPU. The idea to use the GPU to process similarities could be combined with the methods presented in this paper.

With mufin.com there also exists a commercial music recommendation service which computes acoustic audio similarities for a very large database of music (6 million tracks as of April 2009). However, their website gives no hint on how their service works [2].

[1] http://www.music-ir.org/mirexwiki/
[2] http://www.mufin.com/us/faq.html

1.2 Contributions of this paper

The contribution of this paper is three-fold:

- First, we present a filter-and-refine method based on FastMap which allows quick music similarity query processing. It is designed to work with very large music databases which use Gaussian timbre models and the Kullback-Leibler divergence as music similarity measure.
- Second, we show how a rescaling of the divergence values and a new FastMap pivot object selection heuristic substantially increase the nearest neighbor recall of the algorithm.
- Finally, we present an implementation of a music recommendation system using the proposed techniques which handles a 2.5 million tracks evaluation collection in a very efficient way.

2. PRELIMINARIES

2.1 Data

Throughout this paper we use a collection of 2.5 million songs to evaluate the performance and to show the practical feasibility of our approach. The 2.5 million tracks consist of 30 second snippets of songs gathered by crawling an online music store offering free audio preview files.

2.2 Similarity

We extract timbre features from the snippets and compute a single Gaussian timbre representation using the method proposed by Mandel & Ellis [2]. We compute 25 Mel Frequency Cepstrum Coefficients (MFCCs) for each audio frame, so that a Gaussian timbre model x finally consists of a 25-dimensional mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. For performance reasons we also precompute and store the inverted covariance matrix $\boldsymbol{\Sigma^{-1}}$.

To compute acoustic timbre similarity we use the symmetrized version (SKL) of the Kullback-Leibler divergence (KL, [11]), defined between two multivariate normal distributions $x_1 \sim \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$ and $x_2 \sim \mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$:

$$SKL(x_1, x_2) = \frac{1}{2}KL(x_1, x_2) + \frac{1}{2}KL(x_2, x_1). \quad (1)$$

A query for similar songs is processed in a linear scan by computing the SKL between the Gaussian x_1 of the seed song and all other songs in the database. The songs with the lowest divergence to the seed song are its nearest neighbors and possible recommendations.

2.3 Nearest neighbor recall

To compare the effectiveness of the nearest neighbor retrieval variants evaluated, we used what we call nearest neighbor (*NN*) recall. We define it as the ratio of true nearest neighbors found by some algorithm (NN_{found}) to the number of true nearest neighbors (NN_{true}). The true nearest neighbors are found by a full linear scan.

$$recall = \frac{|NN_{found} \cap NN_{true}|}{|NN_{true}|} \quad (2)$$

3. THE METHOD

To build our filter-and-refine method for fast similarity queries we use an adopted version of FastMap [1], a Multidimensional Scaling (MDS) technique. MDS [12] is a widely used method for visualizing high-dimensional data. It takes the distance matrix of a set of items as input and maps the data to vectors into an arbitrary-dimensional Euclidean space. FastMap is straightforward to use even for large databases since it only needs a low and constant number of rows of the similarity matrix to compute the vector mapping. However, FastMap requires the distances to adhere to metric properties.

3.1 Original FastMap

The original FastMap [1] algorithm uses a simple mapping formula (Equation 3) to compute a k-dimensional projection of objects into the Euclidean vector space. The dimension k is arbitrary and can be chosen as required. Usually higher dimensions yield a more accurate mapping of the original similarity space.

To project objects into a k-dimensional Euclidean vector space, first two pivot objects from the feature database have to be selected for each of the k dimensions. The original algorithm uses a simple heuristic to select those pivot objects: for each dimension ($j = 1..k$), (1) chose a random object x_r from the database, (2) search for the most distant object of x_r using the original distance measure $D()$ and select it as the first pivot object $x_{j,1}$ for the dimension, (3) the second pivot object $x_{j,2}$ is the object most distant to $x_{j,1}$ in the original space.

After the $2k$ pivot objects have been selected, the vector representation of an object x is computed by estimating $F_j(x)$ for each dimension ($j = 1..k$):

$$F_j(x) = \frac{D(x, x_{j,1})^2 + D(x_{j,1}, x_{j,2})^2 - D(x, x_{j,2})^2}{2D(x_{j,1}, x_{j,2})} \quad (3)$$

This method depends on metric properties of D to produce meaningful mappings. However, it has been noted that FastMap works surprisingly well also for non-metric divergence measures [7].

As FastMap only requires a distance function D and pivot objects to compute the vector mapping, it can be instantly applied to map the Gaussian timbre models with the SKL as distance function to Euclidean vectors (ignoring the fact that the SKL is not metric).

3.2 A Filter-And-Refine Method using FastMap

To use FastMap to quickly process music recommendation queries, we initially use it to map the Gaussian timbre models to k-dimensional vectors. In a two step filter-and-refine process we then use those vectors as a prefilter: first we *filter* the whole collection in the vector space (with the squared Euclidean distance) to return a number (*filter-size*) of possible nearest neighbors, then we *refine* the result by computing the exact SKL on the candidate subset to return the nearest neighbors. By using the SKL to *refine* the results, the correct nearest neighbor ranking is ensured. We set the parameter *filter-size* to a fraction of the whole collection.

Since the complexity of a single SKL comparison is much higher than a simple vector comparison, the use of the squared Euclidean distance to prefilter the data results in large speedups compared to a linear scan over the whole collection using the SKL. Table 1 compares the computational cost (in floating point operations, $flops$) of the SKL to the squared Euclidean distance d^2 using different vector dimensions (k) to prefilter candidate nearest neighbors.

Divergence	$flops$	$flops/flops_{SKL}$
SKL	3552	1
$d^2, k = 20$	60	0.017
$d^2, k = 40$	120	0.034
$d^2, k = 60$	180	0.051

Table 1. The computational complexity (in $flops$) of computing the squared Euclidean distance (d^2) is, even for high mapping dimensions like $k = 60$, much lower than the costs of computing a single SKL comparison. *Note:* We already use an optimized implementation of the SKL exploiting matrix symmetry and the sequence of matrix operations [13].

Unfortunately, as we show in the next section (3.3), applying FastMap to the problem without any modifications yields very poor results.

3.3 Modifications

In our implementation we have included two important modifications which improve the quality of FastMap mappings for nearest neighbor retrieval. The modifications are centered around two thoughts: (1) a metric divergence measure would produce better vector mappings, and (2) a more specialized heuristic for pivot object selection could produce better mappings especially for the near neighbors, which are the center of our interest.

3.3.1 Rescaling

Before mapping the objects $x_i \in X$ to a k-dimensional vector (Equation 3), we propose to rescale the original symmetric Kullback-Leibler divergences (SKL) by taking the square-root:

$$D(x_1, x_2) = \sqrt{SKL(x_1, x_2)}. \tag{4}$$

This rescaling has the effect of making the SKL behave more like a metric. As the SKL already has the important properties of being symmetric and non-negative, it only fails to fulfill the triangle inequality. Taking the square root has the effect to partly fix the divergence, making it more

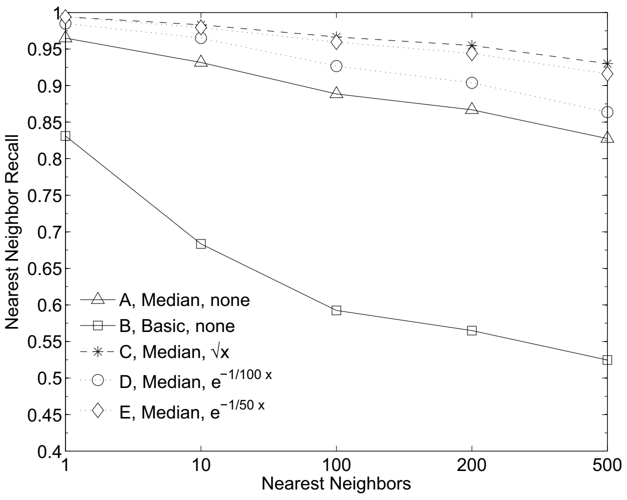


Figure 1. Nearest neighbor (NN) recall of two pivot object selection methods (*median*: the proposed pivot object selection heuristic, *basic*: the original FastMap heuristic) in combination with three divergence rescaling methods (*no-rescaling*, $e^{\lambda x}$, $\sqrt{x}$). NN recall is averaged over five independent evaluation runs (10.000 queries per run), each time using a new random collection. Parameters: $k = 40$, *filter-size=10%*, *collection size=100.000.*

metric [14]. Another more common way is to rescale the SKL with $e^{\lambda SKL()}$ (see [3] or [2]).

We have experimentally verified the effect of rescaling on a collection of 100.000 randomly drawn Gaussian timbre models (Table 2), by checking the triangle inequality. The table clearly shows that exponentiating indeed reduces the number of cases where the triangle inequality is violated, but it does not work as well as taking the square root, which makes the SKL obey the triangle inequality in more than 99% of the cases in our experimental setup.

Divergence	% triangle inequality
$SKL()$	91.57%
$1 - e^{\lambda SKL()}, \lambda = -\frac{1}{100}$	93.71%
$1 - e^{\lambda SKL()}, \lambda = -\frac{1}{50}$	95.60%
$\sqrt{SKL()}$	99.32%

Table 2. Percentage of Gaussian object triples fulfilling the triangle inequality ($D(x, z) \leq D(x, y) + D(y, z)$) with and without rescaling. The triangle inequality was checked for all possible triples in a collection of 100.000 randomly selected Gaussian timbre models.

3.3.2 Pivot Object Selection

To select the pivot objects which are needed to map an object x to a vector space, the original algorithm uses two objects for each dimension which lie as far away from each other as possible (see Section 3.1). In contrast to the original heuristic we propose to select the pivot objects using an adapted strategy: (1) first we randomly select an object x_r and compute the distance to all other objects; (2) we then select the first pivot object x_1 to be the object lying at the distance median, i.e. the object at the index $i = \lfloor N/2 \rfloor$ on the sorted list of divergences; (3) likewise, the second pivot object x_2 is selected to be the object with the distance median of all divergences from x_1 to all other objects.

By using pivot objects at the median distance we avoid using objects with extremely high divergence values which often occur in the divergence tails when using the SKL. Since we are also particularly interested in optimally mapping the near neighbors and not the whole divergence space, this strategy should also help in preserving the neighborhoods.

3.3.3 Improvements

Finally, we measure how these modifications improve the filter-and-refine method by experimentally computing the nearest neighbor (NN) recall of each change on a 100.000 songs collection. Figure 1 shows the result of the experiment. A huge improvement in the nearest neighbor recall can be seen for all strategies which use the median pivot object selection heuristic (*A*, *C*, *D*, *E*) compared to the original FastMap heuristic (*B*). The figure also shows that rescaling the SKL values helps to further increase the NN recall. The suggested strategy (*C*) using the median pivot object selection strategy together with square-root-rescaling gives the best results.

4. IMPLEMENTATION

The implementation of the filter-and-refine music recommendation engine is straightforward: in an initial step the whole collection is preprocessed with the proposed mapping method, transforming the database objects into a k-dimensional vector space. This is a linear process since only $2k$ pivot objects have to be selected and each object in the database is mapped to a vector using Equation 3 once. Our implementation saves the pivot objects for each dimension and the vector mappings of processed objects to disk. This allows fast restarting of the system and easy processing of new objects.

To query for similar objects we use the previously described filter-and-refine method, filtering out a predefined number (*filter-size*, a percentage of the collection size) of nearest neighbor candidates using the vector representation and refining the result with the exact SKL.

This outlines the general method we propose, but obviously two parameters which have a huge impact on the retrieval quality (nearest neighbor (NN) recall) and the query speed have not been discussed yet: the number of vector dimensions k and the *filter-size*.

4.1 Recall and Speed

It is obvious that a larger *filter-size* results in better NN recall values but higher computational costs. Likewise, a higher k used for the vector mapping results in a more accurate mapping of the divergence space, but with each dimension the computational costs to compute the squared Euclidean distance in the prefilter steps are increased.

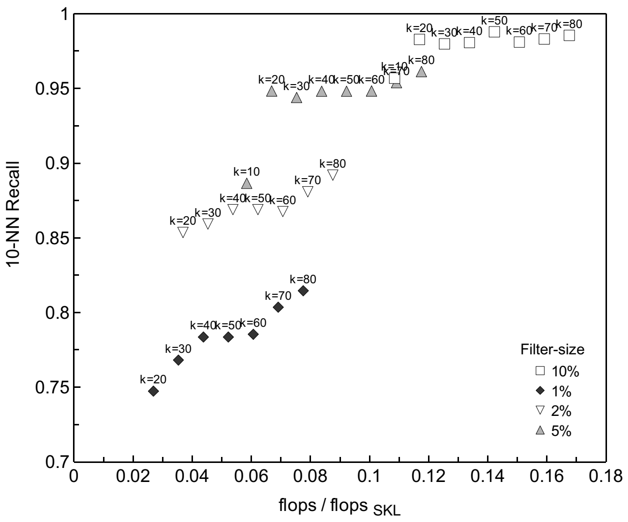


Figure 2. Plot relating the nearest neighbor recall and the floating point operations resulting from different filter-and-refine parameter combinations, to a full linear scan ($flops/flops_{SKL}$). Recall is computed for the 10 nearest neighbors for different parameter combinations of k and *filter-size* in a collection of 100.000 songs. A good combination (good recall, low computational cost) would be mapped to the upper left corner of the plot.

Figure 2 evaluates different parameter combinations of k and *filter-size* and their impact on nearest neighbor recall and computational cost (and thus query speed). The diagram was compiled using a collection of 100.000 Gaussian timbre models. It shows the 10-NN retrieval recall and query speed (computational cost in terms of flops).

The figure shows that a parameter combination of $k = 20$ and *filter-size*$=5\%$ can be selected to achieve about 95% 10-NN recall. That combination would take only 7% of the query time required by a linear scan with the SKL. If a 10-NN recall of 85% is acceptable a parameter combination requiring only 3.5% the computational cost of a linear scan is possible ($k = 20$ and *filter-size*$=2\%$). Almost perfect 10-NN recall values ($> 98 - 99\%$) can be reached when setting *filter-size* to about 10% of the collection size, which still requires only 10% of the time a linear scan would need.

This evaluation shows how a good parameter combination for a collection should be selected. In Section 5 we plot a similar diagram (Figure 3) to select the best parameters for a 2.5 million song collection achieving 99% 1-NN, 98% 10-NN and 95% 100-NN recall on the collection.

4.2 Errors

Another aspect which is of interest is how falsely reported nearest neighbors (false positives) affect the average quality of music recommendations. We have done a 1-NN genre evaluation (with artist filter, see [3]). This is a standard evaluation to test the quality of a music recommendation algorithm.

We tested four different collections (three in-house collections and the *Ismir 2004* Magnatune music collection which is freely available for testing purposes[3]). Table 3 summarizes the results. It appears that the errors which are being made do not affect the classification accuracy in an adverse way. Classification accuracy decreases only by about 0.1% for the two larger collections and by about 0.5% for the two small collections.

$Collection, size$	Genres	F&R	Full Scan
$\#1, N = 16781$	21	30.17%	30.28%
$\#2, N = 9369$	16	28.55%	28.66%
$\#3, N = 2527$	22	28.27%	28.78%
Ismir 2004, $N = 729$	6	64.47%	64.88%

Table 3. 1-NN genre evaluation results (with artist filter) on four different collections. The table compares the genre classification accuracy of the filter-and-refine (F&R) approach presented in this paper with a full exact linear scan. Parameters: $k = 40$, *filter-size*=5%

5. PROTOTYPE PERFORMANCE

To show the practical feasibility of using this filter-and-refine method with large music databases we use the method on the 2.5 million song collection and build a prototype music recommendation system. The system should be able to answer queries for the 100 nearest neighbors with high speed and recall.

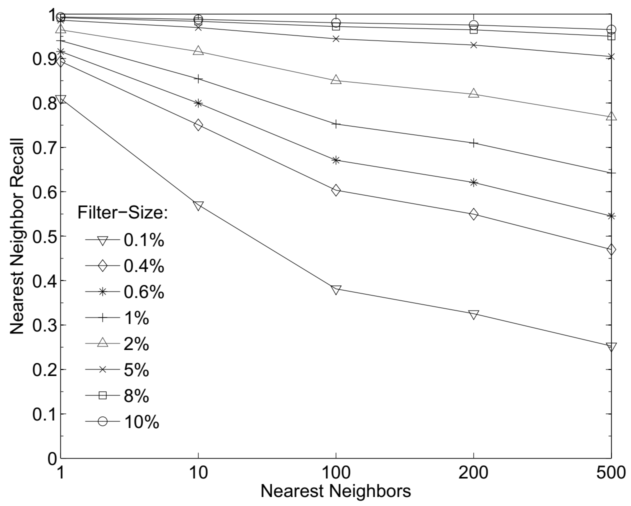


Figure 3. NN recall with different *filter-sizes* evaluated on 1% ($= 25.000$) of the 2.5 million songs collection. With a *filter-size* of 5% one can achieve 95% 100-NN recall and 98% 10-NN and 99% 1-NN recall. $k = 40$.

To select the optimal parameter we ran an experiment to determine the best *filter-size*, k was set to 40. Figure 3 shows the recall values for different NN and *filter-sizes*. It can be seen that the true 1-NN and 10-NN are retrieved almost always if the *filter-size* is set to 5%, 8% or 10% of the collection size.

[3] http://ismir2004.ismir.net/genre_contest/index.htm

In a second experiment (Figure 4) we compare the actual query response times of three different *filter-size* settings (*filter-size*$= 8\%, 5\%, 2\%$, $k = 40$) to a full linear scan. It can be seen that the system running on a single standard CPU core is capable of answering music recommendation queries in half a second while returning about 95% of the correct 100 nearest neighbors compared to a linear scan which would take about $7.8sec$ on the system.

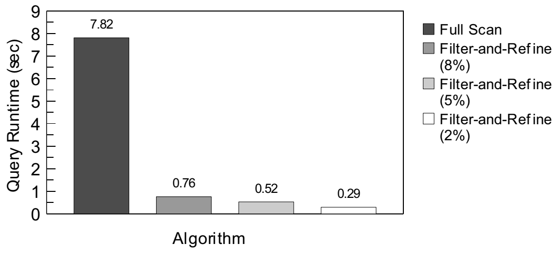


Figure 4. Comparison of the time it takes to query a 2.5 million song collection for nearest neighbors using a full scan compared to a scan using the filter-and-refine method proposed. The PC used a standard Intel Core Duo CPU (2.5GHz) and had all Gaussian models loaded to RAM.

6. CONCLUSIONS

We have described a filter-and-refine method for fast approximate music similarity search in large collections. The method is designed for Gaussian music timbre features using the symmetric Kullback-Leibler divergence to compute acoustic similarity, but could be generalized to other distance measures. A prototype implementation of our method handling 2.5 million tracks is able to answer music similarity queries in about half a second on a standard desktop CPU.

By accelerating similarity queries by a factor 10 to 30, we show how a large scale music recommendation service relying on recent music information retrieval techniques could operate.

ACKNOWLEDGMENTS

This research is supported by the Austrian Research Fund (FWF) under grant L511-N15, and by the Austrian Research Promotion Agency (FFG) under project number 815474-BRIDGE.

7. REFERENCES

[1] C. Faloutsos and K.I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 163–174. ACM New York, NY, USA, 1995.

[2] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR05), London, UK*, 2005.

[3] E. Pampalk. Computational models of music similarity and their application in music information retrieval. *Docteral dissertation, Vienna University of Technology, Austria*, 2006.

[4] T. Pohle and D. Schnitzer. Striving for an improved audio similarity measure. *4th Annual Music Information Retrieval Evaluation Exchange*, 2007.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is“ nearest neighbor” meaningful? *Lecture Notes in Computer Science*, pages 217–235, 1999.

[6] V. Athitsos, J. Alon, and S. Sclaroff. Efficient nearest neighbor classification using a cascade of approximate similarity measures. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 1, 2005.

[7] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. BoostMap: A method for efficient approximate similarity rankings. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2.

[8] P. Cano, M. Kaltenbrunner, F. Gouyon, and E. Batlle. On the use of FastMap for audio retrieval and browsing. In *Proc. Int. Conf. Music Information Retrieval (ISMIR)*, pages 275–276, 2002.

[9] P. Roy, J.J. Aucouturier, F. Pachet, and A. Beurive. Exploiting the tradeoff between precision and cpu-time to speed up nearest neighbor search. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR05), London, UK*, 2005.

[10] V. Garcia, E. Debreuve, and M. Barlaud. Fast k nearest neighbor search using GPU. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPR Workshops 2008*, pages 1–6, 2008.

[11] W.D. Penny. Kullback-Leibler divergences of normal, gamma, Dirichlet and Wishart densities. *Wellcome Department of Cognitive Neurology*, 2001.

[12] T.F. Cox and M.A.A. Cox. *Multidimensional scaling*. CRC Press, 2001.

[13] D. Schnitzer. Mirage – High-Performance Music Similarity Computation and Automatic Playlist Generation. *Master's thesis, Vienna University of Technology*, 2007.

[14] K.L. Clarkson. Nearest-neighbor searching and metric space dimensions. *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59, 2006.

A MEASURE OF MELODIC SIMILARITY BASED ON A GRAPH REPRESENTATION OF THE MUSIC STRUCTURE

Nicola Orio
Department of Information Engineering
University of Padova
nicola.orio@dei.unipd.it

Antonio Rodà
Lab. AVIRES
University of Udine
antonio.roda@uniud.it

ABSTRACT

Content-based music retrieval requires to define a similarity measure between music documents. In this paper, we propose a novel similarity measure between melodic content, as represented in symbolic notation, that takes into account musicological aspects on the structural function of the melodic elements. The approach is based on the representation of a collection of music scores with a graph structure, where terminal nodes directly describe the music content, internal nodes represent its incremental generalization, and arcs denote the relationships among them. The similarity between two melodies can be computed by analyzing the graph structure and finding the shortest path between the corresponding nodes inside the graph. Preliminary results in terms of music similarity are presented using a small test collection.

1. INTRODUCTION

One approach to content-based access to music documents is to provide users with tools to retrieve music documents that are *similar* to a set of one or more documents already known, which are the starting point of a query-by-example paradigm. The effectiveness of the results depends on the way a measure of music similarity is computed. This task is difficult to define, because the notion of music similarity is subjective and also because the role played by the different music dimensions – i.e., rhythm, melody, harmony, timbre, orchestration, tempo – is task dependent. For instance, the perceived similarity of two ballroom songs is mainly related to rhythm, while in jazz music it can depend on chord progressions.

In recent years, a major interest has been given to the retrieval of audio documents, typically in compressed formats such as MP3. This trend is explained by the increasing availability of large collections of audio files, and by the fact that users without a music training are usually not interested in accessing symbolic representations. For this reason, the notion of music similarity has been biased towards the music dimensions that, on the one hand, are more relevant for music listeners and, on the other hand, can be reliably extracted from audio files. A typical approach is to extract some timbre descriptors to address different tasks, such as genre and artist identification, automatic playlist generation, or music collection visualization [12].

Music documents can be represented also in symbolic forms, such as a notated digital score or a MIDI file. The access to these documents can be based on higher level features, such as melodic profile and harmonic progressions, which are not easily extracted from audio files. In particular, the melodic profile has been often used as the main dimension to compute the similarity between music documents [1, 7]. A typical task of melody-based retrieval is the automatic identification of a melody sung or hummed by the user. For this application, the similarity measure has to be robust to local mismatches due to imprecise recall from memory and to a lack of singing skills by the users, because it is assumed that the query and the relevant documents are representing the same information. Although the term *query-by-humming* was very popular in the early days of MIR research [3], recently it has been often replaced by the more general term query-by-example because it is assumed that users can easily record with a portable device an excerpt of the song they are interested to retrieve and are not willing to hum a melody in front of an user interface. Approaches of this kind may be based on approximate matching to identify the music work corresponding to the recorded performance, a task that is usually defined as cover [5] or music [8] identification.

The computation of melodic similarity can be useful also for applications other than an identification task. For instance, in musicological analysis the study of the melodic material used by different composers, or consistently used by a given composer, is of particular interest. Also ethnomusicological studies can take advantage from melodic similarity in order to track the evolution of a given song over the centuries and its diffusion in different geographical regions. Melodic similarity can be exploited to retrieve music that "sounds like" other well known songs, for instance to find a suitable soundtrack for a TV show.

We propose a novel similarity measure computed between music content in symbolic format, that takes into account the musical structure of the composition through the application of an analytic method. Our approach aims at representing a music collection with a graph structure where terminal nodes directly describe the music content, internal nodes represent its incremental generalization, and arcs take track of the relationships among them. Results are presented using an excerpt of the dataset used for the melodic similarity task at the Music Information Retrieval Evaluation eXchange (MIREX) campaign of 2005 [17].

2. MEASURING MELODIC SIMILARITY

A common approach to the computation of melodic similarity is to make some assumptions on the perception of pure tones and melodic intervals and to simplify the melodic representation accordingly. This process can be considered as a variant of stemming applied to the music domain, because it aims at conflating into a single stem all the melodic variants that are musically or perceptually similar. A simple example is the representation of the melodic profile using only three classes of pitch intervals – ascending, descending, and same nome – as proposed initially in [3]. Clearly, more complex representations are possible using either a finer quantization of the intervals or the analysis of the harmonic role of melodic intervals. In all these cases, it is assumed that similar melodic excerpts share the same representation. Segmentation can be applied to melodic information [10], where the similarity between melodies can be computed as a weighted sum of the similarities between pairs of segments. This latter approach aims also at efficiency, because retrieval can be based on indexing [2].

Another approach is to exploit the properties of well known distance measures, such as the edit distance using approximate string matching techniques [4] or the earth mover's distance [16], in order to deal with variants in music content. An alternative is to apply statistical modeling, such as Markov chains described in [14], to cope with local variations. The general idea is that the melody used to query the system is transformed in order to be matched with the melodies in the collection, assuming that the cost of the transformation is related to the melodic similarity. One limitation of these approaches is that they make little use of structural information and musicological analysis.

The similarity measure proposed in this paper is based on a different approach. The basic idea is that all melodies – the ones in the music collection and the query – undergo a process of generalization (or simplification). The idea is motivated by the results of musicological studies, such as the Generative Theory of Tonal Music [6], the Implication-Realization Model [9], and Schenkerian analysis [13]. In particular, we aim at finding structural dependencies among the notes of a composition in order to organize them into a coherent hierarchy. This task can be achieved by means of a series of simplifications of the melodic content of a piece, assuming that these simplifications correspond to a generalization of the melodic profile.

The central part of our approach is the determination of which notes in a music passage are more structurally significant than others. We use this property to build a hierarchical representation of a single music document and, incrementally, of a collection of documents. At each step, the

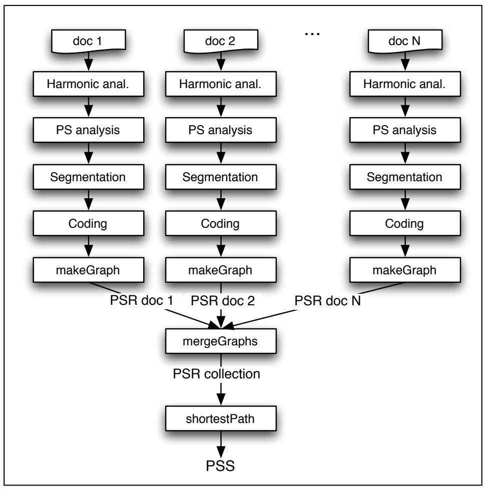


Figure 1. From the music document to the Pseudo-Structural Representation.

melodic content is analyzed and transformed in a melody with fewer notes and with a simpler profile that should represent the most musicologically relevant content information. The procedure ends when the melodic segments are represented by a single note. We called this representation Pseudo-Structural Representation (PSR), because it is inspired by structural analysis, yet the algorithm implementing it exploits some simplifications. The PSR is a graph-based representation, in which the terminal nodes are related to music surface and the internal nodes are progressive generalizations of the surface.

The steps to build the PSR of a collection of documents are depicted in Figure 1. First, each music document undergoes harmonic analysis, which highlights the harmonic function – i.e., tonic, dominant, subdominant – of each chord. Although a number of automatic routines is available for the computation of the harmonic function, in our experiments we choose to manually annotate the chords. This task, which is the only manual intervention, is in general not required for polyphonic documents for which reliable systems for inferring the chord progression already exists. Given that the evaluation has been carried out using the MIREX 2005 dataset for the Symbolic Melodic Similarity task, which documents contain only the main melody, we preferred to manually annotate the chords to not introduce possible sources of mismatch while evaluating a novel approach. Automatic chord annotation will be addressed in future work, with the aim to create a totally automatic procedure. At this stage we prefer to not introduce possible sources of preprocessing errors that are not dependent on the approach.

The second step consists in the progressive simplification of the melodic profile, by means of an algorithm inspired by musicological analyzes. First of all, the surface melody is processed to assign three weight coefficients to each note. These coefficients are related to: the underlying harmonic function (harmonic weight), the metric position (metric weight), and the pitch interval between the tone of the melody and the root of the underlying chord

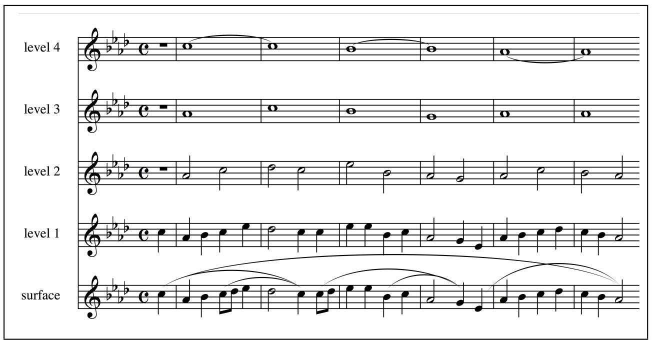


Figure 2. Example of Pseudo-Structural analysis on a Bach's Choral.

(melodic weight). A number of weighting schemes have been devised and evaluated experimentally, as presented in Section 4. The basic idea is to assign higher weights to relevant notes. For instance, a note with tonic function has a higher harmonic weight than a note with a subdominant function, the same applies to notes in strong beats in respect to notes in weak beats for the metric weight; as regards the melodic weight, a perfect fifth with the chord root has higher weight than a perfect four or a major second.

The algorithm works locally on a sliding window, progressively eliminating notes with smallest weights within a window. In the particular implementation, window length has been set equal to the double of the minimum duration of the melody, thus a window contains at most two notes. For instance, the surface melody of Figure 2 has a minimum duration of an eight-note and consequently the window length is a quarter-note. In case the window contains two notes with the same weight, the algorithm applies additional heuristics that take into account (in descending order of relevance): only melodic weight, only metric weight, only harmonic weight, and finally the relative position. The less relevant note is removed and the other one is prolonged to cover the duration of the removed note.

When the sliding window reaches the end of the melody, the first level of abstraction is completely calculated. Then, the algorithm is applied iteratively to calculate the higher levels. The algorithm stops when the highest level has one or two notes. Figure 2 shows an example of Pseudo-Structural analysis applied to the first six bars of the Bach's Choral BWV345, with four levels of progressive generalization. The higher the level the more general representation of the melodic profile.

The analyzed documents are then segmented in musical phrases, a task that can be carried out using one of the different algorithms that have been presented in the literature [11]. With the aim of separately measuring the effect of all the components, we perform segmentation both manually and automatically, using the algorithms provided by the Miditoolbox. All the approaches have been evaluated on the same test collection. Segmentation is carried out at the surface level and inherited by the higher levels. In general segments can overlap, although automatic segmentation algorithms usually provide non overlapping segments.

The subsequent step regards the coding of the pitch sequences, one for each segment. Because duration is used to perform the Pseudo-Structural analysis, it is implicitly modeled at the higher levels of generalization and not directly represented in the PSR. Pitch information is represented in the form of melodic intervals, that is the difference between two subsequent notes. Apart from the segments representing the melodic surface, pitch information undergoes different levels of quantization, from a coarse representation using three symbols (0 unison; ± 1 up/down tone) to the distance in semitones.

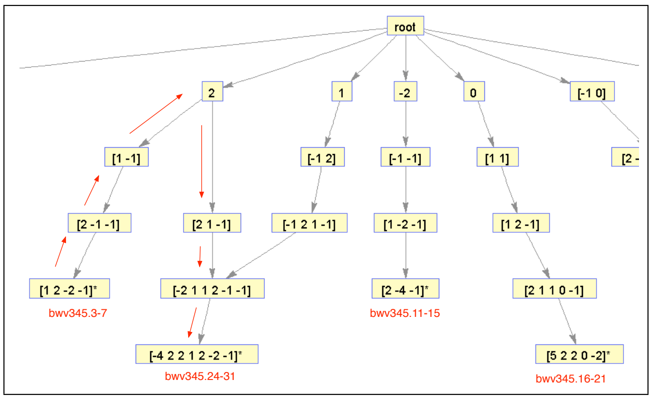


Figure 3. Pseudo-Structural Representation (partial view) of the document analyzed in Figure 2.

Each coded segment is then inserted into the PSR graph structure. In particular, *terminal nodes* contain segments of the melodic surface and *internal nodes* contain segments which are the results of the generalization process. As a result of the Pseudo-Structural analysis, an internal node is connected with a directed arc to all the nodes, either internal or terminal, that correspond to the immediate lower level of generalization. Nodes that hold the same content are joined together in a single one, that inherits the ancestors and the descendants of the starting nodes, obtaining a direct acyclic graph. An example of the PSR of a music document is shown in Figure 3. The PSR of a complete collection can be built by iterating the process for all the segments and the documents in the collection.

Given a PSR, we can define the distance between two melodic segments $s1$ and $s2$, represented by the terminal nodes $n1$ and $n2$, as the length of shortest path from $n1$ and $n2$ considering PSR as an undirect graph where all the arcs have the weight set to 1. It can be noted that PSR could be also a weighted graph, where each arc has a weight that takes into account the kind of applied simplification or the relative frequency nodes appear in a music work and in a collection. This aspect will be investigated in future work. The way the distance is computed can be described through an example. With reference to the music document shown in Figure 3, the distance between the melody segment from beat 3 to beat 7 (coded by [1 2 -2 -1]) and the melody segment from beat 24 to beat 31 (coded by [-1 2 2 1 2 -2 -1]) is equal to 6. This distance is a metric, because it is clearly reflexive and symmetric. Moreover, being based on the shortest path inside an undirect graph, it is easy to show that the triangular inequality holds.

3. APPLICATIONS

The first application of the similarity measure is a musicologically grounded approach to content-based retrieval of music documents using a query by example paradigm. For this application, retrieval can be carried out by performing the same Pseudo-Structural analysis also to the query, which is then (temporarily) added to the PSR graph, in order to compute the similarity. As for other approaches, melodic similarity of the complete documents can be computed as a weighted sum of the melodic similarity of their segments. In particular, the distance between two documents $d(c_i, c_j)$ is defined as the mean of the PSD between all the segments of c_i and c_j. The similarity $s(c_i, q)$ between c_i belonging to a collection of N documents and the query q is calculated through equation

$$s(c_i, q) = \left(1 + \frac{d(c_i, q)}{\sum_{j=1}^{N} \frac{d(c_i, c_j)}{N-1}}\right)^{-1}, \qquad (1)$$

where the similarity is proportional to the reciprocal of the distance $d(c_i, q)$ between the document c_i and the query q divided by a normalizing factor, which is the mean distance between c_i and all the other documents in the collection. The normalizing factors can be computed off-line in order to speed up retrieval.

The graph representation provides a simple way to define the maximum allowable distance between two segments. For instance, the user may choose to limit the length of the paths across the graph or to define the maximum allowed level of generalization, that is the number of times a path can jump to a higher level. Moreover, the user can be presented with a list of documents, their relevant segments, and a representation of the internal nodes that shows which is the path across the PSR that transforms the query into the retrieved segments. It is interesting to note that through this approach, the user can modify its personal view of the PSR, because past queries can be stored in the graph and eventually affect the results of the current retrieval session.

The analysis of the structure of the PSR can provide novel tools for exploring a music collection. For instance, the user can choose a branch of the PSR and explore the melodic excerpt that are represented by internal or by terminal nodes and, in the latter case, to listen to the compositions they belong to. Thus, the user can navigate inside the music segments of a collection, and their generalization based on musicological properties. To this end, informal tests showed that PSR tends to group similar composing styles in close regions of the graph. This ability of the proposed approach will be explored in future work.

4. EXPERIMENTAL EVALUATION

The methodology has been experimentally evaluated using the dataset provided for the Symbolic Melodic Similarity task at MIREX 2005 [17]. The dataset is based on the RISM collection of incipits, where relevance judgments on melodic similarity have been provided by a pool of expert musicologists. We present three different measures of retrieval effectiveness. The common measures *average precision* (AP) and *R-precision* (RP), and the *Average Dynamic Recall* (ADR) which takes into account that relevance judgments are not binary [15] and has been used as the main parameter at MIREX 2005.

# symbols	ADR	AP	R-P
3	0.65	0.60	0.54
5	0.66	0.60	0.52
7	0.65	0.59	0.51
no quantization	0.67	0.64	0.56

Table 1. Results with manual segmentation and using different levels of quantization.

segmentation	ADR	AP	R-P
manual	0.67	0.64	0.56
gestalt	0.69	0.64	0.55
probabilistic	0.67	0.61	0.53
LBDM	0.61	0.53	0.50

Table 2. Results with no quantization and using different approaches to segmentation.

Results on a subset of 110 incipits using 11 queries are reported in Table 1, showing the effect of different levels of quantization when manual segmentation is applied. The reduced size of the collection is due to the fact that, for this initial evaluation, part of the process shown in Figure 1 – annotation of the harmonic function and segmentation – has been carried out manually. As it can be seen, results are similar, with slightly better performances when no quantization is applied, although the differences are not statistically significative. This aspect should be investigated in more detail with a larger collection because a coarse quantization allows us to reduce the size of the PSR improving efficiency, yet a fine quantization preserves more information about the melodic content.

We carried out an experiment using three automatic approaches to segmentation when no quantization was applied. Segmentation algorithms are the ones provided by the Miditoolbox, which are based on gestalt concepts, on probabilistic model, and on the Local Boundaries Detection Model respectively. Results are reported in Table 2, showing that the gestalt-based approach gives results completely comparable with manual segmentation. A test on the statistical significance of the differences between these results showed that none of the differences reached the significance, thus this step can be carried out automatically without affecting retrieval effectiveness.

Other experiments have been carried out to evaluate different weighting schemes for the Pseudo-Structural analysis. As regards the harmonic weight, we tested the effectiveness of grouping the harmonic functions in 3, 4, or 7 classes (denoted with letter H). For instance, for $3H$ we grouped the harmonic functions depending on their degree on the scale, namely I and VI had the highest weight, IV and V had a intermediate weight and other degrees had the

weighting scheme	ADR	AP	R-P
$3H, MH, 3M$	0.67	0.64	0.56
$4H, MH, 3M$	0.65	0.63	0.56
$7H, MH, 3M$	0.65	0.64	0.56
$3H, MH, 4M$	0.67	0.64	0.55
$3H, MH, 7M$	0.61	0.60	0.51
$3H, MS, 3M$	0.66	0.63	0.52

Table 3. Results using different weighting schemes.

smallest weight. The metric weight was varied considering either a simple subdivision (MS) in strong and weak beats or a hierarchical organization (MH) depending on the position in the measure. Finally, the melodic weight has been tested in a similar fashion of the harmonic weight, with 3, 4, and 7 classes (denoted with letter M) where, for example, in $3M$ notes forming an interval of a unison/octave, third or fifth from the fundamental had the highest weight, a seventh had an intermediate weight, and other intervals had the smallest weight. Weighting schemes with more classes, such as $4H$ and $7H$ for harmonic weight, simply introduce new classes either taking into account new harmonic functions or splitting an existing class in two or more smaller classes. Similar considerations apply to the melodic weights.

Results are reported in Table 3, for some combination of weighting schemes, showing that the use of three classes for both the harmonic ($3H$) and the melodic ($3M$) weights with a hierarchical metric (MH) weight gave the best performances, although differences are minimal and not statistically significative. Considering that in all the experiments the three effectiveness measures are considerably higher than the ones obtained at MIREX, we can assume that with a larger collection the performances will be at least comparable with other approaches.

An important characteristic of the PSR is the relationship between the number of documents in the collection and the number of different nodes in its graph representation. It is expected that the size of the graph will increase with sublinear trend when documents are added to the PSR. Figure 4 shows this trend, where the number of new nodes – both internal and terminal – that are added with each new document decreases with the number of documents, although local variations can still be seen due to the reduced size of the test collection.

Figure 5 shows the results of a nearest neighbor query on the collection, with the aim of highlighting which features are captured by the proposed similarity measure and whether this definition can take into account for progressive differences among melodies. The searched pattern is the surface melody of the first segment of $000.109.406 - 1.1.1$ in the RISM collection. Pseudo-Structural analysis of this composition highlights that the notes at positions 2, 4, and 6 (notes B, B, and G respectively) are less relevant than the others. Indeed, all these notes are passing notes on a weak metric position. At an higher level of generalization in the PSR, these notes are therefore omitted, and the

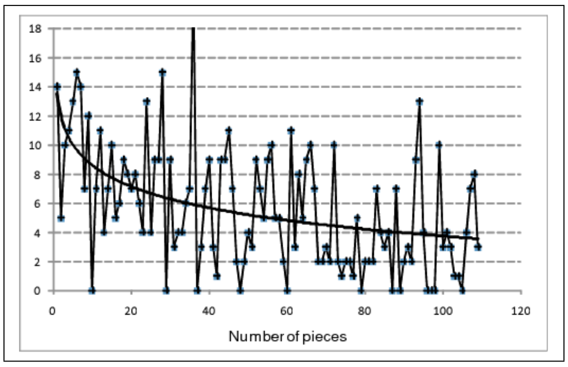


Figure 4. Number of new nodes added to the PSR graph when new music documents are added. The bold line is a logaritmic aproximation in a least square error sense.

segment is represented by an ascending melodic interval (A-C), followed by two descending melodic intervals (C-A and A-F), which correspond to the code [+2 -2 -2] using the coarsest quantization. At a distance $d = 1$, the result reports a segment with identical pitch, but with augmented duration values. It can be noted that the ratio among durations are unchanged in respect to the query segment. At a distance $d = 2$, there are segments with similar but not equal pitches, and with minor metric variations. Finally, at distance $d = 3$ and $d = 4$ there are melodic segments composed by notes that at least share with the query segment the same higher level code [+2 -2 -2].

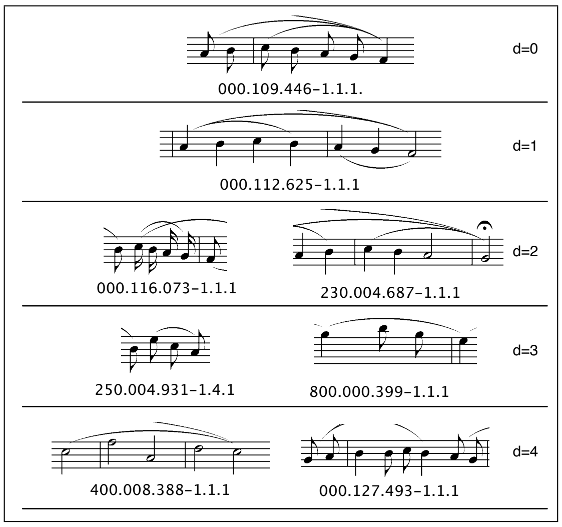


Figure 5. Results of a nearest neighbor query.

5. CONCLUSIONS

The proposed approach aims at representing the structural relationships of a music collection with an undirected graph, which is built from the analysis of the melodic content of music documents. Terminal nodes represent melodic segments of the documents, while internal nodes represent a progressive simplification/generalization of their content. Music similarity is then measured by the length of the shortest path between terminal nodes. This representation allows us to retrieve music documents that are relevant at least from a musicological point of view. Moreover, the proposed similarity is a metric because it is based on a topological distance, allowing us to efficiently carry out a number of retrieval tasks, such as range queries, k-nearest neighbor, and document clustering.

The approach has been tested with a collection of incipits. Moreover, qualitative analysis have been carried out on the relationships between the graph structure and the melodic content of the documents. Results are encouraging, both in terms of average precision of the retrieval results and in terms of musicological significance. The approach can be further exploited for browsing a collection of music documents based on the traversal of the graph representing the music documents and their relationship.

The described similarity measure is tailored to music genres where harmony plays a functional role, like in Western tonal music, because the weighting schemes presented in Section 2 are mostly based on the harmonic content. The idea itself of generalizing the melodic content through structural analysis is motivated by musicological studies on Western music. Although we believe that it is difficult, if not impossible, to develop a general purpose approach to music similarity searches, it is likely that the idea of representing music content with a hierarchical graph, where levels are associated to an incremental simplification of the musical content, can be generalized to other music features and to other genres.

A major limitation is that, at the moment, the methodology is still partially based on manual annotation of the chord progressions of the musical documents. Given the encouraging results, future work will focus on the complete automatization of the analyzes.

6. ACKNOWLEDGMENTS

The work has been partially supported by a grant of the University of Padova for the project "Analysis, design, and development of novel methodologies for the study and the dissemination of music works".

7. REFERENCES

[1] J.S. Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.

[2] J.S. Downie and M. Nelson. Evaluation of a simple and effective music information retrieval method. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 73–80, 2000.

[3] A. Ghias, J. Logan, D. Chamberlin, and B.C. Smith. Query by humming: Musical information retrieval in an audio database. In *Proceedings of the ACM Conference on Digital Libraries*, pages 231–236, 1995.

[4] H.H. Hoos, K. Renz, and M. Görg. GUIDO/MIR – an experimental musical information retrieval system based on GUIDO music notation. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 41–50, 2001.

[5] P. Herrera J. Serrá, E. Gómez and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008.

[6] F. Lerdhal and R. Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, MA, 1983.

[7] M. Melucci and N. Orio. Combining melody processing and information retrieval techniques: Methodology, evaluation, and system implementation. *Journal of the American Society for Information Science and Technology*, 55(12):1058–1066, 2004.

[8] R. Miotto and N. Orio. A music identification system based on chroma indexing and statistical modeling. In *Proceedings of the International Conference on Music Information Retrieval*, pages 301–306, 2008.

[9] E. Narmour. *The Analysis and Cognition of Basic Melodic Structures*. University of Chicago Press, Chicago, MI, 1990.

[10] G. Neve and N. Orio. A comparison of melodic segmentation techniques for music information retrieval. In *Proceedings of the European Conference on Digital Libraries*, pages 49–56, 2005.

[11] B.S. Ong. Structural analysis and segmentation of music signals. Master's thesis, Universitat Pompeu Fabra, 2006.

[12] N. Orio. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1(1):1–90, 2006.

[13] H. Schenker. *Der Freie Satz, Neue musikalische Theorien und Phantasien*. Universal Wien, O. Jonas, 1956 edition, 1935.

[14] J. Shifrin, B. Pardo, C. Meek, and W. Birmingham. HMM-based musical query retrieval. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, pages 295–300, 2002.

[15] R. Typke, R.C. Veltkamp, and F. Wiering. Evaluating retrieval techniques based on partially ordered ground truth lists. In *Proceedings of the International Conference of Multimedia and Expo*, 2006.

[16] R. Typke, F. Wiering, and R.C. Veltkamp. A search method for notated polyphonic music with pitch and tempo fluctuations. In *Proceedings of the International Conference of Music Information Retrieval*, pages 281–288, 2004.

[17] Mirex 2005 Wiki. First annual Music Information Retrieval Evaluation eXchange, July 2006. http://www.music-ir.org/mirex2005/.

MODELING HARMONIC SIMILARITY USING A GENERATIVE GRAMMAR OF TONAL HARMONY

W. Bas de Haas Utrecht University Bas.deHaas@cs.uu.nl
Martin Rohrmeier University of Cambridge mr397@cam.ac.uk
Remco C. Veltkamp Utrecht University Remco.Veltkamp@cs.uu.nl
Frans Wiering Utrecht University Frans.Wiering@cs.uu.nl

ABSTRACT

In this paper we investigate a new approach to the similarity of tonal harmony. We create a fully functional remodeling of an earlier version of Rohrmeier's grammar of harmony. With this grammar an automatic harmonic analysis of a sequence of symbolic chord labels is obtained in the form of a parse tree. The harmonic similarity is determined by finding and examining the largest labeled common embeddable subtree (LLCES) of two parse trees. For the calculation of the LLCES a new $O(\min(n, m)nm)$ time algorithm is presented, where n and m are the sizes of the trees. For the analysis of the LLCES we propose six distance measures that exploit several structural characteristics of the Combined LLCES. We demonstrate in a retrieval experiment that at least one of these new methods significantly outperforms a baseline string matching approach and thereby show that using additional musical knowledge from music cognitive and music theoretic models actually helps improving retrieval performance.

1. INTRODUCTION

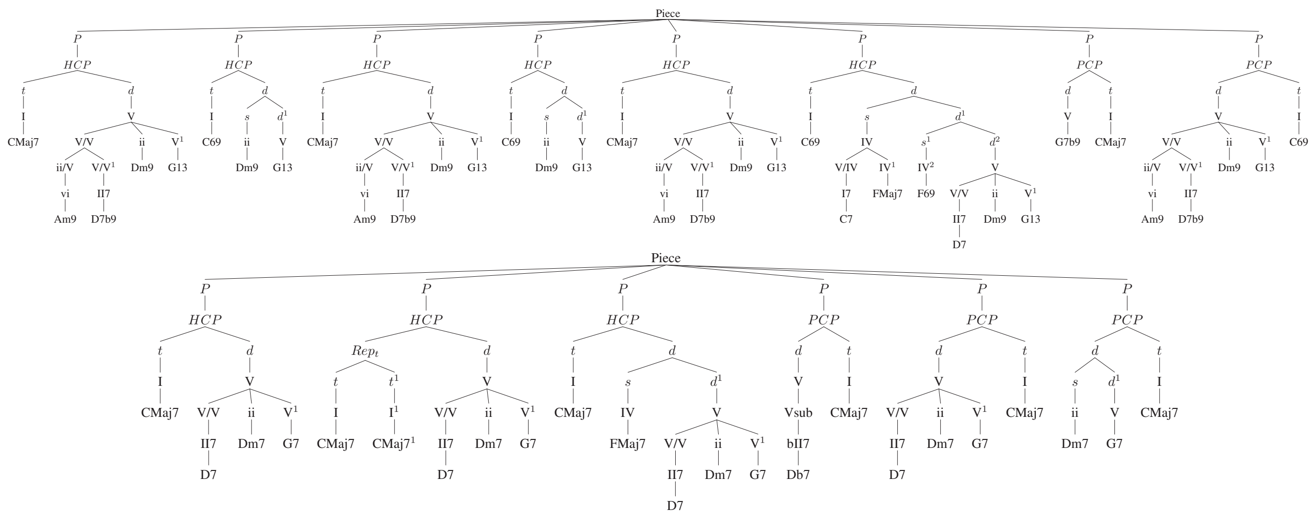
Harmonic Similarity is a relatively new research topic within Music Information Retrieval (MIR) that is concerned with determining the similarity of the chord sequences in songs and enables users to search for songs on the basis of their harmony. Retrieval based on harmony offers obvious benefits: it allows for finding cover songs (especially when melodies vary), songs of a certain family (like Blues or Rhythm Changes), or variations over standard basses in instrumental baroque music, to name a few. So far, very few measures of harmonic similarity have been proposed. De Haas et al. [1] developed a distance measure based on Lerdahl's Tonal Pitch Space [2].

When researching MIR, it is important to realize that only part of the information needed for good similarity judgment can be found in the musical data. Musically schooled as well as unschooled listeners have extensive knowledge about music [3,4] and one important task of a MIR researcher is to select or develop the appropriate music cognitive and music theoretical models that provide the knowledge needed for making good similarity judgments. We strongly believe that such a model is necessary, and that systems without such additional musical knowledge are incapable of capturing a large number of important musical features. In this study we report a new method of harmonic similarity matching that applies a remodeling of Rohrmeier's [5] phrase-structure grammar for tonal harmony as underlying cognitive and music theoretical model.

In analogy to linguistics, various hierarchical models of musical structure have been proposed since the 1980s and have been brought up recently in cognitive and computational discussions [6–8]. In this context, Rohrmeier's generative grammar of diatonic harmony [5] transfers notions about the hierarchical organization of tonal music [6,7] to the area of harmony. It is based on the assumption that, within a sequence of harmonies, different chords have different degrees of stability and dependency, based on their position within the hierarchical structure. In a chord sequence several chords may be replaced, inserted or omitted in such a way that the harmonic structure remains intact, whereas the changes of structurally important anchor chords may result in large structural modifications of the entire dependency structure of the harmony sequence. These dependency features and relationships resemble constituent structure and dependencies in linguistic syntax and can be modeled with a grammatical formalism [9].

These variable relationships between chords and their structural roles motivate the rationale not to base our harmony matching methods on sequence matching methods–which assume the equal importance of all chords in a sequence–but on a hierarchical formalization that incorporates the differences in structural function. Figure 1, displaying two versions of the jazz standard *Take the 'A' train*, illustrates this idea. Even though both sequences appear to be substantially different when compared element by element, an analysis of their formal dependencies reveals that both derive from a common harmonic pattern that is represented by the parse trees and fits human intuition.

We present a fully functional remodeling of Rohrmeier's grammar [5], which parses sequences of symbolic chord labels and returns parse trees like the ones in Figure 1, in section 3. A parse tree is more than a harmonic analysis alone, since it contains all the structural relations of the harmonies used in a song, and is therefore very suitable for determining harmonic similarity. We compare parse trees by finding and examining the combined Largest Labeled



Piece
P
HCP
PCP
t d s
I V IV
CMaj7 V/V ii V1 C69 Dm9 G13
ii/V V/V1 vi II7 Am9 D7b9
V/IV IV1 I7 C7 FMaj7 F69 D7
G7b9
Rept
I1 CMaj71
Vsub bII7 Db7
Dm7 G7

Figure 1. Two parse trees of different versions of the same jazz standard *Take the 'A' train*. The leafs of the tree represent the actual chords of the sequence.

Common Embeddable Subtree (LLCES). The LLCES is the tree that can be included in both parse trees while maintaining the labels and the ancestor relations. In section 4.2 we present a new algorithm that finds the LLCES. Using the LLCES we define a series of similarity measures for tonal harmony.

Contribution: First, we present a remodeling of a formal grammar for tonal harmony and propose solutions for some of the typical problems of its application. Second, we present a new $O(\min(n,m)nm)$ time algorithm that calculates the LLCES, where n and m are the sizes of the trees. Third, six LLCES based distance measures are defined to compare the parse trees. Last, the retrieval performance of these distance measures is experimentally verified on a dataset of 72 symbolic chord sequences of jazz standards.

2. RELATED WORK

In the last century, numerous formal theoretical approaches to western tonal music have been proposed. Formalizing Schenker's theory [6], Lerdahl and Jackendoff [7] proposed a generative theory that organized western tonal music by recursive hierarchical dependency relationships between musical elements in terms of time-span reduction and prolongation structure. They formalized the interaction between these structures with metrical and grouping structure in terms of constraint based preference rules. Similarly, there is some theoretical evidence that tonal harmony is organized in a comparable, hierarchical way. Early attempts by Kostka and Payne ([10] ch. 13) and Baroni [11] suggest that harmony is organized in hierarchical layers. Current theoretical approaches [5,12–15] suggest that the structure of harmony sequences exceeds the simplicity of a straightforward chord transition table (or finite-state grammar [9]), like Piston's table of root progressions [16], and may be modeled by hierarchical, context-free or phrase-structure grammars [9]. Pachet [17] proposes a set of rewrite rules for jazz harmony similar to Steedman's grammar [12]. He shows that these rules could be learned form chord sequence data in an automated fashion. Rohrmeier [5] gave an encompassing account how tonal harmonic relationships may be formalized using a generative context-free grammar with variable binding.

3. A GRAMMAR FOR TONAL HARMONY

The generative formalism proposed by Rohrmeier [5] expands on earlier approaches in a number of ways. Steedman's approach [12,13] is merely concerned with Blues progressions and, featuring only seven context-sensitive rules (with variations), omits a number of theoretically important features to extend to a broader domain. Rohrmeier's approach extends on these ideas and gives an overarching account of tonal harmony and tonal-phrase structure independently of a specific style or musical form. In addition, it proposes to incorporate the structural distinctions between form, theoretical harmonic function [18], scale degree prolongation [6,7] and surface feature realization into different levels of the syntactic derivation. The present study proposes a remodeling of the grammar without modulation and with limited local tonicization and scale adaptation in order to reduce the complexity for the implementation of a first-stage working system. The current remodeling was optimized for jazz, but the aim is to develop a set of core rules that explain basic harmony structures which can be augmented with style specific rules.

The grammar incorporates four levels: a phrase level, functional level, scale-degree level and surface level. The phrase level divides a piece into phrases, the functional level specifies the functional role a certain scale-degree has within a phrase. The scale-degree captures the relation between the chord and the key and the surface level expresses the actual chord with all its possible additions, inversions, etc.

Below, the main rules of the grammar are listed in order to give an outline of the architecture of the grammar. A piece always consists of one or more phrases (P). On this phrase level the grammar distinguishes two types of phrases: phrases which end on a perfect cadence (PCP) and phrases which end with a half-cadence (HCP). Perfect cadence phrases are distinguished by ending with a tonic function (t) upon which all subordinate harmonic elements are dependent, whereas half-cadence phrases force a phrase to end with a dominant function (d) which results in a tonicization of, or a perfect or imperfect cadence on the dominant.

1. $Piece \rightarrow P+$
2. $P \rightarrow PCP$
3. $P \rightarrow HCP$
4. $PCP \rightarrow d\ t+ \mid d\ d\ t+ \mid t\ d\ t$
5. $HCP \rightarrow t+\ d$

At the functional level, the grammar encapsulates core relationships between the three main harmonic functions: tonic (t), dominant (d) and subdominant (s).

6. $d \rightarrow s\ d$
7. $t \rightarrow tpg$

These functional rules can be applied recursively, but finally translate into scale-degrees. Rule 9 deals with certain forms of parallels (tpg).

8. $t \rightarrow$ I
9. $tpg \rightarrow$ vi | iii
10. $d \rightarrow$ V | vii^0
11. $s \rightarrow$ ii | IV

The functional level also incorporates a number of additional prolongational rules that allow for the derivation of more distant harmonic structures such as the preparatory use of iii and tritone substitutions. Rule 12 incorporates a feature specifically added for modeling the prototypical II-V-I sequences in jazz harmony that are less frequent in other styles.

12. $x \rightarrow$ V(x) x | ii(x) V(x) x for any scale degree x
13. IV $\rightarrow$ iii IV
14. V(x) $\rightarrow$ bII(x) for any scale degree x

At the surface level scale degree symbols are translated into the actual surface chord. These translation steps are straightforward when the key is known beforehand. For instance, a VI symbol in the key of C minor would translate into an A♭-chord. In addition, elaborations of chords are added at this level of description: a surface realization of a VI chord may result in a A♭6 chord. Some of these surface elaborations of chords are tied to their structural functions (strong typing), e.g. an Em7♭5 chord label indicates a subdominant function ii in a ii-V-I sequence, or a D7 chord label indicates a dominant function (except in blues contexts where minor sevenths loose their functional connotation).

3.1 Implementation and Parsing

There are some additional rules that have been implemented, but are not described here. Among these are rules for typical voice-leading and prolongational structures and some typical borrowings from the parallel key. Since we have not incorporated modulation yet, it is necessary to label these phenomena to be able to explain the remainder of the piece. Furthermore, there are rules that deal with typical well-known diminished chord transitions in various descending and ascending forms.

The grammar as specified above is not strictly a context free grammar, because rule 12 and rule 14 use a variable binding. However, by expanding a rule for every element x that it holds, a set of context free rules can be created that yields the exact same outcome. Having a context free grammar, a free Java implementation [19] of an Earley Parser [20] is used to parse the chord sequences in $O(n^3)$ time, where n is the number of chords.

Context free grammars often create multiple ambiguous parse trees. To select the optimal parse tree out of the set of parse trees, we provided the rules with weights (set by hand) and defined the total weight of a parse tree as the product of the weights of the rules used in its construction. Because of this, some rules have less chance to be used in the final derivation. This allows to select the best tree from the ambiguous parse trees. The complete grammar as well as the lead-sheets of the examples in Fig 1 are available online [1] .

4. COMMON HIERARCHICAL STRUCTURES

In this section we present six distance measures for the parse trees generated by the grammar discussed in the previous section. For the comparison of parse trees, we propose an approach based on the problem of tree inclusion, which is elaborately dealt with in [21]. Given the parse trees of two songs, the general idea is to find the collection of largest labeled common embeddable subtrees (LLCESs) for every combination of phrases. The LLCES is the largest tree that is included in both parse trees. This means that there exists a one-to-one mapping from the nodes of the LLCES to the nodes with the same label in both parse trees that preserves ancestor relations, but not necessarily parent relations. When processing harmony parse trees, this is a natural thing to do because typically a chord progression is augmented by adding a structure to the left branch and repeating the right branch, e.g. when a Dm is prepared by an A7 chord. Hence, if both trees are similar, the LLCES reflect the structure of the parse trees it is generated from, and if both trees are dissimilar, the resulting LLCES will be much smaller and less grammatical. In the next sections we explain the calculation of the LLCES, and how we use it to define six distance measures.

4.1 Preliminaries

A rooted tree is a structure denoted with $P = (V, E, P_{root})$, where V is a finite set of nodes, E is the set of edges con-

[1] http://give-lab.cs.uu.nl/music/

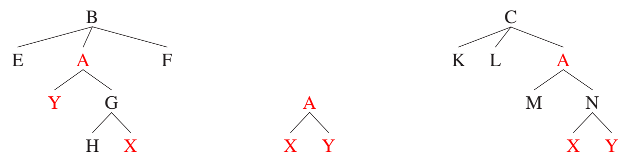


Figure 2. An example of a rooted by A that can be embedded into two larger trees rooted by B and C.

necting nodes in V, and P_{root} is the root of the tree P. The nodes of the parse trees generated by the grammar of section 3 are all labeled and the label of node v is denoted with $label(v)$. The subtree of P rooted at node v is denoted with $P[v]$ and $children(v)$ denotes the subset of V with nodes that have v as parent. Similarly $desc(v)$, denotes the decedents of v, i.e. the subset of V that is contained in $P[v]$ and have v as ancestor. Furthermore we use a few additional functions, $po(v)$ denotes the post order number that is assigned to a node v in a postorder traversal. $depth(P)$ denotes the depth of a tree, i.e. the number of nodes in the longest path from leaf to the root. Finally, the $degree(P)$ is the degree of a tree, i.e. the maximum number of children.

We say that a tree $P = (V, E, P_{root})$ is *included* in a tree $T = (W, F, T_{root})$ if there exists an embedding of P into T. An embedding is an injective function f, mapping each node in P to a node in T, that preserves labels and ancestorship. Figure 2 shows an example of an included tree. Note that a left-to-right ordering of the descendants is not required. Formally, this means that for all nodes u and v in P it is required that:

1. $f(u) = f(v)$ if and only if $u = v$,
2. $label(u) = label(f(v))$,
3. u is an ancestor of v in P if and only if $f(u)$ is an ancestor of $f(v)$ in T.

4.2 Largest Labeled Common Embeddable Subtree

We are not aware of an algorithm that calculates the largest common embeddable subtree for labeled trees. Gupta and Nishimura [22] have developed a $O(n^{2.5} \log n)$ time algorithm for finding this tree for two unlabeled trees. The algorithm we present here calculates the largest common embeddable subtree for the labeled case and expands on the general tree matching ideas as described in [21], ch. 3.

Algorithm 1 calculates the LLCES of two trees $P = (V, E, P_{root})$ and $T = (W, F, T_{root})$. To store the nodes of the subtrees of the LLCES the algorithm uses a table M such that $M[po(w)]$ stores the subtrees that can be embedded into both P and $T[w]$. The algorithm builds the LLCES up from the leaves by traversing the nodes of T and P in postorder. When a node v with an identical label as w is encountered (line 5), the algorithm creates a new node x with the same label as v. In case w is a leaf, x is stored in M (lines 8–9). In case w is an internal node, we look up the subtrees in M that match the children of w. Because the tree is processed in postorder these nodes were previously stored in M and can be retrieved from $M[po(w')]$ for each child w'. If a previously stored subtree rooted by

Algorithm 1 Largest Labeled Common Embeddable Subtree

```
1:  procedure LLCES($P$, $T$)
2:      $M \leftarrow \varnothing$
3:      for all $w \in W$ in postorder do
4:          for all $v \in V$ in postorder do
5:              if $label(v) = label(w)$ then
6:                  $x \leftarrow$ new node
7:                  $label(x) \leftarrow label(v)$
8:                  if $children(w) = \varnothing$ then
9:                      add $x$ to $M[po(w)]$
10:                 else
11:                     for all $w' \in children(w)$ do
12:                         for all $x' \in M[po(w')]$ do
13:                             if $x' \in desc(v)$ then
14:                                 add $(x, x')$ to $M[po(w)]$
15:                             else
16:                                 add $x'$ to $M[po(w)]$
17:                             end if
18:                         end for
19:                     end for
20:                     add $x$ to $M[po(w)]$
21:                 end if
22:             end if
23:         end for
24:         if $M[po(w)] = \varnothing$ then
25:             for all $w' \in children(w)$ do
26:                 add $M[po(w')]$ to $M[po(w)]$
27:             end for
28:         end if
29:     end for
30:     return $M[po(T_{root})]$
31: end procedure
```

x' is a descendant of v, this subtree becomes a child of the new node x, by adding a new edge (x, x') to $M[po(w)]$ (lines 10–15). Otherwise, if x' is not a descendant of v, x' is stored in $M[po(w)]$ (line 16). After all, a common ancestor can show up in a next iteration. Finally, the new subtree x is stored in M as well (line 20). If the label of w does not match any of the labels of the nodes in P, the subtrees stored in M for all children w' of w are added to $M[po(w)]$ (lines 24–28). This process continues until all nodes of T have been matched against all nodes of P and finally $M[po(T_{root})]$, the LLCES of P and T, is returned. A drawback of our algorithm is that it is incapable of dealing with duplicate labels. Therefore we number the duplicate labels that descent the same phrase.

The running time of the algorithm is dominated by the lines 3-23. For each of the $O(nm)$ combinations of w and v (lines 3, 4) a constant time test is performed. Because the labels are unique, only $\min(n, m)$ times each of the $O(n)$ nodes in the subtrees that has been stored in $M[po(w)]$ so far (line 12), is checked against each of the $O(m)$ descendants of node v (line 13). This results in a time complexity for the whole algorithm of $O(\min(n, m)nm)$.

4.3 Distance Measures

We base the distance measures on the LLCES, but we do not calculate the LLCES of two parse trees directly for two reasons. First, as we can see in Figure 1, there are quite some duplicate labels in the parse trees which our algorithm cannot handle. Second, if a parse tree of a song contains a repetition of a phrase that the matched song does not have, the repeated phrase cannot be matched. To solve

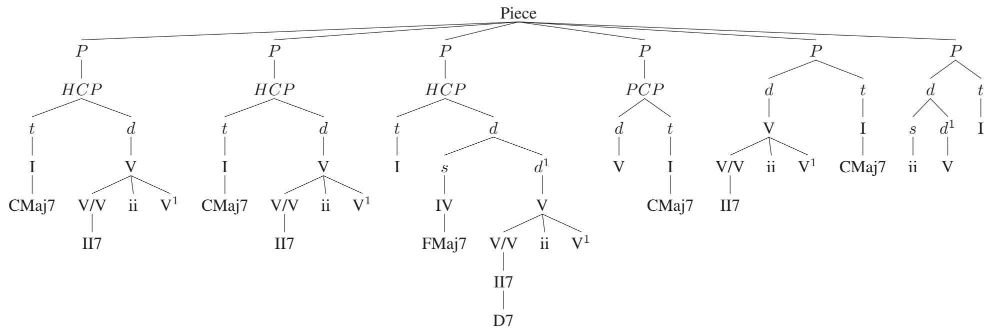


Figure 3. The Combined LLCES for every combination of phrases of the parse trees of *Take the 'A' Train* as in Fig. 1.

these two problems we compare parse trees in a phrase-wise fashion. For every phrase in the target tree T we calculate the LLCES for every phrase of the pattern parse tree P and pick the largest LLCES to create a *Combined LLCES* (see Fig. 3). The duplicate labels are re-labeled per phrase too in preorder (see the superscripts in Fig. 1 and 3). Because the number of duplicate labels per phrase is small, the labellings will be nearly identical if two trees have a similar harmonic structure. Note that if the two parse trees T and P have a different number of phrases, the structure of the Combined LLCES will differ if P is used as a target tree, because for every phase in the T a LLCES is constructed. This makes every Combined LLCES based measure is asymmetrical.

We propose three distance measures for sequences of symbolic chord labels based on the Combined LLCES:

1. *Relative Combined LLCES Size Distance (Rel)*: By dividing the number of nodes in the target tree T by the number of nodes in the Combined LLCES a distance measure between 0 and 1 is obtained that is normalized by the size of T.
2. *Grammar Violation Distance (Viol)*: if two trees are not similar, the combined LLCES will contain connections between nodes that cannot be explained by the grammar. By dividing the number of nodes in the target tree T (which are grammatical by definition) by the number of grammatical nodes in the Combined LLCES we obtain a distance measure between 1 and 0 that is normalized by the size of T.
3. *Average Depth Distance (Dep)*: if trees are very similar, the level of complexity in the harmonic structure in the Combined LLCES will be comparable to the level of complexity target tree T. By dividing the average leaf depth of T by the average leaf depth of the Combined LLCES, we obtain a distance measure between 1 and 0, that is normalized by the size of T.

One can observe in Figure 1 that, having the actual parse tree structure, the actual chord labels are not of much importance anymore. Given two similar sequences, it is rather arbitrary whether the chords labels match or not: the structure of the harmony determines the similarity. Therefore we can remove each leaf node describing a surface chord from the Combined LLCES and target trees. The structure of the phrase, functional and scale-degree level remains unchanged. As a consequence, this yields three additional harmonic distance measures that are concerned with the structure of the harmony only. Other Combined LLCES distance measures can be thought of.

5. EXPERIMENT

We have evaluated the six LLCES based distance measures described in the previous section in an experiment. We assembled a dataset of 72 symbolic chord label sequences extracted from user-generated Band-in-a-Box files that were collected on the Internet. Band-in-a-Box is a software package that generates accompaniment given a certain chord sequence provided by the user. This dataset consists of 51 different pieces of which 16 pieces contain two or three versions, forming 16 song classes. These pieces are all jazz standards from before the 1970's and can all be found in the Real Book [23] or similar sources. All parse trees of these pieces are available online[2]. The task is to retrieve the other versions of a song class, given a certain query song from that class. All songs containing more than one version are used as a query and the rankings are analyzed by calculating the mean average precision (MAP). To place the results in perspective, we calculate the edit distance [24] between all chord sequences, represented as a string of chord labels, as a baseline measure.

The results are presented in Table 1. It seems that all Combined LLCES based methods perform better than the baseline edit distance, but only the difference between the *Viol* distance measure without chord symbol nodes scores significantly better than the baseline edit distance ($p < .01$, two-tailed T-test). The results show therefore that the number of grammatical connections in the Combined LLCES is a good indicator for harmonic similarity. The lack of significance of the other measures might be explained by the limited size of the relatively small dataset. However, the experiment does show that a matching method that analyzes the structure of the harmony outperforms a sequence-based method that does not use any musical knowledge.

6. CONCLUDING REMARKS

This paper introduced a new approach to harmonic similarity. We showed that a grammar of tonal harmony can

[2] http://give-lab.cs.uu.nl/music/

	Chord Symbols			*No Chord Symbols*			
Distance:	Rel	Viol	Dep	Rel	Viol	Dep	Edit
MAP:	0,79	0,81	0,72	0,81	0,86	0,73	0.67

Table 1. The MAP of the six Combined LLCES based similarity measures and a baseline edit distance.

be adapted in such a way that is usable for matching harmony sequences. However, there are some open issues. At the moment we cannot calculate distance measures to pieces that do not parse and for every grammar there are always pieces imaginable that do not parse. A solution to this problem can be found in partial matching. Often only one or two chords cannot be explained by the grammar. By removing these chords and parse the left and the right side separately, it is possible to obtain a parse tree that can be used for matching.

A property of context free grammars is that sequences can have multiple ambiguous parse trees. Using the grammar presented here, many chord sequences are intrinsically ambiguous and have multiple derivations. One solution might be to incorporate intrinsically ambiguous parse trees in the creation of the Combined LLCES. Nevertheless, it is important to keep the number of unwanted ambiguous parse trees as low as possible. By making the grammar strongly typed and adding weights to rules, we controlled the number and the selection of parse trees. Still, the grammar as presented here features several problems with respect to the parsing of phrase boundaries, which constitutes a main source of ambiguities (as in Fig. 1). A set of additional preference rules will be designed for future versions of the model to rule out unlikely phrase-boundaries. These will be based on metrical information which is not yet incorporated in the present model. Yet another way of improving the expressive power of the grammar and limiting the number of ambiguous parse trees at the same time, is to start parsing with a very strict grammar and, only after a rejection of the chord sequence, to add more loosely typed rules that can explain the more exotic harmonic phenomena.

The research presented here demonstrates how a grammar of harmony may characterize harmonic similarity in a musical way. This will have a large impact on the quality of the representation, analysis and retrieval of tonal music. This research also provides a case study that demonstrates the importance of cognitive and theoretic models of music in the design of appropriate methods for MIR tasks that have been neglected so far because of their inherent musical complexity.

7. ACKNOWLEDGMENTS

This work was supported by the Dutch ICES/KIS III Bsik project MultimediaN and in part by Microsoft Research through the European PhD Scholarship Programme.

8. REFERENCES

[1] W.B. de Haas, R.C. Veltkamp, and F. Wiering. Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 51–56, 2008.

[2] F. Lerdahl. *Tonal Pitch Space*. Oxford University Press, 2001.

[3] I. Deliège, M. Mélen, D. Stammers, and I. Cross. Musical Schemata in Real Time Listening to a Piece of Music. *Music Perception*, 14(2):117–160, 1996.

[4] E. Bigand. More About the Musical Expertise of Musically Untrained Listeners. *Annals of the New York Academy of Sciences*, 999:304–312, 2003.

[5] M. Rohrmeier. A Generative Grammar Approach to Diatonic Harmonic Structure. In Anagnostopoulou Georgaki, Kouroupetroglou, editor, *Proceedings of the 4th Sound and Music Computing Conference*, pages 97–100, 2007.

[6] H. Schenker. Der Freie Satz. Neue musikalische Theorien und Phantasien, 1935.

[7] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1983.

[8] A.D. Patel. Language, Music, Syntax and the Brain. *Nature Neuroscience*, 6:674–681, 2003.

[9] N. Chomsky. *Syntactic Structures*. Mouton, 1957.

[10] S. Kostka and D. Payne. *Tonal Harmony with an Introduction to 20th-century Music*. McGraw-Hill, 1984.

[11] M. Baroni, S. Maguire, and W. Drabkin. The Concept of Musical Grammar. *Music Analysis*, 2(2):175–208, 1983.

[12] M. J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception*, 2(1):52–77, 1984.

[13] M. J. Steedman. *The Blues and the Abstract truth: Music and Mental Models*, chapter 15, pages 305 – 318. Psychology Press, 1996.

[14] M. Chemillier. Toward a Formal Study of Jazz Chord Sequences Generated by Steedmans Grammar. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9):617–622, 2004.

[15] S. Tojo, Y. Oka, and M. Nishida. Analysis of Chord Progression by HPSG. In *Proceedings of the 24th IASTED international conference on Artificial intelligence and applications*, pages 305–310. ACTA Press Anaheim, CA, USA, 2006.

[16] W. Piston. *Harmony*. Norton, W. W. & Company, New York, 1948.

[17] F. Pachet. Surprising Harmonies. *International Journal of Computing Anticipatory Systems*, 4, 1999.

[18] H. Riemann. *Vereinfachte Harmonielehre; oder, die Lehre von den tonalen Funktionen der Akkorde*. Augener, 1893.

[19] S. Martin. Pep is an Earley Parser. http://www.ling.ohio-state.edu/˜scott/, 2007.

[20] J. Earley. An Efficient Context-free Parsing Algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[21] P. Kilpeläinen. *Tree Matching Problems with Applications to Structured Text Databases*. PhD thesis, Departement of Computer Science, University of Helsinki, November 1992.

[22] A. Gupta and N. Nishimura. Finding Largest Subtrees and Smallest Supertrees. *Algorithmica*, 21(2):183–210, 1998.

[23] Various Authors. *The Real Book*. Hal Leonard Corporation, 6th edition, 2004.

[24] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

SYMBOLIC AND STRUCTURAL REPRESENTATION OF MELODIC EXPRESSION

Christopher Raphael
School of Informatics and Computing
Indiana Univ., Bloomington

ABSTRACT

A method for expressive melody synthesis is presented seeking to capture the structural and prosodic (stress, direction, and grouping) elements of musical interpretation. The interpretation of melody is represented through a hierarchical structural decomposition and a note-level prosodic annotation. An audio performance of the melody is constructed using the time-evolving frequency and intensity functions. A method is presented that transforms the expressive annotation into the frequency and intensity functions, thus giving the audio performance. In this framework, the problem of expressive rendering is cast as estimation of structural decomposition and the prosodic annotation. Examples are presented on a dataset of around 50 folk-like melodies, realized both from hand-marked and estimated annotations.

1. INTRODUCTION

A traditional musical score represents music *symbolically* in terms of notes, formed from a discrete alphabet of possible pitches and durations. Human performance of music often deviates substantially from the score's literal interpretation, by inflecting, stretching and coloring the music in ways that bring it to life. *Expressive music synthesis* seeks algorithmic approaches to this expressive rendering task, so natural to humans.

There is really a great deal of past work on expressive synthesis — more than can be summarized here, though some of the leading authors give an overview of several important lines of work in [1]. Most past work, for example [2], [3], [4], as well as the many RENCON piano competition entries, for example [5] [6], has concentrated on piano music. The piano is attractive for one simple reason: a piano performance can be described by giving the onset time, damping time, and initial loudness of each note. Since a piano performance is easy to represent, it is easy to define the task of expressive piano synthesis as an estimation problem: one must simply estimate these three numbers for each note.

This work supported by NSF grants IIS-0739563 and IIS-0812244

In contrast, we treat here the synthesis of *melody*, which finds its richest form with "continuously controlled" instruments, such as the violin, saxophone or voice. This area has been treated by a handful of authors, including the KTH group [7], [8], as well as a number others, including a commercial singing voice system. Continuously controlled instruments simultaneously modulate many different parameters, leading to wide variety of tone color, articulation, dynamics, vibrato, and other musical elements, making it difficult to represent the performance of a melody. However, it is not necessary to replicate any of these familiar instruments to effectively address the heart of the melody synthesis problem. We will propose a minimal audio representation we call the theremin, due to its obvious connection with the early electronic instrument by the same name [9]. Our theremin controls only time-varying pitch and intensity, thus giving a relatively simple, yet capable, representation of a melody performance.

The efforts cited above include some of the most successful attempts to date. All of these approaches map observable elements in the musical score, such as note length and pitch, to aspects of the performance, such as tempo and dynamics. One example is the rule-based KTH system, which grows out of several decades of focused effort. In this system, each rule maps various musical contexts into performance decisions, which can be layered, so that many rules can be simultaneously applied. The rules were chosen, and iteratively refined, by a music expert seeking to articulate and generalize a wealth of experience into performance principles. In contrast, the work of Widmer [2], [4] takes a machine learning perspective by *automatically* learning rules from actual piano performances. We share the perspective of machine learning. In [4], phrase-level tempo and dynamic curve estimates are combined with the learned rule-based prescriptions, through a case-based reasoning paradigm. That is, this approach seeks musical phrases in a training set that are "close" to the phrase being synthesized, using the tempo and dynamic curves from the closest training example. As with the KTH work, the performance parameters are computed directly from the observable score attributes with no real attempt to describe any *interpretive* goals such as repose, passing tone, local climax, surprise, etc.

Our work differs significantly from these, and all other past work we know of, by explicitly trying to *represent the interpretation itself.* Previous work does not represent the interpretation, but rather treats the *consequences* of this interpretation, such as dynamic and timing changes. We represent the interpretation in two ways. This first uses a tree-like structural decomposition that makes explicit various levels of repetition or parallelism in the melody. This idea is familiar from other work such as [3], though we introduce a framework for automatically estimating the structure. This approach has connections with [10], which finds phrase decompositions from symbolic music. Secondly, we introduce a hidden sequence of variables representing the prosodic interpretation (stress and grouping) itself, by annotating the role of each note in the larger prosodic context. We believe these representations are naturally positioned between the musical score and the observable aspects of the interpretation. Thus the separate problems of estimating the representations and generating the actual performance from the representations require shorter leaps, and are therefore easier, than directly bridging the chasm that separates score and performance.

2. THE THEREMIN

Our goal of expressive melody synthesis must, in the end, produce actual sound. We introduce here an audio representation we believe provides a good trade-off between expressive power and simplicity.

Consider the case of a sine wave in which both frequency, $f(t)$, and amplitude, $a(t)$, are modulated over time:

$$s(t) = a(t)\sin(2\pi \int_0^t f(\tau)d\tau). \tag{1}$$

These two time-varying parameters are the ones controlled in the early electronic instrument known as the *theremin*. Continuous control of these parameters can produce a variety of musical effects such as expressive timing, vibrato, glissando, variety of attack and dynamics. Thus, the theremin is capable of producing a rich range of expression. One significant aspect of musical expression the theremin *cannot* capture is tone color — as a time varying sine wave, the timbre of the theremin is always the same. Partly because of this weakness, we have modified the above representation to allow tone color to change as a function of amplitude:

$$s(t) = \sum_{h=1}^{H} A_h(a(t), f(t))\sin(2\pi h \int_0^t f(\tau)d\tau) \tag{2}$$

where the $\{A_h\}$ are hand-designed functions, monotonically increasing in the first argument. Thus our sound is still parametrized by $f(t)$ and $a(t)$, while we increase the perceived dynamic range.

3. REPRESENTING MUSICAL INTERPRETATION

There are, no doubt, more aspects of musical interpretation than can possibly be treated here. Palmer [11] gives a very nice overview of current thinking on this subject from the Psychology perspective. Broadly speaking, there are

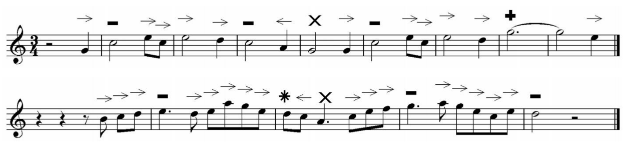


Figure 1. *Amazing Grace* (**top**) and *Danny Boy* (**bot**) showing the note-level labeling of the music using symbols from our alphabet.

at least three important components to musical interpretation: conveying musical structure, and, in particular, the way it relates to the notion of *phrase*; musical prosody — the placing, avoidance, and foreshadowing of local (note-level) stress and the associated low-level groupings that follow; and musical affect such happy, sad, intense, agitated, etc. We will focus only on phrase structure and prosody here, acknowledging that this is only a piece of the larger interpretive picture.

The folk-like music we treat here is mostly composed of simple musical structure, with a high degree of repetition of rhythm, pitch contour, chord sequence, and other musical elements. Typically the hierarchical structure of these melodies is captured by simple tree structures, often involving binary groupings at various levels of grouping: it is no accident that 34 out of the 48 melodies in our dataset have 2^n measures for some n. Within this hierarchy, musical phrases correspond to “levels” of this tree. When a melody is not captured by a perfectly regular tree structure, it often corresponds to the concatenation of such regular trees. For instance, the familiar melody, *God Save the Queen*, may be described (2-2-2)+((2-2)-(2-2)) where each number represents a group of measures, '+' denotes concatenation and '-' denotes grouping. Thus the melody has 3 groups of two measures followed by a two levels of binary structure for the last eight measures. While there is a subjective component to the partition into *phrases*, the first 6 and last 8 measures seem like reasonable choices, perhaps splitting the last 8 measures into two 4-bar phrases. In this example phrase boundaries correspond exactly to measure boundaries, though often this is not the case. Thus we must also indicate the length of the “pickup” for each group of measures.

While conveying musical structure is an important part of expressive synthesis, the main focus of our effort here is on musical *prosody*. We introduce now a way of *representing* the desired musicality in a manner that makes clear interpretive choices and conveys these unambiguously. Our representation labels each melody note with a symbol from a small alphabet,

$$A = \{l^-, l^\times, l^+, l^\rightarrow, l^\leftarrow, l^*\}$$

describing the role the note plays in the larger context. These labels, to some extent, borrow from the familiar vocabulary of symbols musicians use to notate phrasing in printed music. The symbols $\{l^-, l^\times, l^+\}$ all denote stresses or points of “arrival.” The variety of stress symbols allows for some distinction among the kinds of arrivals we

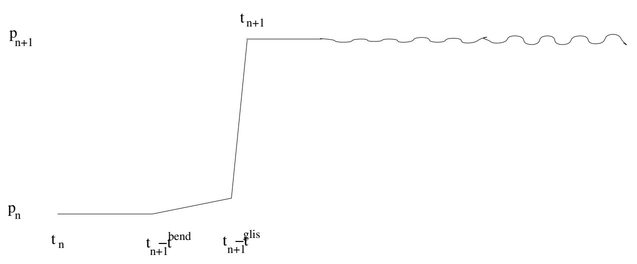


Figure 2. A graph of the frequency function, $f(t)$, between two notes. Pitches are bent in the direction of the next pitch and make small *glissandi* over the transitions.

can represent: l^- is the most direct and assertive stress; $l^\times$ is the "soft landing" stress in which we relax into repose; l^+ denotes a stress that continues *forward* in anticipation of future unfolding, as with some phrases that end in the dominant chord. Examples of the use of these stresses, as well as the other symbols are given in Figure 1. The symbols $\{l^\rightarrow, l^*\}$ are used to represent notes that move *forward* towards a future goal (stress). Thus these are usually shorter notes we pass through without significant event. Of these, $l^\rightarrow$ is the "garden-variety" passing tone, while l^* is reserved for the passing stress, as in a brief dissonance, or to highlight a recurring beat-level emphasis, still within the context of forward motion. Finally, the $l^\leftarrow$ symbol denotes receding movement as when a note is connected to the stress that precedes it. This commonly occurs when relaxing out of a strong-beat dissonance *en route* to harmonic stability. We will write $x = x_1, \ldots, x_N$ with $x_n \in A$ for the prosodic labeling of the notes.

These concepts are illustrated with the examples of *Amazing Grace* and *Danny Boy* in Figure 1. Of course, there may be several reasonable choices in a given musical scenario, however, we also believe that most labellings do *not* make interpretive sense and offer evidence of this is Section 7. Our entire musical collection is marked in this manner and available at

http://www.music.informatics.indiana.edu/papers/ismir09

4. FROM LABELING TO AUDIO

Ultimately, the prosodic labeling of a melody, using symbols from A, must be translated into the amplitude and frequency functions we use for sound synthesis. We have devised a deterministic mapping from our prosodically-labeled score to the actual audio parameter outlined here.

Our synthesis of $f(t)$ and $a(t)$ begins by modifying the literal interpretation of musical timing expressed in the score to include *ritardandi* (slowing down) at the ends of phrases. While we have not done so here, [3] recommends larger changes at higher levels of the phrase hierarchy, as expressed by our structural representation. We further modify $f(t)$ to include vibrato to long and stressed notes. Finally, we bend each pitch in towards the following pitch with a final *glissando* to encourage a sense of legato. Figure 2 shows a short piece of this pitch function over the two consecutive two notes.

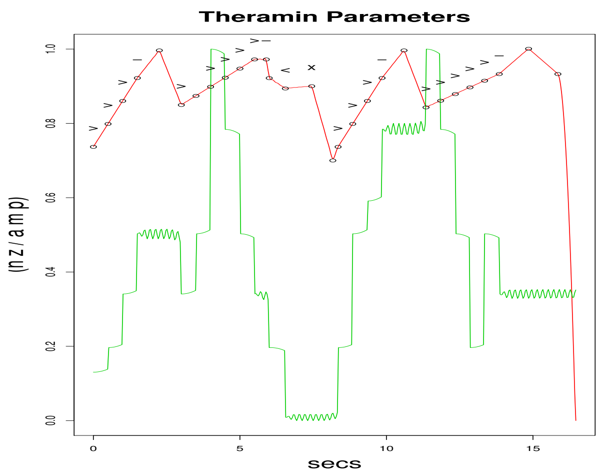


Figure 3. The functions $f(t)$ (green) and $a(t)$ (red) for the first phrase of *Danny Boy*. These functions have different units so their ranges have been scaled to 0-1 to facilitate comparison.

The heart of the transformation, however, is in the construction of the amplitude function $a(t)$. This function is created through a series of soft constraints that are placed on the amplitude defined at various "knot" locations over time. These constraints are taken from from the prosodically-annotated score and the structural representation. For instance, we want phrase beginnings, as indicated by the structural representation, to be low in amplitude; thus we add a quadratic penalty that encourages this characteristic. Similarly, we want stressed notes to be high in amplitude and add similar quadratic penalties to encourage this. In addition we want forward-moving notes to be increasing in amplitude, and thus add quadratic terms that encourage this relationship between a forward-moving note and its successor. Similar terms are added for receding notes. We then compute the values at the knot locations by minimizing the quadratic penalty function, and interpolate the resulting amplitudes at the knot locations. A more detailed presentation of this process is described in [12]. An example of both the $a(t)$ and $f(t)$ functions for a familiar examples are given in Figure 3.

5. HOW MUCH MUSICALITY DOES THE REPRESENTATION CAPTURE?

The theremin parameters, $f(t), a(t)$, and hence the audio signal, $s(t)$, depend entirely on the structural representation, the prosodic labeling, and the musical score, through the mapping described in Section 4. We want to understand the degree to which our representation captures musically important interpretive notions. To this end, we have constructed a dataset of about 50 simple melodies containing a combination of genuine folk songs, folk-like songs, Christmas carols, and examples from popular and art music of various eras. The melodies were chosen to be familiar, having simple chords, simple phrase structure, all at moderate to slow tempo, and appropriate for *legato* phrasing. Examples include *Danny Boy*, *Away in a Manger*, *Loch Lomond*, *By the Waters of Babylon*, etc. These melodies were painstakingly hand-annotated with structure and prosody by the author.

We rendered these melodies into audio according to our hand-marked annotations and the process of Section 4. For each of these audio files we provide harmonic context by superimposing sustained chords, as indicated in the scores. The entire collection of symbolic melodies, along with rendered audio files, is available at the aforementioned web site.

We do observe some aspects of musical interpretation that are not captured by our representation. For example, the interpretation of *Danny Boy* clearly requires a climax at the highest note, as do a number of the musical examples. We currently do not represent such an event through our markup. It is possible that we could add a new category of stress corresponding to such a highpoint, though we suspect that the degree of emphasis is continuous, thus not well captured by a discrete alphabet of symbols.

Another occasional shortcoming is the failure to distinguish contrasting material, as in *O Come O Come Emanuel*. This melody has a Gregorian chant-like feel and should mostly be rendered with deliberate calmness. However, the short outburst corresponding to the word "Rejoice" takes on a more declarative affect. Our prosodically-oriented markup simply has no way to represent such a contrast of styles, though it is hinted at in the structural decomposition of ((3-3)-(3-3))+(2-2)+3.

There are, perhaps, some other general shortcomings of the interpretations, though we believe there is quite a bit that is "right" in them, especially considering the simplicity of our representation of interpretation. However, we hope readers will make independent judgments.

6. ESTIMATING THE INTERPRETATION

The essential goal of this work is to *algorithmically* generate expressive renderings of melody. Having formally represented our notion of musical interpretation, we can generate an expressive rendering by *estimating* this representation.

6.1 Estimating Phrase Structure

We estimate the structural decomposition of our melody by maximizing an objective function defined on the decomposition using dynamic programming. The approach begins by labeling each note subsequence containing two bar lines as a *terminal state*, and scoring the plausibility of each possible label for the subsequence (the score function will be discussed presently). We then proceed *inductively* to find the optimal labelings of progressively larger subsequences, ultimately terminating with a labeling for the entire melody.

Suppose we have have found the possible labelings of each note subsequence containing $m-1$ bar lines, and have computed the best-scoring derivation of each such labeled subsequence (the labels will be described below). We can find the optimal score of each label on each contiguous region containing m bar lines by piecing together various contiguous subsequences containing less than m bar lines. We allow three possible ways to do this, as follows

1. We can label a subsequence containing m bar lines as a *terminal* state, corresponding to a single grouping with no subdivisions. We label such a group of measures as m — the number of measures composing the group. The subsequence need not begin or end at a measure boundary.

2. If the number of measures, m, has a factor, f, in $\{2, 3, \ldots, 5\}$, we consider all partitions of the region into f contiguous regions each containing $k = n/f$ bar lines. For each such partition, we consider piecing together k identically labeled segments and labeling the result as $(k - k - \ldots - k)$. For instance, if we consider a region containing 8 bar lines and consider composing this region of two identically labeled contiguous regions, we could group regions labeled as either 4 or (2-2). Any such production would result in a region labeled as (4-4), denoting the binary split. We *cannot* combine two contiguous regions labeled as 4 and (2-2) to make a (4-4) region.

3. For the final production phase, which considers the complete collection of melody notes containing, say, M bar lines, we allow the previously-described productions as well as a *concatenation* operation. The concatenation pieces together any pair or triple of contiguous regions composing the complete melody. Such concatenations will be denoted as $A + B$ or $A + B + C$ where A, B, C are any possible labelings of the individual regions.

Each of these productions generates a score for the resulting labeling. When we use the terminal state label, we want the collection of measures to make sense as an isolated unit. Thus we will score such labels to reward relatively long final notes and chord changes at the following bar line.

When applying our factoring rule, we wish to group together note sequences that exhibit parallelism. The rhythmic parallelism between two note groups can be measured by the symmetric difference of the rhythms — the number of notes that do note "line up" when the bar lines are aligned. This measure rewards similar rhythmic structures and encourages groups to have the same pickup length. When more than two groups are considered, we can compute an average symmetric difference. We have used such average symmetric differences on rhythm, pitch, and chord to achieve an overall measure of parallelism. The score of a particular factor label will then be the sum of the individual labeled subsequence scores plus the score for overall parallelism.

The final production type is concatenation. Generally speaking, we wish to discourage such explanations, so we give a fixed penalty every time the concatenation operation is invoked. Thus the score for a label involving concatenation is the sum of the individual scores, plus a parallelism score between the concatenated sections, plus the concatenation penalty.

With this description in mind, it is simple to find the overall best scoring labeling. After computing and scoring all possible labelings of regions containing m bar lines, we retain only the best scoring parse for each particular label — this is the essential idea of dynamic programming. Finally, when we consider the entire collection of notes, we choose the best scoring of all labelings as our structure estimate.

At present we have simply hand-chosen the score function and make no claims for the optimality of this choice. Both the automatic training and evaluation of this method are the focus of ongoing work. As an example, our algorithm recognized *O Come O Come Emmanuel* as ((3-3)-(3-3))+7 with each segment containing a quarter note pickup, showing an ability to recognize interesting asymmetries. Appropriately, most often we recognized simple binary structures to our melodies.

6.2 Estimating the Prosodic Labeling

Our estimation of the unobserved sequence of prosodic labels, $x_1, \ldots, x_N$, depends on various observables, $y_1, \ldots, y_N$, where the feature vector $y_n = y_n^1, \ldots, y_n^J$ measures attributes of the musical score at the nth note. The features we consider are surface-level attributes of the musical score. While a great many possibilities were considered, we ultimately culled the set to the metric strength of the onset position, as well as the first and second differences of note length, in seconds, and MIDI pitch.

Our fundamental modeling assumption views the label sequence, x, as a Markov chain, given the data, y:

$$\begin{aligned} p(x|y) &= p(x_1|y_1) \prod_{n=2}^{N} p(x_n|x_{n-1}, y_n, y_{n-1}) \quad (3) \\ &= p(x_1|y_1) \prod_{n=2}^{N} p(x_n|x_{n-1}, z_n) \end{aligned}$$

where $z_n = (y_n, y_{n-1})$. The intuition behind this assumption is the observation (or opinion) that much of phrasing results from a cyclic alternation between forward moving notes, $\{l^{\rightarrow}, l^*\}$, stressed notes, $\{l^-, l^+, l^\times\}$, and optional receding notes $\{l^{\leftarrow}\}$. Often structural boundaries occur when one moves from either stressed or receding states to forward moving states. Thus the notion of *state*, as in a Markov chain, seems to be relevant.

We estimate the conditional distributions $p(x_n|x_{n-1}, z_n)$ for each choice of $x_{n-1} \in A$, as well as $p(x_1|y_1)$, using our labeled data. We will use the notation

$$p_l(x|z) \stackrel{\text{def}}{=} p(x_n = x|x_{n-1} = l, z_n = z)$$

for $l \in A$. In training these distributions we split our score data into $|A|$ groups, $D_l = \{(x_{li}, z_{li})\}$, where D_l is the collection of all (class label, feature vector) pairs over all notes that immediately follow a note of class l.

	l^*	$l^{\rightarrow}$	$l^{\leftarrow}$	l^-	$l^\times$	l^+	total
l^*	135	112	0	18	2	0	267
$l^{\rightarrow}$	62	1683	8	17	0	0	1770
$l^{\leftarrow}$	3	210	45	6	2	0	266
l^-	49	48	4	103	15	0	219
$l^\times$	5	32	2	65	30	0	134
l^+	0	3	0	12	3	0	18
total	254	2088	59	221	52	0	2674

Figure 4. Confusion matrix of errors over the various classes. The rows represent the true labels while the columns represent the predicted labels. The block structure indicated in the table shows the confusion on the coarser categories of stress, forward movement, and receding movement

We model the $p_l(x|z)$ distributions using the classification tree methodology of CART [13]. That is, for each D_l we begin with a "split," $z^j > c$ separating D_l into two sets: $D_l^0 = \{(x_{li}, z_{li}) : z_{li}^j > c\}$ and $D_l^1 = \{(x_{li}, z_{li}) : z_{li}^j \leq c\}$. We choose the feature, j, and cutoff, c, to achieve maximal "purity" in the sets D_l^0 and D_l^1 as measured by the average entropy over the class labels. We continue to split the sets D_l^0 and D_l^1, splitting their "offspring," etc., in a greedy manner, until the number of examples at a tree node is less than some minimum value. Our estimate $\hat{p}_l(x|z)$ is then computed by finding the terminal tree node associated with z and using the empirical label distribution over the class labels $\{x_{li}\}$ whose associated $\{z_{li}\}$ fall to the same terminal tree node.

Given a piece of music with feature vector $z_1, \ldots, z_N$,we can compute the optimizing labeling

$$\hat{x}_1 \ldots, \hat{x}_N = \arg \max_{x_1, \ldots, x_N} \hat{p}(x_1|y_1) \prod_{n=2}^{N} \hat{p}(x_n|x_{n-1}, z_n)$$

using dynamic programming.

7. RESULTS

We estimated a labeling for each of the $C = 48$ pieces in our corpus by training our model on the remaining $C - 1$ pieces and finding the most likely labeling, $\hat{x}_1, \ldots, \hat{x}_N$, as described above. When computing the most likely labeling for each melody in our corpus we found a total of 678/2674 errors (25.3%) with detailed results as presented in Figure 4.

The notion of "error" is somewhat ambiguous, however, since there really is no correct labeling. In particular, the choices among the forward-moving labels: $\{l^*, l^{\rightarrow}\}$, and stress labels: $\{l^-, l^\times, l^+\}$ are especially subject to interpretation. If we compute an error rate using these categories, as indicated in the table, the error rate is reduced to 15.3%.

One should note a mismatch between our evaluation metric of recognition errors with our estimation strategy. Using a forward-backward-like algorithm it is possible to compute $p(x_n|y_1,\ldots,y_N)$. Thus if we choose

$$\bar{x}_n = \arg\max_{x_n \in A} p(x_n|y_1,\ldots,y_N),$$

then the sequence $\bar{x}_1,\ldots,\bar{x}_N$ minimizes the expected number of estimation errors

$$E(\text{errors}|y_1,\ldots,y_N) = \sum_n p(x_n \neq \bar{x}_n|y_1,\ldots,y_N)$$

We have not chosen this latter metric because we want a *sequence* that behaves reasonably. It the sequential nature of the labeling that captures the prosodic interpretation, so the most likely sequence $\hat{x}_1,\ldots,\hat{x}_n$ seems like a more reasonable choice.

In an effort to measure what we believe to be *most* important — the perceived musicality of the performances — we performed a small user study. We took a subset of the most well-known melodies of the dataset and created audio files from the random, hand, and estimated annotations. The estimated annotations were produced using ground truth for the structure while estimating the prosodic labelings. We presented all three versions of each melody to a collection of 23 subjects who were students in our University's music school, as well as some other comparably educated listeners. The subjects were presented with random orderings of the three versions, with different orderings for each user, and asked to respond to the statement: "The performance sounds musical and expressive" with the Likert-style ratings 1=strongly disagree, 2=disagree, 3=neutral, 4=agree, 5=strongly agree, as well as to rank the three performances in terms of musicality (the ranking does not always follow from the Likert ratings). Out of a total of 244 triples that were evaluated in this way, the randomly-generated annotation received a mean score of 2.96 while the hand and estimated annotations received mean scores of 3.48 and 3.46. The rankings showed no preference for the hand annotations over the estimated annotations ($p = .64$), while both the hand and estimated annotations were clearly preferred to the random annotations ($p = .0002$, $p = .0003$).

Perhaps the most surprising aspect of these results is the high score of the random labelings — in spite of the meaningless nature of these labelings, the listeners were, in aggregate, "neutral" in judging the musicality of the examples. We believe the reason for this is that musical prosody, accounts for only a portion of what listeners respond to. All of our examples were rendered with human-supplied structural representations and the same sound engine of Section 4 which tries to create a sense of smoothness in the delivery with appropriate use of vibrato and timbral variation. We imagine that the listeners were partly swayed by these aspects, even when the use of prosody was not satisfactory. The results also show that our estimation produced annotations that were, essentially, as good as the hand-labeled annotations. This demonstrates a success of our research. The computer-generated interpretations clearly demonstrate some musicality with an average listener rating of 3.46 — halfway between "neutral" and "agree." However, there is considerable room for improvement.

The melodies were also rendered using structural representations estimated as in Section 6.2, thus leaving the entire musical interpretation to the computer. The audio files documenting this experiment are available on the aforementioned web site.

8. REFERENCES

[1] Goebl W., Dixon S., De Poli G., Friberg A., and Bresin R. and Gerhard Widmer. *Sense in expressive music performance: Data acquisition, computational studies, and models*, chapter 5, pages 195–242. Logos Verlag, Berlin, may 2008.

[2] Widmer G. and Goebl W. Computational models for expressive music performance: The state of the art. *Journal of New Music Research*, 33(3):203–216, 2004.

[3] Todd N. P. M. The kinematics of musical expression. *Journal of the Acoustical Society of America*, 97(3):1940–1949, 1995.

[4] Widmer G. and Tobudic A. Playing Mozart by analogy: Learning multi-level timing and dynamics strategies. *Journal of New Music Research*, 33(3):203–216, 2003.

[5] Hiraga R., Bresin R., Hirata K., and Katayose H. Rencon 2004: Turing Test for musical expression, Proceedings of the 2004 Conference on New Interfaces for Musical Expression (NIME04), 120–123, 2004.

[6] Hashida Y., Nakra T., Katayose H., and Murao Y. Rencon: Performance Rendering Contest for Automated Music Systems, Proceedings of the 10th Int. Conf. on Music Perception and Cognition (ICMPC 10), Sapporo, Japan, 53-57, 2008.

[7] Sundberg J. The KTH synthesis of singing. *Advances in Cognitive Psychology. Special issue on Music Performance*, 2(2-3):131–143, 2006.

[8] Friberg A., Bresin R. and Sundberg J. Overview of the KTH rule system for musical performance. *Advances in Cognitive Psychology*, 2(2-3):145–161, 2006.

[9] Roads C. *The Computer Music Tutorial*. MIT Press, 1996.

[10] Bod R. A Unified Model of Structural Organization in Language and Music. *Journal of Artificial Intelligence Research*, 17:289-308, 2002.

[11] Palmer C. Music Performance. *Annual Review Psychology*, 48:115-138, 1997.

[12] omitted for review Representation and Synthesis of Melodic Expression. *Proc. of Int. Joint Conf. on Art. Int. (IJCAI))*, to appear.

[13] Breiman L., Friedman J., Olshen R. and Stone C. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

USE OF HIDDEN MARKOV MODELS AND FACTORED LANGUAGE MODELS FOR AUTOMATIC CHORD RECOGNITION

Maksim Khadkevich
FBK-irst, Universitá degli studi di Trento,
Via Sommarive, 14 - Povo - 38050, Trento, Italy
khadkevich@fbk.eu

Maurizio Omologo
Fondazione Bruno Kessler-irst
Via Sommarive, 18 - Povo - 38050 Trento, Italy
omologo@fbk.eu

ABSTRACT

This paper focuses on automatic extraction of acoustic chord sequences from a musical piece. Standard and factored language models are analyzed in terms of applicability to the chord recognition task. Pitch class profile vectors that represent harmonic information are extracted from the given audio signal. The resulting chord sequence is obtained by running a Viterbi decoder on trained hidden Markov models and subsequent lattice rescoring, applying the language model weight. We performed several experiments using the proposed technique. Results obtained on 175 manually-labeled songs provided an increase in accuracy of about 2%.

1. INTRODUCTION

Among all existing musical styles, western tonal music, which is one of the most popular nowadays, is known for its strong relationship to harmony. Harmonic structure can be used for the purposes of content-based indexing and retrieval since it is correlated to the mood, style and genre of musical composition. Automatic analysis of digital music signals has attracted the attention of many researchers, establishing and evolving the Music Information Retrieval (MIR) community. One of the largest research areas of the interdisciplinary science of MIR is music transcription. A subtask of this problem, which deals with the extraction of harmonic properties of audio signal, is chord recognition. Basically, harmony denotes a combination of simultaneously or progressively sounding notes, forming chords and their progressions. In almost all cases the harmonic structure of a piece of music can be converted into a chord sequence. A great interest in chords can be indicated by a number of websites containing chord databases for existing popular songs. Automatic extraction of harmonic structure can also be of great use to musicologists, who perform harmonic analysis over large collections of audio data.

As in the case of speech recognition, one of the most critical issues in chord recognition is the choice of the acoustic feature set to use in order to represent the waveform in a compact way. One of the most successfully used feature set is chromagram, which can be represented as a sequence of chroma vectors. Each chroma vector, also called Pitch Class Profile (PCP), describes the harmonic content of a given frame. The amount of energy for each pitch class is described by one component in the PCP vector. Since a chord consists of a number of tones and can be uniquely determined by their positions, chroma vectors can be used effectively for chord representation. The chroma feature was firstly introduced for music computing tasks by Fujishima [1]. He proposed a real-time chord recognition system, describing extraction of 12-dimensional chroma vectors from the Discrete Fourier Transform (DFT) of the audio signal and introducing a numerical pattern matching method using built-in chord-type templates to determine the most likely root and chord type. The statistical learning method for chord recognition was suggested by Sheh and Ellis [2]. They exploited the Expectation-Maximization (EM) algorithm to train hidden Markov models, while chords were treated as hidden states. Statistical information about chord progressions in their approach is represented by the state transitions in HMM. The approach of Papadopoulos and Peeters [3] incorporates simultaneous estimation of chord progression and downbeats from an audio file. They paid a lot of attention to possible interaction of the metrical structure and the harmonic information of a piece of music.

Incorporating statistical information on chord progressions into a chord recognition system is an important issue. It has been addressed in several works through different techniques. Mauch and Dixon [4] used one of the simplest forms of N-grams – the bigram language model. In the approaches of Papadopoulos and Peeters, Lee and Slaney [3, 5] chord sequence modeling is introduced through state transition probabilities in HMM. In their case "language model" is a part of HMM and is derived from the Markov assumption, where chord probability is defined by only one predecessor. Yoshioka et al. [6] presented an automatic chord transcription system which is based on generating hypotheses about tuples of chord symbols and chord boundaries, and further evaluating the hypotheses, taking into account three criteria: acoustic features, chord progression patterns and bass sounds. This approach was further developed by Sumi et al. [7]. They mainly focused on the interrelationship among musical elements and made an attempt to efficiently integrate information about bass lines into chord recognition framework. They used two 2-gram models, one for major keys and one for minor keys, which are obtained in advance from real music. A large study on the modeling of chord sequences by probabilistic N-grams was performed by Scholz et al. [8]. Unal et al. [9] used perplexity-based scoring to test the likelihoods of possible transcription sequences.

This paper investigates the applicability of standard and factored language models of high orders (3-gram, 4-gram). Experiments with different back-off strategies for factored language models are carried out.

The rest of the paper is organized as follows: section 2 describes the front-end processing. In section 3 the here adopted HMM-based classification engine is briefly outlined. Language modeling is presented in section 4. Section 5 is devoted to the description of the whole proposed chord recognition system. The experimental results and conclusion are then given in section 6 and section 7, respectively.

2. FRONT-END PROCESSING

Before extracting features, the tuning procedure described in [10] is applied in order to find the mis-tuning rate and set the reference frequency f_{ref} for the ”A4” tone. The necessity of tuning appears when audio was recorded from instruments that were not properly tuned in terms of semitone scale.

The feature extraction process starts with downsampling the signal to 11025 Hz and converting it to the frequency domain by a DFT applying Hamming window of 185.7 ms with 50% overlapping. The harmonic content is extracted from the frequency range between 100 Hz and 2 kHz only. The main reason for this is the fact that in this range the energy of the harmonic frequencies is stronger than non-harmonic frequencies of the semitones. A sequence of conventional 12-dimensional Pitch Class Profile (PCP) vectors, known as chromagram is used as acoustic feature set. Each element of PCP vector corresponds to the energy of one of the 12 pitch classes. The process of PCP extraction can be decomposed into several steps. After applying DFT, the energy spectrum is mapped to the chroma domain, as shown in (1).

$$n(f_k) = 12\log_2\left(\frac{f_k}{f_{ref}}\right) + 69, n \in \Re^+ \quad (1)$$

where f_{ref} denotes the reference frequency of ”A4” tone, while f_k and n are the frequencies of Fourier transform and the semitone bin scale index, respectively. To reduce transients and noise we apply smoothing over time using median filtering, similarly to Peeters [11] and Mauch et al. [4]. At the last stage semitone bins are mapped to pitch classes, which results in the sequence of 12-dimensional PCP vectors:

$$c(n) = \mathrm{mod}(n, 12) \quad (2)$$

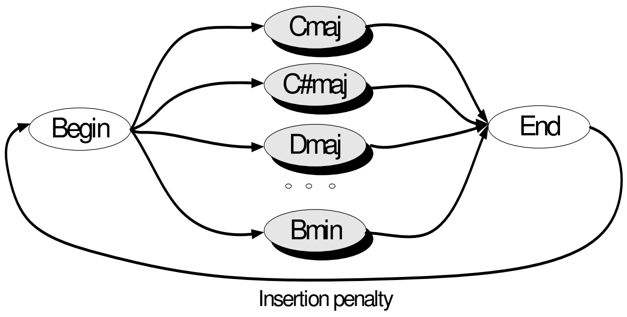


Figure 1. Connection scheme of trained models for decoding.

3. HIDDEN MARKOV MODELS

Hidden Markov models, which have been successfully used for modeling temporal sequences, are utilized in the proposed approach.

In contrast to many existing approaches [2, 3, 5], where chord is represented as a hidden state in one ergodic HMM, a separate left-to-right model is here created for each chord. In the given system configuration each model consists of 3 hidden states. The entry and exit states of a HMM are non-emitting, while the observation probabilities are identical for all emitting states. Observation vector probabilities in the emitting states can be approximated by a number of Gaussians in 12 dimensions, described by a mean vector and a covariance matrix. The feature vector components are assumed to be uncorrelated with one another, so the covariance matrix has a diagonal form. For each observation we use a mixture of 512 12-dimensional Gaussians. Songs from the training set are segmented according to the ground-truth labels so that each segment represents one chord. Chromagrams extracted from these segments are used for training, which is based on the application of the Baum-Welch algorithm.

Before running the recognition task, we extract a chromagram for each song from the test data. There is no preliminary segmentation as done on the training data for which a chroma vector sequence is extracted for each chord segment; only one chromagram is obtained for the whole test song. The trained chord HMMs are connected as shown in figure 1. Such parameter as insertion penalty is introduced, which allows for obtaining labels with different degrees of fragmentation. The Viterbi algorithm is then applied to the test data by using the resulting connected trained model in order to estimate the most likely chord sequence for each song and to produce a chord lattice.

4. LANGUAGE MODELING

A lot of different statistical language models have been proposed over years. The most successful among them appeared to be finite state transducers. In Natural Language processing N-grams are used for word prediction. Given $N-1$ predecessors, it can provide the probability of N-th element appearing. Language models have a variety of applications such as automatic speech recognition and statistical machine translation. The main goal of language modeling can be explained as follows: having a sentence, which consists of K words $(w_1, w_2, ...w_K)$, generate a probability model $p(w_1, w_2, ...w_K)$. In most common cases it can be expressed as (3).

$$p(w_1, w_2...w_K) = \prod_t p(w_t|w_1, w_2...w_{t-1}) = \prod_t p(w_t|h_t) \tag{3}$$

where h_t is the history sufficient for determining the probability of w_t word. In standard N-gram models the history consists of the immediately adjacent $N-1$ words. For example, in 3-gram model the probability of current word can be expressed as: $p(w_t|w_{t-1}, w_{t-2})$.

While estimating language model parameters, there exists the problem of sparse data. It is caused by the impossibility of producing maximum likelihood estimate of the model, because all combinations of N-word sequences are unlikely to be found in the training corpus. Since any training corpus is limited, some acceptable sequences can be missing from it, which leads to setting zero probability to plenty of N-grams. In order to cope with the problem, different techniques, such as back-off, smoothing and interpolation are used [12–14]. The main principle of back-off is to rely on lower-order model (e.g $p(w_t|w_{t-1})$) if there is zero evidence for higher-order (e.g. $p(w_t|w_{t-1}, w_{t-2})$) model. The order of dropping variables is known as back-off order. In the case of standard language models it is obvious that information taken from older predecessor will be less beneficial and it should be dropped prior to other predecessors.

In the proposed approach we draw direct analogy between a sentence in speech and a tune in a piece of music. The above-described strategy can be successfully used in chord sequences modeling. In this case a chord is the equivalent of a word and the sequence of chords can be modeled by means of the same technique.

4.1 Factored language models

Western music is known to be highly structural in terms of rhythm and harmony. In order to take advantage of mutual dependency between these two phenomena, we have studied the interrelationship between beat structure and chord durations. The number of occurrences as a function of chord duration in beats histogram is shown in figure 2. It is clearly seen that a greater part of chord durations is correlated to the metrical structure (2, 4, 8, 12, 16, 24, 32 beats), which suggests that including also chord durations in the language model is more convenient than analyzing just a sequence of chord symbols. This can be easily done with the help of factored language models (FLMs), which treat a word (chord) as a set of factors. FLMs have been recently proposed by Bilmes and Kirchoff [15] and showed promising results in modeling highly inflected languages, such as Arabic [16].

In a factored language model, a word (chord) can be represented as a bundle of factors: $w_t = \{f_t^1, f_t^2, ..., f_t^K\}$. The probability for FLM is given in (4), where $\pi(f_t^k)$ is a set of variables (parents), which influence the probability of f_t^k. In our case to model chord sequences we use two factors: chord label C_t and chord duration D_t: $w_t = \{C_t, D_t\}$.

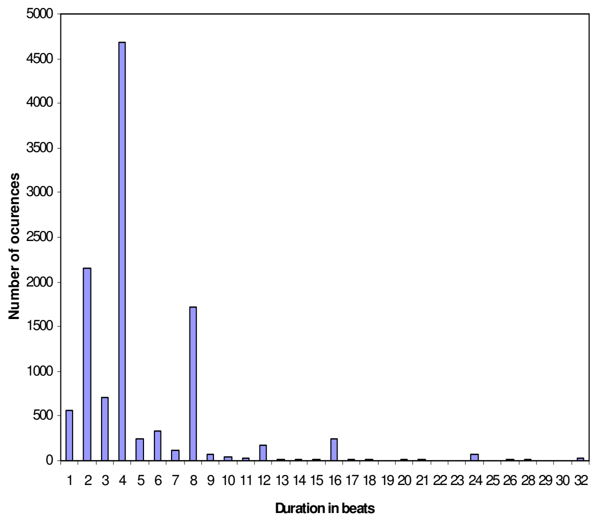


Figure 2. Chord Duration Histogram.

$$p(w_t|h_t) = \prod_k p(f_t^k|\pi(f_t^k)) \tag{4}$$

As opposed to standard language models, where older predecessors give less relevant information at the given time instant, in FLMs there is no obvious order to drop parents $\pi(f_t^k)$. There are a lot of possibilities to choose less informative factors to drop among the others. Moreover, keeping some factors of older predecessors can be of greater benefit than keeping the value of some other factors, which are more relevant to the given time instant. One of the possible solutions is to use ”generalized parallel back-off”, which was initially proposed and well described by Bilmes and Kirchoff [15]. The main idea is to back-off factors simultaneously. The given set of back-off paths is determined dynamically based on the current values of the variables. (For a more detailed description, see [15]).

At the experimental stage we explore the standard back-off (a) and the parallel back-off (b) techniques, whose graphs are presented in figure 3. In both cases the chronological order is kept, while in the standard back-off case a higher priority to the factor of chord symbol is assigned. The arrows are marked with the factor being dropped at the current back-off step; blocks include the variables that influence the probability of chord label being estimated.

5. CHORD RECOGNITION SYSTEM

The full scheme of chord recognition system is depicted in figure 4.

Feature extraction part has been described in section 2. The beat extraction algorithm used here is introduced by Dixon [17] and is exploited as a separate module, called

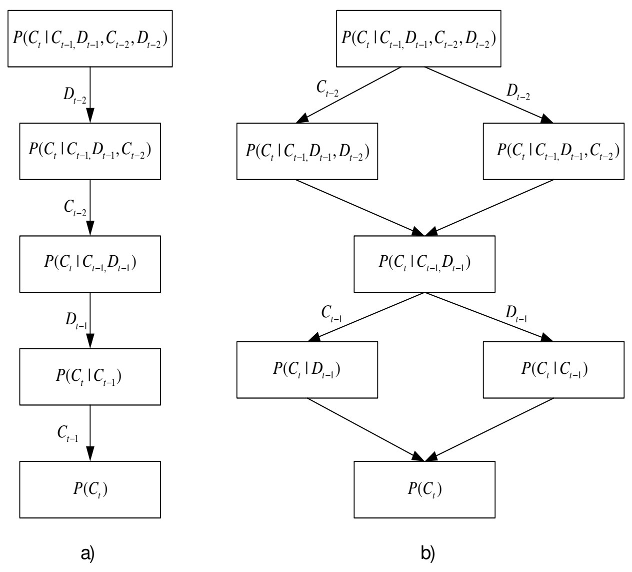

$P(C_t \mid C_{t-1}, D_{t-1}, C_{t-2}, D_{t-2})$
D_{t-2}
$P(C_t \mid C_{t-1}, D_{t-1}, C_{t-2})$
C_{t-2}
$P(C_t \mid C_{t-1}, D_{t-1})$
D_{t-1}
$P(C_t \mid C_{t-1})$
C_{t-1}
$P(C_t)$
a)
$P(C_t \mid C_{t-1}, D_{t-1}, C_{t-2}, D_{t-2})$
C_{t-2}
D_{t-2}
$P(C_t \mid C_{t-1}, D_{t-1}, D_{t-2})$
$P(C_t \mid C_{t-1}, D_{t-1}, C_{t-2})$
$P(C_t \mid C_{t-1}, D_{t-1})$
C_{t-1}
D_{t-1}
$P(C_t \mid D_{t-1})$
$P(C_t \mid C_{t-1})$
$P(C_t)$
b)

Figure 3. Standard back-off (a) and parallel back-off (b) graphs for tri-gram LM.

BeatRoot [1] .

The key detection module utilizes the approach suggested by Peeters [11], where trained HMMs are used to find the best score from 24 possible keys for the given sequence of chroma vectors for each test song. In the suggested system the key is assumed to be constant.

On the training stage, features extracted from waveforms are used to train hidden Markov models, while chord labels from training corpus are used as an input for language model parameter estimation. Language model training includes training either standard LMs or FLMs. For training standard LMs chord sequences taken from the training labels are used as input. For building text for FLM the information combined from beat extraction module and the training labels is used. For each chord symbol from ground-truth labels we estimate the duration in beats and produce an output in the form: ”C-(chord type):D-(duration)”. To minimize the problem of sparse data, all duration values are quantized by a relatively-small set of or integer values. Our codebook consists of the following values: 1, 2, 3, 4, 6, 8, 12, 16, 24 and 32 beats. The suggested codebook is supposed to be well-suited for the pop songs. This assumption is made on the basis of metrical analysis of the Beatles data (see fig. 2). The suggested scheme however might not be sufficient while modeling jazz or other genres.

In order to make our system key invariant, a key transformation technique is proposed here. In fact, the training corpus might not contain some type of chords and chord transitions due to the fact that keys with a lot of accidentals are much less widespread (G# maj, Ab min). Moreover, while estimating chord transition probabilities the relative change in the context of the given key (e.g. tonic – dominant – subdominant) is more relevant than exact chord names. For training data we have ground-truth table of keys for each song, while for test data we estimate key in the key detection module. Then, similar to training HMMs, by applying circular permutation, features and labels are converted to the Cmaj (in case of major key) or to Amin (in case of minor key). After the decoding procedure in order to produce final labels (in the original key of the analyzed song) obtained labels are converted back using the same scheme.

Similar to the approach of multiple-pass decoding, which has been successfully used in speech recognition [14], the decoding procedure consists of two steps. During the first step time-and-space efficient bigram language model is applied on the stage of Viterbi decoding, producing a lattice. A lattice can be represented by a directed graph, where nodes denote time instants and arcs are different hypotheses. Since lattices contain the information on the time boundaries, it is possible to make an estimation of duration in beats for each hypothesis. During the second step the obtained lattice is rescored applying more sophisticated language models (trigram and higher) on the reduced search space. Since the main problem is to extract chord labels, it is not necessary to model chord duration probabilities explicitly. Our decoding scheme, applying language modeling, is based on Viterbi decoding and subsequent lattice rescoring, where lattices contain the information on possible chord boundaries. Chord durations are used only to define chord label probabilities and the resulting chord boundaries are obtained from the lattices. Generally, standard LMs do not take into account duration factor at all, the only important thing here is just a sequence of labels. The advantage of FLM is that when applying the language model weight on the stage of lattice rescoring, chord durations contribute to the probabilities of different hypotheses in the lattice.

Standard LMs are manipulated using HTK [2] tools, while FLMs are managed using SRILM [18] toolkit, since HTK does not support this type of language models.

6. EXPERIMENTS

Evaluation of the proposed system was performed on the songs taken from 12 Beatles albums, ground-truth annotations for which were kindly provided by C. A. Harte [19]. The system can distinguish 24 different chord types (major and minor for each of 12 roots). 7th, min7, maj7, minmaj7, min6, maj6, 9, maj9, min9 chords are merged to their root triads; suspended augmented and diminished chords are discarded from the evaluation task. The percentage of duration of discarded chords results to be 2.71% of the whole material. In order to prevent the lack of training data (some chord types can appear only few times in the training corpus) only two models are trained: C-major and C-minor. For this purpose, all chroma vectors obtained from labeled segments are mapped to the C-root using circular permutation. After that mean vectors and covariance matrices are estimated for the two models. All the other models can be obtained by a circular permutation procedure.

[1] http://www.elec.qmul.ac.uk/people/simond/beatroot/index.html

[2] http://htk.eng.cam.ac.uk/

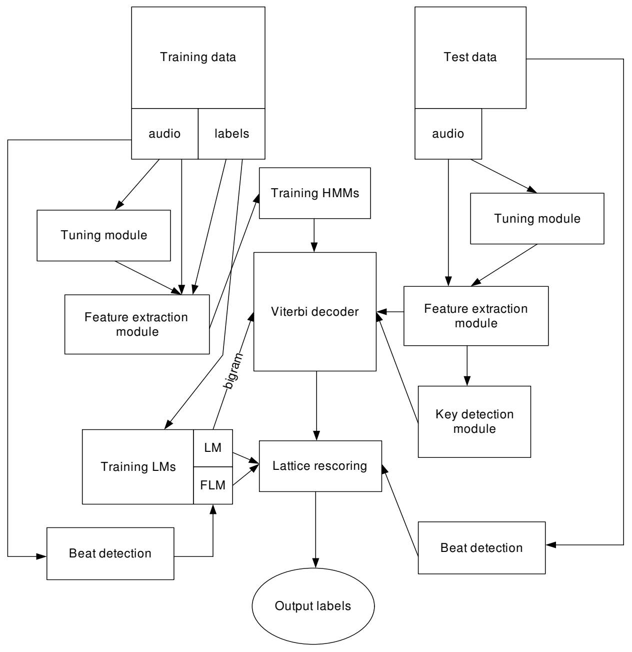


Figure 4. Chord recognition system.

For evaluation, the recognition rate measure was used, which in the given case corresponds to the total duration of correctly classified chords divided by the total duration of chords, as reported in the following:

$$rec.rate = \frac{|recognized\_chords| \cap |ground - truth\_chords|}{|ground - truth\_chords|} \quad (5)$$

The evaluation was performed frame by frame, as it was done under the MIREX[3] competition. In our experiments 3-gram and 4-gram language models were used. While working with FLMs, we exploited standard and generalized parallel back-off strategies (see figure 3; 4-gram graphs have the same structure and can be obtained from 3-gram graphs by adding one level).

It is worth mentioning that applying different language model weights on the stage of lattice rescoring one can obtain different recognition rates. Figure 5 indicates how recognition rate depends on the LM weight. In this case the curves correspond to the LM- and FLM-based systems; experiments were conducted on the fold 1 with 4-gram configuration.

In order to estimate the increase in performance introduced by including LM block and in order to compare efficiency of standard and factored language models, a 5-fold cross-validation was accomplished on the given data set. The folds were built in a random way and there is high album overlap. The recognition rates are shown in Table 1. Here ”bl” is baseline system, ”3lm” ”3flm” ”3flmgpb” are trigram configurations with key transformation for standard LM, FLM, and FLM with generalized parallel back-off respectively, ”4lm” ”4flm” ”4flmgpb” are 4-gram configurations. For any of the given configurations, an average standard deviation of about 15% was also observed,

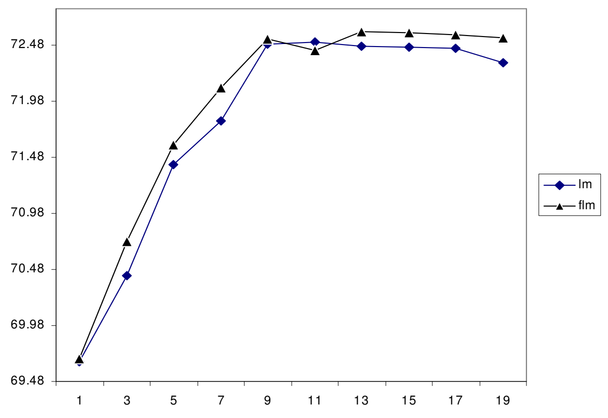


Figure 5. Recognition rate as a function of LM weight.

which was derived from the recognition rates computed on a song-by-song basis.

Experimental results showed that introducing language modeling increases the performance of the system, while generalized parallel back-off strategy for FLM did not show any advantages over standard back-off for the chord recognition task. Meanwhile, using FLM show very slight improvement (0.25 %) in comparison to standard LM. The differences in the output labels for LMs and FLMs are mainly on the junctions of chords. While using standard LM one can get a slight boundary deviation from its ground-truth value (e.g. 1 beat), using FLM fixes this in most cases because it takes into account the duration factor. That is why the difference in recognition rates is so small.

7. CONCLUSION

In this paper a set of experiments on chord recognition task including language modeling functionality as a separate layer has been conducted. The experimental results in a 5-fold cross-validation were conducted on a commonly used database of the songs by the Beatles. Factored language models were compared with standard language models and showed small increase in performance for the task. The main advantage of FLMs is that they possess a better chord recognition ability on the chord junctions. Comparing back-off techniques, we can assume that using generalized parallel back-off for the chord recognition task does not result in better performance.

However, the suggested system has a number of limitations: assuming the key of the song constant, one can not cope with key changes. A deeper study on different model smoothing and selection techniques as those addressed by Scholz et al. [8] could be reprised.

In general, experimental results showed that utilizing language models leads to an increase in accuracy by about 2%. This relatively small difference in performance may be due to the size of vocabulary for the chord recognition task in comparison with that of many speech recognition applications. The performance of chord recognition sys-

[3] http://www.music-ir.org/mirex/2008/index.php/Main_Page

data	bl	3lm	3flm	3flmgpb	4lm	4flm	4flmgpb
fold 1	70.81	72.22	72.55	72.56	72.39	72.53	72.27
fold 2	70.23	70.78	71.15	71.51	71.09	71.38	71.25
fold 3	65.87	66.81	66.59	67.01	67.22	66.89	67.17
fold 4	66.20	67.15	67.60	67.61	67.64	67.62	67.51
fold 5	66.19	69.73	69.72	68.55	68.55	69.72	69.77
average	67.86	69.34	69.52	69.45	69.38	69.63	69.59

Table 1. Evaluation results: recognition rates.

tems is perhaps influenced primarily by relevance and accuracy of the extracted features and related acoustic modeling.

8. REFERENCES

[1] Takuya Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the International Computer Music Conference*, Beijing, 1999.

[2] A. Sheh and D. P. Ellis. Chord segmentation and recognition using em-trained hidden markov models. In *Proc. 4th International Conference on Music Information Retrieval*, 2003.

[3] H. Papadopoulos and G. Peeters. Simultaneous estimation of chord progression and downbeats from an audio file. In *Proc. ICASSP*, 2008.

[4] Matthias Mauch and Simon Dixon. A discrete mixture model for chord labelling. In *Proceedings of the 2008 ISMIR Conference*, Philadelphia, 2008.

[5] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2), february 2008.

[6] T. Yoshioka, T.Kitahara, K. Komatani, T. Ogata, and H.G. Okuno. Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, 2004.

[7] K. Sumi, K. Itoyama, K. Yoshii, K. Komatani, T. Ogata, and H. G. Okuno. Automatic chord recognition based on probabilistic integration of chord transition and bass pitch estimation. In *Proceedings of the 2008 ISMIR Conference*, Philadelphia, 2008.

[8] R. Scholz, E. Vincent, and F. Bimbot. Robust modeling of musical chord sequences using probabilistic n-grams. In *Proc. ICASSP*, 2009.

[9] E. Unal, P. Georgiou, S. Narayanan, and E. Chew. Statistical modeling and retrieval of polyphonic music. In *Proc. IEEE MMSP*, 2007.

[10] M. Khadkevich and M. Omologo. Phase-change based tuning for automatic chord recognition. In *Proceedings of DAFX*, Como, Italy, 2009.

[11] G. Peeters. Chroma-based estimation of musical key from audio-signal analysis. In *Proceedings of the 2006 ISMIR Conference*, Victoria, Canada, 2006.

[12] J. Goodman. A bit of progress in language modeling. In *Computer, Speech and Language*, 2001.

[13] F. Jelinek. Statistical methods for speech recognition. In *MIT Press*, 1997.

[14] D. Jurafsky and J. H. Martin, editors. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2000.

[15] J. Bilmes and K. Kirchoff. Factored language models and generalized parallel backoff. In *HLT-NAACL*, 2003.

[16] K. Kirchhoff, D. Vergyri, K. Duh, J. Bilmes, and A. Stolcke. Morphology-based language modeling for arabic speech recognition. In *Computer, Speech and Language*, 2006.

[17] S. Dixon. Onset detection revisited. In *Proceedings of DAFX*, McGill, Montreal, Canada, 2006.

[18] A. Stolcke. Srilm. an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, 2002.

[19] C. Harte and M. Sandler. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the 2005 ISMIR Conference*, 2005.

AUDITORY SPECTRAL SUMMARISATION FOR AUDIO SIGNALS WITH MUSICAL APPLICATIONS

Sam Ferguson
Faculty of Design, Architecture and Building
University of Technology, Sydney
samuel.ferguson@uts.edu.au

Densil Cabrera
Faculty of Architecture, Design and Planning
The University of Sydney
d.cabrera@arch.usyd.edu.au

ABSTRACT

Methods for spectral analysis of audio signals and their graphical display are widespread. However, assessing music and audio in the visual domain involves a number of challenges in the translation between auditory images into mental or symbolically represented concepts. This paper presents a spectral analysis method that exists entirely in the auditory domain, and results in an auditory presentation of a spectrum. It aims to strip a segment of audio signal of its temporal content, resulting in a quasi-stationary signal that possesses a similar spectrum to the original signal. The method is extended and applied for the purpose of music summarisation.

1. INTRODUCTION

Graphical display is the predominant approach to conveying musical sound analysis information to people, including via spectrograms, spectra, waveform graphics and musical manuscript. While the visual system is dominant in many information transfer contexts, sonification (the representation of information through non-speech sound) offers many (often complementary) possibilities for information transfer [1]. As audio and music are data that are experienced primarily in the auditory domain, sonification would appear to be an appropriate method for analysis and representation of audio data, as it sidesteps the translation process from the auditory domain to the visual domain that is inherent in using visual representations.

A variety of simple techniques for sonification of sound in the context of audio education have been proposed by Cabrera and Ferguson [2, 3], and Ferguson has developed techniques for techniques for statistical sonifications of audio in his Ph.D thesis [4]. These sonification methods provide auditory analogues to common statistical visual displays (such as cumulative distribution functions and box plots), but with much richer information than visual charts. One of the solutions proposed in the thesis is a method of displaying spectral data, which is the focus of this paper.

Spectral analysis is one of the most fundamental, powerful, and widely used methods for the investigation of audio. This paper discusses an approach to spectral display that does not use Fourier analysis, and exists completely within the auditory domain. Instead of a Fourier or related transform implemented through signal processing, the method uses the spectral analysis of the human auditory system. While almost all listening could be thought of as involving auditory spectral processing, in listening that is focused on spectral features, temporal features are distractions that should be removed. Such features include rhythm, prosody, language, and more generally, the time structure of the sound being analysed. Put simply, the technique blurs temporally fluctuating audio signals to create quasi-stationary signals with almost identical spectra envelopes to the original signals, but without any semblance to the original time-dependent fluctuation. This technique is rooted in the theories of Gabor [5, 6] and granular synthesis [7], and has been strongly influenced by the recent advances in concatenative synthesis by Schwarz [8, 9].

Information visualisation literature has focused on methods for presenting data in ways that present large overviews of data, but allow a user to 'zoom and filter' the representation to find information that is important [10]. Fry's *Computational Information Design* outlines a method for developing interactive information representation systems [11]. A sonification method that would improve on visual methods may; use the original audio as the sound material for the analytic representation; filter the content of that audio in some way; maintain context and meaning of the audio; and draw relationships and present pertinent contrasts.

Schenkerian analysis of musical works is well-known and features in many undergraduate music curricula [12]. This graphical analysis method based on musical manuscripts allowed Schenker to reveal the various layers of a composition. The spectral sonification method described in this paper has the potential to be used in a similar manner allowing a scaling of perspective from large to small scale structures depending on the periods analysed.

2. SPECTRAL SUMMARISATION ALGORITHM

The core technique this paper presents is a method for spectrally summarising a larger audio signal. A representation that can convey the spectrum using audio without frequency domain signal processing can be built using a simple digital algorithm based on fragmenting the time signal into potentially overlapping windows, and recombining them in a way that (i) maintains a roughly constant power spectral envelope that matches the signal's long term spectrum; and (ii) removes distracting (non-spectral) features. This can be achieved by concatenating short windows (or grains) of audio by averaging a large proportion, but not all, of the original signal windows. A number of unique but spectrally similar windows need to be concatenated together, since if a single audio grain is repeated the resulting sound will be dominated by amplitude modulation related to the repetition rate. A systematic explanation of a process to create and concatenate unique but spectrally similar grains is as follows:

1. For a user-selected window length w_n samples, randomly select a window length w_{nr} from the range of values between $w_n - \frac{w_n}{2}$ and $w_n + \frac{w_n}{2}$.
2. Randomly select windows of length w_{nr} from the signal to be averaged to create a set $\{w_1, w_2, ...w_m\}$ of m unique windows.
3. Sum this set of windows, and divide by the square root of the number of windows ($\frac{\{w_1+w_2,...+w_m\}}{\sqrt{m}}$) to produce a single frame of w_{nr} samples duration.
4. Repeat steps 1, 2 and 3 (re-randomising each time) until enough unique but spectrally similar audio frames are produced to build a stationary sound of a chosen duration.
5. Concatenate these audio frames, using overlapping and adding with a custom window function. Ramps taken from either side of a Hanning window function are applied only to the overlapping proportion (typically only 5-10% of w_n) to maintain a constant sum between concatenation boundaries (see Figure 3).

This method is simple, but it is successful at creating a quasi-stationary sound with a spectral profile that matches the original file, while keeping the time variance to a minimum.

2.1 Validation, Tradeoffs and Limitations

To validate the appropriateness of the averaging process we undertook a comparison of spectra created by this spectral averaging method against the spectrum of the unmodified sample. A distinction worth mentioning is that through mixing we are amplitude (pressure) averaging, rather than power (pressure squared) averaging. Summing a large number of randomly selected signals as described above may be considered to be an operation on incoherent signals, which is why the square root of the number of windows is used in the denominator of the algorithms third step. Hence, the power spectrum of the sonification approximates the power spectrum of the original wave. While there is some potential for a substantial discrepancy between the power of the resulting spectrum and a true long term power spectrum, tests have shown that discrepancies are not severe for realistic signals if the averaging method uses a window size larger than 1024 samples.

There is a significant smearing of energy when using

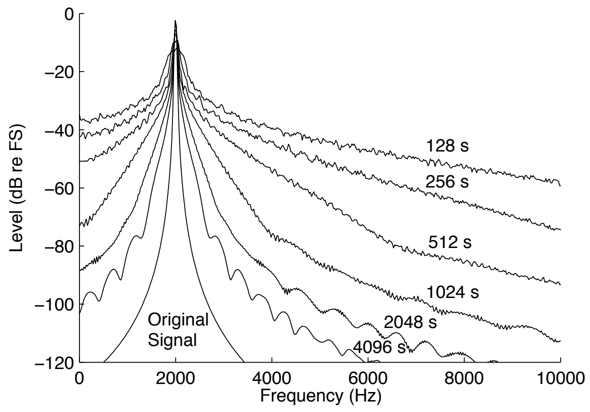


Figure 1. A 2kHz sine wave is spectrally averaged using a variety of window sizes, and compared with the original sine wave signal. The larger window sizes result in less spectral smearing.

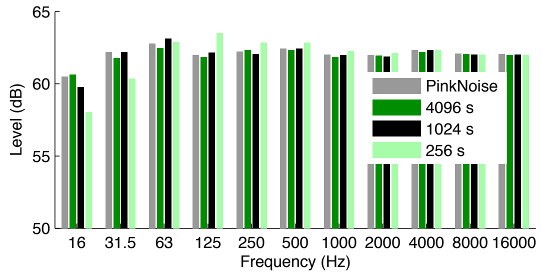


Figure 2. A pink noise recording is compared against spectral representations of pink noise using various window lengths - longer window length result in less spectral deviation.

shorter window lengths. We demonstrated this by comparing a spectrally averaged sine tone at 2kHz, using a range of window lengths, to the original signal. Figure 1 demonstrates this effect. Generally, window lengths of 1024 samples or greater decrease smearing and increase spectral representation quality significantly. Subjectively, short window lengths tend towards extremely noise-like signals bearing little resemblance to the tonal spectra expected.

The window length used in the spectral summarisation algorithm has a small effect on the low frequency range of the spectrum reproduced. To investigate this we compared a spectrum of a sample of pink noise (with a 48000 Hz sample rate) against three spectral summarisations, one using a 4096 point window, one with a 1024 point window, and one with a 256 point window. The length of the window determines the frequency below which the spectral representation begins to attenuate – at 4096 points there is little effect, but at 256 points it starts to become more significant and further reductions in window length result in the low cutoff frequency increasing proportionally. The cutoff frequency (fc) is apparently based on the largest wave period that can be represented by a specific window length (w_n), summarised by the relationship:

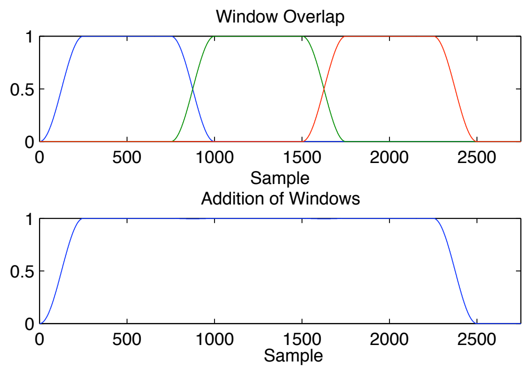


Figure 3. A custom window is designed to maintain a constant power sum between concatenated windows of audio.

$$f_c = \frac{f_s}{w_n} \tag{1}$$

This investigation tends to support the use of window sizes of at least 1024 samples and upwards for this spectral averaging technique. Larger window sizes will tend to allow more temporal fluctuation, depending on the temporal fluctuation present in the original signal, so there is a tradeoff present, however window sizes smaller than 1024 samples seem likely to significantly alter the spectrum to a degree where it is unrecognisable and non-representative. These parameters are likely to be experienced interactively, and therefore there is probably a subjective element to a user's selection for the most appropriate window size.

The windowing and overlapping at the concatenation stage must not introduce either discontinuities (clicks) or amplitude modulation, and thus we have designed a custom window shape that incorporates a large plateau (similar to a Tukey window) as well as a *Hanning* window function's ramps at either end. The proportion of the window devoted to the ramp is determined by the number of overlapping samples, and the resulting window shape maintains a constant sum at the window boundaries (see Figure 3). Furthermore, the randomisation around a central window length, ameliorates amplitude modulation effects that may arise out of periodic selection window length.

3. HARMONY ANALYSIS APPLICATIONS

Harmony analysis typically requires a familiarity with reading musical manuscripts, a difficult skill that is analogous to learning a new language. This, of course, places an immediate barrier to those users without these skills, but it also presumes a level of expertise in cross-modal perception in those users who possess skills in this language. A user who is presented with a harmony analysis on a manuscript is expected to ascertain the auditory meaning of the symbols and their consequences within the musical structure. This is not necessarily straightforward, and many users will 'interact' with the manuscript by using a piano to play back the pitches and compare their significance, while other users develop skills in producing auditory images of the various pitches. Methods that bridge the gap between symbolic representations of pitch relationships and auditory pitches are possible alternative solutions for these issues. Generally, the idea of this exploration is to make the patterns within music clearer than they are in a typical musical recording, so that users may understand harmonic patterns at multiple structural levels, and in intuitive manners.

The problem of producing a sonification of the harmony within the audio recording is therefore one of filtering the audio recording to contain less information, with an emphasis on that information which would be included in a harmonic representation. Such information may include musical elements like the fundamental frequency of the bass notes, and other notes presented either loudly or for comparatively long periods, while avoiding short decorative notes, or quick scalar passages. It would seek to remove, generally speaking, the temporal presentation of the notes, as well as their amplitude envelopes, resulting in a stationary sound with each important pitch presented simultaneously to build a chordal sound, accentuating the harmonic contribution each note makes, and diminishing each note's individual quality.

A structural representation would also need to describe how each section of the music relates to each other. The form of the piece is a crucial element in musicology, but it can be difficult to understand music at the formal scale from reading a musical manuscript, or from listening to an audio recording. Snyder [13] describes three levels of musical memory: the early processing level – which deals with characteristics of single notes, the short-term memory level – which holds musical phrases and rhythmic pattern, and the long-term memory level – which deals with formal sections, movements or entire pieces. Snyder also describes how long-term or formal memory deals with sections of music that are too long to be understood in the present, and their order needs to be consciously reconstructed as they do not automatically retain their time-order.

Simplifications or shorter versions of the musical sample can be used to describe the form of the piece in an amount of time that can be held within short-term memory. By presenting an auditory representation that is shorter than the original audio recording, but is proportional to the original, form can be more appropriately presented. Elements like key changes and voice ledaing become more obvious, as the ear can compare the short term memory of the old key to the new.

3.1 Voice Leading and Chord Patterns

The algorithm for building an averaged chord progression is as follows:

1. Get time information to use for time value boundaries – this may be based on extracted symbolic musical information, rhythmic information, timeseries descriptor peaks or various other time markup methods.
2. Find the first time boundary and the second time boundary to be averaged across and find the audio

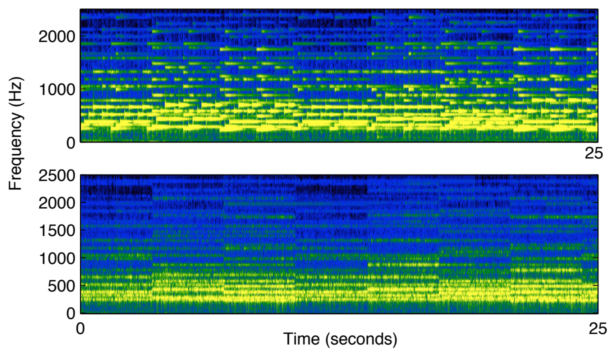


Figure 4. Comparing the sonification (top) and the original audio (bottom) we can see that the sonification attempts to blur the spectral components from each bar into a stationary sound. This sound is changed at every bar-line, approximately once every 3 seconds. The graphic only demonstrates the first 25 seconds of the piece.

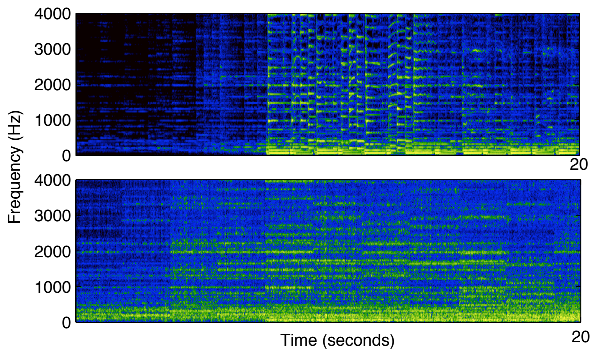


Figure 5. A second chord sonification example that demonstrates the structure of a Beatles song, *Norwegian Wood*. The upper graph is a spectrogram of the original audio, and the lower is the spectrally averaged sonification. The graphic shows only the first 20 seconds.

data in between the two times.

3. Apply the averaging method to this time interval to create the same duration of averaged audio, or perhaps a duration altered by a constant factor.
4. Place the resulting audio data at the corresponding sample numbers of the output audio, using appropriate overlapping.
5. Repeat the process after stepping forward to the second and third boundary, and continuing to step forward until the entire recording has been averaged and the output sonification built.

A simple method for finding time boundaries with which to segment the chordal structure is to extract the beats and assume that chord changes will be synchronised with beats, or more likely with bars. Depending on the meter of the piece (3/4, 4/4 or 6/8 commonly) we use particular beats as time boundaries, and in the following examples we have manually set the meter based on listening to the music, but advanced beat tracking algorithms may correctly estimate it as well. We also need to set an *anacrusis* value, that describes whether the piece begins on the first beat of the bar. Beat tracking is well-researched, and we use Dixon's *Beatroot* algorithm [14].

3.2 Harmonic Pattern Examples

We will attempt to use this algorithm to represent the long-term structure of some pieces of music.

One piece that is defined primarily by its chordal content (as opposed to its melodic or rhythmic) is Bach's Prelude No 1. from 'the Well-tempered Clavier'. A Schenkerian analysis has also been published for this piece [12]. By applying the algorithm to the audio we produce a sonification that is presented in Figure 4. The sonification created is not a completely stationary sound, like a set of tones, nor is it a sound that has discernible starting or ending notes. It demonstrates characteristics of the timbre of the instrument, but primarily it presents the notes that have sounded. The quality of the sound is similar to the sound that would be produced if the pianist stopped at the end of the bar and held down all the keys played in the bar.

The averaging across the bar is particularly appropriate for this particular piece, due to the manner in which Bach presents a single chord per bar. For other situations this may result in chords blurring into other chords, resulting in strong dissonance. Despite this, there are a large majority of pieces where this simple scheme would be sufficient. The remainder may be dealt with using more sophisticated methods, that employ harmonic and rhythm based pre-processing to carefully avoid averaging across chord changes incorrectly.

One purpose of the blurring of the audio is to be able to place one bar's harmonic content temporally adjacent to the next's. This should allow each harmonic change to be understood in terms of the notes within each chord, and to which notes they each move. An example of a pattern that might be uncovered through this process is the bassline in this prelude. While these notes are strongly sounded at the beginning of the bar, they decay by the end of the bar, and other higher notes are dominant by this stage. The blurring applied places each of these sounds adjacent to each other, yielding a *legato* bassline.

The other useful process possible by using the blurring of the audio is that the speed of the example can be arbitrarily altered. The blurred audio has no temporal content, so a bar's worth of sound may be presented over 3 seconds or in half a second. By setting an arbitrary compression factor for the duration, we can proportionally change the duration of the piece while maintaining the formal structure. This can be used to alter a 3 minute piece, whose structure can be 'remembered', into a 20 second piece, whose structure can be 'heard'.

In the structural sonification of *Norwegian Wood* (Figure 5), we hear a clear structure of descending melodic notes that define the chord structure. The structure is a lot simpler than that of the Prelude, and each formal section can be clearly heard.

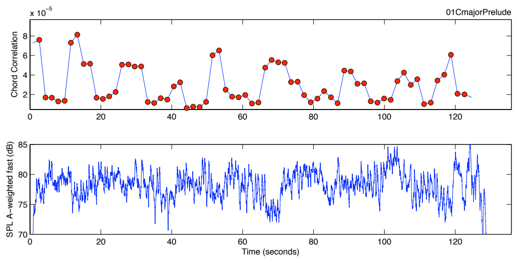


Figure 6. There is a 15dB SPL range over the time period of the musical example. High levels can be heard to correspond to the chords which have the least correlation and are the most dissonant.

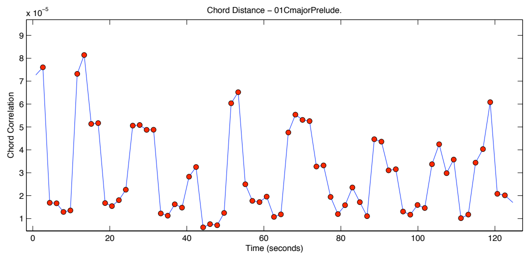


Figure 7. The correlation between various chords seems to follow a predictable pattern. The gain function sonification makes that pattern more apparent by exaggerating it, and highlighting low correlation values. Low values of correlation are analogous to high values of chord distance.

3.3 Chordal Patterns and Context

While we wish to maintain the temporal order of the various chords, due to their importance to the overall direction and purpose of a piece of music, it may be interesting to annotate the sonification in terms of the values of other parameters. A simple parameter to investigate is the sound pressure level (SPL). From listening to the sonification we can hear that often tonic chords are quieter compared with the chords that lead into them. A comparison of SPL against chord estimates shows the decrease in SPL that accompanies every return to the tonic (see Figure 6). We may wish to accentuate this further. By normalising the level of audio from of each section, and then mapping the SPL extracted from the audio to an expanded gain function that is then re-applied to each section, we can experience the structural implications of the performer's use of dynamics more clearly.

While this is a straightforward example, the use of sound pressure level as the mapping target is arbitrary, and many other such targets exist. Another candidate parameter to base a gain function sonification on is the harmonic distance the chord is from the tonic. One can attempt to approach this from a chord recognition perspective, but in this case we will use the chroma pattern only and will compare it against the first (tonic) chord.

The virtual pitch algorithm of Terhardt takes a template matching approach to finding pitches in the sound [15]. It applies a peak picking algorithm to find the points in the spectrum that are peaks. These are then applied to a successive template matching algorithm that attempts to place the peaks under a pitch template. The pattern of pitches across the audible range can be constrained to create a chroma pattern, representing the strength of each of the 12 pitches within an octave, regardless of pitch-height.

Using this method we calculate an average chroma pattern for each bar, and then multiply and sum those chroma pattern vectors to create a number representing the correlation between each bar and the first chord. A high number represents a high number of similar notes, or low chord distance, while a low number represents a large amount of difference. These values are clearly a useful target for mapping to a gain function. With larger chord distances associated to greater SPL, and smaller chord distances associated to low SPL, the effect should be similar to typical musical approaches.

An alternative approach to expanding parameters such as sound pressure level is to use a parameter as the basis of temporallycutting and reorganising the harmonic units or bars. Any measurable parameter that can be derived from a steady-state spectrum could be used for this purpose. The time periods (in this case bars) that are used to average from in the algorithm described in section 3.1 are then associated with a median parameter value taken for their time period. The median parameter value determines the reordering, and then the audio units are rearranged based on the new order. If the chord correlation values mentioned above are used to order the bars in ascending order, then the sonification progresses from chords that are generally dissimilar to the tonic, to chords that are more consonant – giving an overall impression of a long cadence. During this process particular chords can be assessed within the overall scheme of chords.

3.4 Beyond Chords

Analysis of sections of music larger than individual chords or bars is easily implemented using this method. The long term spectra of entire pieces can be sonified into a short sound, so that, for example, the average spectra of each of Bachs Preludes and Fugues (one in each major and minor key) can be quickly compared by ear. The overall spectrum of the entire collection of preludes could then, for example, be sonified so as to be compared by ear to those of other similar collections of preludes or etudes played on piano (such as those of Chopin or Listz).

4. CONCLUSIONS

We have presented an algorithm for spectral summarisation, and have applied it to the problem of music summarisation. This method is based on concatenative and granular synthesis and aims to strip musical audio of its fine temporal content while maintaining the spectral shape and energy. We have described methods for applying this basic spectral summarisation technique to the analysis of harmony and voice leading, and for using it to compare chords against the tonic chord of a musical piece.

4.1 Applications, Limitations and Future Work

These techniques are useful for assessing musical samples in a semi-automated manner - in a way that hopefully falls somewhere between listening to the entire audio file, and an abstract information retrieval algorithm. In this way it may be possible to apply this method in the design or checking of music information retrieval algorithms. This representation method may also be useful in education contexts, for the assessment of spectra, and to introduce ideas of structure and tonality. Its application to auditory browsing, for instance of digital archives of musical recordings, is also worth consideration.

Short signals highlight a limitation of this method – a certain amount of audio data is required to reliably build an average window from. For short signals the use of large windows is also difficult, leading to the tradeoff between spectral smearing and window length described in 2.1. This method is likely to be experienced in an interactive context, due to its reliance on computer technology. Investigation into good ways to provide interactive user access to this algorithm is likely to greatly improve its usefulness. Lastly, the suppression of the audio's temporal information throws away a lot of temporal qualities that are fundamental in musical practice. Modifications of this method that seek to systematically explore aspects of music apart from only the spectral and harmonic qualities are worth careful consideration.

It is easy to forget how powerful auditory analysis can be when visual and textual presentation of data are overwhelmingly common. Sonification of audio is more than a tautology, and extends beyond the trivial case of merely playing the original audio recording. This paper examines one simple technique for the sonification of sound recordings which focuses on spectral features. One of the attractive features of this technique is that it does not employ any spectral analysis using digital signal processing instead the spectrum analysis is achieved in the ear, and the purpose of the technique is to prepare the audio so as to provide a sound that focuses attention on spectral features. In other work we have examined other spectrum sonification techniques that do use Fourier transforms, such as exaggerating spectral features through auto-convolution (raising the spectrum to an integer power) [2].

Applications of this technique extend beyond conventional harmony-based music, and beyond music. Broadly speaking, it is applicable to audio recordings that have medium or long term spectral features of interest (including harmony, timbre) that might be difficult to clearly discern without the removal of temporal structure and/or the compression of duration.

4.2 Acknowledgements

This research was supported by an Australian Postgraduate Award and a Departmental Supplementary Scholarship.

5. REFERENCES

[1] G. Kramer, B. N. Walker, Terri Bonebright, P. R. Cook, J. H. Flowers, Nadine Miner, and John G. Neuhoff. Sonification report: Status of the field and research agenda. Technical report, NSF, 1997.

[2] D. Cabrera, & S. Ferguson. Auditory Display of Audio. In *120th Audio Engineering Society Convention*, Paris, France, 2006.

[3] D. Cabrera & S. Ferguson. Sonification of sound: Tools for teaching acoustics and audio, In *13th International Conference on Auditory Display*, Montreal, Canada, 2007.

[4] S. Ferguson. *Exploratory Sound Analysis: Statistical Sonifications for the Investigation of Sound*. Ph.D Thesis, University of Sydney, 2009.

[5] D. Gabor. Theory of communication. *J. Inst. Elec. Eng.*, 93:429–457, 1946.

[6] D. Gabor. Acoustical quanta and the theory of hearing. *Nature*, 159:591–594, 1947.

[7] C. Roads. *Microsound*. MIT Press, Cambridge, 2001.

[8] D. Schwarz. *Data-driven Concatenative Sound Synthesis*. Ph.D Thesis, University of Paris 6 Pierre et Marie Curie, 2004.

[9] D. Schwarz, R. Cahen, and S. Britton. Principles and applications of interactive corpus-based concatenative synthesis. In *Journées d'Informatique Musicale (JIM'08)*, Albi, 2008.

[10] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, Boulder, CO, USA, 1996.

[11] B. Fry. *Computational Information Design*. Ph.D Thesis, MIT, Cambridge, MA, 2004.

[12] H. Schenker and F. Salzer. *Five graphic music analyses (Funf Urlinie-Tafeln)*. Dover Publications, New York, 1969.

[13] B. Snyder. *Music and Memory: An Introduction*. MIT Press, Cambridge, MA, 2001.

[14] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *J. New Music Res.*, 30(1), 2001.

[15] E. Terhardt, G. Stoll, and M. Seewan. Algorithm for extraction of pitch and pitch salience from complex tonal signals. *J. Acoust. Soc. Am.*, 71(3):679–688, 1982.

COVER SONG RETRIEVAL: A COMPARATIVE STUDY OF SYSTEM COMPONENT CHOICES

Cynthia C.S. Liem, Alan Hanjalic
Department of Mediamatics, Delft University of Technology, The Netherlands
{c.c.s.liem,a.hanjalic}@tudelft.nl

ABSTRACT

The Cover Song Retrieval (CSR) problem has received considerable attention in the MIREX 2006-2008 evaluation sessions. While the reported performance figures provide a general idea about the strengths of the submitted systems, it is not clear what actually causes the reported performance of a certain system. In other words, the question arises whether some system component design choices are more critical for a system's performance results than others. In order to obtain a better understanding of the performance of current CSR approaches and to give recommendations for future research in the field of CSR, we designed and performed a comparative study involving system component design approaches from the best-performing systems in MIREX 2006 and 2007. The datasets used for evaluation were carefully chosen to cover the broad spectrum of the cover song domain, while still providing designated test cases. While the choice of the dissimilarity assessment method was found to cause the largest CSR performance boost and very good retrieval results were obtained on classical opus retrieval cases, results obtained on a new test case, involving recordings originating from different microphone sets, point out new challenges in optimizing the feature representation step.

1. INTRODUCTION

Cover Song Retrieval (CSR) generally refers to the problem of identifying different interpretations of the same musical work. Since 2006, this challenge has been included in the centralized yearly Music Information Retrieval (MIR) evaluation sessions known as the MIR EXchange (MIREX). Ever since, several systems for this task have been submitted and evaluated on a fixed, but undisclosed dataset. As the results obtained by these systems are expressed in the form of general performance numbers, no information is provided that could reveal the influence of specific CSR system component design choices and the composition of the evaluation dataset on the obtained retrieval results.

Although CSR appears to be more specific than e.g. music genre retrieval, cover songs still span a broad range of types, each with their own variants and invariants, posing specific challenges on the design of the CSR system. In order to validate design motivations and identify which system aspects are most critical for performance results, it is necessary to consider CSR systems as combinations of general system components and review performance with respect to these components. Additionally, the design of the evaluation dataset is critical for obtaining true insight into the performance of CSR systems.

In this paper, a comparative study is presented with special attention to the influence of individual system components and the composition of evaluation datasets on CSR system performance. We look at the two best-performing systems in MIREX 2006-2007, breaking them down into separate, generic components, which are recombined into alternative combinations. These are evaluated on 4 different datasets. Attention will hereby be paid to the validation of several 'semantically intuitive' choices in the systems. In this way, we aim at achieving better understanding of current CSR approaches, identifying which system components are most critical for the final performance results and which research directions deserve further attention in future CSR research.

2. PROBLEM DESCRIPTION

2.1 Definition of 'Cover Song'

While the term 'cover song' (or simply 'cover') used to suggest a pop music phenomenon, it has more recently been defined as 'a recording of a song or tune which has previously been recorded by someone else' [1]. This broad definition has typically been accepted in the MIR research field, accepting alternate takes of a song by the same artist to be covers as well. When considering the broad range of cover songs according to this definition, many musical aspects can be thought of that may vary among different covers. Several good suggestions for musical aspects that can be used in characterizing cover songs are given in [1].

2.2 Cover Song System Components

For the CSR problem discussed in this paper, a setting is assumed in which an example raw audio file is provided

[1] This also is seen in dictionaries, e.g. see http://dictionary.cambridge.org/define.asp?key=17817&dict=CALD, accessed May 2009.

as a query to a dataset, after which the audio files in the dataset are returned in an ordered way, according to their similarity score compared to the query. In this setting, CSR systems can be characterized using a general model, consisting of two main components:

- *Feature representation*, transforming a raw audio file into a representation suitable for further matching;
- *Dissimilarity assessment*, achieving the actual matching, applying a chosen dissimilarity measure.

Two more system aspects concerned with post-processing of the feature representation will further be considered:

- The typical approach of using short-time harmonic features for the feature representation produces very much data. In order to reduce this amount of data, an *averaging step* is adopted.
- In order to handle varying sound intensity levels, which can be caused both by the quality of the recording and by musical dynamics, a *normalization procedure* is usually applied to the chosen feature representation.

The musical variants expected in cover songs have influenced design choices for the mentioned system components. For the feature representation, chromagrams are commonly chosen[2]. These are considered to model melodic/harmonic progression over time without the need for exact transcriptions, while being robust to specific instrument timbres. Besides, multiple interpretations of a song will inevitably introduce tempo and timing variations, which also should be accounted for in CSR systems.

If system evaluations are done as a whole, it will not be clear from the results which of these component choices are most important to the final performance results. Additionally, validation of design choice motivations (such as the timbre-robustness of chromagrams) will be difficult.

2.3 Importance of Evaluation Dataset Composition

While during the system design, attention is paid to possible musical variants in cover songs, these do not appear to be considered with the same importance in system evaluation. Evaluation datasets typically are colorful cross-sections of private music collections, which are sought to contain as much musical variation as possible. However, the more variants in the dataset, the more difficult it will be to interpret an overall performance number. Given the broadness of the cover song spectrum, understanding of a system's performance can only be achieved if attention is paid to the types of cover song similarity test cases posed by a dataset.

A common problem in audio-based MIR research is the lack of public benchmark data. When different authors report performance numbers on different private music collections, comparison of their approaches cannot be made easily. The MIREX endeavour offered a centralized solution to this, comparing multiple algorithms on the same evaluation data. However, as details regarding the evaluation dataset composition are not revealed to the participants, only comparative information on total system performance is provided, while algorithm behavior on specific test cases once again remains unclear.

2.4 Contribution

To address the problems described above and gain more in-depth understanding of CSR performance in current approaches, in this paper, we describe a comparative study with two main focus points:

- to investigate the impact of choices in each individual general CSR system component listed above on the CSR performance;
- to relate the achieved performance results to specific test cases provided by the evaluation data.

The setup of this comparative study is explained in Section 3, while the results are reported and discussed in Section 4. We finish the paper in Section 5 with conclusions and recommendations for future work.

3. EVALUATION SETUP

In this section we first explain the systems we selected and implemented for our comparative study.

3.1 Basic Systems

3.1.1 Best CSR system in MIREX 2006

The system proposed by Ellis et al. in [2] was the best-performing system in the first MIREX CSR Task, held in 2006. We use the implementation that has been made available by the author [3], which is very similar to this original 2006 MIREX CSR submission.

Regarding the feature representation, chromagrams based on instantaneous frequency (CIF) are used. Features are averaged over beats, which appears to be a semantically intuitive choice, allowing robustness to tempo variances; a beat tracker is needed in order to achieve this. For normalization, each 12-bin chroma vector in the chromagram is normalized to unit norm.

For similarity assessment, cross-correlation (CC) is performed. In order to allow for different key transpositions, all 12 possible chroma transpositions are considered in this correlation step. Subsequently, a similarity score is achieved through the maximum peak correlation value found. This can be changed into a dissimilarity score by taking the reciprocal of this value.

3.1.2 Best CSR system in MIREX 2007

The system proposed by Serrà et al. in [4] was the best-performing system in the second MIREX CSR Task, held in 2007. This system showed a striking performance increase compared to all other systems; an improved version

[2] see for example the extended abstracts on `http://www.music-ir.org/mirex/[yearofsession]/index.php/Audio_Cover_Song_Identification_Results`, with 2006, 2007 and 2008 as possible session years (accessed May 2009).

also convincingly showed the best performance results in the 2008 MIREX CSR Task [5].

As no implementations of this system are publicly available, for the experiments described, the system has been reimplemented from the literature, using the information in [1, 4, 6]. The preprocessing steps (transient localization and spectral normalization) have still been omitted in our implementation, as it was not completely clear which procedures were exactly followed for these steps. For the same reasons, the system changes and parameter tunings mentioned in [5] could not be implemented, so our implemented system will show the most resemblance to the MIREX 2007 submission by the authors.

For feature representation, Harmonic Pitch Class Profiles (HPCPs) [6] are used. These are chromagrams (or pitch class profiles) in which each spectral peak contribution is weighted across multiple chroma bins. Additional contribution is weighted into the final representation by taking into account the first 8 harmonics of each spectral peak. Averaging is done over a fixed number of frames, as beat tracking was found to include additional errors that decreased performance (this also was noted in [7]). Normalization is performed by dividing a HPCP instance by the maximal value found in this instance, yielding a profile in which the maximum value is 1.

For matching, a procedure was devised called Dynamic Programming Local Alignment (DPLA), using binary similarity. For the two audio HPCP vectors to be matched, first an Optimal Transposition Index is computed. Subsequently, after applying the found optimal key transposition, a binary similarity matrix is constructed, based on remaining optimal transposition indices per HPCP short-time instance after the global transposition. Subsequently, in a way similar to string or DNA matching, a dynamic programming procedure with local constraints (for tempo fluctuations) is applied. The best path found will decide the similarity score, which is normalized to a dissimilarity score. More information on these procedures can be found in [1]. Parameter choices have been directly taken from [1]; as for the averaging factor, the choice was made to consider an averaging factor of 10 frames. Furthermore, only 12-bin HPCPs are considered instead of the suggested 36 bins, as 12 bins were used both in the Ellis et al. system and in later versions of the Serrà et al. system.

3.2 Considered Approaches

Using the systems described above, several possible general design choices can be extracted. The following choices have been verified in our algorithms:

- The general choice of feature representation: (1) *Chromagrams based on Instantaneous Frequency (CIF)*, (2) *Harmonic Pitch Class Profiles (HPCP)* and (3) *Pitch Class Profiles (PCP)*, which are constructed similarly to HPCPs, but omitting the additional harmonic weighting.
- The averaging factor for the feature representation: (1) *averaging over beats* and (2) *averaging over a fixed number of frames*.
- The matching procedure for dissimilarity assessment: (1) *cross-correlation (CC)* and (2) *Dynamic Programming Local Alignment (DPLA)*.

All possible combinations of these choices have been tested, with three possible normalization choices regarding feature representation: (1) *no normalization*, (2) *normalization to unit norm* and (3) *normalization by the maximum*.

3.3 Performance measures

We evaluate the systems using 2 evaluation measures, which also were adopted in the most recent MIREX evaluations [8]:

- (Arithmetic) Mean of average precisions (MAP);
- Mean rank of 1st correctly identified cover (MR1st).

The most recent MIREX evaluations employ two more evaluation measures focusing on the top-10 retrieval results. However, in our experiments, only MAP and MR1st will be suitable performance indicators: our datasets, which are discussed hereafter, contain cover sets of different sizes, as opposed to the MIREX dataset which contained 10 relevant cover versions per query song.

3.4 Datasets

4 datasets have been used in our experiments, which will be described now. The construction and choice for the datasets has largely been motivated by the need to provide clear and designated test cases. The choice was made to use 4 separate datasets in order to provide a clear-cut corpus per dataset. All audio tracks have been converted to the MP3 format. For each dataset, each audio file in the dataset was matched against all other files in the same dataset.

3.4.1 Covers80 dataset

This dataset, containing pop song covers, was made available by Ellis [3]. 166 recordings are included, encompassing 80 'cover sets', which means the average number of versions is just 2.05. With the dataset being constructed rather randomly, musical variants within the dataset differ greatly and interpreting performance measures will be difficult. We decided to include results on this set anyway for reference reasons.

3.4.2 Beethoven piano sonatas

This dataset contains multiple interpretations of movements from 4 Beethoven piano sonatas. The data in this dataset originates from private music collections of the authors and the Beeld en Geluid (BeG) vinyl collection [3] in the European archive. The dataset contains 128 recordings, encompassing 13 'cover sets'. As piano sonatas are

[3] `http://europarchive.org/collection.php?id=public_classical_music_BeG`, accessed May 2009.

considered, all covers will consist of very similar instrument timbres and will be played in exactly the same key. Therefore, the set has very clear invariants and poses well-defined (although not too challenging) similarity tasks. A set of similar composition was used for a CSR system mentioned in [9], which showed near-perfect performance.

3.4.3 Songs

This dataset departs from recordings of classical art songs that were performed at the 1st International Student Lied Duo Competition, held in Enschede in April 2009. It contains study recordings from one of the participating duos, made at rehearsals and try-outs in preparation for the competition. Additionally, recordings of all the participants made during the official competition rounds are included. More specifically, included songs encompass compulsory songs, as well as songs that were performed by multiple different participants. Finally, the set was extended with extra song interpretations from private music collections, the BeG vinyl collection and the vinyl recordings from the King's Sound Archive [4] . In total, the dataset contains 205 recordings, encompassing 21 'cover sets'.

At the competition, recordings have been made with two different pairs of microphones at two different locations in the hall (on stage and in the hall). While recordings from these two pairs contain exactly the same musical interpretation, the recordings do show considerable acoustical differences. As this poses an interesting test case for the CSR algorithms, the takes from both microphone pairs have been included in the dataset.

Both this songs dataset and the Beethoven dataset consider multiple interpretations of exactly the same score. The problem of retrieving such interpretations has sometimes been considered as a subtask within CSR, known as opus retrieval. The difference between both sets is that the songs set shows much more variation in instrument timbre and musical keys, as the performing singers have different voice types.

3.4.4 Beatles

This dataset aims at being a slightly more specific dataset than the covers80 set with larger 'cover set' sizes, while still reflecting a similar corpus. The dataset contains original Beatles songs (including alternative takes and versions), as well as various covers taken from tribute CDs, including Baroque, R&B, Latin and easy listening styles. In total there are 197 audio files, encompassing 51 'cover sets'. On one of the CDs used, 4 covers were present of songs from individual Beatles members. These were included in our database without providing alternative versions. Typical CSR evaluation experiments contain even more of such 'outlier noise' files in evaluation datasets (e.g. MIREX), but in our experiments they explicitly have not been included extensively in order to focus on system behavior on actual covers.

[4] `http://www.kcl.ac.uk/kis/schools/hums/music/ksa/ksa_sound.html`, accessed May 2009.

4. RESULTS

Each of the possible combinations mentioned in Subsection 3.2 has been tested on all 4 datasets. The resulting MAP and MR1st scores are plotted twice, both in Figure 1 and Figure 2. While the performance scores are the same in both figures (expressed in data points at the same locations on the vertical axis), the used data markers indicate different system choices. In Figure 1, the data point markers indicate corresponding combinations of feature representations and dissimilarity assessment, while the data point markers in Figure 2 indicate corresponding combinations of normalization and averaging choices. In the figures, baseline results for random guessing are indicated as well, which were obtained by generating 50 random similarity matrices for each dataset and averaging the obtained results. Because of space limitations, only the results of the best-performing system combinations are numerically expressed in Table 1.

Dataset	MAP	MR1st
covers80	0.648	15.817
Beethoven	1	1
songs	0.986	1
Beatles	0.693	5.699

Table 1. Best performance scores for each of the datasets

The best results turn out to occur for the same combination consistently: the CIF feature representation, averaged over a fixed number of 10 frames, normalized to unit norm and with dissimilarity assessment based on DPLA. This means aspects from both studied original systems combine into an optimally performing system.

With respect to the feature representation choice, the CIF representation generally does not perform worse than HPCPs or PCPs. In the pop music datasets (covers80 and Beatles), it even performs clearly better than HPCPs and PCPs. Besides, as mentioned above, the CIF representation consistently occurs in the best-performing system component combinations for each of the 4 datasets. Regarding the difference between HPCPs and PCPs, the harmonic weighting in the HPCPs does not give convincing performance increases when compared to PCPs. While HPCPs were known to yield the highest correlation scores when compared to symbolic note information [6], this does not appear to be a convincing advantage in the CSR problem, which deals with approximate matches.

The notion in [1,4] that DPLA dissimilarity assessment yields much better results than CC is convincingly confirmed for all 4 datasets. This also holds for the statement in [1, 4, 7] that averaging over a fixed number of frames improves performance in comparison to averaging over tracked beats. While all better-performing system combinations contain normalized feature representations, the performance increase from the normalization step is much smaller than the increases caused by choosing DPLA dissimilarity and averaging over a fixed number of frames. Furthermore, there is no specific normalization

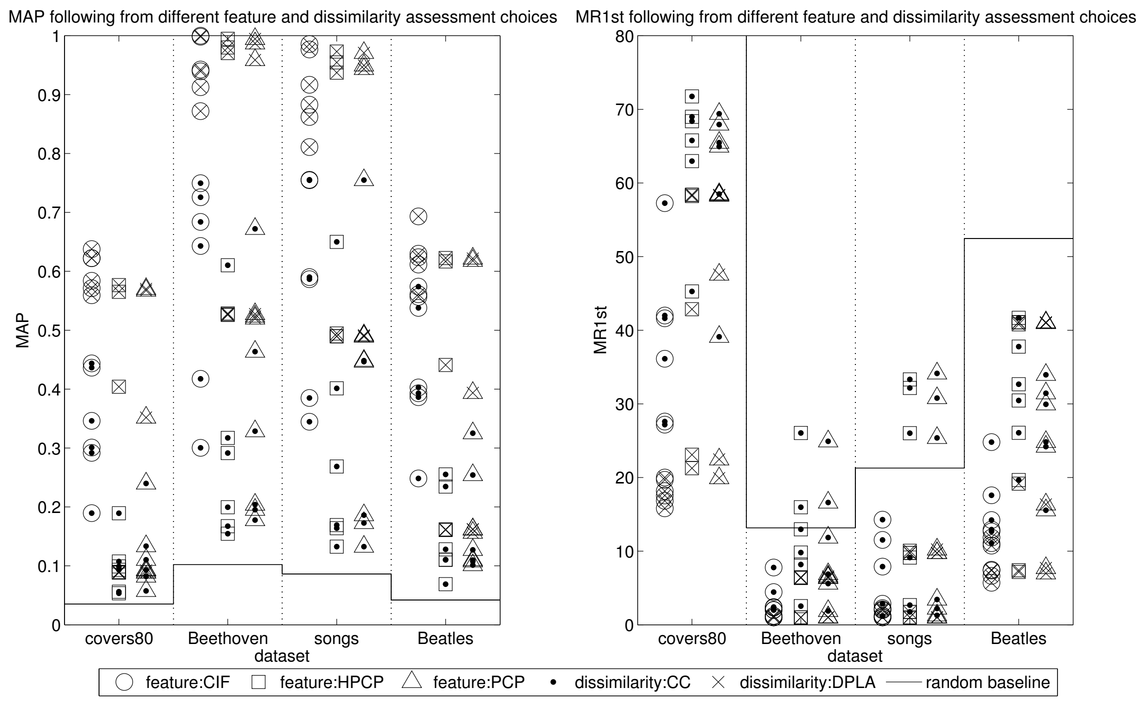


Figure 1. MAP and MR1st for the 4 datasets with feature and dissimilarity assessment choices indicated.

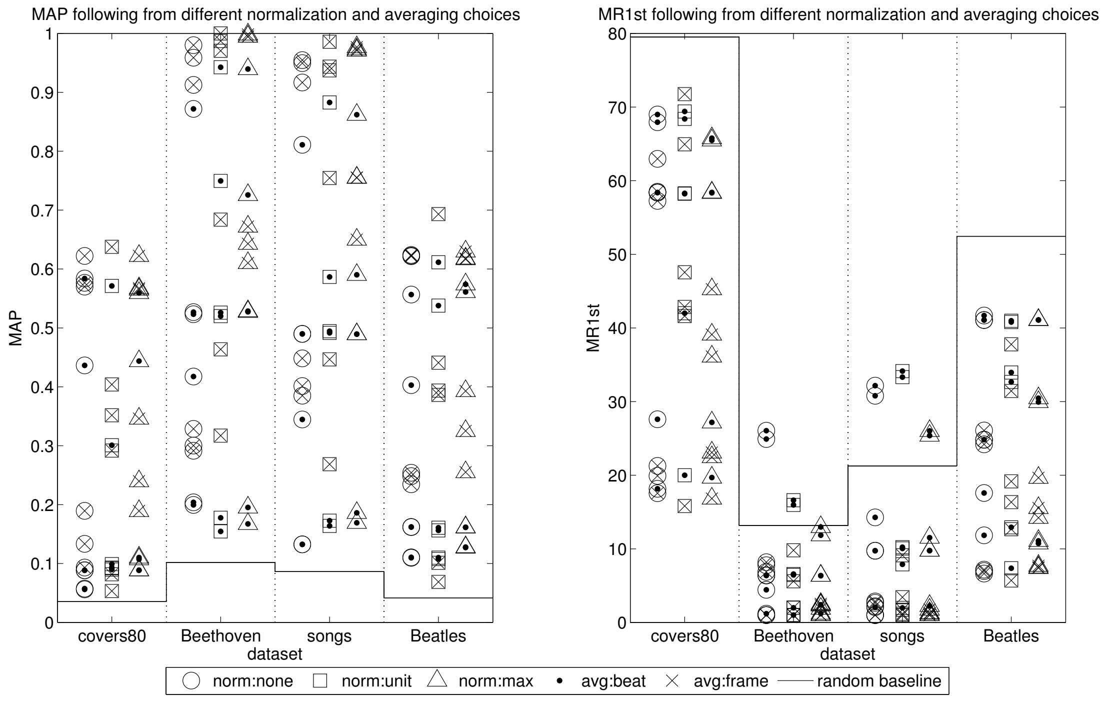


Figure 2. MAP and MR1st for the 4 datasets with normalization and averaging choices indicated.

choice performing convincingly better than other normalization choices.

As expected, the results for the classical opus retrieval cases (Beethoven and songs) are better than those on the pop music datasets. However, it is remarkable how close the performance on both classical datasets is, despite the much larger variations in timbre and key in the songs dataset. Errors in near-perfect results on the Beethoven set are caused by the historic vinyl recordings, which are degraded in quality compared to modern recordings. However, as shown in Table 1, the best-performing system combination was robust to the vinyl recording sound distortions, having perfect retrieval results on this set. In the Beatles database, if besides a query multiple alternative recordings of the same artists exist, these recordings are ranked very high in the retrieval results. However, the performance results worsen because of the inability of all implemented approaches to deal with the freer covers, such as the easy listening piano versions of the Beatles songs.

The similarity test cases posed by the songs dataset demonstrate some other interesting properties of the current CSR approaches. If a song is available in multiple interpretations from the same musicians, these interpretations are usually ranked higher than interpretations of other musicians. This might be due to timing aspects rather than timbral aspects, as interpretations of other singers of the same voice type as the singer in the query do not consistently rank higher than interpretations of other singers of other voice types or even the other gender. This validates the hypothesis that the followed approaches show timbre-robustness. The claimed key invariance of all approaches is also confirmed in our results, as songs sung in the same key as a query do not always rank higher than recordings of the song in other keys.

In the best system combinations for the songs database, if an alternate microphone recording of a given song is available, it is retrieved as the best-matching song. However, while such recording pairs undoubtedly contain exactly the same musical interpretation, the found dissimilarity scores of both pair members compared to a query of another interpretation are not identical. It even is not guaranteed that both pair members will be neighbors in the corresponding dissimilarity ranking to the query. This is an interesting notion that does not match our human notion of interpretation similarity.

5. CONCLUSION AND DISCUSSION

In this paper, more insight into the performance of current CSR approaches was sought through a comparative study, in which different combinations of CSR system components were evaluated on 4 carefully constructed datasets. The obtained results show that choices that semantically seemed intuitive do not necessarily yield better performance results: including harmonic weighting into a feature representation does not convincingly show performance improvement, while averaging the representations over beats actually makes the results worse.

Regarding the system components, the best feature representation found consistently in our experiments is the CIF representation, which is not the representation used in the best MIREX systems of 2007 and 2008. However, the dissimilarity assessment method used in those systems, binary similarity using DPLA, gives a large performance increase in comparison to using CC. This suggests that the dissimilarity measure has been the crucial factor in the success of the best MIREX CSR system submissions of 2007 and 2008. The remaining system aspect that was tested, the feature normalization, only gives a slight increase in performance.

Successful CSR system component combinations can deal very well with opus retrieval tasks, even if large timbre and key variance is present. However, ranking results for the different microphone recording pairs in the songs dataset show different ranks for identical musical interpretations which only differ in terms of the acoustical conditions. Therefore, the difference in the dissimilarity must be due to the feature representation, suggesting that further improvement is still possible here.

While the major changes from the best-performing system of MIREX 2007 to that of 2008 mainly focused on improving the dissimilarity assessment part [5], improvement possibilities in the other system components, especially the feature representation, are clearly not excluded. Further experiments are needed into alternatives that will be able to yield results that better approach our human notions of cover song similarity.

6. REFERENCES

[1] J. Serrà. Music similarity based on sequences of descriptors: Tonal features applied to audio cover song identification. Master's thesis, University Pompeu Fabra, Barcelona, Spain, September 2007.

[2] D.P.W. Ellis and G.E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume IV, pages 1429–1432, Honolulu, USA, April 2007.

[3] D. Ellis. The `covers80` cover song data set. Web resource, available: `http://labrosa.ee.columbia.edu/projects/coversongs/covers80`, 2007.

[4] J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Trans. on Audio, Speech and Language Proc.*, 16:1138–1151, August 2008.

[5] J. Serrà, E. Gómez, and P. Herrera. Improving binary similarity and local alignment for cover song detection. MIREX 2008 extended abstract, available: `http://www.music-ir.org/mirex/2008/abs/CS_Serra.pdf`, September 2008.

[6] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, July 2006.

[7] J.P. Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *Proc. of the Intl. Conf. on MIR (ISMIR)*, Vienna, Austria, September 2007.

[8] J.S. Downie, M. Bay, A.F. Ehmann, and M.C. Jones. Audio cover song identification: MIREX 2006-2007 results and analyses. In *Proc. of the Intl. Conf. on MIR (ISMIR)*, Philadelphia, USA, September 2008.

[9] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *Proc. of the IEEE*, 96(4):668–696, April 2008.

AUGMENTING TEXT-BASED MUSIC RETRIEVAL WITH AUDIO SIMILARITY

P. Knees[1], T. Pohle[1], M. Schedl[1], D. Schnitzer[1,2], K. Seyerlehner[1], and G. Widmer[1,2]
[1]Department of Computational Perception, Johannes Kepler University Linz, Austria
[2]Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria

ABSTRACT

We investigate an approach to a music search engine that indexes music pieces based on related Web documents. This allows for searching for relevant music pieces by issuing descriptive textual queries. In this paper, we examine the effects of incorporating audio-based similarity into the text-based ranking process – either by directly modifying the retrieval process or by performing post-hoc audio-based re-ranking of the search results. The aim of this combination is to improve ranking quality by including relevant tracks that are left out by text-based retrieval approaches. Our evaluations show overall improvements but also expose limitations of these unsupervised approaches to combining sources. Evaluations are carried out on two collections, one large real-world collection containing about 35,000 tracks and on the CAL500 set.

1. MOTIVATION AND RELATED WORK

In the last years, the development of *query-by-description* music search engines has drawn increasing attention [1–5]. Given the size of (commercial) digital music collections nowadays (several millions of tracks), this is not a big surprise. While most "traditional" music retrieval approaches pursue a *query-by-example* strategy, i.e., given a music piece, find me other pieces that sound alike, query-by-description systems are capable of retrieving relevant pieces by allowing to type in textual queries targeting musical or contextual properties beyond common meta-data descriptors. As this method of issuing queries is the common way to search the Web, it appears desirable to offer this type of functionality also in the music domain.

Several approaches to accomplish this goal have been presented – all of them with a slightly different focus. In [1], Baumann et al. present a system that incorporates meta-data, lyrics, and acoustic properties all linked together by a semantic ontology. Queries are analyzed by means of natural language processing and tokens have to be mapped to the corresponding concepts. Celma et al. [2] use a Web crawler focused on audio blogs and exploit the texts from the blogs to index the associated music pieces. Based on the text-based retrieval result, also musically similar songs can be discovered. In [3], we propose to combine audio similarity and textual content from Web documents obtained via Google queries to create representations of music pieces in a term vector space. A modification to this approach is presented in [6]. Instead of constructing term vector representations, an index of all downloaded Web documents is created. Relevance wrt. a given query is assessed by querying the Web document index and applying a technique called *rank-based relevance scoring* that takes into account the associations between music tracks and Web documents (cf. Section 2.1). Evaluations show that this document-centered approach is superior to the vector space approach. However, as this method is solely based on texts from the Web it may neglect important acoustic properties and suffer from effects such as popularity bias. Furthermore, inadequately represented tracks and tracks not present on the Web are penalized by this approach. In this paper, we aim at remedying these shortcomings and improving ranking quality by incorporating audio similarity into the retrieval process.

Recently, the method of relevance scoring has also been adapted to serve as a source of information for automatically tagging music pieces with semantic labels. In [5], Barrington et al. successfully combine audio content features (MFCC and Chroma) with social context features (Web documents and last.fm tags) via machine learning methods and therefore improve prediction accuracy. The usefulness of audio similarity for automatic tagging is also shown in [4] where tags from well-tagged tracks are propagated to untagged tracks based on acoustic similarity.

The remainder of this paper is organized as follows: In the next section we review methods for Web-based music track indexing and audio-based similarity computation. Section 3 describes two possible modifications of the initial approach that are examined in Section 4. In Section 5, based on these results, we discuss perspectives and limitations of combining Web- and audio-based approaches before drawing conclusions in Section 6.

2. INCORPORATED TECHNIQUES

In the following, we explain the methods for constructing a Web-based retrieval system and calculating audio similarity, which we combine in Section 3.

2.1 Web-based Indexing and RRS Ranking

The idea of Web-based indexing is to collect a high number of texts related to the pieces in the music collection to gather many diverse descriptions (and hence a rich indexing vocabulary) and allow for a large number of possible queries. In our first approach, we aimed at permitting virtually any query by involving Google for query expansion [3]. When introducing *rank-based relevance scoring (RRS)*, we renounced this step in favor of reduced complexity and improved ranking results [6]. From our point it is very reasonable to limit the indexing vocabulary to terms that actually co-occur with the music pieces (which is still very large). Construction of an index with a corresponding retrieval scheme is performed as follows.

To obtain a broad basis of track specific texts, for each music piece m in the collection M, three queries are issued to Google based on the information found in the id3 tags of the music pieces:

1. "*artist*" music
2. "*artist*" "*album*" music review -lyrics
3. "*artist*" "*title*" music review -lyrics

For each query, at most 100 of the top-ranked Web pages are retrieved and joined into a set (denoted as D_m in the following). For retrieval, we utilize the open source package *Nutch*[1]. Beside efficient retrieval, a further benefit is that all retrieved pages are also automatically indexed by *Lucene*[2] that uses a *tf*x*idf* variant as scoring function [7]. The resulting Web page index is then used to obtain a relevance ranking of Web pages for arbitrary queries. This page ranking, together with the information on associations between pages and tracks, serves as input to the RRS scheme. Compared to the original formulation in [6], we introduce the additional parameter n that is used to limit the number of top-ranked documents when querying the page index. For large collections, this is necessary to keep response time of the system short. For a given query q and for each music piece m, scores are calculated as:

$$RRS_n(m,q) = \sum_{p \in D_m \cap D_{q,n}} 1 + |D_{q,n}| - rnk(p, D_{q,n}), \tag{1}$$

where D_m is the set of text documents associated with music piece m (see above), $D_{q,n}$ the ordered set (i.e., the ranking) of n most relevant text documents with respect to query q, and $rnk(p, D_{q,n})$ a function that returns the rank of document p in $D_{q,n}$ (highest relevance corresponds to rank 1, lowest to rank $|D_{q,n}|$). The final relevance ranking of music tracks is then obtained by sorting the music pieces according to their RRS value.

Note that, as suggested in [5, 8], we also experimented with a weight-based version of relevance scoring (WRS) that incorporates the scores of the Web page retrieval step rather than the ranks. In our framework this modification worsened performance. Possible explanations are the differences in the underlying page scoring function or the different sources of Web pages (cf. [8]).

2.1.1 Pseudo-Document Indexing

Instead of modifying the page scoring scheme, we invented a simple alternative approach for text-based indexing that lies conceptually between the first vector space approach [3] and the relevance scoring scheme. For each music piece m, we concatenate all retrieved texts (i.e., all texts from D_m) into a single document which we index with *Lucene*. Hence, each music piece is represented by a single document that contains all relevant texts. Querying this *pseudo-document index* results directly in a ranking of music pieces. This rather "quick-and-dirty" indexing method will serve as a reference point in the evaluations and give insights into the capabilities of purely Web-based retrieval.

2.2 Audio-Based Similarity

For calculating music similarities, or more precisely, distances of tracks based on the audio content, we apply our algorithm which competed successfully in the "Audio Music Similarity and Retrieval" task of MIREX 2007 [9]. For each piece of music, *Mel Frequency Cepstral Coefficients (MFCCs)* are computed on short-time audio frames to characterize the frequency distribution of each frame and hence model aspects of timbre. On each frame, 25 MFCCs are computed. Each song is then represented as a *Gaussian Mixture Model (GMM)* of the distribution of MFCCs using a Single Gaussian Model with full covariance matrix [10]. The distance between these models is denoted by d_G.

Beside the MFCC-based distance component, also *Fluctuation Patterns (FPs)* are computed as proposed in [11]. A track is represented as a 12-band spectrogram and for each band, a *Fast Fourier Transformation (FFT)* of the amplitude is taken over a window of six seconds. The resulting matrix is referred to as the Fluctuation Pattern of the song. The FPs of two songs are compared by calculating the cosine distance, denoted by d_{FP}. Furthermore, two additional FP-related features are computed: *Bass (FPB)* and *Gravity (FPG)*. These two features are scalar and the distance between two songs is calculated by subtracting them, denoted by d_{FPB} and d_{FPG}. To obtain an overall distance value d measuring the (dis)similarity of two songs, all described distance measures are z-normalized and then combined by a simple arithmetic weighting:

$$d = 0.7 \cdot z_G + 0.1 \cdot (z_{FP} + z_{FPB} + z_{FPG}) \tag{2}$$

where z_x is the value of d_x after z-normalization. Finally, distances between two songs are symmetrized. For similarity computation, we ignore all pairs of songs by the same artist (artist filtering, cf. [12]) since this similarity is already represented within the Web features.

3. COMBINATION APPROACHES

This section describes two different approaches for combining the purely text-based retrieval approach with the

[1] `http://lucene.apache.org/nutch`
[2] `http://lucene.apache.org`

audio-based similarity information. According to [5, 13], the first approach can be considered an *early fusion* approach, since it incorporates the audio similarity information directly into the relevance scoring scheme, whereas the second approach can be considered a *late fusion* approach, since it modifies the ranking results obtained from the Web-based retrieval. Basically, both algorithms incorporate the idea of including tracks that sound similar to tracks already present through text-only retrieval. The score of a track m is calculated by summing up a score for being present in the text-based ranking and scores for being present within the nearest audio neighbors of tracks associated with the text-based ranking.

3.1 Modifying the Scoring Scheme (aRRS)

With this approach, we try to incorporate the audio similarity directly into the scoring scheme of RRS. The advantage is that this has to be calculated only once and does not require post-processing steps. The *audio-influenced RRS (aRRS)* is calculated as:

$$aRRS_n(m,q) = \sum_{p \in P_{m,q,n}} RF(p, D_{q,n}) \cdot MF(m,p), \quad (3)$$

$$RF(p, D_{q,n}) = 1 + |D_{q,n}| - rnk(p, D_{q,n}), \quad (4)$$

$$MF(m,p) = \alpha \cdot I(p, D_m) + \sum_{a \in A_m} I(p, D_a), \quad (5)$$

where $P_{m,q,n} = (D_m \cup D_{A_m}) \cap D_{q,n}$, $N_{a,k}$ the k nearest audio neighbors of a, A_m the set of all tracks a that contain m in their nearest audio neighbor set, i.e., all a for which $m \in N_{a,k}$, D_{A_m} the set of all documents associated with any member of A_m, and $I(x, D)$ a function that returns 1 iff $x \in D$ and 0 otherwise. Informally speaking, also tracks sounding similar to track m participate if a page relevant to m occurs in the page ranking for query q. The parameter α is used to control the influence of tracks that are directly associated with a Web page (in contrast to tracks associated via audio neighbors). In our experiments we set $\alpha = 10$. Note that aRRS is a generalization of RRS, as they are identical for $k = 0$.

3.2 Post-Hoc Audio-Based Re-Ranking (PAR)

The second approach incorporates audio similarity into an already existing ranking R. The advantage of this approach is that it can deal with outputs from arbitrary ranking algorithms. The post-hoc audio-based re-ranking (PAR) is calculated as:

$$PAR(m,R) = \sum_{t \in (m \cup A_m) \cap R} RF(t,R) \cdot NF(m,t), \quad (6)$$

$$NF(m,t) = \alpha \cdot I(m, \{t\}) + G(rnk(m, N_{t,k})) \cdot I(m, N_{t,k}), \quad (7)$$

$$G(i) = e^{-\frac{(i/2)^2}{2}} / \sqrt{2\pi}, \quad (8)$$

We included the gaussian weighting G in this re-ranking scheme because it yielded best results when exploring possible weightings. Parameter α can be used to control the scoring of tracks already present in R. Note that for $k = 0$, R remains unchanged.

4. EVALUATION

For evaluation, we decided to use two test collections with different characteristics. The first collection is a large real-world collection and contains mostly popular pieces. The second collection is the CAL500 set, a manually annotated corpus of 500 tracks by 500 distinct artists [14]. In the following, we describe both test collections in more detail.

4.1 The c35k Collection

The *c35k* collection is a large real-world collection, originating from a subset of a digital music retailer's catalog. The full evaluation collection contains about 60,000 tracks. Filtering of duplicates (including remixes, live versions, etc.; cf. [3]) reduces the number of tracks to about 48,000. As groundtruth for this collection, we utilize last.fm tags. Tags can be used directly as test queries to the system and serve also as relevance indicator (i.e., a track is considered to be relevant for query q if it has been tagged with tag q). From the 48,000 tracks, we were able to find track-specific last.fm tags for about 35,000 of the tracks. To obtain a set of test queries, we started with last.fm's list of top-tags and manually removed tags useless for our purpose (such as *seen live* or tags starting with *favorite*). We also searched for redundant tags (such as *hiphop*, *hip hop*, and *hip-hop*) and harmonized their sets of tagged tracks. However, all forms are kept as queries if they translate to different queries (in the example above, *hiphop* translates to a query with one token, *hip hop* to two tokens, and *hip-hop* to a phrase). As result, a set of 223 queries remained. From the 223 tags we further removed all tags with a number of associated tracks above the 0.95-percentile and below the 0.05-percentile, resulting in 200 test queries. A common way to increase the number of tagged examples is to use artist-specific tags if no track-specific tags are present [3, 8]. Since, in our indexing approach, tracks by the same artist share a large portion of relevant Websites, we decided against combination with artist tags to avoid overestimation of performance.

4.2 The CAL500 Set

The CAL500 set is a highly valuable collection for music information retrieval tasks [14]. It contains 500 songs (each from a different artist) which are manually annotated by at least three reviewers. Annotations are made wrt. a vocabulary consisting of 174 tags describing musically relevant concepts such as genres, emotions, acoustic qualities, instruments, or usage scenarios. Although we consider the fact that our indexing approach is in principle capable of dealing with large and varying vocabularies, some of these tags are not directly suited as query, especially negating concepts (e.g., *NOT-Emotion-Angry*) can

	Recall			**Precision**			**Prec@10**			**r-Precision**			**Avg. Prec. (MAP)**		
Baseline	100.00			3.65			3.60			3.65			3.68		
	Web only		PAR	Web only		PAR	Web only		PAR	Web only		PAR	Web only		PAR
PseudoDoc	93.66		**98.79**	**4.27**		3.67	**39.25**		17.40	**30.78**		22.94	**25.97**		18.81
	RRS	aRRS	PAR	RRS	aRRS	PAR	RRS	aRRS	PAR	RRS	aRRS	PAR	RRS	aRRS	PAR
$n = 10$	2.18	3.67	**10.71**	**30.15**	18.73	6.81	31.19	**31.33**	30.85	2.16	3.27	**6.22**	1.19	1.43	**2.21**
$n = 20$	3.74	6.16	**16.89**	**29.02**	17.95	6.57	**32.40**	32.15	**32.40**	3.63	5.17	**8.46**	1.84	2.25	**3.37**
$n = 50$	7.17	11.28	**27.76**	**27.61**	16.02	6.17	**38.45**	37.85	38.40	6.52	8.37	**11.66**	3.24	3.87	**5.53**
$n = 100$	12.72	19.64	**39.41**	**25.99**	13.72	5.66	**44.10**	43.55	43.95	10.24	12.52	**14.74**	5.54	6.51	**8.44**
$n = 200$	18.67	28.65	**50.98**	**23.77**	12.10	5.25	**47.75**	**47.75**	47.65	14.22	16.67	**17.82**	8.23	9.61	**11.51**
$n = 500$	29.31	44.10	**66.60**	**20.12**	9.77	4.81	**50.30**	49.95	50.15	19.84	21.58	**22.02**	12.39	14.02	**15.91**
$n = 1000$	40.38	58.17	**77.63**	**16.88**	8.12	4.50	**52.55**	51.80	52.35	24.22	24.52	**25.21**	16.10	17.56	**19.23**
$n = 10000$	80.50	95.19	**96.68**	**7.29**	4.25	3.85	57.45	**57.50**	38.20	**35.20**	32.81	32.26	**29.98**	28.48	26.45

Table 1. Evaluation results for the c35k collection: Both re-ranking approaches (aRRS and PAR) are compared against the text-only RRS approach for different numbers of maximum considered top-ranked pages n. For both aRRS and PAR, we set $k = 50$, for PAR, α is also set to 50. Values (given in %) are obtained by averaging over 200 evaluation queries.

	Recall			**Precision**			**Prec@10**			**r-Precision**			**Avg. Prec. (MAP)**		
Baseline	100.00			13.32			13.33			13.31			14.31		
	Web only		PAR	Web only		PAR	Web only		PAR	Web only		PAR	Web only		PAR
PseudoDoc	81.15		**98.83**	**14.50**		13.34	30.72		**31.15**	25.77		**26.28**	22.66		**25.74**
	RRS	aRRS	PAR	RRS	aRRS	PAR	RRS	aRRS	PAR	RRS	aRRS	PAR	RRS	aRRS	PAR
$n = 10$	5.96	**62.23**	59.08	**25.77**	14.49	14.84	**25.77**	23.81	23.74	5.61	18.35	**18.44**	3.58	**14.26**	13.51
$n = 20$	10.19	**80.90**	75.90	**24.87**	13.99	14.34	25.98	**26.40**	25.61	8.84	20.55	**20.70**	5.30	**18.36**	17.20
$n = 50$	17.99	**93.45**	89.52	**22.84**	13.33	13.63	26.06	**27.84**	26.04	13.49	**22.92**	22.23	7.57	**21.55**	20.19
$n = 100$	26.80	**96.60**	94.78	**21.02**	13.15	13.39	29.30	**30.07**	29.28	18.05	**23.88**	23.79	10.59	**23.07**	22.41
$n = 200$	38.63	**97.23**	96.38	**19.15**	13.08	13.22	30.60	**31.58**	30.79	21.58	24.32	**24.38**	13.84	**24.27**	23.90
$n = 500$	56.31	**97.37**	97.05	**16.86**	13.07	13.15	32.68	32.52	**33.17**	24.06	25.79	**25.91**	18.02	25.19	**25.27**
$n = 1000$	66.91	**97.47**	97.18	**15.54**	13.06	13.13	33.47	33.45	**33.60**	24.86	25.90	**26.52**	20.37	**25.85**	25.64
$n = 10000$	73.27	**97.61**	97.31	**14.56**	13.05	13.13	33.62	**33.74**	33.67	25.06	**26.95**	26.76	21.77	**26.58**	25.82

Table 2. Evaluation results for the CAL500 set: Values are obtained by averaging over 139 evaluation queries. Apart from that, the same settings as in Table 1 are applied.

not be used. Hence, we remove all negating tags. Furthermore, we join redundant tags (mostly genre descriptors). For tags consisting of multiple descriptions (e.g., *Emotion-Emotional/Passionate*) we use every description as independent query. This results in a total set of 139 test queries.

4.3 Evaluation Measures and Results

To measure the quality of the obtained rankings and the impact of the combination approaches, we calculate standard evaluation measures for retrieval systems, cf. [15]. Table 1 shows the results for the c35k collection (averaged over all 200 queries): The top row contains the baseline that has been empirically determined by repeated evaluation of random permutations of the collection. Not unexpectedly, the incorporation of additional tracks via the audio similarity measure leads to an increase in overall recall while precision is worsened. However, these global measures are not too important since for rankings one is in general more interested in how fast (i.e., at which position in the ranking) relevant results are returned. To this end, measures like *Precision @ 10 documents*, *r-Precision* (i.e., precision at the r^{th} returned document, where r is the number of tracks relevant to the query), and *(mean) average precision* (i.e., the arithmetic mean of precision values at all encountered relevant documents) give more insight into the quality of a ranking. For r-Precision and average precision we can clearly see that PAR (and also aRRS) perform better than text-based RRS. However, when comparing this to the pseudo-document indexing approach, we see that this simple and efficient ranking technique is in most cases even better than the combination with audio. [3] Thus, although audio similarity may improve results, it can not keep up to a well working text-based approach. Furthermore, we can see that incorporation of audio worsens results if recall of the initial ranking is already high (n=10000, PseudoDoc). The reason is that audio similarity introduces a lot of noise into the ranking. Hence, to preserve the good performance at the top of the rankings, α should be set to a high value. On the other hand, this prevents theoretically possible improvements. For the CAL500 set (Table 2), things look a bit different. Here, the aRRS approach performs clearly superior to RRS. Improvements can even be observed within the first ten documents. For this collection, also results of the PseudoDoc approach can be improved by applying post-hoc audio-based re-ranking. For comparison of different retrieval strategies, we calculated *precision at 11 standard recall levels*. For each query, precision $P(r_j)$ at the 11 standard recall levels $r_j, j \in \{0.0, 0.1, 0.2, ..., 1.0\}$ is interpolated according to $P(r_j) = max_{r_j \leq r \leq r_{j+1}} P(r)$. This allows averaging over all queries and results in character-

[3] Note that the *Web only* recall value of PseudoDoc represents the upper bound for all purely text-based approaches.

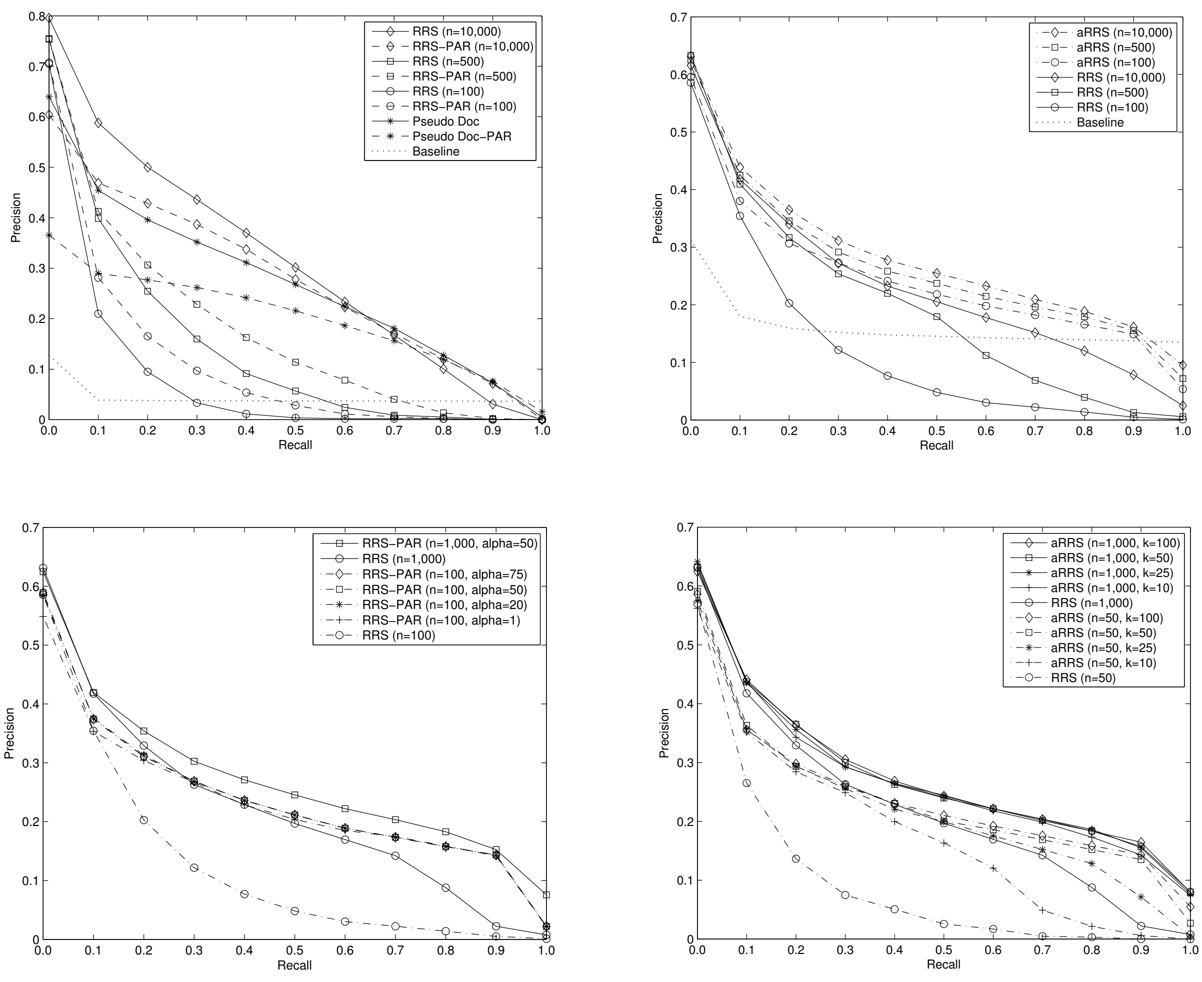


Figure 1. *Precision at 11 Standard Recall Levels* plots: The upper left plot depicts selected curves (averaged over all queries) from evaluating the c35k set for comparison of the RRS approach and subsequent PAR re-rankings. The upper right plot depicts (averaged) curves from the CAL500 set for comparison of the RRS and the aRRS approaches. The lower figures are intended to give an impression of the effects of different parameters for PAR (left) and aRRS (right). Both are calculated on the CAL500 set.

istic curves for each retrieval algorithm, enabling comparison of distinct algorithms/settings. Figure 1 depicts several precision at 11 standard recall level curves. The two plots at the top basically confirm what could be seen in tables 1 and 2. The two plots at the bottom show the influence of parameters α and k on the retrieval quality.

Using the CAL500 set, we can (rather informally) evaluate how audio similarity influences retrieval of tracks from the so-called "long tail". To this end, we restricted the set of relevant tracks for each query to contain only tracks from the (in general not so well known) online record label Magnatune. Absolute numbers resulting from this type of evaluation are rather discouraging, however, when comparing the results from RRS_{200} with those from $aRRS_{200}$ on this modified ground truth, a small improvement can be observed (e.g., MAP increases from 2.03 to 3.68, rPrec from 2.44 to 2.82). Optimistically spoken, a positive tendency is recognizable – from a more realistic perspective, both results are disappointing. In any case, the impact on long tail tracks needs a thorough investigation in future work.

5. DISCUSSION

We have shown that combining Web-based music indexing with audio similarity has the potential to improve retrieval performance. On the other side, we have also seen that even an improved combined retrieval approach may be outperformed by another, rather simple, text-only approach. Possible explanations are inadequate combination functions and/or an inadequate audio similarity measure. To estimate the potential of the audio similarity measure for this task, we examined the 100 nearest audio neighbors for every relevant track for a query and for every query, i.e., at each position $k = 1...100$, we calculated the precision (wrt. the currently examined query). Figure 2 shows the result averaged over all seed songs and queries for the c35k collection. Within the top 10 neighbors, a precision of around 7% can be expected in average based solely on the audio similarity. However, it is questionable whether this can be improved as audio similarity measures (statically) focus on specific musical properties, whereas textual

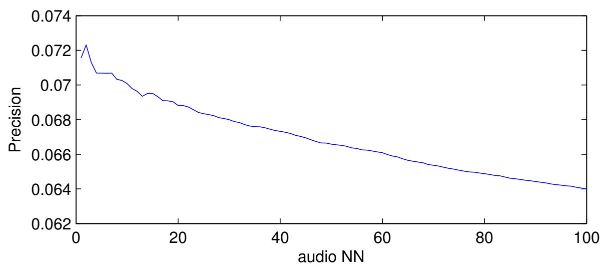


Figure 2. Precision at audio-based nearest neighbor for the c35k set (averaged over all queries; for every query average of rankings with each relevant track as seed).

queries can be aimed at basically every aspect of music, from different acoustic properties, to cultural context, to completely unrelated things.

In general it has to be stated that proper combination of these two sources is rather difficult since they target different directions and applications. Furthermore, a combination function can not be optimized in advance to suit every potential query, i.e., in contrast to, e.g., [5], automatic learning of proper combination functions (e.g., via machine learning methods) is not applicable for this task since we have no learning target. More precisely, Web-based music indexing as we currently apply it is an unsupervised approach. This is implied by the requirement to deal with a large and arbitrary vocabulary.

6. CONCLUSIONS AND FUTURE WORK

We proposed two methods to combine a Web-based music retrieval system with an audio similarity measure to improve overall ranking results and enable including tracks not present on the Internet into search results. Based on our evaluations, we could show that the overall ranking quality can be improved by integrating purely acoustic similarity information. However, we were also confronted with the current limitations of this combination. The first results gathered, open up new questions for future work, e.g., if another audio similarity measure could produce more substantial results. Also the question of combining the different sources will be taken a step further. Possible future enhancements could comprise clustering to find coherent groups of songs. This could be based on learning from many queries and finding stable relations between frequently co-occurring tracks. Another aspect that will be dealt with in future work is the impact on tracks from the long tail. Ideally, a combination would allow for retrieval of relevant tracks irrespective of their presence on the Web.

7. ACKNOWLEDGMENTS

This research is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project number L511-N15. Peter Knees would like to thank the staff of ChEckiT! - Verein Wiener Sozialprojekte (where he carried out his civilian national service) for tolerating and supporting the development of this publication.

8. REFERENCES

[1] S. Baumann, A. Klüter, and M. Norlien. Using natural language input and audio analysis for a human-oriented MIR system. *Proc. 2nd WEDELMUSIC*, 2002.

[2] O. Celma, P. Cano, and P. Herrera. Search Sounds: An audio crawler focused on weblogs. *Proc. 7th ISMIR*, 2006.

[3] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A Music Search Engine Built upon Audio-based and Web-based Similarity Measures. *Proc. 30th ACM SIGIR*, 2007.

[4] M. Sordo, C. Laurier, and O. Celma. Annotating music collections: How content-based similarity helps to propagate labels. *Proc. 8th ISMIR*, 2007.

[5] L. Barrington, D. Turnbull, M. Yazdani, and G. Lanckriet. Combining audio content and social context for semantic music discovery. *Proc. 32nd ACM SIGIR*, 2009.

[6] P. Knees, T. Pohle, M. Schedl, D. Schnitzer, and K. Seyerlehner. A Document-centered Approach to a Natural Language Music Search Engine. *Proc. 30th ECIR*, 2008.

[7] O. Gospodnetić and E. Hatcher. *Lucene in Action*. Manning, 2005.

[8] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. *Proc. 9th ISMIR*, 2008.

[9] T. Pohle and D. Schnitzer. Striving for an Improved Audio Similarity Measure. *4th MIREX*, 2007.

[10] M. Mandel and D. Ellis. Song-Level Features and Support Vector Machines for Music Classification. *Proc. 6th ISMIR*, 2005.

[11] E. Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, 2006.

[12] A. Flexer. A closer look on artist filters for musical genre classification. *Proc. 8th ISMIR*, 2007.

[13] C. Snoek, M. Worring, and A. Smeulders. Early versus late fusion in semantic video analysis. *Proc. 13th ACM Multimedia*, 2005.

[14] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, February 2008.

[15] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Reading, Massachusetts, 1999.

IMPROVING ACCURACY OF POLYPHONIC MUSIC-TO-SCORE ALIGNMENT

Bernhard Niedermayer
Department for Computational Perception
Johannes Kepler University Linz, Austria
bernhard.niedermayer@jku.at

ABSTRACT

This paper presents a new method to refine music-to-score alignments. The proposed system works offline in two passes, where in the first step a state-of-the art alignment based on chroma vectors and dynamic time warping is performed. In the second step a non-negative matrix factorization is calculated within a small search window around each predicted note onset, using pretrained tone models of only those pitches which are expected to be played within that window. Note onsets are then reset according to the pitch activation patterns yielded by the matrix factorization. In doing so, we are able to resolve individual notes within a chord. We show that this method is feasible of increasing the accuracy of aligned note's onsets which are already aligned relatively near to the real note attack. However it is so far not suitable for the detection and correction of outliers which are displaced by a large timespan. We also compared our system to a reference method showing that it outperforms bandpass filtering based onset detection in the refinement step.

1. INTRODUCTION

Opposed to blind audio analysis there are several applications where the recording of an already known piece of music has to be analysed. These applications range from computational musicology, especially performance analysis, and pedagogical systems to augmented audio players and editors as well as special query engines. Knowing that a huge number of symbolic transcriptions of classical as well as modern pieces are publicly available, this leads to the task of automatic music-to-score alignment.

Most current approaches are based on a local distance measure – mainly chroma vectors or features derived from chroma vectors – to compare the similarity between one time frame of the audio and one time frame of the score representation. These distances are then used by a global optimization algorithm, usually Dynamic Time Warping (DTW) or Hidden Markov Models (HMM), which finds the best matching alignment between these two feature sequences.

Recently much attention has been drawn on online algorithms for audio-to-score alignment, also known as score following, like described in [1]. However less work has focused on improvements of the accuracy of offline algorithms. In this paper we present ongoing work towards accurate measurement of individual notes' parameters. The calculation of accurate alignments is not only of use for the above mentioned applications but can also provide training and test data for less informed tasks like blind audio transcription [2].

We propose a two-pass system where in the first step a standard alignment routine based on chroma vectors and DTW is performed. In the second step this alignment is refined using a non-negative matrix factorization (NMF) approach. For each note a search window is set around the estimated note onset. With each of theses windows an NMF using pretrained tone models of only those notes excepted to occur within the respective audio segment plus a noise component is performed. In doing so, the system is able to resolve individual note onsets within whole chords.

We will show that this method provides a good means of refining the estimated onset times of notes that are relatively well detected by standard alignment. However in hard cases where the alignment deviates considerably from the ground truth the method shown here is prone to errors as well.

Section 2 is a brief overview of related work. In Sections 3 and 4 we explain the first alignment step and the NMF-based refinement respectively. Section 5 contains a description of the evaluation method used as well as the experimental results before we conclude our work in Section 6.

2. RELATED WORK

Much work, including [2–5], has focused on audio-to-score alignment based on acoustic features and Dynamic Time Warping (DTW). In [6] chroma vectors, Pitch Histograms, and two Mel-Frequency Cepstrum Coefficient (MFCC) related features have been compared in the context of DTW based audio matching and alignment. It was shown that chroma vectors perform significantly better than the other features.

Since DTW applied on two sequences of length n is of complexity $O(n^2)$ in time as well as in space the resolution of the features used is limited by runtime as well as memory constraints. One way of refining audio alignments is to increase this resolution while keeping computational costs within reasonable bounds. This is done by multi-scale approaches like described in [5] or [7] where the resolutions are increased iteratively but on the other hand search paths are constrained by tentative solutions found so far.

The resolution based refinement does not overcome an important side effect of alignments based on dynamic time warping. Notes that are struck together in the score, like it is the case for chords, can not be treated independently. This is a major drawback in applications like performance analysis, where the accurate timing of individual chord notes is an important expressive characteristic. [8] and [9] use pitch specific energy levels in order to estimate the timings of individual notes.

Another method to iteratively refine audio alignments is a bootstrap approach as described by [4]. There an audio segmenter is trained on an initial alignment. This segmenter can produce a refined alignment which is then used for a repeated training step. This method allows for the application of supervised machine learning techniques without the need for external training data.

Non-negative matrix factorization, as used here, was first applied to audio alignment in [10]. There, the combination of NMF and Hidden Markov Models was able to create alignments for polyphonic instruments in realtime.

3. BASIC ALIGNMENT

3.1 Chroma Feature

In the first pass the proposed system performs a state-of-the-art audio-to-midi alignment based on chroma vectors and Dynamic Time Warping. Chroma vectors have 12 elements representing the single pitch classes (i.e. C, C#, D, D#,...). The values are calculated based on a short time Fourier transform. Each frequency bin is then related to the index i of a pitch class by

$$i = \text{round}\left(12\log_2\left(\frac{f_k}{440}\right)\right) + 9 \mod 12 \qquad (1)$$

where f_k is the center frequency of the k^{th} bin. The tuning frequency is supposed to be 440 Hz but can easily be changed to any other value. The summand 9 shifts the vector such that the pitch class C has index 0. The individual values are then obtained by summing up the energies of all bins corresponding to a certain pitch class.

A similar feature that yields comparable results has been suggested by [11] which on the one hand takes only bins containing energy peaks into account but on the other hand also considers harmonics. At the extraction of the so called Harmonic Pitch Class Profile the energy of a frequency bin k does not only contribute to the pitch class best matching the center frequency f_k but also to those pitch classes best matching f_k/h with $h = 2, 3, 4 \ldots$ This accommodates for the assumption that the energy in bin k can also represent the h^{th} harmonic of a pitch. Since the energy of a partial decreases with the order of the harmonic, an additional weighting factor of $w_{harm} = d^{h-1}$ with $0 < d \leq 1$ is introduced.

The calculation of the chroma representation based on a MIDI file instead of audio data is straightforward since each MIDI event can be directly assigned to the corresponding pitch class. However when using the Harmonic Pitch Class Profile, errors are made when letting the energy of the actual f_0 contribute to the pitch classes corresponding to $f_0/3$, $f_0/5, \ldots$ This inexactness has to be reproduced in order to obtain equivalent representations of audio and score. Likewise when using default chroma vectors, contributions of a note to other pitch classes than the one corresponding to the f_0 caused by harmonics can be considered as well.

Preliminary experiments have shown that chroma vectors and Harmonic Pitch Class Profiles yield comparable results. Therefore chroma vectors have been used for the remainder of this work due to computational advantages.

3.2 Dynamic Time Warping

Based on this chroma representation a globally optimal alignment is calculated. Therefore a sequence of chroma vectors for the audio file as well as for the score representation is calculated. In doing so the score MIDI is divided into time frames such that the overall number of frames and the overlap ratio between frames is the same as of the STFT applied on the audio data. The Euclidean distance is used to compute a similarity matrix SM comparing each frame of one feature sequence to each frame of the other sequence, after all feature vectors have been normalized. Mapping corresponding frames to each other is the same as finding a minimal cost path through this similarity matrix. A path through SM_{ij} is then equivalent to the alignment of frame i of the score feature sequence to frame j of the performance feature sequence. Dynamic time warping (DTW) is a well-established dynamic programming based algorithm that finds such optimal paths. A detailed tutorial can be found in [12].

In order to get meaningful results an alignment path has to meet several constraints.

Continuity The constraint of continuity forces a path to proceed through adjacent cells within the similarity matrix. Jumps would be equal to skipping frames without considering the costs of this operation.

Monotonicity The constraint of monotonicity in both dimensions guarantees that the alignment has the same temporal order of events as the reference sequence.

End-point constraint The end-point constraint forces the ends of the path to be the diagonal corners of the similarity matrix. In doing so it is assured that the alignment covers the whole sequences.

The optimal path according to DTW is calculated in two steps. The forward step starts a partial path at the point

$[0, 0]$ and rates it with the cost SM_{ij}. Then it calculates the minimum path costs for all other partial alignments ending with frame i of the score being aligned to frame j of the recorded performance in a recursive manner according to equation 2.

$$Accu(i,j) = \min \begin{cases} Accu(i-1,j-1) + SM_{ij} * w_d \\ Accu(i-1,j) + SM_{ij} * w_s \\ Accu(i,j-1) + SM_{ij} * w_s \end{cases} \tag{2}$$

The three options correspond to partial paths ending with a diagonal step, an upwards step, and step to the right within the similarity matrix SM. In addition to the actual local distances, weights w_d and w_s are needed to yield reasonable path costs. If there were no such weights, diagonal paths would be strongly favored over straight ones which are twice as long. Experiments have shown that the values 1.4 and 1.0 (still giving diagonal steps a preference over straight ones) perform well. In our implementation we do this cost calculation in place, i.e. overwriting the values SM_{ij} by $Accu(i, j)$ in order to save memory space.

The backtracking step of DTW starts as soon as all values $Accu(i, j)$ have been calculated. $Accu(N-1, M-1)$ is the minimal cost of a complete alignment between the two feature sequences. Therefore the optimal path is reconstructed starting from $[N-1, M-1]$ going back to $[0, 0]$. In order to be able to do so, a second matrix is built during the forward step, memorizing whether the last step leading to a point $[i, j]$ was diagonal, upwards, or to the right.

4. NMF-BASED REFINEMENT

4.1 Non-negative Matrix Factorization

Within the last few years non-negative matrix factorization (NMF) has become of increasing interest in the domain of blind audio transcription. The basic idea is that an input matrix V of size $m \times n$ is decomposed into two output matrices W and H of size $m \times r$ and $r \times n$ respectively where the elements of all these matrices are strictly non-negative and

$$V \approx WH \tag{3}$$

Assuming that V represents real-world data such factorizations will most likely not be perfect. The reconstruction error caused by any deviation of WH from V can be measured by a cost for which the Euclidean distance or the I-divergence are common choices. In minimizing this cost function, W and H are learned as an initially determined number r of basis vectors and their activation patterns over time respectively.

Performing such a decomposition on a spectrogram, as obtained by a short time Fourier transform, will result in a dictionary W of weighted frequency groups and their occurrence H over time. According to the input V and the parameter r, the base components in W will, in the ideal case, represent models of single pitches or chords played on a certain instrument. But due to the unsupervised nature of the method, elements of W might as well correspond to special frequency patterns during the attack, sustain, or decay phase of a note, single partial or just noise.

However, as soon as the piece and its score are known, as it is the case in the context of audio alignment, the instrument(s) used to perform the piece are most probably known as well. So there is no need to learn a set of base components. Instead a number r of tone models can be trained in advance which overcomes the above mentioned uncertainty of unsupervised learning. Also the number and kind of tone models can be adjusted to the respective piece.

With only H being left unknown Equation 3 can be rewritten as

$$v \approx \overline{W} \cdot h \tag{4}$$

where $\overline{W}$ is the fixed dictionary of tone models. v and h are single column vectors of V and H that can now be processed independently, which leads to a much simpler decomposition task [13]. The vectors h are very sparse in nature and represent an f_0 estimation for the corresponding frame.

Throughout this work the mean square criterion given as

$$c_{err} = \frac{1}{2} \parallel \overline{W}h - v \parallel_2^2 \tag{5}$$

is used as cost measure for factorization errors since computationally efficient algorithms for its optimization are available [14].

4.2 Tone Model Training

In order to get meaningful factorizations at least one tone model per possible pitch has to be contained in $\overline{W}$. Given a set of training samples, such tone models can be trained in advance using the same method as described above. In the ideal case those training samples are audio recordings of single pitches played on a certain instrument. Starting from Equation 3 again, W and H become vectors w and h since there is only one basis component present ($r = 1$). h can further be approximated by the amplitude envelope, leaving only w to be unknown. The actual computation is then done by the same implementation as used during the performing step of the algorithm.

Throughout this work we use an additional basis component representing white noise. Experiments have shown that such a noise model significantly improves the alignment results.

4.3 Local Refinement

In the first stage of the proposed system a music-to-score alignment has already been performed. The advantage of this alignment is that it is globally optimized and very robust. However independent from all parameters that can be set, accuracy is limited by the fact, that such an alignment algorithm can never differentiate between notes that are struck together in the score.

To overcome this limitation and still preserve high robustness we define a search window of length l around the initially estimated onset time. Within this local context the refinement step tries to find the exact temporal position of each individual (chord-)note. The parameter l has been chosen to be 2 seconds since preliminary evaluation of the first alignment step has shown that only a marginal number of outliers deviates from the ground truth by more than a second.

For each such search window the contained notes and their pitches are determined in order to define the tonal context of the note under consideration. This information is used to build a dictionary $\overline{W}_{local}$ made up by tone models describing only those pitches that are present within the local context plus an additional (white) noise component. The resulting activation patterns H are smoothed using a median filter and used in order to extract following features for each time frame.

Activation energy Since activation patterns H are very sparse in nature (even when sparsity is not enforced), activation energies greater than zero are strong indicators for note positions.

Energy slopes The first derivative of the activation energy corresponds to energy changes. Positive slopes as they occur at note onsets are filtered by half wave rectification.

Relative energy slopes Since transients at note onsets are characterized by energy burst across the whole spectrum, other pitches – especially ones with shared harmonics – might show low activation energies during such phases as well. Therefore the increases in energy of the pitch under consideration in relation to the overall frame energy is also taken into account.

Experiments have shown that the maxima of the derivatives are good predictors for note attacks while the maximal activation energy itself has turned out to be less significant. Comparing the slope of the absolute energy to the one of the relative energy revealed a slight advantage of the relative energy derivative which was therefore chosen as onset detection criterion.

5. EXPERIMENTAL RESULTS

5.1 Evaluation Method

We limit our evaluation to classical piano music using a database consisting of the first movements of 11 Mozart sonatas played by a professional pianist. The performance was done on a computer monitored Bösendorfer SE290 grand piano, producing an automatic MIDI transcription of the exact ground truth of played notes as well as pedal events. Aligning a single movement instead of a whole sonata at a time is a valid simplification since individual movements are per default separate tracks on audio CDs. Nevertheless the overall performance time of this test set is still about one hour containing more than 30.000 notes.

The tone models used for the NMF-based refinement have been learned from single tones played on the same grand piano. Since such a recording was not available for each pitch, the missing models have been acquired by simple interpolation.

For evaluation purpose we calculated an alignment for each piece using the audio recording of the expressive performance and a mechanical score representation in MIDI format. We compared the resulting onset times to our given ground truth data and took the absolute displacement as evaluation criterion. This evaluation was done for the initial alignment step only as well as for the whole system including the refinement.

Initial alignments were done using a short time Fourier transform (STFT) with a window length of 4096 samples and a hop size of 441 samples, which corresponds to a time resolution of 100 frames per second. For the refinement step a search window of radius one second was used and the STFT hop size was reduced to 256 samples, resulting in time frames of a length of 5.8 ms.

First experiments with this setup have shown that although the calculation of the factorization base feature is narrowed down to a small search window as well as a small pitch range, it is still not as robust as expected. About 10% of the notes have not been detected by the factorization step and therefore left unchanged during refinement.

Concerning the remaining notes it turned out to be the best strategy to only modify those notes where the initial alignment position and the timing resulting from refinement are approximately consistent. This is the case for about half of the overall number of notes. In situations where these two onset candidates differ by more than 20 frames (i.e. 116 ms) a conflict is detected – although its resolution has been left to future work. One cause for such conflicts are repeated notes which cannot be handled by the simple detection mechanism as described above.

5.2 Evaluation Results

In Table 1 the limits of the quartiles as well as the 95^{th} percentile are given. Within the first three quartiles the refinement has improved results for each individual piece. However concerning notes that are displaced by more than 100 ms in the initial alignment tend to be displaced even further by the refinement step.

For most applications a transcription is good as soon as a human listener can not distinguish it from the original. This implies that in the context of music-to-score alignment a note can be counted as correctly aligned if its deviation from the ground truth is less than the just noticeable difference of the human perception. In an experimental environment, where listeners were asked to adjust the timing of one tone within a series, such that the inter-onset intervals became perfectly regular, this just noticeable difference was investigated [15]. It was found to be around 10 ms for notes shorter than 250 ms and about 5% of the note duration for longer ones.

Therefore an evaluation based on this criterion was done as well. In Table 2 the amount of notes with a time dis-

piece	# notes	duration	$25\% < x$		$50\% < x$		$75\% < x$		$95\% < x$	
			bas.	ref.	bas.	ref.	bas.	ref.	bas.	ref.
kv279-1	2803	4:55	7 ms	5 ms	16 ms	12 ms	30 ms	27 ms	103 ms	101 ms
kv280-1	2491	4:48	11 ms	5 ms	23 ms	14 ms	42 ms	34 ms	126 ms	127 ms
kv281-1	2648	4:29	12 ms	6 ms	24 ms	15 ms	42 ms	36 ms	114 ms	112 ms
kv282-1	1907	7:35	10 ms	6 ms	23 ms	15 ms	53 ms	44 ms	337 ms	380 ms
kv283-1	3304	5:22	7 ms	5 ms	15 ms	12 ms	27 ms	26 ms	62 ms	65 ms
kv284-1	3700	5:17	7 ms	6 ms	15 ms	13 ms	31 ms	29 ms	97 ms	98 ms
kv330-1	3160	6:14	7 ms	5 ms	15 ms	11 ms	28 ms	24 ms	118 ms	124 ms
kv332-1	3470	6:02	9 ms	7 ms	20 ms	18 ms	39 ms	37 ms	138 ms	147 ms
kv333-1	3774	6:44	8 ms	5 ms	16 ms	13 ms	29 ms	20 ms	79 ms	80 ms
kv457-1	2993	6:15	10 ms	6 ms	19 ms	15 ms	37 ms	35 ms	214 ms	257 ms
kv475-1	1284	4:58	13 ms	11 ms	30 ms	24 ms	78 ms	75 ms	360 ms	393 ms
all	31534	1:02:39	8.3 ms	5.6 ms	18 ms	14 ms	35 ms	32 ms	132 ms	137 ms

Table 1. Comparison between accuracy after the basic alignment step (bas.) and the additional refinement (ref.)

piece	$x < 10$ ms		$x < 50$ ms	
	bas.	ref.	bas.	ref.
kv279-1	33.8%	43.2%	88.2%	88.4%
kv280-1	22.4%	42.5%	81.5%	85.0%
kv281-1	20.1%	38.5%	80.4%	83.4%
kv282-1	25.3%	39.2%	73.7%	76.8%
kv283-1	36.2%	44.2%	92.6%	92.2%
kv284-1	34.6%	41.7%	86.9%	87.2%
kv330-1	35.5%	46.7%	89.9%	89.7%
kv332-1	27.1%	32.5%	83.0%	82.7%
kv333-1	31.5%	42.2%	90.1%	90.1%
kv457-1	27.3%	35.9%	82.5%	83.2%
kv475-1	20.0%	23.6%	63.9%	66.8%
all	29.6%	40.0%	84.8%	85.6%

Table 2. Comparison between accuracy after the basic alignment step (bas.) and the additional refinement (ref.)

placement less than 10 ms is shown for the initial and the refined alignment. According to the chosen STFT time resolution this corresponds to a deviation of one frame at maximum. In addition the number of notes having a displacement error less than 50 ms is given as well since this is a common evaluation criterion in onset detection.

Again it is shown that the refinement improves those notes already aligned relatively close to their real onset. The amount of notes with displacement errors less than 10 ms was increased from about 30% to 40% while the number of notes with errors below 50 ms was only moderately changed from 84.8% to 85.6%.

5.3 Feature comparison

From the list of related work presented in section 2, [8] is the one that presents the approach which is most similar to the system proposed here. There onset detection by selective bandpass filtering is described in the context of score supported audio transcription. According to this method a note is found by summing up the energy in all frequency bands corresponding to the f_0 as well as the harmonics of a pitch and then finding a maximum in the derivative of this indication function. In order to avoid the influence of other pitches with overlapping harmonics, partials that collide with those of an other note struck at the same time are neglected.

	fact.	s.b.f.
$25\% < x$	5.6 ms	10.0 ms
$50\% < x$	14 ms	20 ms
$75\% < x$	32 ms	40 ms
$95\% < x$	137 ms	128 ms
$x < 10$ ms	40.0%	24.9%
$x < 50$ ms	85.6%	81.3%

Table 3. Comparison between refinement based on factorization (fact.) and based on selective bandpass filtering (s.b.f.) [8]

We have compared our system to an own implementation of this approach. In doing so, we used the same computational framework and only exchanged the factorization feature in the refinement step by this onset detector based on selective bandpass filtering. The accumulated results on the whole test set are shown in Table 3. It demonstrates that bandpass filtering yields results less accurate than those produced by NMF, and mostly even less accurate than those achieved by the alignment based on chroma vectors. A possible reason is that the STFT based version of selective bandpass filtering relies on just a few frequency bins while NMF takes the whole spectrogram into account.

6. CONCLUSION AND FUTURE WORK

We have introduced a new method to increase accuracy of music-to-score alignments by a two-pass system. Whereas the first step consists of a state-of-the-art alignment using chroma features and dynamic time warping the second step is a refinement based on non-negative matrix factorization.

We have shown that this refinement step performs very well on notes which have already been detected relatively close to their real onset time by the alignment step. The number of notes placed with a time deviation below the just noticeable difference according to [15] of 10 ms has been increased from about 30% to 40%. This is remarkable since so far only those notes without any conflicting features have been modified.

However the method does not bring any improvements for notes where the deviation of the initial alignment from the ground truth is large. On one hand the refinement step only works within a search window which should be kept as small as possible. Notes that are misaligned such that the actual onset is out of this window can never be corrected by the method described here. On the other hand chroma features as well as factorization based pitch separation rely on prominent energy peaks in the spectrogram. If the spectrogram is blurred due to heavy use of pedal or very rich polyphony both approaches are prone to errors.

This clearly dictates future work to concentrate on the problem of detecting and handling possible outliers and 'hard' regions. The most obvious approach is to develop a method of handling conflicting features as this is the case for about 40% of all notes. We think that introducing a tempo model and enforcing reasonable inter-onset intervals entails the potential of further improvements.

Also the 10% of notes that have not been covered by the factorization based feature are worth being reconsidered. Standard STFT favors the detection of higher pitches due to its linear frequency scale. Additional spectral transformations like multi-rate filterbanks or a constant-Q transform could help to enhance the note detection, especially within low pitch ranges.

7. ACKNOWLEDGMENTS

The research presented in this paper is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project number P19349-N15.

8. REFERENCES

[1] S. Dixon: "Live Tracking of Musical Performances Using On-Line Time Warping", *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx)*, Madrid, 2005.

[2] R. J. Turetsky and D. P. W. Ellis: "Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses", *Proceedings of the 4th International Symposium of Music Information Retrieval (ISMIR)* Baltimore, MD, 2003.

[3] Y. Meron and K. Hirose: "Automatic alignment of a musical score to performed music", *Acoustical Science and Technology*, Vol. 22, No. 3, pp. 189–198, 2001.

[4] N. Hu and R. B. Dannenberg: "A Bootstrap Method for Training an Accurate Audio Segmenter", *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, 2005.

[5] M. Müller, F. Kurth, and T. Röder: "Towards an efficient algorithm for automatic score-to-audio synchronization", *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, 2006.

[6] N. Hu, R. B. Dannenberg, and G. Tzanetakis: "Polyphonic Audio Matching and Alignment for Music Retrieval", *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, 2003.

[7] N. Adams, D. Marquez, and G. Wakefield : "Iterative Deepening for Melody Alignment and Retrieval", *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, 2005).

[8] E. D. Scheirer: "Using Musical Knowledge to Extract Expressive Performance Information from Audio Recordings", *Readings in Computational Auditory Scene Analysis*, H. G. Okuno and D. F. Rosenthal (eds.), Lawrence Erlbaum Publication, Mahweh, NJ, 1997.

[9] M. Müller, F. Kurth, and M. Clausen: "Audio Matching via Chroma-based Statistical Features", *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, 2005.

[10] A. Cont: "Realtime Audio to Score Alignment for Polyphonic Music Istruments Using Sparse Non-negative Constraints and Hierarchical HMMs", *Proceedings of the IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP)*, Toulouse, 2006.

[11] E. Gómez and P. Herrera: "Automatic Extraction of Tonal Metadata from Polyphonic Audio Recordings", *Proceedings of 25th International AES Conference*, London, 2004.

[12] Rabiner, L. R. and Juang, B.-H. "Fundamentals of speech recognition". Prentice Hall, Englewood Cliffs, NJ, 1993.

[13] F. Sha and L. Saul: "Real-time pitch determination of one or more voices by nonnegative matrix factorization", *Advances in Neural Information Processing Systems 17*, K. Saul, Y. Weiss, and L. Bottou (eds.), MIT Press, Cambridge, MA, 2005.

[14] Lawson, C. L. and Hanson, R. J. "Solving least squares problems", *Prentice Hall*, Lebanon, Indiana, 1974.

[15] A. Friberg and J. Sundberg: "Perception of just noticeable time displacement of a tone presented in a metrical sequence at different tempos", *Proceedings of the Stockholm Music Acoustics Conference*, pp. 39–43, Stockholm, 1993.

FORMALIZING INVARIANCES FOR CONTENT-BASED MUSIC RETRIEVAL

Kjell Lemström
Department of Computer Science
University of Helsinki

Geraint A. Wiggins
Department of Computing
Goldsmiths, University of London

ABSTRACT

Invariances are central concepts in content-based music retrieval. Musical representations and similarity measures are designed to capture musically relevant invariances, such as transposition invariance. Though regularly used, their explicit definition is usually omitted because of the heavy formalism required. The lack of explicit definition, however, can result in misuse or misunderstanding of the terms.

We discuss the musical relevance of various musical invariances and develop a set-theoretic formalism, for defining and classifying them. Using it, we define the most common invariances, and give a taxonomy which they inhabit. The taxonomy serves as a useful tool for idetinfying where work is needed to address real world problems in content-based music retrieval.

1. INTRODUCTION

To effectively perform content-based music retrieval (CBMR), the intrinsic features of music must be taken into account. Some of the most important features correspond directly with invariances. Invariances related to pitch, tempo and duration are widely used, but usually without proper definition or discussion of their inter-relationship. Indeed, a single term is sometimes used to name multiple phenomena, admitting confusion about its real meaning.

Western musical scales may be transformed, or *transposed*, to any other key so that the corresponding pitch intervals remain intact. Indeed, Western people tend to listen to music analytically, observing pitch intervals rather than absolute pitch values. Thus, musical works are identified regardless of the prevalent musical key. The same observation is valid for tempo: two pieces of music are considered the same if the other is just played slower than the other (i.e., a different time scale is used). So transposition and time-scale invariance are important in CBMR applications.

However, in some cases mere transposition and time-scale invariance are not enough. For example, in query by humming, untrained singers often cannot produce pitch intervals accurately enough to constitute a match. To address this, several pitch class generalizations have been suggested, such as pitch contour [9] and qpi classification [4]. Using these generalizations, only direction of interval (contour) or the order of magnitude of interval (small, medium or large) is observed, respectively.

In this paper, we will define what it means when a representation or a method (algorithm) is invariant under a given notion arising from a musical phenomenon. We will give definitions for widely used invariances related to three main dimensions of music: *pitch*, *onset time* and *duration*. The latter two are temporal features and, usually, the third is derivable from the second. However, it is sometimes useful to separate them since the invariances as applied, categorised by our taxonomy, may differ. We will also define a set of more abstract, structural invariances. All of these inhabit a taxonomy that shows the relationships between the invariances, and also serves as a tool for identifying areas where further work in CBMR is needed.

2. DEFINING THE INVARIANCES

2.1 The representation

Let us start by defining the notion of a *representation*. In this context, we are modelling an observed phenomenon (music perception), and it is important not to presuppose that the data *is* the phenomenon; therefore, making the representation explicit is important too.

Let the size of a set S be denoted by $|S|$. Let the set of ordered subsets of set S of size between n and m, inclusive, be denoted by $S^{n\dots m}$, and where $n = m$, S^n; S^* is the power set of S. Given a set of features, $f_i \in F$, each with a unique type, $\tau_i \in \tau$, identified by an injection $T : F \mapsto \tau$, an *abstract representation*, ρ, is a subset of F. The type of each feature should be a mathematical specification (e.g., linear Abelian group for pitch) which is chosen to model the corresponding reality appropriately [13]. Given an abstract representation, ρ, a *concrete representation*, r, is a set of tuples

$$\{\langle f, \Sigma_f, \succ_f, \Phi_f, \Pi_f\rangle \mid f \in \rho\}$$

where Σ_f is an alphabet adequate to express f, $\succ_f$ is a partial order on Σ_f, Φ_f and Π_f are sets of functions and predicates, respectively, which apply to members of Σ_f defining the operations and tests required for the algebra of $T(f)$. Wiggins et al. [13] give detailed examples of datatypes for

	Feature invariances			Structural invariances
	Pitch	Onset Time	Duration	
Weaker/more specific	transposition (2)			ω-permutation (11/1)
$\downarrow$	pitch-transposition (3)	time-position (6)		strongly permutation (11/2)
$\downarrow$	pitch-warp (4)	time-scale (7)	time-scale (7)	ω-concatenation (12/1)
Stronger/less specific	Parsons (5)	time-warp (9)	duration-warp (8)	strongly concatenation (12/2)

Table 1. A sparse taxonomy on considered invariances. An invariance in the table subsumes the invariances above it, if no horizontal line appears in between. The number in parenthesis is that of the associated definition in Sections 2.4 and 3.

pitch and time. In general, $\succ_f$ is needed for the working of our formalism, not for the representation itself (there would be a member of Π_f for this, where appropriate); it is kept separate so that it may be different from any orders that are internal to the feature implementation, if necessary.

Let $\hat{}$ be a function which maps a concrete representation to its corresponding abstract representation.

Given a concrete representation, r, let an element e, $e \in r$, be a set of values, e_i, with concrete datatypes corresponding with r.

Let a *dataset*, E, be a set of elements. E is in r iff each e_i in E is in r.

2.2 A concatenator

To define invariances, we use a *concatenator* constructor.

A *concatenator*, $C^{r'}_{\omega}(E)$, constructs a lexicographically ordered multiset[1] of elements from a dataset, E, represented in r. The lexicographical order is specified by the ordered set $\omega \in \hat{r}^{1\cdots|r|}$ and the $\succ_i$ of the members of r corresponding with the members of ω. The superscript of the concatenator $r' \subseteq r$, gives the dimensions to be displayed.

For example, given a dataset, E, in an (abstract) representation including $\{pitch, onset, duration\}$ features, the concatenator $C^{\{pitch\}}_{\{onset\}}(E)$ creates a set of pitches ordered by onset time; one might use it to extract the pitches in a monophonic melody. If we generalise this to arbitrary features and combinations thereof, and consider only sequences including the first note of a piece, we arrive at the *viewpoint* representation of Conklin and Witten [3].

Evidently, the projective properties of this operator account for representational invariances where the invariant feature is an explicit feature in the representation, or a combination thereof. We use the term *capture* to denote this capacity: so projection to subsets of the existing feature set *captures* this kind of invariance.

For notational convenience we write operations applied to each member of an ordered set in order as operations on the set itself, where this is unambiguous, so, where A is a set of values and e is a value, $A \cdot e = \{a \cdot e \mid a \in A\}$; similarly, the elements of two sets of the same size, A, B may be combined pairwise in order under $\cdot$: $A \cdot B = \{a \cdot b \mid a_i \in A, b_i \in B\}$. Finally, to combine a value, $v \in \Sigma_f$, under an operation, $\cdot$, with one feature, f, of an element e, leaving other features unchanged, we write $e \cdot_f v$, so $e +_{pitch} k$ adds k to the $pitch$ feature of e.

In order neatly to specify a particular kind of derived invariance, we use $\langle S\rangle^{\cdot}_{f}$, where S is an ordered set, to denote the ordered set produced by ordered, pairwise operation on the feature f of elements $s_i \in S$ under $\cdot$. So, $\langle S\rangle^{-}_{f} = \{s_{i+1} -_f s_i \mid 1 \leq i < |S|\}$. This operation has consequences for the representation of the result: each feature type must be replaced by a derived type (corresponding with predefined ones where appropriate). For our concerns here, $pitch$ is replaced by $interval$, and $onset$ is replaced by ioi (inter-onset-interval), in the obvious way. We will need also a second-order derived invariance to be used with onsets (arriving at ioi proportions), thus: $\langle\langle S\rangle^{-}_{onset}\rangle^{\div}_{ioi} = \{\frac{\langle s_{i+i}\rangle^{-}_{onset}}{\langle s_i\rangle^{-}_{onset}} \mid 1 \leq i < |S| - 1\}$.

2.3 Representational invariance

Some invariances can be captured by a change of representation. Whether or not this is possible depends on the representation used and on the nature of the phenomenon modelled. In many cases, a change of representation like this can usefully be thought of as indexing, and so it is helpful to know what remains invariant.

For example, because $pitch$ can be modelled by an Abelian group, it follows that for any set of pitches, E, thus modelled, there is another set formed by combining a constant member of Σ_{pitch} under the *plus* function in Φ_{pitch} with each member of E (the members of Σ_{pitch} are by definition in one-to-one correspondence with a partition of $\Sigma_{interval}$). It is implicit in the specification of the abstract representation that this operation, which is mathematically *translation*, models *musical pitch transposition*. Reversing this argment, it follows that any sequence of $pitch$es can be expressed as a sequence of $pitch$ differences, or *intervals*. Now, again because of the mathematical properties of the representation, it happens that each such interval is represented in Σ_{pitch}, and the algebra defined by Φ_{pitch} models the additive behaviour of intervals too: they also form an Abelian group. Thus, it is possible to produce a transposition-invariant version of any dataset, E, in any representation which contains $pitch$ and $onset$ information, by computing the ordered set whose members are computed by calculating $\langle C^{\{pitch\}}_{\{onset,pitch\}}(E)\rangle^{-}_{p}$. If the music modelled by E is monophonic, then this is the familiar interval sequence representation; however, if the music is not monophonic, care must be taken, because the *relative*

[1] This is a multiset because it is possible for the concatenator to map more than one element of E to any given element in the resulting representation; it may be necessary to know that this has happened.

nature of this representation makes its values dependent on their position in the sequence generated by the concatenator; therefore, one cannot apply many of the operations one would like. This fact is well-known to representers of music: an interval-based representation is not readily amenable to the representation of non-monophonic music. However, in this change of representation, relatively little information is lost: just one constant value, which tells us on what pitch the original dataset started; given that information, the entire original E may be reconstructed. In this sense, we say that the change is *structurally conservative*. However, though useful in itself, this property is neither necessary nor sufficient for a transformation to be useful.

For example, a familiar invariance transformation is that based on perceptual octave-equivalence, used in computing a chromagram. Here, perception maps exactly on to the mathematics, and so perceptual octave equivalence can be modelled by a chromatic equality function, defined as equality modulo n, where n is the number of divisions of the octave being used in the underlying scale of the pitch system. Here, Σ_{chroma} can very usefully be a contiguous subset of Σ_{pitch}, so $\mathbb{Z}_{12}$ does very nicely, and Φ_{chroma} and Π_{chroma} are equally easily defined. However, this representation change is also not, in general, structurally conservative, and it is mathematically evident why: the mapping from $\mathbb{Z}$ to $\mathbb{Z}_{12}$ is many-to-one, and so information is lost. The same principle, with a mapping to $\mathbb{Z}_8$, gives scale-degree representation, which is also octave-invariant.

A more interesting example is contour, an important aspect of melodic memory [8, §2.3]; Parsons coding [9] is a common way to represent the contour of music. However, $\Sigma_{Parsons} = \{-, 0, +\}$; it is not possible to give a fully defined $plus$ function over this set, while maintaining it as a model of musical contour, for obvious reasons. Therefore, we confirm that information is lost in changing to a representation whose pitch is based on Parsons coding, and one can argue this in advance because the abstract type of the Parsons code is not as expressive as a linear Abelian group. Thus, change of representation to Parsons code from, say, MIDI, is not structurally conservative. The same applies to comparable but more detailed interval-based representations such as the *qpi alphabet* [4].

Parsons coding captures an invariance which is *stronger* than transposition invariance in the sense that the equivalence classes it creates are fewer and larger. We will define two such invariances. In these, contour is preserved, but interval size is not—formal specifications are given in Definitions 4 and 5. Transposition from major to parallel minor is a (rather cautious) example of pitch warping; so, more generally, are interval augmentation and diminution in contrapuntal theory, or expansion and contraction in the music of Bartók. We note that among the passages captured by pitch warping lie also the equivalent transpositions, and this confirms that the stronger pitch-warp invariance is a indeed generalisation of transposition invariance. Therefore, a content-based music retrieval technique using Parsons coding can be seen as a filtering technique for finding transposed occurrences of a query (only filtering and not identifying, because false positives will be generated).

Our remaining common musical features, onset time and note duration, and the corresponding invariances (see Table 1) can be dealt with in the obvious way using the concatenator. For instance, given two datasets, $\mathcal{B}$ and $\mathcal{B}'$ in the same representation [2], two ioi sequences produced by the appropriate concatenator are time-scaled versions of each other if there is a number [3] d such that

$$C^{\{ioi\}}_{\{onset\}}(\mathcal{B}) = C^{\{ioi\}}_{\{onset\}}(\mathcal{B}') \times_{ioi} d.$$

A similar observation to that above, that time-warp invariance is stronger than time-scale invariance, applies here.

2.4 Algorithmic invariance

Music comparison is usually carried out in practice by an algorithm using a distance measure. Like representations, measures can be invariant under some property. At this level, we speak about *algorithmic invariances*. The following partial definition is a necessary but not sufficient condition to that end; it will be completed below.

Definition 1 *Let $\mathcal{M}$ be a CBMR method and P a property of a finite space* [4]*, where $|P|$ is the size of the space under consideration. $\mathcal{M}$ is* algorithmically P-invariant *, if working on datasets in representations in which the underlying datatype(s), explicit or implicit, of P does not introduce a factor into the computational complexity of $\mathcal{M}$ that is dependent from $|P|$.*

This definition rules out invariances achieved by discretizing a search space, enumerating it, and then searching exhaustively. Although such methods are sometimes called P-invariant in the MIR literature, this is really not the case; they are methods that merely appear to take advantage of invariance via brute-force calculation.

2.4.1 Pitch invariances

We now define the invariances in our taxonomy (Table 1), starting with pitch. Recall that our sets are by default *ordered* multisets. We omit duration, which is derivable from ioi, and abbreviate $\{pitch, interval, onset, ioi\}$ to $\{p, i, o, ioi\}$, respectively.

Definition 2 *Let r be a representation including $pitch$ and $onset$. A distance function $\mathcal{D}$ is* transposition-invariant *iff*

$$\forall a, b \in \Sigma_p . \forall \mathcal{A}, \mathcal{B}\ in\ r . \mathcal{D}(C^{\hat{r}}_{\{o\}}(\mathcal{A}), C^{\hat{r}}_{\{o\}}(\mathcal{B})) = \mathcal{D}(C^{\hat{r}}_{\{o\}}(\mathcal{A}) +_p a, C^{\hat{r}}_{\{o\}}(\mathcal{B}) +_p b).$$

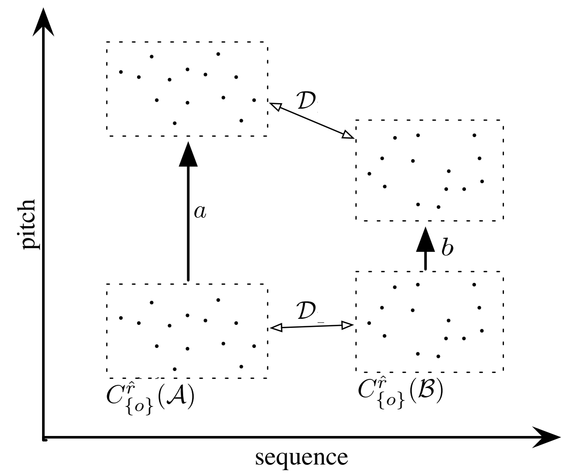
It may be helpful to visualise Definition 2, as in Fig. 1. In this example, $\hat{r} = \{p, o\}$.

Note that Definition 2 captures the exact transposition invariance that a music theorist would expect of that property. At times, however, it is useful to have a more relaxed

[2] This restriction is not mathematically necessary, but to admit comparison between representations here would over-complicate the example.

[3] What kind of number depends on the kind of time representation: a metrical one would use $\mathbb{Z}$ or $\mathbb{Q}$; a real-time one might use $\mathbb{R}$.

[4] It may have been derived by quantizing a continuous space P'.



pitch
sequence
a
b
$\mathcal{D}$
$\mathcal{D}$
$C^{\hat{r}}_{\{o\}}(\mathcal{A})$
$C^{\hat{r}}_{\{o\}}(\mathcal{B})$

Figure 1. Visualisation of Definition 2. $\hat{r} = \{p, o\}$.

version of transposition invariance. Indeed, the following *pitch-transposition invariance*, which omits the exact onset times, is often used in music retrieval applications.

Definition 3 *Let r be a representation including pitch and onset. A distance function $\mathcal{D}$ is* pitch-transposition-invariant *iff*

$$\forall a, b \in \Sigma_p . \forall \mathcal{A}, \mathcal{B} \; in \; r . \mathcal{D}(C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{A}), C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{B})) = \mathcal{D}(C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{A}) +_p a, C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{B}) +_p b).$$

Stronger kinds of pitch invariance than the above (as defined in Section 2.3) are defined as follows.

Definition 4 *Let r be a representation including pitch and onset. A distance function $\mathcal{D}$ is* pitch-warp-invariant *iff*

$$\forall K_{\mathcal{A}} \in \mathcal{N}^{|\mathcal{A}|-1} . \forall K_{\mathcal{B}} \in \mathcal{N}^{|\mathcal{B}|-1} . \forall \mathcal{A}, \mathcal{B} \; in \; r . \mathcal{D}\left(\langle C^{\hat{r}}_{\{o\}}(\mathcal{A})\rangle^{-}_{p}, \langle C^{\hat{r}}_{\{o\}}(\mathcal{B})\rangle^{-}_{p}\right) = \mathcal{D}\left(\langle C^{\hat{r}}_{\{o\}}(\mathcal{A})\rangle^{-}_{p} \times_i K_{\mathcal{A}}, \langle C^{\hat{r}}_{\{o\}}(\mathcal{B})\rangle^{-}_{p} \times_i K_{\mathcal{B}}\right)$$

where $\mathcal{N}$ is one of $\mathbb{Z}^+, \mathbb{Q}^+, \mathbb{R}^+$.

Note that the multiplication operation here needs to be duly definable in terms of functions in Φ_i. If we omit the onset information of that above, we get Parsons invariance:

Definition 5 *Let r be a representation including pitch and onset. A distance function $\mathcal{D}$ is* Parsons-invariant *iff*

$$\forall K_{\mathcal{A}} \in \mathcal{N}^{|\mathcal{A}|-1} . \forall K_{\mathcal{B}} \in \mathcal{N}^{|\mathcal{B}|-1} . \forall \mathcal{A}, \mathcal{B} \; in \; r . \mathcal{D}\left(\langle C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{A})\rangle^{-}_{p}, \langle C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{B})\rangle^{-}_{p}\right) = \mathcal{D}\left(\langle C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{A})\rangle^{-}_{p} \times_i K_{\mathcal{A}}, \langle C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{B})\rangle^{-}_{p} \times_i K_{\mathcal{B}}\right)$$

where $\mathcal{N}$ is one of $\mathbb{Z}^+, \mathbb{Q}^+, \mathbb{R}^+$.

2.4.2 *Temporal invariances.*

We now move to temporal invariances. The first allows for linear time shifts. So, for instance, in musical pattern matching, the pattern may occur anywhere in the database, not just as an incipit. Being additive, it is usually easily combined with the first pitch invariances, above.

Definition 6 *Let r be a representation including pitch and onset. A distance function $\mathcal{D}$ is* time-position-invariant *iff*

$$\forall a, b \in \Sigma_o . \forall \mathcal{A}, \mathcal{B} \; in \; r . \mathcal{D}(C^{\hat{r}}_{\{o\}}(\mathcal{A}), C^{\hat{r}}_{\{o\}}(\mathcal{B})) = \mathcal{D}(C^{\hat{r}}_{\{o\}}(\mathcal{A}) +_o a, C^{\hat{r}}_{\{o\}}(\mathcal{B}) +_o b).$$

Note that the above invariance is not meaningful with durations. The next two temporal invariances are of multiplicative nature, the first of which, the time-scale-invariance, is applicable both with onsets and durations.

Definition 7 *Let r be a representation including pitch and onset. A distance function $\mathcal{D}$ is* time-scale-invariant *iff*

$$\forall F_{\mathcal{A}} \in \mathcal{N}^{|\mathcal{A}|}, F_{\mathcal{B}} \in \mathcal{N}^{|\mathcal{B}|}, K_{\mathcal{A}} \in \Sigma_o^{|\mathcal{A}|}, K_{\mathcal{B}} \in \Sigma_o^{|\mathcal{B}|} . \forall \mathcal{A}, \mathcal{B} \; in \; r . \mathcal{D}(C^{\hat{r}}_{\{o\}}(\mathcal{A}), C^{\hat{r}}_{\{o\}}(\mathcal{B})) = \mathcal{D}(C^{\hat{r}}_{\{o\}}(\mathcal{A}) \times_o F_{\mathcal{A}} +_o K_{\mathcal{A}}, C^{\hat{r}}_{\{o\}}(\mathcal{B}) \times_o F_{\mathcal{B}} +_o K_{\mathcal{B}}).$$

where $\mathcal{N}$ is one of $\mathbb{Z}^+, \mathbb{Q}^+, \mathbb{R}^+$.

The next *duration-warp* invariance is most useful with duration sequences; it is "durational Parsons invariance", i.e., the one for which "shorter, longer, same" encoding is often used. To this end we use the second order derivation of sets $\mathcal{A}$ and $\mathcal{B}$ with ioi proportions, abbreviated ip below.

Definition 8 *Let r be a representation including pitch and onset. A distance function $\mathcal{D}$ is* duration-warp-invariant *iff*

$$\forall K_{\mathcal{A}} \in \mathcal{N}^{|\mathcal{A}|-2}, K_{\mathcal{B}} \in \mathcal{N}^{|\mathcal{B}|-2} . \forall \mathcal{A}, \mathcal{B} \; in \; r . \mathcal{D}\left(\langle\langle C^{\hat{r}}_{\{o\}}(\mathcal{A})\rangle^{-}_{o}\rangle^{\div}_{ioi}, \langle\langle C^{\hat{r}}_{\{o\}}(\mathcal{B})\rangle^{-}_{o}\rangle^{\div}_{ioi}\right) = \mathcal{D}\left(\langle\langle C^{\hat{r}}_{\{o\}}(\mathcal{A})\rangle^{-}_{o}\rangle^{\div}_{ioi} \wedge_{ip} K_{\mathcal{A}}, \langle\langle C^{\hat{r}}_{\{o\}}(\mathcal{B})\rangle^{-}_{o}\rangle^{\div}_{ioi} \wedge_{ip} K_{\mathcal{B}}\right)$$

where $\wedge$ is the power operator and $\mathcal{N}$ is one of $\mathbb{Z}^+, \mathbb{Q}^+, \mathbb{R}^+$.

The last temporal invariance does not bother with the onset information, except in as far as order is preserved. This is the case, for instance, with CBMR methods based on string representations that omit explicit onset times. Note that, although it is temporal, there is no intuitive interpretation of this invariance to duration information.

Definition 9 *Let r be a representation including pitch and onset, and let $K_{\mathcal{A}} \in \mathcal{N}^{|\mathcal{A}|}, K_{\mathcal{B}} \in \mathcal{N}^{|\mathcal{B}|}$ be such that*

$$\begin{aligned} a_{i-1} +_o K_{\mathcal{A}}(i-1) &< a_i +_o K_{\mathcal{A}}(i) \; and \\ b_{i-1} +_o K_{\mathcal{B}}(i-1) &< b_i +_o K_{\mathcal{B}}(i) \end{aligned}$$

for $2 \leq i \leq |K_{\mathcal{A}}|, |K_{\mathcal{B}}|$. *A distance function $\mathcal{D}$ is* time-warp-invariant *iff*

$$\forall \mathcal{A}, \mathcal{B} \; in \; r . \; \mathcal{D}\left(C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{A}), C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{B})\right) = \mathcal{D}\left(C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{A}) +_o K_{\mathcal{A}}, C^{\hat{r}\setminus\{o\}}_{\{o\}}(\mathcal{B}) +_o K_{\mathcal{B}}\right)$$

where $\mathcal{N}$ is one of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$.

Now, we can fully define algorithmic invariance.

Definition 10 *A method $\mathcal{M}$ is* algorithmically P-invariant *iff $\mathcal{M}$ satisfies Definition 1 and its similarity measure satisfies the definitions above corresponding with property P.*

3. STRUCTURAL INVARIANCES

Let us now consider a set of stronger invariances that relate primarily not to the music represented, but to the results proven using our order-based formalism. To be maximally useful, it is helpful to know how strongly the results apply: in particular, does the order imposed by our concatenator make a difference to the outcome? For example, in the following *permutation invariances*, when applied to contour-based melody comparison, onset-order matters, but in a pitch-class-distribution comparison, it probably does not.

Definition 11 *Let r be a representation and $\omega \subseteq \hat{r}$. A distance function $\mathcal{D}$ is ω*-permutation-invariant *iff*

$$\forall \mathcal{A}, \mathcal{B}\ in\ r.\mathcal{D}(C^{\hat{r}}_{\omega}(\mathcal{A}), C^{\hat{r}}_{\omega}(\mathcal{B})) = \mathcal{D}(\mathcal{P}(C^{\hat{r}}_{\omega}(\mathcal{A})), \mathcal{P}(C^{\hat{r}}_{\omega}(\mathcal{B})))$$

where $\mathcal{P}$ is any size-preserving permutation operator on ω. If $\omega = \hat{r}$, the distance function is strongly permutation-invariant.

Further, it may be useful to know that a distance is preserved no matter which dimension is used for ordering.

Definition 12 *Let r be a representation. A distance function $\mathcal{D}$ is ω-* concatenation-invariant *iff*

$$\forall \omega_1, \omega_2 \subset \hat{r}.\forall \mathcal{A}, \mathcal{B}\ in\ r.\mathcal{D}(C^{\hat{r}}_{\omega_1}(\mathcal{A}), C^{\hat{r}}_{\omega_1}(\mathcal{B})) = \mathcal{D}(C^{\hat{r}}_{\omega_2}(\mathcal{A}), C^{\hat{r}}_{\omega_2}(\mathcal{B})).$$

If $\omega_1, \omega_2 = \hat{r}$, the distance function is strongly concatenation-invariant.

For a strongly concatenation-invariant distance function the ordering does not make any difference at all. Note that a strongly permutation invariant distance function is also a strongly concatenation invariant, and vice versa.

4. INVARIANCES IN POLYPHONIC CONTENT-BASED MUSIC RETRIEVAL

4.1 Representations of non-monophonic music

The concatenated representations used here are evidently directly applicable when dealing with monophonic music. In the case of (discretely represented) polyphonic music, a geometrical representation [1, 11, 12, 14] is a more effective and natural choice [5]. An example of geometrical music matching (under transpositional equivalence, in this example) is given in Figure 2, where the common pitch-against time-representation, giving the onset times but not durations, is used. Several possible ways to represent durations have been suggested [10, 11, 12].

As Figure 2 suggests, the maximal subset match of the given query pattern of length m within the database of length n can be found by observing the translation vectors. Note that a translation corresponds to two musically distinct phenomena: a vertical move corresponds to pitch-shift while a horizontal move corresponds to aligning the pattern time-wise; the combination of these is what a musician calls "a transposition" (to be distinguished from the *process* of transposition, performed during performance).

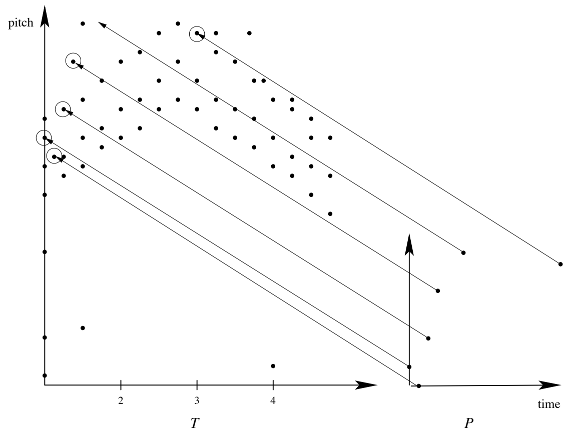


Figure 2. Pointset P, to the right, represents a pointset (musical) pattern to be matched against a pointset database to the left. The arrows represent translation vectors, from pattern to database, that give maximal occurrence.

Thus, working on the translation vectors captures transposition and position invariances, in the terms defined here.

Ukkonen et al. [12] gave an algorithm to solve the maximal subset matching problem in $O(mn \log m)$ time. It is still the fastest known deterministic algorithm for the problem. Clifford et al. [2] showed that quadratic running times are probably the best one can achieve for this problem by proving that the maximal subset matching problem is `3SUM`-hard. They also gave a randomized algorithm for the problem that works in time $O(n \log n)$.

4.2 Combining invariances

When using the sequence (string) representation, pitch-transposition invariance is easily combined with time-warp invariance (and the latter serves as a filtering method for time-scale invariance). However, the explicit encoding of the onset times in the geometrical representation makes it difficult to combine transposition invariance with most of the temporal invariances, such as time-scale invariance. The difficulty of combining transposition invariance and time-scale invariance is due to the fact that the former is an additive property, while the latter is multiplicative.

Romming and Selfridge-Field [10] gave the only known non-brute-force algorithm capable of dealing with polyphonic music, transposition invariance and time-scale invariance. Their algorithm is based on geometrical hashing and works in $O(n^3)$ space and $O(n^2m^3)$ time. By applying a window on the database such that w is the maximum number of events that occur in any window, the above complexities can be restated as $O(w^2n)$ and $O(wnm^3)$, respectively. The algorithm works on all three of the musical features discussed here (pitch, onset time and duration), finding a maximal subset match in such a scenario. However, as with the SIA algorithm family [7], its applicability to real world problems is reduced due to the fact that matches are mathematically exact, and so performance expression and error is difficult to account for.

5. CONCLUSIONS

In this paper we have discussed invariances related to content-based music retrieval; they are central concepts in defining and developing effective representations, similarity measures and algorithms to that end. Because of their centrality to the matter, invariances are widely used in the literature—but very seldom are they properly defined or their relationship discussed which has occasionally resulted in misuse of the term and confusion.

We have given a sparse taxonomy of the invariances along three featural dimensions of music—pitch, onset time and duration. We also defined stronger invariances, intrinsic to our formalism. The taxonomy shows explicitly the relationships of these invariances to each other. Moreover, we have precisely defined them, minimizing confusion in future discussion. The taxonomy works also as a useful tool in discussing what has been done, and in identifying where there is still much space for future developments towards efficient and effective CBMR tools.

It seems that the geometrical framework provides the best (and most natural) representation when dealing with polyphonic music. Using this framework, however, it is not easy to combine translation and time-scale invariances in a computationally efficient way; there is still a huge gap to be bridged in this respect to be able to meet the real world requirements for responsive and error-tolerant database queries. One way to improve error-tolerance—as is evident in our taxonomy—would be to adapt the geometrical frameworks to work also on the level of the more general invariances. To date, there is next to no work in this direction, though Lubiw and Tanur [6] presented an algorithm that measures the distance between the desired pitches and observed pitches that are combined in a final similarity value. So, with respect to our taxonomy, their work resides somewhere in between the two ends. Their method, although built on discrete space, does not straightforwardly lend itself to a non-strict time-scale invariance.

We are currently studying how to adapt the geometrical approach to the more general classes of our taxonomy thus achieving more error-tolerant geometrical methods for content-based music retrieval. Another direction is to refine the definitions in order to be able to discriminate methods that allow“gaps” (as the geometrical methods usually do) from those that do not (for instance, methods based on exact string matching).

6. ACKNLOWEDGEMENTS

We gratefully acknowledge funding for Kjell Lemström from the Academy of Finland (Grants #108547 and #129909). The work was carried out during his visit to the ISMS group at Goldsmiths, University of London. We also acknowledge the intellectual contribution of the ISMS group, in particular Richard Lewis.

7. REFERENCES

[1] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. Proms: A web-based tool for searching in polyphonic music. In *Proc. ISMIR'00*, Plymouth, MA, October 2000.

[2] R. Clifford, M. Christodoulakis, T. Crawford, D. Meredith, and G. Wiggins. A fast, randomised, maximal subset matching algorithm for document-level music retrieval. In *Proc. ISMIR'06*, pp. 150–155, Victoria, BC, Canada, October 2006.

[3] D. Conklin and I. H. Witten. Multiple viewpoint systems for music prediction. *J. New Music Research*, 24:51–73, 1995.

[4] K. Lemström and P. Laine. Musical information retrieval using musical parameters. In *Proc. ICMC'98*, pages 341–348, Ann Arbor, MI, 1998.

[5] K. Lemström and A. Pienimäki. On comparing edit distance and geometric frameworks in content-based retrieval of symbolically encoded polyphonic music. *Musicae Scientiae*, 4a:135–152, 2007.

[6] A. Lubiw and L. Tanur. Pattern matching in polyphonic music as a weighted geometric translation problem. In *Proc. ISMIR'04*, pages 289–296, Barcelona, October 2004.

[7] D. Meredith, K. Lemström, and G. A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *J. New Music Research*, 31(4):321–345, 2002.

[8] D. Müllensiefen, G. A. Wiggins, and D. Lewis. High-level feature descriptors and corpus-based musicology: Techniques for modelling music cognition. In A. Schneider, editor, *Systematic and Comparative Musicology: Concepts, Methods, Findings*, number 24 in Hamburger Jahrbuch für Musikwissenschaft. Peter Lang, Frankfurt am Main, 2008.

[9] D. Parsons. *The Directory of Tunes and Musical Themes*. S. Brown (Cambridge, Eng.), 1975.

[10] C. A. Romming and E. Selfridge-Field. Algorithms for polyphonic music retrieval: The hausdorff metric and geometric hashing. In *Proc. ISMIR'07*, Vienna, Austria, October 2007.

[11] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R.v. Oostrum. Using transportation distances for measuring melodic similarity. In *Proc. ISMIR'03*, pp. 107–114, Baltimore, MA, October 2003.

[12] E. Ukkonen, K. Lemström, and V. Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proc. ISMIR'03*, pages 193–199, Baltimore, MA, October 2003.

[13] G. A. Wiggins, M. Harris, and A. Smaill. Representing music for analysis and composition. In M. Balaban, K. Ebcioglu, O. Laske, C. Lischka, and L. Sorisio, editors, *Proc. 2nd IJCAI AI/Music Workshop*, pages 63–71, Detroit, Michigan, 1989.

[14] G. A. Wiggins, K. Lemström, and D. Meredith. SIA(M)ESE: An algorithm for transposition invariant, polyphonic content-based music retrieval. In *Proc. ISMIR'02*, pages 283–284, Paris, France, October 2002.

CALCULATING SIMILARITY OF FOLK SONG VARIANTS WITH MELODY-BASED FEATURES

Ciril Bohak, Matija Marolt
Faculty of Computer and Information Science
University of Ljubljana, Slovenia
{ciril.bohak, matija.marolt}@fri.uni-lj.si

ABSTRACT

As folk songs live largely through oral transmission, there usually is no standard form of a song - each performance of a folk song may be unique. Different interpretations of the same song are called song variants, all variants of a song belong to the same variant type. In the paper, we explore how various melody-based features relate to folk song variants. Specifically, we explore whether we can derive a melodic similarity measure that would correlate to variant types in the sense that it would measure songs belonging to the same variant type as more similar, in contrast to songs from different variant types. The measure would be useful for folk song retrieval based on variant types, classification of unknown tunes, as well as a measure of similarity between variant types. We experimented with a number of melodic features calculated from symbolic representations of folk song melodies and combined them into a melody-based folk song similarity measure. We evaluated the measure on the task of classifying an unknown melody into a set of existing variant types. We show that the proposed measure gives the correct variant type in the top 10 list for 68% of queries in our data set.

1. INTRODUCTION

With the rapid growth of digitization and appearance of digital libraries, folk song archives are (slowly but surely) entering the digital age. More and more folk song and music archives are being digitized, while most new data are already being collected in digital form.

Folk music is music that lives in oral tradition. It was composed by everyday people, and has in most cases never been written down or at least never published. It was mostly passed on to the next generation verbally and not in written form. Until folk music researchers started to put together folk music collections containing transcriptions, lyrics and other metadata, melodies were never put down in scores or any other symbolic representation. Several folk song collections are widely available; probably the most well known of them is the Essen Folksong Database [1] that includes 20.000 songs, mostly from Germany, Poland and China and minor collections from some other (mostly European) countries. The digital archive of Finnish Folk Tunes [2] is also a well known collection, containing approximately 9.000 folk tunes that were published as a collection of books between 1898 and 1933 and were digitized in 2002-2003. Some other collections are: The American Folk Song Collection [3], Australian Folk Songs [4], etc. We conducted our researh on songs from the the Ethnomuse archive [5], which contains folk music and dance collections of the Institute of Ethnomusicology, Scientific Research Centre of Slovene Academy of Sciences and Arts. The archive is especially suitable for our purpose, because it contains classifications of songs into *variant types*, tune families and genres.

Because folk songs live largely through oral transmission, there usually is no standard form of a song. As songs are passed through generations, they undergo an evolutionary process, parts change, they may be dropped and other parts may be added. Lyrics, as well as melodies get changed in the process. Each performance of a folk song may be unique and interpretations of the same song represent song *variants*. All *variants* of a song belong to the same *variant type*.

In this paper, we explore how measures extracted from folk song melodies relate to folk song variants. Specifically, we explore whether we can derive a melodic similarity measure that would correlate to variant types in the sense that it would measure songs belonging to the same variant type as more similar, in contrast to songs from different variant types.

The use of music information retrieval tools in folk music research was very limited until recently; a good overview can be found in the technical report of the Witchcraft project [6] as well as in [7, 8].

Our research is focused on developing an algorithm that calculates the similarity of two songs. In the following works several different approaches of a calculating the similarity measure are described. In [9] a method for melodic similarity of songs is presented; in [10], a method is described which uses each extracted statistical feature for training of separate self-organising map (SOM). All of the maps are later on used for training of a Supermap, on which melodies with similar features are located closer together; in [11] a comparison of different computational approaches to rhythmic and melodic similarity is made to find the features that characterise similarity of Dutch folk songs. A rhythmic similarity measure of folk songs is presented in [12]. Another similarity measure that uses pitch stability among a group of aligned folk songs is described in [13]. Which songs are similar, or how much they are alike, is not a precise problem. Not even humans always agree on whether two songs are similar or not, or which two songs are most alike. The study of how much experts agree on a manual annotation method for melodic similarity and the study of melody feature sets is described in [14].

In our paper we are proposing a system that uses simple melody-based features for classification of songs into variant types. While most of the previously mentioned papers describe methods for calculating rhythmic or melodic similarities in collections or finding features that are relevant in calculations of such similarities, our goal is to create a retrieval system for melodies, that will help us classify new unknown songs into already defined variant types.

2. SIMILARITY MEASURE

The main hypotesis of our paper is: *It is possible to classify folk song melodies into correct variant types based on statistical features of their melodies alone.* To either accept or reject our hypothesis, we first have to answer the following questions: *What kind of data do we have at our disposal and how much of it? Which features are we going to use and how to choose them? Which statistical methods should we use and how to choose them?*

The goal is to train a classifier that will classify individual variants into variant types. For this we created pairs of songs from Ethnomuse archive. *A positive example* is a pair of songs that are from the same variant type; a *negative example* is a pair of songs from different variant types.

We selected 650 folk songs belonging to 40 different variant types from the dataset. The scores for these melodies are available in Sibelius format, which we converted to MIDI. The set was split into two subsets: a *learning set* of 600 and an *independent test set* of 50 songs. The learning set was again split into two subsets: an *attribute selection learning set*, and an *attribute selection test set*. For the attribute selection learning set we only used songs from variant types with more then 7 variants. From variant types with more then 10 songs, we only used 10 randomly selected ones. The attribute selection test set was put together from 100 randomly selected songs from the learning set. For the purpose of training the classifier we created pairs of all songs in each of the attribute selection sets. For these sets we calculated percentages of positive and negative examples. The attribute selection learning set consists of 12.7% positive examples and 87.3% negative examples; attribute selection test set consists of 15,48% positive examples and 84.52% of negative examples.

For each melody, we calculated a set of 94 melody-based features with the help of the MIDI Toolbox [15]. We analyzed whether these features can be used to compare pairs of melodies and decide whether they belong to the same variant type or not. Because the task involves pairs of melodies, we created pairs of songs, for which we calculated compounded attributes as the quotient and absolute difference of individual features. All of the calculated attribute values were normalised and the SMO attribute selection method [16] used on the attribute selection sets to rank the attributes. We found the following attributes to be useful for variant type classification (details on the attributes can be found in [17]):

complebm: the measure is an expectancy based model of melodic complexity based on optimal combination of pitch and rhythm-related components calibrated in relation to the Essen Folksong Collection, where higher value means higher complexity.

entropy: relative entropy of note distribution in a matrix representation of note events.

meteraccent: measure of phenomenal accent synchrony. Meter accent is defined as:

$$meterAccent = mean(mh \cdot ma \cdot du) * (-1) \quad (1)$$

where vector *Metric hierarchy* (mh) indicates the locations of notes in the metric hierarchy (meter is calculated as an autocorrelation-based estimate - for further information see [17]); *Melodic accent* (ma) assigns melodic accents according to the possible melodic contours arising in 3-pitch windows. One can say that the melodic accent will be greater in places where the pitch changes. *Duration accent* (du) is defined in [19]

***gradus*:** degree of melodiousness (mean of Gradus suavitatis) was defined by Euler [18]. Gradus suavitatis bases on prime factorisation of note frequency ratios decreased by 1 and summed together with 1:

$$gradus\_suavitatis = 1 + \sum_{p_1 \in P} (p_i - 1) \qquad (2)$$

where P is set of all prime factors of frequency ratio and note frequency ratios are acquired from nominator and denominator matrices. Degree of melodiousness (gradus) is mean value of Gradus suavitatis for all note intervals:

$$gradus = mean(\sum_{n_i \in N} (n_i)) \qquad (3)$$

where N is set of all note intervals.

compltrans: Simonton's melodic originality score based on 2nd order pitch- class distribution of classical music derived from a set of music themes.

The selected features were used to train a logistic regression (LR) model. For each pair of melodies, the model outputs values between 0 and 1 for each instance; the closer the calculated value is to 1, the more probable it is that the pair of melodies belongs to the same variant type and vice versa, the closer the value is to 0, the lower the chance that the selected pair of songs is from same variant type. For the calculated values we had to set the threshold, that determines when the songs of a selected pair are from same variant type, and when not. The threshold was set so that the F-Measure reached the maximum on the *attribute selection learning set*. For later evaluation we have also calculated the F-Measure on the *attribute selection test set*.

For comparison we have also used the data to build the same model with SVM Regression (SVM) as well as Multilayer perceptron (MP). The Table 1 shows that there are only small differences between different machine learning models and that all the models are better than random classifier (RC) in all measures.

Table 1. F-Measure, precision and recall values of different models, for the attribute selection test set

Method	F-Measure	Precision	Recall
LR	0.2837	0.9888	0.1656
SVM	0.3396	0.8750	0.2107
MP	0.3237	0.8661	0.1990
RC	0.2649	0.8503	0.1600

3. EVALUATION AND DISCUSSION

3.1 Testing the model

To evaluate the logistic regression based similarity measure on a realistic task, we set up a retrieval system that takes an unknown melody as a query and returns an ordered list of melodies that should belong to the same variant type as the query. The queries were chosen from the *independent test set* and were compared to songs in the *learning set* with the logistic regression classifier trained as described previously; its output was used to rank the results.

Table 2. Ranks of first correct hits according to the proposed similarity measure.

Rank	Number of correct hits
1st	5
2nd	10
3rd or 4th	6
5th - 20th	18
21st - 30th	4
31st or worse	7

The ranked list of hits contains 600 songs; the *correct hits* are those belonging to the same variant type as the query song. For our model, the majority of first correct hits are ranked 30th or better (in 43 of 50 cases). The overall worst first correct hit is on 422nd place. Table 2 shows where first correct hits were ranked for the entire test set. For our model 68% of first correct hits lie within the top 10; a random classifier would reach 41%, so this is a significant improvement. The average rank of the first correct hit is 20.60th place, but if we exclude the most divergent results, the average rank of the first correct hit is 7.21th place for all but the worst 7 songs.

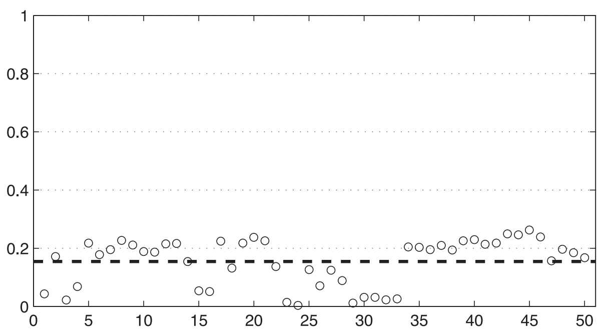
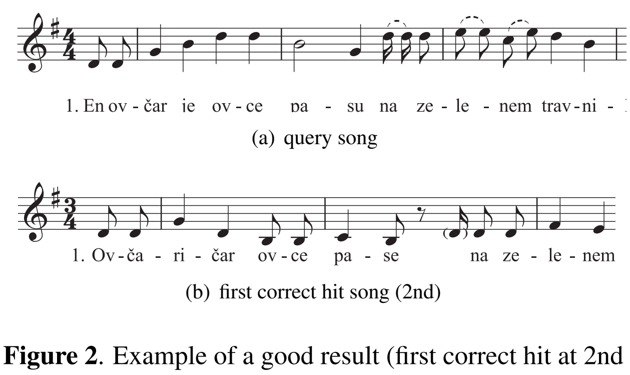


Figure 1. 11 point precision averages of test set items and their mean value.

Another measure frequently used for MIR system evaluation is 11 point precision average. This measure is calculated as the average precision at recall levels 0.0, 0.1, . . . , 0.9, 1.0. For our test set the calculated value of 11 point precision average for songs from the independent test set is 0.1544. In Figure 1 the circles represent 11 point precision average measure of each of the test set items. The dashed line indicates the mean value for all the test set items. Most of the worst cases (those under the mean line in Figure 1) are either from variant types with less then 7 variants or variants from bigger variant types that derogate the most.

3.2 Case study

The variant type with the most songs in our data set contains 163 songs. The best first correct hit for a query song from this variant type is 2nd, while the worst first correct hit is in the 31st place. 11 point precision average for this variant type is 0.2096, which is close to 11 point precision of the best query song - 0.2494. Following is the comparison of the best first correct hit and worst first correct hit examples for this variant type.

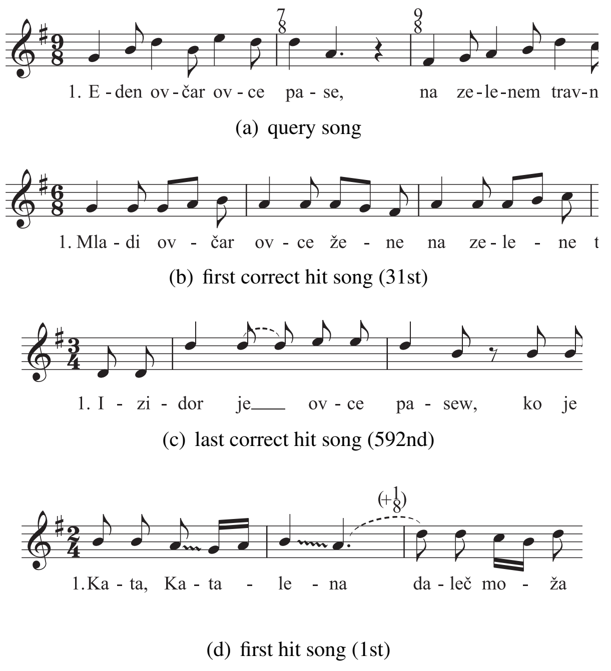


1. En ov - čar je ov - ce pa - su na ze - le - nem trav-ni -
(a) query song
1. Ov - ča - ri - čar ov - ce pa - se na ze - le - nem
(b) first correct hit song (2nd)

Figure 2. Example of a good result (first correct hit at 2nd place) for the same variant type as in Figure 3.

Figure 2 shows an example, where the first correct hit was on the 2nd place; the query and the correct hit are shown. The reason why this result is ranked so good (it was ranked 2nd) is because not only *complebm* (the values, 5.1332 of query and 5.2631 of first hit song, are quite similar), *meteraccent* (pitch in both, query and first correct hit song, is not monotonic) and *gradus* (both songs have quite high melodiousness) values are very similar with the values for query song, but also other two features (*entropy* and *compltrans*) are very similar with values for query song; which is not true for the previous example.

Figure 3 shows the worst first correct hit example. The query song, its first and last correct hits and the first hit song on the ranked list returned by our system are given. The main reason why the song in Figure 3(c) was ranked so low, is because of the major differences in *complebm* (the query song value is 5.0880, the first correct hit song value is 4.8136 and the last correct hit song value is 3.9494), *meteraccent* and *gradus* melodic features in comparison to the query (Figure 3(a)); on the other hand *complebm* and *compltrans* values of the first hit song (Figure 3(d)), are closer to the query song values, then the first correct hit song values.


1. E - den ov - čar ov - ce pa - se, na ze - le - nem trav-n
1. Mla - di ov - čar ov - ce že - ne na ze - le - ne t
1. I - zi - dor je ov - ce pa - sew, ko je
1.Ka - ta, Ka - ta - le - na da - leč mo - ža

(a) query song

(b) first correct hit song (31st)

(c) last correct hit song (592nd)

(d) first hit song (1st)

Figure 3. Example of a bad result (first correct hit at 31st place) for the variant type with the most examples.

4. CONCLUSIONS AND FUTURE WORK

As we show, there is some correspondence between simple statistical measures calculated on folk song melodies and the classification of folk songs into variant types. While results are far from very good and such a basic approach cannot be used to build a fully automatic variant type classification system, the obtained similarity measure is good enough to create a retrieval system for melodies of an unknown variant type that will give us list of a few (in our case 10) variant types, that will contain the correct type with high probability (in our case 68%). We also plan to combine the obtained melodic similarity measure with lyrics-based similarity measures and to use it for visualization of folk song melodies in the Ethnomuse archive.

5. ACKNOWLEDGMENT

Authors would like to thank anonymous reviewers for their constructive comments, suggestions and for pointing out errors; they helped make this article better.

The work was partly supported by the Slovenian Government-Founded R&D project EthnoCatalogue: creating semantic descriptions of Slovene folk song and music.

6. REFERENCES

[1] Helmut Schaffrath. The Essen Folksong Collection. D. Huron (ed.), Stanford, CA, 1995. computer database.

[2] Finnish Folk Tunes. University of Jyvskyl, 2004. URL: esavelmat.jyu.fi.

[3] The American Folk Song Collection, 2004. URL: kodaly.hnu.edu/home.cfm.

[4] Australian Folk Songs, 1994. URL: http://folkstream.com/.

[5] Grega Strle and Matija Marolt. Conceptualizing the Ethnomuse: Application of CIDOC CRM and FRBR. *Proceedings of CIDOC2007*, 2007.

[6] Peter van Kranenburg, Jörg Garbers, Anja Volk, Frans Wiering, Louis P. Grijp, and Remco C. Veltkamp. Towards integration of music information retrieval and folk song research. Technical Report UU-CS-2007-016, Department of Information and Computing Sciences, Utrecht University, 2007.

[7] Petri Toiviainen and Tuomas Eerola. Visualization in comparative music research. In *COMPSTAT 2006 - Proceedings in Computational Statistics*, pages 209–221, 2006.

[8] Peter van Kranenburg, Jörg Garbers, Anja Volk, Frans Wiering, Louis P. Grijp, and Remco C. Veltkamp. Towards integration of mir and folk song research. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 505–508, Vienna, Austria, September 2007. Österreichische Computer Gesellschaft.

[9] Rainer Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Utrecht University, Netherlands, February 2007.

[10] Petri Toiviainen and Tuomas Eerola. Method for comparative analysis of folk music based on musical feature extraction and neural networks. In *In III International Conference on Cognitive Musicology*, pages 41–45, 2001.

[11] Anja Volk, Jörg Garbers, Peter van Kranenburg, Frans Wiering, Louis P. Grijp, and Remco C. Veltkamp. Comparing Computational Approaches to Rhythmic

and Melodic Similarity In Folksong Research. In *Proc. MCM 2007*, 2007.

[12] Anja Volk, Jörg Garbers, Peter van Kranenburg, Frans Wiering, Remco C. Veltkamp, and Louis P. Grijp. Applying rhythmic similarity based on inner metric analysis to folksong research. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 293–296, Vienna, Austria, September 2007. Österreichische Computer Gesellschaft.

[13] Jörg Garbers, Peter van Kranenburg, Anja Volk, Frans Wiering, Remco C. Veltkamp, and Louis P. Grijp. Using pitch stability among a group of aligned query melodies to retrieve unidentified variant melodies. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 451–456, Vienna, Austria, September 2007. Österreichische Computer Gesellschaft.

[14] Anja Volk, Peter van Kranenburg, Jörg Garbers, Frans Wiering, Remco C. Veltkamp, and Louis P. Grijp. A manual annotation method for melodic similarity and the study of melody feature set. *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, September 2008.

[15] Tuomas Eerola and Petri Toiviainen. Mir in matlab: The midi toolbox. In *ISMIR*, 2004.

[16] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.

[17] Tuomas Eerola and Petri Toiviainen. *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä, Jyväskylä, Finland, 2004.

[18] Leonhard Euler. *Tentamen novae theoriae musicae*. 1739.

[19] Richard Parncutt. A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception*, 11(4):409–464, 1994.

AUTOMATIC GENERATION OF LEAD SHEETS FROM POLYPHONIC MUSIC SIGNALS

Jan Weil, Thomas Sikora
Communication Systems Group
Technische Universität Berlin

J.-L. Durrieu, Gaël Richard
Institut Telecom
Telecom ParisTech
CNRS LTCI

ABSTRACT

A lead sheet is a type of music notation which summarizes the content of a song. The usual elements that are reproduced are the melody, chords, tempo, time signature, style and the lyrics, if any. In this paper we propose a system that aims at transcribing both the melody and the associated chords in a beat-synchronous framework. A beat tracker identifies the pulse positions and thus defines a beat grid on which the chord sequence and the melody notes are mapped. The harmonic changes are used to estimate the time signature and the down beats as well as the key of the piece. The different modules perform very well on each of the different tasks, and the lead sheets that were rendered show the potential of the approaches adopted in this paper.

1. INTRODUCTION

The lead sheet format is a convenient form of music notation for songs. It is mostly used for popular music and famously represented by collections of Jazz standards, e.g., *The Real Book*. It allows the musician to see all the important elements necessary to perform a song in a very compact format. It mostly consists of a single staff; the melody is notated in Western music standard, with the associated lyrics under the staff and the chord sequence noted above it. The lead sheet also often specifies the style, i.e., the way the melody has to be played, e.g., straight or swung rhythm, and the way the accompaniment should be generated from the chords. Of course, it also defines the time signature, the key and the tempo.

Very few works have been oriented towards producing usable music scores directly from audio. In [1], the authors estimate the melody, the bass line, and the chords. However, the results are not temporally quantized, so the output is not completely suited for lead sheet generation itself. This temporal quantization is indeed a non-trivial problem and we propose a potential solution in this paper.

The proposed lead sheet transcription system can be broken down into four seperate modules which exchange intermediate results. These modules are depicted in Fig. 1. The beat tracker provides a continuous pulse grid which forms the temporal basis for the other modules. The algorithm favours faster tempi such that the risk of phase errors is minimized and ensures a continuos beat grid. The reader is refered to [2] for details about the chosen approach. In this article, we directly use the output of this algorithm. The i^{th} beat position in seconds is denoted b_i. The harmonic module estimates beat-aligned chord sequences, the most likely measure grid, and the key of the piece. The measure grid is in turn used to refine the chord sequence by making chord change probabilities depend on the position in the measure. The chord detection module is based on the approach described in [3]. The melody module first separates the main melody and the accompaniment building on the approach presented in [4]. The model is extended such that the fundamental frequencies of the main melody and the musical (MIDI) notes of the melody are jointly estimated. The rendering module determines the appropriate time signature, quantizes the note onsets and durations of the melody to sub-divisions of the beat level, divides both melody and chords in measure blocks, and applies pitch spelling depending on the estimated key.

In the following section we describe the chord detection scheme and how the down-beat positions are estimated using the detected chord sequence. After that the key estimation method is introduced. The melody extraction is discussed in Sec. 4. In Sec. 5, we describe how the lead sheets are rendered. Finally, we present the results as well as our conclusions and perspectives.

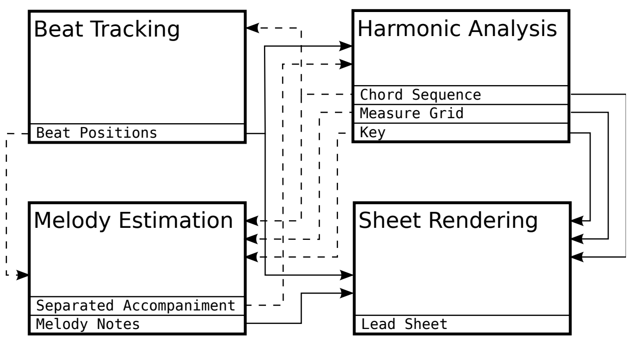


Figure 1. Modules of the proposed system along with the intermediate results they exchange. Dashed lines mark potential future dependencies.

2. ESTIMATION OF CHORDS AND MEASURES

2.1 Chord detection

The chord detection module can be considered one of the numerous followers of the approaches described in [5] and [3], which are based on Hidden Markov Models (HMM). We model the chords as states of the HMM using a chord alphabet comprising major and minor chords, i.e., the ergodic model has $S = 24$ states ω_k, $k \in [1, S]$. The chord sequence is given as the most likely sequence of states given the observed feature sequence; this is known as the decoding problem which is solved using the Viterbi algorithm. Training and decoding is done in a 10-fold cross-validation setup.

2.1.1 Feature extraction

Beat-synchronous chroma vectors computed from the audio data form the observable features. The audio signal is mixed to a single channel and downsampled to 11025 Hz. We compute a constant-Q spectrogram [6] from note E2 (82.4 Hz) to note D#6 (1.24 kHz) using a hop size of 512 samples [1]. Due to the chosen lowest frequency the length of the longest window is 4096 samples. Chroma vectors are computed by summing up the magnitude of the transform for each of the 12 pitch classes over all four octaves. We then use the result from the beat tracking module to average all feature vectors within beat boundaries. Let $\boldsymbol{x}_c(i)$ denote the 12-dimensional chroma vector representing the time segment between beat positions b_i and b_{i+1}, $i = 1, 2, ...$

2.1.2 Training

The observation distribution is modeled as a multivariate Gaussian per state with mean vectors $\boldsymbol{\mu}_k$ and (full) covariance matrices $\boldsymbol{\Sigma}_k$, $k \in [1, S]$. The prior probabilities are considered uniformly distributed. Both the transition probabilities and the observation distribution are computed from the training sets using beat-quantized annotation data in a similar fashion as described for methods 1 and D in [7].

2.1.3 Initial chord sequence decoding

In the first stage, the chord sequence is decoded using the classic Viterbi algorithm. Let $q_1(i)$ denote the initial estimate of the decoded chord symbol which is assumed to have emitted $\boldsymbol{x}_c(i)$. Based on this initially decoded sequence we estimate the measure grid.

2.2 Estimating the measure grid

We assume that the probability of chord changes depends on the position in a measure and that, generally, chords are more likely to change at the beginning of measures [8]. We also assume a constant time signature; we do not, however, assume a 4/4 meter (although it clearly dominates our database). We consider a set of measure grid candidates of width $\nu \in [3, 4, ..., 8]$, i.e., each third, fourth, ..., eighth

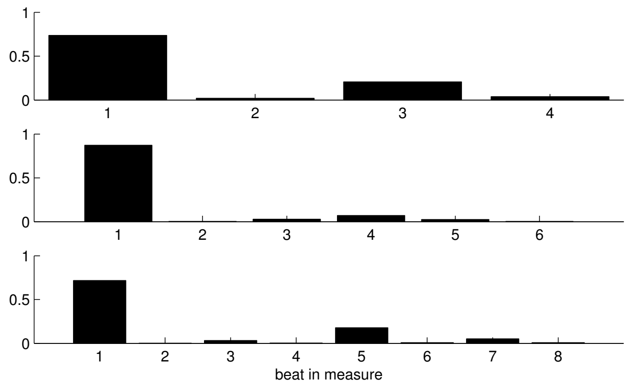


Figure 2. Probability of chord changes depending on the position in the measure for 4, 6, and 8 beats in a measure, respectively.

beat is assumed a down-beat. For each ν we have to consider ν potential phase candidates $\phi \in [1, \nu]$, i.e., the first down-beat is b_1, b_2, ..., or b_ν. For each of these grid width and phase candidate pairs, we compute the score

$$s(\nu, \phi) = T_{cc}(\nu, \phi) - F_{cc}(\nu, \phi), \qquad (1)$$

where $T_{cc}(\nu, \phi)$ denotes the number of grid points which fall on beat positions with a chord change, i.e., $q_1(i-1) \neq q_1(i)$, and $F_{cc}(\nu, \phi)$ denotes the number of grid points without chord changes. The pair $(\nu_o, \phi_o) = \arg\max s(\nu, \phi)$ determines the chosen measure grid. Note that ν does not necessarily correspond to the numerator of the time signature as the beat we tracked may actually reflect half-time or double-time tempo.

2.3 Refined chord sequence decoding

We use the measure grid estimate to compute the refined chord sequence $q_2(i)$ by making the transition probabilities depend on the position in the measure. Based on the down-beat information given by the annotation we compute the distribution of chord change positions relative to the measures from the training set. For the database we used, which will be discussed in Sec. 6, there are three possible values of ν: 4 (4/4 meter), 6 (6/8 meter), and 8 (4/4 meter; beat represents 8th notes). Fig. 2 depicts an example for the resulting probability profiles. As anticipated, chords are most likely to change on the beginning of a measure. We now propose a modified Viterbi decoding procedure. As we assume a continuous beat and measure grid, we can compute the current beat position in a measure $b_m = (i - \phi_o) \pmod{\nu_o} + 1$. Now the transition probability matrix is modified in the following manner: the diagonal, i.e., the probability to remain in the current state, is set to $1 - p_{cc}(b_m)$, where $p_{cc}(b_m)$ denotes the probability of a chord change at beat position b_m in the measure. The remaining non-diagonal elements are scaled such that they add up to $p_{cc}(b_m)$. Decoding the HMM using these varying transition probabilities gives the refined chord sequence $q_2(i)$.

[1] Note that, for the database we used, we can consider all pieces perfectly tuned to A4 = 440 Hz

3. KEY ESTIMATION

To estimate the key one can compute an average chroma profile and correlate it to key-specific templates [9]. Instead, we propose to compute the mean vector of the chord likelihoods using the trained Gaussian distributions for the chord states. We compare both approaches. We train key template profiles for major and minor keys which are circularly shifted to form all 24 possible key profiles. To this end, $\boldsymbol{x}_c(i)$ is circularly shifted such that the key is mapped to the root C for all pieces in the training set. The chroma-based templates $\boldsymbol{\mu}_{K1}(m)$ for both key modes m, major and minor, are computed as the mean vector of all shifted chroma vectors representing mode m. These templates have dimension 12. For the chord-based templates, the multi-variate Gaussian distribution is evaluated to compute the likelihoods $P(\boldsymbol{x}_c|\omega_k)$. The 24-dimensional mean vector of these chord likelihoods for both modes m, normalized to add up to one, gives the second set of key templates $\boldsymbol{\mu}_{K2}(m)$. To estimate the key of a piece we compute both the mean chroma vector and the normalized mean chord likelihoods. Then we compute the dot product of these test profiles and all 12 shifted variants of the two key templates as a measure of correlation. Note that for $\boldsymbol{\mu}_{K2}(m)$ the two halves of the likelihood vectors representing major and minor chords must be shifted independently. The key for which the template maximizes the dot product is chosen. This is done for both $\boldsymbol{\mu}_{K1}(m)$ and $\boldsymbol{\mu}_{K2}(m)$ to compare the results.

4. MAIN MELODY ESTIMATION

4.1 Global model for main melody sequence

Our model for melody estimation is based on the model proposed in [4]. In order to achieve a meaningful quantization of the desired melody line, we adapted the note duration model initially proposed in [10].
The observation audio signal x is considered as the instantaneous mixture of two contributions, the main instrument voice v playing the main melody and the accompaniment or background music m, i.e., $x = v + m$. This relation stays valid for the short time Fourier transform matrices X, V and M of these signals. We assume that the signal was decomposed into N frames, with Fourier transforms of F positive frequency bins. We model the complex Fourier transforms as complex proper centered Gaussians, for which we more specifically model the variances.

On one hand, for the accompaniment M, the "Nonnegative Matrix Factorization" (NMF) model is retained. The resulting variance $S_{M_n}(f)$ for $M_n(f)$, at frame n and frequency f is then given by:

$$S_{M_n}(f) = \sum_{r=1}^{R} W_M(f,r)H_M(r,n), \qquad (2)$$

where R is the number of elements in the spectrum dictionary W_M and H_M is the activation coefficient matrix associated to W_M. In matrix notation, with the variance matrix S_M such that $S_M(f,n) = S_{M_n}(f)$: $S_M = W_M H_M$.

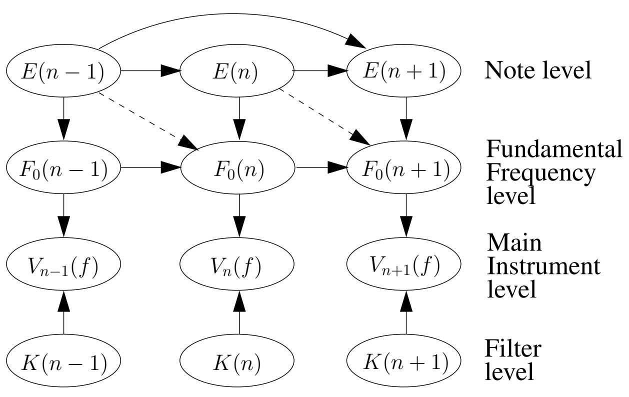


Figure 3. Generative model for the main instrument source/filter model.

On the other hand, the main instrument voice V is modelled through a source/filter model. The source part is driven by a three-layer generative model, shown on the upper part of Fig. 3. The filter part is modelled thanks to a two-layer model (lower part of Fig. 3). Note that the main instrument level V is also a hidden layer which, along with the accompaniment level M, gives the mixture observation level X.

The source level comprises two hidden levels. First, the fundamental frequency level $F_0(n)$ controls the pitch of the main instrument. These variables are dependent on the second layer, the note level. The evolution between the states of the note level $E(n)$ and the fundamental frequency states are explained in Sec. 4.2.

The filter layer is simpler, because here, we are more interested in the note and frequency levels. Therefore, we allow more flexibility in the evolution of the filter part and do not model any constraint on the corresponding sequence.

The main instrument level is then generated with the filter and fundamental frequency levels. The variance matrix S_V for $V_n(f)$, such that $S_V(f,n) = S_{V_n}(f)$, is given by:

$$S_V = \underbrace{(\overbrace{W_\Gamma H_\Gamma}^{W_\Phi} H_\Phi)}_{\text{Filter part}} .* \underbrace{(W_{F_0} H_{F_0})}_{\text{Source part}}, \qquad (3)$$

where W_Γ is a $F \times P$ dictionary of P smooth atomic elements, W_{F_0} a dictionary of N_{F_0} spectral combs for the voiced source part and H_Γ the coefficient matrix such that the actual filter dictionary $W_\Phi = W_\Gamma H_\Gamma$. The activation coefficient matrices for the filter and the source parts respectively are H_Φ and H_{F_0}.

The optimal note sequence $E = \{E(1), \ldots, E(N)\}$ is estimated within a Maximum Likelihood (ML) framework:

$$\hat{E}, \hat{F}_0, \hat{K} = \text{argmax}_{E,F_0,K} \log \text{p}(X, E, F_0, K). \qquad (4)$$

Such an estimation is computationally too intensive, and we propose in the next section some simplifications to estimate the different levels of the problem.

4.2 Model Approximations

In order to estimate the desired note sequence, we first neglect the constraint of having only one filter per frame. We then limit the problem to:

$$\hat{E}, \hat{F_0} = \text{argmax}_{E,F_0} \log \text{p}(X, E, F_0), \tag{5}$$

The right-hand side of Eq. (5) can be further expressed as:

$$\log \text{p}(X, E, F_0) = \log \text{p}(X|F_0) + \log \text{p}(F_0|E) + \log \text{p}(E),$$

where, as shown on Fig. 3, we use that the sequence X is independent from E conditional on F_0. Furthermore, we assume that:

$$\log \text{p}(X|F_0) \approx \sum_n \log \tilde{H}_{F_0}(F_0(n), n). \tag{6}$$

In (6), the observation likelihood conditional on the melody fundamental frequency is approximated with a modified version $\tilde{H}_{F_0}$ of the source activation coefficient matrix H_{F_0} calculated on the data as described in [4]. During this first estimation round, the observation frames are assumed independent. We set $\tilde{H}_{F_0} = H_{F_0}$ and then normalize each column of $\tilde{H}_{F_0}$ by its maximum value.

The log-likelihood of the fundamental frequency sequence, conditional on the note state sequence, $\log \text{p}(F_0|E)$ is equal to:

$$\sum_{n=2}^{N} \log \text{p}(F_0(n)|F_0(n-1), E(n)) + \log \text{p}(F_0(1)|E(1))$$

Strictly speaking, $F_0(n)$ should also depend on $E(n-1)$, but for simplicity, we drop this dependency. We further assume that $\text{p}(F_0(n)|F_0(n-1), E(n))$ is proportional to the product:

$$\text{p}(F_0(n)|F_0(n-1)) \times \text{p}(F_0(n)|E(n)).$$

$\text{p}(F_0(n)|F_0(n-1))$ is a *prior* that simulates smooth f_0 variations. $\text{p}(F_0(n)|E(n))$ penalizes the distance between the fundamental frequency and the "expected" frequency for the note state $E(n)$. These functions are set to:

$$\text{p}(F_0(n) = f_2|F_0(n-1) = f_1) \propto \exp(-\alpha|\log_2(\frac{f_2}{f_1})|),$$
$$\text{p}(F_0(n) = f_0|E(n) = e) \propto \exp(-\beta|\log_2(f_0/f_e)|^2),$$

where f_e is the "standard" frequency for note $E = e$.
At last, we use the "segmental" duration model in [10] for the note state evolution:

$$\log \text{p}(E_{1:n}) = \log \text{p}(E_n|E_{1:n-1}) \log \text{p}(E_{1:n-1}). \tag{7}$$

The interested reader may find more information on this model in [10] , especially on the exact equations for the durations as well as on the beam searching algorithm that allows to find an optimal path for the sequence E.

To put it in a nutshell, we proceed as follows:

1. First assuming the independence of neighbouring frames, the parameters for the fundamental frequency and the filters are globally estimated.
2. We then extract pitch candidates for the main melody from the matrix H_{F_0} and use them to restrain the range of pitches to be tested when looking for the optimal path.
3. Finally, we find the optimal path of sequences E and F_0 using a beam search strategy, maximizing the approximated likelihood Eq. (5).

4.3 Generating a usable melody track transcription

The note sequence must be further quantized to produce a musical score. The fundamental frequencies are quantized onto the Western musical scale using the model for the sequence E. The temporal quantization is yielded to the rendering module such that the time signature can be considered.

5. LEAD SHEET GENERATION

Eventually, all the pieces of information are put together to render a readable transcription. Depending on ν_o and the estimated tempo we choose an appropriate time signature. Both the chords and the melody are processed in measure chunks. The onsets and the duration of the melody notes are quantized to a subdivision of quarter notes. These are usually eighth notes, which gives a good tradeoff between quantization errors and spurious notes. Depending on the estimated key, a simple pitch spelling algorithm is applied for both notes and chords. Basically, we choose note and chord names such that the distance on the circle of fifths is minimized.

6. RESULTS AND EVALUATION

In order to assess the different modules of the transcription system, we need a database for which the chords, the beat, and the melody line are annotated. Assembling such a database by manually annotating audio recordings is highly time-consuming. We found using the Band-In-A-Box [2] (BIAB) format a convenient way of generating the annotation in a semi-automatic way. BIAB is software which generates musical accompaniment given a sequence of chords, a tempo, and a style; it also supports melody tracks. Thus, BIAB files contain all the information which is relevant for the lead sheet generation task. Actually, BIAB even features lead sheet printouts, which gives a convenient reference for the subjective assessment of the results.

Our database comprises 278 files adding up to about 16.5 hours of audio material. It is a subset of the *Pop & Rock* database gathered by members of the Yahoo BIAB user group. Details are available on-line [11]. The files are rendered substituting the oboe for the singing voice, which is an instrument that shares a number of acoustic properties with the human voice. We used a modified version of one of the BIAB parsers available on-line to extract the relevant information from the BIAB files.

[2] http://www.band-in-a-box.com/

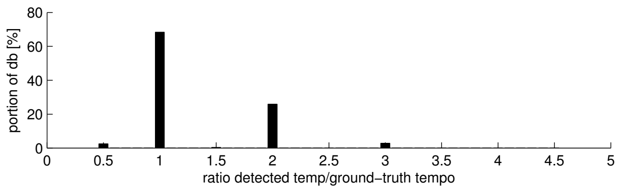


Figure 4. Histogram of the ratio *detected tempo / ground-truth tempo* over the entire database.

6.1 Beat tracking evaluation

We use the same metric as in [2] to evaluate the beat tracking module. The performance measure is the fraction of the longest continuous portion of the piece for which all beats are detected. A ground-truth beat is considered correctly tracked if the absolute distance to the nearest detected beat is smaller than 17.5 % of the period. If the ratio of the detected tempo to the ground-truth tempo is either two or three, we only consider every second or third beat, respectively, during the evaluation and choose the starting beat which maximizes the performance (see [2] for details). Fig. 4 depicts the histogram of the ratio *detected tempo / ground-truth tempo*. There is a single file for which the ratio is 1.5 which must be considered wrong. The average beat tracker performance is 94.1 %. For 91.4 % of all pieces we correctly track more than 90 % of the beats.

6.2 Down-beat tracking evaluation

The down-beat information implicitly given in the BIAB files cannot be trusted. Historically, BIAB's support for meters other than 4/4 is weak and sometimes the system is abused, e.g., a 6/8 meter would be recorded as a slower 4/4 meter where each beat of the 4/4 meter collects three beats of the 6/8 meter. Generally, the beat given in BIAB files is not guaranteed to correspond to the tactus period, i.e., the denominator of the time signature. It may reflect half-time tempo, double-time tempo, or ternary meters. To assess the proposed measure grid estimation approach we have to take these peculiarities into account. In compliance with the beat tracker performance measure we consider a down-beat correctly detected if the absolute distance to the closest ground-truth downbeat is less than 17.5 % of the period estimated by the beat tracking module. We compute the down-beat performance measure as the fraction of the longest continuous portion for which all down-beats were correctly detected. This is a particularly conservative measure as it combines both the result of the beat tracking module and the estimated measure grid based on detected chord change points. The average down-beat performance is 87.3 %.

6.3 Chord estimation evalution

We use basically the same evaluation measure as applied to the 2008 MIREX chord detection task[3]. All annotated chord symbols are mapped to their root triads resulting in five chord classes: major, minor, diminished, augmented, and suspended. (Note that 98.3 % of the chord symbols in our database fall into the major and minor categories.) This results in $5 \cdot 12 + 1$ possible states, including the no-chord state, which is used in the two pickup bars. The evaluation measure is the overlap in seconds between the detected chord sequence and the ground-truth sequence mapped to the 61 possible states as described above. The average overlap for the entire database is 76.4 % for the initial chord detection phase and 79.3 % for the refined estimation using transition probabilities depending on the position in the measure. The average overlap quantized to beats, which is more relevant to the transcription task, is 80.0 %; it is 82.7 % when the pickup bars are discarded.

6.4 Key estimation evaluation

For transcription purposes, a confusion of relative major and minor keys does not matter as the key signature remains the same. To evaluate the key estimation algorithms we thus compute the difference in the numbers of sharp or flat symbols, i.e., the smallest distance on the circle of fifths either clockwise (positive) or counterclockwise (negative). Fig. 5 shows the histogram of the key error measure for both key estimation approaches over the entire database. Both approaches correctly estimate the key signature for the majority of the pieces. However, the portion of the database for which the absolute key signature error is not greater than one is 80.2 % using the chroma profiles and 93.5 % using the mean chord likelihoods. The chroma-based approach is prone to confuse minor keys with their relative major keys (+3), e.g., A major instead of A minor, or with the key of the (major) dominant in the case of harmonic minor (+4), e.g., E major instead of A minor. Examining the statistics reveals that the variance remains significant for the chroma profiles. One could try to use a Gaussian classifier instead but, here, the method using the mean chord likelihoods works very well. In Pop and Rock music the chord range of the diatonic scale is often extended to include chords of keys which are close on the circle of fifths, e.g., a major chord on the minor 7th degree of a major scale (subdominant of the subdominant); this explains absolute key signature errors of one.

6.5 Melody tracking evaluation

For the melody estimation, we selected 11 songs that fit our definition of the main melody. For each song, the melody estimation algorithm returns the transcribed notes of the melody, with their MIDI note number, onset and offset times. A transcribed note is considered correct if there is a note in the reference with the same MIDI note number of which the onset time is close to the one of the transcribed note. The absolute difference between these onset times should be less than 150 ms. We compute precision, recall, and f-measure, and we provide the score obtained using the perceptually motivated measures in [12]. On our database, we obtain average recall, precision and f-measure of, respectively, 63 %, 68 % and 63 %. The average perceptive F-measure is 69 %. Fig. 6 shows the box and whiskers for

[3] MIREX 2008 Evaluation Campaign, website: http://www.music-ir.org/mirex/2008/

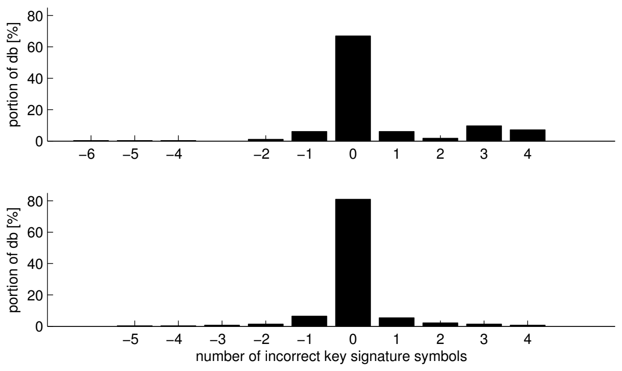


Figure 5. Histogram of the key signature error in steps on the circle of fifths for the chroma-based (top) and the chord likelihood-based method (bottom).

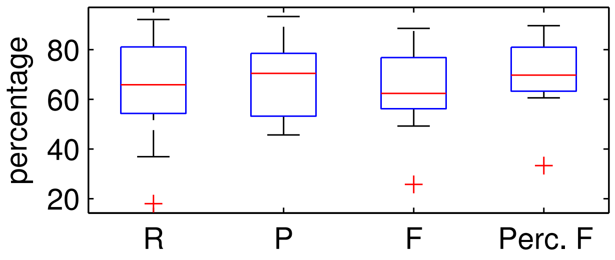


Figure 6. Box and whiskers plot of the results for melody estimation: Recall (R), Precision (P), F-measure (F) and perceptive F-measure (Perc. F).

the 11 songs. The outlier corresponds to a song for which the melody was too fast and too variable for the melody tracker to follow. The results are promising; however, the database we used was rather small and experiments on a bigger and more realistic database should be held in the future.

7. CONCLUSIONS AND PERSPECTIVES

We have proposed a lead sheet generation system. The tempo, time signature, chords, key, and melody were handled by several modules that can interact with each other. The chord sequence helps in determining the time signature, which in turn can be used to refine the chord sequence and also defines the minimum note duration for quantizing the melody. Our approach groups several modules that achieve state-of-the-art performance on each sub-task. Assessing the overall quality of the generated transcription is not trivial and subjective evaluation should be held for that purpose. For some examples available on-line [11] the resulting score is close to musician expectations. Some assumptions make the system targeted at Western music genres like Pop and Rock as represented by the chosen database. Evaluation of the sub-systems on real audio data remains to be done. The system could be further improved by allowing more joint estimations. A global model could cover all the aspects of the problem for which all the parameters for the different modules are jointly estimated. However, as for the melody module, such a model might be too complicated to be directly solved. Instead, this integration can be approximated for instance by including the detected beat positions in the melody note duration model. The melody estimation and separation can also be used to improve the chord sequence estimation.

8. ACKNOWLEDGEMENTS

This work was supported by the European Commission under contract FP6-027026 (KSpace), by the European Community's Seventh Framework Programme [FP7/2007-2011] under grant agreement n^{o} 216444 (PetaMedia) and by OSEO, French State agency for innovation, as part of the Quaero Programme.

9. REFERENCES

[1] M. P. Ryynanen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *CMJ*, 32(3):72–86, 2008.

[2] J. Weil, J.-L. Durrieu, G. Richard, and T. Sikora. Beat tracking using the delta-phase matrix. Technical report, Institut Telecom, Telecom ParisTech, CNRS LTCI, 2009.

[3] J.P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *ISMIR*, pages 304–311, 2005.

[4] J.-L. Durrieu, G. Richard, and B. David. An iterative approach to monaural musical mixture de-soloing. In *ICASSP*, pages 105–108, 2009.

[5] A. Sheh and D.P.W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *ISMIR*, pages 185–191, 2003.

[6] J.C. Brown and M.S. Puckette. An efficient algorithm for the calculation of a constant Q transform. *JASA*, 92:2698–2698, 1992.

[7] H. Papadopoulos and G. Peeters. Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *CBMI'07*, pages 53–60, 2007.

[8] H. Papadopoulos and G. Peeters. Simultaneous estimation of chord progression and downbeats from an audio file. In *ICASSP*, pages 121–124, 2008.

[9] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, 2006.

[10] E. Vincent. Musical source separation using time-frequency source priors. *IEEE Trans. on Audio, Speech, and Lang. Proc.*, 14(1):91–98, 2006.

[11] Accompanying website. http://www.nue.tu-berlin.de/research/leadsheets/.

[12] A. Daniel, V. Emiya, and B. David. Perceptually-based evaluation of the errors usually made when automatically transcribing music. In *ISMIR*, pages 550–555, 2008.

MINIMUM CLASSIFICATION ERROR TRAINING TO IMPROVE ISOLATED CHORD RECOGNITION

J.T. Reed[1], Yushi Ueda[2], S. Siniscalchi[3], Yuki Uchiyama[2], Shigeki Sagayama[2], C.-H. Lee[1]
[1]School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
{jreed,chl}@ece.gatech.edu
[2]Graduate School of Information Science and Technology
The University of Tokyo, Hongo, Bunkyo-ku, Tokyo 113-8656 Japan
{ueda,uchiyama,sagayama}@hil.t.u-tokyo.ac.jp
[3]Department of Electronics and Telecommunications
Norwegian University of Science and Technology, Trondheim, Norway
marco77@iet.ntnu.no

ABSTRACT

Audio chord detection is the combination of two separate tasks: recognizing what chords are played and determining when chords are played. Most current audio chord detection algorithms use hidden Markov model (HMM) classifiers because of the task similarity with automatic speech recognition. For most speech recognition algorithms, the performance is measured by word error rate; i.e., only the identity of recognized segments is considered because word boundaries in continuous speech are often ambiguous. In contrast, audio chord detection performance is typically measured in terms of frame error rate, which considers both timing and classification. This paper treats these two tasks separately and focuses on the first problem; i.e., classifying the correct chords given boundary information. The best performing chroma/HMM chord detection algorithm, as measured in the 2008 MIREX Audio Chord Detection Contest, is used as the baseline in this paper. Further improvements are made to reduce feature correlation, account for differences in tuning, and incorporate minimum classification error (MCE) training in obtaining chord HMMs. Experiments demonstrate that classification rates can be improved with tuning compensation and MCE discriminative training.

1. INTRODUCTION

As online music databases continue to grow in size, more effective retrieval mechanisms are needed. In particular, recognizing certain musicological, acoustical, and cultural factors in a musical piece impact notions of similarity. One such musicological factor which has seen an increased research focus is automatic chord detection, which is a mid-level representation and a first-step in identifying the harmony of a given musical work.

Most recent approaches to identifying chords from the acoustic signal are based on using chroma features as inputs into a hidden Markov model (HMM) based system. An early approach in literature using an HMM-based system was [1], where an ergodic HMM provides the initial chord progression modeling and updated using *N*-best rescoring techniques. Sheh and Ellis [2] deal with an inadequate amount of training data by assuming that chroma vectors from the same mode (e.g., *Major*) and different pitch classes can be considered as rotated versions of one another. Bello *et al.* [3] incorporate musical knowledge into the transition probabilities and HMM parameters to improve the results. Lee and Slaney [4] increase the amount of training data available by synthesizing audio to provide accurate chord and boundary information. Improvement is made in [5] by using key-dependent ergodic HMMs and warping the chroma features into tonal centroid features [6], which gives the relation of the chroma features on the circles of fifths, minor thirds, and major thirds.

The HMM framework is inspired by automatic speech recognition (ASR), where HMMs represent words or subword units. However, the ASR community only considers the recognition rate (i.e., what was said) as important and ignores timing information (i.e., recognizing when each spoken unit begins and ends). In contrast, audio chord detection is measured in terms of frame error rate (FER), which incorporates both tasks. This paper proposes optimizing these two task separately and focuses on the problem of classification rate (i.e., recognizing what was said). To the authors' knowledge, only [2] evaluates these tasks separately. Specifically, Sheh and Ellis consider forced alignment, where the correct chord sequence is known and the timing information is extracted.

This paper implements several improvements to evaluate the limits of the chroma/HMM system for classification rates. In particular, the goal of this paper is to improve the classification rate between highly confused chords in the feature space. For instance, one of the most common chord detection errors is between parallel modes, which share the same root, but differ in their key signature; e.g, *C Major* versus *C minor*. The cause for confusion is because the difference between a *Major* and *minor* chord is a flattening of the major third to a minor third, which may only be a few hertz.

The baseline system adopted in this study is the current state-of-the-art and placed first in the 2008 MIREX Audio Chord Detection task (Task 2: no pre-training) [7]. To attenuate percussive sounds, the harmonic-percussive source separation (HPSS) algorithm [8] is used in the baseline system to isolate the harmonic part of the spectrum prior to chroma extraction and maximum likelihood (ML) estimation. This paper incorporates the improvements of automatic tuning compensation and minimum classification error (MCE) training [9].

The automatic tuning algorithm is a simplification of the one proposed in [3]. Small, uniform databases, such as the Beatles Chord Database [10], experience improved performance with tuning normalization because slight differences in tuning cause confusion between highly competing chords. This can lead to confusion among parallel modes since they differ by a single note, for example. Due to the trade-off between spectral and time-based resolution in frame-based music processing, a slight difference in tuning could allow for energy to bleed into neighboring energy bands, leading to confusion.

MCE, a highly successful discriminative training approach, enhances ASR performance by overcoming two assumptions made by parametric approaches. Like speech, the assumption that the true distribution of chroma vectors is an HMM is an approximation to yield a parametric fit. ML techniques estimate parameters corresponding to the mode of the likelihood function. However, if the true distribution differs from the assumed model, the ML technique is not guaranteed to yield an optimal performance. In addition, the strength of the parametric fit relies on accurate parameter estimates. However, current acoustic chord databases are quite small in size and contain around 100 songs from one to five artists. With such small artist diversity, it is unlikely that current databases are a good representation of the entire acoustic space. MCE integrates a discriminative training approach into the parameter estimation problem by directly optimizing the performance classification; i.e., classification error. Specifically, a logistic transform incorporates the classification error rate into the objective function so that gradient probabilistic descent will yield an improved set of parameters.

The baseline algorithm, is described in Section 2 and improvements are detailed in Section 3. Experimental results in Section 4 compare modifications to the ML baseline. Finally, Section 5 gives concluding remarks.

2. BASELINE ALGORITHM: MIREX 2008 SUBMISSION

2.1 Harmonic/Percussion Source Separation

As noted in [11], transients and noise decrease the chord recognition accuracy in chroma-based approaches. This is largely due to percussive sources, which spread energy across the entire frequency spectrum. While the authors in [11] use a median filter to smooth percussive effects, this paper uses the HPSS algorithm [8], which integrates the harmonic and percussive separation into the objective function

$$J(\mathbf{H},\mathbf{P}) = \frac{1}{2\sigma_H^2}\sum_{k,n}(H_{k,n-1}-H_{k,n})^2 + \frac{1}{2\sigma_P^2}\sum_{k,n}(P_{k-1,n}-P_{k,n})^2 \tag{1}$$

where $H_{k,n}$ and $P_{k,n}$ are the values of the power spectrum at frequency index k and time index n for the harmonic spectrum, $\mathbf{H}$, and the percussive spectrum, $\mathbf{P}$, respectively. The parameters σ_P^2 and σ_H^2 need to be set experimentally. To ensure that each time-frequency component of the harmonic and percussive spectrum components sum to a value equal to the original spectrum, $W_{k,n}$, and to ensure that power spectrums remain positive, the following constraints are added to the minimization of (1)

$$H_{k,n} + P_{k,n} = W_{k,n} \tag{2}$$
$$H_{k,n} \geq 0 \tag{3}$$
$$P_{k,n} \geq 0 \tag{4}$$

Note that minimizing (1) is equivalent to maximum likelihood estimation under the assumption that $(H_{k,n-1} - H_{k,n})$ and $(P_{k-1,n} - P_{k,n})$ are independent Gaussian distributed variables. This simplification leads to a set of iterative update equations for the harmonic and percussive spectrums. At the output of HPSS are two waveforms; one of these contains a percussive-dominated spectrum and the other a harmonic-dominated spectrum. Further details can be found in [8].

2.2 Chromagram

Chroma vectors are the most common features in audio chord detection algorithms and describe the energy distribution among the 12 chromas; i.e., pitch classes. To derive chroma vectors, the harmonic-emphasized music signal is first downsampled to 11025 Hz. Next, the signal is broken into frames of 2048 samples with a 50% overlap. The constant Q transform [12] provides spectral analysis using a logarithmic spacing of the frequency domain, whereas the traditional discrete Fourier transform (DFT) uses a linear spacing of the frequency domain. The resulting spectrum, S, of the audio signal $s(t)$ is given by

$$S(k) = \sum_{t=0}^{T(k)-1} w(t,k)s(t)e^{-j2\pi f_k t} \tag{5}$$

where the analysis window, $w(t, k)$, and the window size, $T(k)$, are functions of the frequency bin index, k. The center frequency of the k-th bin is designed to match the equal-temperament scale [13]. For example, if it is desired to have one bin per note on an 88-piano keyboard, then the bin center frequencies are

$$f_l = 2^{l/\beta} f_{\text{ref}} \tag{6}$$

with the number of bins per octave, β, set to 12, the minimum reference frequency, f_{ref}, set to the frequency of *A0* (i.e., 27.5 Hz), and $l = \{1, 2, ..., 88\}$. The resulting chroma vector for frame n is

$$c_n(b) = \sum_{r=0}^{R} |S(b + r\beta)| \tag{7}$$

where $b = \{1, 2, ..., \beta\}$ is the chroma bin number and R is the number of octaves considered.

2.3 HMM classifier

The optimal chord sequence, W^* is decoded such that [14]

$$\begin{aligned} W^* &= \arg\max_W P(W|C) \\ &= \arg\max_W \frac{P(C|W)P(W)}{P(C)} \\ &\propto \arg\max_W P(C|W)P(W) \end{aligned} \tag{8}$$

where $C = \{c_1, c_2, \ldots, c_N\}$ is the sequence of chroma vectors. The probabilities of the acoustic model and tonality model are $P(C|W)$ and $P(W)$, respectively. Note that in speech, $P(W)$ is the language model; i.e., the prior probability for a sequence of words, W. For this paper, the tonality model assumes that every chord is equally likely. The reason for the proportionality in (8) is that $P(C)$ is the same for all chord sequences. The acoustic model is the probability of producing the observed chroma vectors for chord W and is modeled with a HMM; i.e.,

$$P(C|W) = \pi_{q_0} \prod_{n=1}^{N} a_{q_{n-1}q_n} b_{q_n}(c_n) \tag{9}$$

where π_{q_0} is the initial state probability, $a_{q_{n-1}q_n}$ is the transition probability from state q_{n-1} to q_n, and $b_{q_n}(c_n)$ is the output likelihood, which is modeled by a Gaussian mixture model (GMM)

$$b_{q_n}(c_n) = \sum_{d=1}^{D} \omega_d N(\mu_d, \Sigma_d) \tag{10}$$

where D is the number of mixtures, ω_d is the mixture weight for component, d, and $N(\mu_d, \Sigma_d)$ is a Gaussian density with mean μ_d and covariance Σ_d. If the features are uncorrelated, then the covariance matrix is diagonal, $\Sigma_d = diag(\sigma_1, \sigma_2, ..., \sigma_{12})$. Note that a single state HMM is equivalent to a GMM.

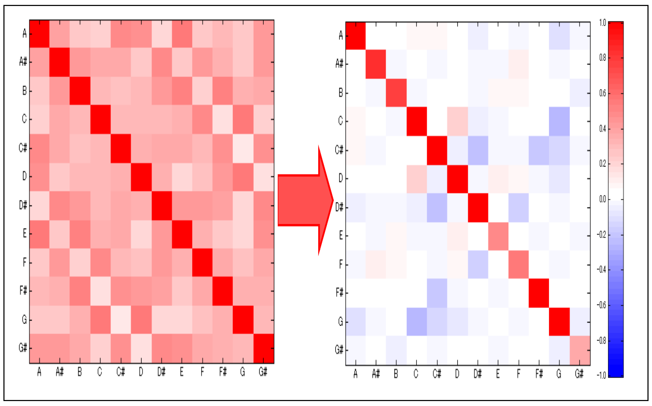


Figure 1. Cross correlations of chroma features. Right: original chroma features. Left: DFT chroma features. Dark shades (or red if in color) indicate higher correlation (light shades (or blue in color) indicate low correlation.

3. IMPROVED ALGORITHM

Since the baseline algorithm is in a format compatible with the automatic speech recognition paradigm, it provides a good framework to test more advanced speech processing techniques, such as MCE, as an alternative model estimation step. In addition, parameter reduction and tuning compensation are implemented and compared to the baseline.

3.1 Fourier Transform Chroma Features

As noted in [3], chroma features are highly correlated because harmonics of different pitch classes overlap and is demonstrated in the left part of Figure 1. For instance, the third harmonic of *C4* (261.63 Hz fundamental, 784.89 Hz third harmonic) is highly confusable with *G5* (783.99 Hz fundamental). However, as shown on the right of Figure 1, the resulting feature dimensions have less cross-correlation after applying a DFT on the chroma features.

3.2 Tuning Compensation

A second enhancement is tuning compensation. Standard tuning is such that the *A* note above middle *C* on a piano keyboard (i.e., *A4*) is approximately 440 Hz. However, artists may intentionally or unintentionally have a reference tuning different from the standard. This can lead to confusion in algorithms which assume that all music is tuned to the standard reference, as shown in Figure 2. In the upper part of Figure 2, a 12-dimensional chroma is applied to a piece of music whose energy distribution is higher in frequency than standard tuning ($A4 \simeq 440Hz$). Therefore, the signal energy is distributed between the intended note (e.g., *C3*) and the neighboring note (e.g., *C#3*). Considering that *Major* and *minor* chords differ by only one semi-tone in a single note, this can lead to large confusion between the *Major* and *minor* modes of a given chord.

To account for differences in tuning, a simplified version of [3] is implemented. The original tuning compensation algorithm uses 36 bins per octave ($\beta = 36$ in (7)) in the calculation of chroma vectors. A peak picking algorithm produces a histogram, which gives information about

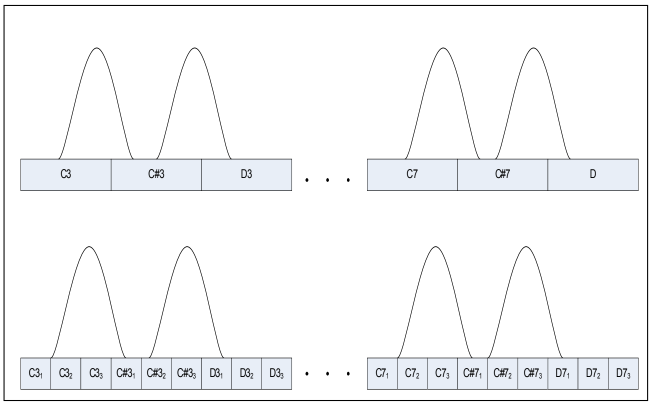


Figure 2. Upper: Hypothetical mistuned energy distribution. Bottom: Find tuning alignment giving maximum energy distribution at sampled points.

the tuning of the piece. A circular shift is then applied to the chroma vector as a corrective factor. The reason for peak picking is that noise sources (e.g., percussion and transients) corrupt the chroma vectors with non-harmonic sources.

However, because of HPSS, a simplified procedure filters percussive noise sources and leaves energy due to harmonic sources. The new algorithm takes a 36-dimension chroma vector for each frame in a song, so that each note considered is divided into a three bins

$$\tilde{c}_n^{(\alpha)}(b) = \sum_{r=0}^{R} |S(b + \alpha + r\beta)| \tag{11}$$

where $\alpha = \{1, 2, 3\}$ and $b = \{1, 2, \ldots, 12\}$. The algorithm then retains the set the α which produces the chroma vector with the greatest Euclidean length

$$c_n = \arg\max_{\tilde{c}_n^{(\alpha)}} \left(\tilde{c}_n^{(\alpha)} \cdot \tilde{c}_n^{(\alpha)}\right) \tag{12}$$

3.3 Minimum Classification Error Learning

As mentioned in the Introduction, MCE is a highly successful discriminative training approach to improving automatic speech recognizers over ML and MAP estimation. The optimization criterion in MCE is to minimize the estimated classification loss

$$L(\Lambda) = \frac{1}{J} \sum_{j=1}^{J} \sum_{m=1}^{M} l_m(X_j; \Lambda) 1(X_j \in \Omega_m) \tag{13}$$

where Λ are the model parameters, J is the number of training examples, $\{X_1, X_2, \ldots, X_J\}$, M is the number of categories (i.e., chords), $l_m(\cdot)$ is a loss function, and $1(X_j \in \Omega_m)$ is one if X_j is in category Ω_m and zero otherwise. Typically, a 0-1 loss is used for $l_m(\cdot)$, which makes the objective function discrete and difficult to optimize. However, a common approximation for the loss function is to replace the 0-1 loss with a logistic function [9],

$$l_m(X_j; \Lambda) = \frac{1}{1 + \exp(-\gamma d_m(X_j; \Lambda) + \theta)} \tag{14}$$

where γ and θ are experimental constants and $d_m(X_j; \Lambda)$ is a misclassification measure, which is negative with a correct classification and positive when a classification error is made.

A good indication of misclassification is the distance between the correct class and competing classes; therefore, the chosen misclassification measure is based on the generalized log-likelihood ratio [9]:

$$d_m(X; \Lambda) = -\log g_m(X; \Lambda) + \log \left[G_m(X; \Lambda)\right]^{1/\eta} \tag{15}$$

where

$$g_m(X; \Lambda) = \max_q \pi_{q_0}^{(m)} \prod_{n=1}^{N} a_{q_{n-1} q_n}^{(m)} b_{q_n}^{(m)}(c_n) \tag{16}$$

$$G_m(X; \Lambda) = \frac{1}{M-1} \sum_{p, p \neq m} \exp[g_p(X; \Lambda)\eta] \tag{17}$$

where η is an experimental positive constant and the superscript $^{(m)}$ refers to the m-th HMM. Note the misclassification measure in (15) compares the probability of the target class against a geometric average of the competing classes. The parameter η determines the importance of the competing classes by the degree of competition with the target class. In particular, as $\eta \rightarrow \infty$, (17) returns only the most competitive class. A gradient probabilistic descent procedure [9] produces a set of parameters that yields a local optimum of (13) through the update equations

$$\Lambda_{\tau+1} = \Lambda_\tau - \epsilon \left. \frac{\partial l_m(X_j; \Lambda)}{\partial \Lambda} \right|_{\Lambda = \Lambda_\tau} \tag{18}$$

In order to keep the necessary constraints for an HMM density, the following transformations are used [9]:

$$\tilde{\mu}_d^{(m)}(b) = \frac{\mu_d^{(m)}(b)}{\sigma_d^{(m)}(b)} \tag{19}$$

$$\tilde{\sigma}_d^{(m)}(b) = \log \sigma_d^{(m)}(b) \tag{20}$$

4. RESULTS

4.1 Experimental Setup

The evaluation database is the set of studio albums by *The Beatles*, which were transcribed at the chord level by Chris Harte *et. al* [10]. As in the 2008 MIREX contest, only the *Major* and *minor* chords are used for the evaluation. All extended chords, *Augmented*, and *diminished* chords are mapped to the the base root, *Major*, and *minor* chords, respectively. A two-fold evaluation is implemented, where half the albums are used as a training set and the remaining half are used as a test set in the first fold. In the second fold, the roles of the training and test set are swapped. Note all songs from a particular album occur in either the test or training set for each individual fold. This is the same setup as MIREX 2008, but the third fold in MIREX 2008 was removed because it was observed that the test cases where already covered in the first two folds. Prior to HPSS, audio is downsampled to 11025 Hz. In addition, chord boundary

Fold	BL	FT	FT+TC	FT+TC+MCE
1	57.84	57.84	61.91	73.59
2	61.74	61.74	64.97	71.35
Total	59.95	59.95	63.57	72.37

Table 1. Classification accuracies for Fourier transform features and tuning compensation (BL = baseline, FT = Fourier Transform, TC = tuning compensation, MCE = minimum classification error).

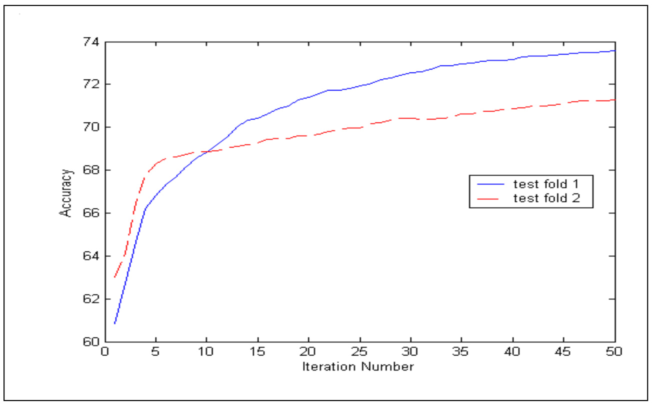


Figure 3. Classification accuracy versus iteration number.

information is assumed to be known and results are given in percentage of correctly recognized isolated chord segments, except in Section 4.3, where results are given in frame accuracy.

4.2 Isolated Chord Recognition Results

Table 1 details the improvement over the ML approach, where chords are modeled with a single Gaussian distribution with a full covariance matrix for the baseline, Fourier transform, and Fourier transform with tuning compensation cases. The MCE results listed used 50 iterations of the gradient probabilistic descent algorithm. The parameters γ, θ, and η were found experimentally by using a set of five songs from the training set as a cross-validation set. After cross-validation, the entire training set is used to retrain the system.

Tuning compensation provides a modest, but consistent gain in performance. Applying a Fourier transform does not change the performance from the baseline. However, the main advantage of applying a discrete Fourier transform is to attenuate the correlation in chroma features, and is equivalent to a discrete cosine transform for real, symmetric data [14]. In addition speech processing algorithms, such as MCE, assume diagonal covariance matrices in the GMM observation probability. Therefore, applying MCE is straightforward and results in a drastic increase in performance over ML estimation. In particular Figure 3 demonstrates, generally, each iteration of the gradient probabilistic descent algorithm improves the classification rate.

To understand the types of errors that remain, the confusion matrix is presented in Table 2. It is observed that *Major* chords are classified more accurately than *minor* chords. Specifically, many errors are due to recognizing *minor* chords with the correct root, but wrong mode; i.e., the parallel *Major* chord. For example, 82% of *c minor* chords are recognized as *C Major*. The second most common type of error is in mistaking a *Major* chord for its *minor*, which are chords that share the same key signature, but differ in the root note. For example, *e minor* is confused with *G Major* 12% of the time. Note, that no language model is used in this current paper since the goal of this paper is to study the confusions that arise due acoustic confusability in the chroma/HMM framework.

Fold	BL	FT	FT+TC	FT+TC+MCE
1	74.96	74.96	77.04	77.90
2	72.95	72.95	73.54	74.51
Total	73.46	73.46	75.20	76.10

Table 3. Frame accuracy for continuous chord recognition.

# Frames	BL	FT+TC	FT+TC+MCE
0	73.91	75.20	76.12
1	76.59	77.92	78.93
2	78.61	79.94	80.99

Table 4. Frame accuracy versus number of frames removed at chord boundary.

4.3 Continuous Chord Recognition

While this paper is mainly concerned with isolated chord classification, an additional experiment demonstrates the performance of continuous chord recognition. In this case, the frame accuracy is used as the performance metric. The results are presented in Table 3. As expected, improvement is less pronounced with adequate boundary information. Further analysis shows that one reason for the performance drop is due to identifying chord boundaries. As shown in Table 4, allowing a tolerance region of two frames on either side of a true chord transition point increases the frame accuracy. Specifically, 20% of the error occurred within two frames of a chord transition point when at least one chord to either side of the transition point was detected correctly. In [2], it was demonstrated that chroma/HMM setup performed well during forced alignment (i.e., when the chord sequence is given), but poorly when no information on the chord sequence was given. These results indicate that the chord detection problem might benefit from treating the two tasks separately and optimizing each task individually.

5. CONCLUSIONS

This paper considers audio chord detection as two separate tasks: (1) classifying what chords are played and (2) determining when chords begin and end. Several advanced pre-processing techniques are implemented such as HPSS, which attempts to separate transients and percussive sources from the harmonic spectrum. Further, eliminating

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	c	c#	d	d#	e	f	f#	g	g#	a	a#	b
C	91					2		2			2											2		
C#		71	2		2		12					5		7										
D	1		89		1			2		2												1		3
D#				81		1					9		1			6					1			
E					94					1							2							1
F	3					91		2			1		1		1							2		
F#	1			1			91		1	1				3					2				1	
G	1		1					94									1							
G#							3		83									3			10			
A			1		1					94									1			1		
A#				1	1	1					91	1			1					3		1	1	
B				1	1		1		1	3	1	85							1		2			3
c	82							9					9											
c#		1			3				1	5	1	2		87										
d	6		12			13		3		3	1				57							4		1
d#												33				67								
e	5		1		19			12		5							50					8		1
f	1	1			1	11			16		6							63					1	
f#			2	1	2		7			11									76					1
g	3							14			9				1		1			71				
g#					4							12									85			
a	5				2	2				13												78		
a#		2		2							40												56	
b			6					1				20												71

Table 2. Chord confusion matrix (%). Rows are true chords, columns are hypothesized chords. Capital letters represent *Major* chords and lowercase letters represent *minor* chords.

the correlation between chroma features allows for the use of many speech processing tools because these tools are built using the assumption of diagonal matrices in the observation probability densities.

In this paper, tuning compensation and MCE enhance the chord recognition task over traditional maximum likelihood by reducing the confusion due to noise in the feature extraction stage. In the future, the authors hope to incorporate other advanced speech processing techniques, such as *N*-best re-scoring, to combat other areas of confusion such as the confusion between *minor* chords and their relative and parallel *Major* equivalents. Finally, it was observed that even when chords are detected correctly, 20% of the error occurred at chord transition points. Therefore, the authors are investigating chord transition detection algorithms to optimize the second task of chord detection.

6. REFERENCES

[1] T. Kawakami, M. Nakai, H. Shimodaira, S. Sagayama: "Hidden Markov Model Applied to Automatic Harmonization of Given Melodies," *IPSJ Technical Report*, 99-MUS-34, pp. 59-66, Feb., 2000. (*in Japanese*)

[2] A. Sheh and D.P.W. Ellis: "Chord Segmentation and Recognition Using EM-trained Hidden Markov Models," *Proc. ISMIR*, pp. 183–189, 2003.

[3] J.P. Bello and J. Pickens: "A Robust Mid-level Representation for Harmonic Content in Musical Signals," *Proc. ISMIR*, pp. 304-311, 2005.

[4] K. Lee and M. Slaney: "Automatic Chord Recognition from Audio Using an HMM with Supervised Learning," *Proc. ISMIR*, pp. 133-137, 2006.

[5] K. Lee and M. Slaney: "Acoustic Chord Transcription and Key Extraction from Audio Using Key-dependent HMMs Trained Synthesized Audio," *IEEE TASLP*, Vol. 16, No. 2, pp. 291-301.

[6] C.A. Harte, M.B. Sandler, and M. Gasser: "Detecting Harmonic Change in Musical Audio," *Proc. Audio Music Comput. Multimedia Workshop*, pp. 21-26, 2006.

[7] J. Downie: "Music Information Retrieval Evaluation eXchange (MIREX)," [Online]. Available: http://www.musicir.org/mirex/2008/index.php/

[8] N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka, S. Sagayama "Separation of a Monaural Audio Signal into Harmonic/Percussive Components by Complementary Diffusion on Spectrogram," *Proc. EUSIPCO*, 2008.

[9] B.-H. Juang, W. Chou, C.-H. Lee: "Minimum Classification Error Rate Methods for Speech Recognition," *IEEE TSAP*, Vol. 5, No. 3, pp. 257-265, 1997.

[10] C. Harte, M. Sandler, S. Abdallah, and E. Gómez "Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations," *Proc. ISMIR*, pp. 66-71, 2005.

[11] H. Papadopoulos and G. Peeters: "Large-scale Study of Chord Estimation Algorithms Based on Chroma Representation and HMM," *Intern. Wkshp. Content-Based Multimedia Indexing*, pp. 53-60, 2007.

[12] J. Brown: "Calculation of a constant Q spectral transform," *J. Acoust. Society America*, Vol. 89, No. 1, pp. 425-434, 1991.

[13] S. Kostka and D. Payne: *Tonal Harmony*, McGraw Hill, 2004.

[14] L. Rabiner and B.-H. Juang: *Fundamentals of Speech Recognition*, Prentice Hall, New Jersey, 1993.

A METHOD FOR VISUALIZING THE PITCH CONTENT OF POLYPHONIC MUSIC SIGNALS

Anssi Klapuri
Department of Signal Processing, Tampere University of Technology
anssi.klapuri@tut.fi

ABSTRACT

This paper proposes a method for visualizing the pitch content of polyphonic music signals. More specifically, a model is proposed for calculating the salience of pitch candidates within a given pitch range, and an optimization technique is proposed to find the parameters of the model. The aim is to produce a continuous function which shows peaks at the positions of true pitches and where spurious peaks at multiples and submultiples of the true pitches are suppressed. The proposed method was evaluated using synthesized MIDI signals, for which it outperformed a baseline method in terms of precision and recall. A straightforward visualization technique is proposed to render the pitch salience function on the traditional staves when the musical key and barline information is available.

1. INTRODUCTION

Pitch analysis of polyphonic music is a challenging task where computational methods have not yet achieved the accuracy and flexibility of trained musicians. Several different approaches have been proposed towards solving the problem. Some methods are based on a statistical model of the input signal [1], whereas some others model the human auditory system [2]. Joint detection of multiple pitches has been proposed [3], contrasted by techniques which carry out iterative pitch detection and cancellation [4]. Some methods are based on unsupervised learning [5] and some others on supervised classification [6]. These examples illustrate the remarkable variety of methods that have arisen in an attempt to mimic the human ability to make sense of complex sound mixtures. A nice review of multipitch detection algorithms can be found in [7].

A drawback of many of the existing multipitch analysis methods is that they produce a discrete set of detected pitch values (or, fundamental frequencies, F0s [1]) instead of a continuous detection function that would show the likelihoods of all possible pitch values within the pitch range of interest. For pitch content visualization and acoustic feature extraction purposes, a continuous detection function is often more desirable since it allows the human eye or a subsequent post-processing algorithm to pick the interesting features from the detection function and to decide which peaks correspond to true pitches.

A fundamental difficulty in computing such detection functions (think of the autocorrelation function for example) is that they do not show a peak only at the position of the true pitch, but also at twice and half the correct pitch, and often at all multiples and submultiples of it. This ambiguity is particularly challenging in multipitch detection where the detection function easily becomes congested with spurious peaks due to the ambiguity associated with each component sound.

Various techniques have been proposed to suppress the extraneous peaks in a detection function. For example, it has been proposed to detect F0s either iteratively or jointly and to cancel all the spurious peaks that are already explained by the detected F0s [3,4]. However, these methods produce only a discrete set of F0s. Karjalainen and Tolonen proposed a method which produces an entire detection function, where the spurious peaks were suppressed using an “enhancing” procedure [2].

In this paper, we propose a model for calculating the salience (or, strength) of all pitch values within a given range of interest, and investigate a numerical optimization technique to find the model parameters so that the truly existing pitch frequencies are indicated with peaks that tend towards unity value and spurious peaks are forced towards zero. We also propose a visualization technique, where the computed pitch salience is rendered on the staves of common musical notation. This allows people who are able to read the music notation to play directly from the visualization, or to use it to study performance nuances, such as pitch glides, vibrato, and expressive timing.

2. METHOD

In the proposed method, an audio signal is first blocked into frames which are short-time Fourier transformed. The spectra are whitened (see Sec. 2.1) and the noise floor in the spectrum due to drums and other non-pitched sounds is estimated (Sec. 2.2). The whitened spectrum and the noise spectrum are used in the pitch salience model (Secs. 2.3 and 2.4). These steps are now explained in more detail.

[1] The terms pitch and F0 are used here interchangeably.

2.1 Level normalization and spectral whitening

The time-domain audio signal $x(n)$ is blocked into partly-overlapping analysis frames that are windowed using the Hamming window. The signal within each frame is level-normalized to unity variance, zero-padded to twice its length, and then discrete Fourier transformed to obtain the magnitude spectrum $X_t(k)$ in frame t. Each frame is processed independently, therefore we drop the frame index t in the following for convenience.

Spectral whitening, or flattening, is applied on $X(k)$ in order to suppress timbral information and thereby make the subsequent pitch analysis more robust to various sound sources. This is achieved by calculating power σ_c^2 of the signal within narrow frequency bands c and by scaling the signal within each band by $g_c = \sigma_c^{\nu-1}$, where $\nu = 0.16$ is a parameter determining the amount of whitening. Center frequencies f_c of the subbands are distributed uniformly on the critical band scale, $f_c = 229(10^{(0.33c+1)/21.4} - 1)$, and each subband $c = 1, \ldots, 96$ has a triangular power response extending from f_{c-3} to f_{c+3}. The resulting whitened magnitude spectrum is denoted by $Y(k)$.

2.2 Noise estimation

From the viewpoint of pitch analysis, the sounds of drums and the beginning transients of many pitched instruments are considered as “noise”. Several methods have been proposed in the literature for estimating the “noise” (stochastic spectral component) in music (see [3] for review). Perhaps the most widely used is the sinusoids plus noise model, where sinusoidal components are detected and subtracted in the frequency domain, and the residual is considered as coloured (filtered) white noise. Another, quite robust method is to calculate a moving median at local regions of the magnitude spectrum.

Here the emphasis is laid on the computational efficiency and on the robustness of the method against spectral peaks which are assumed to correspond to pitched sounds and should not affect the estimate. The proposed noise spectrum estimation method consists of the following steps. First, a moving average $N'(k)$ over the whitened spectrum $Y(k)$ is calculated as

$$N'(k) = \frac{1}{u_k - l_k} \sum_{k'=l_k}^{u_k} Y(k') \qquad (1)$$

where l_k and u_k define the lower and upper boundaries of the critical-band subbands within which $N'(k)$ is calculated. Note that (1) can be computed very efficiently by first calculating cumulative sum $\bar{Y}(k)$ over $Y(k)$ and then $N'(k) = \bar{Y}(u_k) - \bar{Y}(l_k - 1)$. The band boundaries l_k and u_k can be pre-stored.

To make the noise estimate immune to spectral peaks, another moving average is calculated, but including in the averaging only frequency bins for which $Y(k) < N'(k)$. The resulting noise spectrum estimate is denoted by $N''(k)$. Performing one more local averaging of $N''(k)$ (by substituting $N''(k)$ in place of $Y(k)$ in (1)) produces the final noise spectrum estimate $N(k)$.

The presented noise estimation procedure is besides simple, also computationally efficient. If the input signal consists of coloured white noise (without pitched sounds), it can be shown that $\mathsf{E}[N(k)] = 0.61\mathsf{E}[N_0(k)]$, where $\mathsf{E}[\cdot]$ denotes expectation and $N_0(k)$ is the true noise spectrum being estimated. In other words, the estimated spectrum depends linearly on the true spectrum. In practice, however, the input signal may contain pitched sounds which affect the estimate and the scaling factor can be anything between 0.61 and about 1.0. In our case, the subsequent optimization process explained below takes care of finding the linear scaling factor for $N(k)$, therefore the proposed noise estimation method is well suited here.

2.3 Pitch salience model

For convenience, the whitened spectrum $Y(k)$ and the estimated noise spectrum $N(k)$ are stored as columns in matrix $\mathbf{Y}$, together with an all-one “spectrum”:

$$\mathbf{Y} = \begin{bmatrix} Y(0) & N(0) & 1 \\ Y(1) & N(1) & 1 \\ \vdots & \vdots & \vdots \\ Y(K-1) & N(K-1) & 1 \end{bmatrix} \qquad (2)$$

Let us also define basis functions

$$a_m(k) = [\log(k+1)]^{m-1} \qquad (3)$$

where k denotes the frequency index and $m = 1, 2, \ldots, M$ indexes the basis functions. This is a polynomial basis on the log-frequency scale. For convenience, the bases are collected as columns in matrix $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_M]$.

The columns of $\mathbf{Y}$ are linearly combined into a single spectrum $Z(k)$ according to the following linear model:

$$Z(k) = (\mathbf{AW}.\times\mathbf{Y})\,\mathbf{1} \qquad (4)$$

where $.\times$ denotes element-wise multiplication, $\mathbf{W}$ is a matrix of size $(M \times 3)$ that contains the model parameters, and $\mathbf{1}$ is an all-one vector of length 3 (the number of columns in $\mathbf{Y}$). Note that the product $\mathbf{AW}$ is a matrix of size $(K \times 3)$, the same size as $\mathbf{Y}$. The three columns define frequency responses for the three columns of $\mathbf{Y}$, before they are summed (by multiplying with $\mathbf{1}$) to obtain the final spectrum $Z(k)$. The first column of $\mathbf{AW}$ defines the frequency response of the whitened spectrum $Y(k)$. The second column defines the frequency response of the noise spectrum $N(k)$ and allowing it to be negative leads to noise subtraction from the final spectrum $Z(k)$. The third column of $\mathbf{AW}$ is multiplied by the all-one spectrum in $\mathbf{Y}$ and allows an additive frequency-dependent curve to be added to the final spectrum $Z(k)$.

Crucial for the model are the parameters in matrix $\mathbf{W}$. The M parameters in column i of $\mathbf{W}$ (together with the fixed basis functions $\mathbf{A}$) determine the frequency response of column i in $\mathbf{Y}$. The basis functions are necessary in order to be able to represent the frequency responses with only a few parameter values. An algorithm for learning the parameters will be described in Sec. 2.4.

A harmonic transform is applied on the spectrum $Z(k)$ to obtain a "raw" salience function $r(\tau)$ which indicates the strength of pitch period candidates τ:

$$r(\tau) = \sum_{h=1}^{H} Z(k_{\tau,h}). \tag{5}$$

The period τ corresponds to the F0 value f_s/τ, where f_s denotes the sampling rate. The frequency bin $k_{\tau,h}$ corresponds to the h:th harmonic (integer multiple) of the F0 and is determined by the largest value of $Z(k)$ in the vicinity of the frequency hK/τ. More exactly, the maximum is found in range $\lfloor hK/(\tau + \Delta\tau/2) \rceil, \ldots, \lfloor hK/(\tau - \Delta\tau/2) \rceil$, where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer, K is the length of the Fourier transform, and $\Delta\tau = 0.5$ denotes the spacing between successive period candidates τ. The number of harmonic partials $H = 20$.

The harmonic transform (5) is motivated by the Fourier theorem which states that a periodic signal can be represented with spectral components at integer multiples of the inverse of the period. Pitch perception, in turn, is closely linked to the time-domain periodicity of sounds.

The function $r(\tau)$ contains peaks at the positions of true pitch periods, but it requires further processing to suppress peaks that often occur at integer (sub)multiples of the true period(s). The method proposed in the following bears resemblance to the "enhancing" technique of Karjalainen et al. [2] which suppresses the peaks at integer multiples of the true period(s) in the autocorrelation function (ACF). They clipped the ACF to positive values, scaled it to twice its length, and subtracted the result from the original clipped ACF. This was repreated for time-scaling factors up to about five to suppress the peaks occurring at integer multiples of the true period(s).

The method proposed in the following is a generalization of the above idea. First, let us create scaled versions of $r(\tau)$. The original function $r(\tau)$ is scaled by a factor j by inserting zeros between the original samples, lowpass filtering the result with cutoff frequency $\frac{1}{2}f_s/j$ and multiplying the filtered signal by j. The resulting signal, denoted by $r_j(\tau)$, is finally truncated to the same length as $r(\tau)$. Stretched versions with scaling factors $j = 2, 3, \ldots, J$ are calculated (here $J = 5$).

Secondly, we calculate shrunk versions of $r(\tau)$ by scaling with factor $1/j$. This is done by lowpass filtering $r(\tau)$ with cutoff frequency $\frac{1}{2}f_s/j$ and then copying every j:th sample of $r(\tau)$ to a signal denoted by $r_{1/j}(\tau)$. Since the length of $r_{1/j}(\tau)$ is only r:th fraction of $r(\tau)$, new values have to be calculated for long periods τ using (5) in order that the shrunk function would be of the same length as the original $r(\tau)$.

For convenience, the strecthed and shrunk versions of $r(\tau)$ are stored as columns in matrix $\mathbf{R}$,

$$\mathbf{R} = \left[\mathbf{r}, \mathbf{1}, \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_J, \mathbf{r}_{1/2}, \mathbf{r}_{1/3}, \ldots, \mathbf{r}_{1/J}\right] \tag{6}$$

where we have denoted $\mathbf{r} \equiv r(\tau)$, $\mathbf{r}_2 \equiv r_2(\tau)$, and so on for convenience, and $\mathbf{1}$ denotes an all-one vector.

Let us define basis functions

$$b_n(\tau) = [\log(\tau + 1)]^{n-1} \tag{7}$$

where τ is the period and $n = 1, 2, \ldots, N$ indexes the basis functions. This is a polynomial basis on the log-period scale. For convenience, the bases are collected as columns in a matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N]$.

The final pitch salience function $s(\tau)$ is calculated as a linear function of the columns of $\mathbf{R}$:

$$s(\tau) = (\mathbf{BV}.\times\mathbf{R})\,\mathbf{1} \tag{8}$$

where $\mathbf{V}$ is a matrix of size $(N \times 2J)$ that contains the model parameters, and $\mathbf{1}$ is an all-one vector of length $2J$ (the number of columns in $\mathbf{R}$). The product $\mathbf{BV}$ gives a matrix of the same size as $\mathbf{R}$. Its columns define period-dependent weights for the columns of $\mathbf{R}$, that is, for the original raw salience $\mathbf{r}$ and its stretched and shrunk versions, $\mathbf{r}_j$ and $\mathbf{r}_{1/j}$. For example, setting small negative weights for the columns that correspond to $\mathbf{r}_2$ and $\mathbf{r}_{1/2}$ implements suppression of the peaks that occur an octave above and below each true pitch period. The N parameters in column j of $\mathbf{V}$ (together with the fixed basis functions $\mathbf{B}$) determine the period-dependent weights for column j in $\mathbf{R}$. Finally, multiplication with $\mathbf{1}$ is equivalent to summing over the columns and yields $s(\tau)$.

The proposed pitch salience model is now fully defined, except for the two parameter matrices $\mathbf{W}$ and $\mathbf{V}$ in (4) and (8), respectively.

2.4 Algorithm for learning the parameters W and V

The described pitch salience model may look quite complicated at a first sight, therefore we start from a simplified case to develop an intuition how the model works. Let us set the parameters $\mathbf{W}$ in (4) to zero for all except the first column which corresponds to the first column of $\mathbf{Y}$, and let us set the values in the first column so that $Z(k) \approx \frac{K}{k}Y(k)$ where K is the Fourier transform length. Furthermore, let us set the parameters $\mathbf{V}$ in (8) to zero for all except the first column (which correspond to the first column of $\mathbf{R}$), and set the values so that $s(\tau) \approx \frac{1}{\tau}r(\tau)$. Substituting $r(\tau)$ from (5), the overall model becomes

$$s(\tau) \approx \frac{1}{\tau}\sum_{h=1}^{H}\frac{K}{k_{\tau,h}}Y(k_{\tau,h}) \approx \sum_{h=1}^{H}\frac{1}{h}Y(k_{\tau,h}). \tag{9}$$

where the latter equivalence is because $k_{\tau,h} \approx hK/\tau$. In other words, salience is computed as $\frac{1}{h}$-weighted sum of the partial amplitudes in the whitened spectrum $Y(k)$, which is a reasonable (although simplistic) way of computing pitch salience.

The above simplified model is actually exactly how the parameters $\mathbf{W}$ and $\mathbf{V}$ are initialized in the learning algorithm to be described here. The simple model (9) is a good starting point, from where we iteratively refine the values.

An overview of the learning algorithm is as follows:

0) Matrices $\mathbf{W}$ and $\mathbf{V}$ are initialized to values that correspond to the simple model (9).

1) Matrix $\mathbf{W}$ is updated, keeping $\mathbf{V}$ fixed.

2) Matrix $\mathbf{V}$ is updated, keeping $\mathbf{W}$ fixed.

3) Steps 1 and 2 are repeated until $\mathbf{W}$ and $\mathbf{V}$ converge.

In practice, it was found to be sufficient to repeat the steps 1 and 2 just a couple of times.

The exact goal of the optimization is to find such parameters $\mathbf{W}$ and $\mathbf{V}$ that the salience function $s(\tau)$ is as close as possible to unity value at points τ that correspond to true pitch periods, and as close as possible to zero at next-largest peaks that correspond to "false" pitch periods. The steps are now described in more detail.

Initialization. As already mentioned, $\mathbf{W}$ and $\mathbf{V}$ are initialized to values that correspond to the simple model in (9). Matrix $\mathbf{W}$ is initialized so that $Z(k) \approx \frac{K}{k}Y(k)$ and matrix $\mathbf{V}$ so that $s(\tau) \approx \frac{1}{\tau}r(\tau)$. The initial values in the first column of $\mathbf{W}$ are calculated by least-squares fit $\mathbf{w}_1 = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\alpha$, where vector $\alpha(k) = K/(k+\epsilon)$ denotes the target function and regularization using $\epsilon \approx 50$ is needed to avoid fitting only the largest values near the zero frequency. Similarly, the initial values in the first column of $\mathbf{V}$ are calculated by $\mathbf{v}_1 = (\mathbf{B}^\mathsf{T}\mathbf{B})^{-1}\mathbf{B}^\mathsf{T}\beta$, where vector $\beta(k) = 1/(\tau + \epsilon)$ denotes the target function and $\epsilon \approx 50$ is again needed to avoid fitting only the largest values near the zero period.

Updating W. In order to learn better values for $\mathbf{W}$, some training material is neeeded. For this purpose, we mixed samples from 32 musical instruments with equal mean-square levels. Random mixtures up to six simultaneous sounds were generated using the McGill University Master Samples (MUMS) database.

For each training instance g, $g = 1, 2, \ldots, G$, the following operations are performed:

1.a) The salience function $s(\tau)$ is calculated using (8) and the current parameters $\mathbf{W}$ and $\mathbf{V}$.

1.b) From $s(\tau)$, we record the exact period values of the P annotated true pitches in training instance g. In addition, we record the period values of $10-P$ next-largest "false" peaks in $s(\tau)$. The peak periods are denoted by τ_p, $p = 1, \ldots, 10$, and the types of the peak by $\phi_p = [1, \ldots, 1, 0, \ldots, 0]$ where 1 indicates true peaks and 0 the false ones.

1.c) Parameter-specific salience functions $s_{m,i}(\tau)$ are calculated using (8) and current $\mathbf{V}$ and special $\mathbf{W}$ which has value 1 at position $[\mathbf{W}]_{m,i}$ and 0 elsewhere.

1.d) For each true or false peak $p = 1, \ldots, 10$, the value of $s_{m,i}(\tau_p)$ is stored in matrix $\mathbf{Q}$ on row $p+10(g-1)$ and column $m+(i-1)M$. The peak type ϕ_p is stored in vector $\mathbf{c}$ on row $p + 10(g - 1)$.

After all instances g have been processed and the corresponding values stored in matrix $\mathbf{Q}$ and vector $\mathbf{c}$, updated parameters $\mathbf{w}$ are obtained by least-squares estimation

$$\mathbf{w} = (\mathbf{Q}^\mathsf{T}\mathbf{Q})^{-1}\mathbf{Q}^\mathsf{T}\mathbf{c}. \tag{10}$$

The corresponding matrix $\mathbf{W}$ is obtained by storing the $3M$ values in vector $\mathbf{w}$ to the three columns of $\mathbf{W}$. Equation (10) finds parameters which satisfy $s(\tau) \approx 1$ at the positions of "true" peaks, and $s(\tau) \approx 0$ for the false ones.

Updating V. Updating the matrix $\mathbf{V}$ is analogous to above. For each training instance g, the following operations are performed:

2.a) and 2.b) are identical to 1.a) and 1.b), respectively.

2.c) Parameter-specific salience functions $s_{n,j}(\tau)$ are calculated using (8) and current $\mathbf{W}$ and special $\mathbf{V}$ which has value 1 at position $[\mathbf{V}]_{n,j}$ and 0 elsewhere.

2.d) For each true or false peak $p = 1, \ldots, 10$, the value of $s_{n,j}(\tau_p)$ is stored in matrix $\mathbf{O}$ on row $p+10(g-1)$ and column $n+(j-1)N$. The peak type ϕ_p is stored in vector $\mathbf{d}$ on row $p + 10(g - 1)$.

After all cases g have been processed and the corresponding values stored in matrix $\mathbf{O}$ and vector $\mathbf{d}$, updated parameters $\mathbf{v}$ are obtained by least-squares estimation $\mathbf{v} = (\mathbf{O}^\mathsf{T}\mathbf{O})^{-1}\mathbf{O}^\mathsf{T}\mathbf{d}$. The corresponding matrix $\mathbf{V}$ is obtained by storing the $2JN$ values in vector $\mathbf{v}$ to the $2J$ columns of $\mathbf{V}$.

3. RESULTS

Figure 1 shows some example salience functions calculated for random sound mixtures using the proposed method (right panels) and, for comparison, for a baseline method (left panels). As a baseline method, we chose the salience function proposed in [4, Eq. (3)]. [2] The baseline method is practically identical to the simple model (9) which is used to initialize the parameter learning process here.

The two panels on top of Figure 1 show the output of the baseline and the proposed method for a single harmonic sound. The true pitch period is marked with a circle and the remaining largest false peaks are indicated with crosses. The proposed method is effective in suppressing the extraneous peaks to zero level (indicated by the horizontal line) and in forcing the true peak towards unity value. For curiosity, the next two panels show the outputs of the baseline and the proposed system for a single sinusoidal component. The last four panels show the output of the baseline system and the proposed system for a random combination of two and four sounds. As the polyphony increases, the proposed method too shows many spurious peaks although its result is still considerably cleaner that the baseline method.

Figure 2 shows precision, recall, and F-measure for the proposed method (solid line) and for the baseline method (dashed line) using synthesized MIDI signals as test material. The results were calculated by fixing a threshold value T_0, picking all the peaks in all frames above the threshold, and then calculating the resulting precision $\pi = \frac{C(\text{corr.})}{C(\text{det.})}$, recall $\rho = \frac{C(\text{corr.})}{C(\text{ref.})}$, and F-measure $\varphi = 2\pi\rho/(\pi+\rho)$. Here $C(\cdot)$, denotes the count of correct pitches found (corr.), count of all pitches detected (det.), or count of pitches in the reference (ref.). By varying the threshold, different precision/recall tradeoffs were obtained.

[2] The subsequent iterative detection and cancellation process in [4] was not used here, since it would lead to a discrete set of F0 values.

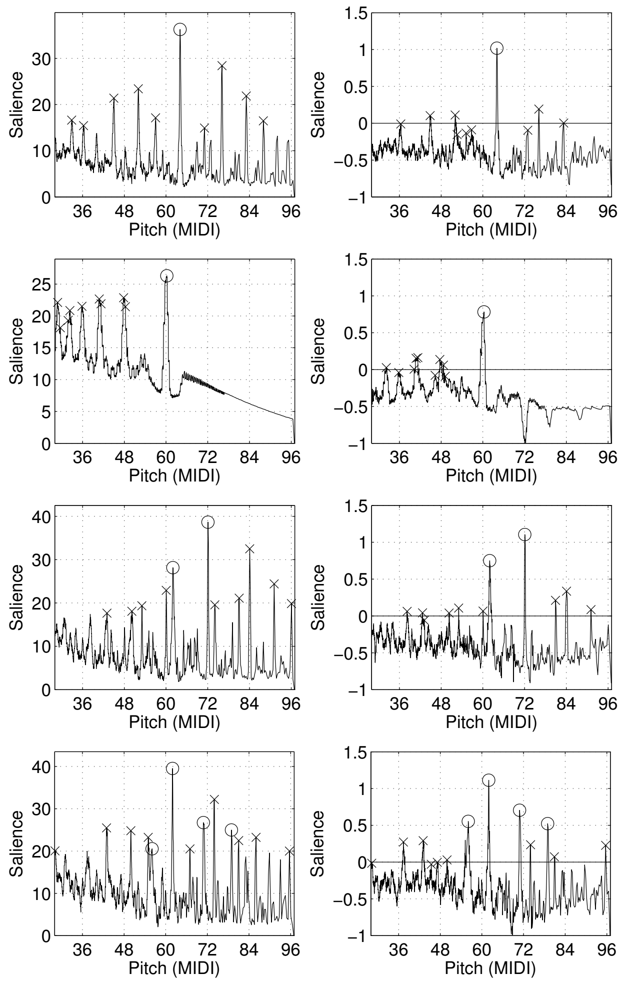


Figure 1. Example salience functions for the baseline method (left panels) and for the proposed method (right panels). The four cases from top to bottom represent 1) harmonic sound, 2) single sinusoidal component, 3) mixture of two harmonic sounds, and 4) mixture of four sounds. Peaks corresponding to the true pitch are circled.

The MIDI pieces were obtained by synthesizing random pieces from the RWC Pop and RWC Genre databases [8] and from midifarm.com. Synthesis of the MIDI files was used in order to ensure the correctness and synchronization between the synthesized file and the reference MIDI. Timidity software synthesizer and GeneralUser GS 1.4 soundfont were used for the synthesis. As can be seen in Fig. 2, the proposed method improves significantly over the baseline method. Here one should not pay too much attention on the absolute numerical values, since the polyphony of the pieces is quite high and especially the tails of long sounds can be very weak and difficult to detect.

4. APPLICATION TO PITCH VISUALIZATION

Figure 3 shows the computed salience $s(\tau)$ as a function of time for the piece No. 34 in RWC Popular Music database [8]. Here, the audio from the database was used instead of synthesizing from MIDI. The reference MIDI file is rendered on top of the salience function as boxes. In this

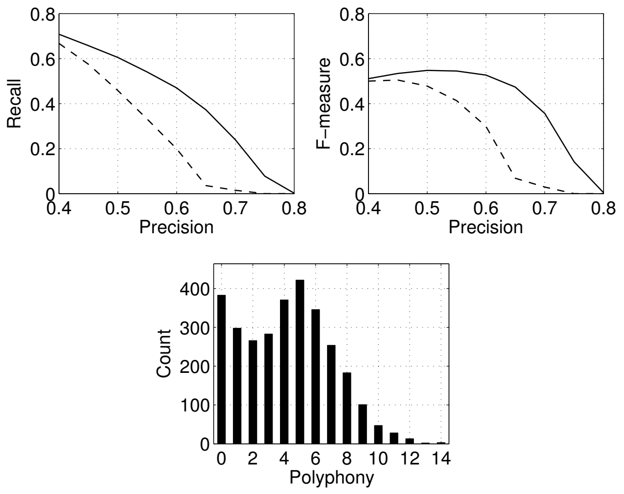


Figure 2. Precision, recall, and F-measure calculated for synthesized MIDI signals. Results are shown for the proposed method (solid line) and for the baseline method (dashed line). The third panel shows a histrogram of the number of concurrent sounds in the test data.

"piano roll" representation, the notes are arranged on the vertical axis and time flows from left to right.

Many people are not comfortable with reading music directly from a piano-roll. Therefore we propose here to map the data from the piano-roll to the traditional staves. This "fuzzy score" is very handy since it allows studying performance nuances, such as timing deviations and singing pitch glides and vibrato quite easily.

Figure 4 illustrates the mapping of different notes on the lines and spaces of the staves. Important to notice is that the note positions depend on the musical key of the piece, therefore key estimation is necessary to render the salience function on the staves. Here we used the key estimator from [9]. Secondly, the mapping is not linear: the distance between a line and its neighbouring space on the staves can be either one or two semitones. For this purpose, the piano-roll representation is "stretched" or "shrunk" to align with the staves.

Another requirement to make the score readable are bar lines which function as temporal anchors and make the timing of notes readable. Here we used the meter analysis method from [10]. The barlines are indicated with vertical lines and possible tempo changes appear as varying distances between the barlines.

Figure 5 shows the resulting "fuzzy score" representation for the same example that was shown in Fig. 3. The reference MIDI from the RWC database is drawn with circles on top of the salience. More examples of the computed fuzzy scores and the corresponding audio excerpts can be found at http://www.cs.tut.fi/sgn/arg/klap/ismir09/.

5. CONCLUSIONS

The proposed pitch salience model was shown to improve over the baseline method in terms of precision and recall when detecting multiple simultaneous pitches in synthe-

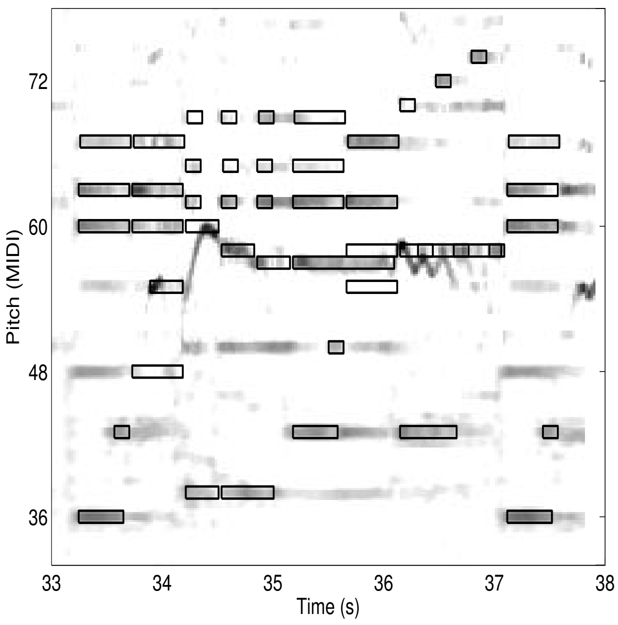


Figure 3. Computed pitch salience for an excerpt of piece No. 34 in RWC Pop database.

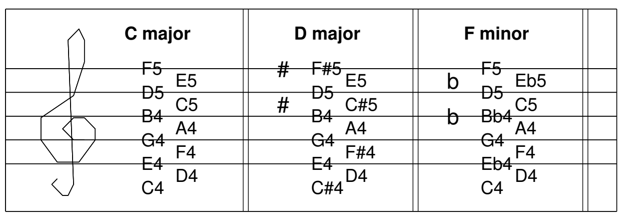


Figure 4. Mapping of pitch values on the staves positions in a few example musical keys.

sized MIDI files. This is due to the salience model which allows suppressing the peaks that occur at (sub)multiples of the true pitches in the salience function. The main advantage of the proposed method compared to many existing multipitch detection methods, however, is that it produces a continuous function that indicates the salience of all pitch candidates within a given range. This makes the proposed method particularly suitable for pitch content visualization. To this end, the proposed method was augmented with musical key and meter estimation methods which allow rendering the computed salience on the staves of common musical notation. [3]

6. REFERENCES

[1] H. Kameoka, T. Nishimoto, and S. Sagayama, “A multipitch analyzer based on harmonic temporal structured clustering,” *IEEE Trans. Audio, Speech, and Language Proc.*, vol. 15, no. 3, pp. 982–994, 2007.

[2] M. Karjalainen and T. Tolonen, “Multi-pitch and periodicity analysis model for sound separation and auditory scene analysis,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, USA, 1999.

[3] This work was supported by the Academy of Finland, project 129657, (Finnish Centre of Excellence Program 2006-2011).

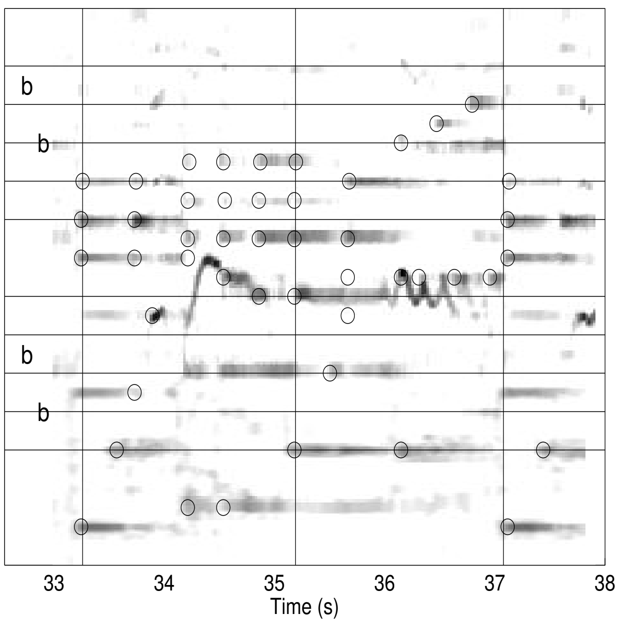


Figure 5. Computed pitch salience data rendered on the staves.

[3] C. Yeh, *Multiple fundamental frequency estimation of polyphonic recordings*, Ph.D. thesis, University of Paris VI, 2008.

[4] A. Klapuri, “Multiple fundamental frequency estimation by summing harmonic amplitudes,” in *Intl. Conf. on Music Information Retrieval*, Victoria, Canada, 2006.

[5] E. Vincent, N. Bertin, and R. Badeau, “Two nonnegative matrix factorization methods for polyphonic pitch transcription,” in *MIREX’07 Extended Abstracts*, 2007.

[6] D. P. W. Ellis and G. Poliner, “Classification-based melody transcription,” *Machine Learning*, vol. 65, no. 2–3, pp. 439–456, 2006.

[7] A. de Cheveigné, “Multiple F0 estimation,” in *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, D. Wang and G. J. Brown, Eds. Wiley–IEEE Press, 2006.

[8] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC music database: Music genre database and musical instrument sound database,” in *International Conference on Music Information Retrieval*, Baltimore, USA, Oct. 2003, pp. 229–230.

[9] M. Ryynänen and A. Klapuri, “Automatic transcription of melody, bass line, and chords in polyphonic music,” *Computer Music Journal*, vol. 32, no. 3, pp. 72–86, 2008.

[10] A. Klapuri, A. Eronen, and J. Astola, “Analysis of the meter of acoustic musical signals,” *IEEE Trans. Speech and Audio Processing*, vol. 14, no. 1, 2006.

PREDICTION OF MULTIDIMENSIONAL EMOTIONAL RATINGS IN MUSIC FROM AUDIO USING MULTIVARIATE REGRESSION MODELS

Tuomas Eerola, Olivier Lartillot, Petri Toiviainen
Finnish Centre of Excellence in Interdisciplinary Music Research,
University of Jyväskylä, Finland
`firstname.lastname@jyu.fi`

ABSTRACT

Content-based prediction of musical emotions and moods has a large number of exciting applications in Music Information Retrieval. However, what should be predicted, and precisely how, remain a challenge in the field. We provide an empirical comparison of two common paradigms of emotion representation in music, opposing a multidimensional space to a set of basic emotions. New ground-truth data consisting of film soundtracks was used to assess the compatibility of these models. The findings suggest that the two are highly compatible and a quantitative mapping between the two is provided. Next we propose a model predicting perceived emotions based on a set of features extracted from the audio. The feature selection and transformation is given special emphasis and three separate data reduction techniques are compared (stepwise regression, principal component analysis, and partial least squares regression). Best linear models consisting of 2-5 predictors from the data reduction process were able to account for between 58 and 85% of the variance. In general, partial least squares models performed the best and the data transformation has a significant role in building linear models.

1. INTRODUCTION

Emotional impact of music is one of the most important reasons for listening to music. A reliable content-based prediction of emotions in music would be a highly useful application of MIR, as suggested by the promising prototypes recently been put forward. It seems however that an improvement of the study would require a precise clarification of the concept under study, which is difficult due to the inherent fuzziness of the topic. Previous research defined mood as "sound and feel" of music (*AllMusicGuide*), of "feeling inspired by the music pieces" (*Last.fm*) [1]. Such broad opening of the study to a large realm of semantic expression, although interesting by itself, makes however the problem particularly difficult to tackle. Dealing with the concept of *emotion* instead, which is rooted on a large background of scientific research, would enable on the contrary a better controlled study of research.

However, robust and generalizable prediction of emotions has been difficult for several reasons, namely due to their conceptual elusiveness, their highly contextual dependencies on situation, context and musical style, and the limitations of the computational approaches utilised to date, which emphasize mainly on low-level acoustic features. The conceptual elusiveness of emotions is apparent in both the multitude of theoretical approaches taken, as well as the high individual variability in the subjective self-reports of emotional experiences. During the past decade, basic emotion model, dimensional models, and domain-specific emotion models have all received support in studies of music and emotion [2]. However, it still remains to be clarified whether models and theories designed for everyday emotions – such as the basic emotion model – can also be applied in an aesthetic context such as music. It has been argued, for example, that a few primary basic emotions seem inadequate to describe the richness of the emotional effects of music [3].

Current computational efforts of modelling polyphonic timbre seem to have reached what Aucouturier has called a 'glass-ceiling' effect, probably due to their strict reliance on low-level audio features. This ceiling appears to be around 50-60% of the variance explained [4]. Out of these three shortcomings, we aim to provide advances in two of them, namely by carrying simultaneous conceptual comparison of basic emotions and the circumplex model, and by performing the selection of relevant audio and musical features by means of multivariate methods.

2. BACKGROUND

2.1 Mood, emotion, and affect terms

Mood ontologies structure emotional adjectives and labels into a set of various mood clusters. Following purely theoretical studies [5, 6], more systematic approaches attempt to automatically infer the set of clusters based on analysis of large set of mood labels that are further reduced with the help of statistical tools: agglomerative hierarchical clustering of 179 AMG mood labels [1], consensus among a set of candidate labels used in literature [7, 8] collected through psychological experiments [9, 10], etc.

Representation of emotion in a dimensional affective space has gained support among researchers in music and emotion [2]. Instead of claiming that independent neural system exists for every basic emotion, the two-dimensional circumplex model [7] proposes that all affective states arise from two independent neurophysiological systems: one related to *valence* (a pleasure-displeasure continuum) and the other to *activity* (activation-deactivation). In contrast, Thayer [11] suggested that the two underlying dimensions of affect were two separate arousal dimensions: *energetic arousal* and *tense arousal*. However, the two-dimensional models have been criticized for their lack of differentiation when it comes to emotions that are close neighbours in the valence-activation space, such as anger and fear. It has also been discovered, that the two-dimensional model is not able to account for all the variance in music-mediated emotions [12] and three-dimensional variant containing valence, energy arousal and tension arousal has given better empirical results [13].

2.2 Ground truth collection

Extensive work has been carried out for the collection of ground truth related to mood ontology [10, 14]. Concerning the dimensional paradigm, Kim et al [15] have collected dynamic ratings expressed on the valence-activity space from thousands of songs drawn randomly from the uspop2002 database via a customized online game.

2.3 Mood and emotion prediction

Previous computational works attempt to predict mood clusters [16, 17] and emotion categories [18, 19]. Lu, Liu, and Zhang [20] studied mood detection and tracking using a variety of acoustic features related to intensity, timbre, and rhythm. Their classifier used Gaussian Mixture Models (GMMs) for Thayer's four principal mood quadrants in the valence-activity representation. The system was trained using a set of 800 classical music clips, each 20 seconds in duration, hand labeled to one of the 4 quadrants. Their system achieved an accuracy of 85% when trained on 75% of the clips and tested on the remaining 25%.

We believe that linear models are more useful than classifications for understanding emotion in music. Indeed, music is often emotionally ambiguous and listeners are not particularly certain of the emotion categories if given complex examples. Valence and activity mapping has been previously done [21, 22], but selecting the optimal set of features is more challenging, due to statistical constraints imposed by linear models.

3. NEW GROUND-TRUTH SET: SOUNDTRACKS

In the present work, both discrete and dimensional models of emotions are simultaneously investigated in order to clarify their mutual relationship and applicability to music and emotions. The three-dimensional model is used to collect data regarding the dimensional approach as it encompasses both lower dimensional models. In order to obtain a large sample of unknown yet emotionally stimulating musical examples, a selection of film soundtracks was used. Soundtracks are composed for the purpose of conveying powerful emotional cues, and may serve as a relatively 'neutral' musical material in terms of music preferences and familiarity. A three-part selection process was utilized. First, 12 experts chose 360 excerpts representing Happy, Sad, Tender, Scary and Angry emotions as well as different quadrants in the 3D affect space.

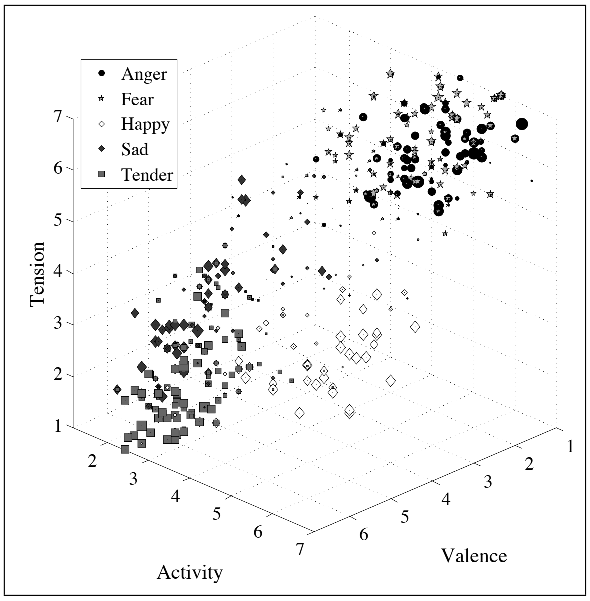


Figure 1. Average ratings of the three dimensions and basic emotions for the 360 soundtrack excerpts.

3.1 Evaluation

The expert panel (music students with extensive musical background) rated the examples, using both basic emotion concepts and dimensional ratings, on Likert scales (cf. Figure 1). Then a sampling of the 360 excerpts using both conceptual frameworks was carried out.

- For the basic emotion examples, the excerpts were categorized and ranked according to the basic emotion concept that received highest rating. From these ranked lists, the top five examples and five moderately high examples were chosen for each basic emotion (happiness, sadness, tenderness, anger and fear), yielding 50 basic emotion examples ([5 top + 5 moderate] × 5 categories).

- For the dimensional model, each dimension was sampled at 4 percentiles along its axis whilst the other two dimensions were kept constant, resulting in 60 audio examples that cover the affect space.

This set of 110 examples will be called *Soundtrack110* set hereafter. The mean duration of the excerpts was 15.3 seconds (SD 1.9 s).

In the next phase, 116 university students aged 18-42 years rated the Soundtrack110 set using both 3D set and

	3D	**2D**
	R^2 (β)	R^2 (β)
Happiness	.89 ($V_{.93}$,$A_{.79}$,$T_{-.35}$)	.89 ($V_{.85}$,$A_{.49}$)
Sadness	.63 ($V_{-.20}$,$A_{-.84}$,$T_{-.22}$)	.63 ($V_{-.05}$,$A_{-.69}$)
Tenderness	.77 ($V_{.33}$,$A_{-.45}$,$T_{-.58}$)	.74 ($V_{.50}$,$A_{-.51}$)
Fear	.87 ($V_{-.83}$,$A_{.07}$,$T_{.63}$)	.87 ($V_{-.90}$,$A_{.24}$)
Anger	.64 ($V_{-.52}$,$A_{.32}$,$T_{.35}$)	.68 ($V_{-.55}$,$A_{.35}$
Mean	.76	.76

Table 1. Ridge regression summary of dimensional models explaining basic emotion model categories. For instance, 89% of the variance (R^2) of Happiness can be explained with Valence (V) and Activity (A), with respective linear coefficient (β) .85 and .49.

basic emotions (on Likert scales). For the ensuing analyses, the means of the ratings across the participants were used as high consensus existed (Cronbach $\alpha > .99$ for each concept).

3.2 Basic emotions vs. dimensional ratings

As could be seen from the Figure 1, at least two emotion dimensions correlated heavily. In numerical terms, tension and valence correlate highly ($r = -.83$) and activity and tension in moderate way ($r = .57$), while valence and activity do not exhibit such a relation ($r = -.08$). The high correlation has implications in the task of constructing regression models for predicting categorical ratings based on the dimensional rating data, because multicollinear variables are problematic for standard versions of the regression. Hence we employed ridge regression since this technique is less influenced by collinearity due to the inclusion of constant variance parameter. This enables to attenuate the influence of collinearity in the calculation of the least squares optimization in regression. Ridge regression was used to predict the dimensional ratings from the categorical ratings and vice versa. The results – displayed in Tables 1 and 2 – demonstrate that the basic emotion model can more accurately explain the results obtained with the three-dimensional model than contrariwise. Nevertheless, the difference is not large (17%, the difference between the mean R^2 from the Tables 1 and 2) and this high degree of overlap between the conceptual frameworks suggests that the conceptual frameworks are highly compatible.

To further examine the validity of the three-dimensional model, its underlying coefficients of determination were also compared with the 2-dimensional circumplex model [7]. The results suggest that these two-dimensional models can explain the results obtained with the basic emotion model virtually as accurately as the three-dimensional model, with the exception of anger and tenderness (minor differences in R^2 values, see Table 1). It is worth pointing out that sadness was explained equally modestly (R^2 = .63), in comparison to other emotion categories, by all the dimensional models. This may reflect the participants' difficulty to rate the valence of sad music, for sadness in music is rarely perceived to represent an unpleasant emotion. Despite this irregularity, these analyses suggest fairly high mutual correspondence between the two conceptual frameworks and stimulus sets.

	Basic emotion model
	R^2 (β)
Valence	.97 ($H_{.35}$, $S_{-.11}$, $T_{.20}$, $F_{-.50}$, $A_{-.14}$)
Activity	.88 ($H_{.47}$, $S_{-.32}$, $T_{-.42}$, $F_{-.05}$, $A_{.36}$)
Tension	.93 ($H_{-.29}$, $S_{-.23}$, $T_{-.55}$, $F_{.18}$, $A_{.12}$)
Mean	.93

Table 2. Ridge regression summary of dimensional models explained by basic emotion model categories: Happiness (H), Sadness (S), Tension (T), Fear (F) and Anger (A).

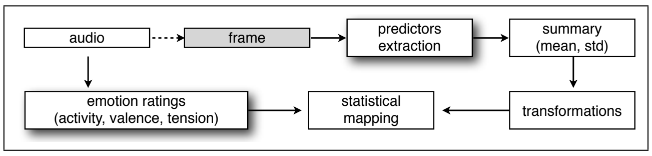


Figure 2. General design of the methodology.

4. AUDIO AND MUSIC FEATURE EXTRACTION AND TRANSFORMATION

The methodology proposed in this study is summarised in Figure 2: The *Soundtrack110* collection has been analysed using *MIRtoolbox* [23], and a set of features has been selected, explained below. We assume that a theoretical selection of features combined with a suitable data reduction techniques will result to the most parsimonious model. In addition, the features may require transformation to linearity before statistical mapping, described in the final section.

4.1 Theoretical selection of features

First, a theoretical selection is made based on the traditional categories of musical elements (rhythm, timbre, pitch, form, etc.) and by representing these categories by a few, non-redundant (non-correlating) features, in total 29. A synthetic description of the complete feature extraction process is given in Figure 3.

4.1.1 Timbre

Based on a spectrogram with a frame length of .046 s and half overlapping, three timbral descriptions are computed: centroid, spread and entropy, the latter predicting the presence of strong peaks. The mean correlation between features, computed using the *sountrack110 set*, is $r = .10$.

4.1.2 Harmony

The peaks configuration in the spectrogram enables to estimate a measure of roughness [24]. The entropy of each spectrum, collapsed into one single octave, indicates the presence of important chroma components. Or more precisely, the spectrum is turned into a chromagram, wrapped into one octave, and tonal information is computed – such

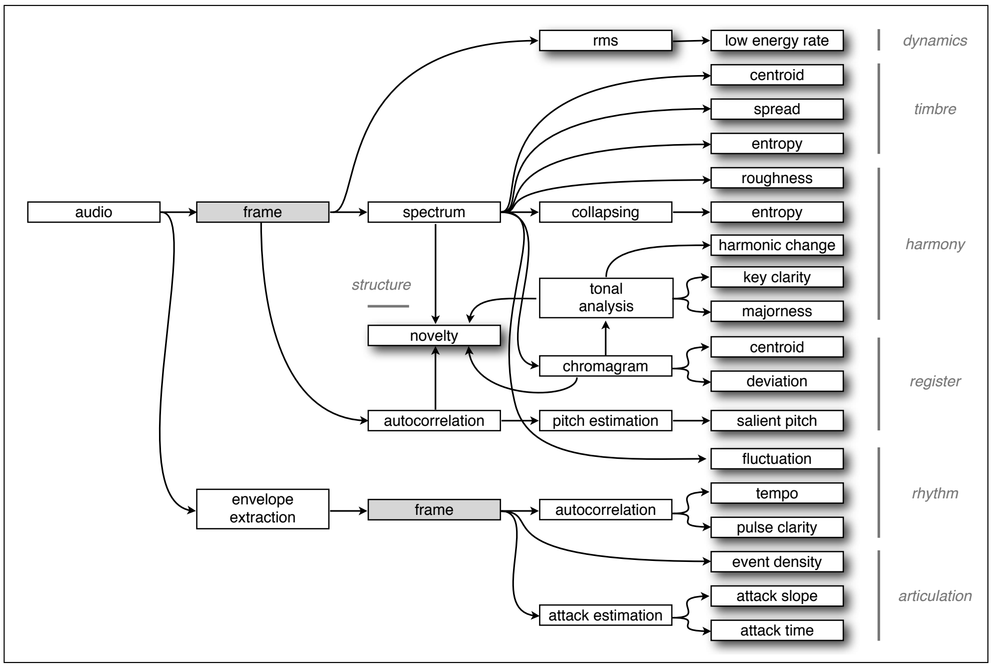


Figure 3. Flowchart of predictor extraction.

as key clarity or harmonic change [25] – based on tonal profile [26, 27]. We also designed a new measure of majorness, related to the difference of amplitude, observed on the tonal profile, between the best major score and the best minor score. For this dimension we obtain a within-feature correlation of $r = .04$

4.1.3 Register

Broad description of the localisation of pitch energy is performed through an estimation of the centroid and deviation of the unwrapped chromagram, and also in parallel a statistic description of pitch component based on advanced pitch extraction method [28]. $r = .27$

4.1.4 Rhythm

Rhythmic periodicity is estimated both from a spectral analysis of each band of the spectrogram, leading to a fluctuation pattern [29], and based on the assessment of autocorrelation in the amplitude envelope extracted from the audio. The clarity of the pulsation can also be assessed through an observation of the global characteristic of the autocorrelation function [30]. $r = .03$

4.1.5 Articulation

Onsets indicated by peaks picked from the amplitude envelope leads to the estimation of the relative amount of event density. For each successive onset, the slope and temporal duration of the corresponding attack phase is also estimated. $r = -.23$

4.1.6 Structure

The multidimensional structure of the pieces of music is estimated through the computation of novelty curves [31] based on various functions already computed such as the spectrogram, the autocorrelation function and the chromagram. $r = .85$

As a whole, the features represent the categories in a non-redundant way, as within-feature correlation is lower than .30, except for structural features.

4.2 Statistical selection of features

The second selection is based on statistical selection of relevant features, in which we compare Multiple Linear Regression (MLR) with a stepwise selection principle, Principal Component Analysis (PCA) followed by a selection of an optimal number of components, and Partial Least Squares Regression (PLS). Linear mapping via regression is known to be problematic as the predictors-to-cases ratio should be 1:10 or larger (we have 29 features, we would need at least 290 observations or more). Moreover, high number of predictors will probably be highly collinear, which is problematic for the establishment of a linear modeling of the data. Principal component analysis will eliminate the problem of collinearity, as the components are orthogonal and enables to use a low number of predictors (PCA components) in the regression. However, this data reduction method is not sensitive to the covariance between the features and the predicted data and thus may discard important features. The third technique, PLS regression [32], carries out simultaneous data reduction and maximization of covariance between features and predicted data, thus preserving any interesting correlational pattern between them. The output from the PLS is similar to PCA, individual, orthogonal components. To select the optimal number of features, Bayesian Information Criterion (BIC) was used.

	Prediction rate (R^2)		
Model	Valence	Activity	Tension
MLR	.64	.75	.67
PCA	.42	.74	.51
PLS	.70	.77	.71
MLR$_\lambda$	.66	.74	.69
PCA$_\lambda$	.51	.73	.63
PLS$_\lambda$	.72	.85	.79

Table 3. Prediction rates of the different models for circumplex model of emotions. $_\lambda$ denotes Box-Cox transformed variables.

	Prediction rate (R^2)				
Model	Angry	Scary	Happy	Sad	Tender
MLR	.46	.55	.46	.38	.38
PCA	.66	.67	.60	.59	.54
PLS	.66	.62	.61	.61	.50
MLR$_\lambda$	.56	.55	.63	.54	.45
PCA$_\lambda$	.56	.47	.53	.52	.45
PLS$_\lambda$	.70	.74	.68	.69	.58

Table 4. Prediction rates for the 5 basic emotions.

4.3 Data Transformation

To apply linear least-squares models, the distribution of the data should be approximately normal. Each feature was tested for normality (Lilliefors $p < .001$) and each non-normally distributed feature was transformed by means of Box-Cox power transform [33] by testing λ values between -2 and 2 in .1 increments and taking the one that yielded the maximal normality. Finally, all features were normalized.

5. RESULTS AND DISCUSSION

Table 3 displays the prediction rate of linear regression models using first 5 components in stepwise linear regression (MLR), and first 5 PCA components, and 2 first components from PLS, with or without data transformations ($_\lambda$). 5-fold cross-validation (80% for training, 20% for prediction) was used in all cases to avoid overfitting. In general, about 70 % of the variance in participants ratings could be predicted with features extracted from the audio. Data transformation has an important contribution to the models. MLR provides fairly successful model but it is problematic due to the serious over optimization stepwise regression does when using 29 predictors to explain 110 observations. PCA with 5 components has less power to predict the ratings but is nevertheless fairly adequate model. It suffers especially from the skewness and lack of normalization of the data. Finally, PLS (normalized) provides the highest prediction rate with only two components. The model adequacy is largely similar for basic emotions, displayed in Table 4.

The resulting predictive models vary depending on the chosen mapping method. Table 5 shows for instance the important features contributing to the perception of the categories of anger and tenderness, as predicted by the MLR method. The predictive models given by the PCA and PLS methods are less easy to represent clearly, are their underlying dimensions are formed by a high number of audio and musical features.

Anger		Tenderness	
Feature	β	Feature	β
Fluctuation peaks	-.14	RMS variance	-.44
Key clarity	-.07	Key clarity	.08
Roughness	.05	Majorness	-.08
Sp. centroid variance	-.04	Sp. centroid	-.05
Tonal novelty	.004	Tonal novelty	-.01

Table 5. Components and standardized beta weights of the MLR$_\lambda$ model for two chosen basic emotions.

When mapping the dimensional ratings onto each of the five basic emotions, the regression models could explain 63 to 89 percent of the variance. No significant improvement was observed with the 3D model over the 2D model, with the exception of anger, for which adding the third dimension increased the variance explained by five per cent. When mapping basic emotions onto the emotion dimensions, even higher proportions of variance could be explained by the models, these ranged from 88 to 97 per cent. These results suggest that there is a high mutual correspondence between the two emotion spaces.

Using a five-fold cross-validation, about 70% of the variance in the participants ratings could be explained by the PLS models. The highest proportion of variance explained (85%) was obtained when predicting activity with the PLS model using transformed features. We examined the effect of the Box-Cox transform on the predictive power of the regression models. In most cases this transform improved the models significantly. This observation suggests that the distributions of the extracted features are a crucial factor in the performance of such predictive models.

The emotion prediction model has been written in *Matlab* and has been integrated into the new version (1.3) of *MIRtoolbox* [23].

6. REFERENCES

[1] X. Hu and J. S. Downie. Exploring mood metadata: relationships with genre, artist and usage metadata. In *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[2] P. N. Juslin and J. A. Sloboda. *Music and Emotion: Theory and Research*. OUP, Oxford, UK, 2001.

[3] M. Zentner, D. Grandjean, and K. Scherer. Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion*, 8(4):494–521, 2008.

[4] J.-J. Aucouturier and F. Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[5] K. Hevner. Experimental studies of the elements of expression in music. *American Journal of Psychology*, (48):246–268, 1936.

[6] P. R. Farnsworth. *The social psychology of music*. The Dryden Press, 1958.

[7] J. A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.

[8] M. Leman, V. Vermeulen, L. De Voogdt, D. Moelants, and M. Lesaffre. Prediction of musical affect using a combination of acoustic structural cues. *Journal of New Music Research*, 34(1):39–67, 2005.

[9] J. Skowronek, M. McKinney, and S. van de Par. Ground-truth for automatic music mood classification. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 295–296, 2006.

[10] J. Skowronek, M. McKinney, and S. van de Par. A demostrator for automatic music mood estimation. In *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[11] R. E. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press, New York, USA, 1989.

[12] E. Bigand, S. Vieillard, F. Madurell, J. Marozeau, and A. Dacquet. Multidimensional scaling of emotional responses to music: The effect of musical expertise and of the duration of the excerpts. *Cognition & Emotion*, 19(8), 2005.

[13] U. Schimmack and R. Reisenzein. Experiencing activation: Energetic arousal and tense arousal are not mixtures of valence and activation. *Emotion*, 2(4):412–417, 2002.

[14] X. Hu, M. Bay, and J. S. Downie. Creating a simplified music mood classification ground-truth set. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 309–310, 2007.

[15] Y. E. Kim, E. Schmidt, and L. Emelle. Moodswing: A collaborative game for music mood label collection. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 231–236, 2008.

[16] T. Li and O. M. Ogihara. Detecting emotion in music. In *Proceedings of the International Symposium on Music Information Retrieval*, 2003.

[17] X. Hu, J. S. Downie, C. Laurier, M. Bay, and A. F. Ehmann. The 2007 mirex audio mood classification task: Lessons learned. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 462–467, 2008.

[18] D. Yang and W. Lee. Disambiguating music emotion using software agents. In *Proceedings of the International Symposium on Music Information Retrieval*, 2004.

[19] Y. Feng, Y. Zhuang, and Y. Pan. Popular music retrieval by detecting mood. In *Proceedings of the 26th annual international ACM SIGIR conference*, Toronto, 2003.

[20] L. Lu, D. Liu, and H.-J. Zhang. Automatic mood detection and tracking of music audio signals. *IEEE Trans.on Audio, Speech, and Language Processing*, 14(1):5–18, 2006.

[21] K.F. MacDorman. Automatic emotion prediction of song excerpts: Index construction, algorithm design, and empirical comparison. *Journal of New Music Research*, 36(4):281–299, 2007.

[22] Y.H. Yang, Y.C. Lin, Y.F. Su, and H.H. Chen. Music emotion classification: A regression approach. In *Proc. IEEE Int. Conf. Multimedia and Expo*, pages 208–211, 2007.

[23] O. Lartillot and P. Toiviainen. MIR in matlab (II): A toolbox for musical feature extraction from audio. In *Proceedings of 5th International Conference on Music Information Retrieval*, 2007.

[24] W. A. Sethares. , *Timbre, Spectrum, Scale*. Springer-Verlag, 1998.

[25] M. B. Sandler C. A. Harte. Detecting harmonic change in musical audio. In *Proceedings of Audio and Music Computing for Multimedia Workshop*, 2006.

[26] C. L. Krumhansl. *Cognitive foundations of musical pitch*. OUP, Oxford, UK, 1990.

[27] E. Gomez. *Tonal description of music audio signal*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2006.

[28] T. Tolonen and M. Karjalainen. A computationally efficient multipitch analysis model. *IEEE Transactions on Speech and Audio Processing*, 8-6:708–716, 2000.

[29] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 570–579.

[30] O. Lartillot, T. Eerola, P. Toiviainen, and J. Fornari. Multi-feature modeling of pulse clarity: Design, validation, and optimization. In *Proceedings of the International Symposium on Music Information Retrieval*, 2008.

[31] M. Cooper J. Foote. Media segmentation using self-similarity decomposition. In *Proceedings of SPIE Storage and Retrieval for Multimedia Databases*, volume 5021, pages 167–175, 2003.

[32] S Wold, M. Sjostrom, and L. Eriksson. Pls-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58:109–130, 2001.

[33] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society*, Series B 26:211–246, 1964.

OPTICAL AUDIO RECONSTRUCTION FOR STEREO PHONOGRAPH RECORDS USING WHITE LIGHT INTERFEROMETRY

Beinan Li
Music Technology Area
Schulich School of Music
McGill University
beinan.li@mail.mcgill.ca

Jordan B. L. Smith
Music Technology Area
Schulich School of Music
McGill University
jordan.smith2@mail.mcgill.ca

Ichiro Fujinaga
Music Technology Area
Schulich School of Music
McGill University
ich@music.mcgill.ca

ABSTRACT

Our work focuses on optically reconstructing the stereo audio signal of a 33 rpm long-playing (LP) record using a white-light interferometry-based approach. Previously, a theoretical framework was presented, alongside the primitive reconstruction result from a few cycles of a stereo sinusoidal test signal. To reconstruct an audible duration of a longer stereo signal requires tackling new problems, such as disc warping, image alignment, and eliminating the effects of noise and broken grooves. This paper proposes solutions to these problems, and presents the complete workflow of our Optical Audio Reconstruction (OAR) system.

1. INTRODUCTION

OAR has proven to be an effective contactless approach to digitizing monophonic phonograph records [1] [2] [3] [4]. Furthermore, it is an available solution for restoring broken records. Li et al. previously presented a theoretical framework for optically reconstructing audio with a white-light interferometry (WLI) microscope and image processing [5]. A few cycles of stereo sinusoidal signal, extracted from a small number of images, illustrated that their approach is capable of extracting stereo signals from LPs. To reconstruct a few seconds of audio, however, the scanning region must be scaled up to a much larger disc area, resulting in thousands of images. A sophisticated image acquisition and post-capture processing workflow is thus desired to tackle the challenges that emerge from large-scale scanning: e.g., disc surface warping, image alignment errors, groove damages, and unwrapping the grooves into a one-dimensional audio signal.

In Section 2, we review previous OAR systems. Our system to acquire record groove images is introduced in Section 3, followed in Section 4 by our image processing procedures for extracting audio from the scanned images. The reconstructed result is illustrated and discussed in Section 5.

2. EXISTING OAR APPROACHES

In this section, four previous OAR approaches are described. Although they operate on recordings of different formats, most OAR frameworks follow the same high-level three-step procedure for reconstructing an audio recording: first, the grooves are scanned; second, the groove undulations are isolated and extracted; third, these undulations are converted into audio. Approaches vary significantly in terms of the hardware used, some using a general-purpose commercial product such as a confocal microscope, others using a custom installation. The hardware, in turn, affects how the grooves are scanned and thus how groove undulations must be extracted. By contrast, the audio conversion step (which may include post-processing, such as equalization) depends solely on the record production procedures that were used for the particular item being scanned. This step almost always includes filtering the signal to undo the RIAA equalization used in production and obtain the audio.

The systems developed by Iwai et al. and Nakamura et al. use a ray-tracing method to obtain the groove contour of a phonograph record [6] [7] [8]. The groove is illuminated with a laser beam, and the groove undulations are measured by detecting the angle at which the beam is reflected. In this way the laser functions as a simulated stylus—a replacement for the mechanical stylus—and can output an analog audio signal directly.

Unfortunately, since such systems must trace out the grooves, they are unable to handle broken records. In addition, two types of errors limit this approach: the errors caused by the finite laser beam width, which leads to echoes and high- and low-frequency noise in the extracted audio signals, and the tracking errors that may occur when the beam misses the groove entirely.

Fadeyev and Haber built an OAR system for 78-rpm records based on confocal laser scanning microscopy [1]. With the help of a low-mass probe, they built another one for wax cylinders [2]. Their system is capable of scanning the record in 3D with a vertical accuracy of around 4.0 microns. However, in their work on 78-rpm records only 2D imaging is emphasized, at a resolution of 0.26 x 0.29 microns per pixel. It takes their system 50 minutes to scan about 1 second of recorded audio, corresponding to 0.5–5 GB of image data. The groove bottom is obtained using 2D edge detection on the pixel illumination data, and the groove undulation is defined as the radial deviation of the groove bottom with respect to an unmodulated trajectory about the centre of the record.

Lengths of the groove are skipped when dust and debris occlude the image and no edges are detectable, but no solution to data restoration is provided; such skipping may thus cause a loss of data. Fadeyev and Haber compared their optically reconstructed audio sample, a turntable-digitized audio sample, and a remastered CD sample of the same recording. The quality of the reconstructed sample was judged to be better than the turntable-reconstructed version, but poorer than the CD version: while the OAR system produced fewer clips and pops than the turntable, and had a lower continuous noise level, it also contained background hissing and low-frequency modulation absent from the CD version.

Stotzer et al. created a system that performs a multi-step optical reconstruction for 78-rpm records [3]. First, a 2D analog camera is used to rapidly photograph the records and the images are preserved on film. This film is then scanned and digitized into 2D digital images, which are further analyzed to extract the audio. During the scanning, the film is placed on a rotating tray with an overhead stationary camera that carefully captures the groove images passing through its field of view (FOV). The rotation of the tray simulates the rotation of the record during playback, and effectively unwraps the groove segments to become uniformly oriented in the resulting images. Similar to Fadeyev and Haver's strategy, edge detection is then used on these images to extract the groove undulations, which are simply described by the edge that separates the groove valley from the space between grooves.

In this system, the imaging resolution is compromised by shading blur, motion blur, and sampling blur introduced by the illumination and the rotating scanner. The blur is estimated to be roughly 24.6 microns along the direction of rotation. The system also suffers from various acquisition artifacts, such as the very low-frequency noise caused by the off-axis placement of the film.

In an effort to restore damaged grooves, smoothing and corrupted-pixel-map-based enhancement are performed. The robustness of the damage detection is nevertheless questionable due to the simplifying assumption that damages such as scratches are solely perpendicular to the grooves. The blurring described above also makes it difficult to reliably detect scratches in the grooves.

The reconstructed sound quality achieved by their system was evaluated according to several standard audio engineering parameters; for example, the signal-to-noise ratio of the system was found to be roughly 16dB.

Tian used 3D scanning based on dual-focal microscopy for reconstructing audio from 78-rpm records. The ability to handle LPs is claimed, but not yet implemented [4]. Contrary to the aforementioned approaches, the groove undulation is defined by the groove sidewall orientation at each tangential increment relative to the disc center, instead of by the edges of either the top or bottom of the groove. Ray-tracing is used to create a 3D image of the entire record groove surfaces, including the sidewalls. The stylus movement across the grooves is represented by the optical flow derived from groove image intensity derivatives. The sidewall orientations are obtained by using dense depth maps and projecting complex surface onto the groove cross-section plane. The microscope in use has a lateral resolution of 1μm per pixel. Tian's optical flow approach requires 1390 x 36 images of the FOV 640 x 480 pixels to represent a two-second audio signal, although on which groove revolution the reconstruction is performed is not reported. It takes three days for four workstations to generate the image representation of a three-second audio. The image acquisition time is unclear in the literature. The equivalent audio sampling rate in their experiment is about 2404.71 Hz.

Although laser turntables can serve as a solution to optically retrieve audio from phonograph records, they require emulating the exact groove-following behavior of a turntable. We would like to find a general image-acquisition-based preservation solution without mimicking turntable behavior to derive digital audio directly from images. We also wish to obtain 3D information: although Fadeyev and Haber claimed their system can be adapted to a 3D groove profile, they did not implement it, while the system of Stotzer el al. does not retrieve 3D information. Tian's system is 3D-based, but his experiments are not performed on stereo LPs, the target considered here. Thus, in contrast to these works, our research focuses on optically reconstructing a digital stereo audio signal from LPs by extracting the lateral and vertical groove undulations from 3D groove information.

3. WLI-BASED IMAGE ACQUISITION

WLI is a powerful scanning technique based on physical optics, as opposed to the geometrical-optics-based approaches that include confocal microscopy and ray-tracing. In WLI, a broadband light source simulates an ensemble of narrow-band interferometers to make high-precision measurements. We used a Wyko NT8000 WLI microscope equipped with both Michelson and Mirau interferometers. Adjusting the vertical focus of these allows one to perform vertical scanning interferometry with a vertical resolution of better than 1 nm [9]. (The amplitudes of the groove depth are typically on the order of 25–100 μm [3].) In the current experiment, only the Michelson interferometer (10X magnification) is used, with a 0.644 x 0.483 mm (or 640 x 480 pixel) FOV. Using the 3X vertical scan speed with 20% overlap between fields of view, it takes roughly 27 minutes to scan one second of audio content. This configuration provides a reasonable tradeoff between acquisition quality and time cost.

Due to warping of the disc surface, the phonograph record is not perfectly flat. Expanding the scanning range to span all possible groove depths is unfeasible because of the greatly increased time cost. On the other hand, re-estimating for each FOV the range of groove depths to scan takes too long as well. To minimize the time cost without risking unfocused images, a hierarchical scanning scheme was chosen: the entire grid of FOVs is divided into sub-regions, and the scanning depth is adjusted only for each sub-region. Compared to the approach with a global range, this adaptive scheme reduces the scanning time by half.

4. AUDIO EXTRACTION BY IMAGE PROCESSING

Once the images have been acquired, the next step is to extract from them the groove undulations, which can then be converted into audio. A stereo phonograph audio signal is encoded in lateral and vertical groove undulations, so both must be extracted. To do so, first the entire grid of FOVs must be realigned using a dynamic programming algorithm. Next, the structures that define groove undulations are identified (and restored, where damaged). Finally, the groove is traced over the entire field and unwrapped, permitting the straightforward extraction of the groove undulations. The extracted undulations may then be decoded into stereo audio. This workflow is illustrated in Figure 1.

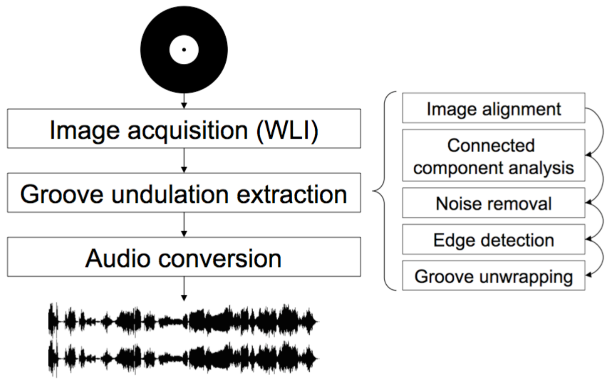


Figure 1. Diagram of the implemented system.

4.1. Image Alignment

As in most OAR systems, the FOVs are scanned with a degree of overlap. Unfortunately, mechanical translation during the image acquisition usually results in the images being misaligned. Before the information in the images can be extracted and combined, the FOVs must be realigned with each other.

This realignment can be achieved with an iterative frame-by-frame image registration approach, using either of two algorithms: a greedy one and a dynamic programming one [10]. In the greedy algorithm, the alignment of local FOV pairs is optimized, ignoring the precision of the overall grid alignment. It therefore can suffer from cumulative registration errors, resulting in unrecoverable gaps between the last few rows of the grid. By comparison, dynamic programming can be used to achieve a globally optimal alignment by forcing the elimination of any inter-row gaps as a constraint. This latter approach was selected.

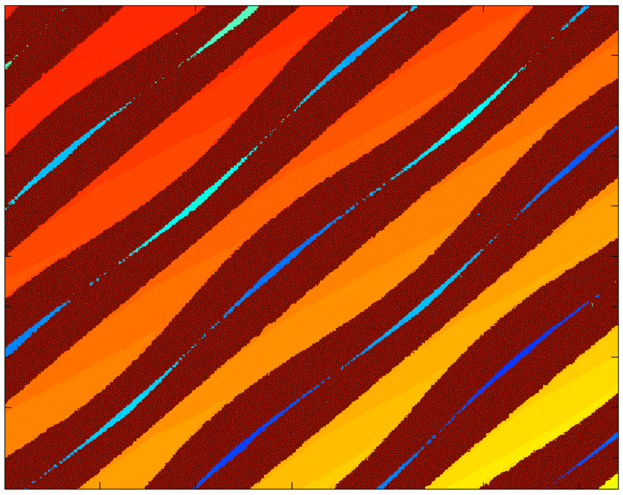


Figure 2. The top view of a typical groove image. The false colors represent vertical coordinates, except for the valleys, which are left monochromatic for clarity. The light color bands are the spaces between the grooves, and the light, thin lines are the valley bottoms.

4.2. The Image Model

Figure 2, a sample FOV, is a top view of a typical scanned region on a phonograph. The three parts of a groove can be seen readily: the tops (*T*), the valleys (*V*), and the bottoms (*B*). The goal is to extract the undulations of these grooves, but first the three parts have to be identified and separated. This is done using connected-component analysis (CCA). Similar to standard CCA in 2D image processing, a binarization process (thresholding) is performed as pre-processing to separate the global height levels of the *T*'s and *B*'s in the FOV; then with CCA, individual *T*'s, *V*'s and *B*'s are recognized.

The original 2D Cartesian coordinates are converted into polar coordinates (r, θ), with the origin being the disc center. We then distinguish three types of connected components (CCs): the top edges $T(r, \theta)$, the valleys $V(r, \theta)$, and the groove bottoms $B(r, \theta)$. Two useful properties are used in the identification of these regions: First, each region should be tangentially continuous, meaning that only one CC is supposed to be found on a single revolution of a *T*, *V*, or *B*. Second, the geometrical relationship between them is known: the *B*'s are completely contained by the *V*'s, while the T's and V's do not contain one another because they both stretch across the entire FOV.

4.3. Noise Removal and Groove Restoration

The raw CCs usually include noise from dust and dirt on the record and must be cleaned. Although heuristics based on the size of the CCs may seem intuitive, they turn out not to be robust in practice. Instead, we simply resort to the theoretical geometrical properties of the CCs, described in Section 4.2: since the tops and the valleys do not contain one another, any top found

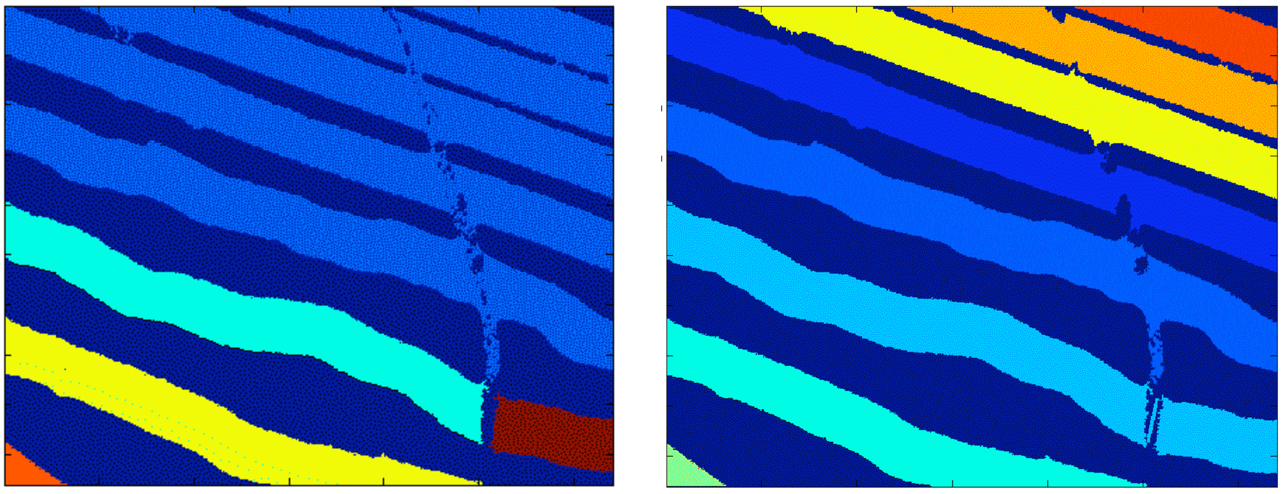


Figure 3. Groove restoration: CCA fails to identify the correct grooves (left) because the scratch connects certain grooves and disconnects others. These faults are corrected in the restored image (right).

contained by a valley (or vice versa) should be noise. This appeared to be a robust noise removal method.

Grooves may also be interrupted by scratches, or appear to be so due to occlusion by dust. Dust may also cause neighbouring grooves to appear attached to each other. These conditions create difficulties for extracting the groove undulations. To restore such grooves, two heuristics based on the tangential continuity property are used. First, by locating discontinuous $V(r, \theta)$ on the same revolution, broken grooves are detected; these may be restored by simply interpolating and reconnecting them. Similarly, attached grooves are detected and restored by locating and tangentially reconnecting discontinuous $T(r, \theta)$ that exist on the same revolution. Both situations are illustrated in Figure 3. Note that cleaning the records before scanning does not guarantee a better noise condition, because dust accumulates throughout the long time span of the scan. (Hosting the microscope in a dust-free environment may be one way to reduce this source of noise.)

4.4. Extracting Groove Undulations

The lateral undulations of the inner and outer groove top edges (with respect to the centre of the disc), and the vertical undulations of the groove bottom are the results of the recording stylus cutting into and across the disc surface. Following CCA and groove restoration, each FOV has been segregated into groove top, valley, and bottom regions. The next step is to trace the undulations, defined by the oscillations of both edges of the groove, as well as by the depth of the groove. Using a search-based edge detection algorithm, the inner and outer (with respect to the centre of the disc) groove top edges can be located; these are denoted as $T_i(r, \theta)$ and $T_o(r, \theta)$, respectively. The edges $T_i(r, \theta)$, $T_o(r, \theta)$, and the groove bottom $B(r, \theta)$ are iteratively overlapped and matched across adjacent, properly-aligned FOVs. In doing so, and in unwrapping the spiral-shaped grooves, the three 1D undulation sequences are obtained. The lateral undulations T_i and T_o are then averaged to obtain a single sequence T corresponding to the center of the groove. Note that phase unwrapping needs to be performed to obtain a continuous “time line” of the undulations, because the polar coordinates have the range of $[0, 2\pi)$. This is due to the fact that the derivation of polar coordinates is done to individual FOVs before their temporal topological order in the audio signal is clear.

4.5. Converting Groove Undulation to Audio

The raw, unwrapped groove undulations need to be resampled at an audio sampling rate to be converted to digital audio. Since the digital image format used here uses rectangular pixel tessellation, the pixel density along the undulation varies. It is therefore necessary to interpolate when sampling the image. We chose a reasonably high industry standard sampling rate (96kHz).

According to the constant velocity cutting scheme of LP production, the tangential velocities contain the audio undulation. This is achieved by performing numerical differentiation on the unwrapped undulations. The stereo audio is derived according to the following equations:

$$Channel_{left}(t) = \Delta T(t) - \Delta D(t)$$

$$Channel_{right}(t) = \Delta T(t) + \Delta D(t)$$

where $\Delta T(t)$ and $\Delta D(t)$ are the resampled and differentiated sequences of the center points and the depths of the groove, respectively. Finally, a counter-RIAA equalization filter is applied to the audio.

5. RESULTS AND DISCUSSIONS

In our experiment, the OAR system presented above was used to extract a roughly 1.8-second stereo audio signal. The result was compared to a turntable-digitized version of the same signal. Waveforms and frequency responses for both signals are displayed in Figures 4 and 5.

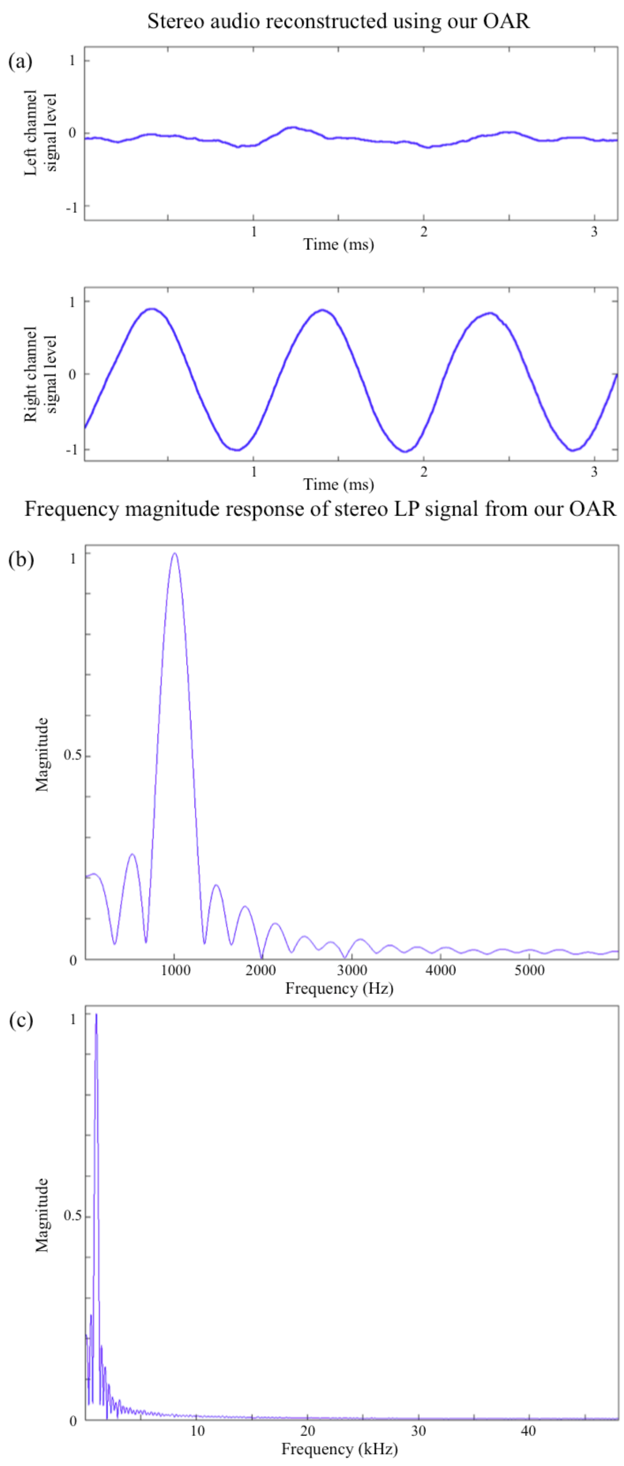


Figure 4. The reconstructed stereo signal: a 1kHz sine wave in the right channel and a silent left channel. A three millisecond segment of the waveform is shown in (a); in (b), the magnitude response of the extracted right channel signal up to 6kHz; in (c), the magnitude response of the same signal up to the Nyquist frequency (48kHz).

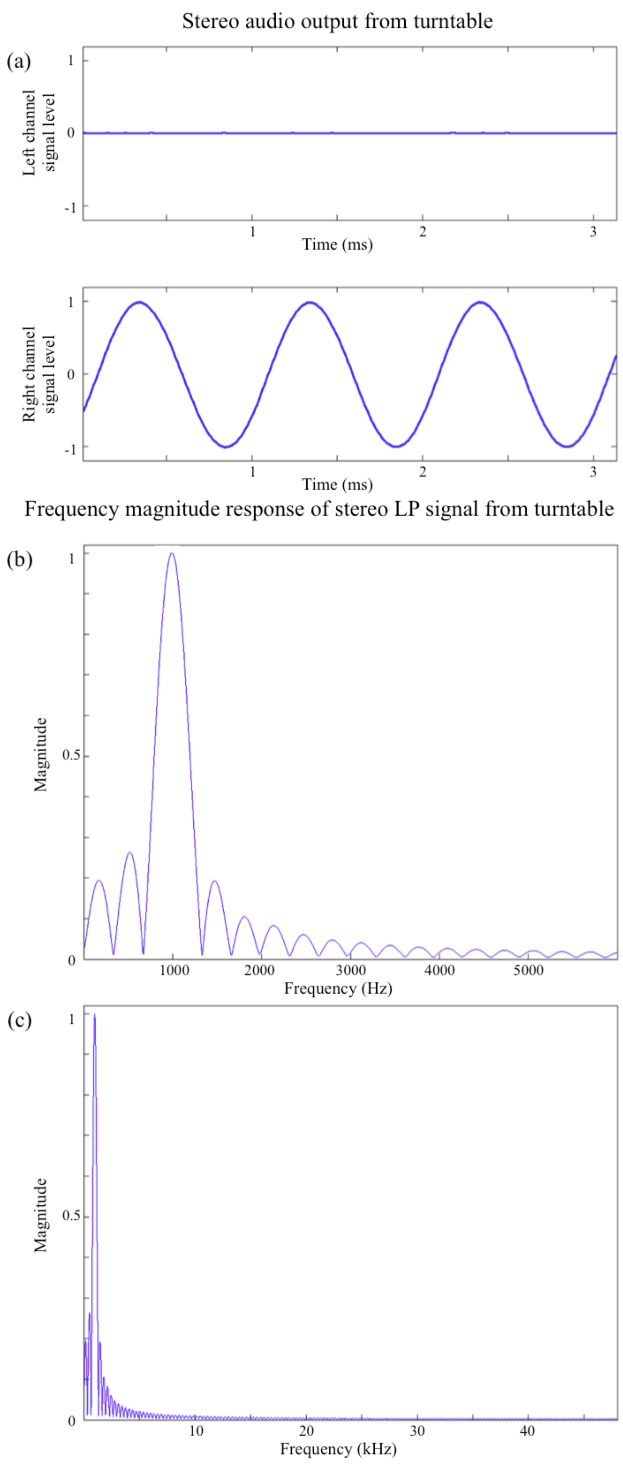


Figure 5. The turntable-digitized version of the same stereo signal.

It can be observed that the reconstructed version very much resembles the turntable-digitized version of the same stereo signal. As expected, the most salient component in the output stereo signals is 1kHz. However, the reconstructed left channel signal is not complete silence. In addition, close inspection of the peaks and troughs of the reconstructed waveform reveals a non-zero DC offset; moreover, the peaks appear to fluctuate slightly over time. This low-frequency wow noise, as in Fadeyev and Haber's system, is partly due to the error in estimating the disc center, which, as the origin of the polar coordinate system, forms the basis of the estimated lateral groove undulations.

6. CONCLUSIONS AND FUTURE WORK

OAR methods have been proven to be effective alternatives in digitizing mono phonograph records. Our WLI-based OAR system has successfully reconstructed digital stereo audio signals from LPs. Future work will be directed to improving the audio quality while decreasing the scanning time. Better center correction strategies will be studied, along with other configurations capable of pushing the audio quality higher. To push down the tremendous time costs, we will also investigate the minimum scanning resolution required to produce an acceptable audio result. On the other hand, the time costs may be reduced as the scanning hardware improves to provide, for instance, a larger field of view and faster vertical scanning speed.

7. ACKNOWLEDGEMENTS

We would like to thank the Canada Foundation for Innovation and the Daniel Langlois Foundation for their financial support.

8. REFERENCES

[1] Fadeyev, V., and C. Haber. Reconstruction of mechanically recorded sound by image processing. *LBNL Report* 51983, 2003.

[2] Fadeyev, V., C. Haber, C. Maul, J. McBride, and M. Golden. Reconstruction of recorded sound from an Edison cylinder using three-dimensional non-contact optical surface metrology. *LBNL Report* 54927, 2004.

[3] Stotzer, S., O. Johnsen, F. Bapst, C. Sudan, and R. Ingold. Phonographic sound extraction using image and signal processing. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* 4:289–92, Montreal, Canada, 2004.

[4] Tian, B. *Reproduction of sound signal from gramophone records using 3D scene reconstruction*. Ph.D. thesis. Department of Computer Science, The University of Western Ontario, 2006.

[5] Li, B., S. de Leon, and I. Fujinaga. Alternative digitization approach for stereo phonograph records using optical audio reconstruction. *Proceedings of International Symposium of Music Information Retrieval* 165–6, Vienna, Austria, 2007.

[6] Iwai, T., T. Asakura, T. Ifubuke, and T. Kawashima. Reproduction of sound from old wax phonograph cylinders using the laser-beam reflection method. *Applied Optics* 25 (5): 597−604, 1986.

[7] Nakamura, T., T. Ushizaka, J. Uozumi, and A. Toshimitsu. Optical reproduction of sounds from old phonographic wax cylinders. *Proceedings of SPIE* 3190: 304−13, 1997.

[8] ELP Corporation. ELP Laser Turntable: plays vinyl records without a needle. http://www.elpj.com/main.html (accessed 10 August 2009).

[9] Veeco Metrology Group. *WYKO NT8000 setup and operation guide*. Tucson: Veeco Instruments, Inc., 2003.

[10] Chow, S., H. Hakozaki, D. Price, N. MacLean, T. Deerinck, J. Bouwer, M. Martone, S. Peltier, and M. Ellisman. Automated microscopy system for mosaic acquisition and processing. *Journal of Microscopy* 222 (2): 76–84, 2006.

SONG RANKING BASED ON PIRACY IN PEER-TO-PEER NETWORKS

Noam Koenigstein **Yuval Shavitt**
School of Electrical Engineering
Tel Aviv University, Israel
{noamk, shavitt}@eng.tau.ac.il

ABSTRACT

Music sales are loosing their role as a means for music dissemination but are still used by the music industry for ranking artist success, e.g., in the Billboard Magazine chart. Thus, it was suggested recently to use social networks as an alternative ranking system; a suggestion which is problematic due to the ease of manipulating the list and the difficulty of implementation. In this work we suggest to use logs of queries from peer-to-peer file-sharing systems for ranking song success. We show that the trend and fluctuations of the popularity of a song in the Billboard list have strong correlation (0.89) to the ones in a list built from the P2P network, and that the P2P list has a week advantage over the Billboard list. Namely, music sales are strongly correlated with music piracy.

1. INTRODUCTION

Peer-to-peer (P2P) networks are one of the internet's most popular applications. The number of users and traffic, is growing dramatically from year to year. Despite several recent high profile legal cases against P2P vendors and users, it seems that the P2P community at large remains strong and healthy. In fact, P2P networks gain more acceptance as many companies and organizations distribute software and updates via networks such as BitTorrent to save bandwidth (e.g., Ubuntu).

Some studies suggest that music piracy might increase legal sales [1, 2], and copyright owners are advised to start developing business models that will allow them to generate revenue from P2P activity. Pioneering suggestions to utilize P2P networks for the benefit of the music industry were made by Bhattacharjee *et al.* [3,4], where P2P activity was used to predict an album's life cycle on the Billboard's top 200 albums chart.

In our previous work [5] we showed how P2P queries can be used for early detecting unknown emerging artists. In this study we take a different approach; we suggest an alternative songs ranking based on file sharing activity, that might replace traditional artists ranking such as the Billboard. We measured music piracy using a data set of geographically identified P2P query string, and compared it to songs ranking on the Billborad Hot 100, which measures sales and air plays. We compiled popularity charts based on P2P activity, and show a strong correlation between music piracy and legal sales and air plays. We argue that ranking songs through measurement of P2P queries is a good predictor of peoples' taste, and has many advantages over other means of popularity ranking, which were suggested in the past, most notably using social networks [6].

The remainder of the paper is organized as follows: In Section 2 we introduce the data set used in this study, and the methodology used to collect it. In Section 3 we focus on comparing song popularity in P2P networks with their ranking on the Billboard. We discuss the significance of our finding and our conclusions in Section 4.

2. DATA-SETS AND METHODOLOGY

We use two data sources for this study:

- **P2P Search Queries**: A data-set of queries collected from the Gnutella file-sharing network over twenty three weeks from January the 7th 2007 to June 8th 2007.
- **The Billboard Hot 100** The Billboad Hot 100 weekly charts for 2007 as published by the Billboard Magazine.

These two data-sets were collected independently, yet this study reveals a strong relationship between them. However, before analyzing the commonalities and differences, let us first describe the data sets and the methodology used to collect them.

2.1 P2P Search Queries

Queries in a file sharing network represent their users current taste and interests. A query is issued upon a request by a user searching for a specific file, or content relevant to the search string. In this study we used data collected from the Gnutella network using the Skyrider systems [1] . This data-set and the technical details of the methodology used to collect it are described in more depth in [7].

[1] Skyrider was a startup company that developed file sharing applications and services. It has recently been closed down. The data-set was collected when the company was still active, and is available for academic research.

2.1.1 *The Gnutella File Sharing Network*

In a study performed by Slyck.com, a website which tracks the number of users of different P2P applications, Gnutella was found among the three most popular P2P file-sharing applications together with eDonkey and FastTrack [8]. Furthermore, according to [9], Gnutella is the most popular file sharing network in the Internet today with a market share of more than 40%. It is mainly used for piracy of music. In [5] the top 500 most popular queries were manually classified, and it was found that 68% of the queries were music related. Together with adult content (22%), these two categories dominate the query traffic, accounting together for 90% of the queries. Gnutella is also among the most studied P2P networks in the literature [5, 7, 10–16].

2.1.2 *Methodology*

A query's origin IP address is required for its geographical classification according to its country of origin. While it is possible to capture a large quantity of Gnutella queries by deploying several hundred ultrapeer nodes [2], it will not be possible to tell the origin IP address of most of these captured queries. The basic problem in identifying the origin of captured queries is that queries do not in general carry information regarding their origin. What they do usually carry is an “Out Of Band” (OOB) return IP address. This address allows clients that have content matching a query to respond to a location close to the origin of the query, without having to backtrack the path taken by the query message. However, as most queries come from firewalled clients, in most cases the OOB address will belong to the ultrapeer connected to the query origin, acting as a proxy on behalf of the query originator. Deducting the missing origin IP address is not trivial. We resolved this problem by using a hop counting technique that is further explained in [7].

The vast majority of the Gnutella network is comprised of Limewire clients (80%-85%) and Bearshare clients (6%-10%) [10]. The Limewire client does not allow users to perform any kind of automatic or robotic queries. It does not allow queries with the SHA1 extension [3], nor does it allow the automatic re-sending of queries. When it does send duplicate queries, it uses a constant Message ID which enables a simple removal of any duplication. By recording only queries originating from Limewire clients, we were able to significantly reduce the amount of duplications and automatic (non-human) queries, without losing too much of the traffic. Capturing only Limewire queries is an easy task as Limewire “signs” the message ID associated with each message it sends. This signature can be easily verified by the intercepting node, allowing it to ignore queries from all other clients.

[2] Ultrapeer nodes are special nodes that route search queries and responses for users connected to them

[3] SHA1 queries are queries in which only the hash key of a known file is sent without a string. This is useful when a client already started downloading and needs more sources.

Rank	String	Occurrences
1	adult	36,290
2	akon	23,468
3	lil wayne	12,518
4	beyonce	11,987
5	this is why i'm hot	10,746
6	justin timberlake	10,193
7	porn	9,144
8	don't matter	9,047
9	fergie	8,979
10	fall out boy	8,077

Table 1. P2P popularity chart for week 9 of 2007

2.2 Data Set Statistics

A daily log file of queries, typically contained 25-40 million record lines, each line consists of the query string, a date/time field, and the IP address of the node issuing the query. The origin country for each query was resolved using MaxMind commercial GeoIp database. Similarly to the Billboard charts, we wanted to concentrate on data originated from the United States. We thus removed all the non US queries reducing 55%-60% of the data records.

Our data-set comprised of query strings collected over a period of 23 weeks from January the 7th 2007 to June 8th 2007. The activity on the Gnutella networks increases by 20%-25% over the weekend [7]. We thus used weekly samples taken on a Saturday or a Sunday of every week of that period. The total number of US originated query strings processed in this study is **185,598,176**.

2.3 The Billboard Hot 100

The Billboard Hot 100 is the United States music industry standard singles popularity chart issued weekly by Billboard magazine [17]. Chart rankings are based on radio play and sales data collected 10 days before the chart is released. The ranking process does not take into account file sharing activity. A new chart is compiled and officially released to the public each Thursday. The chart is dated with the week number of the Saturday after, but in this study we used dates and week numbers according to the actual release date of the chart, and ignored the date issued by Billboard magazine. To simplify time tracking in this paper, we use week numbers instead of full date to chronologically order the Billboard charts and the weekly file sharing data we collected. For example, the Billboard chart which was released on Thursday January 11th 2007 (week number 2), was dated by billboard to January 20th (week 3) but by us to week number 2. The current top 50 singles are published weekly on the magazine website, while the full historical charts are available to on-line subscribers for a small fee. A statistical model of songs ranking in the Hot 100 chart can be found in [18].

3. CORRELATION OF TRENDS

As described above, the Billboard Hot 100 chart ranks songs relative to each other, and does not reveal the number of sales or air-plays measured during that week. In order to compare it to our file-sharing data, we compiled our own weekly P2P popularity charts based on the popularity of search strings. We measured the popularity of each string by aggregating the number of appearances intercepted from a US based origin on that week. Table 1 depicts the top 10 positions of the P2P chart generated on week 9 of 2007 (sampled on March 1 2007).

Obviously, the P2P charts include many non music related strings. The string “adult” for example, was ranked number one on every chart we compiled. Unlike the Billboard charts, the P2P charts included also artists names (not only single titles), and sometimes even different variations of the same strings. In order to avoid inaccuracies, we looked only at the position of a song's exact title in the chart. To have high probability that the Billboard songs are ranked on our chart, we compiled truncated charts of the top 2000 strings each. A weekly log file contained on average 1.73 million different strings. Therefore, the top 2000 is approximately one thousandth of the entire P2P popularity chart. The top songs of our P2P chart were queried about 300,000 times per week in the USA. The songs at location 2000 were queried about 4,500 times. The number of queries per rank follows Zipf's law [7], thus changes in a rank position indicate strong shifts in popularity. When a song is no longer on the top 2000, it exits the P2P chart. This however, doesn't mean it is no longer being downloaded. Similarly when a single exits the Billboard Hot 100 chart, it doesn't mean it is not being played on the radio or sold in stores. Therefore, when considering the correlation of trends between the two charts, one should focus on the weeks where a song is ranked on both charts.

3.1 Correlation Measurements

We define $\overline{B_s}$ and $\overline{P_s}$ as the chart vectors representing the song s on the Billboard and P2P chart respectively.

$$\overline{B_s} = \{b_s(1), b_s(2), ..., b_s(23)\} \quad (1)$$

$$\overline{P_s} = \{p_s(1), p_s(2), ..., p_s(23)\} \quad (2)$$

Where $b_s(w)$ and $p_s(w)$ are the positions of song s on the Billboard and the P2P chart on week w respectively. If song s was not in the chart, we set its position to ∞ for that week. The *support* of a chart vector is the time range that the song was ranked in the chart. Namely where $b_s(w) < \infty$ or $p_s(w) < \infty$. The *joint support* of a song s is the time range in which it simultaneously ranked in both charts.

Fig. 1 depicts the chart vectors $\overline{B_s}$ and $\overline{P_s}$ for 6 different songs. The solid blue graph is the song's ranking on the Billboard Hot 100, while the dashed green graph is the song's ranking on the P2P chart. The horizontal axis (x-axis) depicts the date measured in week numbers in 2007. The song titles and performing artists are written above each graph. Note that lower parts of the graph represent higher position on the charts (i.e., the top of the chart is 1, while the last place is 100 or 2000). Looking at Fig. 1, one can easily notice the correlation between these two time series. This correlation is vivid not only in the general trend of the line, but also in minor trends and fluctuations.

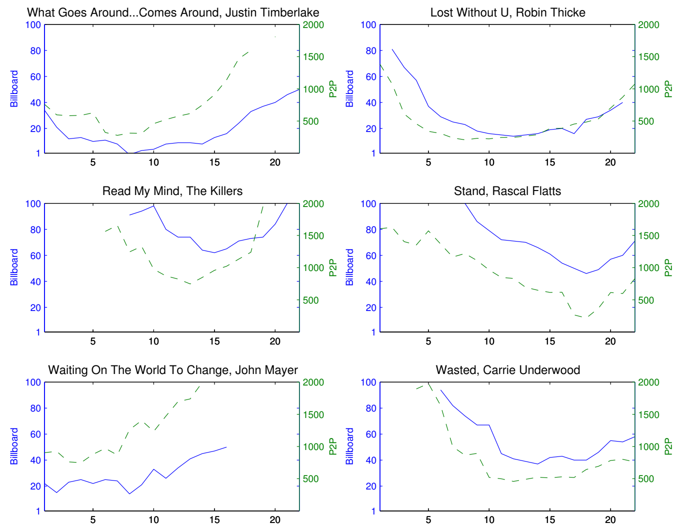


Figure 1. P2P Popularity Chart vs. The Billboard Hot 100

We slightly altered the standard definition of cross-correlation to consider only the joint support of the two series $\overline{B_s}$ and $\overline{P_s}$:

$$corr = \frac{\sum_{i=w_s}^{w_e} [(b_s(i) - E\{\overline{B_s}\}) \cdot (p_s(i) - E\{\overline{P_s}\})]}{\sqrt{\sum_{i=w_s}^{w_e} (b_s(i) - E\{\overline{B_s}\})^2} \sqrt{\sum_{i=w_s}^{w_e} (p_s(i) - E\{\overline{P_s}\})^2}} \quad (3)$$

Where $[w_s, w_{s+1}, ..., w_e]$ is the joint support and $E\{\overline{B_s}\}$ and $E\{\overline{P_s}\}$ are the means of the corresponding series. The correlation coefficient is in the range of $-1 \leq corr \leq 1$, where the bounds indicating exact match up to a scaling factor, while 0 indicates no correlation.

In all our measurements, we required songs to have a joint support of at least 4 weeks. This is the majority of the date-set (over 80%). Songs with a joint support of less than 4 weeks are mainly songs that ranked before or after our measurements, and had only a short “tail” inside our measurement period. Such songs poorly represent correlation of popularity trends over time.

We measured the correlation coefficients of the 135 songs that had a joint support of at least 4 weeks within the first twenty three weeks of 2007. The average joint support was 10.9 weeks. The average correlation coefficients was 0.67 while the median was 0.82, indicating a very strong correlation.

One might argue that the high correlation coefficients are the result of trend similarities of any time series of songs on charts. We thus measured the cross-correlation coefficient between the songs in one chart, and a random permutation in the other chart. Of the 52 songs which had a joint support of at least 4 weeks, the average joint support was 9.72 weeks, the average of the correlation coefficients was -0.006, and the median was 0.023, which negates the above hypothesis.

Title	Artist	No Shift	One Week
What Goes Around...Comes Around	Justin Timberlake	0.729	0.9707
Lost Without U	Robin Thicke	0.7664	0.948
Read My Mind	The Killers	0.1764	0.661
Stand	Rascal Flatts	0.9617	0.8965
Waiting On The World To Change	John Mayer	0.723	0.8965
Wasted	Carrie Underwood	0.8611	0.9456

Table 2. Correlation coefficients of the songs in Fig. 1

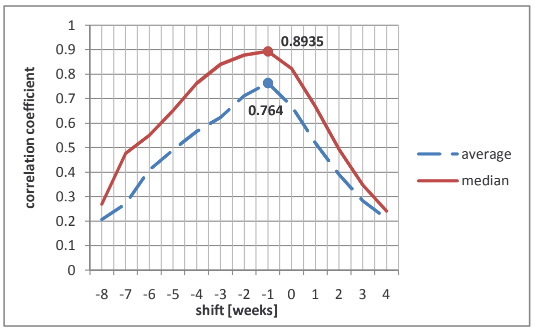


Figure 2. Cross-Correlation Coefficients vs. Time Shift

As mentioned is Section 2, the Billboard charts were dated according to their release date. However, the data used to compile each chart, is collected during the 10 days before the chart is published. Thus, we were interested in the correlation coefficient between the P2P chart and the Billboard chart of the following week. By shifting the Billboard chart vectors backwards, we measured the correlation coefficients of the 130 songs with a joint support of at least 4 weeks. The average joint support was 10.8 weeks. The average correlation coefficient was 0.76, while the median value was 0.89. These values are higher than the previous ones, which indicate a short time shift between the two series. Fig. 2 depicts the average and median values of the correlation coefficients, as a function of the Billboard's time shift. Clearly, minus one is the optimal time shift. We thus conclude that trends on the Billboard chart and on the the P2P charts are highly correlated with the Billboard lagging by one week. Table 2 depicts the correlation coefficients of the example songs in Fig. 1 without shifts, and with a one week time shift. When carefully examining Fig. 1, this time shift is noticed on some of the song graphs. The implication of this finding is obvious: P2P popularity charts can be used in order to predict trends on the Billboard chart. Record companies, for example, might use P2P file sharing activity to improve their marketing decisions.

3.2 Ranking Drift Analysis

In Section 3.1 we showed that songs trends (a climb or a descend) in P2P popularity charts are highly correlated with trends on the Billboard Hot 100. We now ask whether the charts are similar also in the relative ranks of songs. For each week we took the 100 songs from the Billboard chart, and "re-ranked" them according to their relative position on the P2P chart. In accordance with Section 3.1, we used a time shift of one week. We thus created an alternative ranking chart for the Billboard songs based on their P2P activity. This alternative chart is actually a filtered version of P2P chart from Section 3.1 that contains only the songs from the Billboard chart.

Rank	Billboard	Alternative Chart
1	*Irreplaceable*, Beyonce	*Walk It Out*, Unk
2	*I Wanna Love You*, Akon Feat. Snoop Dogg	*You*, Lloyd Feat. Lil Wayne
3	*Fergalicious*, Fergie	*Tim McGraw*, Tim McGraw With Faith Hill
4	*Smack That*, Akon Feat. Eminem	*Smack That*, Akon Feat. Eminem
5	*Say It Right*, Nelly Furtado	*We Fly High*, Jim Jones
6	*My Love*, Justin Timberlake Feat. T.I.	*Runaway Love*, Ludacris Feat. Mary J. Blige
7	*How To Save A Life*, The Fray	*Say It Right*, Nelly Furtado
8	*We Fly High*, Jim Jones	*Walk Away*, Paula DeAnda Feat. The DEY
9	*Welcome To The Black Parade*, My Chemical Romance	*Make It Rain*, Fat Joe Feat. Lil Wayne
10	*It Ends Tonight*, The All-American Rejects	*I Wanna Love You*, Akon Feat. Snoop Dogg

Table 3. Billboard's Top Ten Published on January 11th 2007 vs. The Alternative Chart

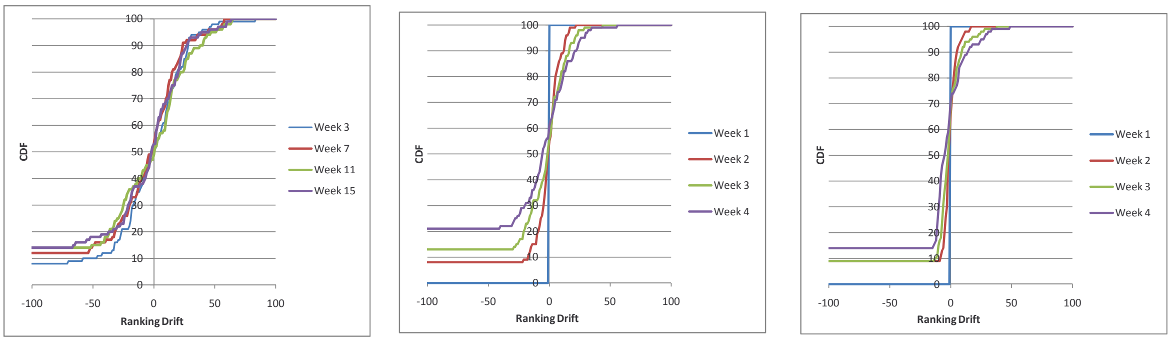
In Table 3 we show the top ten Billboard singles from the chart released on January 11th 2007 (week 2), and our alternative singles chart based on P2P activity on of the previous week. The two charts share four common songs, yet they are quite distinct. For the full 100 songs charts, the median distance of songs on the Billboard from their ranking on the alternative chart is 18.

In order to better understand the difference in song ranking, we define the *ranking drift* of a song as the difference between its rank on the Billboard chart to its rank on the corresponding alternative chart. We then plot the cumulative distribution function (CDF) of this difference for all 100 titles on the Billboard. Fig. 3(a) depicts the CDF of four weekly charts on different weeks in 2007. The week numbers are according to the Billboard charts. The correspondence of the two charts can be evaluated from the shape of graphs. A perfect match between the two charts, would appear as a perfect step function. Fig. 3(a) reveals a moderate correspondence of the Billboard charts with the alternative charts. For instance, the percentage of songs whose rank drift is in the range -25 to 25 is 60% on average.

CDF charts can be further used to compare the dynamics within each chart over time. We thus measured the drift of the songs from their ranking on previous weeks (on the same chart). Fig. 3(b) depicts the ranking drift of songs on the Billboard from the first week of 2007, over a period of 3 weeks. As expected the ranking drift increases for longer time intervals. Fig. 3(c) depicts the ranking drift of songs on the alternative chart during the same time period. Again the drift increases with time. The drift on the alternative chart, however, is smaller than that of the Billboard, indicating less change in songs ranking from week to week.

4. DISCUSSION

In past decades, air-plays and record sales were the primary means of distribution of popular music. The Billboard Hot 100 was therefore a reasonable proxy to popularity. Today, however, new technologies in particular the Internet, have created new means for distribution of music.



(a) Songs ranking drift between the Billboard Hot 100 and the alternative chart (b) Songs time drift on the Billboard Hot 100 (c) Songs time drift on the alternative charts

Figure 3. Cumulative Distribution of Ranking Drift (CDF)

The growing popularity of file sharing make record sales and radio plays an increasingly poor predictor of peoples' taste. The record industry attempts to stop the swapping of pop music through the Internet by taking some P2P vendors to court, but the steady spread of file sharing systems and their technological improvements make them impossible to shut down.

In Section 3 we saw that currently the Billboard's sales based ranking system, is still quite in tune with what people download, but as file sharing becomes ever more prevalent, a need for a new ranking system arises. This observation was first introduced by Grace *et al.* [6], where it was suggested to use opinion mining (OM) on public boards to measure music popularity. In [6], comments on artists' pages on MySpace were used to build an alternative popularity chart of musical artists. Their top ten alternative list was substantially different than that of the Billboard. It was preferred, however, over the Billboard's list by 2-to-1 ratio by their 74 human test subjects.

We argue that popularity ranking based on P2P activity has many advantages over ranking based on opinion mining. First, it eliminates the complex task of classifying opinion polarities based on identifying opinion semantics. When P2P queries are considered, each query is always a positive indication of a user showing interest in the song or the artist. Second, the laborious task of identifying spam content in opinion mining, becomes trivial in a data set of query strings. On top of that, opinion mining in a website such as MySpace is biased towards the typical user of such a website, and biased again towards active users who care to comment on artists pages. Our method, doesn't require an active action on the side of the user. We rather measure queries generated as part of the file sharing process. Nonetheless, these queries disclose the interests of the user. Finally, we argue that opinion mining is more vulnerable to manipulations by stakeholders such as public relation companies acting on behalf of the artist or the record company. Planting comments on MySpace by interested entities is rather easy, while the technological barrier of generating many search queries in a file sharing network is much higher. In fact, networks such as Gnutella, already employ techniques to identify and eliminate non-human automatic search queries (as described in Section 1).

However, ranking songs based on P2P queries still has some open questions. There are, of course, the ethical issues with music piracy which are yet to be addressed. Regarding integrity, the ranking might be biased towards the preferences of file swappers which may differ in taste from the general public. It is also possible that a single P2P network, however large, has a user community which is biased against or for some genres, bringing the need to base the chart on all the top P2P networks and not just the largest one as was done in this study. Some of the open questions on the algorithmic side include the need to develop an artist ranking algorithm based on singles downloads, and to resolve the ranking of songs with confusing titles (e.g., *Love* or *Hot*).

It is not unlikely, that in the foreseeable future, music distribution based on file sharing will become the norm, and music sales will be reduced to a niche market. We expect that as the practice of file sharing becomes even more widespread, this line of research will become increasingly relevant.

5. REFERENCES

[1] Ram D. Gopal and G. Lawrence Sanders. Do artists benefit from online music sharing? *Journal of Business*, 79(3):1503–1534, May 2006.

[2] Martin Peitz and Patrick Waelbroeck. Why the music industry may gain from free downloading – the role of sampling. *International Journal of Industrial Organization*, 24(5):907–913, September 2006.

[3] Sudip Bhattacharjee, Ram D. Gopal, Kaveepan Lertwachara, and James R. Marsden. Using p2p sharing activity to improve business decision making: proof of concept for estimating product lifecycle. *Electronic Commerce Research and Applications*, 4(1):14–20, 2005.

[4] Sudip Bhattacharjee, Ram Gopal, Kaveepan Lertwachara, and James R. Marsden. Whatever happened to payola? an empirical analysis of online

music sharing. *Decis. Support Syst.*, 42(1):104–120, 2006.

[5] Noam Koenigstein, Yuval Shavitt, and Tomer Tankel. Spotting out emerging artists using geo-aware analysis of p2p query strings. In *The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[6] J. Grace, D. Gruhl, K. Haas, M. Nagarajan, C. Robson, and N. Sahoo. Artist ranking through analysis of online community comments. *The 17th International World Wide Web Conference*, 2008.

[7] Adam Shaked Gish, Yuval Shavitt, and Tomer Tankel. Geographical statistics and characteristics of p2p query strings. In *The 6th International Workshop on Peer-to-Peer Systems*.

[8] Daniel Stutzbach, Reza Rejaie, and Subhabrata Sen. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *IEEE/ACM Transactions on Networking*, 16(2), 2008.

[9] Paul Resnikoff. Digital media desktop report, fourth quarter, 2007, April 2008. Digital Music Research Group.

[10] Amir H. Rasti, Daniel Stutzbach, and Reza Rejaie. On the long-term evolution of the two-tier gnutella overlay. In *IEEE Global Internet Symposium*, Barcelona, Spain, April 2006.

[11] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5, 2000.

[12] M. Ripeanu. In *First International Conference on Peer-to-Peer Computing*.

[13] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6:2002, 2002.

[14] A. Klemm, C. Lindemann, M. Vernon, and O. P. Waldhorst. Characterizing the query behavior in peer-to-peer file sharing systems. In *Internet Measurement Conference*.

[15] K. Sripanidkulchai. The popularity of gnutella queries and its implications on scalability, February 2001. Featured on O'Reilly's www.openp2p.com website.

[16] Mihajlo A. Jovanovic. Modeling large-scale peer-to-peer networks and a case study of gnutella. Master's thesis, University of Cincinatti, Cincinatti, OH, USA, 2001.

[17] Wikipedia the free encyclopedia. Billboard hot 100. http://en.wikipedia.org/wiki/Billboard_Hot_100 Last accessed May 2009.

[18] Eric T. Bradlow and Peter S. Fader. A bayesian lifetime model for the "hot 100" billboard songs. *Journal of the American Statistical Association*, 96:368–381, 2001.

METER CLASS PROFILES FOR MUSIC SIMILARITY AND RETRIEVAL

Matthias Robine and Pierre Hanna
LaBRI - University of Bordeaux
351, cours de la Libération
33405 TALENCE Cedex - FRANCE
firstname.name@labri.fr

Mathieu Lagrange
Telecom ParisTech
46, rue Barrault
75634 PARIS Cedex 13 - FRANCE
lagrange@telecom-paristech.fr

ABSTRACT

Rhythm is one of the main properties of Western tonal music. Existing content-based retrieval systems generally deal with melody or style. A few existing ones based on meter or rhythm characteristics have been recently proposed but they require a precise analysis, or they rely on a low-level descriptor. In this paper, we propose a mid-level descriptor: the Meter Class Profile (MCP). The MCP is centered on the tempo and represents the strength of beat multiples, including the measure rate, and the beat subdivisions. The MCP coefficients are estimated by means of the autocorrelation and the Fourier transform of the onset detection curve. Experiments on synthetic and real databases are presented, and the results demonstrate the efficacy of the MCP descriptor in clustering and retrieval of songs according to their metric properties.

1. INTRODUCTION

The amount of digital music is rapidly increasing, and is mostly comprised of Western pop music. New interfaces for browsing, classifying or searching have to be proposed. Content-based retrieval systems generally consider musical properties related to melody or style. But taking into account other characteristics such as rhythm may lead to the development of useful tools for helping users to browse into large databases.

Research regarding rhythmic or metric properties typically involves tempo or meter analysis. Several systems have been developed for improving tempo induction [1,2], meter analysis [3,4], or estimation of the time signature of audio songs [5]. Other works focus on the automatic classification of songs according to their rhythmic properties [6]. In contrast, only a few methods have been presented for retrieving songs by rhythmic similarity. A few existing ones consider a low-level descriptor such as beat spectrum [7], acoustic features [8], or spectral descriptors [9]. Others precisely analyze tempo, meter and/or time signature [5] and are consequently limited by the error analysis.

In this paper, we present a new mid-level descriptor for retrieving music according to the metric properties. The estimation of this descriptor requires neither a complete analysis of the time signature nor the meter of a song, but only the prior knowledge of its tempo. In comparison to a low-level feature, the essential metric information is reduced to a few values which characterize the metric properties of any song. In Section 2, we discuss the notion of metrical structure. The new mid-level descriptor and the associated analysis method are detailed in Section 3. Then, clustering and retrieval experiments are presented in Section 4. Finally, discussion of these results and perspectives are proposed in Section 5.

2. MUSICAL METER

2.1 Metrical Structure

In Western tonal music, the periodic alternation of strong and weak beats leads to a metrical hierarchy known as metrical structure. The symbolic representation of this structure is present in score notation using markings such as a time signature, bar lines, and dynamic accents.

The Generative Theory of Tonal Music (GTTM) [10] proposes a model of the metrical structure, where the meter of a musical piece may be represented by multiple levels of beats. The periodicity of beats is reinforced from level to level, and it is the interaction of the different levels that produces the sensation of meter. This representation is also used by Temperley [11] in his proposal of a preference rule system for meter.

According to [10], while there are five or six metrical levels in a piece, one is particularly central: the *tactus* level. The tactus identifies a perceptually prominent level, with the levels immediately smaller and immediately larger. It refers to the perceived tempo, the internal clock [12]. It corresponds highly with the notated unit time of a musical piece, but it can differs. Lee [13] indicates thus that listeners may revise meter to always get a tactus between 300ms and 600ms.

In the following, we assume that the tactus level corresponds to the tempo and the basic unit time of the music, when it is known or notated. We choose also to use without distinction the terms *tactus* and *beat*. As the metrical structure is hierarchical, levels lower than that of the tactus can be called subtactus levels and represent divisions of the beat. The smallest division is generally called *tatum* or *tick*. Alternatively, higher levels are termed supertactus levels, and contain multiples of the beat duration, including the measure level.

2.2 Time Signature

We can restrict the notion of meter to two levels, the faster of which provides the element, and the slower of which group them. According to Gouyon [14], this is close to the usual description of meter that can be found in a score, given by the time signature and the bar lines.

The time signature consists of 2 integers arranged vertically, e.g. $\frac{4}{4}$ or $\frac{6}{8}$. The upper number of the signature indicates the number of units in a bar, with the value of unit given by the lower number of the signature (*e.g.* 4 for a quarter note). If the upper number is divisible by 3 and the lower by 2, the time is compound and the number of beats per measure is given by the upper number divided by 3. In this case, the unit of time, *i.e.* the beat, is divided triply at the smaller level. Otherwise, in simple time, the upper number indicates the number of beats per measure and the beat is divided duply. Table 1 presents the time signatures mainly used in Western tonal music.

3. METER CLASS PROFILE

We introduce in this section a new descriptor to represent the metrical structure of the music called Meter Class Profile (MCP). We describe the method used for estimating the MCP, and provide examples for illustration of its use.

3.1 Properties

MCP is a real-valued vector providing information of the strength of the different metrical levels within the music. It is centered on the tactus level: the beat multiples, including the measure level, are represented on the left, and the beat subdivisions, including the tatum level, on the right. The choice was made to represent the relative strength of accents at multiple rates of the tempo: $2, 3, 4, 5, 7, 9$, and 11, and also at subdivisions of the tempo: $1/2, 1/3, 1/4, 1/6, 1/8, 1/12$. MCP is thus a vector of thirteen dimensions.

A MCP corresponding to a $4/4$ time signature would contain high amplitudes for the bin corresponding to 4 times the tempo, *i.e.* beat multiple 4 representing the measure periodicity, for beat multiple 2 (in $4/4$, there are accents every two beats), for beat subdivision $1/2$ (because $4/4$ is in simple time), and perhaps beat subdivisions $1/4$ and $1/8$ (if 16^{th} or 32^{th} notes occur). A few examples are presented in Figure 2.

As the MCP is independent from tempo, its representation does not change with tempo variations. It may be considered as a mid-level descriptor, since the metric information is summarized to a 13-dimension vector without identifying a particular time signature. Additionally, the amplitude ratios from the different metrical levels provide a meaningful way to handle the meter of a musical piece. The MCP could, for example, also indicate a degree of swing, by considering the balance between duple and triple beat subdivisions.

3.2 Estimation

The method proposed here for computing MCP relies on existing analysis methods recently described for estimating tempo [15]. It has been implemented using the MIR toolbox [16]. The main steps are illustrated on Figure 1, with the example of the country song *Wanted* (A. Jackson), annotated with a $3/4$ time signature.

In this paper, we consider only one global MCP per audio musical signal. We thus choose to analyze a large frame of music (length 60 seconds). The meter is assumed to be stationary during this frame. An onset-energy function is first extracted from the audio signal by taking into account spectral energy flux [17]. Then, dominant periodicities (or frequencies) are estimated.

Two types of observations, respectively termed Onset Discrete Fourier Transform (ODFT) or Onset Autocorrelation Function (OAF), are considered. Their complementary properties are discussed for tempo estimation in [2]. In the experiment presented in this paper, the OAF and ODFT are both computed and normalized and the tempo frequency is known.

The analysis method of MCP locates both periodicities corresponding to beat multiples (related to measure) and beat subdivisions. Estimation of multiples and subdivisions is carried out using the complementary properties of the two observations. On one hand, the ODFT of a periodic signal is a set of harmonically related frequencies, and it is difficult to determine predominant frequencies above the tempo frequency. Therefore, we only estimate frequencies lower than the tempo frequency. These frequencies correspond to beat multiples (first part of MCP), in particular that of the measure.

On the other hand, the OAF of a periodic signal is a set of periodically related lags. It is thus difficult to measure predominant periodicities higher than the tempo period. The OAF is only considered for estimating periods lower than the tempo periods. These periods are related to beat subdivisions (second part of the MCP).

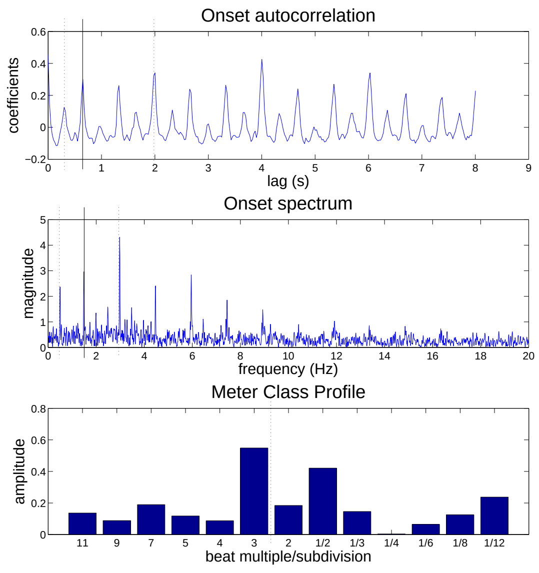


Figure 1. Different stages of the analysis method of Meter Class Profile for the song *Wanted* of A. Jackson (time signature $3/4$): from top to bottom, the autocorrelation function, the spectrum of the onset function and the MCP estimated, showing peaks at beat multiple 3 and beat subdivision $1/2$.

With prior knowledge of the tempo, the first part of the MCP is estimated from the ODFT and the second part is estimated from the OAF. Consequently, period bands corresponding to harmonics of the frequency tempo are analyzed when considering the OAF. Only the six harmonics (2, 3, 4, 6, 8, and 12) are taken into account. The amount of energy of OAF within a thin frequency band around the related periodicities directly determines the amplitude re-

	Duple	Triple	Quadruple	5-uple	7-uple	9-uple	11-uple
Simple	$\frac{2}{1}\ \frac{2}{2}\ \frac{2}{4}\ \frac{2}{8}$	$\frac{3}{1}\ \frac{3}{2}\ \frac{3}{4}\ \frac{3}{8}$	$\frac{4}{1}\ \frac{4}{2}\ \frac{4}{4}\ \frac{4}{8}$	$\frac{5}{1}\ \frac{5}{2}\ \frac{5}{4}\ \frac{5}{8}$	$\frac{7}{1}\ \frac{7}{2}\ \frac{7}{4}\ \frac{7}{8}$	$\frac{9}{1}\ \frac{9}{2}\ \frac{9}{4}\ \frac{9}{8}$	$\frac{11}{1}\ \frac{11}{2}\ \frac{11}{4}\ \frac{11}{8}$
Compound	$\frac{6}{2}\ \frac{6}{4}\ \frac{6}{8}\ \frac{6}{16}$	$\frac{9}{2}\ \frac{9}{4}\ \frac{9}{8}\ \frac{9}{16}$	$\frac{12}{2}\ \frac{12}{4}\ \frac{12}{8}\ \frac{12}{16}$	$\frac{15}{2}\ \frac{15}{4}\ \frac{15}{8}\ \frac{15}{16}$	$\frac{21}{2}\ \frac{21}{4}\ \frac{21}{8}\ \frac{21}{16}$	$\frac{27}{2}\ \frac{27}{4}\ \frac{27}{8}\ \frac{27}{16}$	$\frac{33}{2}\ \frac{33}{4}\ \frac{33}{8}\ \frac{33}{16}$

Table 1. Time signatures. Most common signatures are duple, triple and quadruple time in Western tonal music. Notations 9/2, 9/4 and 9/8 may be used either for a simple measure of 9 pulses, or for a compound measure of 3 pulses.

lated to the beat subdivisions of the MCP. In our implementation, the width of the frequency bands for cumulating the energy in the ODFT and OAF has been set to 5%. In Figure 1, the tempo has been annotated to 1.5Hz and is showed on the OAF with a solid line. When considering periods lower than $\frac{1}{1.5} = 0.66$s, energy is located around period of 0.33s, corresponding to the beat subdivision $1/2$. The contribution to the beat subdivision of the corresponding MCP is thus significant and clearly indicates a simple meter.

The ODFT is considered in a similar way. Only the seven sub-harmonics (2, 3, 4, 5, 7, 9, and 11) of the tempo frequency are considered. Energy within a thin band around these sub-harmonics determines the amplitude related to the beat multiples. In Figure 1, only the energy around frequencies $\frac{1.5}{2}$, $\frac{1.5}{3}$, $\ldots$, $\frac{1.5}{11}$, contributes to the first part of the MCP. In this example, the sub-harmonic $\frac{1.5}{3}$ is significantly predominant and results in a substantial amplitude in the MCP. This high amplitude thus indicates that the song is characterized by 3 beats per measure.

Other examples of MCP computation for real audio songs are shown in Figure 2. In each example, the MCP looks very different, according to their metric properties. In particular, the highest value in the first part of the MCP generally indicates the number of beats per measure, whereas the highest value of the second part is related to the beat subdivision.

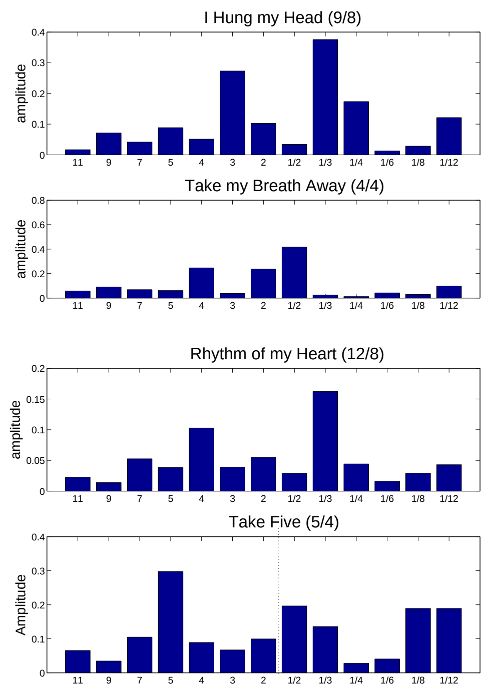


Figure 2. Examples of MCP computed from real audio songs with different time signatures, respectively $9/8$, $4/4$, $12/8$ and $5/4$.

3.3 Distance between MCP

The MCP is proposed for music retrieval purposes. Therefore, a method for computing a matching score between two MCP has to be defined. Several distances are possible, however this is a difficult selection due to the difference in the analysis processes of the two parts of the MCP. We thus propose to consider a global score s as the combination of the two scores s_1 and s_2 obtained with the two parts of the MCP:

$$s = \alpha s_1 + (1 - \alpha)s_2 \tag{1}$$

where α is a fixed weighting value in the interval $[0; 1]$, s_1 the comparison score related to the *meter multiple* part of the MCP, and s_2 the comparison score related to the *meter subdivision*. These two scores s_1 and s_2 are calculated according to correlation:

$$\begin{aligned} s_i(\text{MCP}_1, \text{MCP}_2) &= \frac{c(\text{MCP}_1, \text{MCP}_2)}{\sqrt{c(\text{MCP}_1, \text{MCP}_1)}\sqrt{c(\text{MCP}_2, \text{MCP}_2)}} \\ c(\text{MCP}_1, \text{MCP}_2) &= \sum_{i=1}^{N} \text{MCP}_1(i)\,\text{MCP}_2(i) \end{aligned} \tag{2}$$

where MCP$_1$ and MCP$_2$ are MCP vectors of size N.

4. EXPERIMENTS

In this section, experiments are presented that demonstrate the ability of MCP to discriminate songs that have different metric characteristics. The first experiments deal with clustering abilities, and other second concerns song retrieval.

4.1 Databases

Two databases are considered in this paper. The first one is composed of short artificial audio musical pieces (60 seconds long at 16 kHz) that have been synthesized according to different metric properties. The tempo was set to 1 Hz and classes have been constituted considering the time signature. Classes with 2, 3, 4, 5, 7, 9 and 11 beats per measure have been built, in simple time and in compound time. For each class, 2 different distributions of the strong beats in the measure have been chosen, to synthesize 200 pieces. A process has been achieved to randomly add 16^{th} notes in the pieces. The database contains 2800 different files.

The second database is a collection of real pop audio songs indexed using the time signature. The *noise* collection contains 476 simple-meter songs with 2 or 4 beats per measure (sampling rate 44.1 kHz).We constitute another collection of 54 songs with different metric properties. Some of them are in compound time, while others are characterized by a different number of beats per measure. Therefore, 7 different classes are assumed according to 7 different time signatures. The composition of each class is presented in Table 2. Different classes are also deduced from time signatures: one class is composed of 13 songs with 3 beats per measure, another class comprises 24 compound time songs. For all these 54 songs, tempo has been manually annotated.

It is important to notice that all ground truth annotation of the songs from the *noise* database have not been precisely verified; ambiguous meter, large tempo variations, and short-duration time signature changes may result in evaluation errors that may underestimate the quality of the clustering and retrieval systems presented here.

Simple				Compound		
3/4	**5/4**	**7/4**	**11/4**	**6/8**	**9/8**	**12/8**
11	10	7	2	6	2	16

Table 2. Number of songs within each meter class considered for the experiments.

4.2 Clustering

We present here the clustering abilities of the proposed meter feature on the two different databases.

4.2.1 *Evaluation Metrics*

The task can be here reworded as follows: "Do the elements classified together actually belong to the same time signature?". The similarity of the elements of the given database is first computed. As the databases considered in those experiments are of relatively small sizes, we consider an unsupervised clustering scheme (the k-means) to perform the clustering task, *i.e.* each element e_k is given a clustering tag t_k. The correct number of classes is given to the clustering algorithm.

The clustering matrix M is next computed, where each entry is defined as:

$$M(C_x, C_y) = \frac{\#\{(e_k, e_l) | t_k = t_l \wedge e_k \in C_x \wedge e_l \in C_y\}}{\#\{(e_k, e_l) | e_k \in C_x \wedge e_l \in C_y\}} \tag{3}$$

where C_x and C_y are the classes given as ground truth, and (e_k, e_l) is a couple of elements. In order to attenuate the impact of the random initialization of the k-means algorithm, the classification is done 10 times and the mean result over the 10 iterations is considered. The better the classification, the higher the ratio between the diagonal values of M and the remaining of the matrix.

Evaluation over the synthetic database allows us to validate the proposed approach in a controlled environment. Figure 3 depicts the clustering matrix M for the synthetic database. The results are very satisfying in general as most of the values are concentrated on the diagonal, meaning that most of the cluster generated by the k-means algorithm from the features correspond to the actual metrical structureclasses. Typical errors are due to a confusion between simple and compound time, e.g. between 3/4 and 9/8 time signatures, or sometimes between classes for which the numbers of beats per measure have common factors (e.g., 4/4 and 2/4, or 9/4 and 3/4).

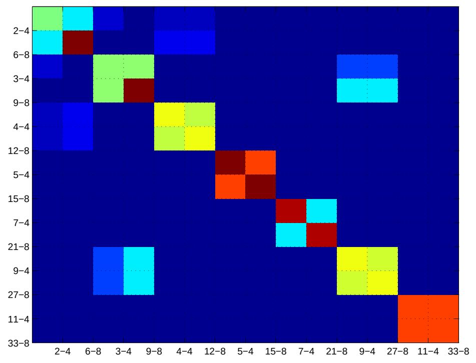


Figure 3. *Clustering matrix over the synthetic database.*

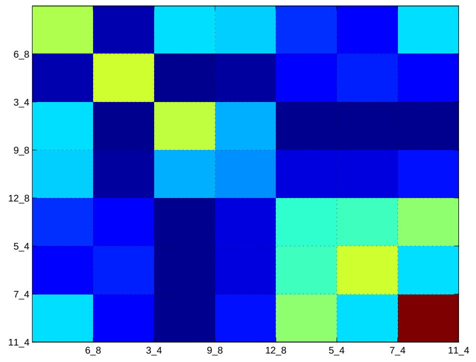


Figure 4. *Clustering matrix over the real database.*

Real data is now considered to evaluate the robustness of the MCP. Evaluations based on clustering allows us to finely analyze the properties of the MCP with respect to the different time signature classes, as shown in Figure 4. We can note that a clear distinction is made between simple and compound time.

4.3 Retrieval

In this section, we propose the evaluation of a music retrieval system based on the MCP. The 54 songs from the real song collection are successively considered as query. The retrieval system computes a similarity score between the query and all the songs of the database. The database is comprised of 530 different songs, including all the queries and all the *noise* collection. These songs are then ranked from most to least similar. For each query, we expect to retrieve all the songs belonging to the class of the query at the top rank. In the following evaluation, results are presented with Precision at Top 1, N and $2N$, in which N denotes the size of the class of the query.

4.3.1 *Synthetic Query*

The first experiments concern retrieval based on a synthetic query. This query is a flat input, *i.e.* a MCP defined by a binary string. For example, if the songs searched have a $3/4$ time signature, the synthetic query is the MCP

$$[0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0]$$

where the only non-null values correspond to the beat multiple 3 and the beat subdivisions $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ are related to simple time subdivisions. Other experiments only concern the left or right half of the MCP, since retrieval may focus on beat multiples or beat subdivisions. For example, retrieving *compound time* songs is tested by considering only the second part of the MCP with a synthetic query such as:

$$[0,0,0,0,0,0,0,0,1,0,1,0,1]$$

which exhibits beat subdivisions $\frac{1}{3}$, $\frac{1}{6}$, and $\frac{1}{12}$.

Table 3 shows the results of the retrieval experiments from those synthetic queries. At the exception of time signatures $11/4$ and $9/8$, one song of the correct class is always correctly retrieved at the first rank. These two exceptions may be explained by the small size the two classes concerned (each comprised of only two songs). Concerning the $11/4$ class, related songs are retrieved at ranks 2 and 11. Concerning the $9/8$ class, related songs are retrieved at ranks 3 and 5. Even if the precision at top 1 is null, the retrieval results are thus quite good.

Average precision at top $2N$ indicates that more than half the songs are generally retrieved within the first $2N$ ranks. Considering one part of the MCP for retrieving songs seems to be effective. For example, a synthetic query allows the retrieval of 75% of the songs of the compound time class at the first N ranks. Almost all the compound time songs are ranked within the first $2N$ best matches. The results of these experiments confirm the quality of the MCP as a metric descriptor for retrieval, since the MCP computed from a real audio file is similar to its time signature properties.

Class	**Size**	**Top 1**	**Top N**	**Top 2N**
6/8	6	1	0.333	0.500
3/4	11	1	0.364	0.636
9/8	2	0	0.000	0.500
12/8	16	1	0.312	0.375
5/4	10	1	0.800	0.900
7/4	7	1	0.286	0.429
11/4	2	0	0.500	0.500
Total by Class	7	0.714	0.371	0.549
3 beats/mes	13	1	0.846	1.000
5 beats/mes	10	1	0.800	1.000
7 beats/mes	7	1	0.571	0.571
11 beats/mes	2	1	0.500	0.500
Compound	24	1	0.750	0.958

Table 3. Results of the retrieval system based on MCP, considering a synthetic query.

4.3.2 Audio Query

We present a second experiment to test the ability of retrieving real songs using a real song as a query. The applications related to these experiments are Query-by-Example systems, which allows users to perform a database search for songs that are similar to a given query song. The similarity here relies on the metric properties, but does not have to be explicitly determined by the user.

Since the query is always retrieved at the first rank, we propose to remove the query from the database. Precision at top N is the number of correct songs retrieved in the first $N-1$ ranks divided by $N-1$, N being the size of the class considered. Precision at top $2N$ is the number of correct songs retrieved in the first $2N-1$ ranks divided by $2N-1$. By using all the songs of each class as a query, N precisions are computed and averaged for each class. Then, the total average is computed by query, or by class. It is respectively denoted *Total by Query*, and *Total by Class*. Such evaluations are respectively named *First Tier* and *Second Tier* in [18].

Table 4 shows the results of the retrieval experiments. The value α determines the weighting between the two parts of the MCP, and has been set to 0.6. The average results by query indicate that 44% of the queries allow the retrieval of one song of the same class at the first rank, 53% of the class is retrieved at the first $2N$ ranks. The accuracy is lower than the results obtained with synthetic queries. This can be explained by the presence of songs in the *noise* database that may be similar to the query. For example, a query with time signature $3/4$ often leads to the retrieval of songs with time signature $9/8$, since the number of beats per measure are the same for each of these two classes and since the beat subdivisions may be varying during the analyzed song (for example in the case of swing). At the opposite, the class $5/4$ leads to the best results: 80% of the correct songs are retrieved.

Moreover, difficulties with annotations of time signatures may lead to errors in evaluation. For example, it is sometimes difficult to discriminate $6/8$ songs from $12/8$ songs. This difficulty is illustrated by the poor results for class $6/8$, whereas the compound time class leads to good results.

Classes	**Size**	**Top 1**	**Top N**	**Top 2N**
6/8	6	0.000	0.033	0.242
3/4	11	0.727	0.500	0.649
9/8	2	0.000	0.000	0.333
12/8	16	0.562	0.579	0.714
5/4	10	0.700	0.567	0.695
7/4	7	0.000	0.095	0.132
11/4	2	0.000	0.000	0.000
Total by Class	7	0.284	0.253	0.395
Total by Query	54	0.444	0.394	0.529
3 beats/mes	13	1.000	0.686	0.825
Compound	24	0.875	0.784	0.863

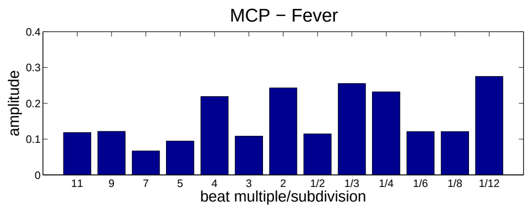
Table 4. Results of the retrieval system based on MCP, considering an audio song as query.

5. CONCLUSION AND PERSPECTIVES

In this paper, we have proposed a new mid-level descriptor, related to the beat multiples and subdivisions. Experiments with synthetic and real songs show that considering the MCP allows the retrieval of songs belonging to the same metric class.

When focusing on the time signature, we reduce the search information from the descriptor. Other considerations are also of interest, such as the amplitudes of all the beat subdivisions, which may denote a certain rhythmic complexity of the music. The MCP of one minute of *Fever* by *Ray Charles* is shown in Figure 5. If the time signature of this piece is generally notated 4/4, beat subdivisions at

1/3 are more prevalent than at 1/2. As studied in [19], this is a consequence of the swing. We see here that MCP contains information more complex than the time signature only, and it could thus be used for very specific retrieval purpose.



The estimation of the MCP assumes the prior knowledge of the correct tempo. Its robustness against tempo estimation has to be improved in the future. If the tempo may be automatically estimated, errors are unavoidable and will significantly limit the accuracy of the MCP, and thus the accuracy of the retrieval system.

Furthermore, since metric properties may change during a song, a song may be represented by a sequence of MCP (computed during short frames), in the same way that tonal properties of a song can be represented by a sequence of chromas [20]. Such representation may allow the discrimination of songs with the same metric properties, but with different evolutions with respect to time.

6. ACKNOWLEDGEMENT

The authors would like to thank Jason Hockman for his useful comments during the writing of the paper. This work has been partly funded by the Quaero project within the task 6.4: “Music Search by Similarity” and the French GIP ANR under contracts ANR-06-JCJC-0027-01, DESAM and ANR-JC07-188930, SIMBALS.

7. REFERENCES

[1] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Speech and Audio Processing*, 14(5):1832–1844, 2006.

[2] G. Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, 2007(1), 2007. 14 pages.

[3] J. Brown. Determination of the meter of musical scores by autocorrelation. *Journal of the Acoustical Society of America*, 94(4):1953–1957, 1993.

[4] A. Klapuri, A. Eronen, and J. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Speech and Audio Processing*, 14(1):342–355, 2006.

[5] C. Uhle and J. Herre. Estimation of tempo, micro time and time signature from percussive music. In *Proc. of the International Conference on Digital Audio Effects (DAFx*, London, UK, 2003.

[6] S. Dixon, E. Pampalk, and G. Widmer. Classification of dance music by periodicity patterns. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 159–165, Baltimore, 2003.

[7] J. Foote, M.D. Cooper, and U. Nam. Audio retrieval by rhythmic similarity. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 265–266, Paris, France, 2002.

[8] J. Paulus and A. Klapuri. Measuring the similarity of rhythmic patterns. In *Proc. of the International Conference on Musical Information Retrieval (ISMIR)*, pages 150–156, Paris, France, 2002.

[9] A. Holzapfel and Y. Stylianou. A scale transform based method for rhythmic similarity of music. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009.

[10] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, Massachussetts, 1985.

[11] D. Temperley. *The Cognition of Basic Musical Structures*. The MIT Press, 2001.

[12] D.J. Povel and P. Essens. Perception of temporal patterns. *Music Perception*, 2:411–440, 1985.

[13] C. S. Lee. *Representing Musical Structure*, chapter The perception of metrical structure: Experimental evidence and a model, pages 59–127. P. Howell, R. West, and I. Cross, London, UK, 1991.

[14] F. Gouyon. *Towards Automatic Description of Musical Audio Signals - Representations, Computional Models and Applications*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2003.

[15] G. Peeters. Rhythm Classification Using Spectral Rhythm Patterns. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pages 644–647, London, UK, 2005.

[16] O. Lartillot and P. Toiviainen. A Matlab Toolbox for Musical Feature Extraction From Audio. In *Proc. of the International Conference on Digital Audio Effects (DAFx)*, Bordeaux, France, 2007.

[17] M. Alonso, G. Richard, and B. David. Tempo And Beat Estimation Of Musical Signals. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004.

[18] B. De Haas, R. Veltkamp, and F. Wiering. Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 51–56, September 14-18 2008.

[19] K.A. Lindsay and P.R. Nordquist. A technical look at swing rhythm in music. *The Journal of the Acoustical Society of America*, 120(5):3005–3005, 2006.

[20] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, July 2006.

SHEET MUSIC-AUDIO IDENTIFICATION

Christian Fremerey, Michael Clausen, Sebastian Ewert
Bonn University, Computer Science III
Bonn, Germany
{fremerey,clausen,ewerts}@cs.uni-bonn.de

Meinard Müller
Saarland University and MPI Informatik
Saarbrücken, Germany
meinard@mpi-inf.mpg.de

ABSTRACT

In this paper, we introduce and discuss the task of sheet music-audio identification. Given a query consisting of a sequence of bars from a sheet music representation, the task is to find corresponding sections within an audio interpretation of the same piece. Two approaches are proposed: a semi-automatic approach using synchronization and a fully automatic approach using matching techniques. A workflow is described that allows for evaluating the matching approach using the results of the more reliable synchronization approach. This workflow makes it possible to handle even complex queries from orchestral scores. Furthermore, we present an evaluation procedure, where we investigate several matching parameters and tempo estimation strategies. Our experiments have been conducted on a dataset comprising pieces of various instrumentations and complexity.

1 INTRODUCTION

When listening to an audio recording of a piece of music, an obvious problem is to decide, which bar of a corresponding sheet music representation is currently played. For technical reasons, we tackle this problem from the viewpoint of *sheet music-audio identification*: Given a sequence of bars from the sheet music as a query, the task is to find all temporal sections in the audio recording, where this bar sequence from the query is played.

One application of this task is to find out, whether there are differences between the default bar sequence following the instructions in the sheet music and what is actually played in the audio interpretation. In case there are differences, sheet music-audio identification may also be used to automatically determine the bar sequence that is played in the interpretation, and to identify special parts like cadenzas that have no counterpart in the sheet music.

If the bar sequence played in the audio interpretation is known in advance, sheet music-audio identification can be solved by first performing sheet music-audio synchronization and then using the synchronization results to identify the temporal sections in the audio that correspond to a given query sequence of bars. In case the correct bar sequence is not known, a more direct approach must be taken. Here, sheet music-audio matching as performed in [1] seems to be a reasonable strategy.

In the literature, alignment, identification and retrieval has been a popular field of research for the single-domain cases of either audio or symbolic data, see [2] and the references therein. For the cross-domain case, a lot of effort has been put into the task of off-line and on-line alignment of score data and audio data [3–6]. Here, the assumption is made that the bar sequence of the score is already known. The idea of using cross-domain synchronization results as ground truth or training data for more complicated music information retrieval tasks has already been formulated for the application of automatic transcription of pop music [7].

First important steps towards cross-domain matching and identification of polyphonic musical works have been conducted by the groups of Pickens and Orio [4, 8]. Using either audio transcription techniques [8] or a statistical model for the production of audio data from polyphonic score data [4] a complete audio track (song or movement) is used as a query to find the corresponding work in the score domain. First experiments for approaching the task of cross-domain work identification by querying arbitrary segments of score data have been conducted by Syoto et al. [9] as well as in our previous work [1]. None of the above approaches explicitly handles differences in bar sequence structure or repeats between the score and audio data, even though this is a common and practically relevant issue in real-world digital music libraries.

The paper is structured as follows. Section 2 specifies the task of sheet-music audio identification in more detail and discusses some difficulties and pitfalls. Our two approaches to sheet music-audio identification are presented in Section 3, one using synchronization and the other using matching. Section 4 explains how MIDI events for comparison with the audio data are created from the sheet music data. The synchronization and matching procedures are outlined in Sections 5 and 6. Section 7 describes an evaluation procedure for the matching approach using the more reliable results of the synchronization approach as a ground truth. Experimental results using our test dataset are discussed in Section 8 before the paper concludes with an outlook on future work in Section 9.

We gratefully acknowledge support from the German Research Foundation DFG. The work presented in this paper was supported by the PROBADO project (http://www.probado.de/, grant INST 11925/1-1) and the ARMADA project (grant CL 64/6-1).

2 SHEET MUSIC-AUDIO IDENTIFICATION

In the following, we assume that we are given one scanned sheet music representation and one audio interpretation of the same piece of music. We assign a unique label (p, b) to each bar written in the sheet music, where p is the page number and b is the bar number on the page. Furthermore, B denotes the set of all bar labels of the piece. Sheet music may contain jump directives like repeat signs, alternative endings, dacapos or segnos. Following these directives as they are written in the sheet music, one obtains a sequence $\delta = (\delta_1, \ldots, \delta_n)$, $\delta_i \in B$, indicating the *default sequence* of bars that is to be played when performing the piece. In practice, however, the given audio recording does not always follow this sequence δ. Performers might, for example, choose to ignore or add repeats, or even introduce shortcuts. This leads to a possibly different sequence $\pi = (\pi_1, \ldots, \pi_d)$, $\pi_i \in B \cup \{\uparrow\}$, which we call *performance sequence*. Here, we use the label $\uparrow$ to mark sections that are not written in the sheet music, e.g., cadenzas. Given the performance sequence π, the audio recording can be segmented into time intervals $I_1, \ldots, I_d$ such that time interval I_i corresponds to the section in the audio data where bar π_i is played (or something that is not written in the score in case $\pi_i = \uparrow$).

Given a query sequence of bars $Q = (q_0, \ldots, q_m)$, Q a substring of δ, the task of sheet music-audio identification is to find all time intervals T in the audio data where the query sequence of bars is played. More formally,

$$\begin{aligned} H(Q) \quad := \{T \mid \quad & \exists j : Q = (\pi_j, \pi_{j+1}, \ldots, \pi_{j+m}) \\ & \wedge T = I_j \cup I_{j+1} \cup \ldots \cup I_{j+m}\} \end{aligned}$$

denotes the set of *hits* w.r.t. Q. Note that in case of repeats that are notated as repeat signs, there can be more than one hit for a given query. Also note that besides the time intervals T there might be other time intervals in the audio data where the same musical content is played, but that belong to a different sequence of bars in the sheet music. We denote this kind of time intervals as *pseudo-hits*.

3 TWO APPROACHES

Given a scanned sheet music representation and an audio recording of the same piece of music, in a first step we use optical music recognition (OMR) software to extract information about musical symbols like staffs, bars and notes from the sheet music scans. Note that the obtained symbolic score data usually suffers from recognition errors. For simplicity, we here assume that the set of bar labels B and the default sequence δ are correctly obtained from the OMR output. Given a query $Q = (q_0, \ldots, q_m)$, which is a substring of δ, we want to find the set of hits $H(Q)$ as specified in Section 2. We now describe two approaches with different preconditions.

For the first approach, we assume that the performance sequence $\pi = (\pi_1, \ldots, \pi_d)$, $\pi_i \in B \cup \{\uparrow\}$, is known. In this case, we are left with the calculation of the corresponding time intervals $I_1, \ldots, I_d$. This can be done by using sheet music-audio synchronization. The set of hits $H(Q)$ can then be computed by finding occurrences of the query sequence in the performance sequence.

In the second approach, the performance sequence π is unknown. In this case, a reasonable strategy is to use sheet music-audio matching to search for sections in the audio recording with a similar musical content compared to the query sequence of bars. These sections may be considered as an approximation of the set of hits $H(Q)$. However, one should be aware of the fact that this method cannot distinguish correct hits from pseudo-hits, and is therefore expected to deliver false positives. In the following, we will refer to such false positives as *content-induced confusion*. Such confusion is also expected to be introduced by query sequences that differ only slightly, either in musical content or by a very small number of bars at the beginning or end of the sequence. This issue becomes particularly relevant, since the presence of OMR errors prohibits using too strict settings for rating similarity in the matching.

Due to the additional information π that is given in the first approach, this approach works much more robust and reliable than the second approach. The required performance sequence π can be created with little effort by manually editing an automatically generated list of jump directives acquired from the available default sequence δ. Therefore, we consider this approach semi-automatic. On the contrary, the second approach is fully automatic, but the results are less reliable. In the optimum case, only content-induced confusion would occur. In practice, however, extra confusion is likely to be introduced by shortcomings of the matching procedure.

The idea followed in this paper is to use the more reliable results of the semi-automatic first approach to create ground truth results for evaluating the less reliable fully automatic second approach. Using this method, we compare different settings of the matching procedure used in the second approach to learn which one works best for the task of sheet music-audio identification.

4 DATA PREPARATION

To compare sheet music data with audio data, we first create MIDI note events from the OMR results. However, OMR results often suffer from non-recognized or misclassified symbols. Especially in orchestral scores with many parts, erroneous or missing clefs and key signatures lead to wrong note pitches when creating MIDI events. Furthermore, orchestral scores can comprise parts for transposing instruments, i.e., the notated pitch is different from the sounding pitch. Such transposition information is not output by current OMR software, but it is essential for creating correctly pitched MIDI events. To be able to handle even complex orchestral scores, a so-called *staff signature* text file is generated from each page and is manually corrected. The staff signature file contains information about the clef, the key signature and the transposition at the beginning of each staff that is found on the page, see Figure 1. It also identifies which staffs belong to the same grand staff. The information from the staff signature files is used to correct errors in the OMR output and to add the missing information about transposing instruments.

There are several choices to be made regarding onset times and tempo, when creating the MIDI events from the OMR results. Since in the OMR output, notes or beams

Clef	Key Signature	Transposition
treble	+2	0
treble	+3	-7
treble	-1	-3
treble	+4	-2
tenor	+2	0
treble	0	-7
tenor	+2	0
alto	+2	0
bass	+2	0
bass	+2	0
bass	+2	0

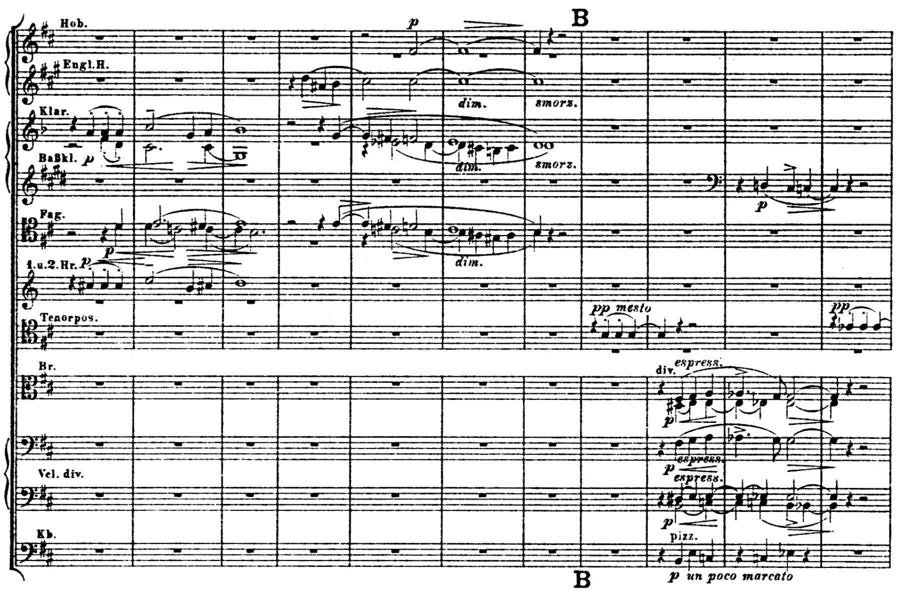


Figure 1. Staff signature annotation for an example grand staff taken from a score of the "Symphony to Dante's Divina Commedia S109 - Inferno" by Franz Liszt. Positive key signature values count the number of sharps, negative values count the number of flats. Transposition values are specified as the amount of semitones the pitch has to be modified with to sound correctly.

are often missed out, the accumulated note durations are not a good estimator for note onset times. This is especially the case for scores with multiple staffs and possibly multiple voices per staff, where the voice onset times might drift apart. Instead we use the horizontal position of notes within each measure as an estimator for the onset time. Even though this does not deliver onset times that perfectly match the musical meter, this method is very robust against surrounding errors and effectively inhibits voices from drifting apart.

Another parameter that is required to convert sheet music data to MIDI events is the tempo. This parameter is usually not output by OMR systems. If the performance sequence π is known in advance, the mean tempo can be calculated from the duration of the audio track. When π is not known, one might either use a fixed tempo or try to estimate a tempo based on the musical content. Note that the actual tempo used in audio interpretations can easily vary from 40 to 220 beats per minute (quarter notes per minute). We will investigate the effects of different tempo estimation strategies in our experiments in Section 8.

Both the MIDI data and the audio data are converted to sequences of normalized chroma-based features. Each feature is a 12-dimensional vector encoding the local energy distribution among the 12 traditional pitch classes of Western classical music commonly labeled C, C$^\sharp$, D, . . .,B.

5 SYNCHRONIZATION

After transforming both the MIDI data as well as the audio data into sequences of normalized chroma vectors, we use dynamic time warping (DTW) to synchronize the two sequences. Here, the main idea is to build up a cross-similarity matrix by computing the pairwise distance between each score chroma vector and each audio chroma vector. In our implementation, we simply use the inner vector product for the comparison. An optimum-cost alignment path is determined from this matrix via dynamic programming. To speed up this computationally expensive procedure, we use an efficient multiscale version of DTW.

6 MATCHING PROCEDURE

The task of the matching procedure is to find sections in the audio interpretation that are considered similar to a given query of score data. In this paper, we use a variant of the *subsequence dynamic time warping* algorithm for this task. For details we refer to the literature [2]. As in the case of synchronization, both the audio data and the score data are first converted to feature sequences. Each feature vector from the score query is compared to each feature vector from the audio database by means of a suitable *local cost measure*. The results of this comparison are stored in a cost matrix, see Figure 2. Finding candidate matches from this cost matrix means finding paths connecting the bottom row and the top row of the matrix. In particular, we are interested in paths p where the sum of the local cost of the matrix cells covered by the path is as small as possible. Such paths are calculated using dynamic programming by iteratively advancing from the bottom left towards the top right using a constrained set of allowed step directions ensuring that a path never runs backwards in time. For each matrix cell, the minimum cost of any valid path leading to that cell is saved in a so-called *accumulated cost matrix*. Matches are then identified by finding minima in the top row of the accumulated cost matrix.

Given a query bar sequence Q, the matching procedure outputs a set of matches $M(Q) = \{(p_1, c_1), \ldots, (p_N, c_N)\}$, where p_i is a path connecting the top and bottom rows and $c_i \in \mathbb{R}_{\geq 0}$ is the cost of

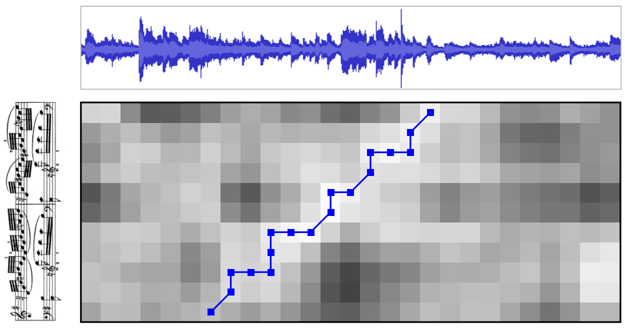


Figure 2. Illustration of the subsequence DTW cost matrix for a score query with a length of two measures accounting for 11 seconds of MIDI data (Beethoven Sonata 3, Opus 2 No 3, Adagio, measures 16–17). An excerpt of 27 seconds of audio data including one correct match is displayed. The optimum-cost path p for the correct match is rendered as a sequence of squares connected by lines.

the path p_i. The results are ranked with respect to the path cost. The choice of allowed step directions can be varied and associated step weights can be introduced to favor certain directions and behaviors. Several settings for step directions and step weights will be discussed in our experiments in Section 8.

7 EVALUATION PROCEDURE

Sheet music-audio matching depends on a multitude of parameters and settings used in the steps of creating MIDI events, creating feature sequences, and performing the matching procedure. In this work, we are interested in finding out which parameters work best for the task of sheet music-audio identification. We do this by evaluating and comparing several parameter sets on a test dataset consisting of a collection of musical *tracks*, with each track being represented by one sheet music representation and one audio interpretation.

In the evaluation, we perform the matching procedure on a set of test queries. For each test query Q, we then evaluate the matching results $M(Q)$ using a set of ground truth hits $H(Q)$ and a suitable confusion measure. To calculate the confusion measure, we first identify which matches output by the matching procedure correspond to ground truth hits. Let $T = [t_0, t_1] \in H(Q)$ be the ground truth hit and $(p, c) \in M(Q)$ be a match whose path p corresponds to the time interval $T' = [t'_0, t'_1]$ in the audio. The match (p, c) is then considered to correspond to the ground truth hit T, if both the durations and the locations roughly coincide. More precisely, with $\Delta := t_1 - t_0$ and $\Delta' := t'_1 - t'_0$ we require that

$$|\Delta' - \Delta| < 0.2\Delta \quad \text{and} \quad |t'_1 - t_1| < 0.2\Delta.$$

In the following, we call a match that corresponds to a ground truth hit a *correct match* and a match that does not correspond to a ground truth hit an *incorrect match*. Let $M(Q) = \{(p_1, c_1), \ldots, (p_N, c_N)\}$ be the set of all matches for a query Q, and let $C \subseteq [1 : N]$ be the set of indices of correct matches and $I \subseteq [1 : N]$ be the set of indices of incorrect matches. The confusion measure we

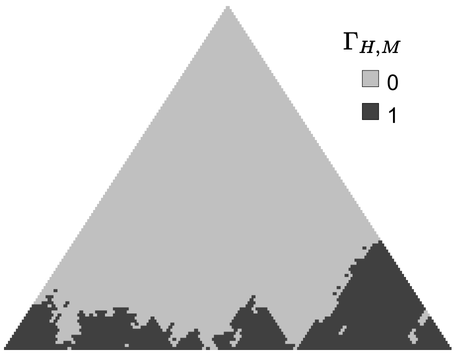


Figure 3. Scape plot for Beethoven's Piano Sonata no.7 op.10 no.3 Rondo (Allegro) using the confusion measure $\Gamma_{H,M}$.

use in this paper is a binary-valued function $\Gamma_{H,M}$ that on input Q takes the value 1 if at least one ground truth hit in $M(Q)$ has no corresponding match or if there is an incorrect match with lower cost than the highest-cost correct match, and 0 otherwise:

$$\Gamma_{H,M}(Q) := \begin{cases} 1 & \text{missed ground truth hit} \\ 1 & \min_{i \in I} c_i < \max_{i \in C} c_i \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $\Gamma_{H,M}(Q) = 0$ if all ground truth hits are found and are ranked higher than any incorrect match. In case of $\Gamma_{H,M}(Q) = 1$ we also speak of *confusion*.

Using the results of sheet music-audio synchronization that have been calculated in a preprocessing step, a set of ground truth hits can be calculated for any input query sequence of bars Q that is a substring of δ. This allows us to test each track using a grid of queries that covers not only the whole track but also a wide range of query lengths. The results can be nicely visualized in a so called *scape plot* [10]. Figure 3 shows a scape plot using the confusion measure $\Gamma_{H,M}$. Time runs from left to right. The lowest row shows the results for the shortest query length. The query length successively increases when moving upwards in the plot. The darker shaded areas indicate confusion.

From Figure 3, one can see that longer queries lead to less confusion and better separability of correct and incorrect matches. The plot also reveals where in the track and up to what query lengths the confusion happens. To not only be able to visually compare parameters for each individual track, but to also enable comparisons for the whole dataset, we summarize the results of all queries in one number per track by simply averaging over the complete grid of queries. Subsequently, we calculate the average over all tracks to end up with a single number for each set of parameters. If one parameter set works better than another parameter set, this fact should manifest in a lower average $\Gamma_{H,M}$ value. Note that one should not compare absolute values of the confusion measure for different tracks or datasets, because the absolute values depend on too many uncontrolled factors like the content-induced confusion, the tempo of the audio interpretation, and the content-dependent "uniqueness" of bars. Therefore, we keep datasets fixed, when studying the effects of using

Composer	Work	Instrumentation	#Pages	#Tracks	Duration
Beethoven	Piano Sonatas 1–15	Piano	278	54	5h 01min
Liszt	"A Symphony to Dante's Divina Commedia"	Symphonic Orchestra	145	2	44min
Mendelssohn	Concert in E minor, Op.64	Violin and Orchestra	55	3	26min
Mozart	String Quartetts 1–13	String Quartett	190	46	2h 46min
Schubert	"Die schöne Müllerin", "Winterreise" and "Schwanensang"	Singer and Piano	257	58	3h 04min

Table 1. Information and statistics on the test dataset used for evaluation.

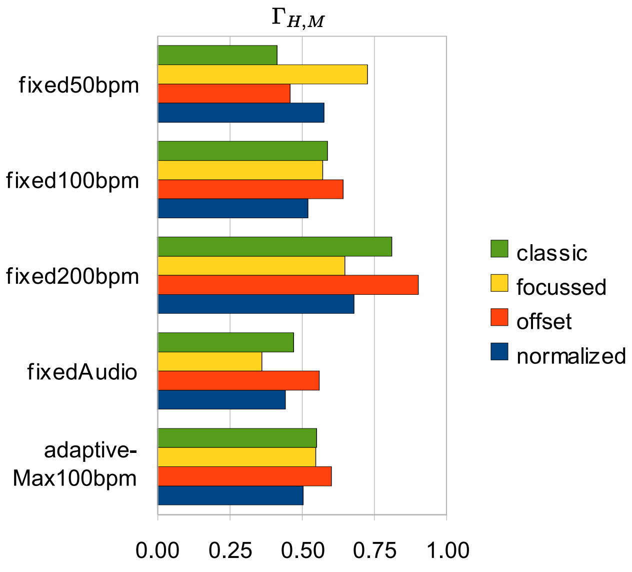


Figure 4. $\Gamma_{H,M}$ values averaged over the complete dataset for every combination of 5 tempo estimation strategies and 4 step direction and cost settings. Lower values are better.

different parameters by comparing the confusion measure values.

8 EXPERIMENTS AND RESULTS

Using the procedures described in the previous sections, there are many aspects whose effect on sheet music-audio identification should be investigated. Due to space limitation, we restrict ourselves to investigating the effects of different tempo estimation strategies in combination with different step settings and cost settings in the subsequence DTW. In particular, we test five tempo estimation strategies: **`fixedXXXbpm`**: Fixed tempo of `XXX` beats per minute, with `XXX` taking the values 50, 100 and 200. **`fixedAudio`**: Fixed mean tempo of the corresponding audio interpretation (estimated via manually annotated π and the duration of the audio file). **`adaptiveMax100bpm`**: The tempo is determined individually for each bar by taking into account the number of different onset times within the bar. The tempo is chosen such that the duration of the bar is 200ms times the number of different onset times. This leads to bars with runs of short-duration notes being slowed down compared to bars with long notes. Additionally, a maximum tempo of 100bpm is used to limit the difference between slow and

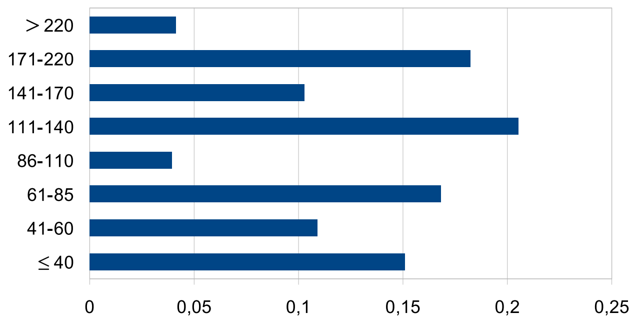


Figure 5. Tempo distribution of the test dataset being weighted the same way as the results in Figure 4

fast bars.

We use four different step and cost settings for the subsequence DTW. **`classic`**: Step vectors $(1,0),(0,1),(1,1)$ and cost weights $1,1,1$. **`focussed`**: Step vectors $(2,1),(1,2),(1,1)$ and cost weights $2,1,1$. **`offset`**: Same as `classic`, but with an additional cost offset of 1 which is added to each cell of the local cost matrix. **`normalized`**: The same as `classic`, but with an additional modification at the stage of calculating the accumulated cost matrix. At each matrix cell, the cost being compared for making the decision about which step vector leading to this cell delivers the minimum accumulated cost are normalized by the accumulated path length up to this cell. This normalization prevents short paths being preferred over long paths, even if the short paths have a higher average cost.

The dataset used for testing consists of 5 sheet music books covering a range of instrumentations and complexities, see Table 1. One audio inpterpretation per track is included. For each track in the dataset, we calculate the $\Gamma_{H,M}$ value for a grid of queries similar to the one used to create the scape plot in Figure 3. We start with a query length of 5 bars and use a hop size of 5 bars to move throughout the track. The query length is successively increased by 5 bars up to a maximum query length of 40 bars.

Figure 4 shows the results for testing all 20 combinations of settings on the test dataset. The $\Gamma_{H,M}$ values illustrated in the figure are average values calculated by first taking the average over all tracks within each scorebook, and then taking the average over all scorebooks. This way, each of the five different types of instrumentation and complexity gets the same weight. Since we are measuring effects that depend on the tempo, we also need to look at the distribution of tempi of the tracks in the test dataset. Figure 5 shows the distribution of tempi being weighted the same way as the results in Figure 4 and confirms that there is no bias towards slower or higher tempi that might distort our results.

From the results in Figure 4 we can see that both the tempo estimation strategy and the tested step direction and cost settings clearly have an effect on the average amount of confusion. The best overall results are achieved by the setting `focussed` when using the mean tempo of the audio interpretation. This was expected, since this setting is more focussed towards the diagonal direction and, therefore, benefits the most from the fact that the tempo is known. However, in cases where the difference between the estimated tempo and the actual tempo of the interpretation becomes too large, the lack of flexibility leads to confusion, as can be seen for the tempo strategies `fixed50bpm` and `fixed200bpm`.

In the cases, where the tempo of the audio interpretation is assumed to be unknown, the best results are achieved by the setting `classic` using the `fixed50bpm` tempo estimation strategy. Both settings `classic` and `offset` work best when the estimated tempo is low. A possible explanation for this effect is that the accumulating cost lead to a preference of short paths. Shorter paths contain less steps and therefore accumulate less cost. When looking at the cost matrix depicted in Figure 2, one may think of the optimum-accumulated-cost paths tending to make shortcuts towards the top of the cost matrix instead of following the lane of minimum local cost. This effect leads to additional confusion when the estimated tempo of the sheet music data is high compared to the actual tempo of the audio interpretation.

The setting `normalized` delivers better results than the `classic` and `offset` settings for every tempo estimation strategy except for the `fixed50bpm`. For that strategy, however, it clearly falls behind and leads to even worse results than in the `fixed100bpm` case. A possible explanation is that, in contrast to the settings `classic` and `offset`, the setting `normalized` does not prefer shorter paths over longer paths. This seems to be an advantage when the estimated tempo is not too low, but in the `fixed50bpm` case, the lack of a driving force towards keeping the path connecting the bottom and top rows short causes paths to become much more sensitive to noise and local dissimilarities.

The `adaptiveMax100` yields only a tiny improvement over the `fixed100bpm` estimation. The reason for that probably is that the difference between the two strategies usually affects only the slower pieces. A test run using only the slower pieces might lead to a bigger advantage for the adaptive strategy.

9 CONCLUSIONS

We introduced and discussed the task of sheet music-audio identification, which is identifying sections of an audio recording where a given query sequence of bars from the sheet music is played. Two approaches to solving the task have been described, a semi-automatic approach using synchronization and a fully automatic approach using matching techniques. We proposed a workflow that allows for evaluating the matching approach by using results from the more reliable synchronization approach. This workflow includes contributions that make it possible to perform synchronization and matching even for complex orchestral scores. We introduced the idea of using scape plots to visualize results of matching or retrieval tasks that are performed on a grid of test queries covering a complete track of music over a wide range of query lengths. Finally, we performed an evaluation using a subsequence DTW based matching technique for the task of sheet music-audio identification. Results were presented and discussed for different sets of settings and tempo estimation strategies.

In our future work, we would like to investigate more aspects of sheet music-audio identification to answer questions like the following: Which features work best? What is the optimum feature resolution? Can the results be improved by using a harmonic model on the MIDI events created from the sheet music? What influence do OMR errors have on the results? Besides comparing the amount of confusion, we are also interested in comparing the temporal accuracy of matches.

10 ACKNOWLEDGEMENTS

We would like to express our thanks to the Bavarian State Library in Munich for their cooperation and for providing the sheet music scans.

11 REFERENCES

[1] C. Fremerey, M. Müller, F. Kurth, and M. Clausen: "Automatic Mapping of Scanned Sheet Music to Audio Recordings," *Proc. ISMIR, Philadelphia, USA*, pp. 413–418, 2008.

[2] M. Müller: *Information Retrieval for Music and Motion*, Springer, 2007.

[3] F. Soulez, X. Rodet, and D. Schwarz: "Improving Polyphonic and Poly-instrumental Music to Score Alignment," *Proc. ISMIR, Baltimore, USA*, pp. 143–148, 2003.

[4] N. Orio: "Alignment of Performances with Scores Aimed at Content-Based Music Access and Retrieval," *Proc. ECDL, Rome, Italy*, pp. 479–492, 2002.

[5] C. Raphael: "Aligning Music Audio with Symbolic Scores Using a Hybrid Graphical Model," *Machine Learning*, Vol. 65 No. 2–3 pp. 389–409, 2006.

[6] R.B. Dannenberg and C. Raphael: "Music Score Alignment and Computer Accompaniment," *Communications of the ACM*, Vol. 49 No. 8 pp. 38–43, 2006.

[7] R.J. Turetsky and D.P.W. Ellis: "Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses," *Proc. ISMIR, Baltimore, USA*, pp. 135–141, 2004.

[8] J. Pickens, J.P. Bello, G. Monti, T. Crawford, M. Dovey, and M. Sandler: "Polyphonic Score Retrieval Using Polyphonic Audio Queries: A Harmonic Modeling Approach," *Proc. ISMIR, Paris, France*, pp. 140–149, 2002.

[9] I.S.H. Suyoto, A.L. Uitdenbogerd, and F. Scholer: "Searching Musical Audio Using Symbolic Queries," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16 No. 2 pp. 372–381, 2008.

[10] C. Sapp: "Comparative Analysis of Multiple Musical Performances," *Proc. ISMIR, Philadelphia, USA*, pp. 497–500, 2008.

SMERS: MUSIC EMOTION RECOGNITION USING SUPPORT VECTOR REGRESSION

Byeong-jun Han, Seungmin Rho
School of Electrical Engineering
Korea University
{hbj1147, smrho}@korea.ac.kr

Roger B. Dannenberg
School of Computer Science
Carnegie Mellon University
rbd@cs.cmu.edu

Eenjun Hwang
School of Electrical Engr.
Korea University
ehwang04@korea.ac.kr

ABSTRACT

Music emotion plays an important role in music retrieval, mood detection and other music-related applications. Many issues for music emotion recognition have been addressed by different disciplines such as physiology, psychology, cognitive science and musicology. We present a support vector regression (SVR) based music emotion recognition system. The recognition process consists of three steps: (i) seven distinct features are extracted from music; (ii) those features are mapped into eleven emotion categories on Thayer's two-dimensional emotion model; (iii) two regression functions are trained using SVR and then arousal and valence values are predicted. We have tested our SVR-based emotion classifier in both Cartesian and polar coordinate system empirically. The result indicates the SVR classifier in the polar representation produces satisfactory result which reaches 94.55% accuracy superior to the SVR (in Cartesian) and other machine learning classification algorithms such as SVM and GMM.

1. INTRODUCTION

With the recent advances in the field of music information retrieval, there is an emerging interest in (automatically) analyzing and understanding the emotional content of music. Due to the diversity and richness of music content, many researchers have been pursuing a multitude of research topics in this field, ranging from computer science, digital signal processing, mathematics, and statistics applied to musicology and psychology. Many computer scientists [1][2] have focused on music retrieval by using musical meta-data (such as title, genre or mood) as well as low-level feature analysis (such as pitch, tempo or rhythm), while music psychologists [3][4] have been interested in studying how music communicates emotion.

Currently, there is no standard method to measure and analyze emotion in music. However, a psychological model of emotion has found increasing use in computational studies. Thayer's two-dimensional emotion model [5] offers a simple but quite effective model for placing emotion in a two-dimensional space. In the model, the amount of arousal and valence is measured along the vertical and horizontal axis, respectively.

The goal of this paper is to develop a music emotion recognition system for predicting the arousal and valence of a song based on audio content. First, we analyzed seven different musical features (such as pitch, tempo, loudness, tonality, key, rhythm and harmonics) and mapped them into eleven categories of emotion: angry, bored, calm, excited, happy, nervous, peaceful, pleased, relaxed, sad and sleepy. This categorization is based on Juslin's theory [3] along with Thayer's emotion model [5]. Secondly, we adopt support vector regression (SVR) [6] as a classifier to train two regression functions for predicting arousal and valence values based on the low-level features, such as pitch, rhythm and tempo, extracted from music. In addition, we compared our SVR-based method with other classification algorithms such as GMM (Gaussian Mixture Model) and SVM (Support Vector Machine) to evaluate the performance.

In the following section, we present a brief overview on the current state-of-the-art music recognition systems, and emotion models. In Section 3, we illustrate a musical feature extraction scheme and give an overview of our proposed system. Section 4 describes our proposed SVR-based music emotion recognition method. Experimental results are given in Section 5. In the last section, we conclude the paper with some observations and future work.

2. RELATED WORK

Many researchers have explored models of emotions and factors that give rise to the perception of emotion in music. Many other researchers investigate the problem of automatically recognizing emotion in music.

2.1 Music and Emotion

Traditional mood and emotion research in music has focused on finding psychological and physiological factors that influence emotion recognition and classification. During the 1980s, several emotion models were proposed, which were largely based on the dimensional approach for emotion rating.

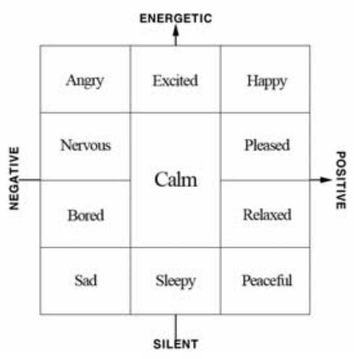


Figure 1. Modified Thayer's 2-dimensional emotion model

The dimensional approach focuses on identifying emotions based on their location on a small number of dimensions such as valence and activity. Russell's [7] circumflex model has had a significant effect on emotion research. This model defines a two-dimensional, circular structure involving the dimensions of activation and valence. Within this structure, emotions that are across the circle from one another, such as sadness and happiness, correlate inversely. Thayer [5] suggested a two-dimensional emotion model that is simple but powerful in organizing different emotion responses: stress and energy. The dimension of stress is called valence while the dimension of energy is called arousal.

As shown in Figure 1, the two-dimensional emotion plane can be divided into four quadrants with eleven emotion adjectives placed over them. We use eleven types based on Juslin's theory and Thayer's emotion model.

During the last decade, many researchers have investigated the influence of music factors like loudness and tonality on the perceived emotional expression [3][5]. They analyzed those factors using diverse techniques, some of which are involved in measuring psychological and physiological correlation between the state of particular musical factor and emotion evocation. According to the [3], Juslin and Sloboda investigated the utilization of acoustic cues in the communication of music emotions by performers and listeners and measured the correlation between emotional expressions (such as anger, sadness and happiness) and acoustic cues (such as tempo, spectrum and articulation).

2.2 Music Emotion Recognition

Automatic emotion detection and recognition in speech and music is growing rapidly with the technological advances of digital signal processing and various effective feature extraction methods. Emotion recognition can play an important role in many other potential applications such as music entertainment and human-computer interaction systems.

One of the first studies of emotion detection in music is presented by Feng *et al.* [8]. Their work, based on Computational Media Aesthetics (CMA), analyzes two dimensions of tempo and articulation which are mapped into four categories of moods: happiness, anger, sadness and fear. Lie et al. [4] developed a hierarchical framework for extracting music emotion automatically from acoustic music data. They used music intensity to represent the energy dimension of Thayer model, and timbre and rhythm for the stress dimension.

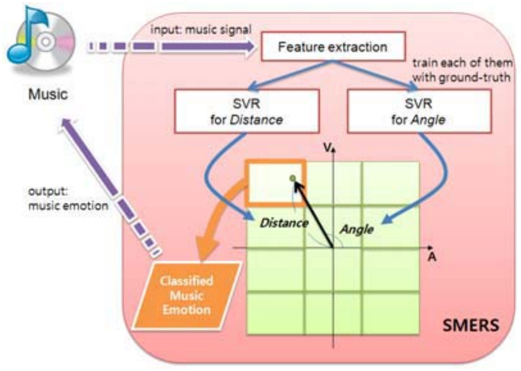


Figure 2. System diagram of the SMERS

FEELTRACE [9] is software that is designed to let observers track the emotional content of stimuli (such as words, faces, music, and video) as they perceive it and taking full account of gradation and variation over time. Yang et al. [10] developed a music emotion recognition (MER) system from a continuous perspective and represented each song as a point in the emotion plane. They also proposed a novel arousal/valence computation method based on regression theory.

3. IMPLEMENTATION

In this paper, we implemented a music recognition system, called SMERS (SVR-based Music Emotion Recognition System). The system diagram is shown in Figure 2 and the details are described as follows.

3.1 System Description

The SMERS mainly consists of three steps: (i) Feature extraction: Seven distinct musical features are extracted and analyzed (Details are described in the Section 3.3); (ii) Mapping: Extracted features are mapped into eleven emotion categories on Thayer's two-dimensional emotion model; (iii) Training: The system uses extracted features as input vectors to train the SVR. We use two distinct SVR functions in a polar coordinate system: one is for *distance* from origin (0, 0) to the emotion in a Thayer-like coordinate system, and the other is for *angle*. Using these two trained SVRs, the system predicts each song's emotion. Based on empirical test results, the polar coordinate system is a better representation than the obvious Cartesian coordinates. (More details about training procedure in both Cartesian and polar coordinate systems are presented in Section 4.1).

3.2 Dataset

The music dataset for training the SMERS is made up of 165 western pop songs. We collected the 15 songs in each of eleven categories of emotion from the large music database, All Music Guide [11], which provides 180 emotional categories for classifying entire songs. To build classifiers we used Support Vector Regression (SVR) and our implementation is based on the LIBSVM library [12], which gives almost full functionalities for SVR training.

3.3 Musical Features

In this paper, we consider various musical features including scale, intensity, rhythm, and harmonics and use them as an input vector in the emotion recognition system.

3.3.1 Scale

Scale is an overall rule of tonic formation of music. In our study, we defined scale as a set of key, mode, and tonality. For accurate scale features, we first analyzed the chromagram for representing the frequencies in musical scales. After that, we applied the key profile matrix by Krumhansl [13]. The following equations show the process of combining chromagram and key characterization:

$$Tonality = \mathbf{C} \cdot \mathbf{KeyProfileMatrix} \tag{1}$$

$$Key = \max_{KeyIndex} \left(Tonality\ (Idx) \right) \tag{2}$$

, where vector C has 12 elements and represents the summed chromagram analyzed for each acoustic frame. KeyProfileMatrix is a key profile matrix, which is composed of 12-by-24 elements. KeyIndex indexes KeyProfileMatrix, where KeyIndex=1,2,…,24. After the inner product of C and KeyProfileMatrix in Equation (1), we obtain a tonality score for each key. Finally, we can obtain the most appropriate key by picking the key having maximum tonality in Equation (2).

3.3.2 Average Energy (AE)

Average energy (AE) of the overall wave sequence is widely adopted to measure the loudness of music. Also, standard deviation (σ) of AE measures the regularity of loudness. Those are defined as:

$$\mathrm{AE}(x) = \frac{1}{N}\sum_{t=0}^{N} x(t)^2, \quad \sigma(\mathrm{AE}(x)) = \sqrt{\frac{1}{N}\sum_{t=0}^{N}\left(\mathrm{AE}(x) - x(t)\right)^2} \tag{3}$$

, where x is an input discrete signal, t is the time in samples, and N is the length of x in samples.

3.3.3 Rhythm

Rhythm, which is composed of rhythmic features such as tempo and beat, is one of the most important elements in music. Beat is a fundamental rhythmic element of music. Tempo is usually defined as the beats per a minute (BPM) which is used to represent the global rhythmic feature of music. Tempo and regularity of beats can be measured in various ways. For beat tracking and tempo analysis, we used the algorithm by Ellis et al. [14]. The features we use are overall tempo (in beats per minute) and the standard deviation of beat intervals, which indicates tempo regularity.

3.3.4 Harmonics

Harmonics can be observed in musical tones. In monophonic music, harmonics are easily observed in the spectrogram. However, it is hard to find harmonics in polyphony, because many instruments and voices are performed at once. To solve this problem, a method to compute harmonic distribution yields

$$\mathrm{HS}(f) = \sum_{k=1}^{M} \min\left(\|\mathrm{X}(f)\|, \|\mathrm{X}(kf)\|\right) \tag{4}$$

Here, M denotes the maximum number of harmonics considered, f is the fundamental frequency, and X is the short-time Fourier transform (STFT) of the source signal. In the equation, the min function is used in such a way that only the strong fundamental and strong harmonics result in a large value for HS. In our implementation, we measured average of each frequency using (4) and then computed their standard deviation to define the harmonic feature.

4. EMOTION RECOGNITION

4.1 Training Process

There are some essential conditions needed for effective emotion recognition. Firstly, the regression function should be trained as perfectly close to ground-truth as it can. If the trained regression function cannot generate proper Arousal/Valence (AV) values for a music emotion adjective, the separation policy also cannot act in a proper way. Secondly, a proper music emotion separation policy on the AV plane should be presented. It acts like a decoder or quantizer of AV values. If the separation policy does not reflect the natural mapping between emotion adjectives and AV values, system might have to learn more complex mapping from features to the AV values.

Our music emotion separation policy in the AV plane is shown in Figure 3. In case of Cartesian representation, the emotion of a song can be represented by (a, v), where a denoting arousal and v denoting valence and their ranges are $a \in [-1,1]$ and $v \in [-1,1]$, respectively. There are also 5 separating lines: $v=v^{(+)}$, $v=v^{(-)}$, $v=0$, $a=a^{(+)}$, and $a=a^{(-)}$. These lines separate the AV plane in 11 areas. As shown in Figure 3, each area has a center point, which is drawn as a black dot. These dots are used as the ground-truth data for training SVRs. On the other hand, the blank dots are outputs of the SVR-based on feature vectors extracted from songs.

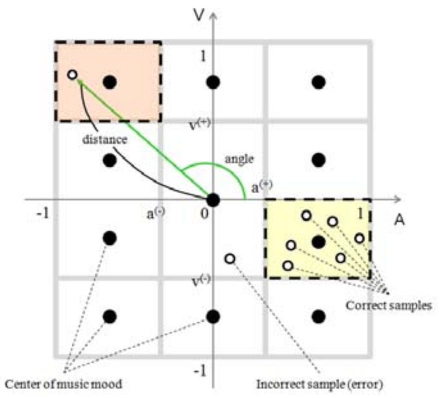


Figure 3. Music emotion separation policy in AV plane (in both Cartesian and polar representation)

For training our emotion classifier, we need two distinct SVR functions. One is for training an arousal value and the other is for a valence value. The training is performed by the musical features of songs as input and the center values of each music emotion as the desired output. Our test verifies whether or not the outputs (arousal and valence values) of trained regression functions are within the range of the proper music emotion in AV plane.

Using Cartesian coordinates, we found that some emotions such as "Peaceful" and "Bored" are misclassified into the "Calm" emotion category in the center of the AV plane. We decided to train using polar coordinates as the desired output to see if that would produce better results.

Assume that $Emotion_c$ and $Emotion_p$ represent an emotion in Cartesian and polar coordinate systems, respectively. We can calculate the *distance* and *angle* values of each emotion and transfer the coordinate system from Cartesian to polar using the following equations:

$$\begin{aligned} &Emotion_C = (Arousal, Valence) \\ &Emotion_P = (Distance, Angle) \\ &s.t. \begin{cases} Distance = \left(Arousal_C{}^2 + Valence_C{}^2\right)^{\frac{1}{2}} \\ Angle = \arctan\left(\dfrac{Arousal_C}{Valence_C}\right) \end{cases} \end{aligned} \tag{5}$$

$$\begin{cases} Arousal_P = Distance \cdot \cos(Angle) \\ Valence_P = Distance \cdot \sin(Angle) \end{cases} \tag{6}$$

4.2 Classification Methods

4.2.1 Support Vector Regression (SVR)-based Training

The basic idea of regression is to determine a function that accurately approximates target values using input values. SVR [6] is an application of SVM to find the mapping function between input and output. There are two major training strategies of SVR. One is ε-SVR, which employs ε-insensitive loss function to solve the quadratic optimization problem. However, ε-SVR has the following limitations: ε should be set before training the SVR model. Also, it is hard to anticipate the range of ε in most problems. The other strategy, named υ-SVR [15], solves the limitations of ε-SVR by limiting the task of finding ε to the quadratic optimization problem.

For the training sets $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with $x_i \in \boldsymbol{R}^n$, $y_i \in \boldsymbol{R}$, and i=1, 2, …, n. The relation between the input x_i and output y_i can be mapped by an optimal regression function $f(x)$ by SVR training. As the result of training, the difference between trained function output from input and ground-truth of input should be lower than the error ε. Assuming linearity, f can be represented as the following hyperplane: $f(x) = \omega \cdot \Phi(x) + b$, where $\omega \in \boldsymbol{R}^n$, $b \in \boldsymbol{R}$, and Φ denotes a nonlinear transformation from $\boldsymbol{R}^n$ to a high-dimensional space.

Our goal is to find the value ω and b. The values of x can be determined by solving following quadratic optimization problem:

$$\begin{aligned} &\min \quad \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\left(\xi_i + \xi_i^* + \nu\varepsilon\right) \\ &\text{s.t.} \quad \begin{cases} -(\varepsilon + \xi_i) \le y_i - (\omega \cdot \Phi(x_i) + b) \le \varepsilon + \xi_i \\ e \ge 0, n \in (0,1] \end{cases} \end{aligned} \tag{7}$$

, where C is a constant value. With some data points α_i and α_i^*, we can write ω to $\omega = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)\Phi(x_i)$, so that f can be rewritten as:

$$f(x) = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)\Phi(x_i) \cdot \Phi(x) + b = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)k(x_i, x) + b \tag{8}$$

, where k is known as the kernel function. On the other hand, (7) can be solved by transforming to the Lagrange function and getting its multipliers, α_i and α_i^*, as indicated in [16]. These are called support vectors and meaningful when they are nonzero values. Also we can get optimal b and ε by the Kuhn-Tucker condition. In our system, we employed Radial Basis Function (RBF) as a kernel function instead of using linear or polynomial functions due to its flexibility.

4.2.2 Support Vector Machine (SVM)-based Training

For emotion classification, we used multi-class SVM. Since SVM classifies only one class at a time, we trained 11 SVMs to classify each emotion separately. This set of classifiers receives input feature vectors extracted from music. Each classifier generates a probability that the music has a specific emotion. The highest probability value determines the final selection of a single emotion label for the music.

4.2.3 Gaussian Mixture Model (GMM)-based Training

All musical features are modeled using Gaussian Mixture Models (GMMs). We use 7 Gaussian models for arousal and valence sets. Each GMM is trained using the Expectation Maximization (EM) algorithm. The step of GMM-based classification is as follows: first of all, 3 and 4 GMMs were trained for labeling arousal and valence, respectively. Next, the two GMMs sets produce two classifications for arousal and valence, respectively. For example, the GMMs set for arousal labeling could classify A is lower than -1/3, between -1/3 and 1/3, or higher than 1/3. In final step, music emotion is determined by combining the results from two GMMs sets.

5. EXPERIMENTS AND RESULTS

In this section, we evaluate the effectiveness of our emotion recognition system in terms of accuracy. Coefficients for SVR, SVM and GMM and kernels are very critical to performance. In our experiment, we tried to find the optimal classification parameters empirically. We also considered the *v*-fold cross-validation method in order to prevent the over-fitting problem. We tested *v*-fold cross-validations using different *v* values.

The best SVR training parameters and optimum values in both Cartesian and polar representation are shown in Table 1 and 2, respectively. We searched for optimal values of all parameters (except "# of folds in cross validation") in steps of about 7%. Moreover, cross validations were carried out 54 times for each step.

In order to evaluate Cartesian coordinate system-based classification methods, we employed three types of classifiers: SVMs with one-to-one training policy, SVR, and GMM. First of all, in SVMs-based classification, one-to-one training policy was employed, since SVM does not support multi-classification basically. In SVR-based classification, we trained two regression functions to represent arousal and valence respectively. Finally, GMM was trained following the procedure in Section

Table 1. SVR training parameters and obtained optimums in Cartesian representation

Name of parameters	Range	Optimum
Nu (υ)	$2^{-5} \sim 2^{-0.1}$	$2^{-1.7}$
Gamma of RBF (g)	$2^{-20} \sim 2^{-0.1}$	$2^{-8.3}$
Cost (C)	$1 \sim 2^{15}$	$2^{7.4}$

Table 2. SVR training parameters and obtained optimums in polar representation

Name of parameters	Distance	Angle
Nu (υ)	2^{-8}	2^{-8}
Gamma of RBF (g)	2^{-10}	2^{-4}
Cost (C)	2^{8}	2^{6}
mean squared error	0.02498	0.09834

4.2.3. On the other hand, in polar coordinate system-based classification, two SVRs, which represent *distance* and *angle* respectively, were trained.

5.1 Confusion Matrix

Confusion matrices of each coordinate system combined with each classifier are presented in Figure 4. As shown in Figure 4, the errors of both SVMs and SVR in Cartesian coordinate system were comparably higher than both GMM in Cartesian coordinate system and SVR in the polar coordinate system.

The result of SVMs in the Cartesian coordinate system, presented in Figure 4(a), was good on specific music emotions such as angry, bored, and peaceful. However, most other diagonal elements had poor results.

The change from multi SVMs to SVR increased the performance as shown in Figure 4(b). On average, 9.5

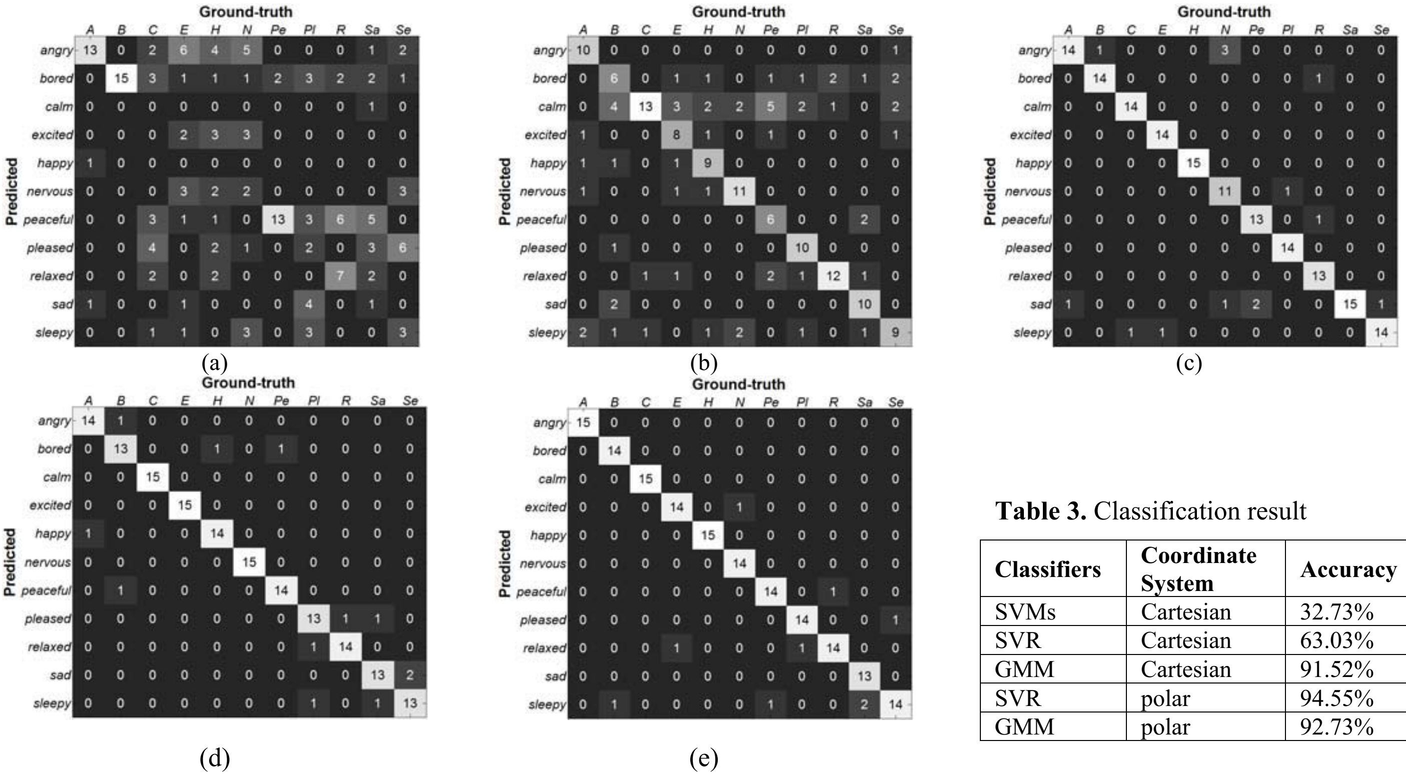


Figure 4. Confusion matrices: Cartesian coordinate system with (a) SVMs (b) SVR (c) GMM, and polar coordinate system with (d) GMM and (e) SVR.

Table 3. Classification result

Classifiers	Coordinate System	Accuracy
SVMs	Cartesian	32.73%
SVR	Cartesian	63.03%
GMM	Cartesian	91.52%
SVR	polar	94.55%
GMM	polar	92.73%

songs were correctly classified, but still some emotions had errors. It can be seen that 12.73% of songs (21 songs) were misclassified into calm in Figure 4(b). This indicates that the calm problem should be solved first.

The result in Figure 4(c) and (d) is better than Figure 4(a) and (b). Most diagonal elements were well classified. In the case of GMM in the Cartesian coordinate system, 12.8 songs on average were classified correctly. However, there is still a concentration of misclassification in some emotions such as angry (4 songs), sad (5 songs), and sleepy (2 songs). However, SVR in the polar coordinate system showed that the imbalanced classifications were significantly reduced: the average number of correct classification was 14.2 songs, and also, misclassification was concentrated only in relaxed (2 songs) and sleepy (3 songs).

5.2 Accuracy

The results are shown in Table 3. In the experiments based on Cartesian coordinate systems, maximum accuracy was 91.52% (151 of 165 samples). By changing coordinate system into polar, the accuracy was increased to 94.55% (156 of 165 samples) using SVR and 92.73% (153 of 165 samples) using GMM.

6. CONCLUSIONS AND FUTURE WORK

In this paper, automatic emotion recognition of music has been evaluated using various machine learning classification algorithms such as SVM, SVR and GMM. In our experiment, it is shown that the SVR-based classification in the polar coordinate system remarkably improved the accuracy of the emotion recognition from 63.03% to 94.55%. However, the GMM classification with polar coordinates only improved from 91.52% to 92.73%.

For further research, more perceptual features should be considered and other classification algorithms such as fuzzy and kNN (k-Nearest Neighbor). We also plan to compare the result of machine learning (ML)-based emotion recognition with human performed arousal/valence data.

7. REFERENCES

[1] W. Birmingham, R. Dannenberg and B. Pardo: "An Introduction to Query by Humming with the Vocal Search System," *Communications of the ACM*, Vol. 49 (8), pp. 49-52, 2006.

[2] S. Rho, B. Han, E. Hwang and M. Kim: "MUSEMBLE: A Novel Music Retrieval System with Automatic Voice Query Transcription and Reformulation," *Journal of Systems and Software (Elsevier)*, Vol. 81(7), pp. 1065-1080, 2008.

[3] P.N. Juslin and J.A. Sloboda: "Music and Emotion: Theory and research," *Oxford Univ. Press*, 2001.

[4] L. Lie, D. Liu and Hong-Jiang Zhang: "Automatic Mood Detection and Tracking of Music Audio Signals," *IEEE Trans. on ASLP*, Vol. 14(1), 2006.

[5] R. E. Thayer: "The Biopsychology of Mood and Arousal," *New York: Oxford University Press*, 1989.

[6] Smola, Alex J., et al.: "A tutorial on support vector regression," *Statistics and Computing*, Vol.14, pp.199-222, 2004.

[7] J. A. Russell: "A Circumplex Model of Affect," *Journal of Personality and Social Psychology*, Vol. 39, 1980.

[8] Y. Feng, Y. Zhuang, Y. Pan : "Music information retrieval by detecting mood via computational media aesthetics," *Proc. of IEEE/WIC Intl. Conf., Web Intelligence*, pp. 235-241, 2003.

[9] E. Cowie, *et al.*: "'FEELTRACE': An instrument for recording perceived emotion in real time," *Proc. of Speech Emotion*, pp. 19–24, 2000.

[10] Y.H. Yang, et al.: "A regression approach to music emotion recognition," *IEEE Trans. on ASLP*, Vol. 16 (2), pp. 448–457, 2008.

[11] "The All Music Guide," Available: http://www.allmusic.com.

[12] Chih-Chung. Chang, and Lin, Chih-Jen: "LIBSVM: a library for support vector machines," 2001. Available: http://www.csie.ntu.edu.tw/ ~cjlin/libsvm.

[13] C. Krumhansl: "Cognitive foundations of musical pitch," *Oxford University Press*, 1990.

[14] D. Ellis, P.W. Poliner, E. Graham: "Identifying 'Cover Songs' with chroma features and dynamic programming beat tracking," *IEEE Conf. on ICASSP*, Vol. 4, 1429-1432, 2007.

[15] B. Schölkopf, et. al.: "New support vector algorithms," *Neural Computation*, Vol.12, 2000.

[16] A. Smola, T. Freiß, B. Schölkopf: "Semiparametric support vector and linear programming machines," *Nuero COLT TR*, NC2-TR-1998-024, 1998.

[17] Hsu, Chih-Wei., and Lin, Chih-Jen: "A comparison of methods for multiclass support vector machines," *IEEE Trans. on Neural Networks*, Vol.13(2), pp.415-425, 2002.

MUSIC MOOD AND THEME CLASSIFICATION - A HYBRID APPROACH

Kerstin Bischoff, Claudiu S. Firan, Raluca Paiu, Wolfgang Nejdl
L3S Research Center,
Appelstr. 4, Hannover, Germany
{bischoff,firan,paiu,nejdl}@L3S.de

Cyril Laurier, Mohamed Sordo
Music Technology Group,
Universitat Pompeu Fabra
cyril.laurier@upf.edu
mohamed.sordo@upf.edu

ABSTRACT

Music perception is highly intertwined with both emotions and context. Not surprisingly, many of the users' information seeking actions aim at retrieving music songs based on these perceptual dimensions – moods and themes, expressing how people feel about music or which situations they associate it with. In order to successfully support music retrieval along these dimensions, powerful methods are needed. Still, most existing approaches aiming at inferring some of the songs' latent characteristics focus on identifying musical genres. In this paper we aim at bridging this gap between users' information needs and indexed music features by developing algorithms for classifying music songs by moods and themes. We extend existing approaches by also considering the songs' thematic dimensions and by using social data from the *Last.fm* music portal, as support for the classification tasks. Our methods exploit both audio features and collaborative user annotations, fusing them to improve overall performance. Evaluation performed against the *AllMusic.com* ground truth shows that both kinds of information are complementary and should be merged for enhanced classification accuracy.

1. INTRODUCTION

General music perception – *i.e.* how we think and talk about music – is heavily influenced by emotions and context. Consequently, users' music information seeking behavior also reflects the importance of opinion/mood and theme associations for music songs. Searching for music usually is an exploratory and social process, in which people make use of collective knowledge, as well as the opinions and recommendations of other people [1]. Related is their need for contextual metadata expressing, for example, which situations/events are often associated with the songs. Thus, besides directly searching or browsing music by artist or title, associated usage, theme/main subject and mood/emotional state are used in every third (navigational) query [1]. Similarly, [2] found that the majority of music queries from a search engine log falls into these categories – 30% of the queries are theme-related (*e.g.* "party music", "wedding songs") and 15% target mood information. Such statistics thus show the necessity of indexing music collections according to mood and theme classes.

Hence, our goal in this paper is to automatically derive mood and theme metadata for music tracks to better cover diverse facets reflecting the complex real-world music information needs of users. With the "mood of a song" we denote the state or the quality of a particular feeling induced by listening to that song (*e.g. aggressive*, *happy*, *sad*, *etc.*). The "theme of a song" refers to the context or situation which fits best when listening to the song, *e.g. at the beach*, *night driving*, *party time*, *etc.*

Currently available state-of-the-art music search engines still do not explicitly support music retrieval based on mood and theme information, and content-based approaches trying to address this problem mainly focus on identifying the moods of songs and do not tackle the thematic aspects of the music resources. Several works in Music Information Retrieval have shown a potential to model the mood from audio content (like [3–6], see [7] for an extensive review). Although this task is quite complex, satisfying results can be achieved if the problem is reduced to simple models [7]. However, an important limitation of these approaches is that they concentrate on the mood only expressed in the audio signal itself and can not capture other sources of emotionality.

Apart from analyzing the low-level features of music resources to identify the songs' corresponding mood or theme, another powerful source of information that can be used are Web 2.0 portals. Collaborative tagging platforms have become extremely popular in recent years – users associate descriptive keywords to various types of content (*e.g.* pictures, Web pages, music). Especially for multimedia data, such as music, the gain provided by the newly available textual information is substantial, since with most prominent search engines on the Web, users are currently still constrained to search for music using textual queries.

The contributions of the paper are twofold:

- We show the feasibility of automatic music classification according to contextual aspects like themes.
- We successfully exploit collective knowledge in form of tags in order to complement the intrinsic information derived from audio features.

The algorithms can be used in various ways: predicted mood and theme labels can be indexed to enrich the metadata index of music search engines enabling a more social and context-aware search (or browsing). Besides, such labels will be valuable for recommendation and playlist generation, *e.g.* for listening to "Party Time"-like songs.

2. RELATED WORK

Music enrichment recently focuses on deriving mood information based on extracted acoustic data [3–5]. [3] proposes a content-based method, tailored to classical music, that uses the Thayer's model [8] for classification. For detecting the mood of music, timbre, intensity and rhythm, features are extracted and a Gaussian Mixture Model is used to model each feature set. In [4], the authors propose a schema such that music databases are indexed on four labels of music mood: "happiness", "sadness", "anger" and "fear". The relative tempo of the music tracks, the mean and standard deviation of average silence ratio are used to classify moods, using a neural network as classifier. For automatically detecting mood for music tracks, [5] uses a set of 12 mood classes which are not mutually exclusive. However, the main focus of the paper is creating a ground truth database for music mood classification.

Several existing papers aim at automatically inferring additional information from available content as well as (user generated) metadata. [9] present a music retrieval system that uses supervised multiclass Naïve Bayes classification for learning a relationship between acoustic features and words from expert reviews on songs, thus enabling query-by-text for music. Similarly, [10, 11] aim at enriching songs with textual descriptions for improving music IR. [10] uses a variant of the AdaBoost algorithm, FilterBoost, in order to predict social tags of the songs based on the information captured in the audio features. Nevertheless, the tags learned by the classifier pertain to multiple categories of tags (genres, styles, moods and contexts) and there is no special focus on mood and theme-related tags, like in our case. [11] compares five methods for collecting tags: user surveys, harvesting social tags, annotation games, mining web documents and auto-tagging audio content. Again, here there is no discussion about the performance of the described methods for predicting mood and theme tags. Moreover, both [10, 11] are not comparable with our approach, since there is no clear definition for mood and theme classes and the data sets on which evaluation was performed differ from ours.

[12] and [13] investigate social tags for improving music recommendations – [12] to attenuate the cold-start problem by automatically predicting additional tags based on the learned relationship between existing tags and acoustic features, [13] to make better recommendations based on the latent factors hidden in user-tag-item relations. For this, the authors successfully apply Higher Order Singular Value Decomposition on the triplets. Again, while both approaches make use of *Last.fm* to predict (the likelihood) of all kinds of tags, our work explicitly focuses on inferring mood and theme annotations.

In [14], *Last.fm* user tags have been used together with content-based features for automatic genre classification. Two classification strategies are proposed that make implicit use of tags: A graph of music tracks is constructed that captures their semantic similarity in terms of tags associated. Both the baseline low-level feature only classifier as well as a single-layer classifier, considering audio features and implicit tag similarity simultaneously, are clearly outperformed by a double-layer classifier, which firsts learns genre labels based on audio information and then iteratively updates its models considering the tag-based neighborhood of tracks.

Thus it seems that especially for multimedia user generated tags are valuable, since low-level features may not be expressive enough. [15] found that *Last.fm* tags define a low-dimensional semantic space which - especially at the track level highly organized by artist and genre - is able to effectively capture sensible attributes as well as music similarity. Somewhat complementary to our approach, [16] aims at studying the relationships between moods and artists, genres and usage metadata. As a test set for the experiments, the authors use *AllMusic.com*, *Epinions.com* and a subset of *Last.fm* data. The authors point out an interesting finding: Many of the individual mood terms were highly synonymous, or described aspects of the same underlying mood space. The experiments also showed that decreasing the mood vocabulary size in some ways clarified the underlying mood of the items being described.

We use *Last.fm*'s valuable folksonomy [1] information for inferring mood and theme labels for songs. While in earlier experiments only tags were used for deriving moods, themes and styles/genres [17], in this paper we also investigate fusion with audio-based methods. Extending existing music metadata enrichment studies, we fuse social tags and low-level audio features of the tracks to infer mood or theme labels showing that both sources provide helpful complementary information.

3. DATA SETS

AllMusic.com. In 1995, the *AllMusic.com* (AMG) website was created as a place and community for music fans. Almost all music genres and styles are covered, ranging from the most commercial/popular to very obscure ones. Not only genres can be found on *AllMusic.com*, but also reviews of albums and artists within the context of their own genres, as well as classifications of songs and albums according to themes, moods or instruments. All these reviews and classifications are manually created by music experts from the *AllMusic.com* team, therefore the data found here serves as a good ground truth corpus.

For our experiments, we collected *AllMusic.com* pages corresponding to music themes and moods, finding 178 different moods and 73 themes. From the pages corresponding to moods and themes, we also gathered information related to which music tracks fall into these categories. This way, we ended up with 5,770 songs. Looking at the

[1] folk + taxonomy: collaboratively created classification scheme

Nr.	**MOOD CLUSTERS – MIREX**
MM1	Passionate, Rousing, Confident, Boisterous, Rowdy
MM2	Rollicking, Cheerful, Fun, Sweet, Amiable/Good natured
MM3	Literate, Poignant, Wistful, Bittersweet, Autumnal, Brooding
MM4	Humorous, Silly, Campy, Quirky, Whimsical, Witty, Wry
MM5	Aggressive, Fiery, Tense/Anxious, Intense, Volatile, Visceral

Nr.	**MOOD CLUSTERS – THAYER**
MT1	**high energy / high stress:** Tense/Anxious, Angst-Ridden, Spooky, Eerie, Rowdy, Fiery, Angry, Fierce, Provocative, Boisterous, Hostile, Aggressive, Volatile, Rebellious, Confrontational, Paranoid, Outrageous, Unsettling, Brittle
MT2	**high energy / low stress:** Rollicking, Exuberant, Happy, Sexy, Exciting, Energetic, Party/Celebratory, Intense, Gleeful, Lively, Cheerful, Fun, Rousing, Freewheeling, Carefree, Passionate, Playful, Gritty, Joyous,
MT3	**low energy / low stress:** Calm/Peaceful, Sentimental, Cathartic, Soft, Romantic, Springlike, Warm, Precious, Laid-Back/Mellow, Confident, Hypnotic, Naive, Intimate, Innocent, Relaxed, Soothing, Dreamy, Smooth, Gentle
MT4	**low energy / high stress:** Sad, Melancholy, Detached, Whimsical, Gloomy, Ironic, Snide, Somber, Autumnal, Wry, Wintry, Plaintive, Yearning, Austere, Bittersweet, Fractured, Bleak, Cynical/Sarcastic, Bitter, Acerbic

Table 1. (Samples from) Mood clusters

songs identified in each of the categories, we have 8,158 track-mood and 1,218 track-theme assignments. On average songs are annotated with 1.73 moods and 1.21 themes respectively, with maximum number of annotations of 12 and 6 respectively.

***Last.fm*.** For the tracks collected from *AllMusic.com*, we obtained the *Last.fm* tags users had assigned to these songs together with the corresponding frequencies. *Last.fm* is a popular UK-based Internet radio and music community website. In a comparative study on tagging [2] found that the majority of the generally accurate and reliable user tags on *Last.fm* fall into the genre category (60%). Considerably less frequent are tags referring to moods/opinions/ qualities (20%) or themes/context/usage (5%) of the music songs. According to [15], at the track level the tags often name the genre and artist of a song. As not all *AllMusic.com* songs have user tags in *Last.fm*, our set of tracks is reduced to 4,737. Using the AudioScrobbler API, we collected in total 59,525 different tags for this set of songs.

Audio. For each track from the previous collections found in our audio database, we have a 30 seconds excerpt in mp3 format with a bitrate of 192 kbps. From these audio tracks, we automatically extracted several state-of-the-art MIR audio features of different type: timbral, tonal, rhythmic including MFCCs, BPM, chroma features, spectral centroid and others. Please refer to [7] for a complete list. For each excerpt of the data set, its 200ms frame-based extracted features were summarized with their component-wise means and variances. At the end of the process, we obtained 240 low-level and mid-level audio features.

4. MOOD AND THEME CLASSIFICATION

For predicting themes and moods, we base our solution on social knowledge – *i.e.* collaboratively created tags associated to music tracks – extracted from *Last.fm*, as well as on audio information. Building upon already provided user tags, on the audio content of music tracks, or on combinations of both, we build multiclass classifiers to infer additional annotations corresponding to moods and themes.

4.1 AllMusic.com Class Clustering

Given that the number of classes existing in *AllMusic.com* is quite large (*e.g.* 178 different moods) with many of the individual terms being highly synonymous or denoting the same concept in well known models of emotions [2] [16], clustering was applied to the initial set of *AllMusic.com* moods as well as the themes.

Mood Clustering. For comparison reasons, we choose the five mood categories used for the MIREX Audio Music Mood Classification Track (see Table 1). Each of the clusters is a collection of five to seven *AllMusic.com* mood labels that together define the cluster. These categories were proposed in [16], derived from a popular set (of Top Songs, Top Albums). The MIREX mood clusters effectively reduce the diverse mood space, and yet root in the social-cultural context of pop music [3]. Restricting our data set to tracks whose assigned moods fall into exactly one of these categories, we had 1192 distinct songs left for machine learning. To balance cluster size for our multiclass classifiers the cutoff was set to 200 instances per cluster.

Since many *AllMusic.com* mood labels and thus the corresponding songs classified by human experts are not used in MIREX, we as well experimented with the well known two-dimensional models of emotion/mood. In the Thayer energy-stress model [8], emotions are classified along the two axes of (low - high) energy and (low - high) stress. Thus, the two factors divide the mood space into the four clusters "exuberance", "anxious/frantic", "depression" and "contentment". Similarly, Russell/Thayer's bipolar model differentiates emotions based on arousal and valence. In the psychological literature there is little agreement on the number of basic emotional categories or dimensions. However, the Thayer model has been proven useful for music classification and the four categories resulting seem a fair compromise: reducing the mood space to enable clear classificatory distinction and still providing valuable extra-musical metadata for exploratory information needs. During clustering all *AllMusic.com* labels were manually mapped into the two-dimensional mood space by the authors adopting a similarity sorting method as described below for themes. The four resulting clusters are shown together with some example *AllMusic.com* labels in Table 1. Again, clusters were balanced by randomly choosing 403 instances for each cluster.

Theme Clustering. Since *AllMusic.com* themes do not directly correspond to human emotions, mapping the 73 theme terms into the mood spaces used before was not possible (though themes may often be strongly related to specific moods). For manual clustering, we adopted a similarity sorting procedure, in which all *AllMusic.com* themes written on cards were sorted by the authors into as many and as high piles as appropriate. Co-occurrence matrices

[2] Moods are considered to be similar to emotions, but being longer in duration, less intensive and missing object directedness

[3] `http://www.music-ir.org/mirex2007/index.php/Audio_Music_Mood_Classification`

Nr.	THEME CLUSTERS
T1	Party Time, Birthday Party, Celebration, Prom, Late Night, Guys Night Out, Girls Night Out, At the Beach, Drinking, Cool & Cocky, TGIF, Pool Party, Club, Summertime
T2	Sexy, Seduction, Slow Dance, Romantic Evening, In Love New Love, Wedding, Dinner Ambiance
T3	Background Music, Exercise/Workout Music, Playful The Sporting Life, Long Walk, The Great Outdoors, Picnic, Motivation, Empowering, Affirmation, The Creative Side, Victory, Day Driving, Road Trip, At the Office
T4	D-I-V-O-R-C-E, Heartache, Feeling Blue, Breakup, Regret, Loss/Grief, Jealousy, Autumn, Rainy Day, Stay in Bed, Solitude, Reminiscing, Introspection, Reflection, Winter, Sunday Afternoon

Table 2. Theme clusters

were built and added to find good groupings by analyzing the clusters. Unclear membership of singular labels was resolved after discussion. Applying this method resulted in a theme list with 13 labels. Classes containing too few songs are discarded in order to have a minimal representative learning corpus for the classifier, such that the remaining four theme clusters (Table 2) contain 74 songs each.

4.2 Classification

The core of our mood and theme classification methods are multiclass classifiers trained on the *AllMusic.com* ground truth using tags or audio information as features. We experiment both with classifiers created separately for the two different types of features we consider, which are then combined in order to produce for each song a final mood/ theme classification, as well as with a classifier taking as input a combination of audio and tag features. After several experiments, we could observe that SVM classifiers with Radial Basis Function (RBF) kernel performed best for the case of audio input features (it outperformed Logistic Regression, Random Forest, GMM, K-NN and Decision Trees), whereas in the case of tag features, Naïve Bayes Multinomial achieved the best performance. Additionally, the linear combination of the separate classifiers for audio and tag features performed better than the classifier trained on the combination of audio and tag features. Only the best obtained classification results are presented in this paper. We have classifiers trained for the whole set of classes (*i.e.* either for moods or themes) and these classifiers produce for every song in the test set a probability distribution over all classes (*e.g.* over all moods). The highest probability is considered in order to assign the songs to the corresponding class. We experimented with feature selection based on automatic methods (*e.g.* Information Gain) but the results showed that the full set is better suitable for learning, even though it contains some noise.

Algorithm 1 presents the main steps of our classification approach, where classifiers are trained separately for the two different types of input features – tags and audio information. We show the algorithm for mood classification, the case of themes classification being similar.

Step 1 (optional) of the algorithm above aims at reducing the number of mood classes to be predicted for the songs. If two classes are clustered, the resulted class will contain all songs which have been originally assigned to any of the composing classes. As we need a certain amount of input data in order to be able to consistently train the classifiers, we discard those classes containing less than a certain number of songs[4] assigned (step 2).

Alg. 1. Mood classification
Input:ftype – feature type

$$ftype = \begin{cases} 0, \text{for tag features;} \\ 1, \text{for audio features.} \end{cases}$$

M – mood classes to be learned
S_{total} *– set of songs*

```
1:  Apply clustering method to cluster moods (see Section 4.1)
2:  Select classes of moods $M$ to be learned
        For each mood class
            If the class does not contain at least $X$ songs
                Discard class
3:  Classifier learns a model
3a: Split song set $S_{total}$ into
            $S_{train}$ = songs used for training the classifier
            $S_{test}$ = songs used for testing the classifiers' learned model
3b: Select features for training the classifier
      If $(ftype = 0)$ // tag features
          For each song $s_i \in S_{train}$
              Create feature vector $F_t(s_i) = \{t_j | t_j \in T\}$, where
              $T$ = set of tags from all songs in all mood classes
              $t_j = \begin{cases} log(freq(t_j) + 1), \text{if } s_i \text{ has tag } t_j; \\ 0, \text{otherwise.} \end{cases}$
      Else // audio features
          For each song $s_i \in S_{train}$
              Create feature vector $F_a(s_i) = \{a_j | a_j \in A\}$, where
              $A$ = set of audio features from all songs in all mood classes
              $a_j = standardize(a_j)$
3c: Train and test classifier
      If $(ftype = 0)$ // tag features
        Train Naïve Bayes (NB) on $S_{train}$ using $\{F_t(s_i); s_i \in S_{train}\}$
        Test Naïve Bayes (NB) on $S_{test}$
      Else // audio features
          Train SVM on $S_{train}$ using $\{F_a(s_i); s_i \in S_{train}\}$
          Test SVM on $S_{test}$
4:  Classify songs into mood classes
    For each song $s_i \in S_{total}$
    If $(ftype = 0)$ // tag features
          Compute probability distribution $P_t(s_i)$ as
          $P_t(s_i) = \{p_{NB}(m_j|s_i); m_j \in M\}$
          Assign $s_i$ to $m_j$, where $max(p_{NB}(m_j|s_i))$
    Else // audio features
          Compute probability distribution $P_a(s_i)$ as
          $P_a(s_i) = \{p_{SVM}(m_j|s_i); m_j \in M\}$
          Assign $s_i$ to $m_j$, where $max(p_{SVM}(m_j|s_i))$
```

After selecting separate sets of songs for training and testing in step 3a, we build the feature vectors corresponding to each song in the training set (step 3b). In the case of features based on tags, the vectors have as many elements as the total number of distinct tags assigned to the songs belonging to the mood classes. The elements of a vector will have values depending on the frequency of the tags occurring along with the song. We experimented with different variations for computing the vector elements, but the formula based on the logarithm of the tag frequency provided best results. Audio features are standardized for better suitability with the SVM classifier. Here, a one-vs-one multiclass approach was taken with the parameters selected via grid search (C and gamma with 3-fold cross validation method). Probability estimations are made by pairwise coupling [18].

Once the feature vectors are constructed, they are fed into the classifier and used for training. The assignment of a song to a class is done based on the maximum predicted probability for a song among all possible classes

[4] The threshold depends on the type of clustering and class type. The exact numbers are given in Section 4.1.

Classifier	Class	R	P	F1	Acc
SVM (audio)	Mood MIREX	0.450	0.442	0.420	0.450
NB (tags)	Mood MIREX	0.565	0.566	0.564	0.565
Comb ($\alpha = 0.7$)	Mood MIREX	0.575	0.573	0.572	0.575
SVM (audio)	Mood THAYER	0.517	0.515	0.515	0.517
NB (tags)	Mood THAYER	0.539	0.542	0.539	0.539
Comb ($\alpha = 0.8$)	Mood THAYER	0.570	0.569	0.569	0.569
SVM (audio)	Themes clustered	0.528	0.581	0.522	0.527
NB (tags)	Themes clustered	0.595	0.582	0.575	0.595
Comb ($\alpha = 0.9$)	Themes clustered	0.625	0.617	0.614	0.625

Table 3. Experimental results: P, R, $F1$, Acc for the different classifiers and mood/theme classes

(step 4). As already mentioned, we also experiment with a linear combination of the predictions of the two separately trained classifiers (details are presented in Algorithm 2).

Alg. 2. Mood classification – classifiers' linear combination
Input: M – mood classes to be learned
S_{total} – set of songs

```
1: For each song $s_i \in S_{total}$
      Compute $P_a(s_i) = \{p_{SVM}(m_j|s_i)\} = \{p_a(m_j|s_i)\}$
      and $P_t(s_i) = \{p_{NB}(m_j|s_i)\} = \{p_t(m_j|s_i)\}$ (see Alg. 1, step 4)
2: For each $\alpha$=0.1,...,0.9, $step$=0.1
      For each song $s_i \in S_{total}$
         For each mood $m_j \in M$
            $p_{at}(m_j|s_i) = \alpha \cdot p_a(m_j|s_i) + (1-\alpha) \cdot p_t(m_j|s_i)$
      Assign $s_i$ to $m_j$, where $max(p_{at}(m_j|s_i))$
      Compute $P$, $R$, $Acc$, $F1$
3: Select $\alpha = \alpha_{best}$ that produces best results for $P$, $R$, $Acc$, $F1$
4: Classify songs into mood classes, using $\alpha_{best}$ for weighting the
   probabilities outputted by the audio-based classifier and $(1-\alpha_{best})$ for
   weighting the probabilities predicted by the tag-based classifier.
```

The two different classifiers are first trained to make predictions for all songs in the collection. For producing a linear combination of the classifiers as final output, we then experiment with different values of the α parameter.

5. EVALUATION

For measuring the quality of our theme and mood predictions, we compare our output against the AllMusic experts' assignments, using Precision (P), Recall (R), Accuracy (Acc) and F1-Measure ($F1$) for the evaluation. We present the best results achieved among all our experimental runs (10-fold cross validations) in Table 3. These runs correspond to the different combinations of classifiers (audio-based, tag-based, or linear combinations of the two) and classes to be predicted (themes or moods clustered according to Mirex or Russell/Thayer resp.).

For both moods and themes, we observe that the classifiers relying solely on audio features perform worse than the pure tag-based classifiers. However, combining the two types of classifiers leads to improved overall results. For the moods clustered according to Mirex, Russell/Thayer and themes manually clustered, the best values of α are 0.7, 0.8 and 0.9 respectively. These values indicate a higher weight for the audio-based classifiers, though their achieved performance is poorer than that of the tag-based classifiers. This fact is easily explainable, due to the different types of classifiers considered: SVM for audio features and Naïve Bayes for tag features. It is known that Naïve Bayes produces probabilities close to 1 for the most likely identified class, whereas for the rest of classes, the probabilities are closer to 0. On the other hand, SVM produces more even probability distributions, therefore the high probabilities outputted by Naïve Bayes need to be evened out through a lower α weight. The variations of the $F1$ measure with α are depicted in Figure 1. The biggest variations are to be found in the case of moods clustered according to the Russell/Thayer model, where for values of $alpha$ starting with 0.7 we observe a sharp drop of the $F1$ value. For Mirex mood classes the $F1$ values start to deprecate with α values greater than 0.8.

The baseline accuracy for a random classifier trying to assign songs to the Russell/Thayer mood classes or to the theme clusters is 0.25, while for the Mirex mood classes it would be 0.2. The linear combination of the classifiers improves accuracy in the range of 10 to 27.7% for moods and 18.5% for themes over audio-based classifiers. Overall, results are better for theme classification, indicating that themes are easier to distinguish.

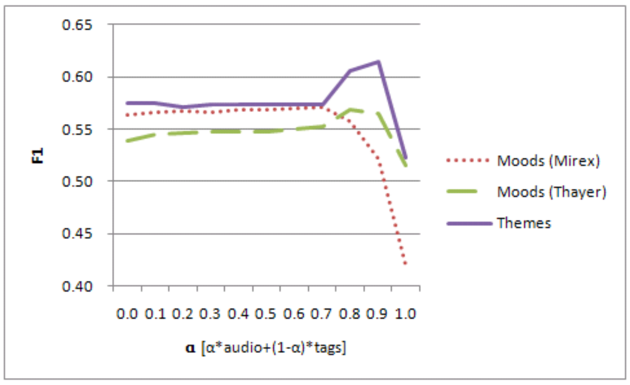
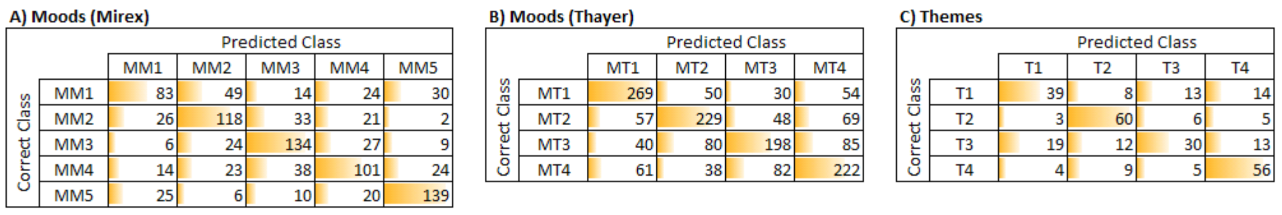


Figure 1. F1 values when varying the α parameter

Analyzing the confusion matrices for the best performing approaches (Figure 2), we observe some prominent confusion patterns: in the case of Mirex clustering, instances belonging to class $MM1$ are often misclassified into $MM2$, $MM4$ instances into $MM3$. Similarly, $MT3$ instances are wrongly classified into $MT4$ for the case of Russell/Thayer clustered moods; also $MT1$ and $MT4$ are often confused. For the latter, the energy dimension does not seem to ease differentiation, given that high stress (negative valence) is characteristic for both classes. $T3$ and $T1$ are the difficult theme classes. Further refinement of these classes should be considered for future work, in order to eliminate this kind of ambiguities (*e.g.* Exercise/Workout music might be as well considered Party-like music).

It is difficult to directly compare our results to the related work cited, as each paper uses a different number of classes. Moreover, experimental goals, ground truth and evaluation procedures vary as well, or detailed descriptions are missing. Comparing to the best algorithms submitted to the MIREX task, we achieve results with lower accuracy. However, knowing that the algorithm used in this paper for audio classification is the same as submitted to MIREX in



A) Moods (Mirex)

Correct Class	MM1	MM2	MM3	MM4	MM5
MM1	83	49	14	24	30
MM2	26	118	33	21	2
MM3	6	24	134	27	9
MM4	14	23	38	101	24
MM5	25	6	10	20	139

Predicted Class

B) Moods (Thayer)

Correct Class	MT1	MT2	MT3	MT4
MT1	269	50	30	54
MT2	57	229	48	69
MT3	40	80	198	85
MT4	61	38	82	222

Predicted Class

C) Themes

Correct Class	T1	T2	T3	T4
T1	39	8	13	14
T2	3	60	6	5
T3	19	12	30	13
T4	4	9	5	56

Predicted Class

Figure 2. Confusion matrix for the best approaches

2007 [19] (obtaining 60.5% accuracy), our conclusion is that the difference comes from the ground truth data. The hypothesis is that our results here are lower because we did not filter the training and test instances using listeners. Moreover for the MIREX collection, listeners were asked to focus on audio only (not lyrics, context or other), which makes it much easier then to classify using audio-based classifiers. In that context, the classification task on our MIREX-like *AllMusic.com* ground truth is more difficult.

6. CONCLUSION

Previous attempts to associate mood labels to music songs often rely on lyrics or audio information for clustering or classifying song corporas. Our algorithms exploit both audio information and social annotations from *Last.fm* for automatically classifying songs according to moods and themes and thus enriching music tracks with information often queried for by users. Themes capturing contextual aspects of songs, in particular, is a facet not considered so far in the literature. The algorithms proposed in this paper rely either on user tags, on audio features, or on combinations of both. Results of an evaluation performed against *AllMusic.com* experts' ground truth indicate that providing such mood and theme information is feasible. The results show that audio and tag information is complementary and should be merged in order to achieve improved overall classification performance. Using our algorithm, music also becomes searchable by associated themes and moods, providing a first step towards effectively searching music by textual, descriptive queries.

For future work, some of the promising ideas to be further investigated refer to refinements of the moods and themes clusters, as well as to other possible combinations of the audio and tag-based classifiers, *i.e.* metaclassifiers.

7. ACKNOWLEDGMENTS

This work was partially supported by the PHAROS project funded by the European Commission under the 6th Framework Programme (IST Contract No. 045035).

8. REFERENCES

[1] J. H. Lee and J. S. Downie: "Survey of music information needs, uses, and seeking behaviours: Preliminary findings," *ISMIR*, 2004.

[2] K. Bischoff, C. S. Firan, W. Nejdl, and R. Paiu: "Can all tags be used for search?," *CIKM*, pp. 193–202, 2008.

[3] D. Liu, L. Lu, and H.-J. Zhang: "Automatic mood detection from acoustic music data," *ISMIR*, 2003.

[4] Y. Feng, Y. Zhuang, and Y. Pan: "Popular music retrieval by detecting mood," *SIGIR*, 2003.

[5] J. Skowronek, M. McKinney, and S. van de Par: "A demonstrator for automatic music mood estimation," *ISMIR*, 2007.

[6] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. H. Chen: "A regression approach to music emotion recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16, No. 2, pp. 448–457, 2008.

[7] C. Laurier and P. Herrera: "Automatic detection of emotion in music: Interaction with emotionally sensitive machines," *Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence*, pp. 9–32, 2009.

[8] R. E. Thayer: *The biopsychology of mood and arousal*, Oxford University Press, 1989.

[9] D. Turnbull, L. Barrington, and G. Lanckriet: "Modeling music and words using a multi-class nave bayes approach," *ISMIR*, 2006.

[10] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere: "Autotagger: A model for predicting social tags from acoustic features on large music databases," *Journal of New Music Research*, Vol. 37, No. 2, pp. 115–135, 2008.

[11] D. Turnbull, L. Barrington, and G. Lanckriet: "Five approaches to collecting tags for music," *ISMIR*, 2008.

[12] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green: "Automatic generation of social tags for music recommendation," *NIPS*, 2007.

[13] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos: "Ternary semantic analysis of social tags for personalized music recommendation," *ISMIR*, 2008.

[14] L. Chen, P. Wright, and W. Nejdl: "Improving music genre classification using collaborative tagging data," *WSDM*, pp. 84–93, 2009.

[15] M. Levy and M. Sandler: "A semantic space for music derived from social tags," *ISMIR*, 2007.

[16] X. Hu and J. S. Downie: "Exploring mood metadata: Relationships with genre, artist and usage metadata," *ISMIR*, 2007.

[17] K. Bischoff, C. S. Firan, W. Nejdl, and R. Paiu: "How do you feel about "dancing queen"?: deriving mood & theme annotations from user tags," *JCDL*, pp. 285–294, 2009.

[18] T.-F. Wu, C.-J. Lin, and R. C. Weng: "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, Vol. 5, pp. 975–1005, 2004.

[19] C. Laurier and P. Herrera: "Audio music mood classification using support vector machine," *MIREX Audio Music Mood Classification contest, ISMIR*, 2007.

USING XML-FORMATTED SCORES IN REAL-TIME APPLICATIONS

Joachim Ganseman, Paul Scheunders
IBBT - Visionlab
Dept. of Physics, University of Antwerp
Universiteitsplein 1, building N
B-2610 Wilrijk (Antwerp), Belgium
{*joachim.ganseman, paul.scheunders*}*@ua.ac.be*

Wim D'haes
Mu Technologies NV
Singelbeekstraat 121
B-3500 Hasselt, Belgium
wim.dhaes@mu-technologies.com

ABSTRACT

In this paper we present fast and scalable methods to access relevant data from music scores stored in an XML based notation format, with the explicit goal of using scores in real-time audio processing frameworks. Quick and easy access is important when accessing or traversing a score, for instance for real-time playback. Any time complexity improvement in these contexts is valuable, while memory constraints are usually less important. We show that with some well chosen design choices and precomputation of the necessary data, runtime time complexity of several key score manipulation operations can be reduced to a level that allows use in a real-time context.

1. INTRODUCTION

In real-time audio processing software, the use of music scores is not commonplace. To fill that gap, we started the construction of a small C++ software library for handling MusicXML files, specially tailored for use in real-time audio processing software frameworks and streaming applications. Since a score is for many people a well-known way to represent music, we consider this important functionality that has been strangely absent in real-time audio frameworks until now.

Being able to use XML-encoded digital music scores natively in real-time environments has clear advantages over the alternatives that are often used now, like MIDI [1] or the development of an own ASCII-based format. The ability to use the countless mature software tools that are available for XML parsing and processing is the main reason to prefer XML-based formats over others. Nowadays most score file formats encode very detailed information, and XML formats can be easily extended or stripped to add or remove information, without needing to adapt the parser, which is much more difficult with binary or plain text file formats. Also most upcoming web developments are centered around XML-based standards (the semantic web etc.). For all of the aforementioned reasons, we will only consider XML-based file formats here.

MusicXML [2, 3] is the most widely used XML-based file format for scores at the moment, but others exist. MEI [4, 5] is a mature alternative and provides valuable functionality to encode versioning and history tracking in documents. The WEDELMUSIC format [6] was developed as all-round multimedia format in an academic context and seems not to be under active development anymore, but its legacy can more recently be found in MPEG SMR [7] and IEEE P1599/MX [8], that are both striving to include score information in a broader multimedia context.

We widen the scope of this paper to all of the aforementioned formats, as they are all XML-based and use similar hierarchies to encode scores. The principles outlined in this paper here thus hold for any of these formats. Also the programming language is of lesser importance: in any major object-oriented programming language, the argumentation for the design decisions will hold.

2. PROBLEM STATEMENT AND REQUIREMENTS

In real-time audio processing, audio data is processed frame by frame, the necessary operations need to be performed within a certain time, and then the results are written to an output buffer. The frames are usually kept small to minimize delays. This leads to very strict time constraints, as also the operating system's scheduler will lay claim to some time for other processes or interrupts.

A music score is layered on several levels (voices, instrument parts, chords), but these layers are often very much interlinked (like voice crossing). This makes it unfeasible to find an ideal single XML hierarchy to represent a score. The result is that notes, voices and/or parts that are active at the same time can be found encoded in very different places in the file. If you need to access all notes occuring at a single moment, you may need to access data at tens to hundreds of different positions in the score file. Processing time is very limited, and user interactivity or display of the score require also quick access to random positions in the score. This makes it becomes quickly undesirable to do the necessary data structure traversals in real-time, based only on the existing XML hierarchy.

XML-based score data formats tend to produce really large data structures: common uncompressed score files contain easily up to 250KB of text for a single A4 size page of piano solo music. When parsed into memory, this results in a relatively large XML tree. Since we envisioned use in real-time environments, we want to absolutely minimize any calculation time that is needed 'on-line' (during processing). There is no time to traverse an entire XML-tree to find the data that we need to access, as is commonplace in the visitor design pattern [9] as used in libmusicxml [10].

Because data that is 'scheduled' to occur at the same time, can be heavily dispersed throughout the file, a SAX-based approach to XML-parsing becomes difficult, and a DOM model is easier to handle. This made us decide to write our own data structure for the score, firmly based on the existing hierarchy of the format but with a few extra additions in functionality and precalculation of data. In the following sections of this paper, we will elaborate on the additions that we had to make to keep run-time calculation load as low as possible. To be able to parse XML files encoded in UTF-16, routines provided by the Unicode consortium can be used for the conversion of UTF-16 to UTF-8 data [11].

We set out to write a software library that could be used from real-time audio plug-in frameworks, as there are VST [12], AudioUnits [13] or RTAS [14]. In the end, we want to be able to use and manipulate a music score in a sequencer the same way we can use and manipulate an audio track. We need:

- quick access to all data.
- an easy method for timewise browsing through a score.
- easy extendability.
- cross-platform operation, documentation, testing ...

On top of that, in practice we need to adhere to several guidelines for real-time programming, amongst which some important ones are [15] :

- not allocating or deallocating memory
- avoiding denormalized floating point numbers

3. IT'S ABOUT TIME

3.1 Timestamping

Music scores are generally structured as follows: scores contain multiple parts or instruments, each of which consists of a series of measures that contain the notes. A score or a part can be subdivided into several sections, or within a measure, multiple voices may be separately encoded. We leave intermediate levels like these out here for clarity. Coda, segno and other repeat signs influence at what absolute time certain notes need to be played. This makes that music scores are rarely written to file in a way that is linear in time. The standard MIDI file format (SMF) comes close, but was never meant to be used for score encoding.

MusicXML stores timing information different than other formats: it only stores the note order and note length, and not the absolute position in the score at which the note occurs. This system was derived from the MuseData format [16]. In a real-time environment, we need absolute timestamps in order to know at any given time where we are in the score. These timestamps thus need to be calculated if they are not present. The quarter note as a unit of absolute time is the most convenient choice. This is portable across scores, whichever time signature they have, and across recordings, whichever tempo they are played in.

In MusicXML, in order to know at what absolute time in the score a specific note occurs, one needs to add up the length of all previous measures, and the previous notes in the same measure. This is a too intensive computation in real-time, therefore we need to precompute any absolute time values that we need in our application. The easiest way to do this is to keep track of a global absolute time value during parsing, and store a timestamp in every element that we encounter.

Having timestamps in the data structure has the following effect on run-time operations, with n the number of elements that need to have such a timestamp:

- worst-case time complexity to compute absolute time values in real-time decreases from O(n) to O(1),
- when a change occurs in the score, these values need to be updated throughout the score, introducing a penalty of O(n).

These figures assume that the notes in a measure are already sorted based on time, and that the timestamp of previous elements can be used to update the timestamp of later ones. Sorting on time is easy to accomplish by creating or overloading a comparison operator and running a generic sorting algorithm after parsing. In general, all time-modifying elements in the file format need to be processed during parsing to calculate timestamps for all notes. We found it handy to also store the time at which a certain timed element ends.

We need to add here that the increased complexity when changes occur to the score (measures or notes added, deleted or moved), rarely outweighs the benefits of having a timestamp on all elements for the applications that we envision. Fast access to useful data is the most important for us, and in real-time applications, especially when user interactivity is in play, score access operations tend to occur thousands of times more often than score manipulation operations. If large-scale content manipulation of scores needs to be done very often, tools like XQuery are more fit for the job [17, 18].

3.2 Some notes on tempo and repetitions

In order to be able to accomodate easily for tempo changes or other elements that affect playback (rallentando, accelerando etc.), we tilted this performance timing information out of the score, and transferred it to a separate datastructure. An elegant solution is the use of a warping function, mapping playback time (in seconds) to score time (in quarter notes). The first derivative of this function is the equivalent of the local tempo at a certain time. Smooth increases or decreases in tempo can be modeled using splines. Cubic Hermite splines are a good choice since those can be calculated based only on two points and the tempo at these points. When constraints are applied to keep the function strict monotonously increasing, an inverse of that function exists which could eventually be used to encode information about performance, like lyricism.

Also affecting playback are structural elements, as there are repeat, coda, segno, ... Since these are usually limited in number and smooth transitions are not applicable here, this data can be stored easiest in a simple table, storing the timestamp values of all sections. When repeats should be skipped, one can just adapt this table, eliminating the need to do processing on the entire score data structure.

Going from playback time to score time (illustrated in fig. 1) then comes down to deciding with what time in quarter notes this corresponds, through the previously defined warping function. If sections are repeated or skipped, offsets to this time need to be added or subtracted, according to the information in the structure table. That way, we come to a corresponding timestamp value in the score structure itself, which can be used to access the necessary data.

Note that in this way, changes in tempo or performance information do not require updating the calculated timestamps in the score, which would be a rather costly operation as mentioned previously. As long as the score data itself remains unchanged, data related to performance and overall structure (repeats etc.) are kept outside the score itself and are quickly and easily accessible and modifiable. This is useful in applications needing some form of audio-to-score alignment [19].

4. STRUCTURE AND DESIGN

4.1 Indexing collections

A score contains a number of parts, a part contains a number of measures, a measure contains a number of notes: it is clear that collections are an important feature in scores. These collections often need to be accessible in multiple ways. For fast direct access, a vector-like construction is ideal: accessing an element is then done in constant time. But a standard XML parser will store elements in a tree structure. In this tree, notes can be interleaved with other data (like harmony indications, certain dynamics elements), and are not necessarily ordered on timestamp.

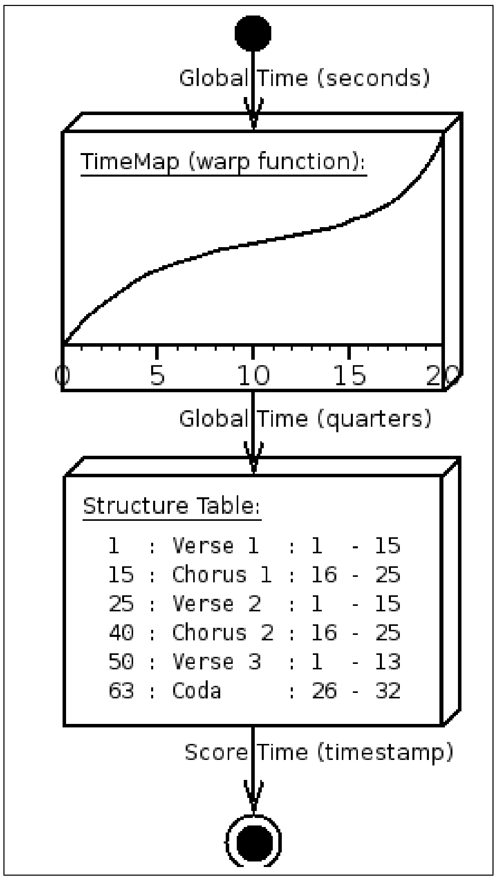


Figure 1. Flowchart: from real time to score timestamps

To gain fast access to the most important data, we implement indices in the data structure, sorted on the criteria we wish to use for retrieval. We obtain fast access to notes by, during parsing, storing them in a map keyed on the timestamp that it has been given. When we need to search for a certain note occurring at a certain time, we can easily retrieve it (a map generally uses a binary search), and iterating over all notes in order can still be done in constant time using the best fit iterators.

Using sorted indexes to access certain structures introduces:

- searching for a certain element can be done in O(log(n)) instead of O(n).
- accessing a specific element takes O(log(n)) for maps and O(1) for arrays.
- when a change occurs, the indexes might need to be updated. The cost depends on the operation being

performed and the data structure used for the index. Insertion or deletion on a map takes only O(log(n)), but if resorting an array is needed, the penalty is O(n log(n)).

Implementing collections with these properties can be done by designing proper templates for them - [20] is an excellent resource on this. Templates are specifically meant to define operations and algorithms independent of type, and can thus be used for any type of element, while still making specializations based on type possible. Operator overloading is a useful programming trick to add additional accessing functionality.

4.2 Cursors and Listeners

For browsing through a score, an iterator system is most elegant. Most programmers are very familiar with this kind of interface. Preferably, a score iterator for real-time use corresponds to a position marker on a sequencer track, we'll therefore call them cursors. The cursor needs access to the tempo and structure information from the score. Multiple cursors on a single score are an asset, but to avoid discrepancies when multiple cursors are used, only one structure table and tempo function should exist for each score. The cursor's internal logic is then responsible for translating the time information from the sequencer to the relevant position in the score.

To keep track of the position in the score, a cursor keeps track of:

- its current position in quarter notes (timestamp)
- for each part, the current measure (this allows for multimetric music)
- for that measure, the next note that needs to be played.

In real-time software, a cursor is moved forward or backward by very small increments. Using the adapted internal score structure, moving the cursor forward comes down to:

- calculate the target timestamp of the cursor.
- for each part in the score, repeat the following until the next note's timestamp is scheduled after the target timestamp:
  - if the next note's timestamp is before the target timestamp, go to the next note
  - if there are no more notes in the current measure, go to the next measure

This could be even more simplified improved upon by collecting all notes of a score together and abstracting away the different measures and parts. But in practice, we found that we often needed to know at a certain moment which measure that a currently played note was in, and to what part it belonged. We consider it easier to keep track of this information in the cursor and request it from there, then to ignore it there but later have to obtain it through the score anyway. The number of measure crossings and the number of parts is also usually limited compared to the number of notes.

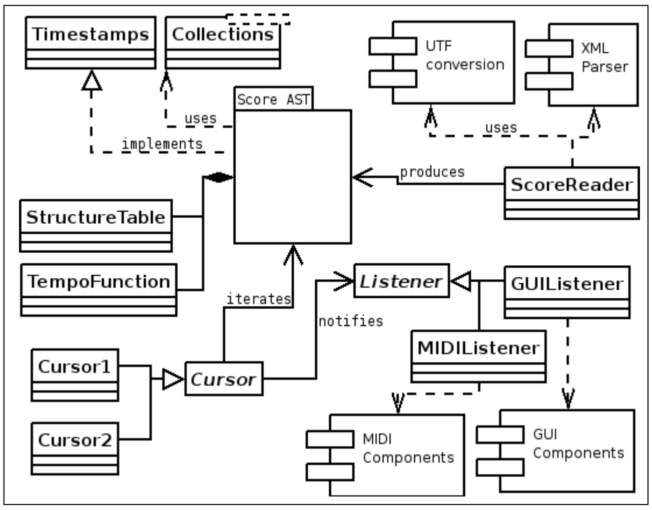


Figure 2. Simplified component diagram of the overall design

As mentioned earlier, intermediate levels in the score hierarchy, like voices or sections, can exist. Each supplementary level may add a nested loop to the aforementioned cursor moving algorithm, so it is beneficial to keep the number of different levels low. The trade-off between removing intermediate levels, keeping the original structure and accessing its information, and performance benefits or losses in different scenario's is difficult to make, and in the end the performance is highly dependent on the scores used: if a large amount of nested loops is kept, a single voice melody score will likely still be iterated over very fast, while traversing an orchestral score will go much slower. On the other hand, by eliminating too many loops we risk to need to introduce a large amount of intermediate variables in the cursor to keep track of all the necessary information, and updating and testing against these also takes time.

A cursor interface can be easily combined with the implementation of an observer pattern [9]. That way, when the cursor passes a note, it can notify another part of the software and trigger an event. In the observer pattern, one or more listeners can be attached to a cursor. The cursor then only needs to notify all of its listeners that an event was encountered. This system can be further generalized to enable notifications to be triggered at whatever element that is encountered in a score, and have them pass data for the listeners.

A simplified component diagram of the overall design of the score handling library is shown in fig. 2 .

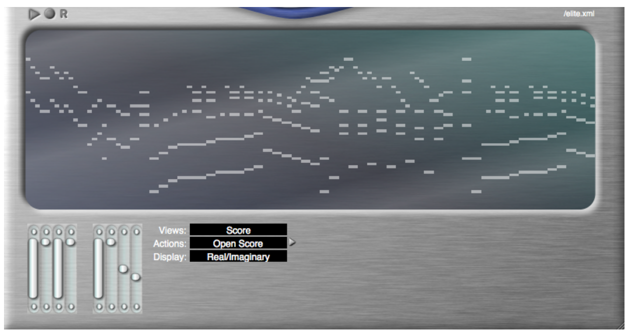
5. IMPLEMENTATION

The library that we developed is part of a larger project, implemented as VST plugin [12]. It is meant to serve as prototyping platform for applications that use both audio and scores in real-time hosts. A screenshot is shown in fig. 3.

As a practical example, if we want simple playback, a cursor is put on the score at the beginning of the score, and incremented in small steps corresponding to the length of the audio data in the processing function of the plugin. When notes are encountered, an event is sent to a specific listener, that will take the note and some other necessary information to generate MIDI events out of it. The start events are sent back to the host, while the stop events are stored in a scheduler to be used when they are necessary. If another event should happen when a note, measure, crescendo, or whatever element in the score is encountered, a developer would only need to write his/her own listener, connect it to the cursor, and configure the cursor in such a way that it reacts to the element needed.

To display the score, we use exactly the same setup, only now the cursor is not moved over a timeline, but over a frame on the screen. A cursor is set on the position corresponding with the left viewport boundary as defined by the GUI's zoom data, scroll data (scrollbars) and window plane. When the screen is redrawn, the cursor is moved to the right viewport boundary. A listener is notified at each note that is encountered, which draws the note onto the window when necessary. There can be several thousands of these events triggered to draw a single score when zoomed out. Nevertheless we experienced that zooming and scrolling go fluently using this design, even if they force several redraws of the screen each second, each time generating a large flow of events.

While fig. 3 only shows a piano roll representation of the notes in the score, the cursor system combined with an implementation of the observer pattern, allows to create other visualizations as well. We might add dynamics information, incorporate or leave out information about the tempo, or leave the notes out and only show rests - the list goes on, and any custom visualization can be created based on the same system that is used for MIDI playback or setting parameters in a plugin. In our work we haven't gone that far though, and we will focus in the near future on the development of real-time plugins using scores for e.g. audio-to-score matching, rather than on visualization.



Views: Score
Actions: Open Score
Display: Real/Imaginary

Figure 3. GUI of a prototype application, showing a basic piano roll representation of Joplin's Elite Syncopations.

6. CONCLUSION

The use of music scores is not yet commonplace in many real-time applications - usually a MIDI representation is used as substitute. In this paper we have presented our efforts to create a library to enable the use of music scores file formats natively in such environments. We have singled out the design desicions that were taken in order to come to a performant library. These considerations are applicable over the boundaries of file formats and computer languages.

In the applications that we envision, the benefits of fast information retrieval from the score and score browsing outweigh the slightly increased complexity on rarely used operations and the precomputation needed. The design considerations presented herein ensure that the computational load during processing is kept to a minimum.

7. ACKNOWLEDGMENTS

This work was funded by a specialization grant from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

8. REFERENCES

[1] MIDI Manufacturers Association, *Complete MIDI 1.0 Detailed Specification, 2nd ed.*, 2001

[2] Recordare LLC: "MusicXML definition, version 2.0," *Available at http://www.recordare.com/xml.html*

[3] M. Good: "Lessons from the Adoption of MusicXML as an Interchange Standard," *Proc. XML 2006 Conference*, 2006.

[4] P. Roland: "The music encoding initiative (MEI)," *Proc. 1st International Conference on Musical Applications Using XML*, pp. 55–59, 2002.

[5] P. Roland and J.S. Downie "Recent Developments in the Music Encoding Initiative Project: Enhancing Digital Musicology and Scholarship," *Proc. 19th Joint International Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing (Digital Humanities 2007)*, pp. 186–189, 2007.

[6] P. Bellini and P. Nesi: "WEDELMUSIC format: an XML music notation format for emerging applications," *Proc. 1st International Conference on Web De-*

livering of Music (WEDELMUSIC 2001), pp. 79–86, 2001.

[7] P. Bellini, P. Nesi and G. Zoia: “Symbolic music representation in MPEG,” *IEEE Multimedia* , Vol. 12, No. 4, pp. 42–49, 2005.

[8] D. Baggi and G. Haus: “The Concept of Interactive Music: the New Standard IEEE P1599 / MX,” *Proc. 2nd International Conference on Semantic and Digital Media Technologies, (SAMT 2007)*, pp. 185–195, 2007

[9] E. Gamma, R. Helm, R. Johnson and J. Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Boston, 1994

[10] GRAME Computer Music Research Lab: “libmusicxml”, *Available at http://libmusicxml.sourceforge.net/*

[11] Unicode Inc.: “Conversions between UTF32, UTF-16, and UTF-8”, *Available at http://www.unicode.org /Public/PROGRAMS/CVTUTF/*

[12] Steinberg Media Technologies GmbH: “Virtual Studio Technology,” *Available at http://www.steinberg.net*

[13] Apple Inc.: “Audio Unit Programming Guide” *Available at http://developer.apple.com /documentation/MusicAudio/Conceptual /AudioUnitProgrammingGuide /AudioUnitProgrammingGuide.pdf*, 2007

[14] Avid Technology, Inc: “Real Time Audio Suite” *Available at http://www.digidesign.com*

[15] L. de Soras: “Denormal numbers in floating point signal processing applications” *Available at http://ldesoras.free.fr/*, 2005

[16] E. Selfridge-Field: *Beyond MIDI: the handbook of musical codes*, MIT Press, Cambridge, MA, 1997

[17] World Wide Web Consortium (W3C): “XQuery 1.0: An XML Query Language - W3C Recommendation 23 January 2007” *Available at http://www.w3.org/TR/xquery/*

[18] J. Ganseman, P. Scheunders and W. D’haes: “Using XQuery on MusicXML Databases for Musicological Analysis” *Proc. 9th International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, PA, USA, 2008.

[19] H. Heijink, P. Desain and L. Windsor: “Make Me a Match: an evaluation of different approaches to Score-Performance Matching,” *Computer Music Journal* , Vol. 24, No. 1, pp. 43–56, 2000.

[20] D. Vandevoorde and N.M. Josuttis: *C++ Templates: the complete guide*, Addison-Wesley, Boston, 2002

GENRE CLASSIFICATION USING HARMONY RULES INDUCED FROM AUTOMATIC CHORD TRANSCRIPTIONS

Amélie Anglade
Queen Mary
University of London
Centre for Digital Music
amelie.anglade@elec.qmul.ac.uk

Rafael Ramirez
Universitat Pompeu Fabra
Music Technology Group
rramirez@iua.upf.edu

Simon Dixon
Queen Mary
University of London
Centre for Digital Music
simon.dixon@elec.qmul.ac.uk

ABSTRACT

We present an automatic genre classification technique making use of frequent chord sequences that can be applied on symbolic as well as audio data. We adopt a first-order logic representation of harmony and musical genres: pieces of music are represented as lists of chords and musical genres are seen as context-free definite clause grammars using subsequences of these chord lists. To induce the context-free definite clause grammars characterising the genres we use a first-order logic decision tree induction algorithm. We report on the adaptation of this classification framework to audio data using an automatic chord transcription algorithm. We also introduce a high-level harmony representation scheme which describes the chords in term of both their degrees and chord categories. When compared to another high-level harmony representation scheme used in a previous study, it obtains better classification accuracies and shorter run times. We test this framework on 856 audio files synthesized from Band in a Box files and covering 3 main genres, and 9 subgenres. We perform 3-way and 2-way classification tasks on these audio files and obtain good classification results: between 67% and 79% accuracy for the 2-way classification tasks and between 58% and 72% accuracy for the 3-way classification tasks.

1. INTRODUCTION

To deal with the ever-increasing amount of digital music data in both personal and commercial musical libraries some automatic classification techniques are generally needed. Although metadata such as ID3 tags are often used to sort such collections, the MIR community has also shown a great interest in incorporating information extracted from the audio signal into the automatic classification process. While low-level representations of harmonic content have been used in several genre classification algorithms (e.g. chroma feature representation in [1]), little attention has been paid to how harmony in its temporal dimension, i.e. chord sequences, can help in this task. However, there seems to be a strong connection between musical genre and the use of different chord progressions [2]. For instance, it is well known that pop-rock tunes mainly follow the classical tonic-subdominant-dominant chord sequence, whereas jazz harmony books propose different series of chord progressions as a standard. We intend to test the extent to which harmonic progressions can be used for genre classification.

In a previous article [3] we have shown that efficient and transparent genre classification models entirely based on a high-level representation of harmony can be built using first-order logic. Music pieces were represented as lists of chords (obtained from symbolic files) and musical genres were seen as context-free definite-clause grammar using subsequences of any length of these chord lists. The grammar representing the genres were built using a first-order logic decision tree induction algorithm. These resulting models not only obtained good classification results when tested on symbolic data (between 72% and 86% accuracy on 2-class problems) but also provided a transparent explanation of the classification to the user. Indeed thanks to the expressiveness of first-order logic the decision trees obtained with this technique can be presented to the user as sets of human readable rules.

In this paper we extend our harmony-based approach to automatic genre classification by introducing a richer harmony representation and present the results of audio data classification. In our previous article we used the intervals between the root notes of consecutive chords. Root interval progressions capture some degree information and do not depend on the tonality. Thus when using root intervals no key extraction is necessary. However, one root interval progression can cover several degree sequences. For instance the degree sequences "IV-I-IV" and "I-V-I" are both represented by the root interval sequence "perfect fifth-perfect fourth". To avoid such generalisations we introduce here another representation of harmony based on the degrees (i.e. I, V, etc.) and chord categories (i.e. min, 7, maj7, etc.). In addition such a representation matches the western representation of harmony and thus our classification models (i.e. decision trees or sets of classification rules describing the harmony) can be more easily interpreted by the users. Finally since degrees are relative to the key, a key estimation step is now needed. This is a requirement but not a limitation as nowadays many chord transcription algorithms from audio (e.g. [4,5]) do also perform key estimation.

The paper is organised as follows: In Section 2 we review some existing studies using high-level representation of harmony for automatic genre classification. In Section 3 we present the details of our methodology, including the knowledge representation and the learning algorithm employed in this study. In Section 4 we present the classification results of our first-order logic classification technique before concluding in Section 5.

2. RELATED WORK

Only a few studies have considered using higher level harmonic structures, such as chord progressions, for automatic genre recognition.

In [6], a rule-based system is used to classify sequences of chords belonging to three categories: Enya, Beatles and Chinese folk songs. A vocabulary of 60 different chords was used, including triads and seventh chords. Classification accuracy ranged from 70% to 84% using two-way classification, and the best results were obtained when trying to distinguish Chinese folk music from the other two styles, which is a reasonable result as both western styles should be closer in terms of harmony.

Paiement et al. [7] also used chord progressions to build probabilistic models. In that work, a set of 52 jazz standards was encoded as sequences of 4-note chords. The authors compared the generalization capabilities of a probabilistic tree model against a Hidden Markov Model (HMM), both capturing stochastic properties of harmony in jazz, and the results suggested that chord structures are a suitable source of information to represent musical genres.

More recently, Lee [8] has proposed genre-specific HMMs that learn chord progression characteristics for each genre. Although the ultimate goal of this work is using the genre models to improve the chord recognition rate, he also presented some results on the genre classification task. For that task a reduced set of chords (major, minor, and diminished) was used.

Finally, Perez-Sancho et al. [9] have investigated if 2, 3 and 4-grams of chords can be used for automatic genre classification on both symbolic and audio data. They report better classification results when using a richer vocabulary (seventh chords) and longer n-grams.

3. METHODOLOGY

Contrary to n-grams that are limited to sequences of length n the first-order logic representation scheme that we adopt can employ chord sequences of variable length to characterise a musical genre. A musical piece is represented as a list of chords. Each musical genre is illustrated by a series of musical pieces. The objective is to find interesting patterns, i.e. chord sequences, that appear in many songs of one genre and do not (frequently) appear in the other genres and use such sets of patterns to classify unknown musical pieces into genres. As there can be several independent patterns and each of them can be of any length we use a context-free definite-clause grammar formalism. Finally to induce such grammars we use TILDE [10], a first-order logic decision tree induction algorithm.

3.1 Knowledge representation

In the definite clause grammar (DCG) formalism a sequence over a finite alphabet of letters is represented as a list of letters. Here the chords (e.g. G7, Db, BM7, F#m7, etc.) are the letters of our alphabet. A DCG is described using predicates. For each predicate `p/2` (or `p/3`) of the form `p(X,Y)` (or `p(c,X,Y)`), `X` is a list representing the sequence to analyse (input) and `Y` is the remaining part of the list `X` when its prefix matching the predicate `p` (or property `c` of the predicate `p`) is removed (output). In the context-free grammar (CFG) formalism, a target concept is defined with a set of rules.

Here our target predicate is `genre/4`, where `genre(g, A,B,Key)` means the song `A` (represented as its full list of chords) in the tonality `Key` belongs to genre `g`. The argument `B`, the output list (i.e. an empty list) is necessary to comply with the definite-clause grammar representation. We are interested in degrees and chord categories to characterise a chord sequence. So the predicates considered to build the rules are `degreeAndCategory/5` and `gap/2`, defined in the background knowledge (cf. Table 1). `degreeAndCategory(d,c,A,B,Key)` means that the first chord of the list `A` has degree `d` and category `c`. The `gap/2` predicate matches any chord sequence of any length, allowing to skip uninteresting subsequences (not characterised by the grammar rules) and to handle large sequences (for which otherwise we would need very large grammars). In addition we constrain the system to use at least two consecutive `degreeAndCategory` predicates between two `gap` predicates. This guarantees that we are considering local chord sequences of a least length 2 (but also larger) in the songs.

```
rootNote(c_,[c|T],T,Key).      rootNote(c_,[cm|T],T,Key).
rootNote(c_s,[cs|T],T,Key).    rootNote(c_s,[csm|T],T,Key).
...                            ...
category(min,[cm|T],T).        category(maj,[c|T],T).
category(min,[csm|T],T).       category(maj,[cs|T],T).
...                            ...
degree(1_,A,B,cmajor) :- rootNote(c_,A,B,cmajor).
degree(1_s,A,B,cmajor) :- rootNote(c_s,A,B,cmajor).
...
degreeAndCategory(Deg,Cat,A,B,Key) :-
degree(Deg,A,B,Key), category(Cat,A,B).

gap(A,A).
gap([_,A],B) :- gap(A,B).
```

Table 1. Background knowledge predicates used in the first-order logic decision tree induction algorithm. For each chord in a chord sequence its root note is identified using the `rootNote/4` predicate. The degrees are defined using the `degree/4` predicate and the key. The chord categories are identified using the `category/3` predicate and finally degrees and categories are united in a single predicate `degreeAndCategory/5`.

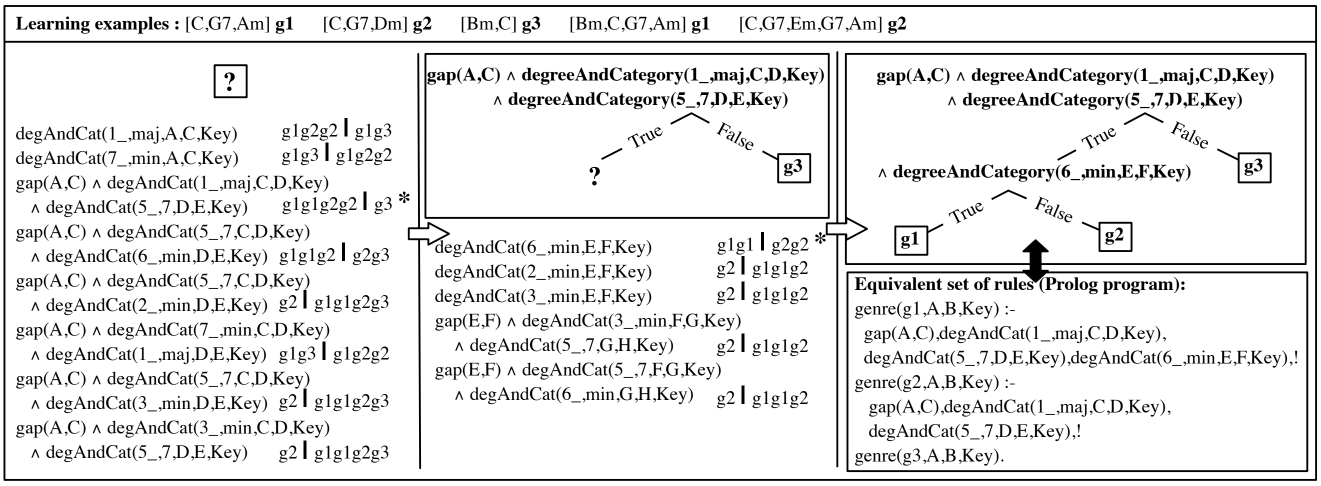

Learning examples : [C,G7,Am] g1 [C,G7,Dm] g2 [Bm,C] g3 [Bm,C,G7,Am] g1 [C,G7,Em,G7,Am] g2
?
degAndCat(1_,maj,A,C,Key) g1g2g2 | g1g3
degAndCat(7_,min,A,C,Key) g1g3 | g1g2g2
gap(A,C) ∧ degAndCat(1_,maj,C,D,Key) ∧ degAndCat(5_,7,D,E,Key) g1g1g2g2 | g3 *
gap(A,C) ∧ degAndCat(5_,7,C,D,Key) ∧ degAndCat(6_,min,D,E,Key) g1g1g2 | g2g3
gap(A,C) ∧ degAndCat(5_,7,C,D,Key) ∧ degAndCat(2_,min,D,E,Key) g2 | g1g1g2g3
gap(A,C) ∧ degAndCat(7_,min,C,D,Key) ∧ degAndCat(1_,maj,D,E,Key) g1g3 | g1g2g2
gap(A,C) ∧ degAndCat(5_,7,C,D,Key) ∧ degAndCat(3_,min,D,E,Key) g2 | g1g1g2g3
gap(A,C) ∧ degAndCat(3_,min,C,D,Key) ∧ degAndCat(5_,7,D,E,Key) g2 | g1g1g2g3
gap(A,C) ∧ degreeAndCategory(1_,maj,C,D,Key) ∧ degreeAndCategory(5_,7,D,E,Key)
True
False
?
g3
degAndCat(6_,min,E,F,Key) g1g1 | g2g2 *
degAndCat(2_,min,E,F,Key) g2 | g1g1g2
degAndCat(3_,min,E,F,Key) g2 | g1g1g2
gap(E,F) ∧ degAndCat(3_,min,F,G,Key) ∧ degAndCat(5_,7,G,H,Key) g2 | g1g1g2
gap(E,F) ∧ degAndCat(5_,7,F,G,Key) ∧ degAndCat(6_,min,G,H,Key) g2 | g1g1g2
gap(A,C) ∧ degreeAndCategory(1_,maj,C,D,Key) ∧ degreeAndCategory(5_,7,D,E,Key)
True
False
∧ degreeAndCategory(6_,min,E,F,Key)
g3
True
False
g1
g2
Equivalent set of rules (Prolog program):
genre(g1,A,B,Key) :-
gap(A,C),degAndCat(1_,maj,C,D,Key),
degAndCat(5_,7,D,E,Key),degAndCat(6_,min,E,F,Key),!
genre(g2,A,B,Key) :-
gap(A,C),degAndCat(1_,maj,C,D,Key),
degAndCat(5_,7,D,E,Key),!
genre(g3,A,B,Key).

Figure 1. Schematic example illustrating the induction of a first-order logic tree for a 3-genre classification problem (based on the 5 learning examples on top). At each step the partial tree (top) and each literal (or conjunction of literals) considered for addition to the tree (bottom) are shown together with the split resulting from the choice of this literal (e.g. g1g1g2|g2 means that two examples of g1 and one of g2 are in the left branch and one example of g2 is in the right branch). The literal resulting in a the best split is indicated with an asterisk. The final tree and the equivalent ordered set of rules (or Prolog program) are shown on the right. The key is C Major for all examples. For space reasons `degAndCat` is used to represent `degreeAngCategory`.

An example of a simple and short grammar rule we can get using this formalism is:
genre(genre1,A,B,Key) :-
gap(A,C),degreeAndCategory(5_,7,C,D,Key),
degreeAndCategory(1_,maj,D,E,Key),gap(E,B).
Which can be translated as : *"Some music pieces of genre1 contain a dominant 7th chord on the dominant followed by a major chord on the tonic"* (i.e. a perfect cadence).
But more complex rules combining several local patterns (of any length larger than or equal to 2) separated by `gaps` can also be constructed with this formalism.

3.2 Learning algorithm

To induce the harmony grammars we apply TILDE's decision tree induction algorithm [10]. TILDE is a first order logic extension of the C4.5 decision tree algorithm [11]. Like C4.5 it is a top-down decision tree induction algorithm: at each step the test resulting in the best split is used to partition the examples. The difference is that at each node of the trees instead of attribute-value pairs, conjunctions of literals are tested. TILDE uses by default the gain-ratio criterion [11] to determine the best split and the post-pruning is the one from C4.5. TILDE builds first-order logic decision trees which can also be represented as ordered sets of rules (or Prolog programs). In the case of classification, the target predicate of each model represents the classification problem. A simple example illustrating the induction of a tree from a set of examples covering three genres is given in Figure 1.

First-order logic enables us to use background knowledge (which is not possible with non relational data mining algorithms). It also provides a more expressive way to represent musical concepts/events/rules which can be transmitted as they are to the users. Thus the classification process can be made transparent to the user.

4. EXPERIMENTS AND RESULTS

4.1 Training data

4.1.1 Audio data

The data used in the experiments reported in this paper has been collected, annotated and kindly provided by the Pattern Recognition and Artificial Intelligence Group of the University of Alicante. It consists in a collection of Band in a Box [1] files (i.e. symbolic files containing chords) from which audio files have been synthesised and it covers three genres: popular, jazz, and academic music. The symbolic files have been converted into a text format in which only the chord changes are available. The Popular music set contains pop, blues, and celtic (mainly Irish jigs and reels) music; jazz consists of a pre-bop class grouping swing, early, and Broadway tunes, bop standards, and bossanovas; and academic music consists of Baroque, Classical and Romantic Period music. All the categories have been defined by music experts who have also collaborated in the task of assigning meta-data tags to the files and rejecting outliers. The total amount of pieces is 856 (Academic 235; Jazz 338; Popular 283) containing a total of 120,510 chords (141 chords per piece in average, a minimum of 3 and a maximum of 522 chords per piece).

The classification tasks that we are interested in are relative to the three main genres of this dataset: academic, jazz and popular music. For all our experiments we consider each time the 3-way classification problem and each of the 2-way classification problems. In addition we also study the 3-way classification problem dealing with the popular music subgenres (blues, celtic and pop music). We do not work on the academic subgenres and jazz subgenres as these two datasets contain very unbalanced subclasses,

[1] http://www.pgmusic.com/products_bb.htm

some of them being represented by only a few examples. Because of this last characteristic removing examples to get the same number of examples per class would lead to poor models built on too few examples. Finally resampling can not be used as TILDE automatically removes identical examples.

For each classification task we perform a 5-fold cross-validation. The minimal coverage of a leaf (a parameter in TILDE) is set to 5.

academic/jazz/popular	Root Int	D&C 3	D&C 7th
Accuracy (baseline = 0.40)	0.619	0.759	0.808
Stderr	0.017	0.015	0.014
# nodes in the tree	40.8	31.0	18.4
# literals in the tree	66.2	90.6	50.8
academic/jazz	Root Int	D&C 3	D&C 7th
Accuracy (baseline = 0.59)	0.861	0.872	0.933
Stderr	0.014	0.014	0.011
# nodes in the tree	11.0	16.4	10.4
# literals in the tree	19.0	46.0	30.8
academic/popular	Root Int	D&C 3	D&C 7th
Accuracy (baseline = 0.54)	0.731	0.824	0.839
Stderr	0.020	0.017	0.016
# nodes in the tree	17.0	12.4	11.0
# literals in the tree	27.6	36.4	31.8
jazz/popular	Root Int	D&C 3	D&C 7th
Accuracy (baseline = 0.55)	0.828	0.811	0.835
Stderr	0.015	0.016	0.015
# nodes in the tree	13.4	17.0	10.6
# literals in the tree	23.2	50.6	29.0
blues/celtic/pop	Root Int	D&C 3	D&C 7th
Accuracy (baseline = 0.36)	0.709	0.703	0.746
Stderr	0.027	0.028	0.026
# nodes in the tree	11.4	16.2	14.0
# literals in the tree	20.4	45.8	40.4

Table 2. Classification results on manual chord transcriptions using a 5-fold cross-validation. The number of nodes and literals present in a tree gives an estimation of its complexity. “Root Int” refers to the root intervals representation scheme. “D&C 3” and “D&C 7th” refers to the degree and chord category representation scheme respectively applied on triads only and on triads and seventh chords.

4.1.2 Chord transcription

The chord transcription algorithm based on harmonic pitch class profiles (HPCP [12]) we apply is described in [13]. It distributes spectral peak contributions to several adjacent HPCP bins and takes peak harmonics into account. In addition to using the local maxima of the spectrum, HPCPs are tuning independent (i.e. the reference frequency can be different from the standard tuning), and consider the presence of harmonic frequencies. In this paper, the resulting HPCP is a 36-bin octave independent histogram representing the relative intensity of each 1/3 of the 12 semitones of the equal tempered scale. We refer to [13] for a detailed description of the algorithm.

The algorithm can be tuned to either extract triads (limited to major and minor chords) or triads and seventh chords (limited to major seventh, minor seventh and dominant seventh). Other chords such as diminished and augmented chords are not included in the transcription (as in many transcription systems) because of the tradeoff between precision and accuracy. After pre-processing, only the chord changes (i.e. when either the root note or the chord category is modified) are kept. Notice that when dealing with the symbolic files (manual transcription) the mapping between the representations is based on the third (major or minor). Since only the chord changes were available in the symbolic files (no timing information) it was not possible to compute the transcription accuracy.

4.2 Validating our new harmony representation scheme

We first study if our new harmony representation scheme based on degrees and chord categories (D&C) can compete with our previous representation scheme based on root intervals (Root Int.). For that we test these two harmony representations on clean data, i.e. on the manual chord transcriptions. We test the degree and chord category representation scheme on both triads-only (D&C 3) and triads and seventh manual transcriptions (D&C 7th). The results (i.e. test results of the 5-fold cross-validation) of these experiments are shown in Table 2.

The D&C representation scheme obtains better results, with accuracies always as high as or higher than the root interval representation scheme classification accuracies. Furthermore the complexity of the models is not increased when using the D&C representation compared to the root interval representation. Indeed, the number of nodes and literals in the built models (trees) are comparable. Using the seventh chord categories leads to much higher accuracies, lower standard errors and lower complexity than when only using the triads.

We also tested these representation schemes when the learning examples are audio files (cf. Section 4.3 for more details on these experiments). However the root interval experiments on audio data were so slow that we were unable to complete a 5-fold cross-validation. We estimate the time needed to build one (2-class) model based on the root interval audio data to 12 hours in average, whereas only 10 to 30 minutes are needed to build a D&C 3 (2-class) model on audio data and around 1 hour and a half for a D&C 7th (2-class) model. In conclusion the degree and category representation scheme outperforms the root interval representation scheme on both classification accuracy and run times.

4.3 Performances on audio data

We now test if our first-order logic classification framework can build good classification models when the learning examples are automatic chord transcriptions from audio files (i.e. noisy data). This is essential for the many applications in which no symbolic representation of the harmony is available. The results of this framework when using the degree and chord category representation scheme on audio data are shown in Table 3.

academic/jazz/popular	D&C 3	D&C 7th
Accuracy (baseline = 0.39)	0.582	0.575
Stderr	0.017	0.017
# nodes in the tree	59.2	66.8
# literals in the tree	171.2	198.4
academic/jazz	D&C 3	D&C 7th
Accuracy (baseline = 0.59)	0.759	0.743
Stderr	0.018	0.018
# nodes in the tree	26.4	31.8
# literals in the tree	76.0	93.8
academic/popular	D&C 3	D&C 7th
Accuracy (baseline = 0.55)	0.685	0.674
Stderr	0.020	0.021
# nodes in the tree	25.8	26.4
# literals in the tree	72.2	74.0
jazz/popular	D&C 3	D&C 7th
Accuracy (baseline = 0.54)	0.789	0.773
Stderr	0.016	0.017
# nodes in the tree	22.4	28.8
# literals in the tree	66.0	86.0
blues/celtic/pop	D&C 3	D&C 7th
Accuracy (baseline = 0.35)	0.724	0.668
Stderr	0.027	0.028
# nodes in the tree	13.2	14.8
# literals in the tree	38.8	43.2

Table 3. Classification results on audio data using a 5-fold cross-validation.

Although the accuracies are still good (significantly above the baseline), it is not surprising that they are lower than the results obtained for clean data (i.e. manual transcriptions). The noise introduced by the automatic chord transcription also leads to a higher complexity of the models derived from audio data. Also using the seventh chords leads to slightly less accurate models than when using triads only. The opposite result was obtained with the manual transcription data, where the seventh chord representation scheme outperformed the triads representation scheme. We surmise that the reason for this difference is the fact that the automatic chord transcription algorithm we use is much less accurate when asked to use seventh chords than when asked to use triads only.

Concerning the classification tasks, all the 2 and 3-class problems are solved with accuracies well above chance level. The 3-class popular music subgenres classification problem seems particularly well handled by our framework with 72% and 67% accuracy when using respectively triads and seventh chords. The best 2-class classification results (between 74% and 79% accuracy) are obtained when trying to distinguish jazz from another genre (academic or popular). Indeed the harmony of classical and popular music can be very similar, whereas jazz music is known for its characteristic chord sequences, very different from other genres harmonic progressions.

4.4 Transparent classification models

To illustrate the transparency of the classification models built using our framework we present here some interesting rules with high coverage extracted from classification models generated from symbolic data. Notice that the classification models are trees (or ordered sets of rules), so a rule in itself can not perform classification both because of having a lower accuracy than the full model and because the ordering of rules in the model is important to the classification (i.e. some rule might never be used on some example because one of the preceding rules in the model covers this example). To illustrate this for each of the following example rules we provide its absolute coverage (i.e. if the order was not taken into account) on each genre.

The following rule was found in the popular subgenres classification models:
[*coverage: blues=42/84; celtic=0/99; pop=2/100*]
genre(blues,A,B,Key) :-
gap(A,C),degreeAndCategory(1_,7,C,D,Key),
degreeAndCategory(4_,7,D,E,Key),gap(E,B).
"Some blues music pieces contain a dominant seventh chord on the tonic directly followed by a dominant seventh chord on the subdominant (IV)."

The following rules were found in the academic/jazz/popular classification models:
[*cov.: jazz=273/338; academic=42/235; popular=52/283*]
genre(jazz,A,B,Key) :-
gap(A,C),degreeAndCategory(2_,min7,C,D,Key),
degreeAndCategory(5_,7,D,E,Key),gap(E,B).
"Some jazz music pieces contain a minor seventh chord on the supertonic (II) directly followed by a dominant seventh chord on the dominant."
[*cov.: jazz=173/338; academic=1/235; popular=17/283*]
genre(jazz,A,B,Key) :-
gap(A,C),degreeAndCategory(6_,7,C,D,Key),
degreeAndCategory(2_,min7,D,E,Key),gap(E,B)
"Some jazz music pieces contain a dominant seventh chord on the submediant (VI) directly followed by a minor seventh chord on the supertonic (II)."

Finally the following rules were found in the academic/jazz classification models:
[*cov.: academic=124/235; jazz=6/338; popular=78/283*]
genre(academic,A,B,Key) :-
gap(A,C),degreeAndCategory(1_,maj,C,D,Key),
degreeAndCategory(5_,maj,D,E,Key),gap(E,B).
"Some academic music pieces contain a major chord on the tonic directly followed by a major chord on the dominant."
[*cov.: academic=133/235; jazz=10/338; popular=68/283*]
genre(academic,A,B,Key) :-
gap(A,C),degreeAndCategory(5_,maj,C,D,Key),
degreeAndCategory(1_,maj,D,E,Key),gap(E,B).
"Some academic music pieces contain a major chord on the dominant directly followed by a major chord on the tonic."
Note that the lack of sevenths distinguishes this last common chord change from its jazz counterparts. Indeed the following rule has a high coverage on jazz:
[*cov.: jazz=146/338; academic=0/235; popular=15/283*]
genre(jazz,A,B,Key) :-
gap(A,C),degreeAndCategory(5_,7,C,D,Key),
degreeAndCategory(1_,maj7,D,E,Key),gap(E,B).

5. CONCLUSION AND FUTURE WORK

In this paper we showed that our genre classification framework based on harmony and first-order logic and previously tested on symbolic data in [3] can also directly learn classification models from audio data that obtain a classification accuracy well above chance level. The use of a chord transcription algorithm allows us to adopt a high-level representation of harmony even when working on audio data. In turn this high-level representation of harmony based on first-order logic allows for human-readable, i.e. transparent, classification models. We increased this transparency by introducing a new harmony representation scheme, based on the western representation of harmony which describes the chords in terms of degrees and chord categories. This representation is not only musically more meaningful than a previous representation we adopted, it also got better classification results and the classification models using it were built faster. Testing our model on manual transcriptions we observed that using seventh chords in the transcription task could considerably increase the classification accuracy. However the automatic transcription algorithm we used for these experiments was not enough accurate when using seventh chords and we could not observe such improvements when using audio data.

Future work includes testing several other chord transcription algorithms to see if they would lead to better classification models when using seventh chords. We also plan to use these chord transcription algorithms to study how the accuracy of classification models built on transcriptions evolves with the accuracy of these transcriptions. In addition the audio data used in these experiments was generated with MIDI synthesis. This is generally cleaner than CD recordings, so we expect a further degradation in results if we were to use audio recordings. Unfortunately we do not possess the corresponding audio tracks that would allow us to make this comparison. We intend to look for such recordings and extend our audio tests to audio files that are not generated from MIDI. Finally with these experiments we showed that a classification system based only on chord progressions can obtain classification results well above chance level. If such a model based only on one dimension of music (harmony) can not compete on its own with state-of-the-art classification models, we believe – and intend to test this hypothesis in future experiments – that if such an approach is combined with classification models based on other dimensions (assumed orthogonal) such as rhythm and timbre we will improve on state-of-the-art classification accuracy.

6. ACKNOWLEDGMENTS

We would like to thank the Pattern Recognition and Artificial Intelligence Group of the University of Alicante for providing the data. This work is supported by the EPSRC project OMRAS2 (EP/ E017614/1) and the Spanish TIN project ProSeMus (TIN2006-14932-C02-01). During her internship at the Music Technology Group the first author was supported by the EPSRC Platform grant EP/E045235/1.

7. REFERENCES

[1] G. Tzanetakis, A. Ermolinskiy, and P. Cook. Pitch histograms in audio and symbolic music information retrieval. In *Proceedings of ISMIR 2002*, Paris, France, 2002.

[2] W. Piston. *Harmony*. Norton, W. W. & Company, Inc., 5th edition, 1987.

[3] Amélie Anglade, Rafael Ramirez, and Simon Dixon. First-order logic classification models of musical genres based on harmony. In *Proceedings of the 6th Sound and Music Computing Conference*, Porto, Portugal, 2009.

[4] T. Yoshioka, T. Kitahara, K. Komatani, T. Ogata, and H. G. Okuno. Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *Proceedings of ISMIR 2004*, pages 100–105, Barcelona, Spain, 2004.

[5] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):291–301, 2008.

[6] M.-K. Shan, F.-F. Kuo, and M.-F. Chen. Music style mining and classification by melody. In *Proceedings of 2002 IEEE International Conference on Multimedia and Expo*, volume 1, pages 97–100, 2002.

[7] J.-F. Paiement, D. Eck, and S. Bengio. A probabilistic model for chord progressions. In *Proceedings of ISMIR 2005*, pages 312–319, London, UK, 2005.

[8] K. Lee. A system for automatic chord transcription using genre-specific hidden markov models. In *Proceedings of the International Workshop on Adaptive Multimedia Retrieval*, Paris, France, 2007.

[9] C. Perez-Sancho, D. Rizo, S. Kersten, and R. Ramirez. Genre classification of music by tonal harmony. In *International Workshop on Machine Learning and Music*, Helsinki, Finland, 2008.

[10] Hendrik Blockeel and Luc De Readt. Top down induction of logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[11] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[12] T. Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *Proceedings of the 1999 International Computer Music Conference*, pages 464–467, Beijing, China, 1999.

[13] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, MTG, Universitat Pompeu Fabra, 2006.

SONGEXPLORER: A TABLETOP APPLICATION FOR EXPLORING LARGE COLLECTIONS OF SONGS

Carles F. Julià, Sergi Jordà
Music Technology Group
Universitat Pompeu Fabra, Barcelona, Spain
{carles.fernandez, sergi.jorda}@upf.edu

ABSTRACT

This paper presents SongExplorer, a system for the exploration of large music collections on tabletop interfaces. SongExplorer addresses the problem of finding new interesting songs on large music databases, from an interaction design perspective. Using high level descriptors of musical songs, SongExplorer creates a coherent 2D map based on similarity, in which neighboring songs tend to be more similar. All songs are represented as throbbing circles that highlight their more relevant high-level properties, and the resulting music map is browseable and zoomable by the users who can use their fingers as well as specially designed tangible pucks, for helping them to find interesting music, independently of their previous knowledge of the collection. SongExplorer also offers basic player capabilities, allowing the users to organize the songs they have just discovered into playlists which can be manipulated as well as played and displayed. In this paper, the system hardware, software and interaction design are explained, and the usability tests carried are presented. Finally, conclusions and future work are discussed.

1. INTRODUCTION

Since the popularization of the Internet and broadband connections, the amount of music which we are exposed to, has been increasing permanently. Nowadays, many websites do offer very large collections of music to the user, either free of charge (e.g. Magnatune [1], Jamendo [2]) or on a fee-paying basis (e.g. iTunes [3], The Orchard [4]). Such a number of available and still undiscovered music records and songs seems too difficult to manage in a sorting and searching-by-keyword way. In order to solve this problem and help users to discover new music, many online music recommendation services have been created (e.g. Pandora [5], Last.fm [6]). One of the main drawbacks of most current music recommenders, independently of the recommendation mechanisms and algorithms they employ (user profiling, experts-based knowledge, statistical models, etc.), is that they apply information filtering techniques to the entire collections, in order to extract and display only a subset of songs that the system believes the user could enjoy. By doing it this way, the user loses the opportunity to discover many new songs which are not presented by the system, whatever the cause may be.

[1] http://www.margatune.com
[2] http://www.jamendo.com
[3] http://www.apple.com/itunes/
[4] http://www.theorchard.com

To solve this problem, we propose to construct maps of the entire collections of songs and allow users to explore them in novel ways. Maps are widely used to explore spaces and also concepts. Although most commonly used to depict geography, maps may represent any space, real or imagined, without regard to context or scale. We use conceptual maps to discuss ideas, we organize data in 2D spaces in order to understand it, and we can get our bearings using topographical maps. SongExplorer's maps are constructed using MIR techniques that provide the high-level descriptors needed successfully organizing the data; they do not filter or hide any content, thus showing the complete collection while highlighting some of the songs' characteristics.

Therefore, SongExplorer provides intuitive and fast ways for promoting the direct exploration of these maps. In the last years, several successful projects have shown that tangible, tabletop and multitouch interfaces exhibit useful properties for advanced control in general (such as continuous, real-time interaction with multidimensional data, and support for complex, skilled, expressive and explorative interaction) [4] and for the exploration of bidimensional spaces in particular [2]. Following this trend, SongExplorer allow users to interact with the maps directly with their hands, touching the surface with their fingers and manipulating physical tokens on top of it. In the following section we will comment some of the most relevant previous works, related to the two main aspects of our project, i.e. (i) the visualization of musical data, and (ii) the direct manipulation of this or any other type of data, in a tabletop interaction context.

[5] http://www.pandora.com
[6] http://www.last.fm

2. RELATED WORK

2.1 Visualization of music collections

In the field of visualization, there is an extensive bibliography on the representation of auditory data. In the particular case we are focusing on, that of the visual organization of musical data, solutions often consist in extracting feature descriptors from data files, and creating a multidimensional feature space that will be projected into a 2D surface, using dimensionality reduction techniques.

A very well known example of this method is the work Islands of Music by Pampalk [13], which uses a landscape metaphor to present a large collection of musical files. In this work, Pampalk uses a Self Organizing Map (SOM) [9] to create a relief map in which the accumulation of songs are presented as the elevation of the terrain over the sea. The islands created as a result of this process roughly correspond to musical genres.

A later attempt to combine different visualizations on a single map was also created by Pampalk et al [14]. By using different parameters to organize the SOM, they created several views of the collection, later interpolating the different solutions for creating a smooth combination of situations with which to explore new information.

Beyond the 2D views, an interesting work on music collections visualization, which distributes the songs on a spherical surface, thus avoiding any edge or discontinuity, is described by Leitich and Topf [11].

In the aforementioned examples, a topological metaphor is taken in advantage to enable users exploring big collections of data. A different and original visualization approach is chosen in Musicream [1], an interesting example of exploratory search in music databases, using the search by example paradigm. In Musicream, songs are represented using colored circles, which fall down from the top of the screen. When selected, these songs show their title on their center, and they can be later used to ”fish” similar ones.

2.2 Tangible tabletop interaction

In the domain of Tabletop and Tangible User Interfaces (TUI) there is also a growing interest in working with musical applications. As a matter of fact, from the Audiopad [15] to the Reactable [5], music performance and creation has arguably become the most popular and successful application field in the entire lifetime of this interaction paradigm. This is, according to Jordà [4], because of the specific affordances of this type of interfaces: support of collaboration and sharing of control; continuous, real-time interaction with multidimensional data; and support of complex, expressive and explorative interaction. In this sense, and although less prolific than the applications strictly conceived for musical performance, some interesting works have also been developed to interact with large music collections.

Musictable [16] takes a visualization approach similar to the one chosen in Palmpalk's Islands of Music, to create a two dimensional map that, when projected on a table, is used to make collaborative decisions to generate playlists. Another adaptation into the tabletop domain is the work from Hitchner et al [3], which uses a SOM to build the map and also creates a low resolution mosaic that is shown to the user. The users can redistribute the songs on this mosaic and thus changing the whole distribution of SOM according to the user's desires.

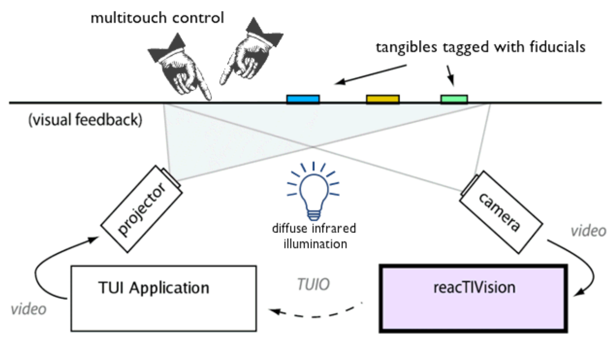


Figure 1. reactiVision framework schema.

We believe this paper also represents a real contribution to the tangible/tabletop user interface community. As noted before, it has been proposed very recently [4] that they can be especially adequate for complex and non-task oriented types of interaction, which could include real-time performance, as well as explorative search. The topic addressed by this paper (N-Dimensional navigation in 2-D) has never been addressed before within tabletop interfaces.

3. HARDWARE

SongExplorer is a tabletop application, i.e. a computer application meant to run on a tangible and multitouch surface, designed for the exploration and discovery of new music. In this section we discuss its main hardware components.

As schematized in Fig.1, the system is made of a translucent plastic surface, some infrared lamps for diffused illumination, an infrared camera for the detection of the user interaction, and a projector for the projection of the visual feedback on the table surface. The surface is round, as in the Reactable case, for encouraging collaboration [5].

The tracking software is based on reacTIVision [6], an open-source framework for the recognition of fingers and objects tagged with fiducials. The images showing the fiducial markers that are stuck into the physical pucks, and the fingers that are in contact with the translucent surface, are captured by the infrared camera and processed by reacTIVision. For each video frame, this software component sends the corresponding data (which includes the positions and IDs of the identified objects and fingers) to SongExplorer, using the TUIO protocol [7]. SongExplorer subsequently identifies the gestures and the actions performed on the table surface, and proceeds with the appropriate responses, finally generating the output image that is displayed by the projector on the translucent surface.

4. SOFTWARE

This section describes the main components of the Song-Explorer software: feature extraction, visualization and interaction.

4.1 Feature Extraction

SongExplorer uses all the songs included in the Magnatune online database, which comprises a total of 6666 songs weighting more than 26 GB. Being Creative Commons-licensed, this library is used in many research projects. These songs are processed by an in-house music annotation library developed at the Music Technology Group (MTG) [10], and the results are transformed to binary files that can be loaded by the system using the Boost [7] C++ library.

4.2 Visualization

From the whole set of available annotated features generated by the annotation library, we are curently using the most high-level properties together with the BPM:

- Beats Per Minute (BPM)
- Happy probability
- Sad probability
- Party probability
- Acoustic probability
- Aggressive probability
- Relaxed probability

All these high level features are independent, and even the moods, which try to cover all the basic emotions, do not depend on each other (i.e. a song could be both sad and happy) [10]. The emotional features can, in fact, be considered binary, with their values indicating the probability of this feature being true.

With this data, a multidimensional feature space (of 7 dimensions) is constructed, in which each song is a single data point with its position defined by these 7 features, all of them being normalized between 0 and 1. From this multidimensional data we construct a 2D space which preserves its topology, and we present it to the user, who will then be able to explore it.

Similarly to other visualization works, a SOM is used to distribute the data on the 2D space. Our implementation of the Kohonen network uses a circular, hexagonally-connected neuron grid, in order to fit the shape of the interactive surface. As opposed to the original implementation of SOM [9], a restriction was added to prevent more than one song falling into a single neuron, so that every representation in the 2D space should be visible and equally distant from its direct neighbors, as shown in Fig. 2.

In the visualization plane, every song is represented by a colored circle, throbbing at the song's BPM. Since there

[7] http://www.boost.org

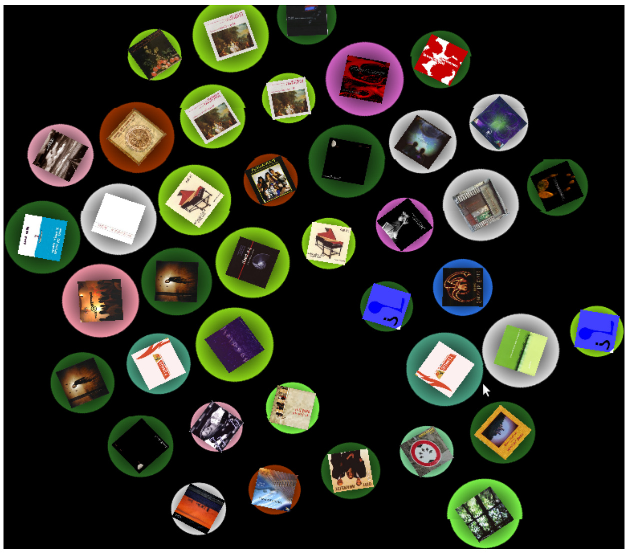


Figure 2. Detail of the hexagonal structure of the grid.

seems to be a strong agreement about the usefulness of artwork to recognize albums or songs [11, 12], depending on the zoom factor, the actual artwork may be shown in the center of each song.

Additionally, colors are used to highlight the different properties of the songs. The coupling {feature $\rightarrow$ color} was defined with an online survey where 25 people had to pair the high-level tags to colors. The color candidates were 7 basic colors with maximum saturation and lightness: red, blue, green, brown, cyan, yellow and magenta. Subjects were only able to choose the best color representation for each tag. The results were: aggressive-red (with an agreement of 100%), relaxed-cyan (43.5%), acoustic-brown (52%), happy-yellow (39%), party-magenta (48%) and sad-blue (56.5%).

For every song, its corresponding property value is mapped into the saturation of the related color (0 meaning no saturation thus resulting on a grey color, 1 corresponding to full saturation), while the lightness is kept to the maximum and the hue is obviously linked to the emotional feature se-

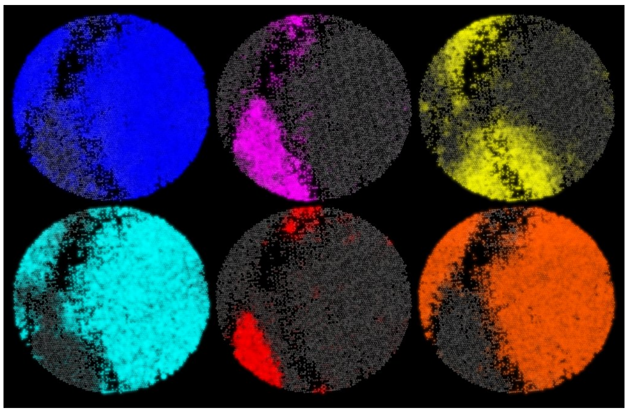


Figure 3. Colors highlighting high-level properties: sad, party, happy, relaxed, aggressive and acoustic (Best seen in color, colors modified for B/W printing).

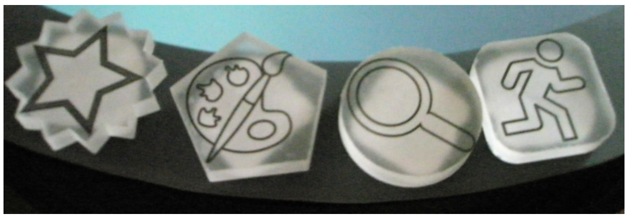


Figure 4. The tangibles of SongExplorer: *playlist navigator*, *color changer*, *magnifying glass* and *navigation menu*

lected, as described in the previous color pairings (Fig. 3 shows the effect of different highlights on the songs). An option to see colors representing genres is also provided, although in that case the pairing between genres and colors is done randomly.

4.3 Interaction

From a users perspective, SongExplorer is a table that shows dynamic images on its surface, which can be manipulated in several ways, using both the fingers as well as some special pucks we will call tangibles, and which will be described later.

4.3.1 Multitouch interaction

Basic finger interaction includes single and multiple finger gestures, and the use of one or two hands. The simplest gesture, selecting and clicking, is implemented by touching a virtual object shown on the table surface, with a single finger and for more than 0.4 seconds. In order to distinguish them from the selection action, other finger gestures involve the use of two simultaneous fingers for each hand. That way, using only one hand, users can translate the map and navigate through it, while the use of both hands allows rotating and zooming the map (see Fig. 5). It should be noted that most of these gestures have become de-facto standards in multitouch and tabletop interaction [8].

4.3.2 Tangible interaction with pucks

Additionally, SongExplorer tangibles also include 4 transparent Plexiglas objects of about 50cm2 each, each one with a different shape and a different icon that suggests its functionality, as described in Table 1. These pucks, which can be kept on the table frame outside the interactive zone (see Fig. 4), become active and illuminated when they get in contact with the interactive surface. As indicated below, some (like the color changer or the navigator) will apply to the whole map, while others (such as the magnifying glass) apply to the selected song.

- The **color changer** puck allows selecting and highlighting one of the different emotional properties of the whole song space. For example, changing the map to red allows us to see the whole map according to its aggressive property, with fully red dots or circles corresponding to the more aggressive songs, and grey dots to the least aggressive ones. Apart from helping to find songs based on a given property, the resulting visual cues also help to memorize and recognize the explored zones of the map.
- When placed on top o a song, the **magnifying glass** puck allows seeing textual information on this particular song, such as the song title, the album, the authors name, as well as the artwork.
- The **navigation** puck displays a navigation menu, which allows the user to perform actions related to the movement and zooming of the map, such as returning to the initial view, or zooming and centering on the currently playing song.
- The **playlist navigator** puck allows the creation and management of the playlist, as described below.

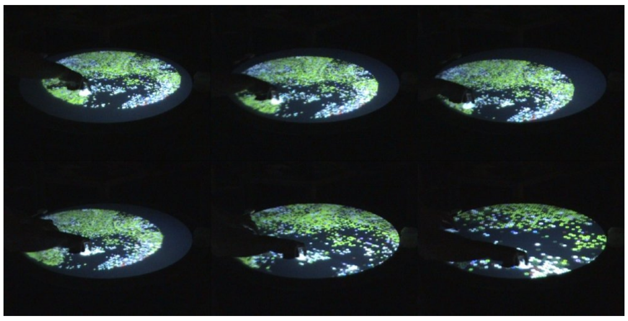


Figure 5. Virtual Map movement (up) and zooming (down)

4.3.3 Managing playlist and playing back songs

SongExplorer has the ability of creating and managing playlists. Playlist are graphically represented on the surface as a constellation, in which the stars (i.e. the corresponding songs it contains) are connected by lines establishing their playing order (see Fig. 6). Most stars show a white stroke, except for the one that is currently playing (red), and the one the playlist navigator is focusing on (green).

Playlists allow several actions using both the fingers and the playlist navigator puck. When clicking on a song, this is automatically added to the playlist. Users can start playing a song by clicking on any star of the playlist. Similarly, crossing out a star removes the corresponding song from the list. A song will stop playing either when it reaches its end, when the song is deleted from the playlist or when another song is selected for playing, and a playlist will keep playing until its end, unless it is stopped with the playlist navigator puck. This object allows several additional actions to be taken on the playlist, such as navigating through its songs and showing information about the focused song in the same way the magnifying glass does.

5. EVALUATION

Some user tests have been undertaken in order to evaluate the system, focusing on the interface design. The evaluation consisted in three areas: subjective experience, adequacy of the visualization and the organization, and interaction.

Symbol	Name	Description
☆	playlist navigator	Permits to run over the songs on the playlist
🎨	color changer	Allows to highlight features of the songs
🔍	magnifying glass	Shows information about songs
🏃	navigation menu	Provides a way to return to known situations

Table 1. Tangibles used in SongExplorer

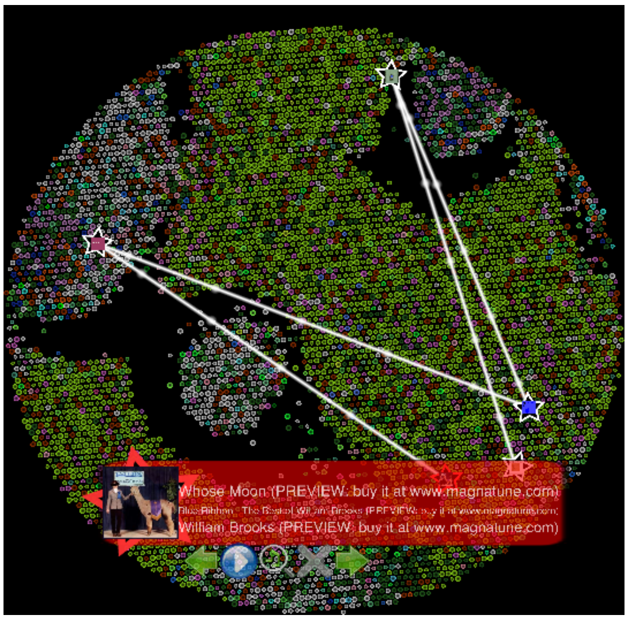

Whose Moon (PREVIEW: buy it at www.magnatune.com)
William Brooks (PREVIEW: buy it at www.magnatune.com)

Figure 6. Playlist and Playlist navigator

5.1 Experiment Procedure

To carry out the tests, an interactive table with SongExplorer up and running was provided. The system was always on an initial state at the beginning. One subject at a time was doing the test. First of all, a little explanation about the purpose, visualization and interaction was given. Then the subject was asked to *find something interesting* in the music collection. No time limit was imposed, and the subject was observed along the process. At the end of the activity, the subject was told to fill a questionnaire, on which she had to rate, using a Likert scale of 10 levels [8], the several aspects of each area. They could also write suggestions at the end of the test.

5.2 Results

After doing the tests the results were quite positive (see Table 2). Regarding the personal experience with SongExplorer, the subjects enjoyed the experience, discovered new and interesting music, felt comfortable, and found it useful

[8] 10: Totally agree, 0: Totally disagree.

	$\mu_{1/2}$	IQR
Enjoyed the experience	8	1
Discovered new music	8	1
Felt comfortable	8	1.5
Found it useful	9	0.5
Found colors correct	8	1.5
Found categories suitable	7	1
Found graphics adequate	9	1.5

Table 2. Evaluation Results. $\mu_{1/2}$: Median, IQR: Interquartile range.

to find interesting music. So the overall experience seemed to be good; we have to notice the low deviation, indicating that there was an agreement about these opinions.

Focusing on the visualization process, there was also a common opinion about the suitability of the colors used. This is not a surprise, as they were extracted from an online poll (details on subsection 4.2). The categories (formerly the high-level properties from the annotation software) were suitable, according to the subjects, for the purpose of describing music. The graphics were also evaluated (meaning the adequacy of icons, the metaphor song-circle, the panels...) and also appreciated.

For the evaluation of the interaction, this paper will not enter into details, because of its extension, but the results were also quite positive. The level of understanding of every gesture and tangible of SongExplorer was tested, as well as their difficulty of use and usefulness. The only noticeable result was that there seemed to be an inverse correlation between previous experiences with tabletops and the perceived difficulty of finger gestures.

Finally there was a general demand for more music-player capabilities like pause or a progress bar for jumping to the middle of the song. The option of bookmarking and storing playlist was also desired.

6. CONCLUSIONS AND FURTHER WORK

We have presented SongExplorer, a new system for large music collections exploration, based on similarity and high level property highlighting that can allow users to find interesting new music.

The user tests have shown that this system can be a good tool for discovering new, valuable music to the users. And this forces us to think about its possible real world applications. As long as this type of interfaces are uncommon, it is not intended for personal use because of its physical nature (size) and its hardware requirements. But other uses than the personal one can be imagined. For instance, some researchers from the annotation software communicated their desire to use SongExplorer to test the reliability of its annotation systems. Using the virtual map they can easily search for inconsistencies. This can be extended to other annotation software systems.

As another real world user case, it would be useful, as a way of promoting music in stores, to have this system available to their customers. An additional feature could be created allowing users to highlight their favorite music so they can then find similar music near to the ones they like, to optionally buy the records afterwards.

In the future versions of SongExplorer, we want to give it the ability of storing playlists, give the user the option of rating songs, adding common player-like capabilities like jumping to the middle of a song, searching songs using actual records (using identifiers on the CD cases) and probably more features.

Video:

`http://www.vimeo.com/4796964`

7. ACKNOWLEDGEMENTS

We thank the MTG team at Universitat Pompeu Fabra for its excellent work on music mood classification that made this project possible. Also to the Reactable project for providing access to its hardware and software platform that is the base of SongExplorer. Finally, we want to thank the open source community for providing the software libraries and free content (Magnatune) on witch this application relies.

8. REFERENCES

[1] M. Goto and T. Goto. Musicream: New music playback interface for streaming, sticking, sorting, and recalling musical pieces. In *ISMIR 2005: Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.

[2] J.Y. Han. Multi-touch interaction wall. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 2006.

[3] S. Hitchner, J. Murdoch, and Tzanetakis G. Music browsing using a tabletop display.

[4] S. Jordà. On stage: the reactable and other musical tangibles go real. *International Journal of Arts and Technology*, 1(3):268–287, 2008.

[5] S. Jorda, M. Kaltenbrunner, G. Geiger, and R. Bencina. The reactable*. In *Proceedings of the International Computer Music Conference (ICMC 2005), Barcelona, Spain*, pages 579–582, 2005.

[6] M. Kaltenbrunner and R. Bencina. reacTIVision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 69–74. ACM New York, NY, USA, 2007.

[7] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. TUIO: A protocol for table-top tangible user interfaces. *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

[8] J. Kim, J. Park, H.K. Kim, and C. Lee. Hci (human computer interaction) using multi-touch tabletop display. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2007. PacRim 2007*, pages 391–394, 2007.

[9] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[10] C. Laurier, O. Meyers, J. Serrà, M. Blech, and P. Herrera. Music mood annotator design and integration. Chania, Crete, Greece, 03/06/2009 2009.

[11] S. Leitich and M. Topf. Globe of Music: Music Library Visualization Using GEOSOM. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007).*

[12] A. Pabst and R. Walk. Augmenting a rugged standard DJ turntable with a tangible interface for music browsing and playback manipulation. In *Intelligent Environments, 2007. IE 07. 3rd IET International Conference on*, pages 533–535, 2007.

[13] E. Pampalk. Islands of Music Analysis, Organization, and Visualization of Music Archives. *Journal of the Austrian Soc. for Artificial Intelligence*, 22(4):20–23, 2003.

[14] E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. *Computer Music Journal*, 28(2):49–62, 2004.

[15] J. Patten, B. Recht, and H. Ishii. Audiopad: A tag-based interface for musical performance. In *Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–6. National University of Singapore Singapore, Singapore, 2002.

[16] I. Stavness, J. Gluck, L. Vilhan, and S. Fels. The MUSICtable: A Map-Based Ubiquitous System for Social Interaction with a Digital Music Collection. *Lecture Notes In Computer Science*, 3711:291, 2005.

AN EFFICIENT SIGNAL-MATCHING APPROACH TO MELODY INDEXING AND SEARCH USING CONTINUOUS PITCH CONTOURS AND WAVELETS

Woojay Jeon, Changxue Ma, and Yan Ming Cheng
Applied Research and Technology Center
Motorola, Inc.
Schaumburg, IL, U.S.A.
{woojay, Changxue.Ma, fyc002}@motorola.com

ABSTRACT

We describe a method of indexing and efficiently searching music melodies based on their continuous dominant fundamental frequency (f0) contours without obtaining note-level transcriptions. Each f0 contour is encoded by a redundant set of wavelet coefficients that represent its shape in level-normalized form at various locations and time scales. This allows a query melody to be exhaustively compared with variable-length portions of a target melody at arbitrary locations while accounting for differences in key and tempo. The method is applied in a Query-by-Humming (QBH) system where users may search a database of recorded pop songs by humming or singing an arbitrary part of the melody of an intended song. The system has fast retrieval times because the wavelet coefficients can be effectively indexed in a binary tree and a vector distance measure instead of dynamic programming is used for comparisons. Using automatic pitch extraction to obtain all f0 contours from acoustic data, the method demonstrates practical performance in an experiment with an existing monophonic data set and in a preliminary experiment with real-world polyphonic music.

1. INTRODUCTION

It has been suggested in the past that using "continuous" (or frame-based) pitch contours may result in more robust matches of music melodies [1] compared to using symbolic string representations (usually note transcriptions). Both methods require reliable extraction of the dominant pitch contour from both query and target for matches to be successful, but the latter approach requires an extra transcription stage of converting the continuous contours to symbolic strings, which can exacerbate the effect of pitch tracking errors because it makes hard decisions on note boundaries and quantization levels. However, the former approach also has the major drawback of high computational complexity, especially when applying string matching techniques to handle differences in tempo and key as well as the well-known insertion, deletion, and substitution errors. Piecewise approximations of the contours have been used for greater efficiency [2], but this still requires query and target melodies to have roughly similar tempi.

Another problem in melody search is the length and location of queries within their target songs. Query-by-Humming(QBH) applications often limit queries to specific music phrases or hooks, hence simplifying the search space, but in other melody search scenarios, the query may be a completely random portion of a song, e.g. a briefly audible segment of a tune in a TV commercial that the viewer wishes to identify.

In this study, we present a method that tries to address both issues – the computational complexity when using continuous pitch contours *and* allowing the search of partial melodies at arbitrary locations – by using redundant wavelet transformations to index and match pitch contours. The method avoids edit-distance comparisons and instead uses distance measures between fixed-dimension vectors while explicitly resolving tempo and key differences from the very beginning of the search process. This is done by dividing target melodies into overlapping, level-normalized segments over a range of lengths and using wavelets to efficiently represent the segments and match them with queries. The wavelet coefficients are stored in vectors that are in turn indexed in a binary K-D Tree [3] for fast search. Although rhythmic inconsistencies within queries are ignored for computational efficiency, the results show that in practice we can achieve reasonable performance. Searching continuous pitch contours at arbitrary locations was tried in the past [4], but computation-intensive dynamic programming was used for the matching.

While we agree that symbolic melody descriptions are the future for robust melody-matching, with reliable music modeling and transcription methods pending we believe it worthwhile to explore the use of continuous pitch contours in a somewhat traditional, signal-matching framework that is fast enough for practical use.

In addition, it is hard to tell from QBH experiments using MIDI target data how well the same system would perform on arbitrary polyphonic music for which the transcriptions are unavailable and must be extracted automatically and crudely. Assuming perfect note transcriptions could lead to QBH methods that are overly sensitive to the integrity of the transcription and turn out to have little value in such real-world scenarios. Therefore, in this study we also conduct a preliminary QBH experiment on "real-world" data, i.e., commercial recordings of polyphonic music from which dominant pitch contours are obtained using an automatic f_0 tracking method.

Wavelets [5] have a rich history of diverse applications in the areas of signal coding and matching. In particular, they have been used in the past to match whole image contours [6] with robustness to affine transformations, and also to encode f_0 contours for speaker identification [7]. In the former case, the wavelet coefficients were used to match whole contours, while in the latter, to encode the f_0 contour using compact dyadic wavelet coefficients. In our study, to match f_0 contours for the purpose of melody matching, we employ "redundant" sets of wavelets defined on non-integer scale and time indices to encode segments of varying locations and time scales.

Note that throughout this paper, we conveniently assume that "main melody" and "dominant pitch contour" both mean "dominant f_0 contour," although strictly speaking, all three concepts have subtle differences.

2. INDEXING VIA REDUNDANT WAVELETS

2.1 Brief Overview of Wavelets and Notation

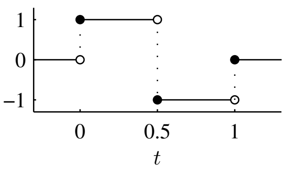


Figure 1. The Haar wavelet, $\psi(t)$

It is well known that a real, continuous-time signal $x(t)$ may be decomposed into a linear combination of a set of wavelets that form an orthonormal basis of a Hilbert Space [5]. First, we define a wavelet as

$$\psi_{m,n}(t) = 2^{-m/2}\psi\left(2^{-m}t - n\right) \qquad (1)$$

for $m, n \in \mathcal{R}$ (real numbers) where m is a dilation factor, n is a displacement factor, and $\psi(t)$ is some mother wavelet function. In this paper, we use the Haar Wavelet in Fig.1. It is easy to see that the support of (1), then, is

$$t \in [n2^m, (n+1)\,2^m) \qquad (2)$$

The corresponding wavelet coefficient of a signal $x(t)$ is

$$\langle x(t), \psi_{m,n}(t)\rangle = \int_{-\infty}^{+\infty} x(t)\psi_{m,n}(t)\,dt \qquad (3)$$

It is well known that when m, n are integers $j, k \in \mathcal{Z}$ (integers), $\{\psi_{j,k}\}$ form an orthonormal basis and $x(t)$ is a linear

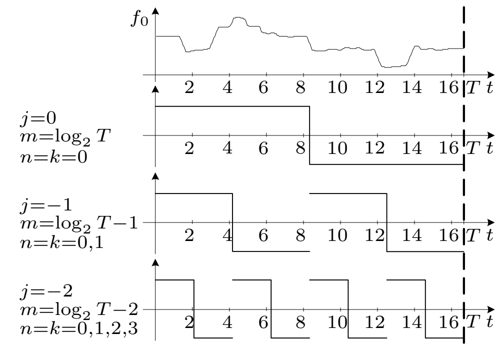


Figure 2. Example query pitch contour $q(t)$ with support $[0, T)$ and "dyadic-equivalent" wavelets $\psi_{m,n}$ that correspond to some of the dyadic wavelets $\psi_{j,k}$ of $q(Tt)$. The vertical dotted line indicates T. The wavelet amplitudes in the figure are not plotted to scale.

combination of the resulting "dyadic" wavelet coefficients:

$$x(t) = \sum_{j,k\in\mathcal{Z}} \langle x(t), \psi_{j,k}(t)\rangle\, \psi_{j,k}(t) \qquad (4)$$

Since signals are often represented by a compact set of coefficients, we can efficiently compare real signals using

$$\int_{-\infty}^{+\infty} \{x(t) - y(t)\}^2\,dt = \sum_{j,k\in\mathcal{Z}} \left(\langle x, \psi_{j,k}\rangle - \langle y, \psi_{j,k}\rangle\right)^2 \qquad (5)$$

Throughout this paper, we always assume $m, n \in \mathcal{R}$ and $j, k \in \mathcal{Z}$.

2.2 Application of Wavelets to Pitch Contour Matching

Assume some query f_0 contour $q(t)$, shown in Fig. 2. Also assume a pitch contour $p(t)$ of a target song, shown in Fig. 3 representing the "dominant" f_0 in a piece of polyphonic music. The query contour closely resembles a portion of the target contour, and our goal is to locate this segment. Given two contour segments representing identical melody, there are two different types of scaling that must be considered before attempting to directly compare them. The first one is in frequency, resulting from difference in musical key, which will cause one contour to be a scaled version of the other in the linear frequency domain. In the log-frequency domain, it will be a linear translation. The second scaling is in the time domain, resulting from difference in tempo. Notice that the two example melodies are sung at different speeds. The query is about 17 seconds long, while the matching segment in the target is about 12 seconds long. Both of these issues prevent us from directly comparing $p(t)$ and $q(t)$, and they will now be addressed.

2.2.1 Key Normalization

First, assume some signal $x(t)$ defined arbitrarily on $[0, 1)$ and 0 elsewhere. Since $\psi_{j,0} = 2^{-j/2}$ in $[0, 1)$ when $j > 0$, we have

$$\langle x, \psi_{j,0}\rangle = 2^{-j/2} S_x \;(j > 0)\,,\; S_x \triangleq \int_0^1 x(t)\,dt \qquad (6)$$

Also note that $\langle x, \psi_{j,k}\rangle = 0$ in $[1, 0)$ for $j > 0$ and $k \neq 0$. From these relations it follows that the wavelet expansion of $x(t)$ can be decomposed as follows:

$$\begin{aligned} x(t) &= \sum_{j\le 0, k\in \mathcal{Z}} \langle x, \psi_{j,k}\rangle \psi_{j,k} + \sum_{j>0,k=0} \langle x, \psi_{j,k}\rangle \psi_{j,k} \\ &\quad + \sum_{j>0,k\neq 0} \langle x, \psi_{j,k}\rangle \psi_{j,k} \\ &= x_N(t) + S_x \sum_{j>0,k=0} 2^{-j} + 0 = x_N(t) + S_x \end{aligned} \tag{7}$$

where we have defined

$$x_N(t) \triangleq \sum_{j\le 0, k\in\mathcal{Z}} \langle x, \psi_{j,k}\rangle \psi_{j,k} \tag{8}$$

From the orthogonality property of the wavelets, and the fact that $x(t)$ is 0 outside of $[0, 1)$, note that

$$\langle x_N, \psi_{j,k}\rangle = \begin{cases} \langle x, \psi_{j,k}\rangle & (j,k) \in \mathcal{W} \\ 0 & \text{all other } j,k \end{cases} \tag{9}$$

where we define the set $\mathcal{W}$ of tuplets (j, k) that correspond to the dyadic wavelets in $[0, 1)$:

$$\mathcal{W} = \left\{(j,k) : j \le 0, 0 \le k \le 2^{-j} - 1, j \in \mathcal{Z}, k \in \mathcal{Z}\right\} \tag{10}$$

Now, assume another signal $y(t) = x(t) + c$ in $[1, 0)$ and 0 elsewhere. Since $S_y = S_x + c$, we can see from (7) that $y_N(t) = x_N(t)$. Hence, for any arbitrary $x(t)$ and $y(t)$ on $[0, 1)$, we can obtain "level-normalized" signals $x_N(t)$ and $y_N(t)$ that are independent of constant bias. In our case, "level" is in fact "key" when $x(t)$ and $y(t)$ are log-frequency pitch contours, since key shifts will result in constant biases. To compute their mean squared distance in a "level(key)-normalized" way, we use, instead of (5),

$$\int_{-\infty}^{+\infty} \{x_N(t) - y_N(t)\}^2 \, dt = \sum_{j,k\in\mathcal{W}} \left(\langle x, \psi\rangle - \langle y, \psi\rangle\right)^2 \tag{11}$$

2.2.2 Time and Key Normalization of the Query

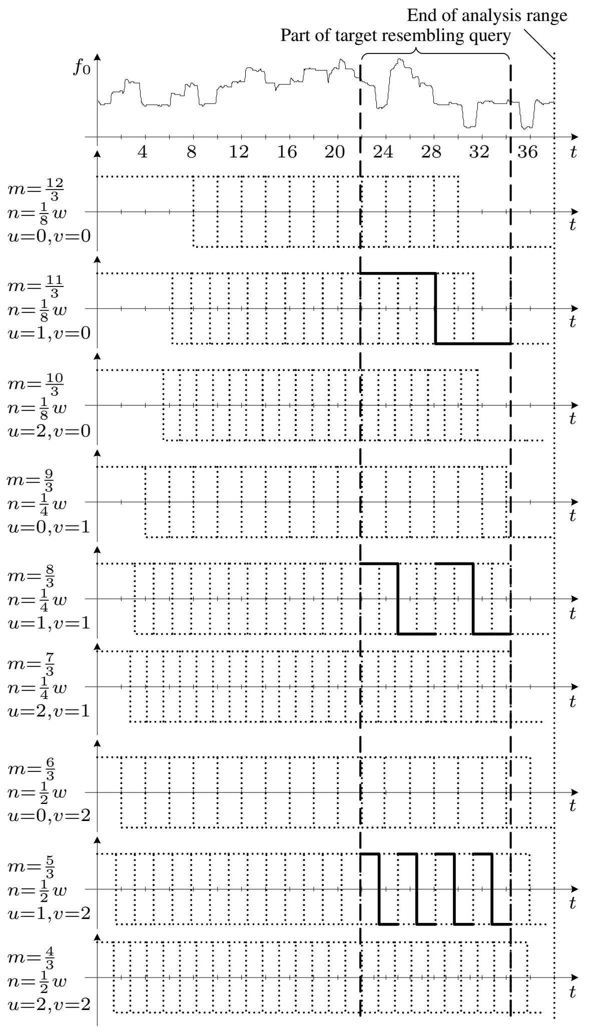
Assume that the query signal $q(t)$ is defined arbitrarily in $[0, T)$ and 0 elsewhere. The first step is to time-scale it into a "time-normalized" signal $q'(t)$ defined on $[0, 1)$ and 0 elsewhere:

$$q'(t) \triangleq q(Tt) \tag{12}$$

Using (3) and (1), it is easy to see that

$$\begin{gathered} \langle q'(t), \psi_{j,k}(t)\rangle = T^{-1/2} \langle q(t), \psi_{m,n}(t)\rangle \\ m = j + \log_2 T, \; n = k \quad (j, k \in \mathcal{Z}) \end{gathered} \tag{13}$$

Fig. 2 shows $\psi_{m,n}$ for $(j, k) \in \mathcal{W}$ when $j = 0, -1$, and -2, which corresponds to $m = \log_2 T$, $-1 + \log_2 T$, and $-2 + \log_2 T$, respectively. The wavelets $\{\psi_{m,n}\}$ could be regarded as the "dyadic-equivalent" wavelets of $q(t)$ – the wavelets applied to $q(t)$ that are equivalent to the dyadic wavelets applied to its time-normalized version $q'(t)$.



End of analysis range
Part of target resembling query
f_0
4
8
12
16
20
24
28
32
36
t
$m=\frac{12}{3}$ $n=\frac{1}{8}w$ $u=0,v=0$
$m=\frac{11}{3}$ $n=\frac{1}{8}w$ $u=1,v=0$
$m=\frac{10}{3}$ $n=\frac{1}{8}w$ $u=2,v=0$
$m=\frac{9}{3}$ $n=\frac{1}{4}w$ $u=0,v=1$
$m=\frac{8}{3}$ $n=\frac{1}{4}w$ $u=1,v=1$
$m=\frac{7}{3}$ $n=\frac{1}{4}w$ $u=2,v=1$
$m=\frac{6}{3}$ $n=\frac{1}{2}w$ $u=0,v=2$
$m=\frac{5}{3}$ $n=\frac{1}{2}w$ $u=1,v=2$
$m=\frac{4}{3}$ $n=\frac{1}{2}w$ $u=2,v=2$

Figure 3. Example target pitch contour $p(t)$ and a redundant set of wavelets with design parameters $D = 3$, $M = 12$, $V = 2$, and $E = 3$ encoding the contour at different locations over a range of time scales. The bold broken line shows the segment resembling the query in Fig.2, and the bold lines show the "dyadic-equivalent" wavelets that encode this segment

Now, if we only compute those wavelet coefficients for $(j, k) \in \mathcal{W}$, we can obtained the *key-normalized*, *time-normalized* signal $q'_N(t)$. From (9) and (13), we have

$$\langle q'_N, \psi_{j,k}\rangle = \begin{cases} T^{-1/2} \langle q, \psi_{m,n}\rangle & \begin{smallmatrix}(j,k)\in\mathcal{W} \\ m=j+\log_2 T, n=k\end{smallmatrix} \\ 0 & \text{all other } j,k \end{cases} \tag{14}$$

2.2.3 Normalization and Redundant Encoding of Targets

For the target pitch contour $p(t)$, we do a *redundant* wavelet analysis so that we can search multiple, overlapping sections of varying time scales in $p(t)$. Some sort of regularity must be imposed on the scale factors and analysis intervals so that the coefficients can be used efficiently. Note that there can be many ways to do this, and here we are proposing one such method. While we present a general formulation of our design, the easiest way to understand this section is by studying the specific example in Fig.3.

We compute a "redundant" set of wavelet coefficients $\{\langle p, \psi_{m,n}\rangle : u, v, w\}$, where we set

$$m = \frac{M-u}{D} - v,\ u = 0, 1, \cdots, D-1,\ v = 0, 1, \cdots, V \tag{15}$$

The constant D represents the amount of resolution in the time scales over which the redundant analysis is done. $M > D$ represents some upper limit in m, u is a time scale factor, v is a nonzero integer, and $V < \frac{M}{D}$ represents some lower limit in m. For each m, the possible values of n are

$$n = \frac{1}{2^{E-v}} w,\ w = 0, 1, \cdots \tag{16}$$

$E > V$ represents the amount of time resolution. Fig. 3 shows the wavelets with $D = 3$, $M = 12$, $V = 2$ and $E = 3$.

Now, consider the part of $p(t)$ in

$$t \in [n_0 2^{m_0}, (n_0 + 1)\, 2^{m_0}) \tag{17}$$

which is exactly the support of ψ_{m_0,n_0} by (2). We also constrain m_0 and n_0 to conform to (15) and (16):

$$\begin{cases} m_0 = \dfrac{M-u_0}{D} - v_0 & \begin{array}{l} 0 \le u_0 \le D-1, 0 \le v_0 \le V, \\ u_0, v_0 \in \mathcal{Z} \end{array} \\ n_0 = \dfrac{1}{2^E} w_0 & w_0 \ge 0, w_0 \in \mathcal{Z} \end{cases} \tag{18}$$

The time-normalized version of this portion of $p(t)$, assuming zero elsewhere, is

$$p'(t) = \begin{cases} p\left(2^{m_0}(t + n_0)\right) & t \in [0, 1) \\ 0 & \text{elsewhere} \end{cases} \tag{19}$$

It is easy to see that the corresponding key-normalized, time-normalized signal $p'_N(t)$ will have wavelet coefficients

$$\langle p'_N, \psi_{j,k}\rangle = \begin{cases} 2^{-m_0/2}\langle p, \psi_{m',n'}\rangle & \begin{array}{l} {\scriptstyle (j,k)\in\mathcal{W}} \\ {\scriptstyle m'=m_0+j} \\ {\scriptstyle n'=k+2^{-j}n_0} \end{array} \\ 0 & \text{all other } j, k \end{cases} \tag{20}$$

Now, from (15), (16), and (18) one can see that all coefficients $\langle p, \psi_{m',n'}\rangle$ required above can always be found in the set of wavelets $\{\langle p, \psi_{m,n}\rangle : u, v, w\}$ up to scale level $j = v_0 - V$. One can also notice that many wavelet coefficients can be "reused" in the sense that they contribute to more than one contour segment. In the example in Fig.3, we see $\{\psi_{m',n'}\}$ for $j = 0, -1, -2$ with $m_0 = \frac{11}{3}$ and $n_0 = \frac{14}{8}$, which encode the section of $p(t)$ that pertains to the query $q(t)$ in Fig.2.

Using the wavelet coefficients in (14) and (20), we can compute the distance between the key- and time-normalized query $q'_N(t)$ and target segment $p'_N(t)$ using (11). The distance will be an approximation, since we cannot take the coefficients over the entire set $\mathcal{W}$ but over a finite number of scale levels that provides sufficient accuracy (e.g., $j = 0$ to $j = -4$ for a 7s pitch contour sampled at 10ms).

To account for variations in tempo, we compare segments over a range of values of m_0. Note first that if the query and target had the same tempo, we should have $m_0 = \log_2 T$, which would produce portions of $p(t)$ with length T in (17), to obtain the most accurate match. Now, if we allow the query's tempo to be as slow as half the target's tempo and as fast as twice the target's tempo, we can let m_0 vary within the range

$$-1 + \log_2 T < m_0 < 1 + \log_2 T \tag{21}$$

which results in around $2D$ different values of m_0 according to the system design.

2.2.4 *Two-Stage Search of Arbitrary Target Locations*

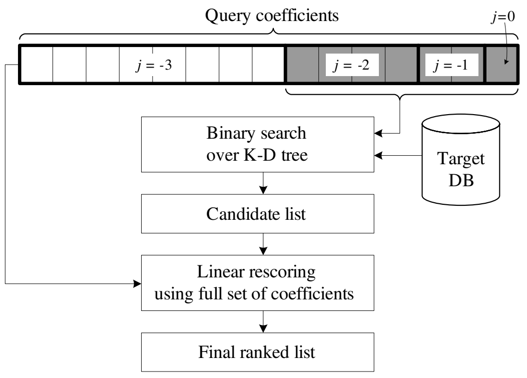


Figure 4. Schematic overview of two-stage search. In this example, 7 wavelet coefficients are indexed by a K-D Tree, and 15 coefficients are used for the linear rescoring.

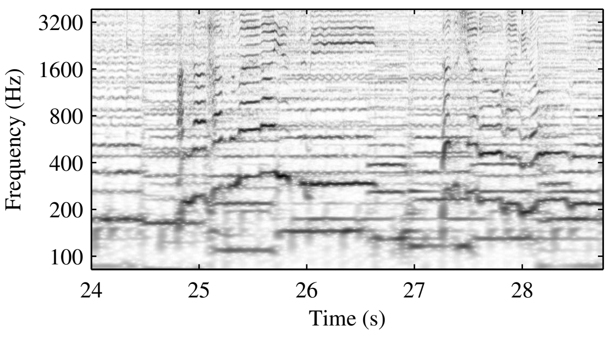
The variable n_0 in (19) controls the location of the target segment compared with the query. The resolution of the wavelet locations can be controlled to find a good compromise between speed and accuracy. For efficient comparison of a query with a large number of targets, the possible dyadic-equivalent coefficients embedded in every target (i.e., the coefficients in (20)) can be indexed as coordinates in a binary K-D Tree [3] with a fixed number of dimensions. Each leaf in the tree coarsely represents a melodic fragment in the target database. At the first stage of the search, the query coefficients are appropriately scaled to form a search sphere that is used to find tree leaves that are spatially close to the query, resulting in a list of candidate melody fragments. At the second stage, a linear search is conducted over the candidates using a larger number of coefficients to more accurately compute (11), which is then used to rank the results as shown in Fig.4.

In practice, no more than 31 wavelet coefficients ($j = 0$ to $j = -4$ in $\mathcal{W}$) are usually sufficient to represent a melody segment with length 7s sampled every 10ms. In such a case, the first 7 coefficients ($j = 0$ to $j = -2$ in $\mathcal{W}$) can be indexed in the K-D tree, while the full 31 coefficients are used in the linear rescoring stage.

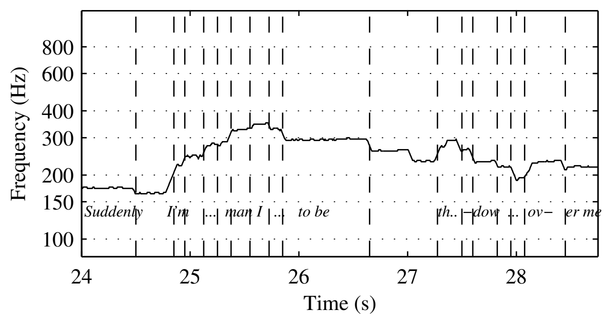
In terms of computational complexity, if dynamic programming(DP) were used to search for a query of length L_q[frames] in a target of length L_t[frames], scores would have to be calculated for $L_q L_t$ coordinates (assuming no pruning). If a linear search were used with the proposed method, we would need to compute only $kL_t(k << L_q)$ vector distances, where k is essentially a constant since the number of wavelet coefficients (as in Fig.3) increases linearly with L_t. The addition of a K-D tree further reduces this number drastically, making the computational gains of the proposed method even more apparent.

3. EXPERIMENT

3.1 Pitch Contour Extraction



(a) Log-frequency spectrum

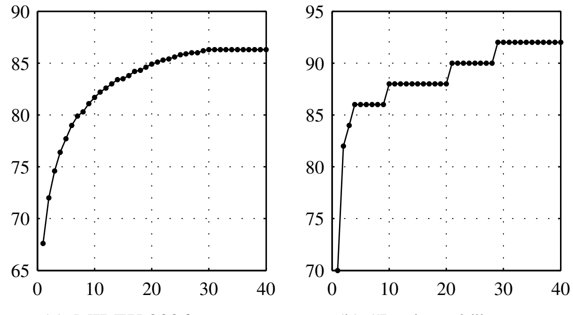


(b) Dominant f_0 contour

Figure 5. (a) Log magnitude of log-frequency spectrum (dark is high) from 82.4 Hz to 3.84 kHz of a segment of *Yesterday* by *The Beatles*. Frequency components of both voice and instrumental accompaniment are clearly visible. (b) Pitch contour of segment with hand-marked note boundaries (broken vertical lines) and corresponding lyrics in select locations (full lyrics are "Suddenly, I'm not half the man I used to be, there's a shadow hanging over me")

A simple method based on known techniques was used to obtain dominant f_0 contours from music recordings. The Constant-Q Transform [8] of each music signal was taken to obtain spectral components on a log-frequency scale. Fig.5(a) shows the spectrogram for a segment of *Yesterday* by *The Beatles*. Next, we assigned scores for each (t, f) on the time-frequency plane by computing weighted sums of the spectral components at harmonics of f [9]. After limiting the range of the dominant pitch via some heuristics, we applied dynamic programming on the $t - f$ plane of scores to obtain a continuous pitch contour [10] that maximizes the sum of pitch scores along its path. Fig.5(b) shows the pitch contour obtained for the *Yesterday* example. While the overall structure of the contour reflects the vocal melody of this part of the song, we can notice that in the non-vocal sections between "Suddenly" and "I'm" and between "to be" and "there's", the dynamic programming picked up the pitch of the strings in the background. Inflections in vocal pitch inevitably produced during singing, and other minor deviations from what is probably the "true" music score are also reflected in the continuous contour. However, we made no attempt to identify and compensate for any such deviations or discriminate between vocal and non-vocal sections, and directly used the whole pitch contour from every target in the database in our experiments.

3.2 QBH Test



(a) MIREX 2006 test set (b) "Real-world" test set

Figure 6. Search performance for (a) MIREX 2006 test set and (b) "real-world" test set. The vertical axis represents the inclusion rate(%), and the horizontal axis is the number of search results.

Two experiments were conducted: the first on monophonic music to validate our method with existing QBH tasks, and the second on polyphonic music to make a preliminary assessment of its use in real-world scenarios. For both experiments, the dominant f_0 contour was automatically extracted from query and target data using the aforementioned method. Contours were sampled every 10ms.

For the first experiment, we used the MIREX 2006 QBSH test set (see description in [2]). All target data in this set are monophonic MIDI data, so we first converted them to WAV format. Each song in the database was 29.9s long on average (17 hours total for the database of 2,048 songs). Fig.6(a) shows the inclusion rate for varying number of search results, i.e., the rate at which the correct melody was ranked within the top n of all returned results. For $n = 20$, the inclusion rate was 84.9%, which is significantly lower than the state-of-the-art [2], 96.4%. Note, however, that the latter system constrained the queries to occur at only the beginning of music phrases. Since almost all queries in the MIREX 2006 test set start at the beginning of their targets, such a data set would greatly favor systems with such constraints. Our proposed system, on the other hand, made no assumptions on starting locations and exhaustively searched over all possible locations, limited only by the wavelet parameters. Also, tempo variation is taken into account from the very beginning of the search, not just at the latter fine search stage. Hence, the search space was larger, which resulted in more room for confusion. At the same time, the search time for each query was usually less than one second on a 3.2GHz processor depending on system parameters.

For the second experiment, we used a "real-world" database consisting of 613 acoustic recordings of songs with instrumental accompaniment, totaling around 37 hours of audio (average 3.6 minutes per song). 155 of the songs were from the RWC Music Database [11], and the rest were commercially-distributed pop songs. A preliminary set of queries were obtained from six non-professional singers – three male, and three female. Each person was asked to sing several easy and well-known songs including "Happy Birthday," "The Alphabet Song," and "Are You Sleeping, Brother John?" from which query segments at random locations were extracted. Each query was 5$\sim$12 seconds long, and there were a total 50 queries. We informally verified that the songs in the target database corresponding to these queries had reasonably clear dominant f_0's, but there were still noticeable errors in the f_0 extraction due to instrumental accompaniment, like in the example in Fig. 5(b). Fig. 6(b) shows the inclusion rate for a varying number of search results. The inclusion rate was 86% for $n = 5$ and 88% for $n = 20$, which seems similar to that of another state-of-the-art system [12] that also allows queries to begin at random locations but uses a MIDI database. We are cautious in directly comparing the performance of the two systems, however, because they differ in experimental setup. Nevertheless, our results are promising because we used a database of polyphonic recordings instead of MIDI data. Larger data sets and larger numbers of queries will have to be used in the future to more rigorously assess real-world performance.

4. CONCLUSION AND FUTURE WORK

We have proposed an efficient method of indexing and matching music melodies based on their continuous pitch contours while allowing partial matches at arbitrary locations using redundant wavelet transformations. By directly comparing continuous pitch contours instead of their note transcriptions as in most existing methods, we avoid the compounding of transcription errors. On the other hand, our method is also computationally efficient because it uses the mean squared sum between fixed vectors instead of dynamic programming, while at the same time being able to adjust for differences in tempo and key. Experiments were conducted on both existing monophonic MIDI databases and preliminarily on real-world recordings with instrumental accompaniment to show that the system can be practically applied, even when using a simple mean squared distance measure between key- and time-normalized contour segments. While the system still depends on reliable dominant pitch extraction, minor pitch tracking errors did not hurt performance because the overall pitch and rhythm structure of contours was compared. One trade-off for the system's efficiency is that it does not explicitly account for rhythmic variations within queries as do techniques based on string-matching. Much work is being done in the MIR community toward model-based symbolic representations that allow a more modular framework for indexing and search, such as via HMMs, and we plan to leverage the insights gained in our work to this end.

5. ACKNOWLEDGMENTS

Thanks to Lei Wang for kindly sharing the data for the MIREX 2006 QBH experiment used in [2].

6. REFERENCES

[1] D. Mazzoni and R. B. Dannenberg. Melody matching directly from audio. In *Proc. ISMIR*, 2001.

[2] L. Wang, S. Huang, S. Hu, J. Liang, and B. Xu. Improving searching speed and accuracy of query by humming system based on three methods: Feature fusion, candidates set reduction and multiple similarity measurement rescoring. In *Proc. INTERSPEECH*, pages 2024–2027, 2008.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Comm. ACM*, 18, 1975.

[4] L. Guo, X. He, Y. Zhang, and Y. Lu. Content-based retrieval of polyphonic music objects using pitch contour. In *IEEE Int. Conf. Acoust., Speech. Signal Processing*, pages 2205–2208, 2008.

[5] I. Daubechies. *Ten Lectures on Wavelets*. SIAM: Society for Industrial and Applied Mathematics, 1992.

[6] Q. M. Tieng and W. W. Boles. Complex daubechies wavelet based affine invariant representation for object recognition. In *IEEE ICIP*, pages 198–202, 1994.

[7] F. Farahani, P.G. Georgiou, and S.S. Narayanan. Speaker identification using supra-segmental pitch pattern dynamics. In *IEEE Int. Conf. Acoust., Speech. Signal Processing*, 2004.

[8] J. C. Brown and M. S. Puckette. An efficient algorithm for the calculation of a constant Q transform. *IEEE Trans. Audio, Speech, and Language Processing*, 92:2698–2701, 1992.

[9] D. J. Hermes. Measurement of pitch by subharmonic summation. *J. Acoust. Soc. Am.*, 83(1):257–264, 1988.

[10] B. Secrest and G. Doddington. An integrated pitch tracking algorithm for speech systems. In *IEEE Int. Conf. Acoust., Speech. Signal Processing*, April 1983.

[11] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical, and jazz music databases. In *Proc. ISMIR*, pages 287–288, 2002.

[12] E. Unal, E. Chew, P.G. Georgiou, and S.S. Narayanan. Challenging uncertainty in query by humming systems: A fingerprinting approach. *IEEE Trans. ASLP*, 16, 2008.

TONAL-ATONAL CLASSIFICATION OF MUSIC AUDIO USING DIFFUSION MAPS

Özgür İzmirli
Center for Arts and Technology
Computer Science Department
Connecticut College
oizm@conncoll.edu

ABSTRACT

In this paper we look at the problem of classifying music audio as tonal or atonal by learning a low-dimensional structure representing tonal relationships among keys. We use a training set composed of tonal pieces which includes all major and minor keys. A kernel eigenmap based method is used for structure learning and discovery. Specifically, a Diffusion Maps (DM) framework is used and its parameter tuning is discussed. Since these methods do not scale well with increasing data size, it becomes infeasible to use these methods in online applications. In order to facilitate on-line classification an out-of-sample extension to the DM framework is given. The learned structure of tonal relationships is presented and a simple scheme for classification of tonal-atonal pieces is proposed. Evaluation results show that the method is able to perform at an accuracy above 90% with the current data set.

1. INTRODUCTION

Audio key estimation is an important aspect of MIR. It informs many other tasks including music analysis, segmentation, cover song detection, modulation tracking, local key finding and chord recognition. In order to estimate the key, most key finding models use a similarity metric between predetermined reference features and the analyzed features from the audio. All of these approaches assume that the fragment of the piece being analyzed contains tonal music and furthermore that musical content is in a single key. These models generally lack mechanisms to detect music that is not tonal and hence would make best-guess estimates regardless of the tonal quality of the input. One important question, which is the topic of this paper, is how to determine whether a piece belongs to the tonal idiom: whether there are clear and unambiguous tonal implications or not.

In this work, we explore the utility of dimensionality reduction, manifold learning and structure discovery in the context of tonal versus atonal music audio classification. We investigate the possibility of learning a low-dimensional structure representing tonal relationships among pieces. We report on experiments that utilize Diffusion Maps to perform dimensionality reduction and feature extraction from high-dimensional spectral data. We use a set of audio recordings representative of all 24 keys as the reference training set and test the model with tonal and atonal audio fragments to evaluate its performance.

The structure of the paper is as follows: The next section makes reference to related work and explains the concept of tonalness. Section 3 describes kernel methods and DM in particular. This section also discusses the tuning of the width parameter of DM. Section 4 outlines the main outcomes and describes the evaluation method. Section 5 concludes the paper.

2. RELATED WORK

Temperley describes a probabilistic framework on symbolic data for measuring tonal implication, tonal ambiguity and tonalness for pitch-class sets [1]. According to his definition, tonal implication is the key implied by the pitch-class set being used. Ambiguity refers to whether a pitch set implies a single key or several keys. Tonalness is the degree to which a set is characteristic of common-practice tonality. In this sense, our work relates directly to the concept of tonalness. Our assumption is that a piece that conforms to pitch distributions of common practice tonality will have certain spectral properties that distinguish it from other types of pitch distributions such as those found in twelve-tone music or polytonality. These spectral properties, or so called spectral signatures, have native representations in a high-dimensional space and therefore need to be mapped to low-dimensional features to be useful - not only for classification purposes but also for visualization and geometrical interpretations. The remainder of the paper discusses a method to classify music audio based on the degree of tonalness.

In her thesis, Gómez applied her key finding method to an atonal piece by Schoenberg [2]. She observed that the correlations of her Harmonic Pitch Class Profile (HPCP) with the major and minor profiles, that are derived from Krumhansl's work, remained low throughout the piece, indicating ambiguity.

Izmirli reported on the performance of a template based key finding algorithm using a low-dimensional representation obtained through dimensionality reduction [3]. He graphed the performance of his method as a function of the number of dimensions and noted that 2 and 3 dimensions produced acceptable accuracy for the particular model he was using.

Purwins briefly discusses poly-tone analysis and tonal ambiguity in relation to Pitch Class Profiles that he uses in his key finding algorithm [4].

3. DIMENSIONALITY REDUCTION, MANIFOLD LEARNING AND STRUCTURE DISCOVERY

3.1 Method

In general, given a set of training data we would like to infer some parameterization of it such that new data can be efficiently compared to the training data. The parameterization can then be used for many different purposes including classification. In the following we present a method that performs dimensionality reduction on a training set of tonal audio in order to find a representative structure. The resulting low dimensional representation is then used to determine whether new input data resembles the training data or not; more specifically, if it is tonal or atonal. This section describes the method of dimensionality reduction used and a scalable extension for new data.

3.2 Kernel Methods

In contrast to the standard linear methods such as Principal Component Analysis (PCA) and Multidimensional Scaling (MDS) for dimensionality reduction, nonlinear methods are better suited to preserving local geometry. This is due to the fact that they attempt to approximate manifolds in the high-dimensional space by considering connectivity between neighboring points as opposed to capturing the global nature of the data. Nonlinear methods include Isometric Feature Mapping (ISOMAP), Kernel PCA and a class of kernel eigenmap methods including Laplacian Eigenmaps, Locally Linear Embedding (LLE), Hessian Eigenmaps (Hessian LLE) and Local Tangent Space Alignment (LTSA). In [5] Coifman and Lafon show that the kernel eigenmap methods are special cases of a general framework based on diffusion processes. Here, we follow a formulation for dimensionality reduction, manifold learning and data parametrization based on DM [5]. The major advantages of this approach over PCA and MDS are that it is nonlinear and preserves local structures. Kernel eigenmap methods rely on the idea that eigenvectors of a transition matrix representing the distances between points in the input space can be interpreted as coordinates on the data set.

3.3 Diffusion Maps

The concept of diffusion maps stems from dynamical systems and it is based on a Markov random walk on the graph of the data. The proximity of the data points is modeled as diffusion distances according to the affinity between neighboring points. DM preserves local geometry present in the high-dimensional input while performing dimensionality reduction.

Assume the data set containing k elements is given by X={x_0, x_1, x_2, ...,x_{k-1}} with x_i element of R^m**.** A pairwise similarity matrix L is calculated using a Gaussian kernel with parameter ε :

$$L_{ij} = w_\varepsilon(x_i, x_j) = e^{-\|x_i - x_j\|^2 / \varepsilon^2} \tag{1}$$

Furthermore, a diagonal normalization matrix is defined to make the sum of the rows of L equal 1:

$$D_{ii} = \sum_j L_{ij} \tag{2}$$

The normalized graph Laplacian is then given by the Markov matrix $M = D^{-1}L$. In order to find a mapping, Φ , from R^m to R^n, where m > n, an eigen-decomposition of M is performed. The eigenvectors and eigenvalues can be found by solving the equivalent generalized eigenvalue problem $L\phi = \lambda D\phi$. When ε in Eq. 1 is large enough, M is fully connected and has a unique eigenvalue of 1. From the remaining k-1 eigenvalues, n of the largest $1 > \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$ and their corresponding eigenvectors $\phi_1, \phi_2, ... \phi_n$ can be retained to map input samples from the high-dimensional space onto the lower dimensional feature space. The mapping is given by

$$\Phi : x_i \rightarrow [\lambda_1\phi_1(i), \lambda_2\phi_2(i), ... \lambda_n\phi_n(i)] \tag{3}$$

where index i in $\phi_n(i)$ represents the i'th element of the eigenvector.

3.4 Determining ε

The width parameter, ε, controlling the Gaussian in Eq. 1 has an effect on the locality of the structure captured. For example, a relatively small ε will capture the local structure better. However, if ε is too small then matrix L will have many small elements and hence, low connectivity, which will prevent it from capturing the desired structure. An unnecessarily large value on the other hand will cause the method to overlook the local structure. Although the value of this parameter is data dependent, fortunately, its choice can be automated.

Several approaches have been proposed to determine the optimal value of ε. The average of the distances between nearest neighbors in the data set are used in [6]. Another method is to adjust the parameter until every point has a significant connection to at least one neighbor. We follow the approach used in [7]. The method consists in searching for a point on the linear segment of the log-log graph of $T(\varepsilon)$ and ε, where

$$T(\varepsilon) = \sum_i \sum_j w_\varepsilon(x_i, x_j) \tag{4}$$

The graph contains two asymptotes, $\lim_{\varepsilon\to\infty} T(\varepsilon)$ and $\lim_{\varepsilon\to 0} T(\varepsilon)$ which are connected by an approximately linear line. We choose ε corresponding to the midpoint between the asymptotes in this graph.

3.5 Scalability and Out-of-Sample Extensions

Kernel methods described in the previous section have been successfully applied to dimensionality reduction and manifold learning. They are, however, computationally expensive and do not scale well to large data sets. They also do not directly accommodate new data and in that sense are limited to their training set requiring a new run every time new data is to be added.

Out-of-sample extensions are approximations that utilize the original eigen-decomposition to compute the mapping of new samples that do not belong to the original data set. In [8] the authors discuss how to compute out-of-sample extensions for various kernel methods. We employ the Nyström extension to find the mapping of the new data point as follows:

$$\tilde{\phi}_j = \lambda_j^{-1} \sum_{i=0}^{k-1} w_\varepsilon(x_{new}, x_i)\,\phi_j(i) \Big/ \sum_{p=0}^{k-1} w_\varepsilon(x_{new}, x_p) \qquad (5)$$

Once $\tilde{\phi}_j$ $(j = 1..n)$ is calculated, it is substituted for the corresponding eigenvectors in Eq. 3 to obtain the position in the lower dimensional feature space.

Calculation of the Nyström extension is computationally light. The denominator of Eq. 5 can be precalculated and the numerator is just a scaled sum of k vectors.

4. LEARNING TONAL STRUCTURE

4.1 Geometric Models of Pitch and Key

Many geometric models of pitch and key space have been proposed that originate from music theory and cognitive science. These include structures such as a circle, torus, helix and double helix (See for example [9] and [10]). Furthermore, most of these geometric structures are cyclic at one if not at multiple levels. In its simplest form, we know that key arrangements of the 12 major keys moving in fifths forms a circle. Similarly minor keys follow the same pattern. Obviously, this is based on the assumption that the music is performed in an equal tempered system.

In [11] it has been demonstrated that this or another cyclic structure can be captured from the audio of musical instruments playing diatonic scales. In this 2-dimensional space, points that represent key centers are organized in such a way that if we draw lines between the closely related keys the resulting arrangement forms a closed loop visiting each key center once.

4.2 Learning Structure from Audio Data

In this work, we explore the utility of structure discovery in the context of tonal versus atonal music audio. We obtain a chroma representation similar to [12] from the Hanning windowed short-time Fourier Transform. A 12-element chroma vector is obtained by summing the semitone frequency ranges of the amplitude spectrum according to pitch-class equivalence. That is, the semitone frequency range around the fundamental frequency of a note, the range around its octave and its second octave etc. all map to a single bin in the chroma vector.

Initially, we employed the method outlined in Section 3 to test if it was able to learn a low-dimensional structure using only recordings of tonal music. The training data, *X*, comprised of chroma vectors calculated from initial fragments of 289 pieces containing compositions mainly from the common practice period. Each point in the data set, x_i, represents the average of 30 seconds of music taken from the beginning of each piece. This duration was determined experimentally and can be chosen to be shorter without significantly effecting the algorithm's output. Note that the training is unsupervised and although the key labels are known from the titles of pieces they are not part of the input. The key distribution of the data set, although not completely uniform, is such that the lowest number of pieces in the same key is 9. For a collection of this size, a completely even distribution would require 12 pieces for each of the 24 keys. Although it would have been possible to either trim all pieces to the same number or add more pieces to bring the key totals to the same level, the current distribution was kept to observe the sensitivity of the DM algorithm to the density of samples on the manifold. It should be mentioned that sampling density is a main concern for many manifold learning algorithms and may need special attention if the spatial distribution is unbalanced.

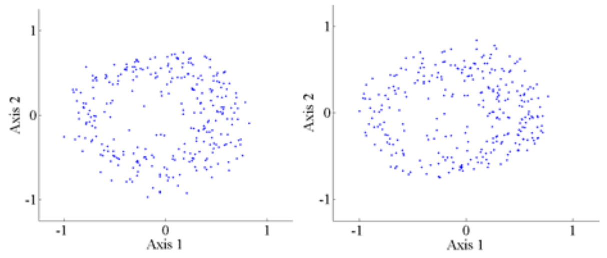


Figure 1. The input data set consisting of tonal pieces mapped to the first two dimensions. A circular structure resembling the circle-of-fifths is captured for the chroma representation (left) and for the spectral representation (right).

The left plot in Figure 1 shows the mapping Φ with $n = 2$ in response to the input data set, *X*, based on the chroma representation as described above. The out-of-sample extension is not used for this part. A circular structure is clearly visible in the figure which means it was able to capture some kind of circularity. Then again, this highly resembles the circle-of-fifths pattern. We verified the order of keys by analyzing their key labels to make sure the neighboring clusters were in a fifths relationship. There was considerable scatter within classes that belong to the same key. There was also significant overlap between classes, yet, the circle-of-fifths pattern was evident. The output for the spectral representation is shown in the right plot in Figure 1. These vectors are the same spectral vectors used to calculate the chroma representation. The reason for inclusion of the spectral vectors is to see if DM is able to obtain a mapping on par with or better than the traditional chroma representation. It should be noted that the uneven density of points does manifest itself in both plots without loss of generality of the result.

To further demonstrate the circle-of-fifths pattern we used chroma templates obtained from the audio of monophonic instrument sounds playing major scales. Each of the 12 templates consists of a single scale over multiple octaves. The details of the construction of the templates can be found in [13] and [14]. The templates were mapped using the out-of-sample method with respect to the tonal training data, X, described above. The results are shown in Figure 2. Here, each template represents an ideal key position in the feature space and the projection serves as a demonstration of the circle-of-fifths relationship among the 12 major keys. A similar order has also been observed for minor keys.

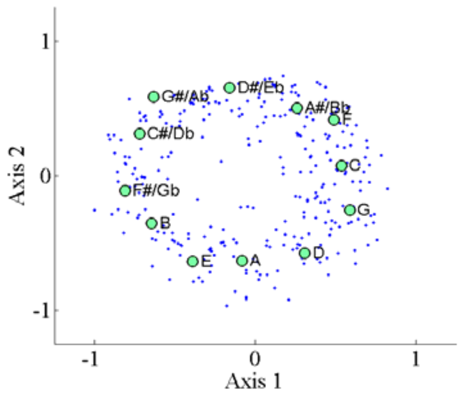


Figure 2. Mapping of audio templates to the first two dimensions. The labeled points representing the major templates are superimposed on the chroma based representation in Figure 1 (left).

4.3 Training and Test Data for Evaluation

Starting from the observation that a data set containing pieces in all 24 keys results in the constellations shown in Figure 1, we turn to testing the DM model with tonal and atonal data using the out-of-sample extension described above. For this part we added 25 complete atonal pieces composed by Boulez, Schoenberg and Webern. Both the tonal and atonal pieces were segmented into 10-second fragments. There are 599 atonal fragments and 925 tonal fragments in the data set. Each fragment is represented as a point, x_i, found by dividing the spectral or chroma vectors by their L_2 norm, and an associated tonal/atonal label serving as ground truth for evaluation purposes. The frequency ranges of interest for both representations are 55 - 2000 Hz. The training data set was constructed as follows: 60% of the tonal points were randomly chosen and were used to train the DM model. The remaining 40% were added to the test set accompanied by an equal number of points randomly chosen from the atonal set. After calculating the original mapping using 60% of the tonal points, the out-of-sample calculations were performed on the test set. Figure 3 shows the mapping of the test results onto the first two dimensions. These results are overlaid with the training points to show the nature of generalization the extension brings.

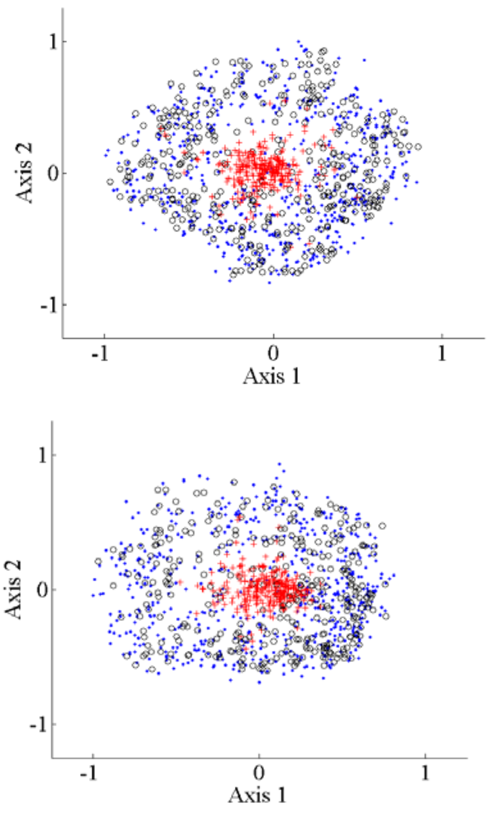


Figure 3. Training and test data mapped to the first two dimensions: chroma based inputs (top) and spectrum based inputs (bottom). Tonal training data are shown with dots (.), the tonal test data are shown with circles (○) and the atonal test data are shown with pluses (+).

4.4 The Tonal-Atonal Classifier

As can be easily observed from Figure 3, the tonal training points and the tonal test points tend to appear at positions closer to the outer circular pattern whereas the atonal test points tend to appear near the center. Therefore, we simply choose to use the Euclidean norm of a point in the feature space to quantify its tonalness as defined in Section 2. For the 2-dimensional case, the performance of the classifier is given by the peak classification accuracy in which a circle acts as the class boundary. It should be noted that although we treat the problem as a two-class classification task in this paper, in fact, the calculated tonalness is a continuous entity and is indeed correlated with the degree of the musical fragment's tonal implication. The distances in the feature space can be used to quantify the degree of tonalness. A study of the tonalness of transpositional type pitch class sets can be found in [15].

4.5 Results

An average accuracy was calculated by running the above classification 10 times. The chroma based classification resulted in an average accuracy of 91.2% and the spectrum based classification resulted in 90.4% accuracy.

As an alternative feature we ran a classification task based on the variance of the chroma and spectrum vectors (x_i) to see how they compared with the presented method. The intuition was that the chroma vector corresponding to tonal pieces would have more variance compared to atonal pieces because it would exhibit a strong interleaved response across bins of the vector. i.e. say, for C major, one would expect the bins corresponding to the white keys to be strong and those of the black keys to be weak. On the other hand, atonal pieces would have a more uniform spread across the bins. The chroma variance feature performed at 84.9% accuracy. The same reasoning does not really apply to the spectrum vectors because they are fairly sparse compared to the chroma vectors but nevertheless we tested the feature and obtained 64.4% accuracy; very low as expected.

5. CONCLUSION

In this paper we have discussed a method based on Diffusion Maps to perform tonal-atonal classification of music audio. Initially, we learn a low-dimensional structure representing pitch distributions that pertain to the tonal idiom. We then extend the learned mapping to new points and test the performance of the method. The learned cyclic structure is demonstrated through a display of the projected circular constellation of the training points and the projection of major scale templates representing ideal key locations in relation to this constellation. The use of the learned cyclic structure in quantifying tonalness is also discussed. Finally, results are presented for the tonal-atonal classification task for chroma representations as well as raw spectral representations. The results are encouraging and promising. Future work involves exploring more general mechanisms for calculating the structure similarity between training and test structures, and finding optimal training sets for faster and more efficient operation.

6. REFERENCES

[1] D. Temperley: "The Tonal Properties of Pitch-Class Sets: Tonal Implication, Tonal Ambiguity, and Tonalness," Eleanor Selfridge-Field and Walter Hewlett, eds. *Computing in Musicology*, *Tonal Theory for the Digital Age*, Vol. 15, 24-38, 2008.

[2] E. Gómez: "Tonal Description of Music Audio Signals," *Ph.D. Dissertation*, Pompeu Fabra University, Barcelona, 2006.

[3] Ö. İzmirli: "Audio Key Finding Using Low-Dimensional Spaces," *Proceedings of the International Conference on Music Information Retrieval*, Victoria, Canada, 2006.

[4] H. Purwins: "Profiles of Pitch Classes Circularity of Relative Pitch and Key – Experiments, Models, Computational Music Analysis, and Perspectives," *Ph.D. Thesis*, Berlin University of Technology, 2005.

[5] R. R. Coifman and S. Lafon: "Diffusion Maps," *Applied and Computational Harmonic Analysis*, 21, pp. 5–30, July, 2006.

[6] S. Lafon: "Diffusion Maps and Geometric Harmonics," *Ph.D. Thesis*, Yale University, New Haven, USA, 2004.

[7] A. Singer, R. Erban, I. Kevrekidis and R. Coifman: "Detecting Intrinsic Slow Variables in Stochastic Dynamical Systems by Anisotropic Diffusion Maps," *Proceedings of the National Academy of Sciences* (PNAS) 2009.

[8] Y. Bengio, J.-F. Paiement, and P. Vincent: "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and Spectral Clustering," *Advances in Neural Information Processing Systems, 16*, 2004.

[9] F. Lerdahl: *Tonal Pitch Space*. New York: Oxford University Press, 2001.

[10] H. Purwins, B. Blankertz, K. Obermayer: "Toroidal Models in Tonal Theory and Pitch-Class Analysis," Eleanor Selfridge-Field and Walter Hewlett, eds. *Computing in Musicology*, *Tonal Theory for the Digital Age*, Vol. 15, 73-98, 2008.

[11] Ö. İzmirli: "Cyclic Distance Patterns Among Spectra of Diatonic Sets: The Case of Instrument Sounds with Major and Minor Scales," Eleanor Selfridge-Field and Walter Hewlett, eds. *Computing in Musicology*, *Tonal Theory for the Digital Age*, Vol. 15, 11-23, 2008.

[12] T. Fujishima: "Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music," *Proceedings of the International Computer Music Conference* (ICMC), Beijing, China, 1999.

[13] Ö. İzmirli: "Template Based Key Finding From Audio," *Proceedings of the International Computer Music Conference* (ICMC), Barcelona, Spain, 2005.

[14] Ö. İzmirli: "An Algorithm for Audio Key Finding," *2005 Music Information Retrieval Evaluation eXchange (MIREX) Audio Key-Finding Contest*, www.music-ir.org/evaluation/mirex-results/articles/key_audio/izmirli.pdf, 2005.

[15] Ö. İzmirli: "Estimating the Tonalness of Transpositional Type Pitch-Class Sets Using Learned Tonal Key Spaces," *Proceedings of Mathematics and Computation in Music*, New Haven, USA, 2009.

EASY DOES IT:
THE ELECTRO-ACOUSTIC MUSIC ANALYSIS TOOLBOX

Tae Hong Park
Tulane University
park@tulane.edu

Zhiye Li
Tulane University
zli3@tulane.edu

Wen Wu
Tulane University
wwu1@tulane.edu

ABSTRACT

In this paper we present the EASY (Electro-Acoustic muSic analYsis) Toolbox software system for assisting electro-acoustic music analysis. The primary aims of the system are to present perceptually relevant features and audio descriptors via visual designs to gain more insight into electro-acoustic music works and provide easy-to-use "click-and-go" software interface paradigms for practical use of the system by non-experts and experts alike. The development of the EASY system exploits MIR techniques with particular emphasis on the electro-acoustic music repertoire – musical pieces that concentrate on timbral dimensions rather than traditional elements such as pitch, melody, harmony, and rhythm. The project was mainly inspired by the lack of software tools available for aiding electro-acoustic music analysis. The system's frameworks, feature analysis algorithms, along with the initial analyses of pieces are presented here.

1. INTRODUCTION

The idea for EASY Toolbox originated between 1999-2000 in the form of a master's thesis entitled "Salient Feature Extraction of Musical Instrument Signals" [11] which included a Java-based feature extraction and visualization software called Jafep (Java Feature Extraction Program). Since then, the project has somewhat been dormant, at least in direct relation to its original intention. Portions of the research evolved to automatic instrument recognition studies and further lead to the FMS software synthesis system [10] and most recently has developed into the EASY Toolbox to assist in the analysis of electro-acoustic music.

There is much interest and on-going research in MIR on various dimensions of music and a wealth of research can be found in pertinence to traditional music especially popular music. Some examples include rhythm analysis, melody analysis, tonality, traditional harmony, music recommendation, genre classification, instrument identification, and composer identification to name a few [2,16]. As far as MIR techniques and its applications in the area of music are concerned, much of the focus seems to be *outside* the realm of electro-acoustic music. One of the reasons for the scarcity in MIR-based research for electro-acoustic music may perhaps be attributed to the need for MIR researchers to be interested *and* actively be involved in composing or be deeply engaged in electro-acoustic music on a musical level. Another reason for this somewhat imbalance may be that the community seems to prioritize resources to the more standard musical repertoire that the general public accesses.

Some works related to the topic of music analysis software include Jafep (Java Feature Extraction Program), jAudio, Wavesurfer, Vivo, JRing, SoniXplorer, Sonic Visualizer [1, 3, 4, 6, 8, 11, 14], and others [17, 18]. Jafep is a Java-based feature extraction system for displaying feature vectors in a two-dimensional canvas and includes a harmonic follower designed mainly for analysis of musical instruments which is similar to jAudio. However, jAudio further concentrates on providing a feature extraction library/repository. Wavesurfer is a system for speech research and displays waveforms, pitch information, spectrograms, and formants. Vivo and JRing focus on pitch-based music, where JRing additionally deals with incorporating traditional scores for musicological studies. SoniXplorer is an interesting application which again primarily pays attention to traditional and popular music using self-organizing clustering algorithms. The Sonic Visualizer seems to be designed to address traditional music also, that is, pieces involving pitch, harmony, and rhythm. Although it has the ability to display audio features, perhaps due to the original design of the software architecture, when analyzing electro-acoustic music type signals, the visualization environment does not seem to be ideally suited for such situations. Marsyas (and to a lesser degree MIR Toolbox) is probably the most extensive environment for MIR research. It seems especially well suited for "DSP experts" and for the more experienced software developers/researchers but is perhaps not ideal for "users" who are looking for out-of-the-box software applications with simple and intuitive GUI interfaces as well as viewing capabilities – ready to use applications for specific purposes.

The EASY Toolbox is a modest initial step towards applying MIR theories with particular emphasis on electro-acoustic music focusing on its potential in gaining musical insights based on salient feature extraction techniques and clustering with the primary objective being that of an analytical tool wrapped with an intuitive GUI environment.

2. THE EASY TOOLBOX

2.1 Core Concept

One of the important characteristics of numerous electro-acoustic music, especially those pieces that are in the tape music genre, is that they are often concerned with aspects of timbre and sound color opposed to traditional musical elements such as pitch, harmony, and rhythm. However, although there are examples of software systems for analyzing "pitch-based music" as discussed in the introduction, there does not seem to be much of any software that is available for the analysis of music that do not adhere to those time-honored musical parameters. There is much software available for viewing raw waveforms and spectrograms but that type of information does not really offer too much insight by itself. Hence, our approach is to utilize salient feature extraction techniques as the basis for music analysis to uncover hidden information that is timbrally and perceptually relevant and perhaps even helpful in revealing additional data about a given work. We have also included segmentation/clustering algorithms using model-based and distance-based techniques. The algorithms that are implemented and used for displaying various features are hidden from the user as much as possible in order to render an easy-to-use interface. Furthermore, we have attempted to present the feature vectors in intuitive ways by plotting data in the time/frequency-domain and timbre spaces using 3D representation/navigation techniques. With a straightforward "click-and-go" environment provided by EASY, we hope that users will be encouraged to explore various timbral dimensions thereby help better understand sound objects and music.

2.2 EASY Features

2.2.1 The EASY Interface

The two main canvases in EASY are time-domain and frequency-domain displays as shown in Figure 1.

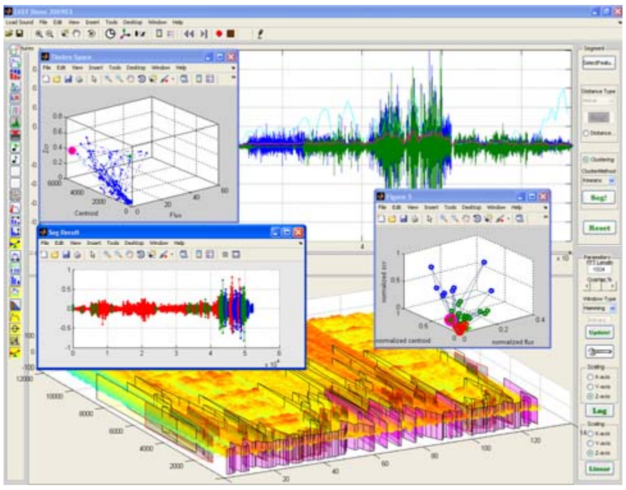


Figure 1. Screenshot of EASY

The approach of designing the EASY interface was driven by the aim of providing the user a 3D visualization environment for sonic exploration and interaction. For example, the waveforms for stereo files or multichannel files are presented in a cascading style along with the corresponding spectrogram.

The control areas of EASY include time/frequency-domain parametric control and feature selection for analysis/display. Standard functionalities such as zoom-in, zoom-out, 3D navigation/rotation, viewing options inherited from MATLAB®, the real-time input DAQ option (see Section 2.2.3), and a transport control are also included. Further controls are available for clustering and segmentation such as feature selection for clustering, number of clusters, and clustering algorithms as further discussed in Section 3.

2.2.2 EASY 3D Timbre Space Plots: the timbregram

EASY provides intuitive 3D timbre space representations adopted from [7] for sonic exploration which we call timbregrams. Figure 2 shows a timbregram example of a time-sequenced three instrument signal – bass guitar followed by clarinet and French horn with three timbre dimensions (spectral spread, spectral centroid, and spectral flux).

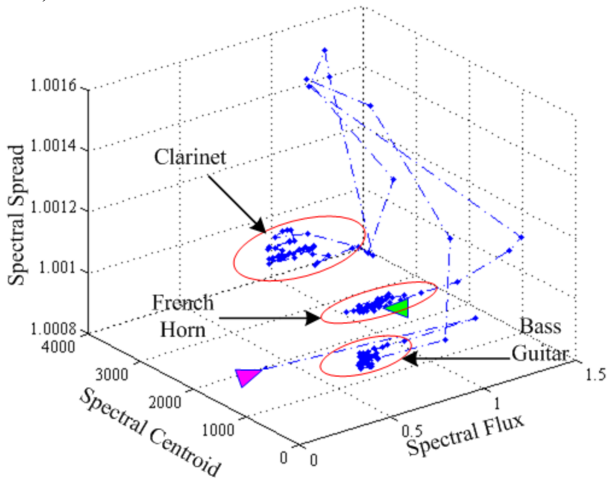


Figure 2. Timbre Space Example

The dots and dashed lines portray the 3D timbral trajectory as a function of time where the right pointing triangle refers to the beginning of the sample and the left pointing triangle the end of the sample. Each node represents a time unit equal to the frame/hop size. During audio playback, feature vector following occurs not only in the time-domain and frequency-domain canvases but also in the timbregram canvas itself (displayed in a separate window as shown in Fig. 1). This allows intuitive observation of sonic events via synchronization between the visuals and the audio that is played back.

2.2.3 "Real-Time" and MATLAB® Data Acquisition Toolbox

One of the advantages in using MATLAB® is the incredible resource of toolboxes available for data analysis and manipulation. One such example is the Data Acquisition (DAQ) Toolbox used for real-time analysis applications. The EASY system exploits the DAQ for analyzing and displaying input signals (mic/line input) in "real-time." It can display one or multiple features (selectable by the user) in the time and frequency-domain as well as the timbregram canvas.

2.3 EASY Algorithms

A total of 26 features in the time and frequency-domain are implemented in this current version of EASY – amplitude envelope, amplitude modulation, attack time, crest factor, dynamic tightness, frequency modulation, low energy ratio, noise content, pitch, release time, sound field vector, temporal centroid, zero-crossing rate, 3D spectral envelope, critical band, harmonic compression, harmonic expansion, inharmonicity, MFCC, modality/harmonicity/noisiness, spectral centroid, spectral flux, spectral jitter, spectral roll-off, spectral shimmer, and spectral smoothness. Many of the feature extraction algorithms themselves were developed in the FMS Toolbox [10, 12] and have been customized for use in EASY. Below, we present a short description of a select number of new features that we developed.

The dynamic tightness feature measures the quantized time-amplitude histogram on a frame-by-frame basis and provides insights into the "tightness" or "holiness" of the distribution of quantized sample values. This idea is shown in Figure 3 showing a highly compressed electric bass slide sample displaying a densely populated bed of samples throughout the amplitude axis bounded by the compressor threshold value.

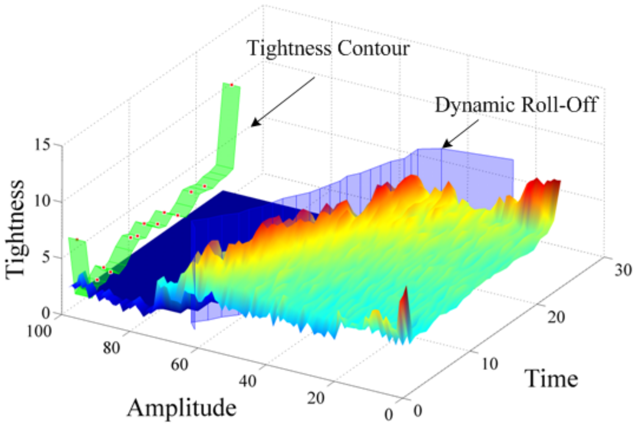


Figure 3. Electric Bass Slide: Compressed

Modality/harmonicity/noisiness is a method for analyzing a signal in terms of its harmonic, modal, and noise content. As shown in Figure 4, the harmonicity, modality, and noise floor levels of a signal are computed and displayed over time. One way of computing the harmonicity and modality is via the fundamental frequency (f_0) and the drift (e_k) in Hz. e_k is found by first determining spectral peaks, followed by computing their distances with respect to the closest ideal harmonic locations. The modality ("excessive inharmonicity") of each harmonic component can be then computed as the ratio of the drift and the fundamental frequency. As expressed in Equation (1) and (2), taking the mean of the inharmonicities of all the harmonics can be used to derive the modality of a signal.

$$Modality = \frac{e_1/f_0 + ... + e_k/f_0}{k} \quad (1)$$

$$Harmonicity = 1 - Modality \quad (2)$$

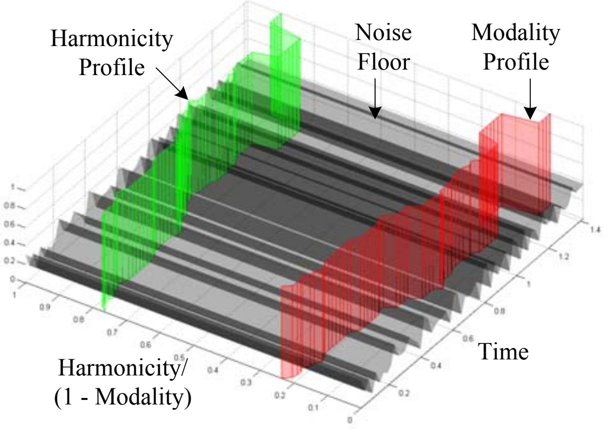


Figure 4. Modality/Harmonicity/Noisiness

The computation of the noise floor is based on sound flatness measure (SFM) – the ratio of the geometric mean and the arithmetic mean which has been used in speech research to extract voiced and unvoiced signals. When the signal is considered to be above the noise threshold (via SFM), the fundamental frequency is estimated which is then followed by modality analysis. On the other hand, if the signal's SFM value is determined to be below the noise threshold, it will be considered as noise. Another feature included in EASY is the multi-channel sound field vector developed by Travis Scharr while at Tulane University. This feature enables mutli-channel audio file display as a vector sum of the energy in each of the audio channels as a function of time.

3. SEGMENTATION ALGORITHMS

The two segmentation methods that we developed are based on clustering and distance measurement-based techniques as described in this section.

3.1 Model-based Segmentation: Clustering

The model-based approach for segmentation exploits a timbral feature vector clustering scheme. The audio input is first subjected to a silence detector followed by frame-by-frame feature extraction. The *N*-dimensional feature space is then piped to the clustering algorithm (eg. k-means). The clusters are then remapped to the time-domain in a color-coded fashion for visual clarity as shown in Figure 5.

3.2 Distance-based Segmentation

The distance-based segmentation algorithm applies statistical analysis of extracted features selectable by the user. The statistical analysis itself uses a long-term windowing scheme (main frames) to compute the average feature trajectory on a window-by-window (via sub-frames) basis – each sub-frame represents a single data point. Each main frame is then analyzed for its mean and standard deviation – the standard deviation is the distance measure used for segmentation. The distance can be computed via Euclidian distance, Kullback-Leibler distance, Bhattacharyya distance, Gish distance, Entropy loss or Mahalanobis distance.

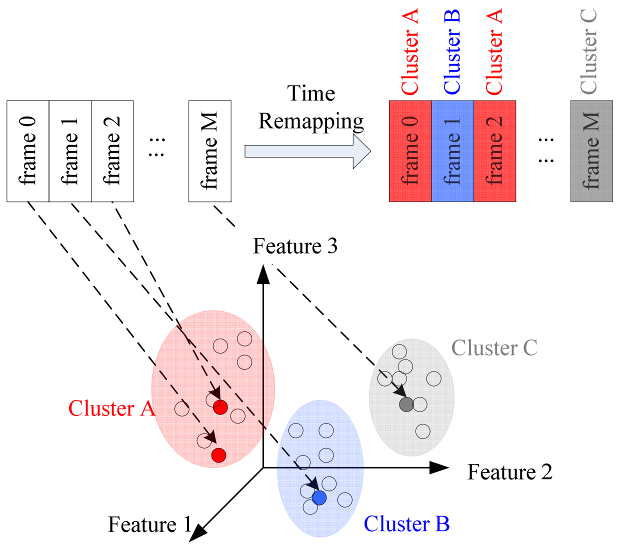


Figure 5. Time-Remapping in Clustering-based Segmentation for 3 Features

4. PREMLINARY ANALYSIS RESULTS

We used two pieces to conduct preliminary analysis of electro-acoustic works – *Machine Stops* (Tae Hong Park) and *Riverrun* (Barry Truax). We chose *Machine Stops* as we have first-hand detailed knowledge about the construction of the piece and *Riverrun* as it's not only an electro-acoustic masterpiece, but also because it is very much based on timbral compositional strategies.

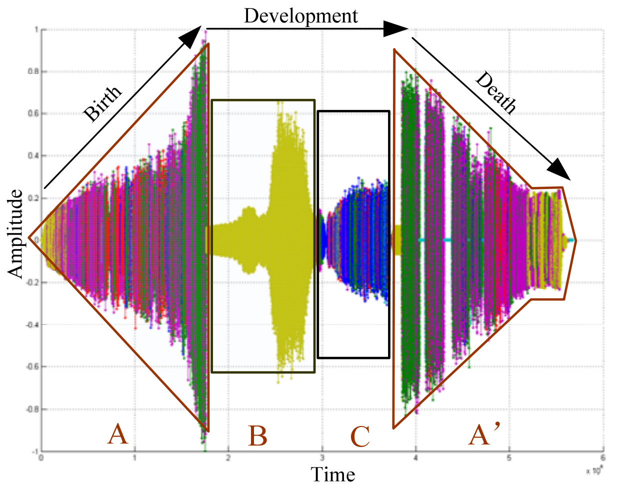


Figure 6. Segmentation Map of *Machine Stops*

A number of general observations could be made just by using single features such as modality/harmonicity/noisiness (MHN), dynamic tightness (DT), spectral centroid (SC), and the spectrogram (SG) itself. The extracted information included insights about where harmonic sections started and ended, where more modal sections occurred (via MHN), locating timbrally bright sounding parts (SC), exposing dynamically compressed areas (DT), and observing overall energy distributions and shifts (SG). However, what was most interesting in our initial analysis was discovering "segmentation maps," "timbregram trajectories," and "segmentation/cluster tracks" as shown in Figures 6, 7, and 8. Looking at the segmentation map we can generally identify four sections (A, B, C, A') via the color-coded segmentation regions and the amplitude envelope. The intro A (labeled as "birth") shows a triangular structure with a general build-up of energy. This is mirrored, slightly fragmented, in A' during the "death" phase of the piece which illustrates the overall arching shape of the piece itself. A' also includes an extended portion of the beginning part of the piece, adding a prolongation of decay towards the end (the "machine" coming to a "stop").

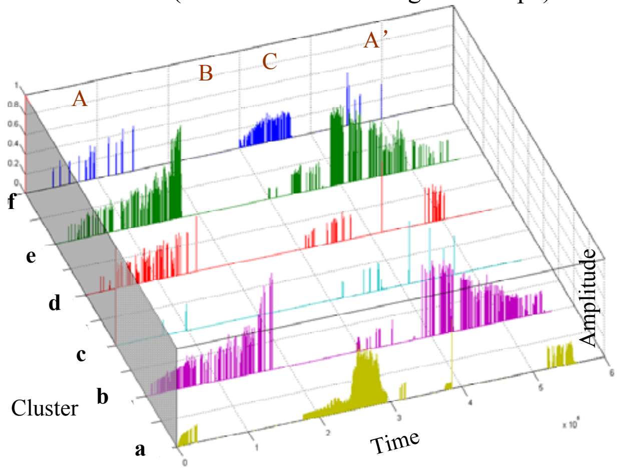


Figure 7. Segmentation/Cluster tracks (*Machine Stops*)

Figure 7 which displays the decomposition of the segmentation map into individual "cluster tracks," further exposes this build up and loss of energy of parts A and A' and also depicts the introduction of section C (cluster f) as new material (⑤ in Figure 8). Section B generally represents a sparse timbral construct exemplified by single and harmonically distorted sine-waves (in the HMN analysis plot, harmonicity is maximal in region B – not shown here).

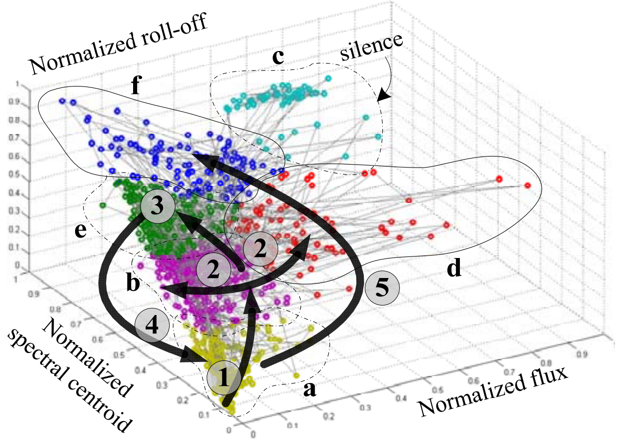


Figure 8. Timbregram Trajectory of *Machine Stops*

As shown in the timbregram plot (Figure 8) we can clearly view (when following the cursor during playback) the timbral trajectory which generally follows ① to ②, ③, ④, and ⑤ during the "birth" and "development" sections of the piece. The timbregram is also useful in displaying continuous timbral changes between cluster **a**, **b**, and **e** while also showing abrupt jumps in the timbre space between clusters **e** and **a** as well as **a** and **f**. The closing triangular portion follows the inverse trajectory ③ to ①.

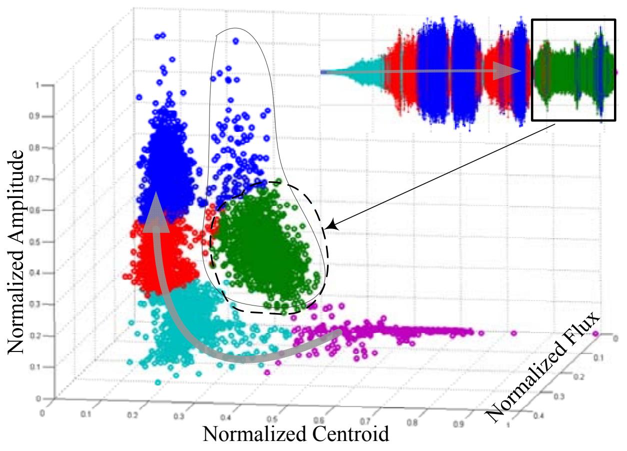


Figure 9. Timbregram and Segmentation Map of *Riverrun*

A similar analysis was conducted for *Riverrun* where we concentrated in particular on the segmentation map, cluster tracks, and timbgregram. It was quite straightforward to identify sectional divisions in the spectrogram as expected, but what was particularly interesting in the timbrgram was the finding that, unlike in *Machine Stops*, the colonization of the timbre space portrayed a distinct separation of one particular cluster from the rest – the timbral cluster pertaining to the closing section of the piece with high spectral centroid as shown in Figure 9. At the same time, the continuous development as described in [13] of *Riverrun* can also be clearly seen in Figure 91 beginning from a sparse quiet group of droplets, developing to rivulets, streams, and massive oceans towards the main part of the piece. Various feature sets have been employed in generating clusters, segmentation maps, and timbregrams. Interestingly enough, for the majority of the cases, the ensuing results have been quite similar when interpreting the various plots. The shapes, however, at times looked quite different in the timbregrams for example, but the overall timbral trajectories usually gravitated to the same conclusions. The same was also true when changing the number of clusters. In general, more clusters gave finer detail in grouping subtleties in the timbre space, whereas smaller number of clusters merged closely spaced clusters into a "supercluster." This is evident in Figure 9, where the ultimate section of the piece becomes one large cluster extending vertically (amplitude) when employing 5 clusters.

5. SUMMARY AND FUTURE WORK

5.1 Summary

In this paper we presented a new software system for assisting analysis of electro-acoustic music with particular emphasis on timbre. We described the functionalities of the toolbox, some of the feature extraction algorithms, the timbre space display interface, real-time possibilities using EASY, conducted preliminary analysis of two musical examples, and discussed pattern recognition modules to help reveal structural elements of an audio signal. The system has been designed with ease of use in mind by providing a "click-and-go" interface while at the same time offering advanced options for more detailed parametric control.

5.2 Future Work

The current version of the EASY Toolbox already includes 26 features but we foresee that more features, especially those that are specific to electro-acoustic music will be encountered in the future as we further develop this system. To facilitate adding new features we plan on providing a template for third party development. We plan to further extensively test and use the EASY Toolbox for analyzing a number of classic electro-acoustic works and expect to report our findings in the near future.

One very interesting and potentially exciting area that could provide promising application for EASY is exploiting more pattern recognition techniques on feature vectors to analyze for "horizontal" and "vertical" relationships and correlations in a given audio signal. That is, analyzing and displaying feature trajectories and patterns not only by comparing frames as one unit but also analyzing the vertical relationships vs. time as shown in Figure 9.
This could be very useful in displaying detailed relationships between frames, sections, motifs, formal structures, referential cues, and many other patterns that can provide insights into the music under scrutiny. One way of implementing such a feature would be using labels to display various icons in the time/frequency-domain canvases and timbregram, which will further allow for annotation possibilities.

Another area that we are interested in exploring is the literature concerning cognitive studies especially those that are related to mood and sound [5, 15]. We are not explicitly interested in measuring mood per se but we would also like to examine other angles to help extract perceptual and cognitive dimensions from the music that is being analyzed.

On top of providing analysis results from feature vectors, we also plan on offering supplementary cultural information acquired from the Internet via search engines and online digital libraries. One approach is using search strings as implemented in jWebMiner [9], which is a software package for extracting cultural features from the web using hit counts. Current MIR technologies such as fingerprinting, artist identification, and genre classification are used for automatically recommending similar musical styles, composers, and artists. Although these technologies have not been specifically applied to "music analysis" software systems that we know of, we foresee great potential in incorporating and exploiting such technologies not just for electro-acoustic music alone, but also for musical research, musicological studies, pedagogy, and composition in general. It is not difficult to imagine being able to have easy access to supplementary information such as scores, program notes, composer/performer/"machine" biographical information, graphics/pictures/videos, or any other related materials/media at one's fingertips and at the click of one button.

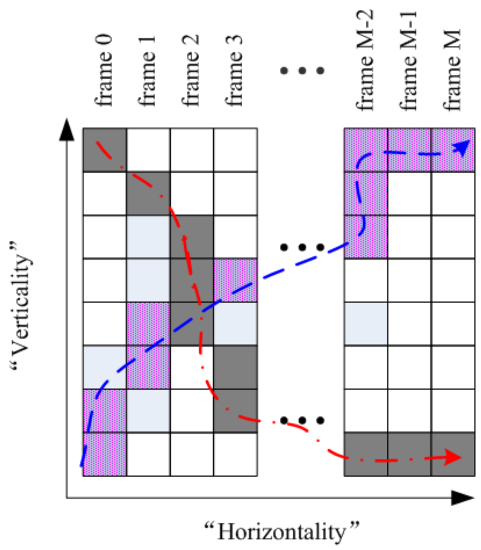


Figure 10. Verticality AND horizontality

Although the current software version is already a stand-alone MATLAB® application and can run on any machine that has the MATALB® run-time library, we plan on porting it to faster and more efficient compiler-based platforms like Cocoa.

6. REFERENCES

[1] Cannam, C., Landone C., Sandler M., Bello J., "The Sonic Visualizer: A Visualization Platform for Semantic Descriptors", *Proceedings of the International Conference on Music Information Retrieval 2006*, Victoria Canada

[2] Cao, C., Li M., Liu J., Yan, Y., "Singing Melody Extraction in Polyphonic Music by Harmonic Tracking", *Proceedings of the International Conference on Music Information Retrieval 2007*, Vienna, Austria.

[3] Kornstädt, A., "The JRing System for Computer-Assisted Musicological Analysis", *Proceedings of the International Conference on Music Information Retrieval 2001*, Indiana, USA

[4] Kuuskankare, M., Laurson, M., "Vivo - Visualizing Harmonic Progressions and Voice-Leading in PWGL", *Proceedings of the International Conference on Music Information Retrieval 2007*, Vienna, Austria.

[5] Li, T., Ogihara, M., "Detecting emotion in music", *Proceedings of the International Conference on Music Information Retrieval 2003*, Washington D.C., USA

[6] Lubbers, D., "SoniXplorer: Combining Visualization and Auralization for Content-Based Exploration of Music Collections", *Proceedings of the International Conference on Music Information Retrieval 2005*, London, U.K.

[7] McAdams, S., Winsberg, S., Donnadieu, S., De Soete, G., and Krimphoff, J. 1995. *Perceptual Scaling of Synthesized Musical Timbres: Common Dimensions, Specificities, and Latent Subject Classes*, Psychological Research 58, 177 - 192.

[8] McEnnis D., McKay C., Fujinaga I., Depalle P., "jAudio: An Feature Extraction Library", *Proceedings of the International Conference on Music Information Retrieval 2005*, London, U.K.

[9] McKay, C., Fujinaga, I., "jWebMiner: A Web-based Feature Extractor", *Proceedings of the International Conference on Music Information Retrieval 2007*, Vienna, Austria.

[10] Park T. H., Biguenet J., Li Z., Richardson C., Scharr T., "Feature Modulation Synthesis", *Proceedings of the International Computer Music Conference 2007*, August, 2007, Copenhagen, Denmark.

[11] Park, T. H. *"Salient Feature Extraction of Musical Instrument Signals"*, Dartmouth College, M.A. Dissertation, 2000.

[12] Park, T. H., Z. Li, Biguenet J., "Not Just More FMS: Taking It To The Next Level", Proceedings of the 2008 ICMC, Belfast, Ireland.

[13] Simoni, M., "Analytical Methods of Electroacoustic Music", Routledge, 2006, Ch. 8, pp. 187 – 238.

[14] Sjölander, K., Beskow, J., "Wavesurfer – An Open Source Speech Tool", *Proceedings of ICSLP 2000*, Beijing, China

[15] Thayer, R.E. 1989. *The Biopsychology of Mood and Arousal*. New York: Oxford University Press

[16] Tzanetakis, G., Essel, G. Cook, P., "Musical genre classification of audio signals", *Proceedings of ISMIR 2001*, Indiana, USA

[17] Tzanetakis, G., Cook, P., "MARSYAS: a framework for audio analysis", *Organised Sound*, Vol. 4 , Issue 3, 1999, Cambridge University Press

[18] Lartillot, O., Toiviainen, P. "A MATLAB TOOLBOX FOR MUSICAL FEATURE EXTRACTION FROM AUDIO", *Proceedings of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, Bordeaux, France.

MIR IN ENP – RULE-BASED MUSIC INFORMATION RETRIEVAL FROM SYMBOLIC MUSIC NOTATION

Mika Kuuskankare
Sibelius Academy
Centre for Music and Technology
mkuuskan@siba.fi

Mikael Laurson
Sibelius Academy
Centre for Music and Technology
laurson@siba.fi

ABSTRACT

Symbolic music information retrieval is one of the most underrepresented areas in the field of MIR. Here, symbolic music means common practice music notation–the musician readable format. In this paper we introduce a novel rule-based symbolic music retrieval mechanism. The Scripting system–ENP-Script–is augmented with MIR functionality. It allows us to perform sophisticated retrieval operations on symbolic musical scores prepared with the help of the music notation system ENP.

We will also give a special attention to visualization of the query results. All the statistical queries, such as histograms, are visualized with the help of common music notation where appropriate. N-grams and more complex queries–the ones dealing with voice leading, for example–are visualized directly in the score.

Our aim is to demonstrate the power and expressivity of the combination of common music notation and a rule-based scripting language through several challenging examples.

1. BACKGROUND

Music (especially Classical music) is primarily a written tradition. Throughout the centuries musical compositions have been preserved in music notation. It is the most complete and widespread method that we know of for notating the complex and interrelated properties of a musical sound: pitch, intensity, time, timbre, and pace. [1] Common music notation is also an invaluable tool in the field of music information retrieval. In this paper we introduce a symbolic music retrieval mechanism based on a scripting language called ENP-Script [2] and music notation system ENP [3].

ENP-script is a rule-based object oriented scripting language that is here augmented with MIR functionality. The extensions allow the user to perform retrieval operations on scores prepared with the help of ENP and visualize the results in a meaningful way. ENP-Script allows us to define complex and musically relevant queries using it's pattern-matching language. It has a uniform and simple syntax. The structural elements of the score (e.g., notes, beats, measures, melodies, harmony, voice-leading, etc.) are accessed using a symbolic naming scheme where a collection of reserved keywords is used to denote the objects of interest.

On the score level the retrieval system is based on ENP's underlying music representation. ENP provides several interesting features in terms of the present application: (1) it can be used to store music using a wide range of notational styles (Western musical notation roughly from 17th century onward, including 20th century notation); (2) it can be used as a user-interface component allowing us to construct both eye-catching and functional visualizations of MIR data; (3) it provides access to its notational data structures, allowing the user to inspect the properties of the notational objects (e.g., time, pitch, duration); and (4) it provides a rich library of standard and user-definable expressions allowing us to annotate the score with analytical information [4]. One further detail of interest is that ENP scores can be written using both mensural (metric) and non-mensural (piano roll like) notation. The system described in this paper works without modifications on both types of notation. This makes it possible to use this system for applications dealing with contemporary music.

In this paper, we will also give a special attention to visualization. All the statistical queries, such as histograms, are visualized with the help of common music notation where appropriate. N-grams and more complex queries are visualized directly in the score. Both approaches allow us to associate the query results directly with the correct musical objects.

A few approaches have been introduced where the aim is to use some kind of symbolic notation as the basic for MIR queries (see, e.g., [5–13]. Most of these systems, however, primarily address large databases of music encoded in an array of formats, such as MIDI, Kern, MusicXML [14], etc. Most of the approaches can be seen as MIR tool chains comprising of several small or even larger utilities chained together. Humdrum is an example of such a system. In [13] Humdrum is even married with Perl [15] and LilyPond [16] to create kind of a meta tool chain. Other approaches use various music formats like GUIDO [17] in [6, 8, 9]; Kern in [5, 8]; and MusicXML in [18].

At the moment the presented retrieval system can be seen as an Analytic/Production MIR System [19]. We concentrate predominantly on posing questions on a symbolic musical score rather than retrieving information from a large music database. Querying a database of all Bach Chorales, for example, is however not out of the scope of the present approach.

Our retrieval system is part of PWGL [20] which is freely available for Macintosh OS X (Intel and PPC versions) and for Windows at `www.siba.fi/PWGL`.

The rest of the paper is organized as follows. Section 2 illustrates how a collection of archetypal MIR assignments can be solved using our retrieval system. The examples are divided roughly into two categories: statistical and analytical. We end the paper with some concluding remarks and outlines for future work.

2. EXAMPLES

While mass queries can tell certain kinds of facts about the music in the database our approach emphasizes the musical meaningfulness of the query. On the one hand one might find out that Bach violated X times the rule Y in Z chorals. On the other hand one might want to reveal these cases in a musical score and study why this may have happened. To be able to do so requires a flexible searching mechanism and flexible notational and visualization tools.

In this section we present a collection of examples based on more or less archetypal MIR assignments. The section is divided roughly in two parts. In the first part we concentrate on statistical queries and more or less traditional visualizations. The second part, in turn, turns focus on more analytical queries and visualizes the results in the score. Altogether, we will address several subjects, e.g., histograms, counting, pitch-class set theory, rhythm, etc. Some of the case studies borrow shamelessly from the Humdrum example database presented at `http://music-cog.ohio-state.edu/Humdrum/`.

Each subsection is accompanied with a code example and potentially also a visualization of the result. It is not in the scope of this paper to give an exhaustive review of the Scripting syntax. The code examples require a little knowledge about Lisp programming language but they should be clear and simple enough to be followed by anyone with some background in programming.

Apart form the examples presented in this paper, many other types of queries could easily made with the current system including those about pitch-class set theory, voice-leading, word painting, harmonic analysis, etc.

The most important points of interest in the following examples are: (1) the terseness and expressivity of the query definitions, (2) the descriptiveness of the visualization, and (3) the overall versatility of the system.

2.1 Statistical Queries

2.1.1 Histograms

One of the prototypical MIR tasks is the histogram.

The musical score used as a starting point for Examples 1–4 is the guitar transcription by Andrés Segovia of the *Tango op. 156a* by Isaac Albéniz. The beginning of the score is shown in page 6 (Figure 6).

Example 1 shows the retrieval rule to generate a pitch histogram of the given score. The pattern given in line 1 means that the rule applies to every note object in the score, thus the traditional wild card $\boxed{\texttt{*}}$. $\boxed{\texttt{?1}}$ is a variable to which every note in the score is bound one by one. `histogram` is a special MIR function that takes care of gathering the values and visualizing the result.

After the execution of the script the pertinent visualization method is called to generate the pitch histogram shown in Figure 1. Instead of the traditional horizontal bar graph we use here a vertical arrangement instead. The histogram values are also shown against a set of piano keys to give a better idea of the register (middle-C is highlighted using a shade of grey).

One special aspect of this particular histogram (including the ones shown in Figures 2 and 3) is that the result can be played back. Either as a whole or by selecting a subset of the result shown. Especially, in case of tonal music, an aural examination of the pitch histogram could among other things reveal potential problems in the integrity of the source material.

Furthermore, as the histogram is realized with the help of ENP, it itself can be scripted to select and highlight pitches above certain threshold, for example.

Example 1 An ENP-script collecting histogram values from a score.

```
1 (* ?1
2    (?if
3     (histogram :value (m ?1))))
```

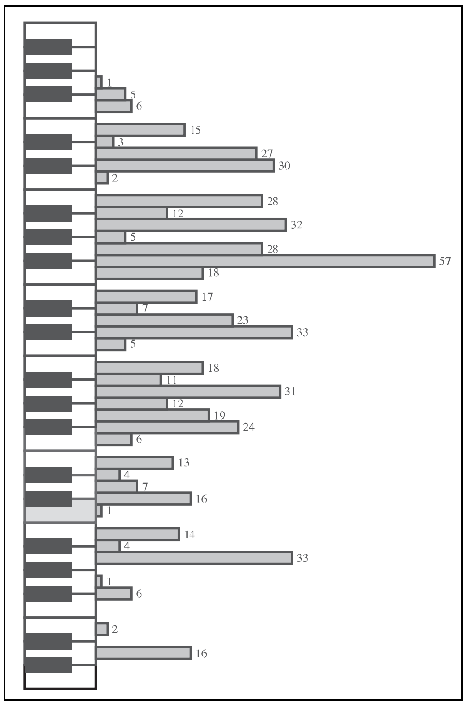


Figure 1. A traditional pitch histogram plotted using ENP as a visualization tool (Albéniz: Tango op. 156a).

2.1.2 *Harmony Histogram*

In addition to horizontal events (i.e., melody, as in Example 1) with ENP-Script it is also possible to access the vertical (harmonic) dimension of the score.

Scores are partitioned by 'harmonic slices' (resembling 'moments' in [5]). A harmonic slice is a vertical entity defined as a point in time when any note event begins or ends in any part. This structural component allows us to perform queries involving simultaneity.

Example 2 shows a script accessing the harmonic slices to produce a 'moment' histogram. This can be accomplished simply by introducing the keyword `:harmony` in the pattern matching part (see line 1). This instructs the script to access all the vertical elements of the score instead of the horizontal ones. Compared to Example 1 the change is minimal but the effect is dramatic.

Another point of interest is the form `(m ?1 :complete? t)` given at the end of line 1. This is in fact a condition and it is used here due to the implementation of the scripting engine and cannot be explained in-depth in the score of this paper. Suffice to say that in addition to accessing the total harmony (i.e., all the notes sounding at a given point in time) we can also access partial harmonic formations (e.g., subsets of the sounding harmony). As we are here interested only in the total harmonies hence the condition.

Furthermore, line 2 introduces yet another additional condition to skip any grace notes that are abundant in our example score. In line 4 we simply record the pitch values of the harmony given by `(m ?1)` as a list of midi values, e.g., (60 64 67).

Example 2 A script collecting histogram values of type harmony.

```
(* ?1 :harmony (m ?1 :complete? t)
   (?if (unless (some #'grace-note-p
                      (m ?1 :object t))
          (histogram :value (m ?1)))))
```

The result is shown as a histogram with the relevant parts visualized using common practice notation. In the analysis same kind of chords are grouped together irrespective of register, or pitch spelling. That means that, e.g., all major chords are identified as equal (i.e., pitch-class set 3-11B).[1] Furthermore, in the histogram the so called prime form is shown. This is why, for example, the second to last entry, the seventh chord, is displayed in first inversion.

2.1.3 *Counting*

Our next example demonstrates the ability of the scripting mechanism to access only parts of the given score. Example 3 shows a simple counting script where we count the number of events in the score. Here, instead of counting all the events we restrict the search to measures 8–16 (see line

[1] The system used here is similar to that of Forte except that the letters A or B are added to distinguish between two inversionally related transpositional set-classes, e.g., 3-11A is the minor triad, and 3-11B is its inversion, the major triad.

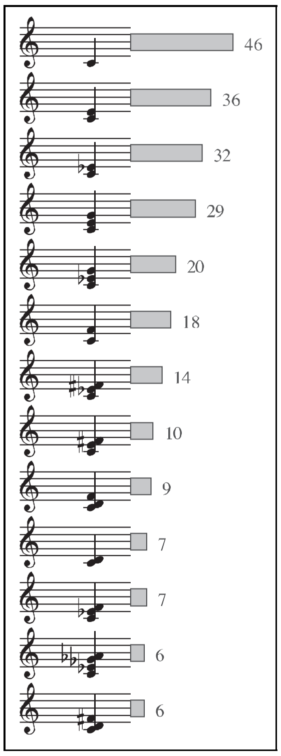


Figure 2. A 'moment' histogram (Albéniz: Tango op. 156a).

1). Our analysis reveals that there are 96 events combined in the measures 8–16 in the Tango by Albéniz. `quantity` in line 3 simply increments a counter when presented with a new event. Note, that this script counts all *note* events. We could similarly count all *chords* by inserting the keyword `:chord` after the variable `?1`.

Example 3 A script counting the number of events in a score from measure 8 through 16.

```
(* ?1 :measures (8_16)
  (?if
   (quantity :value ?1)))
```

For comparison we give the following Unix script in Humdrum performing the equivalent operation:

```
yank -n = -r 8-16 Tango | census -k
```

2.1.4 *Rhythmic Patterns*

Next, we define a search based on the rhythmic dimension of the score. We aim to determine the most common rhythmic pattern spanning a measure. Example 4 shows the retrieval script. Once again our script has changed relatively little. The keyword `:measure` shown in the first line denotes that we want to access measure objects this time. Thus, the variable `?1` is here bound in succession to every measure object found in the score. As we want to access the whole rhythmic entity inside the measure we add once again the condition `(m ?1 :complete? t)`. This ensures that the body of the script is executed only when the rhythm for the entire measure is known. In line 4 we read the rhythmic definition cached by the system in a list form.

Example 4 A script for determining the most common rhythmic pattern spanning a measure.

```
(* ?1 :measure (m ?1 :complete? t)
   (?if
    (histogram :value
         (read-key ?1 :rtm-pattern))))
```

We define here the rhythm histogram again using ENP and common music notation as it allows us to visualize the data in a straightforward manner. Figure 3 shows a part of the rhythm histogram displaying the actual rhythms in rhythm notation and the corresponding values as a bar graph.

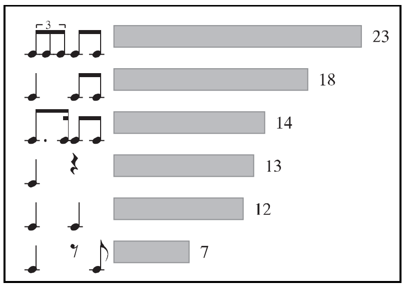


Figure 3. A per measure rhythm histogram (Tango op. 156a by Albéniz).

The above is certainly more descriptive than, let's say, producing print-outs like this:

```
((23 ((1 (1 1 1)) (1 (1 1))))
 (18 ((1 (1)) (1 (1 1)))) ...)
```

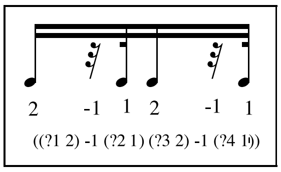
2.2 Analytical Queries

2.2.1 Rhythmic Patterns

Another kind of rhythmic query is presented in Example 5. Here, instead of counting all the possible rhythms we restrict our search to certain kind of rhythmic pattern. Furthermore, we have chosen to visualize the result of the query directly in the score. Figure 4 shows all the occurrences of our rhythmic pattern enclosed inside rectangles (drawn in red color in the original score). To save space we show here only the right-hand melody of the original composition (measures 1–8 of Humoresque in G♭ major by Antonin Dvořák).

The retrieval script is now a bit more complex than in the previous cases. The pattern matching part reads `(* ?1 ?2 ?3 ?4` as in this script we are now interested in the rhythmic formation between four consecutive notes. `match-rtm?` in line 3 is a special function that is used match the score rhythms against a given pattern.

Deciphering the rhythm matching syntax can be at first quite challenging and it requires some knowledge about the internal representation of ENP (see, e.g., [21]). We cannot give a comprehensive review of the format here. However, the line 5 defines a pattern where an event having a duration of 2 units is followed by a rest (a negative number) with a duration of 1 unit and another event with the duration of 1 unit. The durations are relative instead of absolute. The aforementioned pattern is repeated twice in row 5. This description is translated to the following rhythmic pattern:



Example 5 A script searching for a given four-note rhythmical pattern in a score.

```
1 (* ?1 ?2 ?3 ?4
2    (?if
3     (when (match-rtm?
4            (1
5             ((?1 2) -1 (?2 1) (?3 2) -1 (?4 1))))
6       (add-expression 'score-expression
7                       ?1 ?2 ?3 ?4
8                       :color :red))))
```

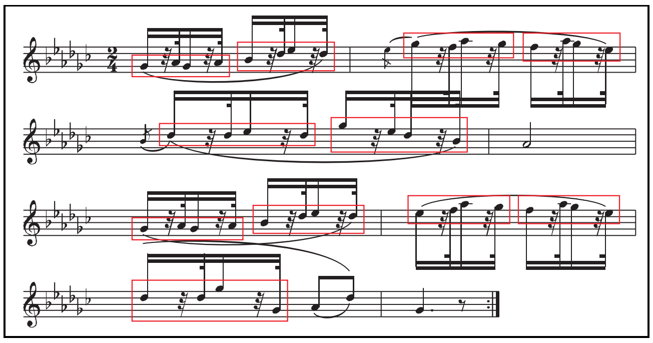


Figure 4. A specific rhythmic pattern visualized in the score (Humoresque in G♭ major by Antonin Dvořák).

2.2.2 N-grams

N-grams have been used extensively in MIR in both monophonic and polyphonic contexts (see, e.g., [19, 22]). In our next example we will show how to represent n-grams with the scripting language and how to mark them in a score. Naturally, instead of marking the n-grams in the score we could have recorded the statistical distribution of n-grams and display them in the same manner as displayed in Figure 2.

Example 6 shows the script definition. This particular script is demonstrating yet another interesting ability of the scripting system. Here, instead of defining a fixed pattern, as in the previous examples, we write the script in a dynamic fashion. The pattern matching part becomes now quite minimalistic again. The complexity of the search lies inside the script definition. In order to represent any n-gram the user is require to change only the list of context lengths enumerated in line 3. Here, we use a list `(2 3)` denoting di- and tri-grams respectively. The body of the script from line 4 onwards is written so that any sized n-grams can be found and visualized. In line 6 we add an ENP expression in the score displaying the extent of the n-gram and also the sizes of the consecutive intervals as can be seen in Figure 5.

3. DISCUSSION

Currently this retrieval system is not suitable for very large corpus of data. The mechanism used here requires that the whole score is read in the memory and that all the musical objects are instantiated. One could possibly provide an alternative loading mechanism that creates only the necessary data structures needed for the scripting language to operate. This would most likely guarantee much more shorter loading times and in turn facilitate larger searches.

Our system needs to support more input formats. Not only MIDI and the native ENP file formats but also at least MusicXML and perhaps even Kern.

However, at its present state the system is capable of performing very sophisticated and complex queries. Also the visualization capabilities are second to none. The ability to be able to mix common music notation with statistical graphics is also beneficial and allows us to represent the query result in a musician readable way.

The presented notational front-end combined with our scripting engine allows us to query, annotate and analyze musical scores and visualize the results in a manner probably not matched by any other software package.

4. ACKNOWLEDGMENTS

The work of Mika Kuuskankare and Mikael Laurson has been supported by the Academy of Finland (SA 114116 and SA 105557).

5. REFERENCES

[1] Curtis Roads. *The Computer Music Tutorial*. The MIT Press, Cambridge, Massachusetts, London, England, 1996.

[2] Mika Kuuskankare and Mikael Laurson. Intelligent Scripting in ENP using PWConstraints. In *Proceedings of International Computer Music Conference*, pages 684–687, Miami, USA, 2004.

[3] Mika Kuuskankare and Mikael Laurson. Expressive Notation Package. *Computer Music Journal*, 30(4):67–79, 2006.

[4] Mika Kuuskankare and Mikael Laurson. Annotating Musical Scores in ENP. In *International Symposium on Music Information Retrieval*, London, UK, 2005.

[5] Michael Droettboom. Expressive and efficient retrieval of symbolic musical data. In *International Symposium on Music Information Retrieval*, 2001.

[6] Andreas Kornstädt. The jring system for computer-assisted musicological analysis. In *ISMIR*, 2001.

[7] Shyamala Doraisamy. An approach towards a polyphonic music retrieval system. In *International Symposium on Music Information Retrieval*, 2001.

[8] Matthew J. Dovey. A technique for regular expression style searching in polyphonic music. In *International Symposium on Music Information Retrieval*, 2001.

[9] Holger H. Hoos, Kai Renz, and Marko Görg. GUIDO/MIR — an experimental musical information retrieval system based on GUIDO music notation. In *International Symposium on Music Information Retrieval*, 2001.

[10] Donncha Ó Maidín and Margaret Cahill. Score Processing for MIR. *International Symposium on Music Information Retrieval*, pages 59–64, 2001.

[11] Goffredo Haus, Maurizio Longari, and Emanuele Pollastri. A score-driven approach to music information retrieval. *J. Am. Soc. Inf. Sci. Technol.*, 55(12):1045–1052, 2004.

[12] David Huron. Music information processing using the humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):15–30, 2002.

[13] Ian Knopke. The perlhumdrum and perllilypond toolkits for symbolic music information retrieval. In *International Symposium on Music Information Retrieval*, pages 147–152, 2008.

[14] M. Good and G. Actor. Using MusicXML for File Interchange. In *Third International Conference on WEB Delivering of Music*, page 153, Los Alamitos, CA, 2003. IEEE Press.

[15] Wikipedia. Perl — wikipedia, the free encyclopedia, 2009. [Online; accessed 8-May-2009].

[16] Han-Wen Nienhuys and Jan Nieuwenhuizen. LilyPond, a system for automated music engraving. In *XIV Colloquium on Musical Informatics (XIV CIM 2003)*, Firenze, Italy, 2003.

[17] H. H. Hoos, K. A. Hamel, K. Renz, and J. Kilian. The GUIDO Music Notation Format - A Novel Approach for Adequately Representing Score-level Music. In *Proceedings of International Computer Music Conference*, pages 451–454, San Francisco, 1998.

[18] Goffredo Haus and Alberto Pinto. Mx structural metadata as mir tools. In *Sound and Music Computing '05*, 2005.

[19] J. Stephen Downie. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-grams as Text*. PhD thesis, University of Western Ontario, 1999.

[20] Mikael Laurson, Mika Kuuskankare, and Vesa Norilo. An Overview of PWGL, a Visual Programming Environment for Music. *Computer Music Journal*, 33(1), 2009.

[21] Mika Kuuskankare and Mikael Laurson. Recent Developments in ENP-score-notation. In *Sound and Music Computing '04*, Paris, France, 2004.

[22] Shyamala Doraisamy. *Polyphonic Music Retrieval: The N-gram Approach*. PhD thesis, University of London, 2004.

Example 6 A script to visualize n-grams directly in an ENP score. Here, di- and tri-grams are shown. Simply by editing the parameter list shown in line 3 (n-grams) it is possible to visualize n-grams of any size. No other changes are necessary.

```
1 (* ?1
2    (?if
3     (let ((n-grams '(2 3)))
4       (iter (for n-gram in n-grams)
5             (?incase-let (intervals (m ?1 :L (1+ n-gram) :data-access :int :complete? t))
6               (add-expression 'group (m ?1 :L (1+ n-gram) :object t)
7                               :kind :bracket-at-end
8                               :info (format () "~{~3,@d ~^|~}"
9                                             intervals)))))))
```

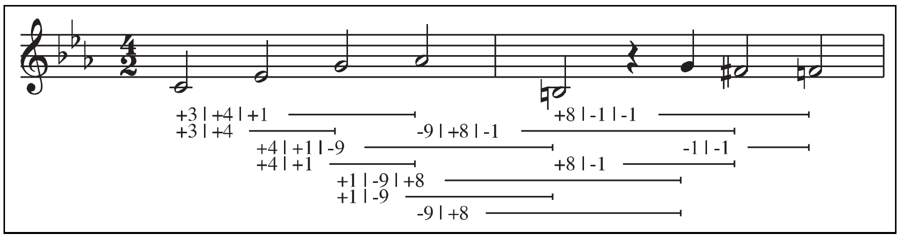


Figure 5. N-grams visualized directly in the score using a dynamically adapting script.

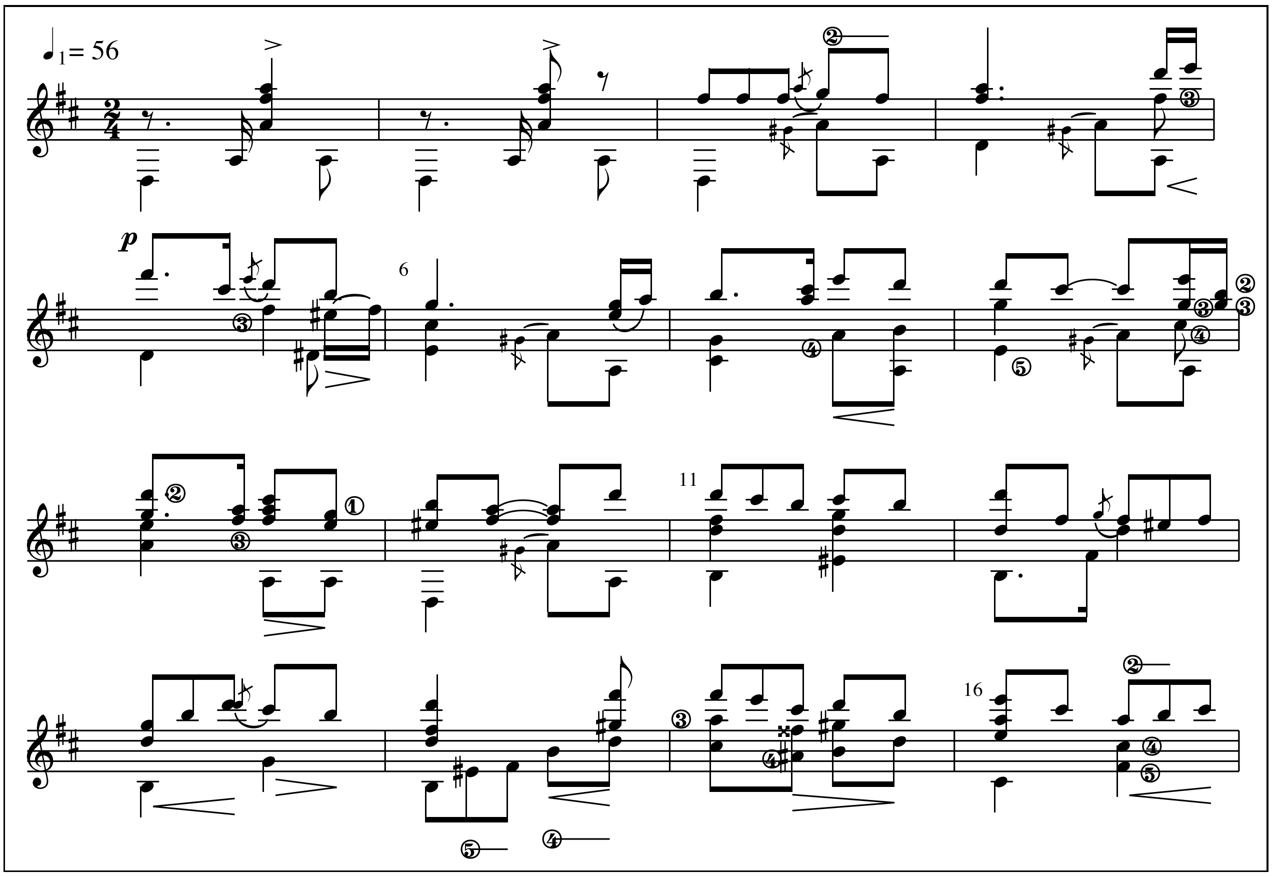


Figure 6. Tango op. 156a by Isaac Albéniz notated with ENP (transcribed for Guitar by Andrés Segovia).

AN INTEGRATED APPROACH TO MUSIC BOUNDARY DETECTION

Min-Yian Su, Yi-Hsuan Yang, Yu-Ching Lin, Homer Chen
National Taiwan University
sui751004@gmail.com, affige@gmail.com, vagante@gmail.com, homer@cc.ee.ntu.edu.tw

ABSTRACT

Music boundary detection is a fundamental step of music analysis and summarization. Existing works use either unsupervised or supervised methodologies to detect boundary. In this paper, we propose an integrated approach that takes advantage of both methodologies. In particular, a graph-theoretic approach is proposed to fuse the results of an unsupervised model and a supervised one by the knowledge of the typical length of a music section. To further improve accuracy, a number of novel mid-level features are developed and incorporated to the boundary detection framework. Evaluation result on the RWC dataset shows the effectiveness of the proposed approach.

1. INTRODUCTION

Popular songs usually comprise several music sections such as intro, verse, chorus, bridge and outro. A music boundary is the time point where a section transits to another. Identifying such boundaries is important because it allows us to divide a song into semantically meaningful sections. This information can also be applied to music summarization [1] and thumbnailing [2] to facilitate music browsing and structure-aware playback [3]. Boundary detection also serves as a front-end processor for music content analysis since it provides a local description of each section rather than a global but coarse representation of the whole song [5].

Although there is a rich literature in music theory about music structure analysis for symbolic music (e.g. [20]), music boundary detection for music signals is still a challenging task because precise pitch detection in polyphonic music is not yet achievable. Under this condition, most work on music boundary detection utilizes the similarity between short-term (e.g., 23ms) audio frames within a song to identify the repetitive parts and divide a song into a number of sections [1–3, 6–8]. A more recent work formulates boundary detection as a clustering problem and considers that the audio frames of each cluster belong to the same music section [9].

The accuracy of this *unsupervised* approach, however, may be limited because only the information of a song itself is exploited. For example, identifying repetitive parts cannot correctly identify the boundary between two adjacent music sections that always occur successively in a song. On the other hand, clustering-based methods tend to produce over-segmented results if the acoustic property of the frames in a music section varies greatly. Using histograms to gather statistic of spectral characteristics of neighboring audio frames [9] does not necessarily solve the problem because the histograms of two adjacent frames are usually similar, making boundary detection even more difficult.

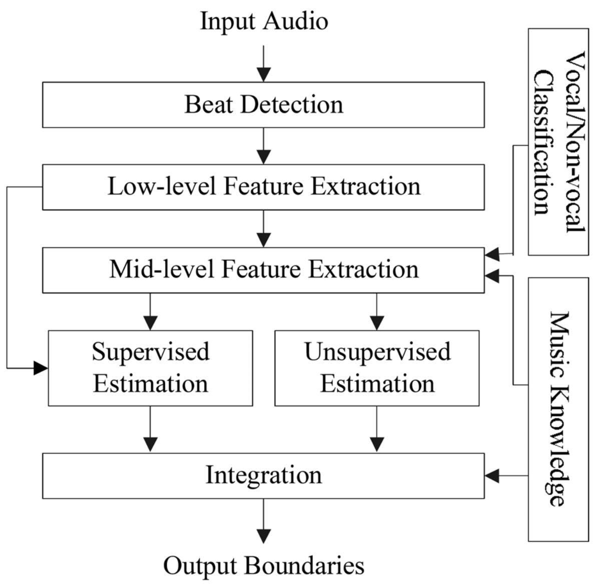


Figure 1. A schematic diagram of the proposed music boundary detection system.

To address the aforementioned drawbacks, Turnbull *et al* formulate music boundary detection as a supervised problem and train a binary classifier to classify whether a time point is a boundary or not [10]. In this way, we can mine more information from a large number of training songs and identify features that are relevant to boundary detection.

However, because a supervised system is pre-trained by using the training data and fixed afterwards, it is not as adaptive to test songs as its unsupervised counterpart. The detection accuracy may significantly degrade when the characteristics of the training data and a test song are considerably different. For instance, if the system detects boundary according to the energy level in a certain frequency range, the system may not work for a song whose energy in that frequency range maintains high throughout the song.

Based on the above observations, we propose to take advantage of both methodologies by aggregating the results of an unsupervised model and a supervised one. In this way, we can exploit the discriminative information provided by the training data and the song-specific information of a test song at the same time. Moreover, to better capture the discriminative characteristics of a boundary, we further propose a number of novel *mid-level* features, including novelty score, dissonance level and vocal occurrence. Comparing to low-level features such as the spectral properties, these mid-level features carry more semantic meaning that improves music boundary detection.

A schematic diagram of the proposed system is shown in Fig. 1. An input song is partitioned by the beat onsets and represented by a set of low-level and mid-level features. The probability of each beat onset of being a boundary is then computed by both supervised and unsupervised methods with the features extracted from the subsequent beat interval. We then model the beat onsets as the vertices of a directed graph, with the vertex weights determined by the probability of being a boundary and the edge weights determined based on the music knowledge of the typical length of a music section [7, 11]. Finally, we formulate music boundary detection as a shortest path problem and identify the true boundaries by the Viterbi algorithm [18].

The paper is organized as follows. Section 2 describes the feature representation of music, including low-level and mid-level features. Section 3 elaborates on the system framework and the adopted supervised and unsupervised approaches. Experimental result is presented in Section 4. Section 5 concludes the paper.

2. MUSICAL REPRESENTATION

Before feature extraction, each song is converted to a standard format (mono channel and 22,050 Hz sampling rate) and partitioned into several beat intervals by the beat onset detection algorithm BeatRoot [12]. We adopt beat interval instead of frame as the basic time unit because the characteristics of a song are more likely to be consistent within a beat interval and because a music boundary tends to occur at a beat onset [7].

2.1 Low-level Features

For low-level local features, we use 40-dim Mel-scale cepstral coefficients (MFCCs), 24-dim chromagram, and 52-dim fluctuation patterns (FPs) [19] to represent the timbre, harmony, and rhythm aspects of music. We extract MFCCs and chromagram with a 40ms and non-overlapping sliding window and aggregate the frame-level features within each beat interval by taking the mean and the standard deviation. FPs are computed directly for each beat interval. These features have been found useful for music boundary detection [10]. Note these features only capture the local property of music.

2.2 Mid-level Features

Below we describe three mid-level features: novelty score, dissonance level, and vocal occurrence. While the first one is originally proposed by Cooper *et al* in [4], it has been used in an unsupervised setting rather than as a mid-level feature in a supervised one. On the other hand, though the latter two features have been studied in the context of music theory [21], few attempts have been made to incorporate them to the task of music boundary detection for raw audio signals.

2.2.1 Novelty Score

The novelty score is computed by two steps [4]. First, a similarity matrix is constructed by measuring the similarity of the low-level feature vectors of every two beats in a song. In this matrix, the two segments beside the boundary produce two adjacent square regions of high within-segment similarity along the main diagonal and two rectangular regions of low between-segment similarity off the main diagonal. As a result, each boundary produces a *checkerboard* pattern in the matrix and the beat interval that boundary occurs is the crux of this checkerboard. To identify these patterns, we correlate a Gaussian-tapered checkerboard kernel along the main diagonal of the similarity matrix to compute the so-called novelty scores, which measures both the dissimilarity between two different adjacent segments beside each potential boundary as well as the similarity within these segments. We define the term *segment* here to represent a set of consecutive beat intervals and the term *section* as a segment which is semantically meaningful (such as verse, chorus or bridge).[1]

In this work, we compute three novelty scores based on the three low-level features. Because the novelty scores of adjacent beats tend to be similar,[2] we also divide the novelty score of a certain beat interval by the sum of the novelty scores of neighboring beat intervals and use the normalized score as additional feature, resulting in a total of 6 features for each beat interval.

2.2.2 Dissonance Level

It is known in musicology that the relaxation or release of tension plays an important role in the transition of music sections. Because changes in tension often occur when dissonance giving way to consonance [13], we develop a novel feature based on the dissonance level of music. We first define the *dissonant intervals* according to the relationship between the pitches of two notes that cause tension (e.g., Tritone and Minor Second [14]), and then compute the dissonance level as the weighted sum of the corresponding dissonant intervals from the unwrapped chromagram of a beat,

$$y_t = \frac{\sum_{q \in D} \sum_{m} k_q c_m c_{m+q}}{\sum_{m} c_m}, \quad (1)$$

where y_t denotes the dissonance level of a beat t, q denotes the interval that has q semitones between the two notes, D is the set of dissonant intervals, c_m is the m_{th} bin of the chromagram, and k_q is a constant corresponding to q, which is empirically set according to the ratio of frequencies of the two pitches in q. The denominator is a normalization term.

[1] While a segment can be of arbitrary length, the length of a section often follows a typical pattern, see Section 3.3.

[2] The novelty scores of adjacent beats are similar because the submatrices of the similarity matrix of these beats overlap a lot.

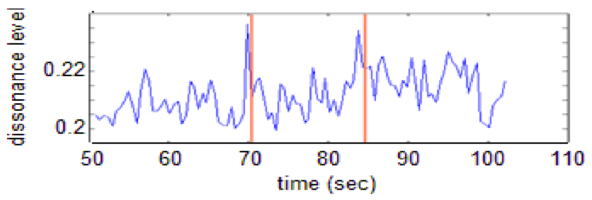


Figure 2. The dissonance level of a part of Billie Jean by Michael Jackson. The two red lines label a transition from verse to bridge and a transition from bridge to chorus. These boundaries occur right after high dissonance levels.

We compute the dissonance level for each beat interval and obtain a sequence of dissonance levels. We compute the derivative from the resulting sequence as the dissonant features to capture the changes in tension,

$$\Delta y_t = \frac{\sum_{m=-p}^{p} m y_{t-m}}{\sum_{m=-p}^{p} m^2}, \qquad (2)$$

where p denotes the window size. In this work, we set p to 1 and 2 and generate a two dimensional dissonance level feature. Fig. 2 illustrates the relationship between music boundary and dissonance level; clearly the music boundaries occur right after peaks of dissonance level (the rise and relax of tension).

2.2.3 Vocal Occurrence

In pop/rock songs, the time points that a vocalist sings often correspond to the music boundaries. For example, if a beat onset falls in the middle of a segment with pure instrument and another segment with singing voice, it is very likely a music boundary. Furthermore, because a music section is comprised of several music phrases,[3] a transition of music sections must also be a transition of music phrases. Therefore, if a beat onset falls in a short instrumental interval between two vocal music phrases, it is more likely to be a music boundary.

In light of the above observation, we train a vocal/non-vocal classifier by support vector machine (SVM) [15], with MFCC as the feature representation, to estimate the probability of the vocal occurrence for each beat interval. If the sum of these probabilities from the beat intervals in a segment exceeds a threshold, we regard the segment as a *vocal segment*. More specifically, the vocal occurrence feature of a certain beat interval is computed as follows. For a beat interval, if both of its neighboring segments are non-vocal, the vocal occurence is set to 0; if only one of the neighboring segments is non-vocal, the vocal occurrence is set to 1. When both neighboring segments are vocal, we set the vocal occurrence according to the following formula:

$$z_t = \frac{1 - v_t}{2w + 1} \sum_{j=t-w, j \neq t}^{t+w} v_j, \qquad (3)$$

where z_t is the vocal occurrence feature of beat interval t,

[3] Several music phrases constitute a music section.

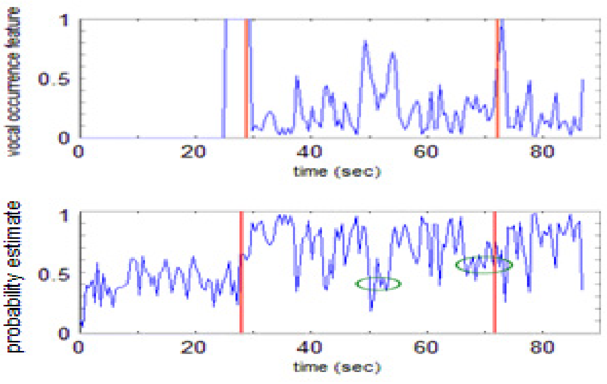


Figure 3. Top: the possibility of vocal estimated by SVM for a part of Billie Jean by Michael Jackson. The two red lines label a transition from intro to verse and a transition from verse to bridge. The green circles label two obvious transition points of music phrases. Bottom: corresponding vocal occurrence feature.

v_t is the probability estimate of beat interval t generated by the vocal/non-vocal SVM classifier, and ω is the window size that represents the length of the segment. We vary the value of ω and generate a multi-dimensional feature vector. In this work we set the value of ω to 8 and 12. An illustrative example is shown in Fig. 3. The first red line labels a transition from a non-vocal section (intro) to a vocal section (verse). The green circles label two obvious transition points of music phrases, while the latter one is in fact a transition point of music sections. We can see the corresponding vocal occurrence feature is highly correlated to music boundaries. A pitfall of this feature is that it may regard every phrase boundary as a section boundary and result in over segmentation. The use of other features may offset this mistake.

Representing the acoustic properties of music by these low-level and mid-level features, we then employ the system described below to detect boundaries.

3. SYSTEM DESCRIPTION

In this section, we first introduce the supervised and unsupervised approaches adopted in our system. Both approaches estimate the possibility of each beat onset of being a music boundary. Second, we describe how we integrate these two estimations with the music knowledge of typical section length.

3.1 Supervised Estimation

We train a SVM classifier with polynomial kernel and probability estimates to obtain the possibility of a beat onset being a music boundary. The label for a beat interval is marked 1 if a boundary occurs at that beat onset and 0 otherwise. Besides mid-level features, we also use the low-level features to train the classifier because low-level features also contain some relevant information. For example, a drum-fill is usually played when a music section ends; this characteristic can be detected by FP. For a test song, the SVM model computes the probability of the occurrence of a boundary at every beat onset. We utilize this probability as the output of the supervised approach.

3.2 Unsupervised Estimation

As for the unsupervised part, we construct three similarity matrices based on the kinds of low-level features and detect the peaks of the mean of the novelty scores from these matrices. We then use these peaks to divide the test song into a number of segments [4]. The low-level features of a segment are integrated to one vector by taking the mean and the standard deviation and a distance matrix among the segments is constructed by computing the pairwise distance between these vectors. The normalized cut algorithm [16] is then performed on the distance matrix to group these segments into acoustic similar clusters. At each beat interval, we further count the cluster indices of neighboring beat intervals within a predefined window size and establish two histograms: one for the beat intervals preceding to the beat onset, and the other for the subsequent beat intervals. The Euclidean distance of the resulting histograms can represent the possibility of a music boundary occurs at the designated beat onset, and the ratio of this possibility value of a beat onset to the sum of the possibility values of its neighboring ones is regarded as the estimation of the unsupervised approach.

3.3 Integration

Because music sections tend to have some typical length (e.g., 8 or 16 bars) [7, 11], it should be beneficial to incorporate this knowledge to the music boundary detection framework. As Fig. 4 illustrates, we construct a directed graph $G = (V, E)$ to integrate the estimates of supervised and unsupervised models and to take advantages of this music knowledge. In this graph, a vertex represents a beat onset, with the weight of it determined by the weighted sum of the estimates of supervised and unsupervised models

$$w_{v_i} = p_{u_i} + k_1 p_{s_i}, \tag{4}$$

where w_{vi} denotes the weight of a vertex i, p_{ui} and p_{si} are the probability estimates produced by an unsupervised model and a supervised one respectively, and k_1 is a parameter balancing the effect of the two models. The music knowledge of section length is incorporated as follows. If there exists the possibility that vertices v_i and v_j are two successive music boundaries, we form an edge between these two vertices. The weight of the edge is determined by the music knowledge of the length of a music section. We gather the statistics from training data to obtain the probability of two beats with specific temporal distance being music boundaries. That is, the weight of e_{ij} equals to the weight of e_{mn} if $j–i$ equals to $n–m$. To achieve this goal, a histogram is constructed by simply counting the number of beats of each music section from the training data.

Therefore, a path in this constructed graph can be regarded as a set of music boundaries. We further define the weight of a path B as the sum of the weights of its constituent edges and vertices,

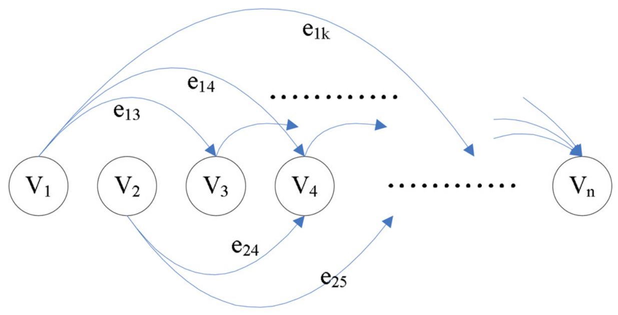


Figure 4. The directed graph G of a song, which has n beat onsets (vertices) and $k—1$ possible section lengths (possible jumps). The vertex weights are determined by the probability of being a boundary and the edge weights are determined based on the music knowledge of the typical length of a music section [7, 11]. We assume that every music section contains at least one beat interval.

$$w_B = \sum_{v\in B} w_v + k_2 \sum_{e\in B} w_e, \tag{5}$$

where w_v and w_e are the weights of a vertex and an edge in B, and k_2 is a constant to balance the effects of vertices and edges. We regard w_B as the probability of the associated beat onsets being correct music boundaries.

Because the path with maximum w_B consists of vertices that are most likely the music boundaries, we formulate the problem as a shortest path problem and employ the Viterbi algorithm [18] to solve it,

$$B^* = \arg\max_B w_B, \tag{6}$$

where B^* denotes the optimal solution. In practice, we only apply Viterbi to a feasible number of paths to reduce the complexity.

4. EXPERIMENT

4.1 Experimental Setup

We conduct an empirical evaluation on the RWC music dataset [17], which contains 100 pieces of song that are originally produced for experiment; most of the pieces (80%) are recorded according to 1990s Japanese chart music, while the rest resemble the 1980s American chart music. RWC dataset provides clear annotations of music boundaries and is adopted in many literatures in music boundary detection [6, 10].

We evaluate the performance in terms of precision (the proportion of true boundaries among the detected ones), recall (the proportion of true boundaries in the ground truth that are detected by the system), and f-score (the harmonic average of precision and recall). A detected boundary is considered correct if it falls within 1.5 seconds of the ground-truth, which is stricter than the one used in prior work [9] and should be reasonable for real-world applications.

For the unsupervised methods, we process each of the 100 songs independently and take the average result. For the supervised methods, we evaluate the system with

Approach	Method	Precision	Recall	F-score
Supervised only	N +D+V+L	0.2461	0.2932	0.2641
Unsupervised only	Cluster-based (normalized cut) [9]	0.2770	0.5166	**0.3517**
	Histogram-based	0.3068	0.3428	0.3124
Directly sum		0.3274	0.3470	0.3385
Integrated with section length	Viterbi algorithm [18]	0.3800	0.4452	**0.4094**

Table 2. Evaluation result of different musical boundary detection methods.

Feature	# feature	Precision	Recall	F-score
MFCC	40	0.1910	0.2574	**0.2142**
chromagram	24	0.1665	0.2131	0.1842
fluct. pattern	52	0.1906	0.2190	0.2019
local (L)	116	0.1982	0.2629	0.2206
difference [10]	6	0.1602	0.2519	0.1885
novelty (N)	6	0.2427	0.2770	**0.2549**
dissonance (D)	2	0.2109	0.2505	0.2198
vocal (V)	2	0.2128	0.2687	0.2240
N+L	122	0.2354	0.2900	0.2594
N+D+V	10	0.2322	0.2909	0.2592
N +D+V+L	126	0.2461	0.2932	**0.2641**

Table 1. Evaluation result of different features used in supervised musical boundary detection methods.

stratified five-fold cross validation: 20 random songs are held out as test data and the rest are used for training. The evaluation is iterated five times to get the average result.

4.2 Results

We first evaluate the supervised approach with different feature representations, including low-level and mid-level features. To compare the performance against previous work, we also implement the difference feature and its derivative proposed in [10]. The difference feature is computed by sliding a window along the audio signal and comparing the statistic of low-level features in the first half of the window with the ones in the second half. A beat onset is detected as a boundary if its probability estimate assigned by SVM exceeds a threshold. Instead of using a fixed threshold, we adaptively set the threshold of each song to be the mean plus one standard deviation of the probability estimates of the song.

The evaluation result is shown in Table 1. The three low-level features bring about similar accuracy, with FPs slightly worse than the other two, implying that the characteristics of music boundaries are represented more in timbre and rhythm. The direct concatenation of the three low-level features, which are denoted as local (L) in the table, further improves the f-score to 0.2206.

We then compare four mid-level features, including the difference feature proposed in [10]. It can be found that, with much lower feature dimension, the use of mid-level features achieves similar or superior performance to that attained by low-level features. The novelty score, in particular, achieve an f-score of 0.2549 that significantly outperform all other low-level or mid-level features. We can also find that the difference feature does not perform well, which possibly due to the disregard of the similarity of the beats in each segment.

The combination of mid-level and low-level features only brings about slight improvement, which somewhat implies that most of the information carried by low-level features has already been well represented by the mid-level features. The combination of novelty score (N), dissonance level (D), vocal occurrence (V), and local features (L) achieves the highest f-score of 0.2641.

We then compare the two unsupervised methods described in Section 3.2. For the cluster-based method, we simply mark the boundary of two consecutive segments that are associated with different clusters as a music boundary without smoothing. The result is shown in Table 2. As expected, the clustering-based approach exhibits a remarkably high recall but a relatively low precision. For the histogram-based method, we consider the segments whose probability estimates exceed a threshold as boundaries. The threshold value is set in the same way as in the supervised methods. The performance of the histogram-based method is slightly worse than the clustering-based one, showing that gathering statistics of neighboring frames does not improve the precision of boundary detection. Moreover, it can be noted that in our evaluation the unsupervised approaches generally outperform the supervised counterparts, showing that the ability of the unsupervised approach to be adaptive to each test song is essential in boundary detection.

Finally, we evaluate the performance of integrating the result of unsupervised and supervised methodologies. For comparison, we further implement a baseline method that simply sums up the supervised and unsupervised estimates with the same weight as the one in proposed graph-theoretical fusion method without exploiting the music knowledge of section length.

The result is also shown in Table 2. It can be found that simply taking the average has achieved a higher f-score than any of the supervised-only or unsupervised one, showing that the two methodologies are indeed complementary and the fusion of them is plausible. The proposed graph-theoretical fusion further improves the f-score to 0.4094, which greatly outperform the taking average baseline, especially in recall. This result shows the integration of the two methodologies and the incorporation of music knowledge are essential to music boundary detection.

A sample segmentation result is displayed in Fig. 5. In this example, all the boundaries can be correctly detected by the proposed system. Nevertheless, there is an over segmentation problem because the characteristics of the segments of the same music section may be incoherent.

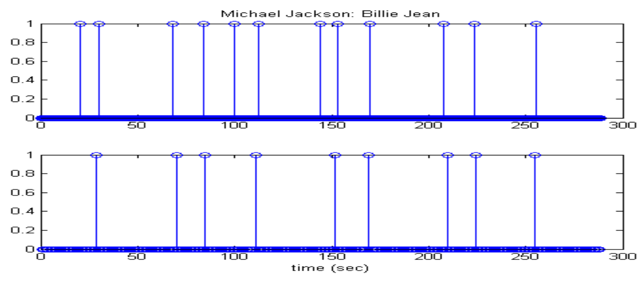


Figure 5. The segmentation result of Billie Jean by Michael Jackson. Top: the boundaries detected by the proposed system. Bottom: the manual annotation.

To resolve this problem, we are working on incorporating more music knowledge and mid-level features.

5. CONCLUSION

In this paper, we have presented an integrated system that combines the information from supervised approach, unsupervised approaches, and music knowledge. We formulate music boundary detection as a shortest path problem and employ the Viterbi algorithm to solve it. We also propose a number of novel mid-level features to better capture the discriminative characteristics of music boundaries. Experiments conducted on the RWC dataset show significant improvement over the state-of-the-art supervised-only and unsupervised-only methods.

6. ACKNOWLEDGEMENT

This work was supported by the National Science Council of Taiwan under the contract number NSC 97-2221-E-002-111-MY3. The authors would like to thank the anonymous reviewers for valuable comments that greatly improved the quality of this paper.

7. REFERENCES

[1] W. Chai, "Semantic segmentation and summarization of music," in *IEEE Signal Processing Magazine,* Vol. 23, No. 2, pp. 124–132, 2006.

[2] M. Levy, M. Sandler, and M. Casey, "Extraction of high-level musical structure from audio data and its application to thumbnail generation," in *Proc. ICASSP,* pp. 1433–1436, 2006.

[3] M. Goto, "A chorus-section detection method for musical audio signals and its application to a music listening station," in *IEEE Trans. Audio, Speech and Language Processing,* Vol.14, No.5, pp. 1783–1784, 2006.

[4] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *Proc. IEEE* Workshop *Applications of Signal Processing to Audio and Acoustics,* pp. 127–130, 2003.

[5] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: current directions and future challenges," in *Proceedings of the IEEE,* Vol. 96, No. 4, pp. 668–696, 2008.

[6] J. Paulus and A. Klapuri, "Music structure analysis using a probabilistic fitness measure and a greedy search algorithm," in *IEEE Trans. Audio, Speech and Language Processing*, Vol. 17, No. 6, pp. 1159–1170, 2009.

[7] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao, "Content-based music structure analysis with applications to music semantics understanding," in *Proc. ACM Multimedia*, pp. 112–119, 2004.

[8] L. Lu, M. Wang, and H. Zhang, "Repeating pattern discovery and structure analysis from acoustic music data," in *Proc. ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, 2004.

[9] M. Levy and M. Sandler, "Structural segmentation of musical audio by constrained clustering," in *IEEE Trans. Audio, Speech and Language Processing*, Vol. 16, No. 2, pp. 318–326, 2008.

[10] D Turnbull, G. Lanckriet, E. Pampalk, and M. Goto, "A supervised approach for detecting boundaries in music using difference features and boosting," in *Proc. ISMIR* 2007.

[11] C. Rhodes et al, "A Markov-chain Monte-Carlo approach to musical audio segmentation," in *Proc. ICASSP*, 2006.

[12] S. Dixon, "Evaluation of the audio beat tracking system BeatRoot," in *Journal of New Music Research*, Vol, 36, No. 1, pp. 39–50, 2007.

[13] DeLone et al, *Aspects of Twentieth-Century Music*. Englewood Cliffs, New Jersey: Prentice-Hal*l*, 1975

[14] Wyatt and Keith, *Harmony & Theory*. Hal Leonard Corporation, pp. 77, 1998.

[15] N. Maddage et al, "An SVM-based classification approach to musical audio," in *Proc. ISMIR,* 2003.

[16] Ji. Shi and J. Malik, "Normalized cuts and image segmentation," in *Proc. CVPR*, pp. 731–737, 1997.

[17] M. Goto, "AIST annotation for RWC music database," in *Proc. ISMIR*, 2006.

[18] G. D. Fomey, "The Viterbi algorithm," in *Proceedings of the IEEE* Vol. 61, No. 3, pp.268–278, 1973.

[19] E. Pampalk, "Computational models of music similarity and their application in music information retrieval," in PhD *thesis, Vienna University of Technology*, 2006.

[20] F. Lerdahl and R. Jackendoff, "An overview of hierarchical structure in music," in *Machine Models of Music*, pp. 289–312, 1993.

[21] D. Pressnitzer, S. McAdams, "Two phase effects in roughness perception," in *The Journal of the Acoustical Society of America,* Vol. 105, No. 5, pp.2773–2782, 1999.

AUTOMATIC DETECTION OF INTERNAL AND IMPERFECT RHYMES IN RAP LYRICS

Hussein Hirjee **Daniel G. Brown**
University of Waterloo
Cheriton School of Computer Science
{hahirjee, browndg}@uwaterloo.ca

ABSTRACT

Imperfect and internal rhymes are two important features in rap music often ignored in the music information retrieval community. We develop a method of scoring potential rhymes using a probabilistic model based on phoneme frequencies in rap lyrics. We use this scoring scheme to automatically identify internal and line-final rhymes in song lyrics and demonstrate the performance of this method compared to rules-based models. Higher level rhyme features are produced and used to compare rhyming styles in song lyrics from different genres, and for different rap artists.

1. INTRODUCTION

Song lyrics have received relatively little attention in music information retrieval, but can provide data about song style or content that is missing from raw audio files or user-input tags. Recent work focusing on lyrics [1–3] involves using lyric text to extract song topic, theme, or mood information; the pattern and sound of the words themselves is usually ignored.

These sound features are central to rap music, providing information about vocal delivery and rhyme scheme. This data can be characteristic of different rappers, as MCs often boast of the uniqueness and superiority of their rhyming style. Lyric rhymes have previously been studied as an aid in characterizing different musical genres [4], but this prior work ignores two stylistic features of rap lyrics: imperfect rhymes, where syllable end sounds are similar but not identical, and internal rhyme, which occurs in the middle of lines.

To study these features, we have developed a system for automatic detection of rap music rhymes. We train a probabilistic scoring model of rhymes using a corpus of rap lyrics known to be rhyming, using ideas derived from bioinformatics. We then use this model to find and categorize various rhymes in different song lyrics, and assess the model's success. Finally, we calculate high-level statistical rhyme scheme features to attempt to quantitatively model and compare rhyming styles between artists and genres. Our work allows the automated study of new features in rap music, and may be extensible to other genres of song lyrics or for poetry analysis.

2. BACKGROUND

Hip hop music is characterized by lyrics with intermittent rhymes being rhythmically chanted (rapped) to an accompanying beat. In "Old School" rap (dating from the late 1970s to mid 1980s), lyrics typically followed a simple pattern and contained a single rhyme falling on the fourth beat of each bar [5]. Contemporary rap features more varied delivery and many complex rhyme stylistic elements that are often overlooked. Key among these are rhymes that are imperfect, extended, or internal. Holtman [6] provides a good overview of the abundance of imperfect rhyme in rap lyrics. A normal rhyme involves two syllables that share the same nucleus (vowel) and coda (ending consonants). Two syllables form an imperfect rhyme if one of these two parts does not correspond exactly. However, these types of rhymes are not just composed of vowels and consonants being paired randomly; there is a constraint to the amount of dissimilarity in these rhymes, determined by the shared articulatory features of matching phonemes.

In Holtman's hierarchy, the most similar consonants are nasals, fricatives, and plosives differing only in place of articulation, as in the line-ending /m/ and /n/ phonemes in:

> Entertain and tear you out of your **frame**
> Leave you in a puddle of blood, then let it **rain**. [7]

(Rhyming syllables in quoted lyrics are displayed with the same font style.) Less similar consonant pairs include those with the same place of articulation, but differing in voice or continuancy, such as the /k/ and /g/ pair in:

> Bring a bullet-proof vest, nothin' to **ricochet**
> Ready to aim at the brain, now what the **trigger say**? [7]

Vowels are most similar when differing only in height or "length" (advanced tongue root), such as the penultimate vowels in:

> I'm the alpha, with no **omega**
> Beginning without the, end **so play the**. [7]

Holtman's work is largely taxonomic and describes known rhymes, rather than discovering them. Hence, we used a statistical model of phonetic similarity based on frequencies in actual rap lyrics. However, the patterns we automatically discovered largely validate her taxonomy.

Rap music often features triplet or longer rhymes with unstressed syllables following the initial stressed pair, which may span multiple words (mosaic rhymes). Longer rhymes can also include more than one pair of stressed syllables:

> Maybe my sense of **húmor gets ínto you**
> But girl, they can make a per**fúme from the scént of you**.
> [8]

(Here the accents mark the syllables with primary stress.) Finally, contemporary rap music features dazzlingly complex internal rhyme. Alim [9] analyzes Pharoahe Monch's 1999 album Internal Affairs [10] as a case study, and identifies chain rhymes, compound rhymes, and bridge rhymes. Chain rhymes are consecutive words or phrases in which each rhymes with the previous, as in:

> New York **City gritty** com**mittee pity** the fool that
> Act **shitty** in the midst of the calm the **witty**, [10]

where "city", "gritty", "committee", and "pity" participate in a chain. Compound rhymes are formed when two pairs of line internal rhymes overlap within a single line. A good example of this is given in "Official":

> Yo, I stick around like hockey, now what the **puck**
> Cooler than **fuck**, ma*neuver* like Van*couver* Ca**nucks**,
> [10]

where "maneuver" and "Vancouver" are found between "fuck" and "Canucks." Bridge rhymes are internal rhymes spanning two lines:

> **How I made it** you **salivated** over my **calibrated**
> RAPS that **validated** my ghetto credi*bility*
> *Still I be* PACKin a*gilities* unseen
> Forreal-a my killin a*bilities* unclean fa*cilities*. [10]

Here, we call pairs in which both members are internal (such as "agilities" / "abilities") bridge rhymes, and those where the first word or phrase is line-final (such as "calibrated" / "validated"), link rhymes.

3. FINDING RHYMES AUTOMATICALLY: A PROBABILISTIC MODEL

We modeled our rhyme detection program after local alignment protein homology detection algorithms using BLOSUM (BLOcks of amino acid SUbstition Matrix) [11]. In this framework, pairs of proteins are modeled as sequences of symbols generated either randomly or based on shared ancestry (homology). Pairs of matched amino acids receive a log-odds score in the BLOSUM matrix M: a positive score indicates the pair more likely co-occurs due to homology, and a negative score indicates the pair is more likely to co-occur due to chance. Scores are in log-odds: $\mathrm{M}[i,j] = \log_2(\Pr[i,j|\mathrm{H}]/\Pr[i,j|\mathrm{R}])$, where H is a model of related proteins (obtained by counting the frequency with which we see symbols i and j matched to each other in proteins known to be homologous) and R is the frequency of the symbols i and j in random proteins (obtained from frequency counts over all proteins). If a pair of protein sequences contains regions in which the amino acids align to give high scores, the pair is considered to be homologous.

In our work, song lyrics are transcribed into sets of sequences of syllables, with each sequence corresponding to a line of text. Similar to Kawahara's [12] treatment of consonants in Japanese rap lyrics, probabilistic methods are used to calculate similarity scores for any given pair of syllables. Phonemes which match with each other in rhyming phrases more often than expected by chance receive positive scores, while those which match less often than expected receive negative scores. Regions with syllables that, when matched to each other, have total score surpassing a threshold are identified as rhymes.

4. RHYMING SYLLABLES

To generate models of rhyming and randomly co-occurring syllables in rap lyrics, we needed a data set of known rhymes. Our training corpus includes the lyrics of 31 influential albums from the "Golden Age" of rap (1984-1994), chosen because they received the highest rating from The Source, the top-selling US rap music magazine of the time, plus nine additional albums by influential artists from the time period (Run-DMC, LL Cool J, The Beastie Boys, Public Enemy, Eric B. and Rakim). We downloaded lyrics from the Web and manually corrected them to fix typos and ensure that pairs of consecutive lines ended with matching rhymes, yielding 27,956 lines of lyrics (13,978 rhymed pairs), approximately 700 lines per album.

We first transcribe plaintext lyrics into sequences of phonemes using a wrapper we built around the Carnegie Mellon University (CMU) Pronouncing Dictionary [13], which gives phonemes and stress markings for words in North American English. We augmented the dictionary with slang terms and common elements of hip-hop vernacular (e.g., the "-in" ending in "runnin' ", or the "-a" ending in "brotha" or "killa"), and reduced the stress assigned to common one-syllable words of minor significance in rhyme ("a", "I", etc.). To handle words not found in the augmented dictionary, we added the Naval Research Laboratory's text-to-phoneme rules [14].

5. SCORING POTENTIAL RHYMES

To generate a log-odds scoring matrix for rhyming syllables, we need models for random syllables and for rhymes. For any pair of syllables i and j, the random model, $\Pr[i,j|\text{Random}]$, gives the likelihood of i and j being matched together by chance while the rhyme model, $\Pr[i,j|\text{Rhyme}]$, gives the likelihood of i and j being paired in a true rhyme. As in BLOSUM [11], the log-odds score is calculated as $\ln(\Pr[i,j|\text{Rhyme}]/\Pr[i,j|\text{Random}])$. To avoid overfitting, we reduce each syllable to its vowel (nucleus), end consonants (coda), and stress—the relevant features for determining rhyme. We approximate the coda by taking the first half (rounded up) of the consonants between adjacent pairs of vowels. Both models are trained using the occurrence frequencies of phonemes in the training data.

In the random model, the likelihood of vowel a matching with vowel b is calculated by taking the product of the frequencies of a and b. The likelihoods for consonants and varying stress are calculated in the same manner. For the rhyming model, the likelihood of vowels a and b being matched is calculated by taking the number of times a and b are seen matching in known rhymes, and dividing by the total number of matched vowel pairs in known rhymes. Then the log-odds score for the vowels is calculated as $\text{vowelScore}(a, b) = \ln(\Pr[a, b|\text{Rhyme}] / \Pr[a, b|\text{Random}])$.

The likelihood for consonants is more complicated since we must also consider unmatched consonants when aligning syllable codas of differing size. We use an iterated approach to solve these problems. In the first pass over the training data, we produce initial vowel and consonant scoring matrices by calculating the statistics above. We consider rhymes in paired lines to be all syllables following the final primary-stressed syllable, after Holtman [6]. In the second pass, we identify the start of rhymes by moving backwards from the end of the line while initial scores for stressed syllables are positive. We perform global alignment [15] on matched codas to determine frequencies for consonants pairing with other consonants, and being unmatched at the start or end of the coda. This distinction is useful since some consonants (such as /l/ and /r/) are more likely to be unmatched at the beginning of clusters, and others (often coronals, such as /d/ and /z/) are more likely to be unmatched at the ends of clusters. A simple example of this is found in the repeated occurrences of "alarmed" rhyming with "bomb" in Public Enemy's "Louder Than A Bomb." [16]

Using these frequency statistics, we produce the rhyming model and log-odds scores for consonants and stress in the same way as for vowels. Finally, we normalize the consonant score by dividing by the length of the coda to avoid the problem of syllables with long codas having the consonant score dominate. Intuitively, "win" and "gin" rhyme as well as "splints" and "mints." Since all the constituent scores are log-odds, they can be added together to form a combined probabilistic log score. The final score for two given syllables is the sum of the vowel score, normalized consonant score, and stress score.

Tables 1 and 2 show the pairwise scoring matrices. The symbols "_*" and "*_" indicate scores for unmatched consonants at the beginning and end of codas, respectively. High scores for pairs like (/m/,/n/) and (/k/,/p/) largely validate Holtman's hierarchy [6].

6. RHYME DETECTION ALGORITHM

With our probabilistic scoring method for matched syllables in place, we need a procedure to identify internal and end rhymes. Our technique is a variant on local alignment [15]; for each syllable, we identify its closest preceding rhyming syllable, and longest preceding rhyming phrase within the current and previous lines. For example,

	AA	AE	AH	AO	AW	AY	EH	ER	EY	IH	IY	OW	OY	UH	UW
AA	2.3	-3.3	-0.8	1.6	-1.7	-2.7	-7.2	-0.6	-3.9	-4.8	-3.9	-1.0	-1.7	-3.3	-3.9
AE		2.1	-1.5	-6.6	-1.9	-3.3	-1.5	-3.4	-1.8	-2.0	-4.3	-4.6	-4.5	-3.7	-6.7
AH			2.2	-1.2	-1.4	-1.4	-0.6	-0.2	-1.7	-0.3	-3.0	-1.0	-0.6	-0.9	-1.5
AO				3.1	-1.0	-3.8	-6.5	-1.1	-3.9	-4.2	-6.3	-0.3	-0.4	1.1	-3.3
AW					3.8	-0.3	-6.0	-4.2	-5.7	-6.0	-5.7	-2.0	-2.9	-4.5	-1.4
AY						2.5	-4.2	-1.1	-7.0	-1.8	-3.2	-4.3	-1.1	-5.7	-6.4
EH							1.9	-1.2	-1.5	0.2	-2.1	-7.0	-4.5	-6.1	-4.3
ER								3.9	-5.6	-1.5	-5.5	-1.6	-2.7	-1.3	-2.6
EY									2.5	-3.4	-2.7	-4.4	-4.3	-5.8	-6.5
IH										2.0	-0.9	-7.1	0.2	-2.2	-3.7
IY											2.4	-4.4	-4.2	-5.8	-6.4
OW												2.8	-4.0	-2.5	-1.5
OY													4.9	0.1	-3.7
UH														2.6	-0.5
UW															3.1

Table 1. Scoring Matrix for Vowels

given the line

> *Unob**tain**able* to the **brain** it's *unex**plain**able* what the verse'll do [10]

from Pharoahe Monch's "Right Here," the middle "ain" syllables all rhyme, while the whole of "unexplainable" also rhymes with "unobtainable."

For every pair of consecutive lines in a set of lyrics, we first construct a two-dimensional matrix of the score for every pair of syllables. Entries in this matrix (corresponding to pairs of syllables in the lines) are selected as "anchors" if they have score above a threshold and contain a stressed syllable or are line-final. From these anchor positions, rhymes are extended forward, ensuring that the length-normalized score is above a syllable threshold. In addition to the iterative extension, a "jump"-type extension is also allowed, in which one or two syllables can be skipped over if the following syllable pair is an anchor type with score above a higher threshold. This was included since longer polysyllabic mosaic rhymes often contain one or two syllables that do not rhyme in the midst of three or four that do. A good example of this can be found in Fabolous' "Can't Deny It":

> I **keep spittin', them clips copped on those calicos**
> **Keep shittin', with ziplocks of that Cali 'dro** [8]

where the two lines rhyme in their entirety, with the exception of "them"/"with" and "those"/"that."

We filtered the set of rhymes to remove one-syllable rhymes including unstressed syllables, as these tended to be noise. After a set of rhymes was identified, we removed duplicates and consolidated consecutive and overlapping rhymes together.

7. VALIDATING THE METHOD

Our first test verifies that our probabilistic score for syllable rhyming is better at identifying perfect and imperfect rhymes than rules-based phonetic similarity measures. We did a 10-fold cross validation where we chose 36 albums from the training data, trained a rhyme model for those albums, and used it to score the known rhyming lines from the other four albums (true positives) as well as randomly selected lines from those four albums (presumed to be true

	B	CH	D	DH	F	G	JH	K	L	M	N	NG	P	R	S	SH	T	TH	V	Z	ZH	_*	*_
B	4.3	-4.8	1.1	0.4	-5.5	1.9	1.9	-6.9	-0.3	-0.5	-1.6	-5.5	0.1	-0.9	-1.6	-4.6	-1.0	-4.3	2.3	0.3	-2.5	-0.6	-1.5
CH		4.2	-1.6	-4.9	-0.3	0.3	0.4	1.5	-6.8	-6.6	-2.8	-5.5	1.1	-6.7	0.3	0.6	0.9	1.4	-6.1	-2.0	-2.5	-6.0	-2.6
D			2.3	-7.0	-7.6	0.1	0.2	-3.1	-1.7	-2.2	-2.2	-3.0	-1.8	-0.9	-9.0	-2.1	0.2	0.0	-0.2	0.0	-4.6	-0.2	1.2
DH				3.5	-5.6	-5.1	-4.2	-0.4	-0.2	-2.0	-7.5	-5.6	-6.2	-1.4	-7.0	-4.8	-0.3	1.3	2.8	1.1	-2.6	-6.0	-3.4
F					3.4	-1.2	-4.9	-0.3	-1.5	-1.3	-3.5	-1.6	1.1	-2.7	1.1	1.2	-0.9	4.0	0.6	-7.3	-3.2	-1.4	-2.9
G						4.2	1.9	0.0	-0.2	-1.0	-1.9	-5.7	-0.6	-0.8	-2.5	-4.9	-1.1	-4.5	0.3	-0.3	-2.7	-0.9	-2.8
JH							5.2	-6.3	-1.5	0.1	-0.5	-4.8	-0.2	-0.3	-0.6	0.6	-1.1	-3.6	1.4	1.0	4.1	-5.3	0.5
K								2.6	-2.9	-2.1	-2.6	-1.3	1.7	-2.1	-0.7	-0.6	0.9	0.5	-1.8	-3.1	-4.7	-1.0	-1.8
L									2.8	-1.8	-1.8	-2.8	-8.1	-0.5	-2.9	-6.6	-2.9	-6.3	-1.3	-1.6	-4.5	0.4	-1.0
M										2.7	1.8	0.7	-3.2	-1.2	-2.9	-1.1	-2.5	0.4	-0.6	-3.7	-4.2	-0.8	-1.7
N											2.2	1.2	-2.5	-1.0	-2.3	-0.7	-1.5	-0.6	-1.5	-2.1	-5.1	-0.4	-2.3
NG												4.1	-6.8	-2.7	-2.3	-5.3	-3.5	-5.0	-2.1	-2.0	-3.2	0.2	-3.9
P													3.3	-2.0	-1.1	-0.7	1.1	0.9	-0.6	-7.9	-3.8	-0.7	-0.8
R														2.8	-2.3	-0.8	-1.2	-6.1	-2.1	-2.2	-4.3	1.7	-0.7
S															2.6	2.4	-1.0	1.0	-2.4	0.5	0.0	0.6	0.6
SH																5.2	-0.6	-4.1	-1.3	-0.2	3.6	-5.8	-7.7
T																	1.7	1.6	-0.9	-9.2	-5.2	0.0	0.7
TH																		4.4	0.5	-6.1	-2.0	-5.4	-0.6
V																			2.9	-0.4	1.6	-1.2	-1.7
Z																				2.6	3.0	-1.3	1.1
ZH																					6.8	-3.7	-5.6

Table 2. Scoring Matrix for Consonants

negatives). We developed implementations of the minimal mismatch of articulatory features and Kondrak alignment [17] metrics to compare the performance of these scoring measures, which are based on the physical process of the human voice. We show receiver operator characteristic (ROC) curves comparing the true positive rate to false positive rate when varying the score threshold for each of the three methods in Figure 1. The probabilistic method significantly outperforms both simpler rules-based methods.

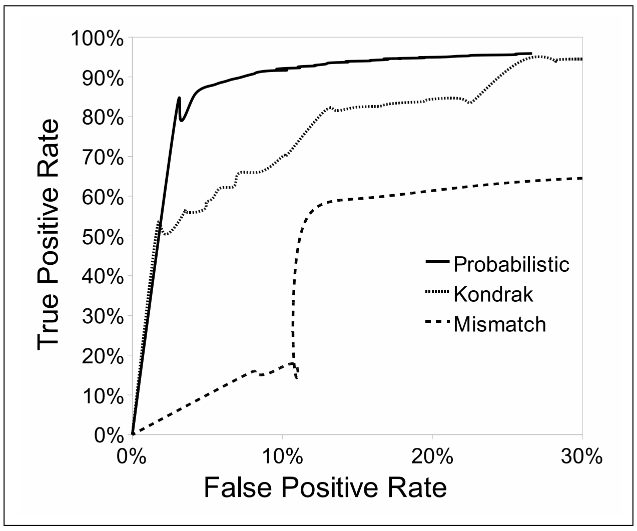


Figure 1. ROC curves for the three different scoring methods, comparing percentage of actual rhymes found by algorithm on the y-axis with percentage of unrelated syllables detected as rhyming on the x-axis

Next, we considered false positives and negatives for detected end rhymes, using the score threshold of 1.5 (meaning matched syllables are at least $e^{1.5}$ times more likely to rhyme than expected by chance). Out of 1000 pairs of unrelated random lines from our training data, 79 syllables were marked as parts of end rhymes ("false positives") by our procedure. Of these, 22 were in fact true rhymes, with scores higher than 3.0. 30 were near-rhymes; that is, that they could be found (though less frequently) as line final rhymes in actual lyrics. Usually scoring above 2.0, they included matches such as "stiff"/"fit", "pen"/"thing", and "cling"/"smothering", with more than one articulatory difference or different stress. 14 matched end syllables (often suffixes), typically with high scores (greater than 3.0). Examples such as "weak**er**"/"drumm**er**" and "tapp**in'**"/"posit**ion**", may have exact matches, but are not relevant rhymes due to their lack of stress. The remaining 13 moderately high scoring (between 1.5 and 2.5) pairs featured either high consonant scores (like "bust"/"test") or high vowel scores due to matching rare vowel sounds ("box"/"wrong").

From a set of 1000 matched pairs of lines, we used the iterative method (moving backwards from the end of the line while scores for stressed syllables are positive) to see which true rhymes would be missed. Pairs with all such matches scoring less than 1.5 were marked and treated as false negatives. Out of 132 such syllables, the largest group (48) were moderately low scoring (between -1.0 and 1.5) pairs participating in polysyllabic and mosaic rhymes. A good example of this is "battery"/"battle me" in Eric B. and Rakim's "No Omega" [7]; many of these were flanked by high scoring pairs, and would be included in rhymes using the jump extension described in the above section. 35 were very low scoring pairs (less than 0.0) which were either caused by words having been transcribed improperly or the lack of a true rhyme in the lyrics. 22 were caused by the rhyme start being extended too far back and starting with a low positive scoring pair. Again, this would not cause problems in our actual detection algorithm since, in that case, rhymes are extended forward from stressed anchors. 17 were caused by differences between the actual pronunciation and the dictionary's pronunciation ("poems" treated as one syllable, or "battles" specifically being pronounced to rhyme with "shadows"). Finally, 10 were caused by deliberate mismatch in syllable stress.

The probabilistic model is quite good at finding both perfect and imperfect rhymes. Quite few syllable pairs (less than 15 in the 1000 line pairs) scored highly without being perceivably rhyming, and most low scoring "true" rhyme pairs take part in complex mosaic and polysyllabic rhymes.

Finally, we used our model on a set of manually annotated rap lyrics, to measure the ability of the program to find both internal and line-final rhymes. We used five songs of varying style: the Beastie Boys' "Intergalactic", a Grammy-winning song in the old-school style; Pharoahe Monch's "The Truth" (featuring Common and Talib Kweli) and "Right Here", which were annotated by Alim [9] and feature high rhyme density and a complicated scheme; Jay-Z and Eminem's "Renegade", which features very high rhyme density; and Fabolous' "Trade It All (Part 2)", a song specifically mentioned by Alim for its prevalence of long (five or six syllable) rhymes. We show the ROC curves for this test set in Figure 2; the best overall performance is for specificity and sensitivity just above 60%. Most "false positive" are rhymes that were not annotated due to lack of rhythmic importance or accidental omission. False negatives included several where the performer created a rhyme from words that do not appear to rhyme as text, and some longer rhymes that were cut off prematurely due to too many non-rhyming syllables within them and lower scoring syllable pairs surrounding them. Finally, some rhymes were missed due to intervening rhymes being found between the rhyming parts, particularly when the threshold for rhymes is set low. This is especially evident in the ROC curves at lower cut-off thresholds, where true positive rates peak around 80% and begin to decline as the threshold is lowered.

8. EXPERIMENTS

We used our procedure to examine a variety of features about the rhymes in several sets of lyrics. We computed the number of syllables per line, the number of rhymes per line, the number of rhymes per syllable, average end rhyme scores, and proportion of rhymes having two, three, four, or more syllables. We also counted all of the complex rhyming features (bridge, link, chain and internal rhymes) per line.

We hypothesized that these features would show differences between genres of popular music, and calculated them for four sets of data: the top 10 songs from Billboard Magazine's 2008 year-end Hot Rap Singles chart; the top 20 songs from the 2008 year-end Hot Modern Rock Songs chart; the first 400 lines of Milton's "Paradise Lost" [18], as a similar-sized sample of non-rhyming verse; and the top 10 songs from the 1998 year-end Hot Rap Singles chart. To compare the verses most of all, the song lyrics were modified to remove intro/outro text, repeated lines, and additional choruses. Our results are in Table 3. High end rhyme scores are indicative of song lyrics in general (relative to unrhymed verse); rap has higher rhyme density, internal rhyme, link rhymes, and bridge rhymes. Interestingly, blank verse and rock lyrics have similar amounts of rhyming per line, but rock lyrics have more rhymes per syllable. The data from 1998 and 2008 rap songs suggest that in their rhyming pattern, there has not been much shift in style.

	Rap '08	Rap '98	Rock	Blank
Number of Lines	476	613	502	400
Number of Syllables	4646	6492	4053	4146
Syllables per Line	9.76	10.59	8.07	10.37
Number of Rhymes	794	1118	476	393
Rhymes per Line	1.67	1.82	0.95	0.98
Rhymes per Syllable	0.17	0.17	0.12	0.09
Rhyme Density	0.28	0.27	0.19	0.12
Average End Score	5.28	5.21	4.36	2.49
per Syllable	3.75	3.67	4.01	2.28
Doubles per Rhyme	0.23	0.29	0.15	0.18
Triples per Rhyme	0.08	0.06	0.04	0.03
Quads per Rhyme	0.02	0.03	0.05	0.00
Longs per Rhyme	0.03	0.02	0.04	0.01
Internals per Line	0.62	0.60	0.27	0.28
Links per Line	0.20	0.28	0.13	0.16
Bridges per Line	0.43	0.48	0.28	0.40
Chaining per Line	0.32	0.18	0.15	0.07

Table 3. Rhyme Features for Different Genres

We also hypothesized that features of individual rappers might also be informative, so we produced these statistics for albums by nine famous MCs from a diverse range of styles and eras: Run-DMC, Rakim, Notorious B.I.G., 2Pac, Jay-Z, Fabolous, Eminem, 50 Cent, and Lil' Wayne. Features were calculated for segments of at least 40 lines to produce means and standard deviations of the statistics for each album. The results indicate that many of these features can be characteristic of different artists' styles. For example, Run-DMC's (1984) old-school style has lower rhyme density and less internal rhyme with an average of 1.5 rhymes per line and only 6% of rhymes being longer than 2 syllables; while Rakim (1987), known for his more complex style, is detected as using more internal rhymes (0.63 per line to Run-DMC's 0.48) and more rhymes longer than 2 syllables (12%). Rival rappers Notorious B.I.G. (1994) and Tupac Shakur (1995) display fairly similar style characteristics: 28% of their rhymes are 2 syllables long, 6% are three syllables, and 3% are longer. However, Biggie's lines are significantly shorter in length, with, on average, 10.8 syllables to 2Pac's 11.6.

Artists from the early 2000s like Jay-Z (2001), Eminem (2000), and especially Fabolous (2001) favour longer rhymes, with 15%, 17%, and 30% respectively of their rhymes being longer than 2 syllables. They also have the most rhyme density overall, with 2.2, 2.3, and 1.9 rhymes per line respectively. Jay-Z and Eminem tend to use more internal rhyme as well, having 0.8 internal rhymes per line–about 25% higher than the average among other MCs. Although he portrays a "thug" persona, 50 Cent (2003) uses the most syllables per line (12.1), while Lil' Wayne (2008) has the fewest (10.2). However, he manages high rhyme density (0.3 rhymed syllables for each syllable used) with relatively few (only 1.8) rhymes per line. In general, we find that automatic rhyme detection can yield characteristic data about performers and genres.

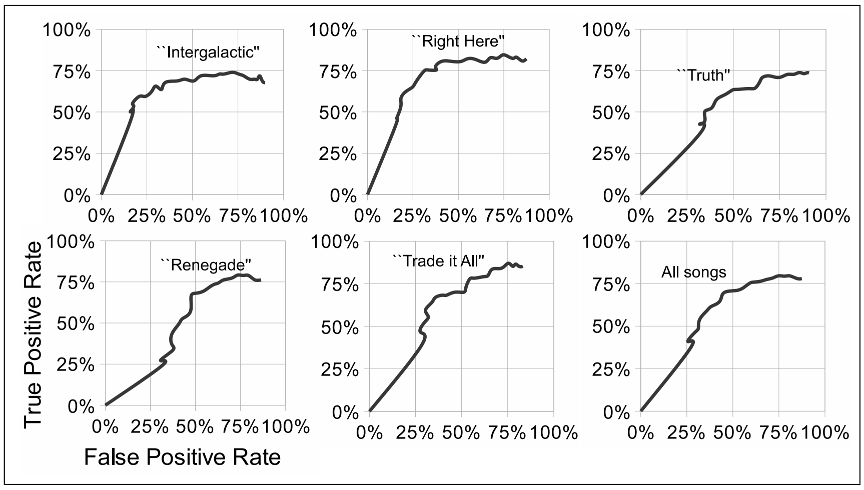


Figure 2. Rhyme Detection Syllable ROC Curves for Different Songs. The y-axis indicates the percentage of true rhymes identified by the algorithm, while the x-axis shows the percentage of automatically identified rhymes not considered to be true rhymes.

9. CONCLUSION

Using a probabilistic scoring model, we were able to identify both perfect and imperfect rhymes with a higher level of accuracy then simpler rules-based methods. The heuristic rhyme detection methods achieved moderate success at finding both internal and line-final rhymes in song lyrics. More importantly, statistical features of these rhymes did correspond to real world characterizations of rhyme style, and many of these features are quite consistent within individual artists' lyrics and varied between different artists. This leads to the possibility that automatically calculated rhyme statistics can be used to make meaningful categorizations and recommendations based on rhyme style.

10. REFERENCES

[1] B. Wei, C. Zhang, M. Ogihara: "Keyword Generation for Lyrics," *Proceedings of the International Conference on Music Information Retrieval*, 2007.

[2] F. Kleedorfer, P. Knees, T. Pohle: "Oh Oh Oh Whoah! Towards Automatic Topic Detection in Song Lyrics," *Proceedings of the International Conference on Music Information Retrieval*, pp. 287–292, 2008.

[3] H. Fujihara, M. Goto, J. Ogata: "Hyperlinking Lyrics: A Method for Creating Hyperlinks Between Phrases in Song Lyrics," *Proceedings of the International Conference on Music Information Retrieval*, pp. 281–286, 2008.

[4] R. Mayer, R. Neumayer, A. Rauber: "Rhyme and Style Features for Musical Genre Classification by Song Lyrics," *Proceedings of the International Conference on Music Information Retrieval*, pp. 337–342, 2008.

[5] Adam Bradley: *Book of Rhymes: The Poetics of Hip Hop*, Basic Civitas Books, 2009.

[6] Astrid Holtman: *A Generative Theory of Rhyme: An Optimality Approach*, PhD dissertation, Utrecht Institute of Linguistics, 1996.

[7] Eric B. and Rakim: *Let the Rhythm Hit 'Em*, MCA Records, 1990.

[8] Fabolous: *Ghetto Fabolous*, Elektra Records, 2001.

[9] H. Samy Alim: "On Some Serious Next Millennium Rap Ishhh: Pharoahe Monch, Hip Hop Poetics, and the Internal Rhymes of Internal Affairs," *Journal of English Linguistics*, Vol. 31, No. 1, pp. 60–84, 2003.

[10] Pharoahe Monch: *Internal Affairs*, Rawkus Records, 1999.

[11] S. Henikoff and J.G. Henikoff "Amino Acid Substitution Matrices from Protein Blocks" *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 89 No. 22 pp. 10915-10919, 1992.

[12] Shigeto Kawahara: "Half rhymes in Japanese rap lyrics and knowledge of similarity," *Journal of East Asian Linguistics*, Vol. 16, No. 2, pp. 113–144, 2007.

[13] Kevin Lenzo: *The CMU Pronouncing Dictionary*, `http://www.speech.cs.cmu.edu/cgi-bin/cmudict,2007`.

[14] H.S. Elovitz, R.W. Johnson, A. McHugh, J.E. Shore "Automatic translation of English text to phonetics by means of letter-to-sound rules," *Interim Report Naval Research Lab*. Washington, DC., 1976 `http://www.speech.cs.cmu.edu/comp.speech/Section5/Synth/text.phoneme.3.html`.

[15] R. Durbin, S. Eddy, A. Krogh, G. Mitchison *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, 1999.

[16] Public Enemy: *It Takes a Nation of Millions to Hold Us Back*, Def Jam Recordings, 1988.

[17] Grzegorz Kondrak: "A New Algorithm for the Alignment of Phonetic Sequences," *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 288–295, 2000.

[18] John Milton: *Paradise Lost*, Samuel Simmons, 1667.

SLAVE: A SCORE-LYRICS-AUDIO-VIDEO-EXPLORER

Verena Thomas **Christian Fremerey** **David Damm** **Michael Clausen**
Department of Computer Science III
University of Bonn, Germany
{thomas,fremerey,damm,clausen}@iai.uni-bonn.de

ABSTRACT

We introduce the music exploration system SLAVE, which is based upon previous developments of our group. SLAVE manages multimedia music collections and allows for multimodal navigation, playback, and visualization in an efficient and user-friendly manner.[1] While previously the focus of our system development has been the simultaneous exploration of digitized sheet music and audio, with SLAVE we enhance the functionalities by video and lyrics to achieve a more comprehensive music interaction. In this paper, we concentrate on two aspects. Firstly, we integrate video documents into our framework. Secondly, we introduce a graphical user interface for semi-automatic feature extraction, indexing, and synchronization of heterogeneous music collections. The output of this GUI is used by SLAVE to offer both high quality audio and video playback with time-synchronous display of digitized sheet music and content-based search.

1. INTRODUCTION

Various aspects of a piece of music can be described by different types of music documents, such as scans of sheet music, symbolic data (e.g., MIDI, MusicXML), text (e.g., lyrics, libretti, music analysis), audio recordings, and video. In modern digital music libraries large collections of these music documents are stored. The availability of digital music collections naturally leads to the necessity of providing tools to automatically process, analyze and prepare this multimedia data for an efficient and user-friendly access. Equally, user interfaces for an adequate multimodal presentation of and interaction with the music documents need to be provided. The last years have witnessed substantial progress in developing automated MIR processing procedures to compute synchronization and indexing information for multimedia music collections [1–4, 6]. In the area of digitalization of multimedia library contents, efforts both towards better automatized digitization techniques and semantic integration of multimedia documents are noticeable [5, 6]. Furthermore there are several proposals for user interfaces to access and present digital music databases in a multimodal manner [1, 6–9].

The most frequently encountered digitally available types of music representation are scanned sheet music, symbolic score data and audio recordings. Therefore most of the presented techniques and frameworks mainly focus on one or several of these data types. However, there is another type of music representation, which can provide library users, musicians and musicologists with rich information on the pieces of music. Today, most live performances are filmed and distributed to a broad audience via video DVD and television. Besides recordings of live performances also specific video productions of pieces of music are available. Hence, the extension of the functionalities of multimodal frameworks to support video documents and the development of user interfaces for video integration suggest themselves.

A holistic presentation using as many different media sources and types as possible can support the process of experiencing the music as well as analyzing the music with respect to different aspects. Prospective conductors, for example, might be interested in watching music videos to learn or compare the conducting style of different conductors. Providing tools to allow fast and smooth comparison between and browsing within interpretations are desirable for this purpose.

In this paper, we introduce the *Score-Lyrics-Audio-Video-Explorer* (SLAVE), which is based upon previous developments of our group. As enhancement, we propose the integration of videos into SLAVE to converge to a holistic exploration of music using various types of music documents in an integrated manner. Furthermore, we introduce a graphical user interface for the semi-automatic processing of multimedia music collections to generate indexing and synchronization structures as well as other derived data types. Note that for videos, we solely use the audio track to perform all required calculations.

The rest of this paper is organized as follows. The subsequent Section 2 provides information on the underlying techniques of feature extraction and music synchronization. In Section 3 the workflow for processing music collections and a GUI for a user-friendly management of the workflow are described. Section 4 presents the inter-

We gratefully acknowledge support from the German Research Foundation DFG. The work presented in this paper was supported by the PROBADO project (http://www.probado.de/, grant INST 11925/1-1) and the ARMADA project (grant CL 64/6-1).

[1] While "multimodal" refers to the perception of music through different modalities (user perspective), "multimedia" corresponds to the different media types (data perspective).

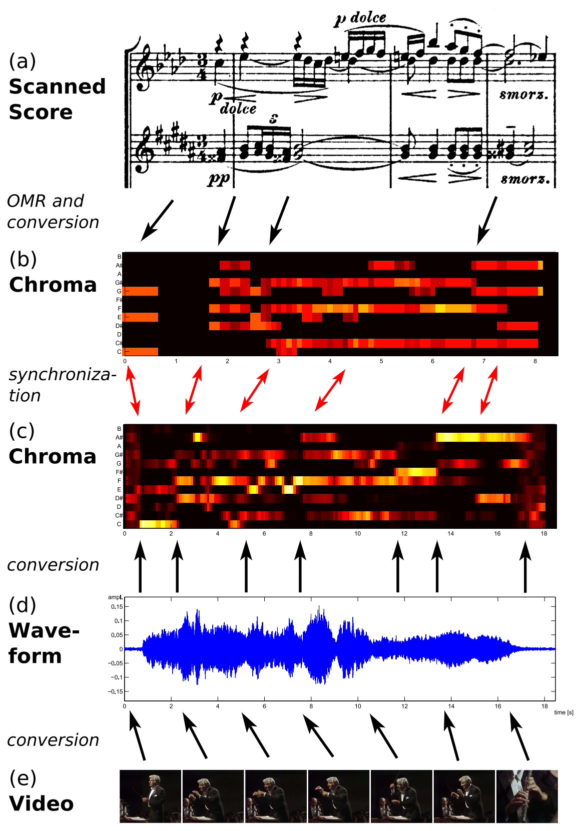


Figure 1. Illustration of the scan-video synchronization, using the first measures of the 2nd movement of Liszt's Faust Symphony. (a) Scanned sheet music. (b) Chromagram of the sheet music. (c) Audio chromagram. (d) Audio track extracted from the video. (e) Music Video. The scan-video synchronization (double-headed arrows) is obtained by chromagram-alignment, see Section 2.2.

face SLAVE and in particular the integration of video documents into this system. The paper closes in Section 5 with prospects on future work.

2. UNDERLYING TECHNIQUES

In this section, we describe the methods needed to process, match and align various types of music documents. The basic idea of the presented processing methods is to transform all music document types into a common feature representation, which allows for direct comparison and alignment independent of the input document types. In this context, chroma-based music features have proven to be a good mid-level representation [10, 11]. At first, the input documents are transformed into sequences of 12-dimensional chroma vectors, where each vector represents an energy distribution over the twelve pitch classes of the equal-tempered scale. In Western music notation, the chroma are commonly indicated by the pitch spelling attributes C, C$^{\#}$, D, ..., B. By considering short-time statistics, these chroma features are transformed into the robust and scalable CENS features, see [12] for details. As an example, the CENS sequences for an extract of scanned sheet music and the CENS features of the corresponding video section are displayed in Figure 1 (b) and (c). Throughout this paper, chroma features with a sample rate of 10 Hz are applied, whereas CENS features of different sample rates are generated and used for alignment and indexing purposes.

2.1 Deriving Chroma-based Features

To time-align two music documents (e.g., sheet music and a video recording) describing the same piece of music, both documents are transformed into CENS features.

The transformation of scanned sheet music into CENS features requires several processing steps, see [13]. At first, using standard software for optical music recognition (OMR), the scanned score data is analyzed and transformed into musical note parameters. Subsequently, based on the gained pitch and timing information, the chroma features can essentially be computed by identifying pitches that belong to the same chroma class. The CENS sequences are gained from these features as previously described. During the feature extraction, a constant tempo of the piece of music represented by the scanned sheet music is assumed.

For a detailed description on methods for CENS feature generation of audio recordings, we refer to the literature [10, 12]. In principle, in our application the audio signal is transformed into chroma features by using short-time Fourier transforms in combination with binning strategies.

To enable the generation of CENS features from video files, the audio track of the video recording is extracted and the feature computation for audio recordings is applied.

2.2 Music Synchronization

Figure 1 gives an example of the alignment procedure for the scanned sheet music and a video recording of the first measures of the 2nd movement of Liszt's Faust Symphony. As described before, the first step for the synchronization of two music documents is, to convert both into a common and meaningful feature representation. Based on these features, multiscale dynamic time warping techniques (MsDTW) are employed to determine the synchronizations between music documents. The essential idea of MsDTW is to recursively compute alignment paths for coarse feature resolutions and project them to the next higher resolution level, where they subsequently are refined. Further details on the MsDTW method are available, e.g., in [14, 15].

During the described synchronization, we assume that the music documents match with respect to their musical structure so that only local and global tempo-variations need to be considered. In Section 3 we go into details on how to deal with structural differences during score-audio and score-video alignment. For information on synchronization of structurally differing audio recordings, see [16].

3. SEMI-AUTOMATIC DATA PROCESSING

To allow for a fast and user-friendly generation of all data files used by the SLAVE system (Section 4), the automation of all required computing steps is desirable. As a first step

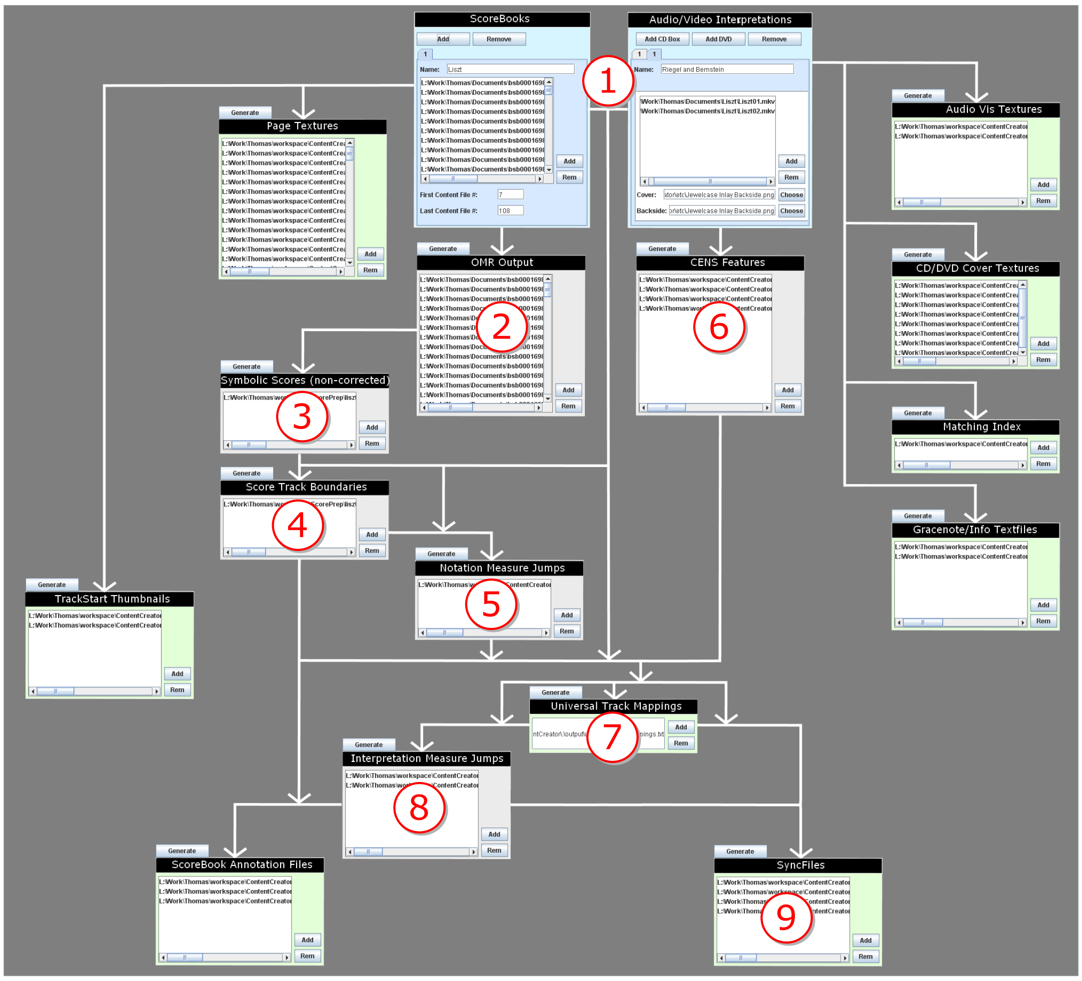


Figure 2. The *ContentCreator* interface for semi-automatic data processing.

towards full automation, we present the *ContentCreator* interface, see Figure 2, for the generation of synchronization information, indexing structures, and further data types for music collections consisting of score scans, audio recordings, and videos.

The *ContentCreator* interface aims at providing an intuitive GUI to support the process chain required to generate all metadata used by the SLAVE system. Within this context, metadata refers to data files containing information that ranges from indexing and synchronization structures to score and CD cover images used by SLAVE for visualization purposes. As shown in Figure 2, the workflow is divided into several interdependent stages. The generation of the results for each stage can be triggered individually. The integrated arrows help to clarify the dependencies between the various stages and also display the different paths for the generation of metadata. Due to the selected division of the workflow into individually triggered stages, the manual manipulation of intermediate files before continuing to the next stage is possible. At the moment, some stages merely generate default or, due to the error-prone OMR, erroneous data files. At these points, a manual rework is essential for a successful workflow. Further details on this issue are addressed in Section 3.2.

During the usage of the *ContentCreator*, three different types of data files can be distinguished. The input files (label 1 in Figure 2) mark the starting point of the process chain and currently consist of scanned pages of a music book, audio recordings, and videos. The second are the intermediate data types (labels 2 – 6). These files are required for the process chain but do not contain information directly used by the SLAVE system. By contrast, the last type of files (labels 7 – 9 and unlabeled boxes) contains metadata. The output stages not further mentioned in this paper generate metadata like texture data for rendering the sheet music pages, information on length and name of the audio and video tracks, and data structures for content-based retrieval.

Methods to save and restore the current state as well as the possibility to export the created metadata for runtime usage with SLAVE are provided.

3.1 Workflow for semi-automatic music alignment

In the following, we describe the chain of stages involved in the process of time-aligning a music book to various interpretations (video and audio).

As first step, the input data needs to be selected and loaded to the application (label 1). The *ContentCreator* interface enables the management of arbitrary numbers of audio and video interpretations of a piece of music (right-hand box) and the synchronization of these interpretations to a scanned music book representing the same piece of music (left-hand box).

To extract the score information from the scanned score pages, OMR is performed (label 2) and the resulting data are subsequently merged into a single *SymbolicScore* file (label 3). This file thereby contains various music information such as note events, key signatures, time signatures, staff information, accidentals, and information on instrumentation and transposition, required for the generation of chroma-based features.

When aiming at the alignment of a whole music book to a set of video or audio recordings, the individual scanned pages of the music book need to be related to the different tracks of the music book. Therefore in the next step, a file containing information on the tracks contained in the music book is generated (label 4). Besides the musical information stored in the *SymbolicScore* files, generated in stage 3, a score also contains information on repetitions and jumps. This data is extracted from the OMR output and saved in the next stage (label 5). Together with the jump information of the video and audio interpretations, this data contributes fundamentally to the success of the synchronization process. On generating the CENS features of the scanned sheet music, the given jump information is employed to ensure structural accordance with the respective features of the video and audio recordings. Subsequently, the CENS features of all loaded audio and video files are generated as described in Section 2.1 (label 6).

Prior to the execution of the synchronization, there are two more steps to be conducted. After splitting the music book into various tracks in stage 4, these need to be mapped to the audio and video files of each loaded interpretation (label 7). Currently, manual rework is required, if the order of the videos or the audio tracks of the interpretation does not coincide with the track order of the music book. There are already proposals for a feature based, automatable creation of mapping information, which can be integrated into the *ContentCreator* interface, see [1]. Finally, the jump and repetition instructions extracted from the score scans might not be consistent with the repetitions and jumps actually performed in the specific audio or video recording. Therefore, the last stage (label 8) before the synchronization consists of the generation of this data for all loaded video and audio interpretations.

After passing all stages described before, the synchronization information for the music book and the loaded video and audio interpretations are computed (label 9), using the MsDTW approach mentioned in Section 2.2. The CENS features of the music book are computed on demand, considering the structural information of the audio or video track used for the current synchronization process.

3.2 Reworks during the workflow

The individual stages of the presented workflow are automatized as far as possible. For stages, where currently no adequate computational methods exist to enable automation, default data files, which need to be reworked by the user, are generated. The user can modify those data files or can import previously generated data files into the currently reworked stage. At the moment, parts of the *SymbolicScore* file (transposition information for the contained instruments, label 3) and the interpretation specific jump and repetition information for video and audio recordings (label 8) might need manual correction. In addition, in stage 3, 4, 5, and 7 manual adjustments might be required for complex pieces of music or low quality music book scans.

To extend the workflow managed by the *ContentCreator* to larger music collections, the applied techniques are currently integrated into the PROBADO library service system build up at the Bavarian State Library, see [1].

4. THE SLAVE SYSTEM

Recently, various computer tools were created to enable the management and presentation of multimedia music collections. However, so far those tools mostly concentrate on sheet music and audio recordings. In this section, we want to present the SLAVE system, which aims at a user-friendly and holistic exploration of music in a multimodal manner.

SLAVE is based upon the *SyncPlayer* system, presented in [7]. The *SyncPlayer* offers – besides basic audio player functionalities – the possibility of adding plug-ins for multimodal music presentation and audio analysis (e.g., a plug-in for the visualization of the musical structure of the current piece of music). SLAVE provides a renewed GUI and includes some of the techniques already available in the *SyncPlayer* as well as some new features. The new framework is envisioned as user interface for the library service system set up at the Bavarian State Library as part of the PROBADO project. First system developements towards SLAVE were recently presented in [1].

The framework consists of several user interfaces for multimodal music presentation, navigation and content-based retrieval. The central component is the *ScoreViewer* interface shown in Figure 3, which offers the visualization of the scanned pages of the underlying music book. When audio playback is started, the corresponding measures within the sheet music are highlighted based on the synchronization information created by the *ContentCreator* system described in Section 3. Some additional features of the *ScoreViewer* are automatic page turning during playback, navigation within the music book, and user-friendly music retrieval based on the query-by-example paradigm. The latter is implemented by enabling the user to select a region within the sheet music using the mouse pointer. The issued query is processed by determining the corresponding audio clipping of the currently active interpretation and performing content-based music retrieval. For details on the employed matching and indexing techniques, we refer to [17].

There might exist several interpretations of the same piece of music, which match to the given music book. The name of the currently active audio interpretation, as well as an icon showing a corresponding CD cover are displayed in the upper left corner of the *ScoreViewer* interface. To seamlessly switch to a different interpretation, a list containing information on all loaded interpretations is available by clicking on the current cover icon.

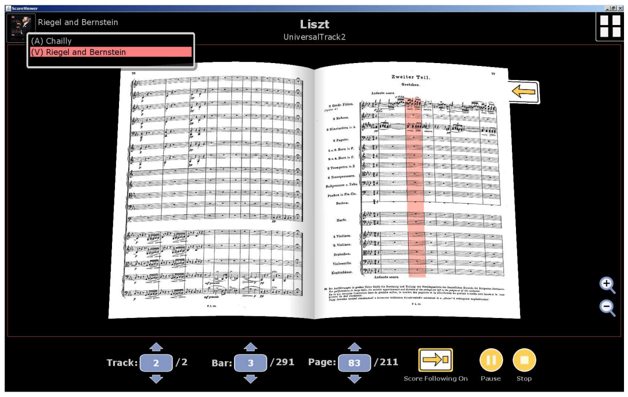


Figure 3. The *ScoreViewer* interface for multimodal music presentation and navigation. During video or audio playback the corresponding musical measures within the sheet music are highlighted. A smooth change between different interpretations of a piece of music is possible.

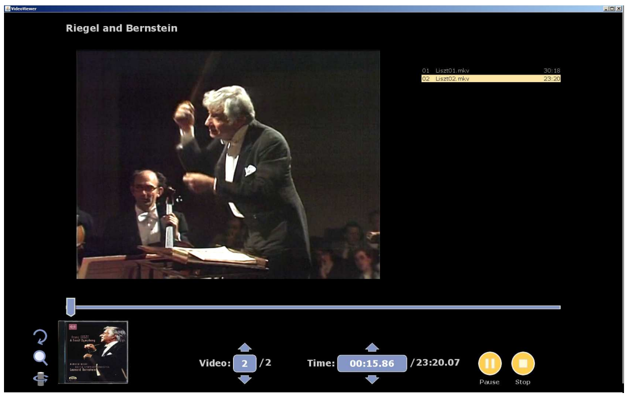


Figure 4. The *VideoViewer* shows the currently played video of the chosen video interpretation and allows browsing within the video as well as within the list of video files of the interpretation.

To include music videos, we added the *VideoViewer* interface which offers basic video player functionalities, see Figure 4. As for the audio recordings, during video playback, the corresponding measures of the sheet music are highlighted. The smooth change between different video and audio interpretations of the piece of music via the *ScoreViewer* interface enables the comparison of different music document types. Furthermore the content-based music query was extended to allow the usage of video extracts as queries and to include video interpretations in the search indices and result lists.

Although, simultaneously looking at both the *Video-* and the *ScoreViewer* might be hard for the user, having a time aligned view on the score constitutes several advantages. In longer video recordings, it might be cumbersome to search for a specific point in time within the video, whereas using the score for navigation is easier and faster. Furthermore with respect to conductings of classical music, it might be of interest to compare recordings of several different conductors at the same musical position. Using the capability of smooth switching between interpretations or performing a content-based query using the sheet music can help to facilitate these tasks.

4.1 Further Extensions

In this section we want to give a preview on further functionalities and interfaces which will be added to SLAVE for a holistic music presentation.

4.1.1 LyricsViewer

Currently, SLAVE enables the combined presentation of scanned music books, audio recordings and videos. However, text documents like libretti of operas and lyrics of song cycles are additional ingredients of digital music collections. Therefore, our current work aims at the integration of a *LyricsViewer*. Foundations for this development are the previously introduced Karaoke Display and Lyrics Seeker of our *SyncPlayer* system [18]. As for the *ScoreViewer*, the currently vocalized words of the song text will be highlighted during video or audio playback. Additionally, search mechanisms based on the lyrics will be supported by the *LyricsViewer* interface.

4.1.2 InterpretationSwitcher

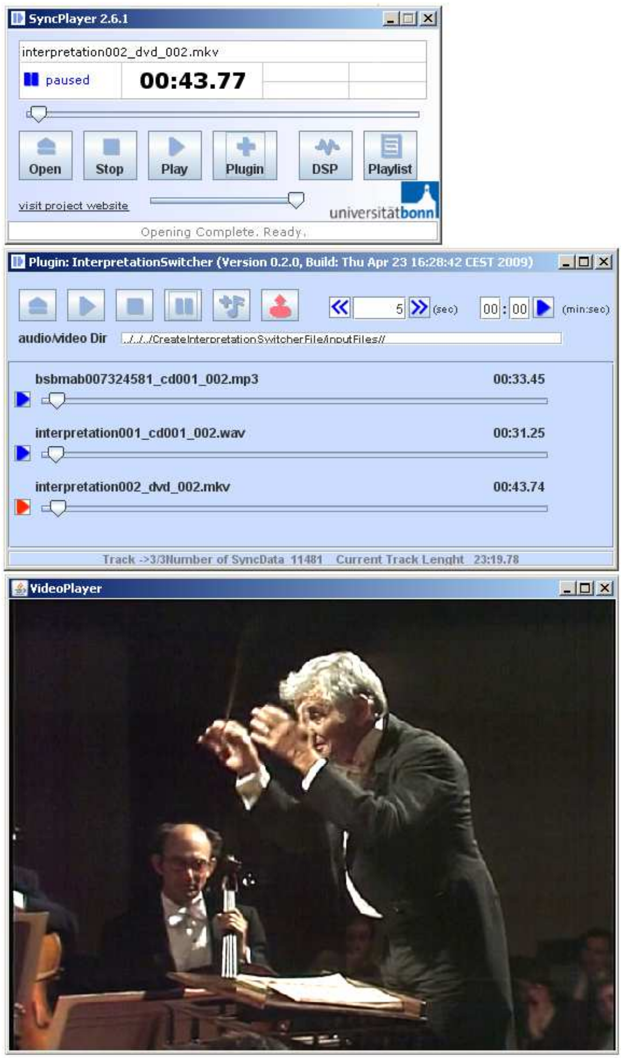
In addition to SLAVE, we enhanced the *SyncPlayer* system [7] (see Figure 5), which is basically a predecessor of our new system, to support video documents.

First, the audio player component of the *SyncPlayer* was modified to allow for the playback of video files and for the inclusion of videos into playlists. Additionally we implemented the *InterpretationSwitcher* plug-in which is basically an extension of the *AudioSwitcher* plug-in [7]. The *InterpretationSwitcher* offers the possibility to switch between different audio and newly video interpretations of a piece of music during playback. On changing to a different interpretation the current playback position in the piece of music is retained and the playback seamlessly continues within the chosen interpretation.

Figure 5 shows an example of two audio interpretations and one video of the second movement of Liszt's Faust Symphony. The sliders enable the user to change to an arbitrary playback position within one of the interpretations. The playback symbols to the left of the sliders mark, which interpretation is currently playing and enable smooth switching between the interpretations. First steps towards the integration of similar functionalities to SLAVE are presented in [19].

4.2 Applications of SLAVE and the VideoViewer

The availability of video documents in the framework presented in this paper offers several advantages for musicians, musicologists, music lovers, and others. As mentioned before, prospective conductors might be able to use the proposed system to compare the work of different conductors within the same piece of music. Looking at more complex pieces of music – as orchestral works – comparing the orchestration and the arrangement of the orchestra might be of interest. Thinking of operas even stage designers, make-up artists and costume designers might have an



interpretation002_dvd_002.mkv
paused 00:43.77
Opening Complete. Ready.

5 (sec) 00 : 00 (min:sec)
audio/video Dir ../../../CreateInterpretationSwitcherFile/inputFiles//

bsbmab007324581_cd001_002.mp3 00:33.45
interpretation001_cd001_002.wav 00:31.25
interpretation002_dvd_002.mkv 00:43.74

Track ->3/3Number of SyncData 11481 Current Track Lenght 23:19.78

Figure 5. *SyncPlayer* with *InterpretationSwitcher* plug-in and video player. The *InterpretationSwitcher* enables the selection of several video or audio interpretations of the same piece of music. Using the sliders and the playback symbols on the left, the user can smoothly switch between and browse within the interpretations.

interest to compare different stagings. The possibility of smooth changes between various interpretations and score based navigation offers thereby significant support. Furthermore, looking at the area of dance, choreographers and dance theorists might benefit from these tools as well.

We therefore hope to experience great acceptance of the newly integrated video capabilities of our framework by the various target groups.

5. CONCLUSIONS

In this paper, we reported on a new user interface for semi-automatic processing of video, audio and score collections to generate synchronization information, indexing structures and metadata required for a holistic presentation of these heterogeneous music collections. We also presented the multimodal music management framework SLAVE and the inclusion of music videos as further music document type. Especially, we introduced the possibility of video playback and simultaneous score highlighting.

Besides the extensions described in this paper, the development of new functionalities and interfaces especially for video documents are envisioned, e.g., after the synchronization of various videos, during the playback of one reference video, the other sources can be shown time aligned to this video. For playback, only the audio track of the reference is used. This type of application will enable a more convenient and direct comparison of video interpretations.

6. REFERENCES

[1] F. Kurth, D. Damm, C. Fremerey, M. Müller, M. Clausen: “A Framework for Managing Multimodal Digitized Music Collections,” *Proc. ECDL, Aarhus, Denmark*, 2008.

[2] A. D’Aguanno, G. Vercellesi: “Automatic Music Synchronization Using Partial Score Representation Based on IEEE 1599,” *Journal of Multimedia*, Vol. 4, No. 1, pp. 19-24, 2009.

[3] A. Klapuri, M. Davy: *Signal Processing Methods for Music Transcription*. Springer, New York, 2006.

[4] B. Pardo: “Music Information Retrieval,” *Special Issue, Commun. ACM*, Vol. 49, No. 8, pp. 28-58, 2006.

[5] BMWi: “CONTENTUS,” *http://theseus-programm.de/en-us/theseus-application-scenarios/contentus/default.aspx*.

[6] C. Landone, J. Harrop, J. Reiss: “Enabling Access to Sound Archives through Integration, Enrichment and Retrieval: the EASAIER Project,” *Proc. ISMIR, Vienna, Austria*, 2007.

[7] C. Fremerey, F. Kurth, M. Müller, M. Clausen: “A Demonstration of the SyncPlayer System,” *Proc. ISMIR, Vienna, Austria*, 2007.

[8] A. Barat, L. A. Ludovico: “Advanced Interfaces for Music Enjoyment,” *Proc. AVI, Napoli, Italy*, 2008.

[9] J. W. Dunn, D. Byrd, M. Notess, J. Riley, R. Scherle: “Variations2: Retrieving and Using Music in an Academic Settings,” *Special Issue, Commun. ACM*, Vol. 49, No. 8, pp. 53-58, 2006.

[10] M. A. Bartsch, G. H. Wakefield: “Audio Thumbnailing of Popular Music using Chroma-based Representations,” *IEEE Trans. on Multimedia*, Vol. 7, No. 1, pp. 96-104, 2005.

[11] N. Hu, R. Dannenberg, G. Tzanetakis: “Polyphonic Audio Matching and Alignment for Music Retrieval,” *Proc. IEEE WASPAA, New Paltz, NY*, 2003.

[12] M. Müller: *Information Retrieval for Music and Motion*. Springer, Berlin, 2007.

[13] F. Kurth, M. Müller, C. Fremerey, Y. Chang, M. Clausen: “Automated Synchronization of Scanned Sheet Music with Audio Recordings,” *Proc. ISMIR, Vienna, Austria*, 2007.

[14] S. Salvador, P. Chan: “FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space,” *Proc. KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[15] M. Müller, H. Mattes, F. Kurth: “An Efficient Multiscale Approach to Audio Synchronization,” *Proc. ISMIR, Victoria, CDN*, 2006.

[16] M. Müller, S. Ewert: “Joint Structure Analysis with Applications to Music Annotation and Synchronization,” *Proc. ISMIR, Philadelphia, USA*, 2008.

[17] F. Kurth, M. Müller: “Efficient Index-based Audio Matching,” *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16, No. 2, pp. 382-395, 2008.

[18] M. Müller, F. Kurth, D. Damm, C. Fremerey, M. Clausen: “Lyrics-based Audio Retrieval and Multimodal Navigation in Music Collections,” *Proc. ECDL, Budapest, Hungary*, 2007.

[19] D. Damm, C. Fremerey, F. Kurth, M. Müller, M. Clausen: “Multimodal Presentation and Browsing of Music,” *Proc. ICMI, Chania, Greece*, 2008.

LYRIC EXTRACTION AND RECOGNITION ON DIGITAL IMAGES OF EARLY MUSIC SOURCES

John Ashley Burgoyne **Yue Ouyang** **Tristan Himmelman**
Johanna Devaney **Laurent Pugin** **Ichiro Fujinaga**

Centre for Interdisciplinary Research in Music and Media Technology
McGill University
Montréal, Québec, Canada
`{ashley,devaney,laurent,ich}@music.mcgill.ca`
`{yue.ouyang,tristan.himmelman}@mail.mcgill.ca`

ABSTRACT

Optical music recognition (OMR) is one of the most promising tools for generating large-scale, distributable libraries of musical data. Much OMR work has focussed on instrumental music, avoiding a special challenge vocal music poses for OMR: lyric recognition. Lyrics complicate the page layout, making it more difficult to identify the regions of the page that carry musical notation. Furthermore, users expect a complete OMR process for vocal music to include recognition of the lyrics, reunification of syllables when they have been separated, and alignment of these lyrics with the recognised music. Unusual layouts and inconsistent practises for syllabification, however, make lyric recognition more challenging than traditional optical character recognition (OCR). This paper surveys historical approaches to lyric recognition, outlines open challenges, and presents a new approach to extracting text lines in medieval manuscripts, one of the frontiers of OMR research today.

1. INTRODUCTION

Researchers in music information retrieval (MIR) have gradually been building bigger databases of music that will enable large-scale computational musicology. One tool to expedite the development of such databases for older music is optical music recognition (OMR), the musical analogue to optical character recognition (OCR). When developing databases of vocal music, however, OMR alone is not enough: lyrics need to be recognised and stored along with the music, and because of certain particularities of musical notation, standard OCR tools are often insufficient for this task. Moreover, lyrics complicate the page layout, making it more difficult to conduct the basic image processing necessary to feed the OMR and OCR pipelines

Vocal music predominates among medieval music manuscripts, copied by hand from the ninth through the sixteenth centuries, as well as early printed music from the fifteenth and sixteenth centuries. These sources pose still more challenges for OMR and lyric recognition. Due to the relatively loose document production techniques of the time, page layouts can be highly non-standard and oriented more for display than consumption. Furthermore, as a result of ageing, early music documents are often in physically poor condition. Typical problems include non-uniform illumination, stains, and irregular page shape. Ink frequently has bled through from the reverse side of the page or shows through as a result of high-contrast microfilm photography. All of these degradations can inhibit the performance of segmentation (identifying the regions of the image that correspond to musical notation, lyrics, or other elements) and recognition (interpreting image shapes as musical notes or letters) [1]. Finally, scanning conventions for early documents themselves can require extra preprocessing to remove elements like rulers and colour bars before images are even sent to an OMR system [2].

This paper outlines some previous approaches to lyric recognition, highlights the open challenges, especially with respect to early documents, and presents two tools we have developed to facilitate work with digital images of early music: a new lyric editor for the Aruspix OMR package [3] and a new technique for extracting lyric lines from digital facsimiles of medieval manuscripts.

2. BACKGROUND

OMR systems have been working to handle lyrics for some time, and most systems share common challenges. One is layout analysis: how can these systems determine which regions of a page carry music and which lyrics? Some systems rely on heuristics based on projections, run lengths, and other local image features, generally seeking to extract the music first and define the remaining regions as text [3,4]; others run OCR first, using those regions where the system successfully identifies letters as lyric regions [6]. Once the regions have been separated, most systems then send the music regions and lyric regions to parallel OMR and OCR pipelines for fuller processing.

OCR itself is difficult for lyrics, however, because of inconsistent syllabification [6]. When sung over many notes, lyric words are usually (but not always) separated into their constituent syllables, which poses problems for traditional OCR methods that rely on statistical language models with word dictionaries [7,8]. The need for syllabicated dictionaries limits the number of languages available for OCR and increases the number of unrecognised inflections [9]. Early documents add another twist on account of archaic spellings and scribal abbreviations [10]. Some systems sidestep these concerns by noting that the output of standard OCR on these documents is often good enough for users to be able to locate documents in a database [11], but for archival purposes, the ultimate goal of these systems is to have lyrics that are complete and accurate.

3. EDITING LYRICS WITH ARUSPIX

No automatic lyric recognition system will ever be perfect, however, and because digital archivists are among the primary target users for recognition software, these mistakes need to be corrected. These corrections can incur significant labour expenses in the absence of efficient software tools, to the point that, as we have seen in [11], sometimes they are not made at all. From the perspective of software design, then, an integrated editor for lyrics is a useful adjunct to any OMR package for vocal music. We chose to extend Aruspix, an open-source OMR application that already includes an integrated editor for musical symbols, to include a new editor for lyrics.

In the model underlying our lyric editor, lyrics are associated with notes in a many-to-one relationship, i.e., each musical note can be associated with one or more lyric elements. The user can modify these relationships and the lyrics themselves with a convenient graphical interface that pairs the editing region with the analogous portion of the original music image, as illustrated in Figure 1. Similar to the music editor in Aruspix, our lyric editor operates in one of two modes: Lyric Editing, which allows users to change the location of lyrics and their associated notes, and Lyric Insertion, which allows users to enter new lyrics and modify existing ones. Using this editor, any recognition errors can be identified and correctly quickly, which reduces labour costs for archival projects and facilitates rapid production of ground truth for researchers.

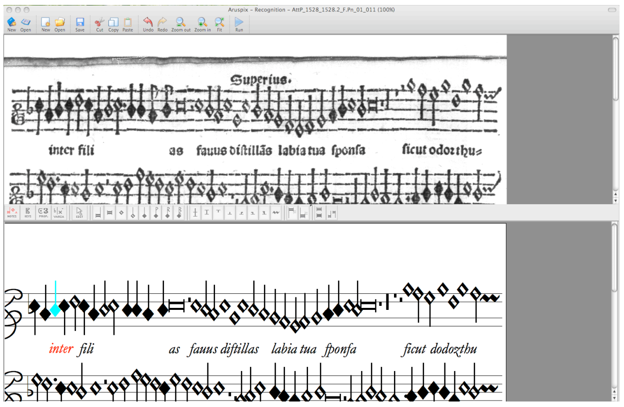

Superius.
inter fili as fauus diſtillas labia tua ſponſa ſicut odorthu
inter fili as fauus diſtillas labia tua ſponſa ſicut dodozthu

Figure 1. Screenshot of the lyric editor in Aruspix. The original image appears in the top pane. The lower pane contains the lyric editor. Each lyric is linked to a note; in this case, the word *inter* is linked to the third note on the top staff.

4. A METHOD FOR LYRIC-LINE EXTRACTION

Text-line detection is the first step in most OCR systems, and a great number of approaches have been developed for different types of documents: projection-based methods, grouping methods, and the Hough transform, among others [12]. Our lyric-line extraction algorithm, illustrated in Figure 2, is derived from an approach to text-line detection that is optimal for undulating lines [13,14]. Although lyrics are laid out less consistently than the text in text-only documents, they are almost always grouped along straight horizontal lines. After the removal of the staves, these straight lyric lines contrast sharply with the undulating lines, also detected by these algorithms, that trace the path of musical notes. More specifically, if baselines of both lyrics and notes are generated, the former will be almost straight while the latter will be highly curved and undulating. Thus, a line of lyrics can be extracted with confidence if many straight segments are found along the line. This assumption is fairly safe when there are a good number of words within a lyric line, e.g.,

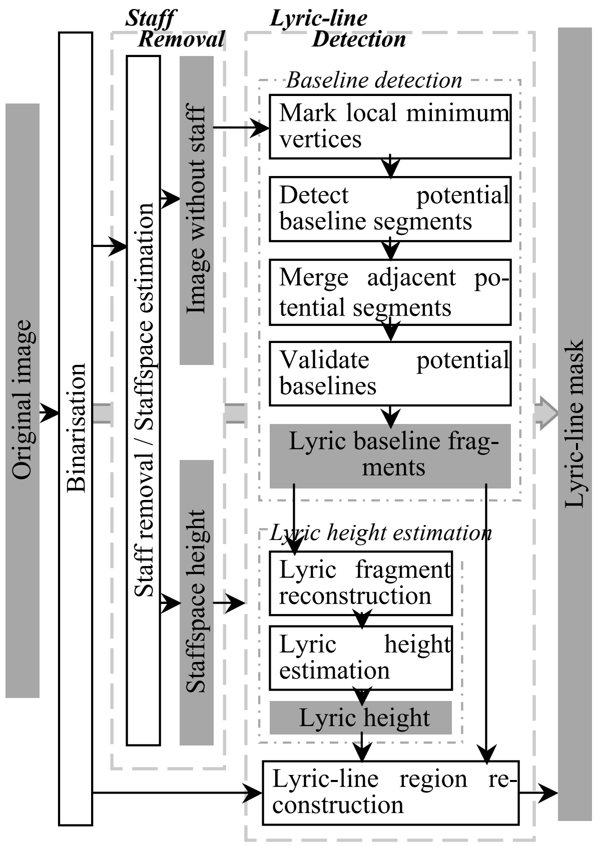
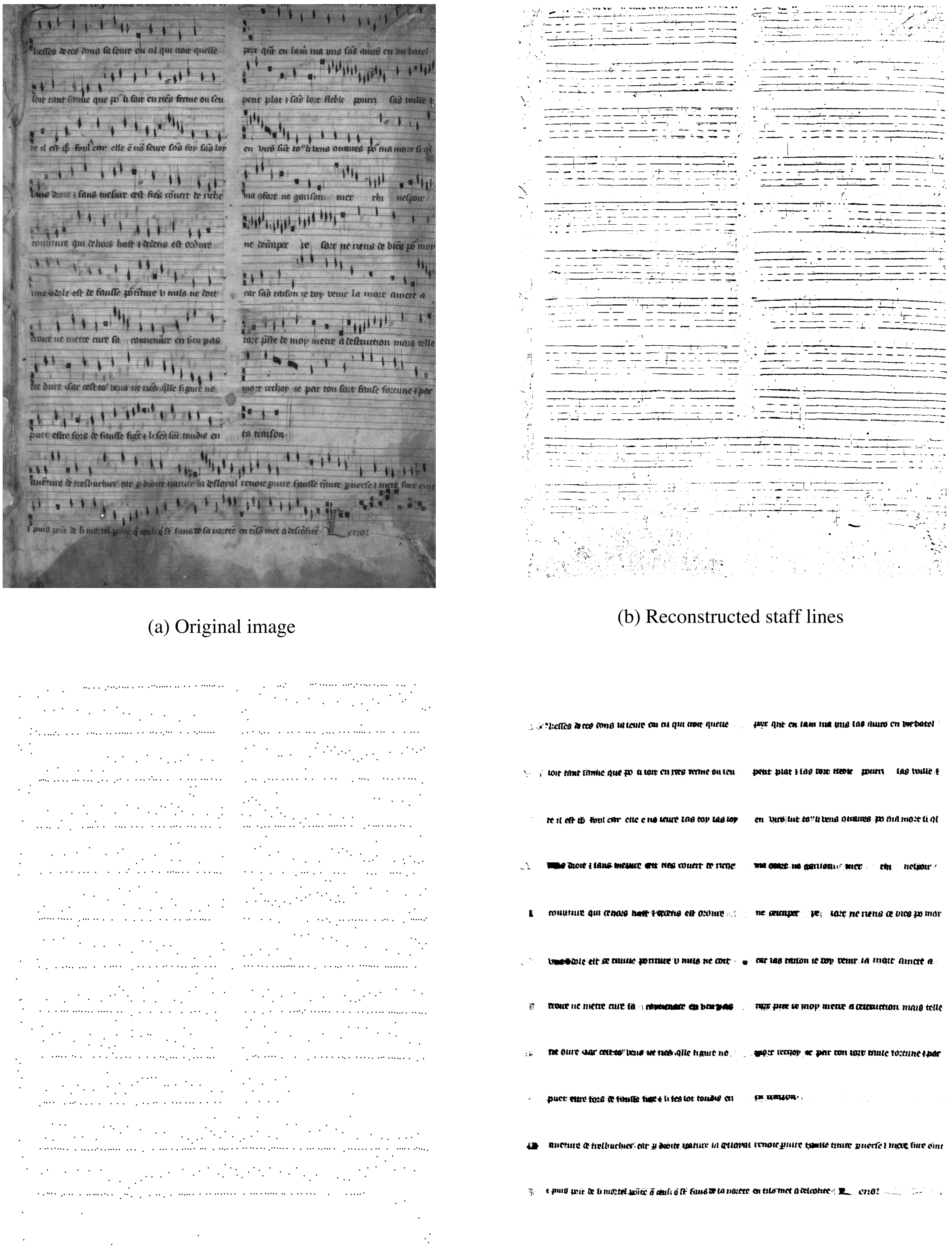

Staff Removal
Lyric-line Detection
Original image
Binarisation
Staff removal / Staffspace estimation
Image without staff
Staffspace height
Baseline detection
Mark local minimum vertices
Detect potential baseline segments
Merge adjacent potential segments
Validate potential baselines
Lyric baseline fragments
Lyric height estimation
Lyric fragment reconstruction
Lyric height estimation
Lyric height
Lyric-line region reconstruction
Lyric-line mask

Figure 2. Workflow for extracting lyric lines. During preprocessing, the image is binarised and staves are removed. A three-step extraction process follows for lyrics.


(a) Original image

(b) Reconstructed staff lines

(c) Local minima after staff-removal.

(d) Extracted lyric lines

Figure 3. Results of text-line extraction. Image (a), the original image, illustrates some of the layout challenges inherent to medieval documents. Reconstructed staff lines from our staff-removal step appear in (b). The “local minima” of each connected component appear in (c), and the final output of lyric-line extraction is in (d).

the lyric lines in Figure 3(a), but it can miss lyrics when they are arranged particularly sparsely.

Before detecting the lyric lines, the images must be preprocessed by global deskewing and staff removal. Because music recognition takes place in a distinct, parallel process, we developed a new staff-removal algorithm that damages noteheads slightly but removes staff lines more reliably in degraded documents than traditional techniques. The technique is in the style of the median-filter approach in [15]. Local horizontal projections and vertical run-length coding are used to estimate staff-space height and staff height. These values are used to construct a directional median-filter window in the form of a thin vertical bar. The bar is set to be tall enough to remove staff lines while remaining short enough to preserve notes and lyrics. Because the window is thin, this filter is able to remove curved staff lines effectively. Unlike some approaches, our algorithm is also able to locate the lyrics in documents without staff lines, e.g., early Aquitanian chant manuscripts; for such images, we obviously skip the staff-removal filter. A sample of staff lines extracted by this algorithm appears in Figure 3(b).

Following preprocessing, our method has three broad steps: baseline detection, estimation of lyric height, and reconstruction of lyric regions.

Baseline estimation begins by binarising the image (classifying pixels as foreground or background) and identifying all connected components of foreground pixels. Every component is represented by groups of vertices constituting the "local minima" of the component: the component is broken into vertical strips that are about as wide as a staff space is high, and the point closest to the bottom of the page is retained as a local minimum for the component. Figure 3(c) illustrates a chart of these local minima for one of our test images.

We then "connect the dots" to extract baselines. Each unconnected local minimum is connected to its nearest neighbour with respect to a quadratic thresholding function that weights the distance between the two points and the angle between them, privileging short distances and approximately horizontal lines (see Figure 4). This thresholding function is rather strict, and so unless lyrics are packed densely across a line, the connected baseline segments usually underscore individual words or letters rather than longer lyric lines. A second pass with a more permissive thresholding function is made to connect sufficiently long segments extracted from the first pass. Finally, all connected segments are validated to ensure that they contain a reasonable number of local minima and have an overall horizontal slope. This final validation step removes any segments that might arise from musical lines with repeated notes or sequences of notes that are close in pitch space.

In extracting the baselines, we discard all information about the height of the lyrics, and so it is necessary to reconstruct lyric regions from the baselines. The process is the inverse of marking the local minima: for each local minimum identified in a baseline, the corresponding connected component is included in the lyric region. Some of these connected components, however, include non-lyric elements, especially when lyric elements close to a staff overlap with low notes. In order to compensate for these problems, an upper bound for lyric height is chosen based on the size of a staff space. Connected components above this upper bound are cropped.

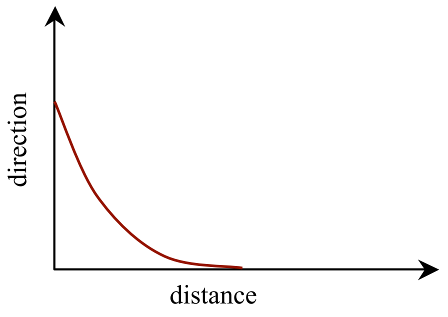


Figure 4. Thresholding function for connecting local minima of the connected components. The boundary is quadratic and privileges horizontal directions.

After the lyric height has been estimated, complete lyric-line regions are generated from the baselines. In the absence of other information about page layout, the lyric lines are extended from the left-most to the right-most points of the page. A simple peak-picking process along the y-axis groups extracted baselines that are part of common lyric lines; the peak picking is tuned with the estimated lyric height as described above. In our experiments, a simple linear regression on the baselines combined with estimated lyric height yields good lyric regions in most cases, although when the pages are non-linearly skewed, higher-order polynomials are necessary.

We tested our algorithm on a set of 40 images from the Digital Image Archive of Medieval Music (DIAMM) chosen for their particularly challenging layouts [16]. A sample image is presented in Figure 3(a), and the output of our algorithm on this image is presented in Figure 3(d). Note in particular that the algorithm proved robust to the two-column format. Despite the challenging layouts and (in some cases) considerable document degradation, our algorithm was able to recall 80.4 percent of the text lines with 88.4 percent precision overall. For clean images, however, the results are nearly perfect: there was only one recall error across our 12 cleanest samples with 100-percent precision.

5. SUMMARY AND FUTURE WORK

Automatic lyric recognition is challenging for any musical document on account of varied page layouts and inconsistent syllabification. This challenge is exacerbated for early music documents, which suffer from an even wider variety of layouts and, often, significant degradation. We have developed software that helps researchers generate ground truth quickly for early music documents and that helps archivists correct any errors in automatic recognition with minimum labour cost and musical sense. We have also extended approaches for document image analysis for historical text documents to be sensitive to the particularities of music manuscripts, resulting in reliable text-line extraction from a number of difficult older documents.

This work is in progress, and we are currently using data we have extracted from these systems to experiment with full-scale recognition using features similar to those in [7] and a variety of labelling models, including hidden Markov models [17] and conditional random fields [18]. The long-term goal of our project is to produce a set of extensions to Aruspix that will make it a fully functional OMR system for early vocal music, including OMR for medieval plainchant notations and automatic lyric recognition for the most common languages of the period.

6. ACKNOWLEDGEMENTS

The authors would like to thank the Canada Foundation for Innovation (CFI) and the "Image, Text, Sound, and Technology" program of the Social Sciences and Humanities Research Council of Canada (SSHRC) for supporting this research.

7. REFERENCES

[1] Pinto, J. C., P. Vieira, M. Ramalho, M. Mengucci, P. Pina, and F. Muge. 2000. Ancient music recovery for digital libraries. In *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, 24–34.

[2] Ouyang, Y., J. A. Burgoyne, L. Pugin, and I. Fujinaga. 2009. A robust border-detection algorithm with application to medieval manuscripts. In *Proceedings of the International Computer Music Conference*.

[3] Pugin, L. 2006. Optical music recognition of early typographic prints using hidden Markov models. In *Proceedings of the 7th International Conference on Music Information Retrieval*, 53–6.

[4] Choudhury, G. S., T. DiLauro, M. Droettboom, I. Fujinaga, and K. MacMillan. 2001. Strike up the score: Deriving searchable and playable digital formats from sheet music. *D-Lib Magazine* 7 (2).

[5] Jones, G., B. Ong, I. Bruno, and K. Ng. 2007. Optical music imaging: Music document digitization, recognition, evaluation. In *Interactive Multimedia Music Technologies*, ed. K. Ng and P. Nesi, 50–79. Hershey, PA: IGI Global.

[6] Droettboom, M. 2004. Beyond transcription: Case studies in special document analysis requirements. In *Proceedings of the International Workshop on Document Image Analysis for Libraries*.

[7] Vinciarelli, A., S. Bengio, and H. Bunke. 2004. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (6): 709–20.

[8] Steinherz, T., E. Rivlin, and N. Intrator. 1999. Offline cursive script word recognition: A survey. *International Journal on Document Analysis and Recognition* 2 (2–3): 90–100.

[9] Wingenroth, B., M. Patton, and T. DiLauro. 2002. Enhancing access to the Levy sheet music collection: Reconstructing full-text lyrics from syllables. In *Proceedings of the ACM-IEEE Joint Conference on Digital Libraries*, 308–9.

[10] Ernst-Gerlach, A., and N. Fuhr. 2007. Retrieval in text collections with historic spelling using linguistic and spelling variants. In *Proceedings of the ACM-IEEE Joint Conference on Digital Libraries*, 333–41.

[11] Diet, J., and F. Kurth. 2007. The Probado music repository at the Bavarian State Library. In *Proceedings of the 8th International Conference on Music Information Retrieval*, 501–4.

[12] Likforman-Sulem, L., Z. Abderrazak, and T. Bruno. 2007. Text line segmentation of historical documents: A survey. *International Journal on Document Analysis and Recognition* 9 (2): 123–38.

[13] Feldbach, M. and K. D. Tonnies. 2001. Line detection and segmentation in historical church registers. In *Proceedings of the 6th International Conference on Document Analysis and Recognition*, 743–7.

[14] Pu, Y., and Z. Shi, Z. 1998. A natural learning algorithm based on Hough transform for text lines extraction in handwritten documents. In *Proceedings of the 6th International Workshop on Frontiers in Handwriting Recognition*, 637–46.

[15] Fornes, A., J. Llados, and G. Sanchez. 2005. Staff and graphical primitive segmentation in old handwritten scores. In *Artificial Intelligence Research and Development*, ed. B. López, J. Meléndez, P. Radeva, and J. Vitrià, 83–90. Amsterdam: IOS Press.

[16] Digital Image Archive of Medieval Music. http://www.diamm.ac.uk/ (accessed 22 May 2009).

[17] Artières, T., N. Gauthier, P. Gallinari, and B. Dorizzi. 2002. A hidden Markov models combination framework for handwriting recognition. *International Journal on Document Analysis and Recognition* 5 (4): 233–43.

[18] Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 282–9.

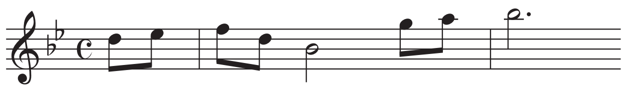
GLOBAL FEATURE VERSUS EVENT MODELS FOR FOLK SONG CLASSIFICATION

Ruben Hillewaere and **Bernard Manderick**
Computational Modeling Lab
Department of Computing
Vrije Universiteit Brussel
Brussels, Belgium
{rhillewa,bmanderi}@vub.ac.be

Darrell Conklin
Music Informatics Research Group
Department of Computing
City University London
United Kingdom
conklin@city.ac.uk

ABSTRACT

Music classification has been widely investigated in the past few years using a variety of machine learning approaches. In this study, a corpus of 3367 folk songs, divided into six geographic regions, has been created and is used to evaluate two popular yet contrasting methods for symbolic melody classification. For the task of folk song classification, a global feature approach, which summarizes a melody as a feature vector, is outperformed by an event model of abstract event features. The best accuracy obtained on the folk song corpus was achieved with an ensemble of event models. These results indicate that the event model should be the default model of choice for folk song classification.



pitch	74	75	77	74	70	79	81	82
melodic interval	$\perp$	+1	+2	-3	-4	+9	+2	+1
melodic contour	$\perp$	u	u	d	d	u	u	u
duration ratio	$\perp$	1	1	1	4	1/4	1	6
average pitch	76.5							
rel. freq. M3	0.143							
Huron contour	ascending							

Figure 1. Excerpt of the English folk tune “Harding’s Folly’s Hornpipe”, illustrating the contrast between global features (lower three) and event features (upper four).

1. INTRODUCTION

Computational folk music analysis is gaining increasing interest in recent years, revitalized by the developing field of computational ethnomusicology and also by interest in non-Western musics, the availability of advanced music data mining methods capable of dealing with very large data sets, and the existence of expanding folk song corpora on the internet. A mechanical process that can accurately locate new folk songs into geographical regions, proposed as early as the 1950s [1], can now be developed. In this work we describe and investigate two very different machine learning methods for folk song classification. Folk songs from six different European regions will be used, and the classification task is to assign unseen songs to their correct regions.

The more precise objective of this study is to determine whether *global feature* models are outperformed by methods based on *event features* for the task of folk song classification. A global feature encapsulates information about a whole piece into a single value: numeric, nominal, or Boolean. Using global features, pieces can be simply re-expressed as feature vectors and a wide range of standard machine learning algorithms can then be applied [2, 3]. Event features, on the other hand, do not summarize a piece into a single value, but rather view a piece as a sequence of events, each event with its own features. A standard technique for working with sequential symbolic music data expressed as event features is the n-gram model, which is particularly well-known for language modeling, a word in language being roughly analogous to an event in music.

Figure 1 illustrates a short melodic fragment, the first measures of an English tune called “Harding’s Folly’s Hornpipe”, expressed using both event features “pitch”, “melodic interval”, “melodic contour”, “duration ratio”, and a few global features “average pitch”, “rel. freq. M3” (relative frequency of major thirds), and “Huron contour”.

Despite many different proposals of global feature sets, and studies comparing a few feature sets to one another [4], there has not yet been a rigorous and systematic study comparing the relative efficacy of global features versus n-gram models for music classification. In this paper, we study four different global feature sets for the task of folk song classification. All feature sets are used for well-known classification techniques such as naive Bayes, logistic regression, SVM, k-nearest neighbours and decision trees. These results will be compared with both a simple n-gram

Origin	# pieces	avg notes/piece
England	990 (29.4%)	93
France	393 (11.7%)	71
Ireland	798 (23.7%)	105
Scotland	445 (13.2%)	119
S.E. Europe	123 (3.7%)	118
Scandinavia	618 (18.3%)	94
Total	3367	

Table 1. The *Europa-6* collection: the number of pieces and the average number of notes per piece in each region.

event model of linked interval/duration and its extension, the multiple viewpoint model [5].

Based on the fact that event models take into account sequential structure, the hypothesis of this study is that event models will outperform global feature models on the task of folk song classification. The remainder of this paper describes the methods and results employed in the exploration of this hypothesis.

2. METHODS

In this section we describe the global feature approach and the event models, and we detail the monophonic data set used for training.

2.1 Experimental data set

To explore the performance of these models, we compare their relative efficiency in terms of classification accuracies on a very large corpus of folk songs, which we call the *Europa-6* collection. This is a collection of folk songs from 6 countries/regions of Europe: see Table 1 for the classes and the piece counts in each class. The classification task is to assign unseen folk songs to their correct region of origin. Initially, 3724 pieces were selected by Li et al. [6] out of a collection of 14,000 folk songs transcribed in the ABC format. The collection was pruned to 3367 pieces by filtering out duplicate files. This was done by clustering all pieces into groups containing identical Jesser feature vectors (Section 2.2). If a group contained pieces spanning different regions (e.g., England and Ireland), all pieces in the group were discarded due to this ambiguity in annotation, otherwise just one piece of the group was retained. Furthermore, we retained only the highest note of double stops present in some instrumental folk songs. To focus on core melodies rather than performance elaboration, we removed all grace notes, trills, staccato, and ignored repeated section indications. Time and key signatures were retained. Since most of these pieces have no tempo indication, all tempo indications that were present were removed. Finally, by means of abc2midi we generated a clean quantized MIDI corpus, and removed all dynamic (velocity) indications generated by the style interpretation mechanism of abc2midi.

2.2 Global feature models

There have been many proposals of global feature sets. Volk et al. [4] provide an evaluation of several global feature sets for the task of comparing folk songs for melodic similarity, and several more sets can be found in the literature. In our experiments, we chose four:

- The first is the *Alicante* set of 28 global features, proposed by Ponce de Léon and Iñesta, applied to classification of 110 MIDI tunes in jazz/classical/pop genres [2]. From this set, we re-implemented a compact subset: the top 12 selected by [2]: Table 1.
- The second is the *Fantastic* set: 92 features computed by the program called Feature ANalysis Technology Accessing STatistics (In a Corpus), currently developed by Müllensiefen [7] (v0.9, downloaded from [8]). For this study, we only include the global features based on a single melody, which reduces the set to 37 features. In addition to some basic descriptive statistics based on pitch and duration, this set also includes a few entropy-based features, and some contour features derived from the work of Steinbeck [9] and Huron [10]. Moreover, the set of 37 features include some statistics of so-called *m-types* that take into account some local sequential note order. By default, Fantastic segments the scores and computes the features on the created phrases, but we instead report results without segmentation since these achieved globally better results.
- The *Jesser* set contains 40 pitch and duration statistics [11]. The pitch-based features are simple relative interval counts, like “amajsecond” (ascending major second). Similar features are present for all ascending and descending intervals in the range of the octave. Almost all features were implemented, only the feature “numlines” was not applicable for folk song classification based on melody.
- The last set is the *McKay* set of 101 global features, developed for the classification of orchestrated (instrumentation and dynamics) MIDI files [3]. Importantly, these features were used in the winning 2005 MIREX symbolic genre classification experiment which used orchestrated files for evaluation. The features were computed with McKay’s software package jSymbolic (version 12.2.0) from [12]. A few attributes “harmonicity of two strongest rhythmic pulses” and “strength ratio of two strongest rhythmic pulses” were removed due to runtime and numerical errors caused by their computation using the jSymbolic tool. For the analysis of the *Europa-6* corpus, many features such as those based on instrumentation, dynamics, polyphonic texture, glissando had the same value for every piece and were removed. The final McKay set contains a total of 62 features.

The four global feature sets above are summarized in Table 2. In this table, we also indicate for each feature

Global feature set	# features	pitch	duration
Alicante	12	7	2
Fantastic	37	21	15
Jesser	39	31	6
McKay	62	35	5

Table 2. Global feature sets used in our experiments. The last two columns show the number of features that are derived from pitch and duration.

set how many features are derived from pitch or duration. A feature is derived from pitch (duration) when at least one pitch (duration) value is inspected for the feature computation. Most features are derived from pitch or duration, spanning from very basic features like "variability of note duration" to more abstract ones, such as "tonalness" or "interval distribution normality". Examples of features that are not derived from pitch or duration are descriptors such as "has meter changes" and "average time between attacks", or more specific ones like "polyrhythms" or "strength of strongest rhythmic pulse". In the Fantastic set some features are based on both pitch and duration, like the step contour and interpolation contour features. These four global feature sets were also chosen as they do not show much overlap in semantic content, aside from some very basic features such as "number of notes" and "pitch range".

A global feature set summarizes a piece as a feature vector, which can be viewed as a data point in a feature space. The classification task can thus easily be addressed with standard machine learning techniques, for which toolboxes are available. The underlying idea is to assess the discriminative power of a global feature set by looking at its performance in terms of classification accuracies.

2.3 Event models

In contrast to global feature models, event models take into account the sequential structure of the melody. A type of event model commonly used for statistical language modeling is the n-gram model [13]. In an n-gram model for music, the probability of a piece $\overline{e_\ell} = [e_1, \ldots, e_\ell]$ is obtained by computing the joint probability of the individual events in the piece:

$$p(\overline{e_\ell}) = \prod_{i=1}^{\ell} p(e_i \,|\, \overline{e_{i-1}}), \tag{1}$$

with suitable restrictions on the context $\overline{e_{i-1}}$, for example, for a trigram model $\overline{e_{i-1}}$ is restricted to $[e_{i-2}, e_{i-1}]$. The conditional event probabilities $p(e_i \,|\, \overline{e_{i-1}})$ are estimated by the n-gram counts of the training data. In addition, Method C smoothing [13] is used to handle the zero frequency problem. To use n-gram models for melody classification, for each class a separate model is built and the predicted class of a piece is the class whose model generates the piece with the highest probability.

Presented with sparse data, n-gram models for music cannot model the pitch or duration directly, hence the music events must first be clustered into more abstract equivalence classes by applying functions called *viewpoints* [14]. Examples of viewpoints are "melodic interval" or "duration contour", which obviously lead to event features that are less sparse than the concrete music events in the corpus. Particularly useful are *linked* viewpoints, which capture correlation between abstract classes, for example, a linked viewpoint of melodic interval and duration, meaning we represent every event as a pair of its melodic interval and duration.

For a viewpoint τ, the event sequence $\overline{e_\ell}$ is transformed to the abstract feature sequence $\widehat{\tau}(\overline{e_\ell}) = [\tau(e_1), \ldots, \tau(e_\ell)]$, and Equation 1 can be adapted as follows, resulting in the *viewpoint model*:

$$p(\overline{e_\ell}) = \prod_{i=1}^{\ell} p(\tau(e_i) | \widehat{\tau}(\overline{e_{i-1}})) \times p(e_i | \tau(e_i)). \tag{2}$$

The first factor in (2) is the probability of the abstract feature using an n-gram model, and the second factor is the probability of the concrete event given the abstract feature, which is modeled by a uniform distribution.

An extension is the *multiple viewpoint model*, an ensemble of viewpoints used in aggregation to compute the probability of a sequence. Multiple viewpoints have been used with success in symbolic music processing tasks such as melody prediction and generation [5] and melody segmentation [15]. To combine the predictions of k viewpoints $\tau_1, \ldots, \tau_k$, one straightforward way is to use the geometric mean of the component viewpoint predictions:

$$p(\overline{e_\ell}) = \frac{1}{Z} \times \left(\prod_{i=1}^{k} p_{\tau_i}(\overline{e_\ell}) \right)^{1/k} \tag{3}$$

where Z is a normalization factor. In the Results section, a multiple viewpoint model will be contrasted with both an n-gram model and a global feature model.

3. RESULTS

In this section we report on the experimental results on the *Europa-6* collection. For the evaluation of the global feature sets, we used a standard machine learning toolbox called Weka (version 3.6.1) [16] which is documented in [17]. Weka contains many different algorithms for classification: we used the well-known classifiers Naive Bayes, k-nearest neighbours, decision trees (J48), Support Vector Machines (SVM) and Logistic regression. The feature sets were compared by computing classification accuracies for each of these classifiers, which were all obtained by 10-fold cross validation. Default Weka configuration parameters were used for all classifiers. Results are reported in Table 3. To measure the difficulty of this classification task, note that always predicting the most frequent class (England) will achieve an accuracy of only 29.4%. For the method of k-nearest neighbours we explored many values of k, and $k = 20$ seemed to yield the best results in general. We observe that the McKay and the Jesser features

Feature set	Alicante	Fantastic	Jesser	McKay
# features	12	37	39	62
Naive Bayes	45.3	52.5	47.1	**53.8**
Decision tree	48.2	47.3	**58.8**	58.4
SVM	51.0	57.6	63.4	**66.7**
kNN (k=20)	52.7	51.3	**61.9**	60.7
Logistic regr.	51.9	46.5	63.8	<u>**67.8**</u>

Table 3. Classification accuracies of the global feature sets on the *Europa-6* collection, obtained by 10-fold cross validation.

Feature selection method	Joined set	CfsSubsetEval (BestFirst)	ClassifSubsetEval (Naive Bayes)	PCA (top 13)
# features	150	41	20	13
Naive Bayes	55.7	60.0	**63.9**	61.6
Decision tree	59.1	**62.7**	59.1	53.5
SVM	<u>**69.7**</u>	68.3	61.6	64.3
kNN (k=20)	**65.9**	66.3	61.9	64.3
Logistic regr.	N/A	**69.5**	49.4	63.8

Table 4. Classification accuracies of the sets created by various feature selection methods, obtained by 10-fold cross validation.

obtain the highest accuracies, and the best overall result is obtained with logistic regression on the McKay features, with an accuracy of 67.8%.

Before comparing these results to those obtained with the event models, we explored whether there might be a compact optimal global feature set for the task of classification of folk tunes. Therefore, we have created a fifth global feature set containing all global features from the Alicante, Fantastic, Jesser and McKay sets. On this joined set, we performed various feature selection methods, either by evaluating attribute subsets on their predictive value, like correlation-based feature selection or classifier subset evaluators, or by using single-attribute evaluators, by measuring the information gain of each attribute or by applying principal component analysis. Several search strategies were explored for the subset evaluators, and for the single-attribute evaluators we searched for the optimal number of top features to include. The best results obtained are detailed in Table 4. The maximum result of 69.7% is obtained on the full joined set with a Support Vector Machine. This confirms the known strength of an SVM classifier when dealing with high dimensional feature vectors. The overall best performing compact subset was found with a correlation-based feature set evaluator and a greedy hill-climbing search (BestFirst) as described in [18], obtaining 69.5% with a multi-class classifier using logistic regression. It was not possible to compute the accuracy with that classifier on the full joined set, as the computation was too heavy.

To evaluate the event models on the *Europa-6*, we implemented a 10-fold cross validation scheme. With a pentagram model of a linked viewpoint of melodic interval and duration, the obtained classification accuracy is **72.7%**, significantly higher than even the best of the global feature models. For the multiple viewpoint model we used an ensemble of four viewpoints, the same collection as used by [5]:

- a linked viewpoint of melodic interval, and pitch class interval from the reference pitch class of the piece. The latter is calculated assuming the major mode;
- a linked viewpoint of melodic interval and inter-onset interval (time difference between two successive onsets, which will be different than an event's duration in the presence of rests);
- a simple viewpoint that returns the pitch of an event;
- a linked viewpoint of pitch class interval from the reference pitch and first metric level. The latter is a Boolean viewpoint which is true if an event is at the beginning of a bar.

In the multiple viewpoint mode, each of the above has its own pentagram model, and the component viewpoints are combined as described in Equation (3). As expected, this multiple viewpoint model achieves a significantly higher accuracy than the linked viewpoint of <u>**76.0%**</u>. This is the best result we have yet obtained on the *Europa-6* corpus.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a first systematic study for folk song classification, comparing two well-known methods to approach this task, namely global feature models and event models. The hypothesis of the event model was validated, since the results show that four established global feature sets using standard classifiers were outperformed by a very simple n-gram model of a linked viewpoint, and even more by the multiple viewpoint model. We have explored methods to find an optimal global feature set by joining the four sets and by performing attribute selection, and we observe a slight improvement, but the event models still achieve higher classification accuracies. We believe the event models perform better, precisely because they retain sequential information that global features do not take into account. In order to identify folk song regions, one needs to capture the inner structure of a musical phrase. The event model should thus be the default model of choice for folk song classification.

There are still some other types of models that we would like to consider in our future work, such as methods where one focuses on the development of a good similarity or distance measure between pieces [19]. Another type of model that has not been thoroughly explored for classification is a pattern model, where one uses a collection of musical patterns and classifies a piece according to the patterns it contains [20]. Another question we want to address is if these results still hold for polyphonic music. Global feature models can easily be expanded to polyphony. Event models, however, are harder to extrapolate to polyphony, because we then have to deal with the problem of finding a suitable harmonic representation and segmentation.

5. ACKNOWLEDGEMENTS

This research is partially funded by Messiaen Weerspiegeld. Special thanks are due to Xiao Li for correspondences on her original ABC corpus, to Daniel Müllensiefen for clarifications of the Fantastic feature set, to Mathieu Bergeron for assistance with the abc2midi code, and to Kerstin Neubarth, Stijn Meganck and Hélène Viratelle for helpful comments on a draft of this article.

6. REFERENCES

[1] A. Lomax: "Folk Song Style: Notes on a Systematic Approach to the Study of Folk Song," *Journal of the International Folk Music Council*, Vol. 8, pp. 48-50, 1956.

[2] P. J. Ponce de Léon and J. M. Iñesta: "Statistical description models for melody analysis and characterization," *Proceedings of the 2004 International Computer Music Conference*, pp. 149–156, 2004.

[3] C. McKay and I. Fujinaga: "Automatic genre classification using large high-level musical feature sets," *Proceedings of the International Conference on Music Information Retrieval*, pp. 525–530, 2004.

[4] A. Volk, P. van Kranenburg, J. Garbers, F. Wiering, R.C. Veltkamp and L.P. Grijp: "A manual annotation method for melodic similarity and the study of melody feature sets," *Proceedings of the 9th International Conference on Music Information Retrieval*, pp. 101–106, 2008.

[5] D. Conklin and I. H. Witten: "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, Vol. 24, No. 1, pp. 51–73, 1995.

[6] X. Li, G. Li and J. Bilmes: "A factored language model of quantized pitch and duration," *International Computer Music Conference*, 2006.

[7] D. Müllensiefen: *FANTASTIC: Feature ANalysis Technology Accessing STatistics (In a Corpus): Technical Report v0.9*, 2009.

[8] `http://doc.gold.ac.uk/isms/mmm`

[9] W. Steinbeck: *Struktur und Ähnlichkeit: Methoden automatisierter Melodieanalyse*, Bärenreiter, Kassel, 1982.

[10] D. Huron: "The melodic arch in western folksongs," *Computing in Musicology, 10*, pp. 3–23, 1996.

[11] B. Jesser: *Interaktive Melodieanalyse*, Peter Lang, Bern, 1991.

[12] `http://jmir.sourceforge.net/jSymbolic.html`

[13] D. Jurafsky and J. Martin: *Speech and Language Processing*. Prentice-Hall, Englewood Cliffs, NJ, 2000.

[14] D. Conklin: "Melodic analysis with segment classes," *Machine Learning*, Vol. 65, pp. 349–360, 2006.

[15] M. T. Pearce, D. Müllensiefen and G. A. Wiggins: "An information-dynamic model of melodic segmentation," presented at the International Workshop on Machine Learning and Music. University of Helsinki, Finland. July, 2008.

[16] `http://www.cs.waikato.ac.nz/ml/weka/`

[17] I. H. Witten and E. Frank: *Data Mining: practical machine learning tools and techniques*, 2nd edition, Morgan Kaufmann, 2005.

[18] M. A. Hall: *Correlation-based Feature Subset Selection for Machine Learning*, PhD Thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1998.

[19] M. Li and R. Sleep: "Melody classification using a similarity metric based on Kolmogorov complexity," *Sound and Music Computing Conference*, Paris, 2004.

[20] C.R. Lin, N.H. Liu, Y.H. Wu and A.L.P. Chen: "Music classification using significant repeating patterns," *Lecture notes in Computer Science*, volume 2973, Springer, pp. 506–518, 2004.

ROBUST SEGMENTATION AND ANNOTATION OF FOLK SONG RECORDINGS

Meinard Müller
Saarland University and
MPI Informatik
Saarbrücken, Germany
meinard@mpi-inf.mpg.de

Peter Grosche
Saarland University and
MPI Informatik
Saarbrücken, Germany
pgrosche@mpi-inf.mpg.de

Frans Wiering
Department of Information and
Computing Sciences, Utrecht University
Utrecht, Netherlands
frans.wiering@cs.uu.nl

ABSTRACT

Even though folk songs have been passed down mainly by oral tradition, most musicologists study the relation between folk songs on the basis of score-based transcriptions. Due to the complexity of audio recordings, once having the transcriptions, the original recorded tunes are often no longer studied in the actual folk song research though they still may contain valuable information. In this paper, we introduce an automated approach for segmenting folk song recordings into its constituent stanzas, which can then be made accessible to folk song researchers by means of suitable visualization, searching, and navigation interfaces. Performed by elderly non-professional singers, the main challenge with the recordings is that most singers have serious problems with the intonation, fluctuating with their voices even over several semitones throughout a song. Using a combination of robust audio features along with various cleaning and audio matching strategies, our approach yields accurate segmentations even in the presence of strong deviations.

1. INTRODUCTION

Generally, a folk song is referred to as a song that is sung by the common people of a region or culture during work or social activities. Since many decades, significant efforts have been carried out to assemble and study large collections of folk songs [7, 12]. Even though folk songs were typically transmitted only by oral tradition without any fixed symbolic notation, most of the folk song research is conducted on the basis of notated music material, which is obtained by transcribing recorded tunes into symbolic, score-based music representations. After the transcription, the audio recordings are often no longer studied in the actual research. Since folk songs are part of oral culture, one may conjecture that performance aspects enclosed in the recorded audio material are likely to bear valuable information, which is no longer contained in the transcriptions. Furthermore, even though the notated music material may be more suitable for classifying and identifying folk songs using automated methods, the user may want to listen to the original recordings rather than to synthesized versions of the transcribed tunes.

It is the object of this paper to indicate how the original recordings can be made more easily accessible for folk song researches and listeners by bridging the gap between the symbolic and the audio domain. In particular, we present a procedure for automatically segmenting a given folk song recording that consists of several repetitions of the same tune into its individual stanzas. Using folk song recordings of the *Onder de groene linde* (OGL), main challenges arise from the fact that the songs are performed by elderly non-professional singers under poor recording conditions. The singers often deviate significantly from the expected pitches and have serious problems with the intonation. Even worse, their voices often fluctuate by several semitones downwards or upwards across the various stanzas of the same recording. As our main contribution, we introduce a combination of robust audio features along with various cleaning and audio matching strategies to account for such deviations and inaccuracies in the audio recordings. Our evaluation on folk song recordings shows that we obtain reliable segmentations even in the presence of strong deviations.

The remainder of this paper is organized as follows. In Sect. 2, we describe the relationship of these investigations to folk song research and describe the folk song collection we employ. In Sect. 3, we show how the recorded songs can be segmented and annotated by locally comparing and aligning the recordings' feature representations with available transcriptions of the tunes. In particular, we introduce various methods for achieving robustness to the aforementioned pitch fluctuations and recording artifacts. Then, in Sect. 4, we report on our systematic experiments conducted on a representative selection of folk song recordings. Finally, in Sect. 5, we indicate how our segmentation results can be used as basis for novel user interfaces, sketch possible applications towards automated performance analysis, and give prospects on future work. Further related work is discussed in the respective sections.

2. FOLK SONG RESEARCH

Folk song reseach has been carried out from many different perspectives. An important problem is to reconstruct and understand the genetic relation between variants of folk songs [12]. Furthermore, by systematically studying entire collections of folk songs, researchers try to discover musical connections and distinctions between different national or regional cultures [7]. To support such research, several databases of encoded folk song melodies have been assembled, the best known of which is the Essen folk song database, [1] which currently contains roughly 20000 folk songs from a variety of sources and cultures. This collection has also been widely used in MIR research.

Even though folk songs have been passed down mainly by oral tradition, most of the folk song research is conducted on the basis of notated music material. However, various folk song collections contain a considerable amount of audio data, which has not yet been explored at a larger scale. One of these collections is *Onder de groene linde* (OGL), which is part of the *Nederlandse Liederenbank* (NLB). The OGL collection comprises several 7277 Dutch folk song recordings along with song transcriptions as well as a rich set of metadata. [2] This metadata includes date and location of recording, information about the singer, and classification by (textual) topic. OGL contains 7277 recordings, which have been digitized as MP3 files. Nearly all of recordings are monophonic, and the vast majority is sung by elderly solo female singers. When the collection was assembled, melodies were transcribed on paper by experts. Usually only one strophe is given in music notation, but variants from other strophes are regularly included. The transcriptions are somewhat idealized: they tend to represent the presumed intention of the singer rather than the actual performance. For about 2500 melodies, transcribed stanzas are available in various symbolic formats including LilyPond, [3] from which MIDI representations have been generated (with a tempo set at 120 BPM for the quarter note).

An important step in unlocking such collections of orally transmitted folk songs is the creation of content-based search engines. The creation of such a search engine is an important goal of the WITCHCRAFT project [8]. The engines should enable a user to search for encoded data using advanced melodic similarity methods. Furthermore, it should also be possible to not only visually present the retrieved items, but also to supply the corresponding audio recordings for acoustic playback. One way of solving this problem is to create robust alignments between retrieved encodings (for example in MIDI format) and the audio recordings. The segmentation and annotation procedure described in the following section exactly accomplishes this task.

[1] `http://www.esac-data.org/`

[2] The OGL collection is currently hosted at the Meertens Institute in Amsterdam. The metadata of the songs are available through `www.liederenbank.nl`

[3] `www.lilypond.org`

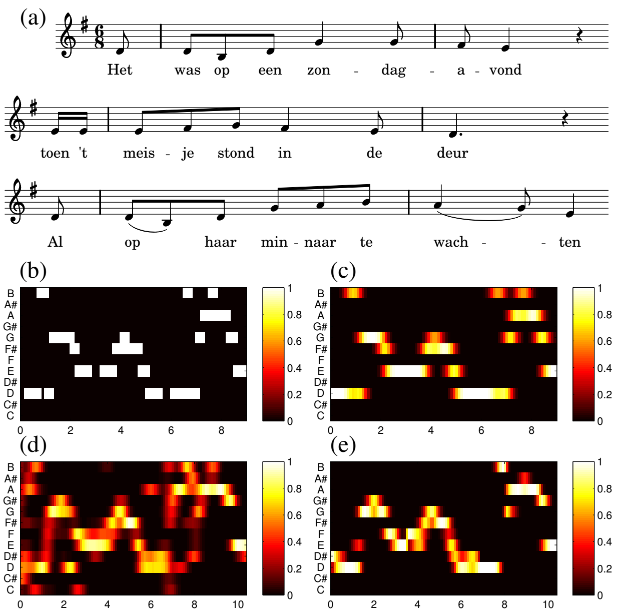


Figure 1. Representations of the beginning of the first stanza of NLB73626 **(a)** Score representation. **(b)** Chromagram of MIDI representation. **(c)** Smoothed MIDI chromagram (CENS). **(d)** Chromagram of audio recording (CENS). **(e)** F0-enhanced chromagram (see Sect. 3.4).

3. FOLK SONG SEGMENTATION

In this section, we present a procedure for automatically segmenting a folk song recording that consists of several repetitions of the same tune into its individual stanzas. Here, we assume that we are given a transcription of a reference tune in form of a MIDI file. Recall from Sect. 2 that this is exactly the situation we have with the songs of the OGL collection. In the first step, we transform the MIDI reference as well as the audio recording into a common mid-level representation. Here, we use the well-known chroma representation, which is summarized in Sect. 3.1. On the basis of this feature representation, the idea is to locally compare the reference with the audio recording by means of a suitable distance function (Sect. 3.2). Using a simple iterative greedy strategy, we derive the segmentation from local minima of the distance function (Sect. 3.3). This approach works well as long as the singer roughly follows the reference tune and stays in tune. However, this is an unrealistic assumption. In particular, most singers have significant problems with the intonation. Their voices often fluctuate by several semitones downwards or upwards across the various stanzas of the same recording. In Sect. 3.4, we show how the segmentation procedure can be improved to account for poor recording conditions, intonation problems, and pitch fluctuations.

3.1 Chroma Features

In order to compare the MIDI reference with the audio recordings, we revert to chroma-based music features, which have turned out to be a powerful mid-level representation for relating harmony-based music, see [1, 6, 9, 11].

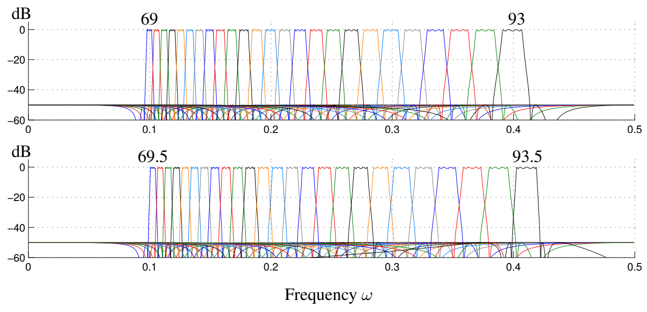


Figure 2. Magnitude responses in dB for some of the pitch filters of the multirate pitch filter bank used for the chroma computation. **Top:** Filters corresponding to MIDI pitches $p \in [69:93]$ (with respect to the sampling rate 4410 Hz). **Bottom:** Filters shifted half a semitone upwards.

Here, the chroma refer to the 12 traditional pitch classes of the equal-tempered scale encoded by the attributes $\mathrm{C}, \mathrm{C}^\sharp, \mathrm{D}, \ldots, \mathrm{B}$. Representing the short-time energy content of the signal in each of the 12 pitch classes, chroma features do not only account for the close octave relationship in both melody and harmony as it is prominent in Western music, but also introduce a high degree of robustness to variations in timbre and articulation [1]. Furthermore, normalizing the features makes them invariant to dynamic variations.

It is straightforward to transform a MIDI representation into a chroma representation or chromagram. Using the explicit MIDI pitch and timing information one basically identifies pitches that belong to the same chroma class within a sliding window of a fixed size, see [6]. Fig. 1 shows a score and the resulting MIDI reference chromagram. For transforming an audio recording into a chromagram, one has to revert to signal processing techniques. Most chroma implementations are based on short-time Fourier transforms in combination with binning strategies [1]. In this paper, we revert to chroma features obtained from a pitch decomposition using a multirate pitch filter bank as described in [9]. The employed pitch filters possess a relatively wide passband, while still properly separating adjacent notes thanks to sharp cutoffs in the transition bands, see Fig. 2. Actually, the pitch filters are robust to deviations of up to ± 25 cents[4] from the respective note's center frequency. The pitch filters will play an important role in Sect. 3.4. Finally, in our implementation, we use a quantized and smoothed version of chroma features referred to as CENS features [9] with a feature resolution of 10 Hz (10 features per second), see (c) and (d) of Fig. 1. For technical details, we refer to the cited literature.

3.2 Distance Function

We now introduce a distance function that expresses the distance of the MIDI reference chromagram with suitable subsegments of the audio chromagram. More precisely, let $X = (X(1), X(2), \ldots, X(K))$ be the sequence of chroma features obtained from the MIDI reference and

[4] The *cent* is a logarithmic unit to measure musical intervals. The semitone interval of the equally-tempered scale equals 100 cents.

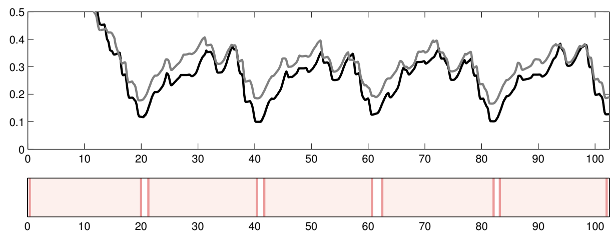


Figure 3. **Top:** Distance function Δ for NLB73626 using original chroma features (gray) and F0-enhanced chroma features (black). **Bottom:** Resulting segmentation.

let $Y = (Y(1), Y(2), \ldots, Y(L))$ be the one obtained from the audio recording. In our case, the features $X(k)$, $k \in [1:K]$, and $Y(\ell)$, $\ell \in [1:L]$, are normalized 12-dimensional vectors. We define the distance function $\Delta := \Delta_{X,Y} : [1:L] \to \mathbb{R} \cup \{\infty\}$ with respect to X and Y using a variant of dynamic time warping (DTW):

$$\Delta(\ell) := \frac{1}{K} \min_{a \in [1:\ell]} \Big(\mathrm{DTW}\big(X\,,\,Y(a:\ell)\big)\Big), \qquad (1)$$

where $Y(a:\ell)$ denotes the subsequence of Y starting at index a and ending at index $\ell \in [1:L]$. Furthermore, $\mathrm{DTW}(X, Y(a:\ell))$ denotes the DTW distance between X and $Y(a:\ell)$ with respect to a suitable local cost measure (in our case, the cosine distance). The distance function Δ can be computed efficiently using dynamic programming. For details on DTW and the distance function, we refer to [9]. The interpretation of Δ is as follows: a small value $\Delta(\ell)$ for some $\ell \in [1:L]$ indicates that the subsequence of Y starting at index a_ℓ (with $a_\ell \in [1:\ell]$ denoting the minimizing index in (1)) and ending at index ℓ is similar to X. Here, the index a_ℓ can be recovered by a simple backtracking algorithm within the DTW computation procedure. The distance function Δ for NLB73626 is shown in Fig. 3 as gray curve. The five pronounced minima of Δ indicate the endings of the five stanzas of the audio recording.

3.3 Audio Segmentation

Recall that we assume that a folk song audio recording basically consists of a number of repeating stanzas. Exploiting the existence of a MIDI reference and assuming the repetitive structure of the recording, we apply the following simple greedy segmentation strategy. Using the distance function Δ, we look for the index $\ell \in [1:L]$ minimizing Δ and compute the starting index a_ℓ. Then, the interval $S_1 := [a_\ell : \ell]$ constitutes the first *segment*. The value $\Delta(\ell)$ is referred to as the *cost* of the segment. To avoid large overlaps between the various segments to be computed, we exclude a neighborhood $[L_\ell : R_\ell] \subset [1:L]$ around the index ℓ from further consideration. In our strategy, we set $L_\ell := \max(1, \ell - \frac{2}{3}K)$ and $R_\ell := \min(L, \ell + \frac{2}{3}K)$, thus excluding a range of two thirds of the reference length to the left as well as to the right of ℓ. To achieve the exclusion, we modify Δ simply by setting $\Delta(m) := \infty$ for $m \in [L_\ell : R_\ell]$. To determine the next segment S_2, the same procedure is repeated using the modified distance function, and so on. This results in a sequence of segments $S_1, S_2, S_3, \ldots$. The procedure is repeated until all values of the modified Δ lie above a suitably chosen *quality threshold* $\tau > 0$. Let N denote the number of resulting segments, then $S_1, S_2, \ldots, S_N$ constitutes the final segmentation result, see Fig. 3 for an illustration.

3.4 Enhancement Strategies

Recall that the comparison of the MIDI reference and the audio recording is performed on the basis of chroma representations. Therefore, the segmentation algorithm described so far only works well in the case that the MIDI reference and the audio recording are in the same musical key. Furthermore, the singer has to stick roughly to the pitches of the well-tempered scale. Both assumptions are violated for most of the songs. Even worse, the singers often fluctuate with their voice by several semitones within a single recording. This often leads to poor local minima or even completely useless distance functions as illustrated Fig. 4. To deal with local and global pitch deviations as well as with poor recording conditions, we use a combination of various enhancement strategies.

In our first strategy, we enhance the quality of the chroma features similar to [4] by picking only dominant spectral coefficients, which results in a significant attenuation of noise components. Dealing with monophonic music, we can go even one step further by only picking spectral components that correspond to the fundamental frequency (F0). More precisely, we use a modified autocorrelation method as suggested in [3] to the estimate the fundamental frequency for each audio frame. For each frame, we then determine the MIDI pitch having a center frequency that is closest to the estimated fundamental frequency. Next, in the pitch decomposition used for the chroma computation, we assign energy only to the pitch subband that corresponds to the determined MIDI pitch—all other pitch subbands are set to zero within this frame. Finally, the resulting sparse pitch representation is projected onto a chroma representation and smoothed as before, see Sect. 3.1. The cleaning effect on the resulting chromagram, which is also referred to as *F0-enhanced chromagram*, is illustrated by (d) and (e) of Fig. 1.

Even though the folk song recordings are monophonic, the F0 estimation is often not accurate enough in view of applications such as automated transcription. However, using chroma representations, octave errors as typical in F0 estimations become irrelevant. Furthermore, the F0-based pitch assignment is capable of suppressing most of the noise resulting from poor recording conditions. Finally, local pitch deviations caused by the singers' intonation problems as well as vibrato are compensated to a substantial degree. As a result, the desired local minima of the distance function Δ, which are crucial in our segmentation procedure, become more pronounced. This effect is also illustrated by Fig. 3.

Next, we show how to deal with global pitch deviations and continuous fluctuation across several semitones. To account for a global difference in key between the MIDI reference and the audio recording, we revert to the observation by Goto [5] that the twelve cyclic shifts of a 12-dimensional chroma vector naturally correspond to the twelve possible transpositions. Therefore, it suffices to determine the shift index that minimizes the chroma distance of the audio recording and MIDI reference and then to cyclically shift the audio chromagram according to this index. Note that instead of shifting the audio chromagram, one can also shift the MIDI chromagram in the inverse direction. The minimizing shift index can be determined either by using averaged chroma vectors as suggested in [11] or by computing twelve different distance functions for the twelve shifts, which are then minimized to obtain a single transposition invariant distance functions. We detail on the latter strategy, since it also solves part of the problem having a fluctuating voice within the audio recording. A similar strategy was used in [10] to achieve transposition invariance for music structure analysis tasks.

We simulate the various pitch shifts by considering all twelve possible cyclic shifts of the MIDI reference chromagram. We then compute a separate distance function for each of the shifted reference chromagrams and the original audio chromagram. Finally, we minimize the twelve resulting distance functions, say $\Delta^0, \ldots, \Delta^{11}$, to obtain a single *transposition invariant* distance function $\Delta^{\mathrm{trans}} : [1:L] \to \mathbb{R} \cup \{\infty\}$:

$$\Delta^{\mathrm{trans}}(\ell) := \min_{i \in [0:11]} \Big(\Delta^i(\ell) \Big). \qquad (2)$$

Fig. 4 shows the resulting function Δ^{trans} for a folk song recording with strong fluctuations. In contrast to the original distance function Δ, the function Δ^{trans} exhibits a number of significant local minima that correctly indicate

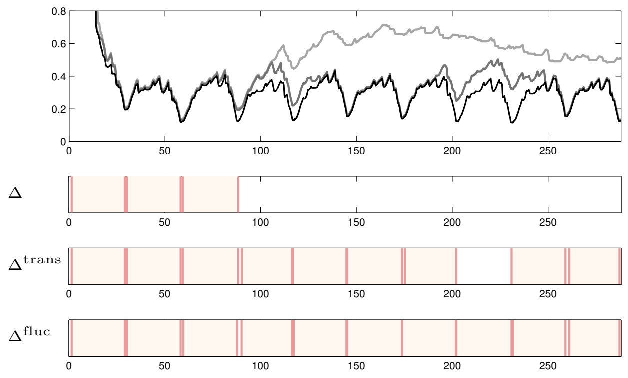


Figure 4. Distance functions Δ (light gray), Δ^{trans} (dark gray), and Δ^{fluc} (black) for the song NLB73286 as well as the resulting segmentations.

Stanza	1	2	3	4	5	6	7	8	9	10
12 shift	5	5	5	4	4	4	4	3	3	3
24 shift	5.0	5.0	4.5	4.5	4.0	4.0	3.5	3.5	3.0	3.0

Table 1. Shift indices (cyclically shifting the audio chromagrams upwards) used for transposing the various stanzas of the audio recording of NLB73286 to optimally match the MIDI reference, see also Fig. 4. The shift indices are given in semitones (obtained by Δ^{trans}) and in half semitones (obtained by Δ^{fluc}).

the segmentation boundaries of the stanzas.

So far, we have accounted for transpositions that refer to the pitch scale of the equal-tempered scale. However, the above mentioned voice fluctuation are fluent in frequency and do not stick to a strict pitch grid. Recall from Sect. 3.1 that our pitch filters can cope with fluctuations of up to ± 25 cents. To cope with pitch deviations between 25 and 50 cents, we employ a second filter bank, in the following referred to as *half-shifted filter bank*, where all pitch filters are shifted by half a semitone (50 cents) upwards, see Fig. 2. Using the half-shifted filter bank, one can compute a second chromagram, referred to as *half-shifted chromagram*. A similar strategy is suggested in [4, 11] where generalized chroma representations with 24 or 36 bins (instead of the usual 12 bins) are derived from a short-time Fourier transform. Now, using the original chromagram as well as the half-shifted chromagram in combination with the respective 12 cyclic shifts, one obtains 24 different distance functions in the same way as described above. Minimization over the 24 functions yields a single function Δ^{fluc} referred to as *fluctuation invariant distance function*. The improvements achieved by this novel distance function are illustrated by Fig. 4. Table 1 shows the optimal shift indices derived from the transposition and fluctuation invariant segmentation strategies, where the decreasing indices indicate to which extend the singer's voice rises across the various stanzas of the song.

4. EXPERIMENTS

Our evaluation is based on a dataset consisting of 47 representative folk song recordings selected from the OGL collection, see Sect. 2. The evaluation audio dataset has a total length of 156 minutes, where each of the recorded song consists of 4 to 34 stanzas amounting to a total number of 465 stanzas. The recordings reveal significant deteriorations concerning the audio quality as well as the singer's performance. Furthermore, in various recordings the tunes are overlayed with sounds such as ringing bells, singing birds, or barking dogs, and sometimes the songs are interrupted by remarks of the singers. We manually annotated all audio recordings by specifying the segment boundaries of the stanzas' occurrences in the recordings. Since for most cases the end of a stanza more or less coincides with the beginning of the next stanza and since the beginnings are more important in view of retrieval and navigation applications, we only consider the starting boundaries of the segments in our evaluation. In the following, these boundaries are referred to as *ground truth boundaries*.

To assess the quality of the final segmentation result, we use precision and recall values. To this end, we check to what extent the 465 manually annotated stanzas within the evaluation dataset have been identified correctly by the segmentation procedure. More precisely, we say that a computed starting boundary is a *true positive*, if it coincidences with a ground truth boundary up to a small tolerance given by a parameter δ measured in seconds. Otherwise, the computed boundary is referred to as a *false positive*. Furthermore, a ground truth boundary that is not in

Strategy	F0	P	R	F	α	β	γ
Δ	$-$	0.898	0.628	0.739	0.338	0.467	0.713
Δ	$+$	0.884	0.688	0.774	0.288	0.447	0.624
Δ^{trans}	$-$	0.866	0.817	0.841	0.294	0.430	0.677
Δ^{trans}	$+$	0.890	0.890	0.890	0.229	0.402	0.559
Δ^{fluc}	$-$	0.899	0.901	0.900	0.266	0.409	0.641
Δ^{fluc}	$+$	0.912	0.940	0.926	0.189	0.374	0.494

Table 2. Performance measures for various segmentation strategies using the tolerance parameter $\delta = 2$ and the quality threshold $\tau = 0.4$. The second column indicates whether original $(-)$ or F0-enhanced $(+)$ chromagrams are used.

δ	P	R	F
1	0.637	0.639	0.638
2	0.912	0.940	0.926
3	0.939	0.968	0.953
4	0.950	0.978	0.964
5	0.958	0.987	0.972

τ	P	R	F
0.1	0.987	0.168	0.287
0.2	0.967	0.628	0.761
0.3	0.950	0.860	0.903
0.4	0.912	0.940	0.926
0.5	0.894	0.944	0.918

Table 3. Dependency of the PR-based performance measures on the tolerance parameter δ and the quality threshold τ. All values refer to Δ^{fluc} using F0-enhanced chromagrams. **Left:** PR-based performance measures for various δ and fixed $\tau = 0.4$. **Right:** PR-based performance measures for various τ and fixed $\delta = 2$.

a δ-neighborhood of a computed boundary is referred to as a *false negative*. We then compute the precision P and the recall R boundary identification task. From these values one obtains the F-measure $\mathrm{F} := 2 \cdot \mathrm{P} \cdot \mathrm{R}/(\mathrm{P} + \mathrm{R})$.

Table 2 shows the PR-based performance measures of our segmentation procedure using different distance functions with original as well as F0-enhanced chromagrams. In this first experiment, the tolerance parameter is set to $\delta = 2$ and the quality threshold to $\tau = 0.4$. Here, a tolerance of up to $\delta = 2$ seconds seems to us an acceptable deviation in view of our intended applications. For example, the most basic distance function Δ with original chromagrams yields an F-measure of $\mathrm{F} = 0.739$. Using F0-enhanced chromagrams instead of the original ones results in $\mathrm{F} = 0.774$. The best result of $\mathrm{F} = 0.926$ is obtained when using Δ^{fluc} with F0-enhanced chromagrams. Note that all of our introduced enhancement strategies result in an improvement in the F-measure. In particular, the recall values improve significantly when using the transposition and fluctuation-invariant distance functions.

A manual inspection of the segmentation results showed that most of the false negatives as well as false positives are due to deviations in particular at the stanzas' beginnings. The entry into a new stanza seems to be a problem for some of the singers, who need some seconds before getting stable in intonation and pitch. A typical example is NLB72355. Increasing the tolerance parameter δ, the PR-based performance measures improve substantially, as indicated by Table 3 (left). For example, using $\delta = 3$ instead of $\delta = 2$, the F-measure increase from $F = 0.926$ to $F = 0.953$. Other sources of error are that the transcriptions sometimes differ significantly from what is actually sung, as is the case for NLB72395. Here, as was already mentioned in Sect. 2, the transcripts represent the presumed intention of the singer rather than the actual performance. Finally, structural differences between the various stanzas are a further reason for segmentation errors. The handling of such structural differences constitutes an interesting research problem, see Sect. 5. In a further experiment, we investigated the role of the quality threshold τ on the final segmentation results, see Table 3 (right). Not surprisingly, a small τ yields a high precision and a low recall. Increasing τ, the recall increases at the cost of a decrease in precision. The value $\tau = 0.4$ was chosen, since it constitutes a good trade-off between recall and precision.

Finally, to complement our PR-based evaluation, we introduce a second type of more softer performance measures that indicate the significance of the desired minima. To this end, we consider the distance functions for all songs with respect to a fixed strategy and chroma type. Let α be the average over the cost of all ground truth segments (given by the value of the distance function at the corresponding ending boundary). Furthermore, let β be the average over all values of all distance functions. Then the quotient $\gamma = \alpha/\beta$ is a weak indicator on how well the desired minima (the desired true positives) are separated from possible irrelevant minima (the potential false positives). A low value for γ indicates a good separability property of the distance functions. As for the PR-based evaluation, the soft performance measures shown in Table 2 support the usefulness of our enhancement strategies.

5. APPLICATIONS AND FUTURE WORK

Based on the segmentation of the folk song recordings, we now sketch some applications that allow folk song researchers to include audio material in their investigations. Once having segmented the audio recording into stanzas, each audio segment can be aligned with the MIDI reference by a separate MIDI-audio synchronization process with the objective to associate note events given by the MIDI file with their physical occurrences in the audio recording, see [9]. The synchronization result can be regarded as an automated annotation of the entire audio recording with available MIDI events. Such annotations facilitate multimodal browsing and retrieval of MIDI and audio data, thus opening new ways of experiencing and researching music [2]. Furthermore, aligning each stanza of the audio recording to the MIDI reference yields a multi-alignment between all stanzas. Exploiting this alignment, one can implement interfaces that allow a user to seamlessly switch between the various stanzas of the recording thus facilitating a direct access and comparison of the audio material [9]. Finally, the segmentation and synchronization techniques can be used for automatically extracting expressive aspects referring to tempo, dynamics, and articulation from the audio recording. This makes the audio material accessible for performance analysis, see [13].

For the future, we plan to extend the segmentation scenario dealing with the following kind of questions. How can the segmentation be done if no MIDI reference is available? How can the segmentation be made robust to structural differences in the stanzas? In which way do the recorded stanzas of a song correlate? Where are the consistencies, where are the inconsistencies? Can one extract from this information musical meaningfully conclusions, for example, regarding the importance of certain notes within the melodies? These questions show that the automated processing of folk song recordings constitutes a new challenging and interdisciplinary field of research with many practical implications to folk song research.

Acknowledgement. The first two authors are supported by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University. Furthermore, the authors thank Anja Volk and Peter van Kranenburg for preparing part of the ground truth segmentations.

6. REFERENCES

[1] M. A. Bartsch and G. H. Wakefield, *Audio thumbnailing of popular music using chroma-based representations*, IEEE Trans. on Multimedia, 7 (2005), pp. 96–104.

[2] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen, *Multimodal presentation and browsing of music*, in Proceedings of the 10th International Conference on Multimodal Interfaces (ICMI 2008), 2008.

[3] A. de Cheveigné and H. Kawahara, *YIN, a fundamental frequency estimator for speech and music*, The Journal of the Acoustical Society of America, 111 (2002), pp. 1917–1930.

[4] E. Gómez, *Tonal Description of Music Audio Signals*, PhD thesis, UPF Barcelona, 2006.

[5] M. Goto, *A chorus-section detecting method for musical audio signals*, in Proc. IEEE ICASSP, Hong Kong, China, 2003, pp. 437–440.

[6] N. Hu, R. Dannenberg, and G. Tzanetakis, *Polyphonic audio matching and alignment for music retrieval*, in Proc. IEEE WASPAA, New Paltz, NY, October 2003.

[7] Z. Juhász, *A systematic comparison of different European folk music traditions using self-organizing maps*, Journal of New Music Research, 35 (June 2006), pp. 95–112(18).

[8] F. Wiering, L. P. Grijp, R. C. Veltkamp, J. Garbers, A. Volk, and P. van Kranenburg, *Modelling folksong melodies*, Interdiciplinary Science Reviews, 34.2 (2009), forthcoming.

[9] M. Müller, *Information Retrieval for Music and Motion*, Springer, 2007.

[10] M. Müller and M. Clausen, *Transposition-invariant self-similarity matrices*, in Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007), September 2007, pp. 47–50.

[11] J. Serrà, E. Gómez, P. Herrera, and X. Serra, *Chroma binary similarity and local alignment applied to cover song identification*, IEEE Transactions on Audio, Speech and Language Processing, 16 (2008), pp. 1138–1151.

[12] P. van Kranenburg, J. Garbers, A. Volk, F. Wiering, L. Grijp, and R. Veltkamp, *Towards integration of music information retrieval and folk song research*, Tech. Report UU-CS-2007-016, Department of Information and Computing Sciences, Utrecht University, 2007.

[13] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic, *In search of the Horowitz factor*, AI Mag., 24 (2003), pp. 111–130.

SUPPORTING FOLK-SONG RESEARCH BY AUTOMATIC METRIC LEARNING AND RANKING

Korinna Bade, Andreas Nürnberger, Sebastian Stober
Otto-von-Guericke University Magdeburg
Faculty of Computer Science
{korinna.bade,andreas.nuernberger,stober}@ovgu.de

Jörg Garbers, Frans Wiering
Utrecht University
Department of Information and Computing Sciences
{garbers,fransw}@cs.uu.nl

ABSTRACT

In folk song research, appropriate similarity measures can be of great help, e.g. for classification of new tunes. Several measures have been developed so far. However, a particular musicological way of classifying songs is usually not directly reflected by just a single one of these measures. We show how a weighted linear combination of different basic similarity measures can be automatically adapted to a specific retrieval task by learning this metric based on a special type of constraints. Further, we describe how these constraints are derived from information provided by experts. In experiments on a folk song database, we show that the proposed approach outperforms the underlying basic similarity measures and study the effect of different levels of adaptation on the performance of the retrieval system.

1. INTRODUCTION

Folk song researchers detect and document relations between folk songs and their performances. This helps to understand oral transmission. Today, folk song researchers can digitally encode their transcriptions using common music notation editors and use computational methods to detect similarities between songs. However, as there are different ways to detect features in music, there are many different ways to compare songs. Usually, a single computational similarity value will not match directly with the classification criteria that a musicologist applies. Therefore, we choose a weighted linear combination of different basic similarity measures. Depending on the retrieval task at hand, the optimal weighting (with best retrieval performance) of such a complex similarity measure may differ.

In this paper, we describe a metric learning approach that can derive a good weighting in a semi-supervised manner. We apply constraint-based metric learning and formalize the weight adaptation as an optimization problem that is solved by gradient descent. Constraints that guide the adaptation process can be derived from an existing classification of tunes from a collection. We compare different ways of employing the derived similarities to support different browsing and classification tasks in a system that accepts both previous classified and unclassified queries.

The main **contribution** of this paper lies in describing and evaluating a general methodology that allows folk-song researches to automatically generate complex task-specific similarity metrics from basic similarity measures.

2. RELATED WORK

Metric learning has been a topic of interest in general information retrieval for some time, as using a suitable similarity measure is crucial for the performance of many commonly used approaches for clustering, classification or ranking. The general objective is either to get a query closer to the relevant objects (in a classic retrieval scenario) or to refine the decision boundary between relevant and irrelevant objects (in a classification scenario which does not necessarily require a query). The highly subjective nature of perception and the large variety of ways to represent and compare music in many "plausible" ways make it hard to manually define and tweak a metric according to the characteristics of the input data and the specific retrieval task. Consequently, there exist only few approaches for direct manipulation of the metric as described e.g. in [1] and [2]. In contrast to this, our approach allows a semi-supervised metric adaptation. This requires some labeled objects as training data. For the experiments discussed in this paper, such data was already provided. However, if such information is not available a priori, a relevance feedback approach is usually taken where a user is asked to judge on the relevance of some objects.

The idea of incorporating relevance feedback to improve the performance of an information retrieval system goes back to the 1970s [3]. Since, it has been widely applied and further elaborated – primarily in text but also in image retrieval. Recently, in the field of music information retrieval, several approaches using explicit feedback have been presented that adaptively combine the results of different music representation schemes [4], associate different music similarity perception models with users [5], learn to discriminate between similar and dissimilar pieces [6], adapt to the way of querying by taking into account user-specific humming errors [7], or generate user-adaptive play lists [8–11]. Alternatively, the required information may be collected through the analysis of user actions such as the skipping behavior [12] or manual rearrangement of objects on a map through drag & drop [13].

All these approaches are related to the one presented in this paper in that they rely on some form of metric adaptation. Our approach differs from these in two ways. First, it targets a significantly different application scenario. Second, none of the above approaches is based on constrained metric learning, which is applied here, except for the approach presented in [13] that uses similar constraints to guide the clustering of a self-organizing map.

3. OUR APPROACH

The goal of this work is to assist a folk song researcher in classifying new tunes (Section 3.1). Fundamental to this task is the computation of similarities between tunes. Different measures were developed in the past (Section 3.2). However, a particular musicological way of classifying songs is usually not directly reflected by just one of these measures. We show here how a weighted measure derived from a certain classification scheme can be automatically learned (Section 3.3). Based on this measure, we can support the classification of new items by presenting a ranked list (ordered by similarity) of already classified tunes (Section 3.4).

3.1 Expert classification support

Folk song researchers at the Meertens Institute study and classify folk song variants. Besides other means, songs are traditionally classified by assigning them a so called *melody norm*. This classification captures aspects of musical similarity and historical relationships. One tune cannot be part of more than one class.

The WITCHCRAFT project supports researchers by providing a system that enables browsing by musical content. The system's similarity measures operate on symbolic representations of tunes (Humdrum $**kern$ and MIDI format). A query melody is usually specified by clicking on a *search link* besides a database item. The system then ranks database tunes according to a chosen similarity measure.

Two types of ranking lists are supported by switching on/off a filter that is based on tune classification. In unfiltered mode, the *tune-ranking-list* presents all tunes ordered by similarity. This is handy when looking for all variants of a given song. In filtered mode, the *class-ranking-list* presents only the best ranked melody from each class. [1] Therefore, much fewer items are shown. This is handy when classifying a previously unclassified song or when questioning an existing classification.

3.2 Basic tune similarity measures

In this paper, we distinguish between *basic* similarity measures $sim_j(t_1, t_2)$ as introduced in the following and *linear combinations* thereof (Eq. (1)). Note that the *basic* similarity measures in this paper are themselves complex constructions of often more basic musical and mathematical transformations [14, 15]. However, in future work, we plan to also use more basic building blocks. In our experiments, we consider 14 similarity measures. However, the methods proposed in the following work with any set of measures. 11 of the 14 measures are taken from the *Simile* package. [2] These are *rawEd*, *diffEd*, *nGrSumCo*, *nGrUkkon*, *harmCorE*, *rhytFuzz*, *rhytGaus*, *opti1*, *opti3*, *accents_opti1* and *accents_opti2*. Two distance measures are based on the spectra of *Laplacean* and *Adjacency* graphs [16] and one is an unpublished pitch sequence edit distance, implemented by us. All distance measures were transformed to a similarity through $\mathrm{sim} = (1 + \mathrm{dist})^{-1}$.

3.3 Estimating a weighted similarity

Having given a certain number of expert classifications, the question remains whether we can find an optimal weighting of different tune similarity measures to reflect the similarity underlying these expert classifications. In particular, we are interested in the following weighted sum of n similarity measures:

$$\mathrm{sim}_{\boldsymbol{w}}(t_1, t_2) = \sum_{j=1}^{n} w_j \mathrm{sim}_j(t_1, t_2), \qquad (1)$$

with $w_j \geq 0$, and $\sum_{j=1}^{n} w_j = 1$.

The weight vector $\boldsymbol{w}$ can be learned by methods of constrained clustering, which target on learning a metric [17, 18]. In particular, must-link-before (MLB) constraints [18] can be used. Originally, MLB constraints were proposed for hierarchical clustering to describe the hierarchical relation between three different items. The constraint (i_x, i_y, i_z) states that items i_x and i_y should be linked on a lower hierarchy level than items i_x and i_z. For our problem at hand, we can use a similarity interpretation instead, i.e., items i_x and i_y should be more similar than items i_x and i_z.

Given a certain query tune q, we know from the expert classification, which other tunes t_r belong to the same (relevant) class and which tunes t_i are irrelevant. As the tunes of the same class should be ranked first and, thus, should be more similar, we can build MLB constraints of the following form: (q, t_r, t_i), which implies that

$$\mathrm{sim}(q, t_r) > \mathrm{sim}(q, t_i). \qquad (2)$$

Hence, the goal is to learn a weight vector $\boldsymbol{w}$, with which the fewest of the MLB constraints known for a certain query are violated. This can be achieved with a gradient descent search similar to the work in [18]. During learning, all constraint triples (q, t_r, t_i) are presented to the algorithm several times until convergence is reached. If a constraint is violated by the current similarity measure, the weighting is updated by trying to maximize

$$obj(q, t_r, t_i) = \mathrm{sim}_{\boldsymbol{w}}(q, t_r) - \mathrm{sim}_{\boldsymbol{w}}(q, t_i), \qquad (3)$$

[1] We found that taking the maximum leads to better results than taking the average of all the similarities.

[2] http://doc.gold.ac.uk/isms/mmm/SIMILE_algo_docs_0.3.pdf

which can be directly derived from (2). This leads to the weight update rule of each individual weight w_j

$$w_j = w_j + \eta \Delta w_j, \quad \text{with} \tag{4}$$

$$\Delta w_j = \frac{\partial obj(q, t_r, t_i)}{\partial w_j} = \mathrm{sim}_j(q, t_r) - \mathrm{sim}_j(q, t_i) \tag{5}$$

where η is the learning rate defining the step width of each adaptation step.

However, this computation does not ensure the bounds on w_j given earlier. To achieve this, an additional step is added that, first, sets all negative weights to 0 and then normalizes the weights to sum up to 1. The complete algorithm is summarized in Figure 1.

```
**learnWeights**(query tune $q$, tunes $T$, expert classification $C$, $\forall k = 1..n$ : similarity $\mathrm{sim}_k$)
  Determine constraints $MLB$ from $T$ and $C$
  Initialize $\boldsymbol{w}$: $\forall j : w_j := 1/n$
  **repeat**
    **for all** $(q, t_r, t_i) \in MLB$ **do**
      **if** $\mathrm{sim}_{\boldsymbol{w}}(q, t_r) \leq \mathrm{sim}_{\boldsymbol{w}}(q, t_i)$ **then**
        $\forall j$ : compute $\Delta w_j$
        $\forall j : w_j := \max(0, w_j + \eta \Delta w_j)$
        $sum_{\boldsymbol{w}} = \sum_j w_j$
        $\forall j : w_j := w_j / sum_{\boldsymbol{w}}$
      **end if**
    **end for**
  **until** convergence
  **return** $\boldsymbol{w}$
```

Figure 1. The weight learning algorithm

This algorithm learns an *individual weighting* $\boldsymbol{w}^q$ based on the set of MLB constraints for a single query q. However, a weighting that works well for several queries would be more useful. In specific, it is interesting to learn *class weightings* $\boldsymbol{w}^{cl(t)}$ that hold for all queries of the same class $cl(t)$ of a tune t and an *overall weighting* $\boldsymbol{w}^a$ that holds for all queries. These can be computed by the same algorithm using the combined constraint sets from all considered rankings.

3.4 Querying with unclassified tunes

The approach from the previous section can learn an optimal similarity measure based on an expert classification. If new tunes are added to a collection, no expert classification is available at first and, hence, no weights optimized for this query are available based on which a ranking list could be build. If there is no perfect global measure that can be applied to all queries, a different strategy can be followed for this query to build the ranking.

This is based on the already known good weightings of all database tunes, which were determined by the method described in the previous section. If we assume that similar songs also have a similar optimal weighted similarity, we can estimate a weighting for the new query tune q by picking the weighting scheme of the closest tune t_{best} in the database. This can be seen as a case-based approach [19] where each tune in the database and its associated weighted similarity correspond to a case and these stored cases are used to decide how to handle the new case, i.e., how to weight concerning the query tune.

However, this does not yet fully solve the problem, because a weighting scheme is already required to find the closest tune t_{best} for a query. A straight forward approach is to use an overall weighting $\boldsymbol{w}^a$. Alternatively, the more specific class weighting $\boldsymbol{w}^{cl(t)}$ or the individual tune weighting $\boldsymbol{w}^t$ associated with each database tune t can be used, because we already know that these similarities are well suited for comparing any tune with t.[3] We will select the case with the largest (local) similarity and use its weighting to finally rank all tunes in the database according to the query tune q: $\boldsymbol{w}_{best} = \arg\max_{\boldsymbol{w}^t} \ \mathrm{sim}_{\boldsymbol{w}^t}(q, t)$.

For ranking, we can also use any of the weightings associated with the closest case t_{best}, i.e., $\boldsymbol{w}^{t_{best}}$, $\boldsymbol{w}^{cl(t_{best})}$ and $\boldsymbol{w}^a$, where the latter is obviously the same for any database tune. In Section 4.4, we use the notation $\boldsymbol{w}_2 \circ \boldsymbol{w}_1$ to indicate that the first step (closest case selection) was performed using a similarity based on $\boldsymbol{w}_1$ and the second step (ranking) is based on $\boldsymbol{w}_2$.

4. EXPERIMENTS

We conducted experiments to study how well the different weighting techniques perform for already classified (Sec. 4.2) and unclassified tunes (Sec. 4.4) with respect to the two different ranking lists (Sec. 4.1). Further, we analyzed the stability of the learned weighting schemes (Sec. 4.3).

4.1 Dataset and measure evaluation method

Our evaluation is done on 360 well understood single melodic strophes (one strophe per recording) described in [20]. The tunes are classified into 26 disjunct classes. For each pair of tunes all 14 basic similarity measures are considered. These $14 \cdot 360^2$ similarity values are precalculated and need not be recomputed in the learning algorithm and in the construction of ranking lists.

As in the application system (Sec. 3.1), our algorithms produce for each (combined) similarity measure and query tune a tune-ranking-list of all database tunes and a class-ranking list. Both are ordered by decreasing computed similarity. All tunes with the same class as the query are marked as being a relevant result. This gives the ground truth for evaluating the ranked lists. As measures, we compute the *average precision* and *average recall* per rank on the tune-ranking-lists for the set of evaluated queries. For evaluation of class-ranking lists, we are interested in the position of the correct class in such a list. We present here the number of misclassifications at rank 1, the *average rank* of the correct class and the *average inverse rank* over all considered queries. The latter average is less sensitive for single extremely bad class retrievals.

[3] Please note that in this case different weightings are used to compute the similarity to different database tunes, which leads to local distortions of the similarity space around each case. While such a locally distorted metric is unsuitable for the computation of the entire ranking, it may still be useful to retrieve only t_{best} as shown in the experiments in Sec. 4.4.

4.2 Querying with classified tunes

In this section we study the retrieval performance of learned weights (cf. Sec. 3.3) and, thus, whether an automatically determined combination of different existing similarity measures performs better than the individual ones. We consider three weightings of different specificity as motivated in Section 3.3: query-specific weighting $\boldsymbol{w}^q$, class-specific weighting $\boldsymbol{w}^{cl(q)}$ and overall weighting $\boldsymbol{w}^a$. The precision/recall curves for these cases are shown in Fig. 3 (left). Additionally, the performance plots of the two best basic similarity measures, $rawEd$ and $opti1$ have been included into the figure for comparison. Table 1 (top) shows the corresponding evaluation of the class-ranking-lists, ordered by best performance, which gives the same order for all three measures, i.e., *average rank* of correct class (smaller is better), *average inverse rank* (larger is better), and the number of *wrong* classifications (inspecting the first rank).

Not surprisingly, using $\boldsymbol{w}^q$ for similarity computation results in the best retrieval performance. It marks the upper bound of what can be achieved with the learning algorithm. Further, it can be observed that $\boldsymbol{w}^{cl(q)}$ indeed performs better than $\boldsymbol{w}^a$. However, if weights get more specific, the danger of overfitting exists. We will discuss this problem in Section 4.4. Nevertheless, this evaluation indicates that there might not be a single perfect overall similarity measure that can be used in general. Instead data/problem specific measures might be needed, which are especially interesting if they can be determined automatically as through our presented method.

It is interesting to see, that the overall weight performs worse than the best basic similarity measure (*rawEd*) in most precision/recall regions (although only slightly) but that *rawEd* performs worse than all shown measures for the class-ranking-lists. This is caused by the convergence behavior of the algorithm, which is not guaranteed to find a global optimum but a local one.

4.3 Stability of the weighting scheme

In order to assess the stability of the individual weighting schemes throughout the different classes, we conducted two experiments. In the first experiment, we analyzed the individual weightings obtained for the classified tunes (cf. Section 4.2) with respect to the classes. The following two measures were computed for each class:

The *average pairwise inner-class similarity* is computed as the average of the similarity of the weightings for all pairs of tunes within the specific class:

$$\overline{\mathrm{sim}}_{inner}(C) = \underset{t_1,t_2\in C, t_1\neq t_2}{\mathrm{average}} \left\{\mathrm{sim}_{cos}\left(\boldsymbol{w}^{t_1}, \boldsymbol{w}^{t_2}\right)\right\}. \quad (6)$$

Analogously, the *average pairwise cross-class similarity* is computed as the average of the similarity of the weightings for all pairs of tunes where one tune belongs to the specific class and the other to a different class:

$$\overline{\mathrm{sim}}_{cross}(C) = \underset{t_1\in C, t_2\notin C}{\mathrm{average}} \left\{\mathrm{sim}_{cos}\left(\boldsymbol{w}^{t_1}, \boldsymbol{w}^{t_2}\right)\right\}. \quad (7)$$

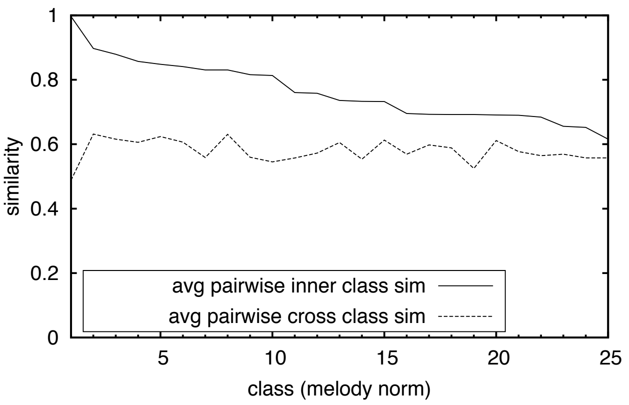


Figure 2. Average pairwise inner- and cross-class similarity of the individual weightings per class (sorted).

For the comparison of two weightings the cosine similarity

$$\mathrm{sim}_{cos}\left(\boldsymbol{w}^{t_1}, \boldsymbol{w}^{t_2}\right) = \frac{\boldsymbol{w}^{t_1}\cdot\boldsymbol{w}^{t_2}}{\|\boldsymbol{w}^{t_1}\|\|\boldsymbol{w}^{t_2}\|} \quad (8)$$

was used. Fig. 2 shows the computed values for all classes (sorted by descending inner-class value for better readability). The inner-class value is always significantly higher than the respective cross-class value. It can be concluded that the individual weightings of tunes belonging to the same class are in general distinct from those belonging to others which explains the usefulness of class weights ($\boldsymbol{w}^{cl(q)}$ in Sec. 4.2). The generally high cross-class values (above 0.5) can be interpreted as an indicator for the existence of a useful overall weighting scheme ($\boldsymbol{w}^a$ in Sec. 4.2).

For the second experiment, we left out one to five relevant tunes selected randomly from a ranking during the learning process for individual weights. The procedure was repeated ten times for each number of excluded tunes. We then compared the $5\cdot 10$ resulting weighting schemes with the one learned with all available information. To save time, we limited the number of tunes used as queries to two for each of the 26 classes resulting in $2\cdot 26\cdot 5\cdot 10 = 2600$ samples compared.

The number of excluded tunes did not seem to have a large observable effect in our experiment. The different weights learned for the same tune with a differing set of relevant tunes were almost identically with an average similarity of 0.969 ($\sigma = 0.086$). Only a few outliers could be measured, the worst with a minimal similarity of 0.278. From the results we can conclude that the learning algorithm still produces stable results even if almost half of the relevant tunes are removed.

4.4 Querying with unclassifed tunes

In this section we study the retrieval performance for previously unclassified tunes, i.e., for which no previously learned weight $\boldsymbol{w}^q$ or $\boldsymbol{w}^{cl(q)}$ exists. Following the case-based approach described in Sec. 3.4, Fig. 3 (middle and right) shows the respective precision/recall curves. We thereby consider two different real-world situations.

In the first case (Fig. 3; middle) the query tune represents a new tune and is therefore not part of the weight learning. However, the other tunes of the same class are already in the database and therefore used for learning. But of course, the system does not know during ranking which ones these are. In the other case (Fig. 3; right) all tunes from the query tune's class were not used for learning, simulating an entirely new class that shall be added to the database. Thus, no information on this class was available for learning. This is of course an even harder case. For computation of the precision and recall values, all tunes were ranked according to the query tune, including the songs of the unknown class. For both scenarios, we used weights with different specificity in the two steps of the case-based approach (cf. Sec. 3.4).

As expected, the comparison of both diagrams shows that the performance of the learned measures is lower for the harder case of a new class than in the case of a new tune from a known class. The *rawEd* measure can thereby be used as a point of comparison, because it has the same curve in both cases. It showed that the more specific weightings, most notably $\boldsymbol{w}^t \circ \boldsymbol{w}^t$, are much better in the middle graph than in the right one. This is because members of the same tune family can be detected as a case, if they are already in the database. However, specific weightings from other tune families are less fitting. Thus, the approach fails for a new tune family, where no good specific measures are in the database yet. In the middle graph, $\boldsymbol{w}^a$ is best used to establish a case and $\boldsymbol{w}^t$ of that case to finally rank: $\boldsymbol{w}^t \circ \boldsymbol{w}^a$. This approach is also quite good in the right graph, although using only $\boldsymbol{w}^a$ is slightly better. *rawEd* is better at the end of the ranking, while it is worse at the beginning. This is also reflected in the evaluation of class-ranking-lists (Tab. 1). Here, *rawEd* performs worst. With respect to automatic classification, $\boldsymbol{w}^a$ performs best with the fewest errors in the first rank. A comparison of $\boldsymbol{w}^t \circ \boldsymbol{w}^a$ with $\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^a$ would be interesting, but the experimental data for $\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^a$ is not available, yet.

As a remark it shall be noted that the described methodology of simulating new tunes is very time-consuming because for each considered query the learning has to be redone without the respective information. Therefore, the experiments were done with only 78 query melodies, three melodies from each melody norm. For the development of new similarity measures, the biased evaluation without resampling as used in Section 4.2 can be used to get a rough idea of which measure might be more promising. However, it can never replace a final unbiased evaluation as in this section. Furthermore, the choice of melodies showed a significant impact on the results. Using, e.g., only the reference melodies from [20], the learned measures perform much better in comparison to *rawEd* - also in the most challenging case, while other query tunes are harder to handle. For our evaluations we used the reference melody and two randomly picked other melodies.

Table 1. Evaluation of the class-ranking-lists. **Top:** classified tunes (Sec. 4.2). **Middle:** unclassified tunes of a known class (Sec. 4.4). **Bottom:** unclassified tunes of an unknown class (Sec. 4.4).

Measure	Rank	Inverse	1st Wrong
$\boldsymbol{w}^q$	1.042	0.989	6 / 360
$\boldsymbol{w}^{cl(q)}$	1.083	0.985	9 / 360
opti1	1.169	0.975	14 / 360
$\boldsymbol{w}^a$	1.172	0.974	14 / 360
rawEd	1.233	0.967	16 / 360
$\boldsymbol{w}^t \circ \boldsymbol{w}^a$	1.218	0.969	4 / 78
$\boldsymbol{w}^a$	1.231	0.981	2 / 78
$\boldsymbol{w}^t \circ \boldsymbol{w}^t$	1.244	0.957	5 / 78
$\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^{cl(t)}$	1.346	0.976	2 / 78
rawEd	1.410	0.946	5 / 78
$\boldsymbol{w}^a$	1.218	0.982	2 / 78
$\boldsymbol{w}^t \circ \boldsymbol{w}^a$	1.244	0.971	3 / 78
$\boldsymbol{w}^t \circ \boldsymbol{w}^t$	1.282	0.942	7 / 78
$\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^{cl(t)}$	1.359	0.970	3 / 78

5. CONCLUSION

We described an adaptive metric learning approach based on constrained clustering that can be used in folk song research to learn a task-specific similarity measure in form of a weighted linear combination of several basic similarity measures. Individual, class and overall weightings provide different levels for specificity of the adaptation. Experiments on a data set of pre-classified folk songs showed that the combined similarity measures using these weightings can outperform the original basic similarities for ranking and automatic classification.

Future experimental work comprises incorporating more basic similarity measures that capture different aspects of the tunes to be classified. Further, the impact of the differing value distributions (within the fixed $[0, 1]$ interval) for the different basic similarities needs to be studied in further experiments as it might cause a bias in the learned weighting schemes.

Future musicological work includes studying clusters of similar weightings. As different weightings represent different metrics, they select different features that separate melody classes. Within a melody norm, several distinct weight clusters suggest the introduction of sub-melody-norms that might be helpful for folk song research. On the other hand, weight clusters shared by different melody norms could be studied to improve the case-based approach. If, e.g., rhythmically ragged melodies generally lead to higher weighted rhythmical similarity measures, then raggedness should be used to select weights instead of rhythmical similarity. For a better support of folk song researchers, the algorithm should be integrated into a graphical user interface. In this context, possible interaction scenarios, e.g., for expert-driven development of new similarity measures, could be examined.

Acknowledgments. This work was supported by the Netherlands Organization for Scientific Research within the WITCH-CRAFT project NWO 640-003-501, the German Research Foundation (DFG), and the German National Merit Foundation. Special thanks to contributing developers and data providers.

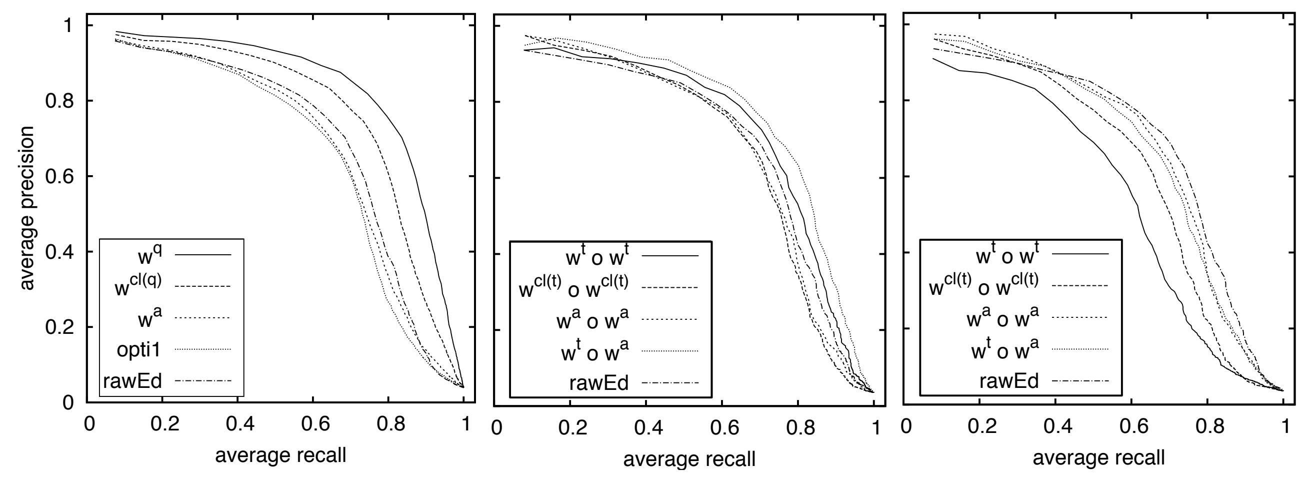


Figure 3. Precison / Recall plots for tune-ranking-lists. **Left:** classified tunes. **Middle:** unclassified tunes of a known class. **Right:** unclassified tunes of an unknown class.

6. REFERENCES

[1] S. Baumann and J. Halloran. An ecological approach to multimodal subjective music similarity perception. In *Proc. of Conf. on Interdisciplinary Musicology*, 2004.

[2] F. Vignoli and S. Pauws. A music retrieval system based on user driven similarity and its evaluation. In *ISMIR*, 2005.

[3] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*, 1971.

[4] T. Mandl and C. Womser-Hacker. Learning to cope with diversity in music retrieval. In *ISMIR*, 2002.

[5] D. N. Sotiropoulos, A. S. Lampropoulos, and G. A. Tsihrintzis. MUSIPER: a system for modeling music similarity perception based on objective feature subset selection. *User Modeling and User-Adapted Interaction*, 18(4):315–348, 2008.

[6] M. I. Mandel, G. E. Poliner, and D. P. W. Ellis. Support vector machine active learning for music retrieval. *Multimedia Systems*, 12(1):3–13, 2006.

[7] D. Little, D. Raffensperger, and B. Pardo. A query by humming system that learns from experience. In *ISMIR*, 2007.

[8] K. Wolter, C. Bastuck, and D. Gärtner. Adaptive user modeling for content-based music retrieval. In *Proc. of the 6th Int. Workshop on Adaptive Multimedia Retrieval*, 2008.

[9] K. Hoashi, K. Matsumoto, and N. Inoue. Personalization of user profiles for content-based music retrieval based on relevance feedback. In *MULTIMEDIA'03: Proc. of the 11th ACM Int. Conf. on Multimedia*, 2003.

[10] M. Grimaldi and P. Cunningham. Experimenting with music taste prediction by user profiling. In *MIR '04: Proc. of the 6th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, 2004.

[11] S. Pauws and B. Eggen. PATS: Realization and user evaluation of an automatic playlist generator. In *ISMIR*, 2002.

[12] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In *ISMIR*, 2005.

[13] S. Stober and A. Nürnberger. Towards user-adaptive structuring and organization of music collections. In *Proc. of 6th Int. Workshop on Adapt. Multimedia Retr.*, 2008.

[14] D. Müllensiefen and K. Frieler. *The SIMILE algorithms documentation 0.3*, 2006.

[15] D. Müllensiefen and K. Frieler. Cognitive adequacy in the measurement of melodic similarity: Algorithmic vs. human judgments. *Computing in Musicology*, 13, 2004.

[16] A. Pinto, R. H. van Leuken, M. F. Demirci, F. Wiering, and R. C. Veltkamp. Indexing music collections through graph spectra. In *ISMIR*, 2007.

[17] K. Bade and A. Nürnberger. Personalized hierarchical clustering. In *Proc. of the IEEE / WIC / ACM Int. Conf. on Web Intelligence*, 2006.

[18] K. Bade and A. Nürnberger. Creating a cluster hierarchy under constraints of a partially known hierarchy. In *Proc. of the SIAM Int. Conf. on Data Mining*, 2008.

[19] A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[20] A. Volk, P. Van Kranenburg, J. Garbers, F. Wiering, R. C. Veltkamp, and L. P. Grijp. A manual annotation method for melodic similarity and the study of melody feature sets. In *ISMIR*, 2008.

EXPLORING SOCIAL MUSIC BEHAVIOR: AN INVESTIGATION OF MUSIC SELECTION AT PARTIES

Sally Jo Cunningham **David M. Nichols**
Department of Computer Science
University of Waikato
Hamilton, New Zealand
{sallyjo, dmn}@cs.waikato.ac.nz

ABSTRACT

This paper builds an understanding how music is currently listened to by small (fewer than 10 individuals) to medium-sized (10 to 40 individuals) gatherings of people—how songs are chosen for playing, how the music fits in with other activities of group members, who supplies the music, the hardware/software that supports song selection and presentation. This fine-grained context emerges from a qualitative analysis of a rich set of participant observations and interviews focusing on the selection of songs to play at social gatherings. We suggest features for software to support music playing at parties.

1. INTRODUCTION

Our experience of music includes both individual and group settings. When music is heard in social situations [1] the question which follows is: who selects the music?

In this paper we explore issues of social music selection in the context of small private gatherings such as parties. The portability of digital music, on devices such as iPods, enables people to easily bring their own music but the selection of music to be played is largely based on the social roles of the participants. Ethnographic methods are used to understand these settings and so inform the design of systems for supporting shared music experiences.

Section 2 outlines previous work on social music systems. We then describe our methods and discuss the support provided by media players. Section 5 outlines the collaborative nature of music selection and we conclude by comparing our results with existing systems.

2. EXISTING SOCIAL MUSIC SYSTEMS

There is "little in the literature to suggest how to design new and unique tools that facilitate social music use within and between the different contexts in which people work, play, and otherwise live their lives" [2]. Rentfrow and Gosling [3] show that music plays an important part in many peoples' lives. Music is a conversation topic, "individuals' music preferences convey consistent and accurate messages about their personalities" and music-genre stereotypes are used when forming opinions of others [3]. Further, "synchronized music consumption among people in physical proximity, as it happens in clubs or during parties, can create a strong emotional connection, more than what an asynchronous download of music over distance could provide" [4].

However, North et al. [1] note that a "lack of ecological validity" constrains much of the research on the social and psychological impact of music in everyday life. North et al. also report that their "data indicate that the great majority of listening episodes occurred in the presence of other people". Several systems have been designed to enhance this shared experience of music.

Several social music systems (e.g. SocialPlaylist [5], tunA [4], Push!Music [6]) use personal mobile technology, such as iPods, PDAs and mobile phones, reflecting the increasing portability of music collections [7]. Liu and Reimer [5] recommend that such systems provide smooth integration between personal and social modes, as inevitably users will occasionally prefer individual selections. In tests of Push!Music the "sharing of music became a prompt for social interaction, but this happened only between users who already knew each other and were socializing face-to-face" [6]. Nettamo et al. [8] note that even though mobile music is widespread, music in the home is often played via computers and that the "home PC acted as music hub".

An alternative to these person-to-person mobile forms of shared music is to allow voting or collaborative recommendation in public spaces. Deployment of the Jukola system [9] allowed users to express their musical preferences and this encouraged "debate, conversation and negotiation around music." [9]. Pering et al. [7] used interviews and observations of music in shared spaces to identify four key types of stakeholders: providers, contributors, proprietors, and listeners. Pering et al also note that the use of audio in shared spaces today may well soon be generalized to other media such as photos and video [7].

The MUSICtable system is designed for a further context of use, the "private social gathering" [10]:

> The user interface of the PC-based digital music player clearly does not support music selection by multiple people in a social situation. One manifesta-

tion…is…"separate party syndrome," wherein a small number of people tend to gather around the desktop computer … dominating the selection of music.

The PartyVote system [11] takes the voting concept from Jukola and applies it in small group situations such as parties. Both PartyVote and MUSICtable also include visualizations to provide awareness feedback to the voters. The Smart Party [12] reflects the preferences of users by dividing a party into several rooms, where each room's music adapts to the preferences of the people present, changing as guests move around the party. Bluemusic supports the same idea, personalization by being there, using Bluetooth from portable devices [13].

Many of these systems are designed without an obvious grounding in the detailed behavior of users with current music technology in social situations [2]. In this paper we investigate the use of music in these "private social gatherings" through ethnographic methods. We also compare our findings with those from other social music settings and consider the interaction between the setting, the users' roles, the technology and the resulting experience.

3. DATA GATHERING

Our research uses data collected in a third year university HCI course. The course focuses on qualitative and quantitative techniques for gaining an understanding of user needs, goals, and preferences, and using these insights to inform the user requirements and initial prototyping stages of a user-centered software development effort. This course adopts the 'practical approach' to incorporating ethnography into software design, as advocated by Randall et al. [14]. Students work individually over the semester to design and prototype a system based around the given focus application, where their designs are informed by a series of ethnographic investigations into behavior associated with the application domain.

In 2008, the students explored the problem of designing a system to support groups of people in selecting and playing music. They began by performing participant observations of social gatherings that included music, with the observations focusing on how the music is chosen for playing, how the music fits in with the other activities being conducted, who supplies the music, and how/who changes the songs or alters the volume. The students then explored subjective social music experiences through interviews, both of themselves ('auto-ethnographies' [15]) and of a friend. These interviews explored aspects of a social gathering that made it more, or less, likely for attendees to participate in selecting the songs, and the social factors that made attendees feel more, or less, comfortable in selecting and playing music. The students also critiqued the usefulness of existing systems for collaborative music selection and playing in social situations.

Thirty student investigators gathered ethnographic data (Table 1). The students were encouraged to construe 'social gathering' very broadly, and so performed participant observations in a variety of settings (including car trips, bars, café's, private homes, and religious institutions) and with a range of size (from two friends in a dormitory room to a hundreds at a rave). For this paper we focus on small- (10 or fewer attendees) and medium-sized gatherings (from 10 to 40 attendees), that are not professionally organized or occur in commercial settings (e.g., informal parties in student flats, birthday parties, a Friday night get-together, a computer gaming session, etc.). Music might be the primary focus of the event (e.g., a gathering to listen to a friend's new CDs) or be a part of the background (e.g., a quiet evening of conversation). This focus often shifts—a party may begin with a meal accompanied by soft music, move to louder music and dancing, and cycle back and forth through the evening.

A set of 43 participant observations met these criteria: 29 small- and 14 medium-sized gatherings. The observations lasted from a minimum of 15 minutes to a maximum of 4 hours, with an average of a little over 2 hours (113 minutes). A total of 88 interviews provided deeper interpretations of the observation experiences. In the following sections, the investigators are identified by a letter/number code (e.g., Participant K, Participant A2).

Table 1. Characteristics of student investigators

Male	Female	National Origin	Count
34	6	NZ/Australia	14
		China	9
		Mid-East	5
		Other	2

Grounded Theory methods [16] were used to analyze the student summaries of their participant observations, interviews, and system critiques. With Grounded Theory, the researchers attempt to generate theory from data, through an inductive analysis of the data. This present paper teases out the behaviors and social issues that influence how songs are selected for playing at parties.

4. MEDIA AND MUSIC PLAYER SUPPORT

The participant observations reference a staggering array of music media and players. While the majority of social gatherings were supported by digital media and players, as was expected, two included cassette tapes and several included 'old school' music CD players with only the rudimentary play, pause, and skip controls. The limitations imposed by cassettes and basic CD players are significant: it is not possible to browse through a cassette or a CD on a simple player; the songs must be played in the original sequence; both contain a limited number of songs, and skipping past songs that are disliked or that do not fit the developing atmosphere of the party further reduces the play time. Physically changing a cassette or CD introduces silences that may break the mood (though in practice it can be faster to physically swap out a CD than for an inexperienced user to wrestle with selecting songs using an unfamiliar piece of software).

An advantage of the cassette and simple CD player is that they can highlight musical knowledge or expertise in their owner. A carefully compiled party-themed mix cassette or mix CD can showcase the creator's ability to establish and sustain a mood—though at the cost of not being able to fine-tune the songs or their sequence of play as the event unfolds.

Portable MP3 players and computer–based music systems feature in the majority of gatherings reported in the participant observations, some quite elaborate (for example, [D] describes a setup in which 'the music was on a computer in another room and being streamed to the X-box 360 via a wireless network. The TV and X-box are connected to an amplifier that powers the surround speakers and the sub-woofer'). Similar home computer based music setups are described in [8]. The MP3 players, laptops, and portable hard drives can support extensive music collections, but is there ever enough storage space? ('Music was provided by me, but it was limited because I had a selection of songs on a 250GB hard drive …' [U]).

An MP3 player is not ideal as the primary device for selecting and playing songs; the limited physical controls (in particular, the lack of a keyboard) and the small or non-existent display ('the information you can see at one time is limited' [X]) make it difficult to search or browse through a collection to select songs. The larger display of a laptop or desktop computer affords music organization software that includes a larger set of searching and browsing facilities, but these more busy interfaces can be confusing to unfamiliar users. Further, there is no standard music organizer (Winamp, Windows Media Player, iTunes, and the command line MPlayer are mentioned in the participant observations and interviews), so the likelihood of a party-goer encountering an unfamiliar setup is high. As will be discussed in Section 5, fear of making mistakes while selecting music can deter people from participating in the selection of songs in a social situation.

5. COLLABORATIVE SELECTION OF MUSIC

We describe patterns in the social setting and expectations for playing, choosing, and changing music.

5.1 The Host, Guest of Honor, and Guests

A social gathering generally includes at least one Host (who may provide the venue and initiate implicit or explicit invitations, and who feels a sense of responsibility for creating an enjoyable occasion), and one or more Guests (attendees at the event). If music is a part of the event—and it commonly is—then the host can be expected to provide the initial stock of songs for a party and the hardware/software needed to play them. Choosing appropriate music is a significant responsibility: the set of songs played at an event and the order of play can have a dramatic impact on the atmosphere (Section N) of the event. Poor selection can have social repercussions: for example, Participant D reports of an interviewee that, '…she likes to host parties and have friends over and if they thort [sic] she had crap music or played crap music then they would not come over any more.'

Before the party, the Host creates the initial party playlist. This preparation may occur at the beginning of the party itself ([D]: 'me and one of the others spent about 30 min on the computer in the other room creating a play list of songs to be listened to'), or begin well in advance ([C]: "downloading music for the party a few days beforehand'). Frequently the playlist is crafted specifically for the event, but a Host may also develop a generic, reusable Party collection.

Sometimes Guests contribute to the party collection, beforehand or at the beginning of the party or as the party progresses--though the latter might be a bit of an insult to the host, as the unsolicited provision of supplemental music implies that the host's selections are not suitable ([D], of interviewee: 'if [the music at the party] 'sucked' then she would bring a CD or something so that she could change it.'). The Host is more likely to invite Guests to contribute songs to a party if the Host is unsure of their musical tastes or if it is a formal or commemorative occasion (for example, a 21st birthday party). The Host retains responsibility, however for selecting the final party playlist from the pool of contributions.

Some parties (eg, birthday) feature a special Guest. The Guest of Honor may or may not also be the Host, but the Guest of Honor assumes a similar role. Where the Host's song selections might be later altered by Guests (Section 5.2), the Guest of Honor's usually are not—changes to a Guest of Honor's playlist would constitute a more serious breach of party protocol. For example, Participant G reports, 'The music was selected by one person only, throughout the entire night. This person was the birthday girl. … The ipod was sitting on the stereo during the party, but it was made clear that nobody else was to adjust the music, except for the birthday girl.'

5.2 The Invitation

The Host normally assumes initial control over the music selection; as the event progresses, the Host may maintain control throughout the occasion, or may pass control to others. Permission to alter the gathering's playlist is passed through The Invitation: the Host explicitly or implicitly invites others to browse available collections and select songs. A Host might overtly encourage Guests to add, delete, or re-order songs on the playlist ('As host I will always make a short play list, long enough to last until most people arrive, then encourage anyone who comments on the music to change it.' [X]), or might more subtly indicate that alterations are acceptable by leaving the control to the music playing device in an easily accessible spot.

5.3 Maintaining the Atmosphere

While music is generally not the main focus for the gatherings described in the participant observations, it was an integral part of the occasions. Awareness of the music naturally ebbs and flows. When conversation flags, Guests are more attentive to the songs playing (music 'is there because during the breaks when people are not talking, the atmosphere feels awfully quiet so music helps lighten the mood' [K]). A song can spark new conversation by serving as a reminder of earlier occasions ('Participant A mentioned on more than one occasion a song's significance in his life, e.g. "When I was at school we used to play this song on our stereos at lunchtime"' [L]). Guests frequently show an interest in discussing unfamiliar songs (eg, [L] reports that Guests would 'occasionally ask questions or make comments about a certain song like, "who sings this song, I really like it" or "we should get this song"). An interest in learning more about new music is expected given that the music was selected in anticipation that it matches the Guests' tastes.

A common pattern is for a gathering to begin with quieter or less obtrusive music during an initial 'socializing' phase, move to 'faster, louder and less organized' [A] songs, and then end the evening with 'chilling out' music. A skilled Host monitors the Guests' interest in the music and its affect on the gathering's ambiance, and modifies the music when necessary to create or enhance the appropriate atmosphere ('During the night there were a number of situations where the music and the mood needed to change' [I]).

5.4 Skipping, Sampling, Searching, and Browsing

Changes to the party playlist are of two types: deletion of undesirable songs and insertion of new songs. The usual case for deletion is to stop the current song that is playing and move to another song—*skipping*. The simplest strategy for choosing a replacement song is simply to skip sequentially through the playlist, playing a few moments of each song (*sampling*) until an acceptable one is encountered. The audio effect is less than ideal, given the abrupt ends of skipped songs, but one benefit for the user is that the interaction is selecting a new song involves simply clicking a single 'next song' button. Skipping is also the strategy of choice for Guests who are unfamiliar with the unfamiliar with the searching/browsing facilities of the music software ('…people will feel uncomfortable [if they] stand in front of the computer for a long time, while they are finding the music they want to listen [to]' [T]), and/or Guests who do not want to expose their ignorance of the gathering's preferred music genres:

> '…when I get told [to] change the music I will simply skip enough songs until I find an improvement … For me this is mostly because I do not remember song names or even mainstream artists. If I have to chose music I have to resort to one of the limited number of artists I know or pick randomly.' [X]

Sampling may also occur outside the context of skipping, when a Guest or Host attempts to identify desirable songs from a pool of potential additions to the party playlist. If the individual cannot identify songs by the available metadata, then a 'good' sample is needed to decide whether to include a song on the playlist (where 'good' probably includes the chorus or other characteristic section of the song, rather than the beginning; 'often they want to skip to the chorus of a song to find out if it is the song they actually want' [G].

Searching and *browsing* to select songs are more rarely reported in the participant observations than skipping and sampling. Searching for a specific song requires some confidence that that song is actually accessible, and possibly knowledge of its approximate location on the physical device ('Participant B requested that another song she knew to be on the laptop be played. Participant A then queued up the requested song (which resided in a different directory)' [L]). Effective browsing (in the absence of sampling) require a deep familiarity with the specific songs in the collection: [X] reports of one successful browser that 'he has a much broader knowledge [of music] … he is able to scroll through large number of music titles and understand what many of them sound like based on the artist and title given and decide if they would be appropriate.'

6. SUPPORTING SOCIAL MUSIC USE

The specific music organization and selection software mentioned in the participant observations and interviews include Winamp, Windows Media Player, iTunes, and the command line MPlayer. Given that music is frequently an integral part of social, festive occasions, it is surprising that only iTunes avoids a clinical, 'somewhat dark' [X] appearance. It seems appropriate that interfaces should enhance the enjoyment and entertainment that people experience when listening to music and interacting with music collections—interfaces should be attractive and playful, appropriate to an enjoyable social gathering.

Existing music organization software was found to be adequate for supporting the Host in developing the initial party playlist—which is not surprising, given that the Host is usually interacting with his/her personal system and music collection. Difficulties arise when Guests or multiple Hosts contribute to the pre-party development of the initial party playlist (Section 5.1). Songs arrive on a variety of media (flash drives, MP3 players, CDs, external hard drives, downloaded from the Web), and in transferring them to the Host's system it is easy to lose metadata (artist, title, genre, etc.) or to discover that metadata values and schemes are incompatible (particularly genre). Better support is needed for creating a pool from multiple sources, and integrating them into a single (possibly ephemeral) collection.

As contributions are pooled, event-specific information is easily lost (eg, who contributed a song, which party Guests it might appeal to, and so forth)—further stressing the Host as s/he makes the final decisions on which songs to eliminate. Additional support for merging contributions into a draft playlist would be welcome—perhaps by encouraging contributing Guests to annotate groups of songs with a justification for its inclusion (for example, whether the songs have a strong beat for dancing, are suitable for chilling out at the end of the evening, etc.). An annotation facility is particularly important because inclusion criteria can be complex, idiosyncratic, and not well-matched to conventional metadata; eg, songs 'that a lot of people would know so they would sing along' [D], or songs suitable for both high school age and 'old(er) guests' [I]. It could be helpful to allow pre-party contributors to view the draft playlist and cast votes for the retention / deletion of songs, with the Host retaining ultimate responsibility for setting the initial playlist.

Creating the initial party playlist requires a great deal of insight into the musical tastes of the Guests and the anticipated atmosphere of the party, and a great deal of skill to match those to the available songs. It is difficult to see how this can be automated effectively. Smart Party [12], for example, automatically builds playlists from party attendees' personal music devices—a strategy that initially appears reasonable. But music in personal collections may not be suitable for public listening. An earlier study reports that subsets of a personal collection may represent 'guilty pleasures' (music that does not fit the public persona of the collection owner) [17], and it could be awkward to have those songs appear in a public party playlist. More fundamentally, personal favorites may not be suitable for a given social occasion: 'music in a group situation that is good for that occasion can be vastly different than music I would usually listen to normally. … music that is listened to with other people and when you are less focused on it can be different to your usual tastes or that you will put up with it even if you don't like it that much to not distance yourself from the group you are with.' [R]

Voting systems such as Jukola [9] and Party Vote [11] may finesse the potential social minefields of correctly interpreting when The Invitation has been given (Section 5.2), by providing an impartial mechanism for suggesting alterations to the music lineup without offending the Host (Section 5.1), and generally foiling 'pushy people, someone who just plays their own music and wont [sic] take any consideration for others' [E]. These systems, like the old-fashioned jukeboxes they derive from, clearly give permission to choose music—that is their primary function. [F] explicitly makes this connection, commenting that he particularly likes jukeboxes because 'the fact that the whole system is set up just so party goers can select music made me feel totally comfortable with using it to select the music I wanted to hear.'

As social gatherings are an opportunity to be exposed to new music (Section 5.3), it should be easy for Guests to access further information about the songs that are played. At a minimum this should include an easy-to-read screen that features basic song metadata (artist, title), with access to more detailed records (eg, lyrics, genre, 'maybe a little trivia associated with the song/artist' [L]). A few participant observations report Guests attempting to use the party's music software to learn more about a song, but an inexpert user can too easily select a song to play when intending to view its metadata ('This would cause people to get annoyed as they would be listening to a song and then someone would skip to a different one part way through [D]). Ideally there would be a clear differentiation between the interface elements that support playing music and those elements that support browsing/searching collections.

The skipping strategy for moving past songs that are disliked or that are inappropriate to the gathering's mood can itself disrupt the party's atmosphere; it is annoying and disconcerting for the song to stop abruptly ('People will have a notice when the song was changed in the middle…' [T]). The availability of a crossfade effect to smooth song transition would not completely solve the problem, but would be an improvement.

Sampling, or listening to brief portions of a song to make a play / no play decision (Section 5.4), could be made more efficient by allowing the user to skip to the chorus or other readily identifiable song extract (e.g., [18]). Alternatively, though aurally not as satisfactory, the system could support skimming through a song by increasing the speed of play ('allows them to quickly listen to the feature of the song' [P]).

A common scenario for sampling involves using it to build up a sequence of songs to play. But for the music management software encountered in the participant observations, it was difficult to 'stack' selections, so the person choosing plays samples until a single acceptable song is identifies; that plays, and then sampling begins again. If this occurs during the party, the mood can be significantly disrupted. Ideally, the user would be able 'to (privately) listen to previews of songs, … to make informed decisions on the music they select' [L], similar to the 'previewing' of tracks in professional DJ software.

Effectively supporting music searching and browsing (Section 5.4) remains an open research problem. Several student investigators suggested the inclusion of lyrics metadata would be the simplest and most straightforward way to support both direct search and browsing (by allowing the user to 'skim' a song without hearing the audio). Lyrics would also be helpful for gathering attendees who wished to sing along to the music.

The clearest directive that emerged from the participant observations was the importance of a simple, clean interface design, preferably with large, clearly labeled controls whose operation do not require fine motor movements. Interaction sequences should be brief and each step in a sequence should be clearly signaled, in large font ('normally the University student would absolutely drink beer while having a party' [Y]).

7. CONCLUSIONS

This study presents a rich picture of collaborative music selection among a large group of (primarily) university students in New Zealand. As such, the insights gained must be treated cautiously—a logical next step is to 'triangulate' through further studies involving participants with different backgrounds.

The environmental conditions revealed in the party observations differ significantly from the austere, controlled environment of a usability laboratory—and so lab testing would be likely to miss significant issues. Testing of a collaborative music system should occur in authentic environments and real social situations, to ensure that the interface is usable with, for example, limited lighting, a noisy setting, and intoxicated users.

8. REFERENCES

[1] A.C. North, D.J. Hargreaves, J.J. Hargreaves: "Uses of Music in Everyday Life," *Music Perception*, Vol. 22 No.1, pp. 41–77, 2004.

[2] F. Bentley, C. Metcalf, G. Harboe: "Personal vs. commercial content: the similarities between consumer use of photos and music," *Procs of CHI'06*, pp. 667-676, 2006.

[3] P.J. Rentfrow, S.D. Gosling: "Message in a Ballad: The Role of Music Preferences in Interpersonal Perception," *Psychological Science* Vol. 17 No. 3, pp. 236-242, 2006.

[4] A. Bassoli, J. Moore, S. Agamanolis.: "tunA: socialising music sharing on the move," In K. O'Hara and B. Brown (Eds), *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, Dordrecht: Springer, pp. 151-172, 2006.

[5] K. Liu, R.A. Reimer: "Social playlist: enabling touch points and enriching ongoing relationships through collaborative mobile music listening," *Procs of MobileHCI '08*, pp. 403-406, 2008.

[6] M. Håkansson, M. Rost, L.E. Holmquist: "Gifts from friends and strangers: a study of mobile music sharing," *Procs of ECSCW'07,* pp. 311-330, 2007.

[7] T. Pering, R. Want, L. Gardere, K. Vadas, E. Welbourne: "Musicology: Bringing Personal Music into Shared Spaces," *Procs of Fourth Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2007),* pp.1-8, 2007.

[8] E. Nettamo, M. Nirhamo, J. Häkkilä : "A cross-cultural study of mobile music: retrieval, management and consumption," *Proceedings of the 18th Australia Conference on Computer-Human Interaction (OZCHI '06)*, pp. 87-94, 2006.

[9] K. O'Hara, M. Lipson, M. Jansen, A. Unger, H. Jeffries, P. Macer: "Distributing the Process of Music Choice in Public Spaces," In K. O'Hara and B. Brown (Eds), *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, Dordrecht: Springer, pp. 87-109, 2006.

[10] I. Stavness, J. Gluck, L. Vilhan, S. Fels: "The MUSICtable: A Map-based Ubiquitous System for Social Interaction with a Digital Music Collection," *Procs of International Conference on Entertainment Computing (ICEC05),* pp. 291-302, 2005.

[11] D. Sprague, F. Wu, M. Tory: "Music selection using the PartyVote democratic jukebox," *Procs of AVI '08,* pp. 433-436, 2008.

[12] K. Eustice, V. Ramakrishna, N. Nam, P. Reiher: "The Smart Party: A Personalized Location-Aware Multimedia Experience," *Procs of the 5th IEEE Consumer Communications and Networking Conference (CCNC 2008),* pp.873-877, 2008.

[13] H. Mahato, D. Kern, P. Holleis, A. Schmidt: "Implicit personalization of public environments using Bluetooth," *Extended Abstracts of CHI '08*, pp. 3093-3098. 2008.

[14] Randall, D., Harper, R., Rouncefield, M.: Fieldwork and Ethnography: A perspective from CSCW, *Proceedings of EPIC 2005*, pp. 81-99, 2005.

[15] Cunningham, S.J., Jones, M.: "Autoethnography: a tool for practice and education," *Procs of the 6th New Zealand Int. Conf. on Computer-Human Interaction (CHINZ 2005)*, pp. 1-8, 2005.

[16] Glaser, B., and Strauss, A.: *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Chicago, 1967.

[17] Cunningham, S.J., Jones, M., and Jones, S.: "Organizing digital music for use: an examination of personal music collections". *Procs of the ISMIR04*, Barcelona (October 2004), pp. 447-454, 2004.

[18] Goto, M.: A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Trans on Audio, Speech, and Language Processing*, 14(5), pp. 1783-1794, 2006

MUSIC AND GEOGRAPHY: CONTENT DESCRIPTION OF MUSICAL AUDIO FROM DIFFERENT PARTS OF THE WORLD

Emilia Gómez, Martín Haro, Perfecto Herrera
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{emilia.gomez,martin.haro,perfecto.herrera}@upf.edu

ABSTRACT

This paper analyses how audio features related to different musical facets can be useful for the comparative analysis and classification of music from diverse parts of the world. The music collection under study gathers around 6,000 pieces, including traditional music from different geographical zones and countries, as well as a varied set of Western musical styles. We achieve promising results when trying to automatically distinguish music from Western and non-Western traditions. A 86.68% of accuracy is obtained using only 23 audio features, which are representative of distinct musical facets (timbre, tonality, rhythm), indicating their complementarity for music description. We also analyze the relative performance of the different facets and the capability of various descriptors to identify certain types of music. We finally present some results on the relationship between geographical location and musical features in terms of extracted descriptors. All the reported outcomes demonstrate that automatic description of audio signals together with data mining techniques provide means to characterize huge music collections from different traditions, complementing ethnomusicological manual analysis and providing a link between music and geography.

1. INTRODUCTION

Most of existing Music Information Retrieval (MIR) technologies and systems focus on mainstream popular music from the so-called "Western tradition". The term *Western* is generally employed to denote most of the cultures of European origin and most of their descendants. The unavailability of scores for most musical traditions makes necessary to work with audio recordings, and some recent works have studied if the available techniques and descriptors for audio content description are suitable when analyzing music from different traditions [12].

We provide in [3] an initial contribution in this direction, with the goal of analyzing the descriptive power of tonal features to discriminate Western vs non-Western music material. These tonal features are derived from chroma representations, computed using an interval resolution of 10 bins per semitone and representative of the employed tuning system and gamut. We found that tonal descriptors were able to distinguish these two classes with an 80% accuracy using different classifiers and an independent set for testing. The music collection was made of 1,500 pieces from different areas of the world. In a similar way, Liu et al. have recently performed a study on the classification, by means of Support Vector Machines, of a music collection of 1,300 pieces containing Western classical music, Chinese and Japanese traditional music, Indian classical music and Arabic and African folk music [7]. The best result (84.06%) was obtained using timbre features, and the results for standard chroma features was very low. This might indicate that one semitone resolution is not accurate enough to represent non-equal tempered scales and gamuts found in various cultures.

The goals of this paper can be summarized as follows: first, to analyze the contribution of the different facets of music description (timbre, rhythm, tonality) for the automatic classification of Western vs non-Western music; second, to evaluate the validity of the different features to characterize certain types of music; and third, to investigate the relationship between extracted descriptors and geographical location of the analyzed pieces (latitude and longitude). In order to do that, we have gathered a music collection covering traditional music from different geographical zones and countries as well as a varied set of Western musical styles. Up to our knowledge, the relationship between geography and extracted descriptors has not been addressed in any previous existing piece of literature, and the present study provides an attempt in this direction.

2. METHODOLOGY

2.1 Music collection

For this study, we gathered a music collection comprising 5,905 pieces from different musical traditions and styles. They were manually divided into Western and non-Western categories and labelled according to the musical genre and geographical location (area and country).

For non-Western music, we gathered a total of 3,185 audio recordings distributed by geographical region, as

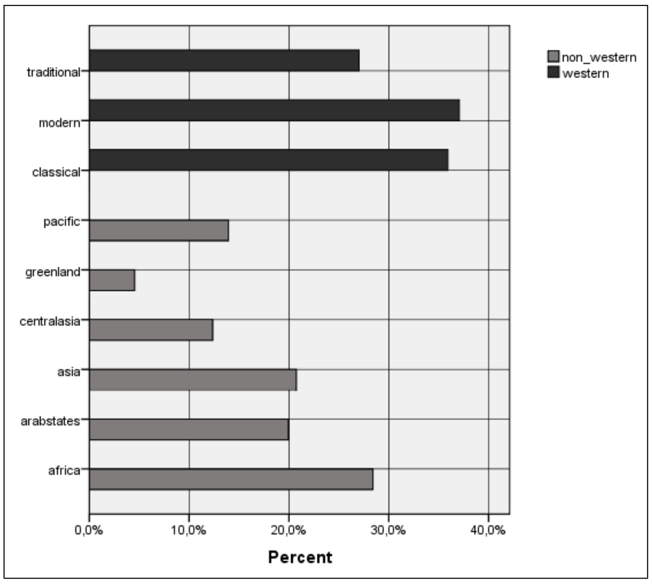


Figure 1. Distribution of the music collection.

defined by UNESCO [1] . They were distributed among the different countries and labelled according to the country of origin and geographical region. We defined the categories *Pacific, Greenland, Central Asia, Asia, Arab States* and *Africa*. These samples contain representative recordings of traditional music from different countries, discarding those having some Western influence (e.g. equal-tempered instruments). They were extracted from CD collections used for ethnomusicological studies (field recordings and compilations of traditional music).

We also considered 2,720 recordings from Western music assigned by UNESCO to the region *Europe and North America*. A set of this data was gathered from commercial CDs and is scattered across different musical genres (alternative, blues, classical, country, disco, electronica, folk, funk, hip-hop, jazz, metal, pop, reggae, rock and soul). A subset of the ”Western” collection that was chosen has been widely used within the MIR community [6, 10, 11]. We also added a collection of traditional music from Western countries (Europe and American folk). This data was labelled according to country of origin and musical genre.

Figure 1 shows the class distribution of the music collection under study. For Western music, we have distinguished between three main classes: classical, traditional music and a general class called *modern* that groups the remaining musical genres. For non-Western material, we have grouped the different countries into the mentioned categories. As it can be seen in the figure, classes are not equally distributed. One reason for that is the variability of pieces available to our analysis, which made it very hard to find the same number of excerpts for all the considered countries (e.g. we only found around 10 pieces for countries such as Vanuatu, Oman, Zimbabwe or Tanzania while the number of pieces for traditional music of European countries had to be restricted to 90 excerpts per country). On the other hand, geographical regions differ on the number of countries and musical traditions. For instance, there were few recordings from Greenland compared to the different styles present in Asia (including Indian music for instance). We will minimize the impact that this might have in the classification problem by balancing the distribution of Western vs non-Western material.

For this study we analyzed the first 30 seconds of each musical piece, and we discarded few non representative parts containing silences or ambiguous introductions (the music on these introductions was not related to the overall content of the piece).

[1] http://portal.unesco.org/geography

2.2 Feature extraction

A main goal of this study is to provide a multi-faceted description of the music collection and compare the relative performance of different musical facets (tonal, timbre and rhythm) for comparative analysis of music from around the world. In order to do that, extracted audio features are related to these different facets:

Tonality: tonal features are related to the pitch class distribution of a piece, its pitch range or tessitura and the employed scale and tuning system. The features in this group include the *tuning frequency*, which estimates the frequency used to tune a musical piece if we consider an equal-tempered scale. This feature is expected to be close to zero for pieces tuned in this temperament. High-resolution pitch class distributions are also obtained as the Harmonic Pitch Class Profile (HPCP), computed with a resolution of 10 bins per semitone and averaged for the analyzed segment. We also obtain a ”transposed” version of the HPCP that we call the THPCP, by ring shifting the HPCP vector according to the position of the maximum value. Some tonal features are then derived from them (*equal-tempered deviation, non-tempered energy ratio* and *diatonic strength*). We finally consider a dissonance measure and a descriptor called *octave centroid*, which is obtained from a multi-octave fundamental frequency representation and corresponds to the geometry centre of the played pitches. We compute this description on a frame basis and then obtain the average and variance for the considered segment. This set of features was used in a previous study [3].

Timbre: we gather here a standard set of timbre features including loudness, spectral flux, spectral flatness, roughness, MFCCs and energy computation in bark bands. These features are computed in a frame basis, and we then obtain statistical measures such as maximum and minimum value, mean and variance. Timbre features are obtained as explained in [9].

Rhythm: in terms of rhythmic features, we consider different attributes such as the estimated global tempo for the analyze excerpt as well as some features obtained from Inter-Onset Interval (IOI) histograms (peak positions and values) and onset rate (number of onsets per second). The algorithm for rhythmic feature computation is based on the system described in [2].

Drum: this group is composed by a set of song-level percussion descriptors computed from the output of a transcription system that detects drum kit events (i.e. bass drum, snare drum and hi-hat) [5]. Other instruments sounding like them are probably detected and considered as being them. These song-level descriptors include: the ratio between the number of detected events per instrument and the total number of onsets (e.g. bass drum/total), the ratio between the number of instances among instruments (e.g. bass drum/hi-hat), the number of detected events per minute (e.g. hi-hat/min) and the peak values of the histogram of the inter-instrument intervals.

2.3 Classification algorithms

We have approached several classification methods but, for the sake of summarization, we only present the results obtained for Support Vector Machines (SVM), considered as one of the best-performing learning algorithms currently available. We have employed the data mining software RapidMiner [8] [2] , which implements SVM using LibSVM [1].

We have used a grid search facility available in RapidMiner to find the following optimal values for the kernel function: linear ($u' \cdot v$), polynomial ($(\gamma \cdot u' \cdot v + coef_0)^{degree}$) and radial basis function ($e^{-\gamma \cdot |u-v|^2}$). $coef_0$ has been set to its default value ($coef_0 = 0$) and a grid search has been run to find the optimal values of kernel type, γ, $degree$ and C, which corresponds to the cost parameter that controls the trade off between allowing training errors and forcing rigid margins. A soft margin then permits some misclassifications. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well [3] . We have adopted an evaluation procedure based on 10-fold cross-validation over equally-distributed classes.

3. RESULTS

3.1 Distribution of features

In order to have a preliminary idea of the usefulness of the different features for comparative analysis, we provide here some analysis of the feature distributions. Figure 2 shows the distribution of the tuning descriptor for Western and non-Western music. As expected, the distribution of tuning deviation with respect to 440 Hz is centered on 0 cents for Western music and equal distributed between -50 and 50 cents for non-Western pieces. We also find some differences in other tonal descriptors such as equal-tempered deviation, representing the deviation from an equal-tempered scale, which also appears to be lower for Western than for non-Western music (see Figure 3).

We can also analyze the feature distribution for the different geographical areas and musical genres. One example is shown in Figure 4, where we represent the distribution of the *drum/total* descriptor, for the different geographical areas and musical styles. This descriptor represents the presence of drum sounds (or other instruments with similar sound) in the analyzed piece. As expected, we observe that the values for this feature are high for the class *modern* (including musical genres such as jazz, pop and rock) and African music (with a significant presence of percussive instruments), and are low for classical music.

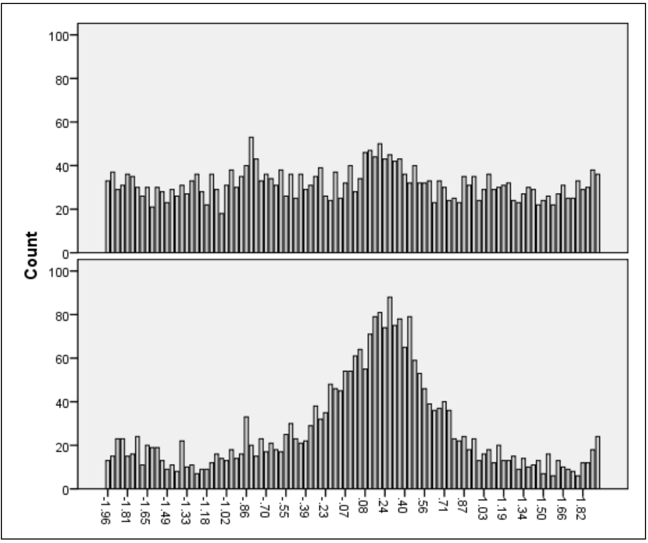


Figure 2. Distribution of tuning frequency (normalized value) for non-Western (top) and Western music (bottom).

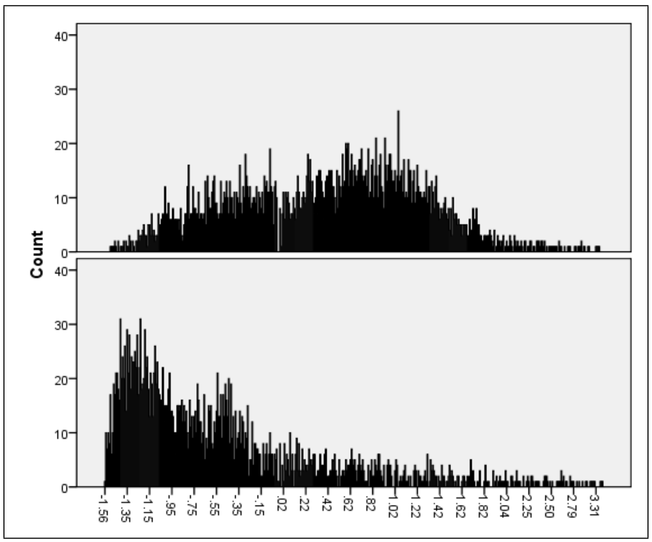


Figure 3. Distribution of equal tempered deviation (normalized value) for non-Western (top) and Western music (bottom).

3.2 Western vs non-Western classification

Our goal here is to have a classifier that automatically assigns the label ”Western” or ”non-Western” to any audio file that is input and analyzed with the mentioned features. We are aware of the limitations of the concept of Western as opposed to non-Western, as this is a first step towards the definition and formalization of stylistic features proper to different kinds of music.

For the Western vs non-Western categories, the achieved

[2] http://rapid-i.com

[3] http://www.dtreg.com/svm.htm

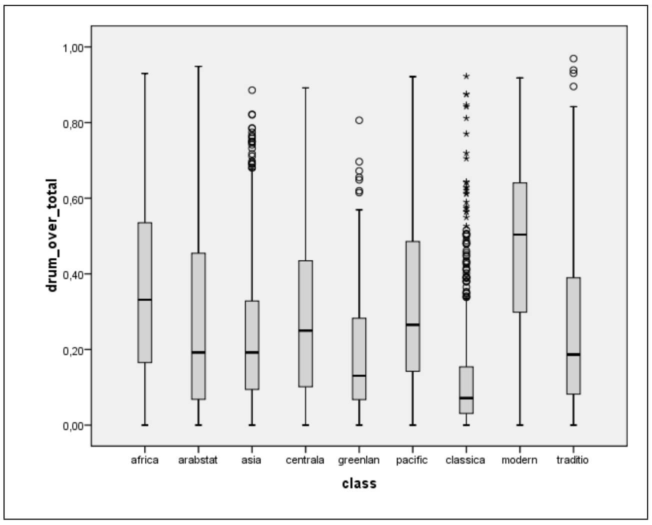


Figure 4. Distribution of the *drum/total* descriptor for the different geographical areas and musical genres.

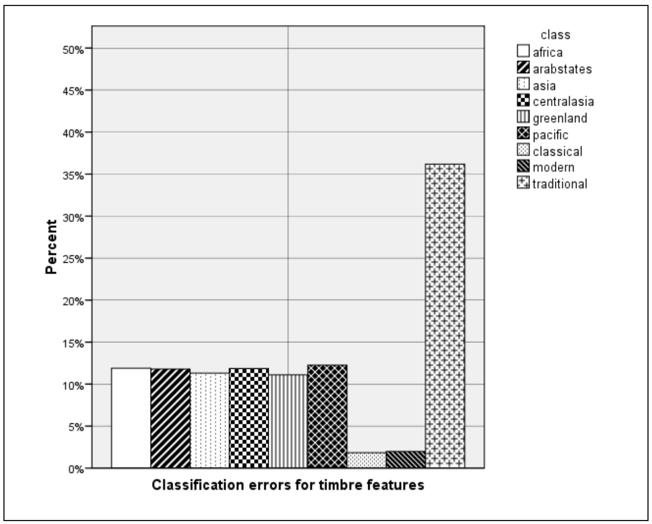


Figure 5. Classification errors (%) for timbre descriptors.

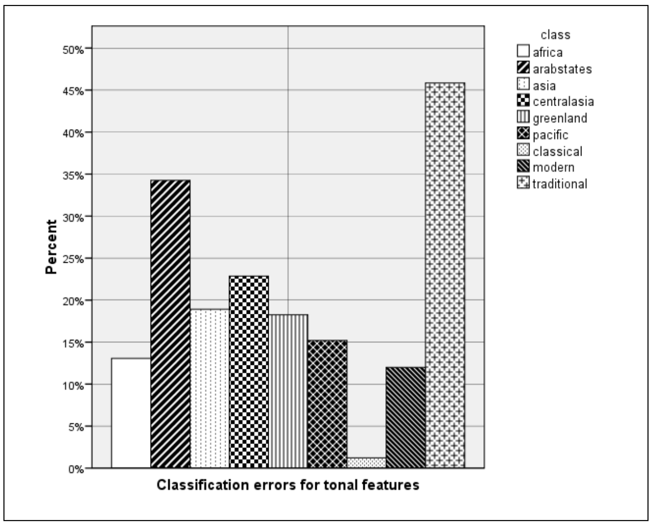


Figure 6. Classification errors (%) for tonal descriptors.

classification results for the different feature sets are summarized in Table 1. The second row indicates the accuracy of timbre features after applying an attribute evaluation method for feature selection, correlation-based feature selection (CFS) [4]. This algorithm selects a near-optimal subset of features that have minimal correlation between them, and maximal correlation with the to-be-predicted classes. This procedure was performed 10 times by means of a 10-fold cross-validation procedure, and only the timbre features that were selected more than 8 times were considered. They include descriptors based on spectral MFCCs, Bark-band energy, spectral flux and roughness.

The last row indicates the accuracy after applying the same feature selection method to the whole feature set. The feature-selection procedure was performed 10 times as well by means of a 10-fold cross-validation procedure, and only the features that were selected 10 times were considered. The selected features include a combination of tonal (tuning frequency, deviations from equal tempered scale and relative intensity of the fourth and fifth degree of a diatonic scale), timbre (features derived from MFCCs, energy in bark bands and spectral flux) and drum features (number of detected hit-hat and bass-drum per minute).

As a general conclusion, we observe that the highest classification accuracy, 88.53%, is obtained using timbre features. Nevertheless, the number of features for this set is very high (176 descriptors). Using a feature selection procedure, the set can be reduced to 25 timbre features with a 83.36% of accuracy. As the non-Western collection contains many field recordings, we think that timbre descriptors may be related to recording quality instead of musical properties of the pieces under study. In this regard, we observe that 81.23% of accuracy is obtained by using 41 tonal features, and, using the feature selection procedure described above, 23 features from the different sets yield to a global accuracy of 86.88%. This descriptor set should be considered robust to recording quality.

We can also see that the performance for rhythmic and drum features is very low, indicating that these descriptor sets are incomplete to discriminate Western from non-Western material. This was expected because Western music includes pieces without drums (e.g. classical and traditional music) and without a steady rhythm. Looking at the F-measure for Western and non-Western classes, we do not find significant differences for timbre, tonal and drum features. Nevertheless, we observe that the value of F-measure for Western music is more than 10% lower than for non-Western music when using rhythmic features. In general, the low performance of rhythmic descriptors may suggest that the implemented features only represent periodicity and tempo of music with a steady rhythm, but can be insufficient to capture more subtle rhythmical aspects of classical and traditional music.

3.3 Classification accuracy for different musical genres and traditions

Figures 5 to 8 present the percentage of classification errors per each class for the different feature sets. We observe

Set	Nb	Kernel function parameters (type, degree, cost, gamma)	Accuracy (%)	F-measure W	F-measure non-W
Timbre	176	polynomial, 3, C=8.87, gamma=0.4	88.53	0.8856	0.8850
Timbre (CFS)	25	polynomial, 3, C=8.87, gamma=0.4	83.36	0.8345	0.8327
Tonal	41	linear, C= 2.14	81.23	0.8152	0.8095
Rhythm	23	linear, C= 2.14	62.02	0.5520	0.6704
Drum	17	radial basis function, C= 0.0, gamma= 0.4	69.83	0.7036	0.6929
Selection (CFS)	23	radial basis function, C= 0.0, gamma= 0.4	86.88	0.8600	0.8765

Table 1. Accuracy using SVM classifier and a grid search procedure.

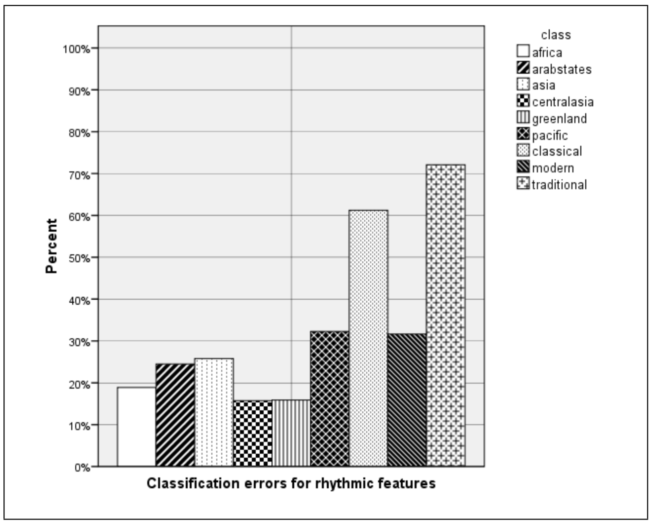


Figure 7. Classification errors (%) for rhythmic descriptors.

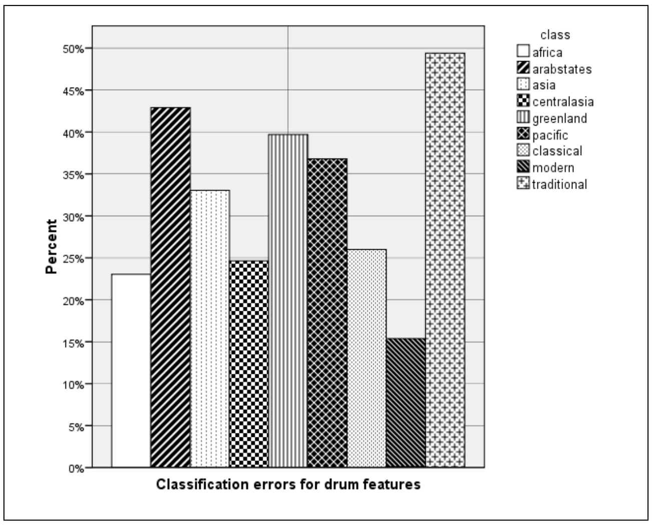


Figure 8. Classification errors (%) for drum descriptors.

that *traditional* music is the most misclassified Western class for all the feature sets. The reason for that may be that traditional pieces are closer to non-Western material with respect, for instance, to instrumentation (e.g. a small number of instruments, similar recording conditions) or tonality (e.g. high degree of ornamentation or scales with deviations from equal tuning).

On the other hand, we observe that more than 60% of the *classical* excerpts are not correctly classified when using rhythmic descriptors, and only 25% when using drum descriptor. We can think that these feature sets may consider Western music as having a constant rhythm and with a high presence of drum sounds, and that classical pieces differ from this assumption. We also observe that *arabic* music is sometimes labeled as Western when using tonal features, as found in [3].

3.4 Geographical location and feature distance

We can also analyze the correspondence between audio features and geographical location by studying the geographical distribution of feature values. In order to do that, we have computed a set of statistics over the audio features for the considered countries. Figure 9 shows an example of the geographical distribution of the equal-tempered deviation feature. We observe that low values are found in Europe, United States and Australia, while higher values are found in African and Asian music.

We have then studied the correlation of audio features averages and the average latitude and longitude coordinates for each country. In the Pearson correlation computation, the Bonferroni method was used to adjust the observed significance level for the fact that multiple comparisons were made. The following average descriptors showed a low (i.e., $0.3 < |r| < 0.45$) but significant correlation with the geographical coordinates:

Latitude: 3rd peak of the Inter-Onset Interval histogram, transposed chroma features (2nd, 3rd, 5th, 7th and 9th equal-tempered positions), chroma features (7th, 10th and 12th equal-tempered positions), equal tempered deviation and non-tempered energy ratio.

Longitude: 4th peak of the Inter-Onset Interval histogram and number of onsets per second.

From the previous list, it is worth to note that latitude is mostly associated to tonal features, while longitude is more associated to rhythmic descriptors. We can also build a regression model for latitude using only the above-mentioned 13 significant descriptors, yielding a correlation value of 0.59. Regarding longitude, the correlation is 0.31. These initial observations need to be carefully re-assessed in the context of theories that might relate geographical and climate variables to constraints on musical instrument construction or on music-centred social activities.

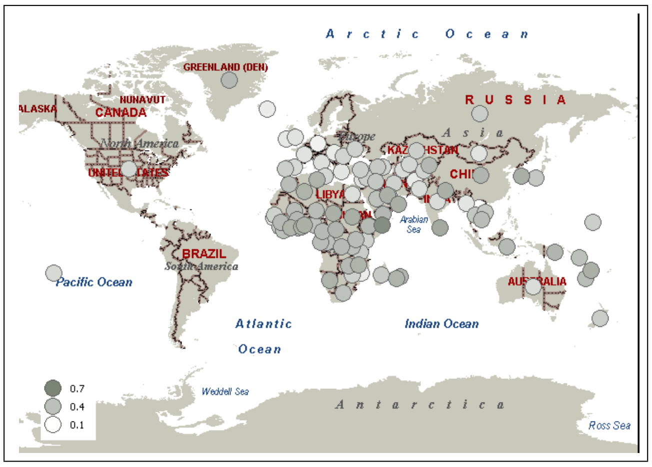


Figure 9. Geographical distribution of the equal tempered deviation descriptor (normalized value).

4. CONCLUSIONS AND FUTURE WORK

We have presented an empirical approach to the comparative analysis of music audio recordings based on tonal, timbre and rhythmic features using a music collection from various parts of the world. We tried to automatically distinguish music from Western and non-Western traditions by means of automatic audio feature extraction and classification. An accuracy of 86.68% was obtained for a music collection of around 6,000 pieces, using only 23 features from different musical facets. This confirms that timbre and rhythmic descriptors complement high-resolution tonal features for the characterization of music from various cultures. Furthermore, each feature set helped to discriminate certain types of music (e.g. drum features were suitable to identify pieces from the *modern* category).

As a future work, we would like to extent this analysis to more specific music collections and complement our current description from an ethnomusicology perspective. In this regard, we will attempt the clustering of the pieces in terms of musical culture. Ideally, we should then be able to define and formalize stylistic features proper to different traditions, and approach genres not just geographically but as a set of traits. We think that this will help to refine our descriptors and similarity measures accordingly. We also plan to complement the current collection with music from the UNESCO region *Latin America and the Caribbean* and explore influences and "frontier music" with this procedure.

As a final consideration, we conclude that existing MIR techniques are of great interest for the comparative study of all existing music traditions in the world, and audio description tools have a great potential to assist in ethnomusicological research. We hope that the present work contributes to the understanding of our musical heritage by means of computational modeling.

5. ACKNOWLEDGEMENTS

The authors are grateful to Georges Tzanetakis for sharing his genre collection. They also want to thank their colleagues at the Music Technology Group (UPF) and the anonymous reviewers for their helpful comments and suggestions. This research has been partially funded by the EU-IP project PHAROS6 (IST-2006-045035) and the e-Content Plus project VARIAZIONI7 (ECP-2005-CULT-038264).

6. REFERENCES

[1] Chih-Chung, C. and Chih-Jen, L. “LIBSVM : a library for support vector machines”, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[2] Dixon, S. ”Automatic extraction of tempo and beat from expressive performances”. *Journal of New Music Research*, 30, pp. 39-58, 2001.

[3] Gomez, E. and Herrera, P. “Comparative analysis of music recordings from Western and non-Western traditions by automatic tonal feature extraction”, *Journal of Empirical Musicology*, 3(3), pp. 140-156, 2008.

[4] Hall, M.A. “Correlation-based feature selection for discrete and numeric class machine learning”, *7th International Conference on Machine Learning*, pp. 359-366, San Francisco, CA, USA, 2000.

[5] Haro, M. and Herrera, P. “From low-level to song-level percussion descriptors of polyphonic music”, *International Conference on Music Information Retrieval*, Kobe, Japan, 2009.

[6] Holzapfel, A. and Stylianou, Y. “A statistical approach to musical genre classification using non-negative matrix factorization”, *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2, pp. 15-20, Honolulu, Hawai, USA, 2007.

[7] Liu, Y., Xiang, Q., Wang, Y. and Cai, L. “Cultural style based music classification of audio signals” *IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei Taiwan, April 2009.

[8] Mierswa, I.,Wurst, M., Klinkenberg, R., Scholz, M. and Euler, T. “YALE: Rapid prototyping for complex data mining tasks”, *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD-06), pp. 935-940, Philadelphia, USA, 2006.

[9] Peeters, G. “A large set of audio features for sound description (similarity and classification) in the cuidado project”, Technical report, CUIDADO I.S.T. project, IRCAM, 2004.

[10] Rentfrow, P. J. and Godsling, S. D. “The do re mi’s of everyday life: The structure and personality correlates of music preferences”, *Journal of Personality and Social Psychology*, 84(6), pp. 1236-1256, 2003.

[11] Tzanetakis, G., Essl, G. and Cook, P.. “Automatic musical genre classification of audio signals”, *International Symposium on Music Information Retrieval*, Bloomington, Indiana, USA, 2001.

[12] Tzanetakis, G., Kapur, A., Schloss, W. and Wright, M. “Computational ethnomusicology”, *Journal of Interdisciplinary Music Studies*, 1(2), pp.1-24, 2007.

YOU CALL *THAT* SINGING? ENSEMBLE CLASSIFICATION FOR MULTI-CULTURAL COLLECTIONS OF MUSIC RECORDINGS

Polina Proutskova
Department of Computing
Goldsmiths, London, UK
p.proutskova@gold.ac.uk

Michael Casey
Bregman Music Research Laboratory
Dartmouth College, USA
mcasey@Dartmouth.edu

ABSTRACT

The wide range of vocal styles, musical textures and recording techniques found in ethnomusicological field recordings leads us to consider the problem of automatically labeling the content to know whether a recording is a song or instrumental work. Furthermore, if it is a song, we are interested in labeling aspects of the vocal texture: e.g. solo, choral, acapella or singing with instruments. We present evidence to suggest that automatic annotation is feasible for recorded collections exhibiting a wide range of recording techniques and representing musical cultures from around the world. Our experiments used the Alan Lomax *Cantometrics* training tapes data set, to encourage future comparative evaluations. Experiments were conducted with a labeled subset consisting of several hundred tracks, annotated at the track and frame levels, as acapella singing, singing plus instruments or instruments only. We trained frame-by-frame SVM classifiers using MFCC features on positive and negative exemplars for two tasks: per-frame labeling of singing and acapella singing. In a further experiment, the frame-by-frame classifier outputs were integrated to estimate the predominant content of whole tracks. Our results show that frame-by-frame classifiers achieved 71% frame accuracy and whole track classifier integration achieved 88% accuracy. We conclude with an analysis of classifier errors suggesting avenues for developing more robust features and classifier strategies for large ethnographically diverse collections.

1. INTRODUCTION

We explore approaches for MIR and ethnomusicology to support each other in the area of cross-cultural research and to contribute new tasks and observations to both fields. Ethnomusicological recordings constitute a major challenge to MIR tools, due to their musical, acoustic and technical diversity, so they can help improve our understanding of machine-music interaction. MIR methods are also driving interest in larger-scale, data-intensive cross-cultural studies in music.

Ethnomusicological recordings document musical repertories outside of Western classical and popular music, often those that are endangered or extinct today. These recordings are used for education or research on these repertories. Some collections have been commercially released by record labels or cultural organizations [1]. Now, due to easily accessible recording equipment, the volume of recordings is growing exponentially. This poses new challenges in managing ethnomusicological collections which can hold up to hundreds of thousands recorded items with tens to thousands of hours of recordings, though only a fraction of these are currently digitized [2].

Recording quality within collections varies greatly and there is often little or no information about the technical and acoustic context for the recording. Sometimes the singer or the leading instrument are not the most dominant part of the recording; social contexts vary greatly from the concert or album settings common in Western musical culture; field recordings often contain sounds of social and natural environments as well as other noise conditions. However, the greatest challenge is the variance in musical material: even if a given collection is homogeneous, the content will differ greatly from Western music so requiring new MIR approaches.

This paper concerns automatic annotation, both at the frame level and track level, of ethnomusicological field recordings. The qualitative nature of their research requires us to approach ethnomusicologists carefully when proposing new technology. For example, a single classification error can have a far reaching impact on interpretation, so we must consider classification errors and their causes and document these for users of automatically labeled archives. To this end, we give an overview of previous work in Section 2, describe our classification experiments in Section 3, present our results and offer detailed observations on classifier errors in Section 4 and conclude in Section 5.

2. PREVIOUS WORK

Downie [3] and Tzanetakis et al. [4] have stressed the need for research on ethnomusicological collections. But publications in this area are still rare in large part due to there being few recorded collections available with annotations to use as ground truth data. Tzanetakis et al. [4]

provide an overview of MIR work related to non-Western musical content and suggests basic guidelines for this kind of study, but this work does not consider cross-cultural research, involving heterogeneous collections, which we are predominantly interested in.

Previous systems for detection of singing employ frame-by-frame classification on data consisting primarily of Western popular music [5,6,7]. These studies employ MFCC features, sometimes with derivatives and other spectral features, combined with statistical models using combinations of Neural Networks, HMMs, GMMs or SVMs to classify into two categories: frames containing singing and non-sung frames. Temporal smoothing is often applied to reduce labeled region fragmentation [6]. Our work differs in the variance of both acoustic and musical conditions of the training and testing data and in the detailed consideration of classifier errors with respect to this variation.

Wembu and Baumann [8] suggested that SVM classifiers yield slightly better results than HMMs and GMMs. Only a few studies used frame-by-frame labeling of the ground truth [9] which we consider to be essential to accurate evaluation. Two MIR studies are related to singing in non-Western cultures: singer identification in Greek Rembetiko [9] and in South Indian Carnatic music [10]. Both of these employed MFCCs. The latter team used signal separation for distinguishing between vocal and instrumental frames.

Other studies involving non-Western music recordings consider single musical cultures or repertories, each of which provides some homogeneity of the musical material. Several researchers performed rhythmic analysis and classification based on beat features for such repertories like Malay, Greek, Central African traditional music as well as Afro-Cuban music [11,12,13]. Chordia et al. [14] found a statistical measure based on pitch classes that distinguishes between different ragas. Sridhar and Geetha [10] developed Carnatic interval cepstral coefficients (CICC), based on the division of the octave in 22 Sruti, to better suit the tonality structure of the Indian Carnatic music.

Holzapfel et al. [9] compiled a database of Rembetiko singers for their artist recognition experiment. They deliberately chose recordings that are very similar in style to avoid identification due to style differences. They labelled training data frame-by-frame with one second window hop. They used aggregate "world model" GMMs to distinguish vocal from instrumental frames using intersection of the maximum-likelihood and minimum likelihood frames of opposing classifiers. This technique resulted in a classification accuracy of 99%. The data set included historical grammophone recordings, but the study was for a homogenous musical style over 21 Rembetiko singers.

Cross-cultural MIR studies include discriminating mood taxonomy of Chinese traditional music and Western classical music [15], retrieval through metric similarity for Greek and Central African music [12] and a study on metrical ambiguity in Bossa Nova, Gahu, Rumba, Soukous, Son, and Shiko [16].

We take a more general approach. To establish a model for cross-cultural research we require a sufficiency of training data to account for variance caused by difference in cultural origins, in recording techniques and in musical textures.

3. VOCAL/INSTRUMENT CLASSIFICATION

The purpose of our study is the evaluation of the baseline performance of widely-used MIR methods on an ethnographically diverse data set and to gain insight into future research potential by performing a detailed analysis of any misclassifications. We prepared a data set consisting of excerpts taken from the Lomax Cantometrics training tapes collection [17,18] which contains a high degree of cultural, technical and textural variance since the data was originally collected to find correlations between musical style and cultural traits such as social organization of a society.

3.1 Data

The Lomax data set consisted of 1000 tracks from all over the globe including recordings sung in different languages, music played on "exotic" instruments, singing, polyrhythmic as well as rhythmically free melodies, non-diatonic, non-tempered scales, a great diversity of voice timbres, rhythms, harmonies and textures from heterophonic to uncoordinated, with considerable variation in the social organization of the performing group.

For our experiments we used 355 of approximately 1000 sound samples. Of these 355 tracks 297 contain singing and 58 are purely instrumental. Of the singing tracks 185 are a'capella singing and 112 contain accompanying instruments; 110 are sung solo, 130 are choral and 57 contain both solo and group singing; 166 tracks contain primarily male singing (solo or group), 60 female singing and 51 mixed male and female singing, 10 are sung by children. More than 50 cultures are represented in the database from 5 continents as well as from large and small islands. Instruments include all kinds of idiophones (rattles, drums, frame drums, sticks, gamelans, xylophones), aerophones (flutes, clarinets, trumpets, tuba, didgeridoo), chordophones (all kinds of lutes, zithers, bow chordophones like fiddles and classical violins).

We received the audio in MP3 128kbt/sec, 44,1kHz. Files had durations of between 10 and 150 seconds. For each audio file we extracted 20-band Mel Frequency Cepstral Coefficients (MFCC). These were extracted using a short-time Fourier transform with hop size 100ms (2205 samples), window length 185.76ms (8192 samples), FFT length 16384 samples (2.69Hz frequency bins). For classification and evaluation we developed tools in Matlab using the libsvm package [19].

Audio files were annotated at the whole track level as being predominantly sung acapella (sa), instrumental (i) or singing plus instruments (si). For a subset we performed frame-by-frame labelling at 50ms increments: 111 for singing (sa + si) vs purely instrumental (i) and 77 for acapella singing (sa) vs instrumental or accompanied singing (i + si).

3.2 Frame-level and track-level classification

The first experiment was to determine the performance of frame-by-frame two-class SVM classifiers consisting of a) frames with singing, consisting of (sa) acapella and singing with instruments (si), verses instruments only (io) and b) acapella frames (sa) versus non-acapella frames; i.e. frames containing singing with instruments (si) and frames containing instruments only (io). The positive and negative training labels used for the binary SVM classifiers are summarized in Table 1.

Building on these two classifiers, we defined a third task to classify whole tracks as predominantly acapella singing (SA), instruments only (IO) or singing plus instruments (SI). The method used for the third task was whole-track integration of the frame-by-frame two-class SVM classifier outputs from the first two tasks.

Classifier	**Labels +**	**Labels -**
Sung Frame (s)	(sa) (si)	(io)
Acapella Frame (sa)	(sa)	(si) (io)

Table 1: training labels for binary classifiers used in the experiments.

For the first round of our experiment we conducted leave-one-out cross-validation on 36 songs labelled frame-by-frame for singing (s) vs. pure instrumental (si) and 30 tracks for acapella singing (sa) vs instrumental (si). We tested on whole songs so that the test set did not consist of frames drawn from any training track.

In the second round we used 111 labelled tracks for cross-validation (77 tracks for pure singing vs instrumental classifier). We applied different ways of preprocessing features to obtain better results: i.e. removing the first MFCC band, unit-norming feature vectors, detecting and removing quiet frames. We also incorporated temporal aspects of features in two ways: derivatives of the feature vectors and concatenation of sequences of three to five feature vectors. These modifications did not influence our results significantly.

A third experiment was conducted for the sung (s) vs pure instrumental (io) classifier for which we trained an SVM model on all 111 frame-by-frame labelled songs and predicted labels using this model for a test set of 244 new tracks, of which 237 contained singing and 7 were purely instrumental. We integrated the classifier output labels to construct whole song predictions for the test set using 30% sung frames as a threshold for a sung track. We compared these predictions with our manual annotation of the test songs to evaluate accuracy.

4. RESULT

The results are summarized in Tables 2-4. The first experiment yielded a mean accuracy of 74% for the sung frame classifier and 77% for the acapella frame classifier. There were a number of problem cases which had a major impact on prediction accuracy, these are discussed in the next section. When recordings from these problem groups were removed from the data set, the mean accuracy of the cross-validation on remaining 27 songs was 87.9% with 18% standard deviation for the sung frame classifier, a substantive improvement. For the next round of the experiment we paid special attention to these problem cases and included additional tracks with these characteristics into the training set.

For the third experiment, the accuracy was 83.6% for sung track classification and 62.2% of acapella tracks in the collection were correctly identified. However, 97% of tracks labeled as sung contained singing which means that the false positive rate for instrumental tracks was impacting performance. We discuss the false positives in the next section as well as what might be done to improve classifier performance.

27 tracks (problem cases excluded)	Mean accuracy	87.9%
	Std. deviation	18.0%
36 tracks	Mean accuracy	73.6%
	Std. deviation	26.0%
111 tracks	Mean accuracy	71.5%
	Std. deviation	22.4%
	Mean recall singing	83.9%
	Mean precision singing	52.6%
	Mean recall pure instrumental	76.1%
	Mean precision pure instrumental	60.5%

Table 2: singing vs. pure instrumental classifier: leave-one-out cross-validation results.

27 tracks (problem cases excluded)	Mean accuracy	85.3%
	Std. deviation	19.5%
30 tracks	Mean accuracy	77.4%
	Std. deviation	26.6%
77 tracks	Mean accuracy	71.1%
	Std. deviation	22.4%
	Mean recall singing	85.2%
	Mean precision singing	51.9%
	Mean recall pure instrumental	76.7%
	Mean precision pure instrumental	58.9%

Table 3: acapella singing vs. instrumental classifier: leave-one-out cross-validation results.

	Singing vs pure instrumental classifier	Acapella singing vs instrumental classifier
Nr training tracks	111	77
Nr test tracks	244	278
Accuracy	83.61%	62.23%
Recall positive	85.65%	42.28%
Precision positive	97.13%	76.83%
Recall negative	14.29%	85.27%
Precision negative	2.86%	56.12%

Table 4: whole track tests results.

5. PROBLEM CASES ANALYSIS

5.1 False positives / false negatives analysis

In general, the reasonable accuracy of our frame-by-frame classifiers, which is further improved in the whole track predictions, suggests this kind of classification can be achieved independently of the origin of musical material and the variance in stylistic parameters.

Similar results of the first and the second round show that we were able to include some of the variance of the data which caused problems in the first round into the training set of the second round. The following sections outline our analysis of the errors of classification and what might have caused them.

The following instrumental sounds were predicted badly during the first experiment and also negatively influenced the ability of the classifiers to predict singing when they were present in the training set:

1. woodwind instruments with a lot of "air" in the sound, like pan pipes also organs, mouth organs
2. shortly plucked or hammered instruments, like mbira, xylophone or marimba, some lutes
3. the dominant instrument playing the melody was misclassified as singing
4. tracks containing both singing and instruments were misclassified as purely instrumental
5. well blended choral singing with a wide pitch range was misclassified as instrumental
6. yodeling was misclassified as instrumental

Cases 3 and 4 could be practically eliminated in the second round of cross-validation with more training data. The performance of the classifiers with the other problem cases has improved with additional training data, but examples of these cases were still present among false positives/false negatives in the second round.

In the second experiment there were some tracks that were misclassified: mbira and mouth organ, gamelans and xylophones, flutes and clarinets (especially when played while "singing" into them) could "cheat" the classifier yielding frame-by-frame accuracies of 30% and lower in these cases. False instrumenttal positives were caused by the Russian state choir with very well blended, wide ranging vocals; the raspy, narrow, heterophonic choral singing of Marajin from Australian Arnhemland; a gospel choir with the roaring sound of Louis Armstrong and an extremely high soprano voice.

Another group of tracks with had classification results close to chance. These also included the examples from above plus:

i. choral singing of complex interlocked motivic structures; this is what Victor Grauer [20] calls pygmy/bushmen style. Also interlocked pan pipes playing, which he considers to be evolutionary related to the pygmy/bushmen style
ii. discoordinated singing in a big group
iii. fiddles, whistles, country/blues guitar and mouth har
iv. wind section of a classical orchestra

The whole track tests problem cases analysis exposed similar problem cases (mbira, xylophones, flutes, discoordinated singing, narrow, low pitched voice, complex polyphonic choral performance) but also introduced new cases which apparently were not included into the training data, such as singing with strong accents, like e.g. Native Americans from the Iroquois Confederacy; voice imitating instruments, e.g. percussion; sprechgesang (very fast spoken/sung text).

To summarize, following classes of sounds are likely to cause misclassification:

1. Instrumental:
   - All kinds of idiophones: drums, percussions, rattles; xylophones; lamellohpones like mbira; gongs like gamelans
   - Aerophones: flutes, clarinets, whistles, pan pipes (but not bagpipes), mouth harp, mouth organ
   - Fiddle and guitar, all kinds of lutes
2. Vocal (solo and homophonic):
   - Singing voice with extreme characteristics: very low or very high pitched; very narrow, nasal or raspy; voices with significant non-harmonic components in the spectrum; brilliance (strong higher frequency components) in the voice; yodeling.
   - Voice imitating instruments or singing with very strong accents
   - sprechgesang, very fast spoken/sung text
3. Polyphonic textures
   - contrapuntal, heterophonic, interlocked as well as discoordinated performances in a wide range, by an orchestra and/or a choir

Recording quality is an important factor for classification accuracy. Though the quality of audio on Cantometrics training tapes is much more varied then of any modern collection of classical or popular music, it is considerably better than many ethnomusicological datasets. We observed misclassification of recordings with extreme sound distortion, but in general the classifiers were able to cope with significant variation in recording quality.

Temporal changes presumably play an important role in distinguishing singing. As is known from speech signal processing, human speech as well as singing contains speech formants which are specific for each vocal ('a', 'e', 'i', 'o', 'u') and change from syllable to syllable. This kind of change in the formants is absent in the spectrum of practically all musical instruments.

We tried incorporating temporal changes, using MFCC derivatives, into the features but that didn't show any significant difference in the result. This suggests using a shorter hop size and window size for our features. The shorter hop size will increase the number of features so this will likely push the running time for experiments above acceptable durations.

5.2 Future Research

A MIREX-like comparison of performance on the Cantometric training tapes dataset would determine the best and cheapest approach and would uncover models implicitly relying on musical features of a specific culture like Western popular musics. Also systematic research into the feature selection for this type of classification is needed.

The next step in approaching the question of generalization with respect to cultural origin and musical style would be to test the SVM model we have trained on other musical collections. Will we be able to detect singing within collections of classical music, popular music, folk songs, non-Western recordings? If not, how much training data is missing, what kind of variance is not covered by our training set? Is the goal of having a single model to detect singing in all music achievable?

The statistical framework using SVMs of our experiments is scalable for use with tens of thousands of tracks. The training on 111 tracks takes a few minutes and prediction takes about 1 sec per song on a current high-end laptop. Prediction runs sequentially on every song, thus the testing is O(n) of the number of tracks to be predicted. With the current model we expect the approximate running time of prediction for 10 000 tracks to be less than 3 hours on our system. Assuming the generality of the model, it allows our software to be applied to much bigger collections of any musical style and origin.

It is also apparent that the same statistical infrastructure can be used to automatically classify other frame-level musical features. This will need a new round of frame-by-frame as well as whole track labeling. We plan to use this approach to classify tracks with singing into solo and choral singing, male, female and mixed singing, to detect specific style patterns like yodeling and drones. It also suggests that we should segment audio according to the prediction of more general classifiers and design hierarchical classification, for instance male/female on segments with singing; or otherwise to use this segmentation as a preliminary step for other techniques, such as pitch extraction for solo singing.

Combining these features with musical parameters obtained by other techniques (such as the amount of percussivity) or of a larger scope (such as average pitch) one would get a multi-facet description of the musical style of a track. Such a representation of a musical style applied to the Cantometrics training tapes opens up various possibilities: to study geographic distribution of a musical parameter, a combination of parameters or style patterns (e.g. choral vs. solo singing or yodel); to revise delineation of music cultures; to study the dynamics of musical style spread and influence. Since this style description is compact and can be extracted automatically from audio, it is easy to add further tracks to the data set and continually refine this research.

6. CONCLUSION

In this paper we presented our work on manual annotation of a diverse ethnomusicological collection for the purposes of testing MIR tools for automatic annotation. We conducted three experiments that served to demonstrate good performance on the data set for the task of labeling regions of different types of sung tracks versus non-sung tracks. We also presented an analysis of errors that suggests strategies for improving the overall accuracy of such classifiers.

We would like to see these methods applied to larger real world collections in ethnomusicological archives enhancing access to our cultural heritage. Practically every public ethnomusicological archive has poorly annotated holdings. Also small private archives are growing fast and could benefit from this kind of automatic annotation. Today, more than ever, technological infrastructure is needed for these types of recordings: archives are challenged to open up their collections, to make them "user-friendly", to provide not only content, but added value like expertise, easy access and fun. Having a whole collection annotated in a consistent way would allow the design of new user interfaces that are able to graphically represent style patterns and regions. Social tagging could be used to counter the errors generated in automatic annotation. Combined with archivists' expertise and moderation, this approach will enable archives to close gaps in annotation and offer hands-on activities to their user communities.

7. REFERENCES

[1] R. Reigle, "Humanistic motivations in ethnomusicological recordings," in *Recorded Music - Philosophical and Critical Reflections* (M. Dogantan-Dack, ed.), Middlesex University Press, 2009. with CD

[2] P. Proutskova, "Musical memory of the world - data infrastructure in ethnomusicological archives," *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[3] J. S. Downie, "Music information retrieval," *Annual Review of Information Science and Technology*, vol. 37, pp. 295–340, 2003.

[4] G. Tzanetakis, A. Kapur, W. A. Schloss, and M. Wright, "Computational ethnomusicology," *journal of interdisciplinary music studies*, vol. 1, no. 2, pp. 1–24, 2007.

[5] G. Tzanetakis, "Song-specific bootstrapping of singing voice structure," *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, vol. 3, pp. 2027– 2030, June 2004.

[6] A. Berenzweig, D. Ellis, and S. Lawrence, "Using voice segments to improve artist classification of music," *AES 22nd International Conference*, 2002.

[7] A. Berenzweig and D. Ellis, "Locating singing voice segments within music signals," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.

[8] S. Vembu and S. Baumann, "Separation of vocals from polyphonic audio recordings," *Proceedings of the International Symposium on Music Information Retrieval*, 2005.

[9] A. Holzapfel and Y. Stylianou, "Singer identification in rembetiko music," *Sound and Music Computing*, 2007.

[10] R. Sridhar and T. Geetha, "Music information retrieval of carnatic songs based on carnatic music singer identification," *Computer and Electrical Engineering, 2008. ICCEE 2008. International Conference on*, pp. 407 – 411, Dec 2008.

[11] S. Doraisamy, S. Golzari, N. M. Norowi, N. Sulaiman, and N. I. Udzir, "A study on feature selection and classification techniques for automatic genre classification of traditional malay music," *Pro-*

ceedings of the International Symposium on Music Information Retrieval, 2008.

[12] I. Antonopoulos, A. Pikrakis, S. T. O. Cornelis, D. Moelants, and M. Leman, "Music retrieval by rhythmic similarity applied on greek and african traditional music," *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[13] M. Wright, G. Tzanetakis, and A. Schloss, "Analyzing afro-cuban rhythm using rotation-aware clave template matching with dynamic programming," *Proceedings of the International Symposium on Music Information Retrieval*, 2008.

[14] P. Chordia and A. Rae, "Raag recognition using pitch-class and pitch-class dyad distributions," *Proceedings of the International Symposium on Music Information Retrieval*, 2007.

[15] W. Wu and L. Xie, "Discriminating mood taxonomy of chinese traditional music and western classical music with content feature sets," *Image and Signal Processing, 2008. CISP '08. Congress on*, vol. 5, pp. 148 – 152, May 2008.

[16] G. Toussaint, "A mathematical analysis of african, brazilian, and cuban clave rhythms," *Proceedings of BRIDGES: Mathematical Connections in Art, Music and Science*, pp. 157–168, 2002.

[17] A. Lomax, *Folk Song Style and Culture*. New Brunswick, New Jersey: Transaction Books, 1968.

[18] A. Lomax, *Cantometrics: An Approach To The Anthropology Of Music*. The University of California, 1976. accompanied by 7 cassettes.

[19] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines,* 2001. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

[20] V. Grauer, "Echoes of our forgotten ancestors," *The World Of Music*, vol. 2, 2006.s

Author Index