Melody Spotting Using Hidden Markov Models

Adriane Swalm Durey
Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
404-894-8361
gte401k@ece.gatech.edu

Mark A. Clements
Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
404-894-4584
clements@ece.gatech.edu

ABSTRACT

As we acquire more digitized copies of musical recordings, it becomes increasingly necessary to have the assistance of a computer in sorting through the information that it stores. In this paper, we propose a new system for melody-based retrieval of a song from a musical database which adapts wordspotting techniques from automatic speech recognition to create a melody spotting system in the musical domain. This system is tested using a variety of feature sets derived from raw audio data. It results in a successful proof of the melody spotting concept which offers great potential for development into a musical database capable of being queried by melody.

1. INTRODUCTION

As the digital age progresses, we have at our disposal an increasing amount of digitized information ranging from plain text to audio, video, and multimedia content. In order for a person to sift through this vast quantity of information, it is necessary to enlist the aid of the computer which stores it. For text, rapid matching methods have become the familiar search engines of the World Wide Web. Work on video (image) recognition and audio (speech) recognition has a long history. Only recently have we recognized the need to develop recognition systems for other types of audio, particularly music.

The problem we will address in this paper is that of musical database query by melody. In such a system, the user provides the query melody (by humming, whistling, keyboard, etc.) and the musical database system returns the most likely matches to it. Thus, we are interested in recognizing the melodic content of a piece of music, that part which is most recognizable to the casual listener, and, therefore, most easy for him to recall.

There is a growing body of research addressing the problem of music information retrieval (MIR) on the melody level. In the following section, we will address some of the methods used to approach this process in previous research. Then in Section 1.2, we will explore one of the new directions in which this research directs us, adapting the use of hidden Markov models from speech recognition to their use for melody recognition. In Section 2 we will look at the details of the system we propose. Section 3 will discuss the results of preliminary testing of this system. Future directions for research and conclusions will be presented in the final section.

1.1 Melody-Level MIR Research

The seminal works in melodic MIR, those by Ghias, et al. [8] and McNab, et al. [11], form the basis for much of the melody-level MIR research which followed them. Following their format, which is generalized in Figure 1, a database of musical recordings is developed by hand in MIDI (or some similar discrete format) and stored in monophonic tracks. These tracks are then further abstracted into melodic and/or rhythmic contours whose content varies from general (up, down, or the same) to exact (plus or minus a specific number of semi-tones.) When the user queries the system, he sings or hums the desired melody. The system then extracts the pitches (and rhythm) in that query and encodes them as the database has been encoded. The database contents and the query are then compared, generally using some variation of dynamic programming (DP), especially that developed by Mongeau and Sankoff [12]. This process produces a ranked list of the songs in the database which are most likely to contain the melody. While this does not apply to all melodic MIR systems which have been proposed, many of them adopt at least some portion of this framework.

There are a number of problems identifiable in this general structure. The first of these is the use of highly structured audio formats, such as MIDI, to represent the database contents. An accurate automatic process for generating MIDI from polyphonic raw audio has yet to be developed; this means that MIDI encoding must often be done by hand and ignores a vast wealth of data already digitized in less structured formats (like MPEG Layer 3). This format also requires that the contents of the database be firmly (even if erroneously) defined before processing begins. Likewise, if the database utilizes only melodies in processing, they too must be pre-extracted by an expert for greatest accuracy

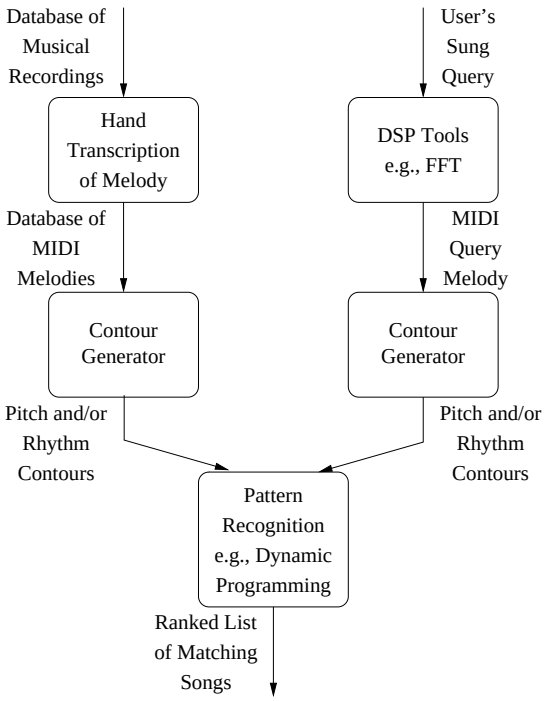

Figure 1: Generalized architecture used for melody-based music information retrieval

(though reasonably good melody-from-MIDI extractors are being developed [17]) and a hard decision made about their content. Polyphonic audio is often not addressed in melody level MIR systems, though sometimes it is represented as several monophonic melodies each representing one voice of the larger polyphonic piece. The encoding of melodies in pitch and rhythm contours is a good way to resolve errors in the database and made by the user. However, we must be careful not to discard too much query information when we can have a reasonable level of confidence in it.

Though other methods applied to melodic MIR have included techniques such as distance measure/template matching [7], n-gram matching [16], tree indexing [3], and text retrieval [4] [6], the primary methodology of most melodic MIR systems is as described above. Such exclusive use of dynamic programming for melodic MIR ignores other comparison techniques which have been successfully applied to other forms of audio signal understanding, particularly those tools used for speech recognition. In the next section, we will explore the new research directions suggested by the problems described above and a particular tool used for automatic speech recognition, the hidden Markov model as it is applied to keyword spotting.

1.2 Hidden Markov Model Research

The analysis of previous work on melody-based MIR suggests a number of potential next steps. The goal of the average user of a MIR system such as we propose is to locate a recording of a piece of music which is much more easily found in an unstructured format (for example, on a CD or in the ubiquitous `mp3` format.) Thus, we would like our system to operate on raw audio formats, such as `wav`, `aiff`, and `mp3`. To avoid the additional problem of making a hard decision about content when generating a structured encoding from raw audio, we would prefer a system that maintains a number of guesses about the content of a recording and qualifies each guess with a likelihood of correctness. This is analogous to the treatment of data in speech recognition databases.

Breaking away from discretized data formats also suggests breaking away to explore new forms of data comparators. Many of the ideas experimented with thus far in MIR research share a common background with speech recognition processing. Tools such as the fast Fourier transform [13] and dynamic programming (DP) [11] have been used for melodic matching with varying levels of success. Other tools such as mel-frequency cepstral coefficients (MFCCs) [7] and hidden Markov models (HMMs) [21] have been used for high-level content matching, classifying audio by type, but are less frequently applied to melody recognition.

There are few references in the literature to the use of HMMs for analysis of musical audio. Logan and Chu [10] used HMMs to analyze the structure of rock and pop songs which had been represented as MFCCs. This structural analysis was then used as a basis for extracting "key phrases" (those of interest structurally, e.g., a chorus) from the audio. This system could be used to supply musical thumbnails of larger recordings to a database access system, but the authors did not specifically address incorporation of their software into such a database. Batlle and Cano [1] likewise used musical audio converted to MFCCs and energy to train competitive HMMs to perform a blind (unsupervised) segmentation task.

HMMs have not yet been applied specifically to melodic content recognition. The application of HMMs closest to MIR appears in Raphael [15] where HMMs are used as a precursor to automatic accompaniment by performing segmentation of raw audio when the score is known. HMMs are constructed based on the pitches and duration noted in a score using features based on amplitude, activity, and frequency content of the signal. After training, these HMMs are used to align audio data with the score that represents it. The problem which we will address allows data to be less well defined in advance and applies a less constrained approach to analysis of the contents of a recording.

HMMs and related techniques have also been applied to the pitch extraction task. Goto [9] used estimation maximization (EM) techniques common to the training of continuous distribution HMMs to evaluate the contributions of various tone models to polyphonic audio sampled from a CD. He then used these weights to make a determination of which frequencies were the most probable melody and bass lines with 88% accuracy and 80% accuracy respectively. Such a system could eventually provide a valuable front end to a musical database system.

Many of the tasks performed in speech processing have direct analogues in music processing. Automatic speech recognition (ASR) can be likened to transcription from raw audio to common music notation. Keyword wordspotting (WS)

can be likened to spotting a query melody occurring in raw audio. The successful use of HMMs in automatic speech recognition and wordspotting applications suggests that, like DP, such tools might make a successful transition to melody recognition. These correlations drive the system developed in this paper.

In speech processing, wordspotting is the task of searching for keywords occurring somewhere in a larger body of less constrained audio data (noises, non-keywords, etc.) [18] In music, we would like to search for specific melodies occurring somewhere in a larger body (the database) of less constrained audio data (other music, harmony, noises, speech, etc.) Some of the most successful methods for wordspotting have involved HMMs: one set trained to recognize a key word or words and one set trained to account for the "garbage" making up the rest of the audio. We propose to define a set of HMMs to recognize a query melody and a set of HMMs trained to account for the auditory "garbage" not associated with that query. Thus, we hope to perform the musical equivalent of wordspotting. The next section will develop the specific configurations of HMMs used in our implementation of melody spotting.

2. EXPERIMENTAL SET-UP

The following two sections will describe the composition of the melody spotting system we test in this paper. First, the methods of data collection used in our experiments are described. We elaborate on the construction of our HMM melody spotting system in Section 2.2.

2.1 Database and Training Data

The data for this series of experiments consists of a collection of simple monophonic melodies. This allows us to begin with a relatively easy task, since the melody is the only musical data in each recording. The melodies were transposed such that no accidentals occurred in play to ensure adequate coverage of each note. All the notes fell into the range from C_4 to G_5. Each melody was played ten times on a keyboard by an amateur musician. The melodies were played each time in the same key and register (the same exact notes) but using five different instrumental voices (two recordings per voice.) The different recordings were produced for each melody to allow for natural variations in play. The changes in instruments and duration add some difficulty to the task; no two renditions of a melody are exactly alike. As the acoustic data was collected, so was the corresponding MIDI data. The use of each type of data will be illustrated in the next section. The selection of tunes recorded is given in Table 1 and the instrument voices in Table 2. (A transcription of the songs as recorded is provided in Appendix A.)

The data were collected from a Yamaha W7 keyboard at a sampling rate of 22050 Hertz using mono audio input and saved in `wav` format. This introduced additional complexity to the task due to the varying levels of noise (sometimes negligible, sometimes nearly overpowering) in the data set generated by the recording process. This resulted in approximately half an hour of recorded data requiring about 100 Mbytes of computer storage.

Table 1: List of Songs Used in Testing

	Song Title
1	*Auld Lang Syne*
2	*Barbara Allen*
3	*Frere Jacques*
4	*Happy Birthday to You*
5	*I'm a Little Teapot*
6	*Mary Had a Little Lamb*
7	*Scarborough Fair*
8	*This Old Man*
9	*Three Blind Mice*
10	*Twinkle, Twinkle, Little Star*

Table 2: List of Instrument Voices Used in Testing

Instrument Name
Clarinet
Flute
Piano
Soprano Saxophone
Violin

2.2 HMM Melody Recognition System

The construction of our melody spotting system is very similar to that of a wordspotting system for speech recognition. We will describe the system from the lowest levels, where the collected data intersect it, to the highest, where we see the melody spotting results, that is, the occurrences in the database of the desired query.

A hidden Markov model (HMM) describes a doubly stochastic process in which only the current state affects the choice of the next state. [14] HMMs model two levels of activity, a visible layer represented by the observation data it produces, and an underlying hidden layer representing the states through which the model passes. The transitions between states and the observations are governed by probabilities learned from the training data using a collection of well-known techniques. Our implementation is based on the HMM Tool Kit (HTK). [20] This system was selected because it has been used successfully in automatic speech recognition research (as in Woodland, et al. [19],) but allows the user enough design freedom to adapt it to a musical task such as our own.

The HMM is the basic building block for our melody spotter. A simple, left-to-right, five state HMM is trained to represent each note for which data is available (C_4–G_5), plus a rest (silence/noise). Such an HMM is shown in Figure 2. The first and final states are non-emitting, serving only to designate the beginning and end of the HMM. Probabilities a_{ij} govern the transitions from state i to state j. Probability distributions $b_i(O_t)$ represent the probability of seeing a given observation O at time t while in state i. The a_{ij} and $b_i(O_t)$ are the parameters that we must train using the available data. In our implementation, the probability density functions $b_i(O_t)$ for each state are modeled as Gaussian distributions with a non-uniform diagonal variance.

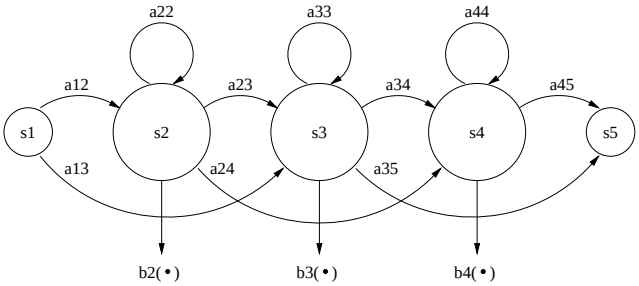

Figure 2: Hidden Markov model representing a basic musical unit: a note (for example, C_4) or rest

The raw audio data described in the previous section are stored for database purposes, but are also processed for use in training and evaluating the HMMs. For comparison purposes, we extract two different sets of features. These features were chosen as simple starting points and will be tested against a selection of other representations in later versions of this system. In both cases, each raw audio file is normalized. The first features are produced using a fast Fourier transform (FFT) with a Hamming window length of 2048 and a skip of 1024 (each window covers approximately 100 msec = .1 sec). The full results of the FFT are then trimmed (band-limited) to use only those bins corresponding to a reasonable range of frequencies for music and for the task at hand (from C_4 to C_6, or approximately 261–1044 Hz.) For the second set of features, a single pitch estimate is selected using an autocorrelation method. [5] The result in both cases is a series of observation vectors ready to be processed by the HMMs, one of length 75, the other of length one. The corresponding MIDI data is aligned with the recording by matching the onset of the first note in the `wav` recording with the first note in the MIDI file. It is then transformed into a label file in the style used by HTK, which will play a role in training and testing of the HMM melody spotting system. Figure 3 outlines this process. In order to save processing time, this preprocessing can be performed once, and the resulting data stored for later use.

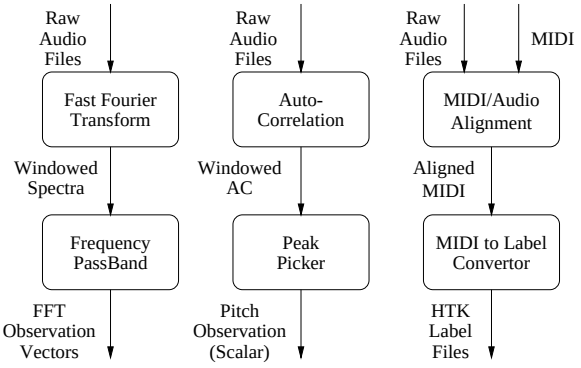

Figure 3: Data processing from raw audio to HMM observation vectors and from MIDI to HTK label data

With the HMM prototype and the preprocessed data, training can now proceed. In this preliminary system, because of the small size of the melodic database, both training and testing utilized all of the available data. We hope that future work will show that it is possible to adequately model the contents of the database without requiring the system to be trained using all of the music contained by the database. First, the label data is used to segment the input data by note as shown in Figure 4. Each segment of data is used to initialize the HMM representing that note using Viterbi alignment. The same feature segments are then used to refine each HMM using Baum-Welch reestimation. Finally, concatenations of the note level HMMs appropriate to each training melody are created and embedded Baum-Welch reestimation is performed on each one using the complete sequence of feature vectors for that recording. [20] (Such a concatenation is illustrated at the top of Figure 5.) Once this training is complete, the system is ready to accept a query.

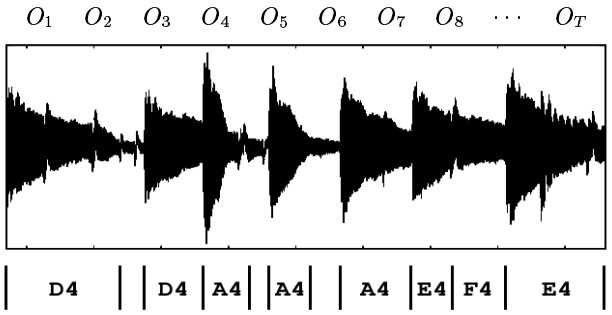

Figure 4: A series of feature vectors, O_t, extracted from raw audio and segmented using the MIDI label data. This is the first eight notes of the song *Scarborough Fair.*

When a sequence of notes defining a query is provided by the user, we are ready to create the remainder of our recognition system. For this beginning system, the query is simply an exact sequence of notes entered as text by the user, e.g, "D_4, D_4, ... , E_4." First, copies of the note-level HMMs making up the query are concatenated together to form an HMM representing the query as a whole. (See Figure 5.) Because of the current selection of models and features, this will seek an exact match to the pitches in the query. Then, all of the available notes and a rest/noise HMM are placed in parallel with the desired melody to act as fillers or garbage collectors. [18] To prevent the fillers from overwhelming the more complex (and therefore less probable) melody HMM, entry into those states is penalized, especially for those notes occurring in the melody. Looping from the final to initial states allows the HMM system, now a musical language model, to process a much larger piece of data, a complete song stored in the database.

The features representing each song in the database are then run through the constructed HMM using Viterbi scoring; the song is sectioned and each section is labelled as to its probability of belonging to the melody or to one of the fillers. The locations labelled as the melody are then sorted according to their probability of occurrence, producing a ranked list of most likely occurrences of the melody in the database songs. In testing, these results can be compared to the MIDI reference data to determine their accuracy. (Once initialization is complete, the MIDI data is not used except for judging the accuracy of the results returned by the HMM system

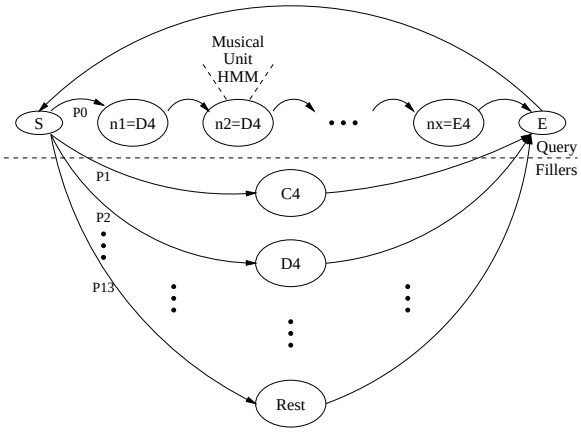

Figure 5: Construction of a musical language model representing a melodic query from the musical unit HMMs shown in Figure 2. The musical unit HMMs are used both to model the query melody and the filler composing the rest of a database entry.

in testing.) If it is an actual query, the results can be presented to the user for audition. The sectioning and scoring also allows us to audition the piece where the melody likely occurs, rather than from the beginning.

It is important to note that the MIDI data is not a requirement for this system, but rather a convenience. Initialization could be accomplished without the exact MIDI data using only a note list (no onset and offset data) in a bootstrap method. [20] It should also be possible to initialize the system with a subset of MIDI-labelled recordings sufficient to train the musical models in the system. The eventual goal would be to add to the database other musical pieces for which such label data is unavailable. Further testing would be needed to determine the efficacy of these plans compared to our current training process.

3. EXPERIMENTAL RESULTS

Melody recognizers based on both sets of observation vectors were trained using the techniques described in the previous section. These systems were then evaluated by searching for a selection of melodies of various lengths, from three to ten notes each. As mentioned previously, the queries are posed as a string of textual note names, e.g. "G_4, F_4, E_4, D_4, C_4." For these initial tests of the system, no errors were introduced into the queries. The test queries were selected based on their frequency of occurrence in the database songs so that several copies would exist to be searched for by the system. The selected query melodies and the songs in which they occur are given in Table 3. For some lengths, multiple queries were used to allow the search to operate over several different songs. Though this is an admittedly small selection of queries from a small collection of melodies, further testing on an expanded database is currently underway.

Once each query is selected, a detection network similar to that in Figure 5 is constructed to represent it. Initial penal-

Table 3: List of Query Melodies and Their Locations

Query	*Note String*	*Found in Songs*
3a.	E_4 D_4 C_4	2, 5, 7, 9
3b.	E_5 D_5 C_5	3, 8, 10
4.	F_4 E_4 D_4 C_4	2, 5, 7, 9
5.	G_4 F_4 E_4 D_4 C_4	2, 7, 9
6.	D_5 C_5 C_5 B_4 B_4 A_4	10
7a.	C_5 C_5 B_4 A_4 B_4 C_5 G_4	9
7b.	G_4 D_5 D_5 E_5 E_5 D_5 C_5	10
8a.	C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4	9
8b.	C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5	5
8c.	E_5 E_5 D_5 C_5 C_5 B_4 B_4 A_4	10
9a.	C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4 G_4	9
9b.	C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5 G_4	5
9c.	G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5	10
10a.	G_4 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4 G_4	9
10b.	G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5 B_4	10

ties for the melody and filler arcs are assigned as:

$$P_n = \begin{cases} 2/M, & \text{arc into melody block} \\ 1/M, & \text{arc into filler note not in melody} \\ 1/1000, & \text{arc into filler note in melody} \end{cases}$$

where M is the number the filler blocks and melody blocks in that network (for example, $M = 14$ for C_4–G_5 plus a rest and a query melody block.) The selection of the penalties for fillers occurring in the melody will be addressed again in the following sections. Once this is done, each song in the database is scored using the query network, resulting in a segmentation of each song into melody and fillers, each with a start and stop time and an associated probability score. Now, the segments labelled as melodies can be ranked by score and provided to a database user as results. Since the data is preprocessed, the scoring requires less than real-time to compute. Real-time here is defined by 100 songs times an average of 20 seconds a song or ≈ 2000 sec $= 33.33$ min. On a 650 MHz Pentium III with 256 Kb of RAM, receiving results from the scoring algorithm requires only about a minute, a speed up of approximately 3300%.

For testing, the results of the scoring are compared to a label file which marks the query locations in the database (the second use of the MIDI reference data.) This test ensures that those database elements labelled as containing the query do so and in the locations given by the recognizer. Two results are provided by this comparison. The first is a numerical figure-of-merit (FOM) commonly used in evaluating speech-based wordspotters. The FOM places an upper-bound estimate on word spotting accuracy (the percentage of true hits) averaged over 1 to 10 false alarms per hour. [20] (The FOM as defined by HTK is further described in Appendix B.) In the case of a melody spotter, however, it is possible for the false alarms to be interpreted differently. They must be examined to determine if they sound similar to the query—an inexact, but still valid, match.

The following two sections address the systems built using each of the observation vectors defined in Section 2. Sec-

tion 3.3 describes a third system tested on a musically based subset of the FFT data. Discussion of results common to all of these systems is presented in Section 3.4. Problems and future work suggested by this testing are addressed in the final section of the paper.

3.1 Pitch-Based Recognizer

Our first melody recognition system is constructed on a length one observation vector, the fundamental frequency of the audio signal as estimated using autocorrelation (AC). [5] This estimator was selected because, of those tested, it provided the best pitch estimates using our data set. The preprocessing time necessary to calculate these features for all of the music in the database is approximately 40 minutes on a 650 MHz Pentium III when computed using Matlab.

For each query melody, we began testing the recognition with a penalty of 1/1000 on fillers occurring in the melody, then tested several lower penalties as well.[1] Generally, new penalties were tried as long as they continued to reduce false alarms without causing any decrease in the number of accurate hits. The penalties were never less than 1/25, a limit slightly greater than the penalties (1/14) applied to the other fillers. The hits, false alarms, actual occurrences, and figures of merit achieved for each query and the penalties applied to the fillers for that value are given in Table 4.

Table 4: Query Results Using Pitch-Based HMM Recognizer

Query	*# Hits*	*# FAs*	*# Actual*	*FOM*	P_n
3a.	70	2	70	95.63	1/500
3b.	19	69	50	0.00	
4.	40	0	40	100.00	1/500
5.	30	10	30	78.80	1/500
6.	40	32	40	0.00	1/50
7a.	30	19	30	42.80	1/250
7b.	8	5	20	28.00	
8a.	30	10	30	78.60	1/500
8b.	20	0	20	100.00	
8c.	18	24	20	27.50	
9a.	30	7	30	93.40	1/500
9b.	20	0	20	100.00	
9c.	8	7	20	23.20	
10a.	20	30	20	23.80	1/500
10b.	8	8	20	20.80	

Overall, the performance of the pitch-based recognizer was poorer than that of the FFT-based recognizer, especially for queries in the range of C_5–G_5. It was observed that the pitch estimator was less accurate for these frequencies than for those in the lower register, so it was expected that the recognizer trained using those pitch estimates would be as well. It was also more difficult to reduce false alarms using the penalties without lowering the hits as well. As might be expected, the quality of the false alarms was not good in all cases, especially when large numbers of them were returned by the system.

[1] Penalty will henceforth refer to the penalty on fillers occurring within the query melody. For our purposes, a lower penalty implies a fraction with a smaller denominator; hence, lowering a penalty increases the probability that a filler will be found in the data set.

3.2 FFT-Based Recognizer

The second melody recognition system is based on the length 75, band-limited, fast Fourier transform (FFT) of the audio data. The preprocessing time required to compute in Matlab the FFTs of all the music in the data set is approximately five minutes.

Again, each query melody was tested under a variety of penalties ranging from 1/1000 to 1/25. The results of testing this recognizer are summarized in Table 5. Generally, the FFT-based recognizer was able to operate reasonably well under the same penalty, 1/1000, locating all but one of the actual occurrences, and, generally, only false alarms sounding similar to the query. This recognizer did not seem to share the upper register difficulties of the pitch-based mechanism, most probably because no hard decision about the content of the data (i.e., the pitch of the melody) was required before processing the observations. In this system, we allowed the HMM to learn statistical representations of the content of the signal, rather than defining the melodic content (the pitch) before applying statistical tools to it. In most cases, it was possible to achieve as much reduction in false alarms as desired without sacrificing successful hits.

Table 5: Query Results Using FFT-Based HMM Recognizer

Query	*# Hits*	*# FAs*	*# Actual*	*FOM*	P_n
3a.	70	1	70	100.00	1/1000
3b.	49	7	50	83.24	
4.	40	3	40	96.10	1/1000
5.	30	0	30	100.00	1/50
6.	40	10	40	99.25	1/1000
7a.	30	0	30	100.00	1/1000
7b.	20	13	20	100.00	
8a.	30	0	30	100.00	1/1000
8b.	20	0	20	100.00	
8c.	20	20	20	100.00	
9a.	30	0	30	100.00	1/1000
9b.	20	0	20	100.00	
9c.	20	10	20	100.00	
10a.	20	10	20	79.60	1/1000
10b.	20	12	20	100.00	

3.3 Scale-Based Recognizer

A third system based on a subset of the FFT observation vector data was implemented and tested as well. First, those FFT bins containing the fundamental frequencies and harmonics of the notes in the range C_4–G_5 (of the equally tempered scale) were identified. They were then used to select a subset of 25 of the 75 bins composing the previously calculated FFT feature vector. The system was designed to offer a compromise between forcing a hard pitch decision before processing (as in the AC pitch feature) and extracting a feature set using no specifically musical knowledge (as in the FFT feature vector). The preprocessing time for this feature set is negligible, since it simply masks the pre-computed FFT data.

The query data are tested as before and the results are summarized in Table 6. These results suggest that the scale-based recognizer strikes a good balance between discarding as much unnecessary information as possible while retaining as much information as it can that is relevant to this musical task. It successfully finds all of the occurrences of each query, generally without producing false alarms which sound too dissimilar from the query. It seems to operate as well as or better than the FFT-based recognizer while reducing the size of the feature vectors by a factor of three (and thus the storage requirements, the number of free parameters to train, and the overall processing time.)

Table 6: Query Results Using Scale-Based HMM Recognizer

Query	*# Hits*	*# FAs*	*# Actual*	*FOM*	P_n
3a.	70	0	70	100.00	1/100
3b.	50	0	50	100.00	
4.	40	0	40	100.00	1/25
5.	30	0	30	100.00	1/25
6.	40	10	40	100.00	1/100
7a.	30	3	30	82.00	1/100
7b.	20	11	20	100.00	
8a.	30	0	30	100.00	1/100
8b.	20	0	20	100.00	
8c.	20	11	20	100.00	
9a.	30	0	30	100.00	1/100
9b.	20	0	20	100.00	
9c.	20	10	20	100.00	
10a.	20	10	20	76.90	1/100
10b.	20	10	20	100.00	

3.4 Common Results

There are several common threads running through the results presented above. The first of these is that, in general, the greater the penalties, the more false alarms and hits allowed by the system, and the lower the penalties, the more false alarms are reduced (though sometimes at the expense of the hits.) In the case of the shorter queries, it is more reasonable to reduce the penalties because they tend to produce a large number of accurate hits. However, as queries become longer, it becomes more desirable to allow more non-exact ("inaccurate") matches to be presented as results, so increasing the penalties to allow more false alarms makes sense. It is worth noting that under many different levels of penalties, nearly all of the desired results were accurately returned by the system. This implies that penalties could be used to dynamically control the level of inexact responses provided by the system.

In all test cases, there is also something the FOM does not tell us: the quality of the segments it labels as false alarms. In the case of wordspotting, distinguishing close but wildly different words like "goat" from "goatee" makes sense, but in music, such a close match often sounds similar enough to count as a good match. One example of this phenomenon from the test data is shown in Figure 6. Generally, those inexact results found by the system resulted from consolidation of repeated notes in the query into one longer note in the response or from deletion of a note from the query. Thus the FOM is both useful and deceptive, since many of the false alarms would be worth presenting to a database user as possible matches to a melodic query.

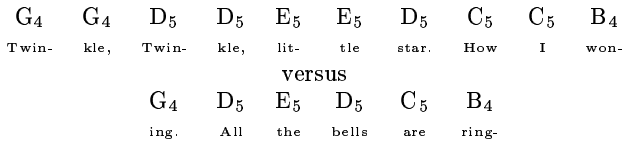

Figure 6: A query melody from *Twinkle, Twinkle, Little Star* and a shorter inexact, but similar sounding, match from *Frere Jacques*

4. CONCLUSIONS AND FUTURE WORK

It is apparent from this work that the melody recognition systems described here may stand only as a proof-of-concept. While we feel it is a successful first foray, there is a great deal of future work that should be done to improve them. This work can be grouped into three areas: that relating to the database and its contents, to the HMM recognizers, and to testing the system. All of these suggestions share the eventual aim of providing a raw audio-based musical database which is friendly to novice and expert user alike.

There are a number of improvements that should be made to the database and the data which it contains. We would like to test this system on a collection of recordings from acoustic instruments, rather than from synthesized ones, since the goal of the system is to operate on such data. Likewise, we would like to test the system with polyphonic data as well as monophonic, thus representing the vast majority of recorded music. We also want to increase the size of any database on which we operate to be more realistic, preferably while retaining an implementation that does not require hand-marking the complete contents of the database. Finally, we would like to improve the interfacing of the database for the benefit of both the researcher and the eventual end-user.

There are many aspects of research on HMMs for automatic speech recognition and wordspotting that we would like to explore in conjunction with our melody recognition system. It would be valuable to experiment with training that does not require full labelling of the data for all database entries and with training on only a subset of the data. We would like to handle different abstractions of query and database data (as other systems have done using melodic contours) based on user confidence in their query. This will likely require us to research a different selection of feature vectors than those presented here. We will also want to examine how different feature vectors affect the quality of our results; for example, a better pitch extractor (as in Goto [9]) might allow us to use smaller observation vectors without sacrificing accuracy or a constant-Q transform [2] might allow us to use only that frequency information pertaining to the musical scale. Such changes might allow us to maintain or increase accuracy while reducing the number of free parameters estimated by the system and the overall processing time required. Feature selection will also impact the types of input accepted by the system; a different selection of features might make it easier for the system to accept sung queries, for example.

Techniques like incorporating bi- and tri-grams, as is done in large vocabulary continuous speech recognizers, might also improve accuracy, as might exploring filler models other than actual notes. We should also further explore penalty selection, hopefully finding a consistent method by which to apply them. We need to look at ways we can ensure that the system accounts for allowable melodic differences (errors users might make in a query) like insertions, substitutions, deletions, etc. We must also explore ways to incorporate duration information in the query (perhaps similarly to [15].) Currently, this information is normalized out of the note level HMMs during training. We need also to address questions of complexity and scalability, perhaps exploring the common ground between our system and those developed for large scale speech recognition tasks. Lastly, we need to collect training data over a broader range of notes so that note level HMMs can be trained for them as well; we could also explore ways of filling in the notes for which adequate data is missing, perhaps with parameter tying. Techniques like parameter tying might also allow a valuable reduction of the number of free parameters to be trained.

Finally, the system needs vastly expanded testing, since the results of such testing would be invaluable in the development of later generations of this melody recognition software. It needs to be tested on a larger collection of exact melodies, as well as on melodies which encompass query errors commonly made by a user. We need to develop a way to convert a user query in an audio format (sung, hummed, whistled, etc.) into a form appropriate to querying the underlying HMM system. (Currently, the most viable method of doing so would be to perform pitch detection on the input, then use the detected pitches in place of our textual query string.) It would also be desirable to compare this system to other comparable systems. However, there are no truly comparable systems in the literature at this time (i.e., those which perform melody recognition on database entries that have not been discretely encoded) nor does there exist a standard body of music on which to perform such a comparison.

In this paper, we have presented a prototype system for spotting a query melody in a larger body of raw musical audio data by adapting HMM wordspotting techniques from the speech recognition domain to the musical domain. While the system as presented here represents a proof-of-concept for this work, it offers great potential for future development. It also offers by-products to other music related applications, such as transcription from raw audio to musical notation and automatic accompaniment. We plan to continue developing this system into a more fully capable melody-based musical database query system, following the suggestions set forth above. To address this desire, the authors are currently working on testing an expanded melody-only and polyphonic database using a collection of feature vectors and incorporating new elements, such as errors in query formulation, into the testing.

Acknowledgments

The authors would like to acknowledge Lincoln Durey for his helpful suggestions, Emperor Linux for generous computer donations, the Interactive Media Technology Center for loan of the Yamaha keyboard, and the Clare Booth Luce Fellowship program for their financial support of this project.

5. REFERENCES

[1] E. Batlle and P. Cano. Automatic segmentation for music classification using competitive hidden Markov models. In *Proc. of ISMIR*, Plymouth, MA, Oct. 2000. Available on the Internet: http://ciir.cs.umass.edu/music2000/.

[2] J. C. Brown. Calculation of a constant Q spectral transform. *J. of the Acoustical Society of America*, 89(1):425–434, Jan. 1991.

[3] J. C. C. Chen and A. L. P. Chen. Query by rhythm: An approach for song retrieval in music databases. In *Proc. of 8th International Workshop on Research Issues in Data Engineering*, pages 139–146, Orlando, FL, Feb. 1998.

[4] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. PROMS: A web-based tool for searching in polyphonic music. In *Proc. of ISMIR*, Plymouth, MA, Oct. 2000. Available on the Internet: http://ciir.cs.umass.edu/music2000/.

[5] J. R. Deller, J. G. Proakis, and J. H. L. Hansen. *Discrete-Time Processing of Speech Signals*. Prentice Hall, Upper Saddle River, NJ, 1987.

[6] J. S. Downie. Music retrieval as text retrieval: Simple yet effective. In *Proc. of SIGIR*, pages 297–298, Berkeley, CA, Aug. 1999.

[7] J. Foote. An overview of audio information retrieval. *ACM Multimedia Systems J.*, 7(1):2–10, Jan. 1999.

[8] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: Musical information retrieval in an audio database. In *Proc. of ACM MM '95*, pages 231–236, San Francisco, CA, Nov. 1995.

[9] M. Goto. A predominant-F0 extimation method for CD recordings: MAP estimation using EM algorithm for adaptive tone models. In *Proc. of ICASSP*, Salt Lake City, UT, May 2001. Electronic Proc.

[10] B. Logan and S. Chu. Music summarization using key phrases. In *Proc. of ICASSP*, volume 2, pages 749–752, Istanbul, Turkey, June 2000.

[11] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten. The New Zealand Digital Library MELody inDEX. *D-Lib Magazine*, May 1997. Available on the Internet: http://www.dlib.org/dlib/may97/meldex/05witten.html.

[12] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, June 1990.

[13] S. Pfeiffer, S. Fischer, and W. Effelsberg. Automatic audio content analysis. In *Proc. of ACM MM '96*, pages 21–30, Boston, MA, Nov. 1996.

[14] L. R. Rabiner and B. J. Juang. An introduction to hidden Markov models. *IEEE Audio, Speech, and Signal Processing Magazine*, pages 4–16, Jan. 1986.

[15] C. Raphael. Automatic segmentation of acoustic musical signals using hidden Markov models. *IEEE Trans. on PAMI*, 21(4):360–370, April 1999.

[16] A. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In *Proc. of ACM MM '99*, pages 57–66, Orlando, FL, Oct.–Nov. 1999.

[17] A. L. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In *Proc. of ACM MM '98*, pages 235–240, Bristol, UK, Sept. 1998.

[18] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. R. Goldman. Automatic recognition of keywords in unconstrained speech using hidden Markov models. *IEEE Trans. on ASSP*, 38(11):1870–1878, Nov. 1990.

[19] P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young. Large vocabulary continuous speech recognition using HTK. In *Proc. of ICASSP*, volume 2, pages II–125–128, Adelaide, Australia, April 1994.

[20] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book (for HTK Version 3.0)*. The Microsoft Corporation, 2000. Available on the internet: http://htk.eng.cam.ac.uk/.

[21] T. Zhang and C.-C. Jay Kuo. Hierarchical system for content-based audio classification and retrieval. In *Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems III*, pages 398–409, Boston, MA, Nov. 1998.

APPENDIX

A. SONG TRANSCRIPTIONS

Note name transcriptions are provided here for each of the pieces of music used in testing our melody spotting system. Durations are excluded for space considerations and because the current implementation effectively normalizes them out of consideration. Examples of each piece may be heard at: `http://www.ece.gatech.edu/~gte401k/melodyspotting/` Though all pitches are played as notated, not all durations are played equally due to the natural variation that was desired in the training data.

Auld Lang Syne

C_4 F_4 E_4 F_4 A_4 G_4 F_4 G_4 A_4 G_4 F_4 F_4 A_4 C_5 D_5
D_5 C_5 A_4 A_4 F_4 G_4 F_4 G_4 A_4 G_4 F_4 D_4 D_4 C_4 F_4

Barbara Allen

C_4 E_4 F_4 G_4 F_4 E_4 D_4 C_4 D_4 E_4 G_4 C_5 C_5 B_4 G_4
C_5 C_5 A_4 G_4 F_4 A_4 G_4 E_4 D_4 C_4 D_4 E_4 F_4 G_4 F_4 E_4 D_4

Frere Jacques

G_4 A_4 B_4 G_4 G_4 A_4 B_4 G_4 B_4 C_5 D_5 B_4 C_5 D_5
D_5 E_5 D_5 C_5 B_4 G_4 D_5 E_5 D_5 C_5 B_4 G_4
G_4 D_4 G_4 G_4 D_4 G_4

Happy Birthday

G_4 G_4 A_4 G_4 C_5 B_4 G_4 G_4 A_4 G_4 D_5 C_5
G_4 G_4 G_5 E_5 C_5 B_4 A_4 F_5 F_5 E_5 C_5 D_5 C_5

I'm a Little Teapot

C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5 G_4
F_4 F_4 G_4 E_4 E_4 D_4 D_4 E_4 C_4
C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5 G_4
C_5 C_4 D_4 E_4 F_4 E_4 D_4 C_4

Mary Had a Little Lamb

B_4 A_4 G_4 A_4 B_4 B_4 B_4 A_4 A_4 A_4 B_4 D_5 D_5
B_4 A_4 G_4 A_4 B_4 B_4 B_4 B_4 A_4 A_4 B_4 A_4 G_4

Scarborough Fair

D_4 D_4 A_4 A_4 A_4 E_4 F_4 E_4 D_4
A_4 C_5 D_5 C_5 A_4 B_4 G_4 A_4
D_5 D_5 D_5 C_5 A_4 A_4 G_4 F_4 E_4 C_4
D_4 A_4 G_4 F_4 E_4 D_4 C_4 D_4

This Old Man

D_5 B_4 D_5 D_5 B_4 D_5 E_5 D_5 C_5 B_4 A_4 B_4 C_5
B_4 C_5 D_5 G_4 G_4 G_4 G_4 G_4 A_4 B_4 C_5 D_5
D_5 A_4 A_4 C_5 B_4 A_4 G_4

Three Blind Mice

E_4 D_4 C_4 E_4 D_4 C_4 G_4 F_4 F_4 E_4 G_4 F_4 F_4 E_4
G_4 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4
G_4 C_5 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4
G_4 G_4 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4 G_4 F_4 E_4 D_4 C_4

Twinkle, Twinkle, Little Star

G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5 B_4 B_4 A_4 A_4 G_4
D_5 D_5 C_5 C_5 B_4 B_4 A_4 D_5 D_5 C_5 C_5 B_4 B_4 A_4
G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5 B_4 B_4 A_4 A_4 G_4

B. FIGURE-OF-MERIT

HTK [20] uses the National Institute of Standards and Technology (NIST) wordspotting Figure-of-Merit defined as "an upper-bound estimate on word spotting accuracy averaged over 1 to 10 false alarms per hour." It is calculated using the following equation:

$$\text{FOM} = \frac{1}{10T}(p_1 + p_2 + ... + p_N + ap_{N+1})$$

where $a = 10T - N$ is a factor that interpolates to 10 false alarms per hour and T is the total hours of duration of the test data. To perform this calculation, first all of the spotted words (melodies) are ranked in score order. Then, the percentage of true hits p_i found before the i'th false alarm is calculated for $i = 1, ..., N + 1$ where N is the first integer greater than or equal to $10T - 0.5$. In the case of the experiments described here, $T = 0.5$ hours of recorded musical data, so:

$$\text{FOM} = \frac{1}{5}(p_1 + p_2 + p_3 + p_4 + p_5)$$