THE IMPORTANCE OF F0 TRACKING IN QUERY-BY-SINGING-HUMMING

Emilio Molina, Lorenzo J. Tardón, Isabel Barbancho, Ana M. Barbancho
Universidad de Málaga, ATIC Research Group, Andalucía Tech,
ETSI Telecomunicación, Campus de Teatinos s/n, 29071 Málaga, SPAIN
emm@ic.uma.es, lorenzo@ic.uma.es, ibp@ic.uma.es, abp@ic.uma.es

ABSTRACT

In this paper, we present a comparative study of several state-of-the-art F0 trackers applied to the context of query-by-singing-humming (QBSH). This study has been carried out using the well known, freely available, MIR-QBSH dataset in different conditions of added pub-style noise and smartphone-style distortion. For audio-to-MIDI melodic matching, we have used two state-of-the-art systems and a simple, easily reproducible baseline method. For the evaluation, we measured the QBSH performance for 189 different combinations of F0 tracker, noise/distortion conditions and matcher. Additionally, the overall accuracy of the F0 transcriptions (as defined in MIREX) was also measured. In the results, we found that F0 tracking overall accuracy correlates with QBSH performance, but it does not totally measure the suitability of a pitch vector for QBSH. In addition, we also found clear differences in robustness to F0 transcription errors between different matchers.

1. INTRODUCTION

Query-by-singing-humming (QBSH) is a music information retrieval task where short hummed or sung audio clips act as queries. Nowadays, several successful commercial applications for QBSH have been released, such as MusicRadar[1] or SoundHound[2], and it is an active field of research. Indeed, there is a task for QBSH in MIREX since 2006, and every year novel and relevant approaches can be found.

Typically, QBSH approaches firstly extract the F0 contour and/or a note-level transcription for a given vocal query, and then a set of candidate melodies are retrieved from a large database using a melodic matcher module. In the literature, many different approaches for matching in QBSH can be found: statistical, note vs. note, frame vs. note, frame vs. frame. Generally, state-of-the-art systems for QBSH typically combines different approaches in order to achieve more reliable results [3, 12].

[1] www.doreso.com
[2] www.soundhound.com

However, even state-of-the-art systems for QBSH have not a totally satisfactory performance in many real-world cases [1], so there is still room for improvement. Nowadays, some challenges related to QBSH are [2]: reliable pitch tracking in noisy environments, automatic song database preparation (predominant melody extraction and transcription), efficient search in very large music collections, dealing with errors of intonation and rhythm in amateur singers, etc.

In this paper, we analyse the performance of various state-of-the-art F0 trackers for QBSH in different conditions of background noise and smartphone-style distortion. For this study, we have considered three different melodic matchers: two state-of-the-art systems (one of which obtained the best results in MIREX 2013), and a simple, easily reproducible baseline method based on frame-to-frame matching using dynamic time warping (DTW). In Figure 1, we show a scheme of our study.

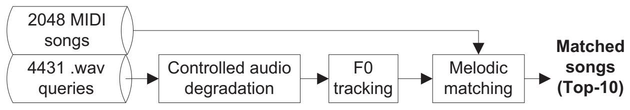

Figure 1. Overall scheme of our study

This paper is organized as follows: Section 2 and Section 3 present the studied algorithms for F0 tracking and melodic matching, respectively. The evaluation strategy is presented in Section 4. Section 5 presents the obtained results and Section 6 draws some conclusions about the present study.

2. F0 TRACKERS

In this section, we describe the F0 trackers considered in our study, together with their specific set of parameters. The literature reports a wide set of algorithms oriented to either monophonic or polyphonic audio, so we have focused on well-known, commonly used algorithms (e.g. Yin [4] or Praat-AC [8]), and some recently published algorithms for F0 estimation (e.g. pYin [6] or MELODIA [15]). Most of the algorithms analysed address F0 estimation in monophonic audio, but we have also studied the performance of MELODIA, which is a method for predominant melody extraction in polyphonic audio, using monophonic audio in noisy conditions. Regarding the used set of parameters, when possible, they have been adjusted by trial and error using ten audio queries. The considered methods for F0 tracking are the following ones:

2.1 YIN

The Yin algorithm was developed by de Cheveigné and Kawahara in 2002 [4]. It resembles the idea of the autocorrelation method [5] but it uses the cumulative mean normalized difference function, which peaks at the local period with lower error rates than the traditional autocorrelation function. In our study, we have used Matthias Mauch's VAMP plugin[3] in Sonic Annotator tool[4].

Parameters used in YIN: step size = 80 samples (0.01 seconds), Block size = 512 samples, Yin threshold = 0.15.

2.2 pYIN

The pYin method has been published by Mauch in 2014 [6], and it basically adds a HMM-based F0 tracking stage in order to find a "smooth" path through the fundamental frequency candidates obtained by Yin. Again, we have used the original Matthias Mauch's VAMP plugin[3] in Sonic Annotator tool[4].

Parameters used in PYIN: step size = 80 samples (0.01 seconds), Block size = 512 samples, Yin threshold distribution = Beta (mean 0.15).

2.3 AC-DEFAULT and AC-ADJUSTED (Praat)

Praat is a well-known tool for speech analysis [7], which includes several methods for F0 estimation. In our case, we have chosen the algorithm created by P. Boersma in 1993 [8]. It is based on the autocorrelation method, but it improves it by considering the effects of the window during the analysis and by including a F0 tracking stage based on dynamic programming. This method has 9 parameters that can be adjusted to achieve a better performance for a specific application. According to [9], this method significantly improves its performance when its parameters are adapted to the input signal. Therefore, we have experimented not only with the default set of parameters (AC-DEFAULT), but also with an adjusted set of parameters in order to limit octave jumps and false positives during the voicing process (AC-ADJUSTED). In our case, we have used the implementation included in the console Praat tool.

Parameters used in AC-DEFAULT: Time step = 0.01 seconds, Pitch floor = 75Hz, Max. number of candidates = 15, Very accurate = off, Silence threshold = 0.03, Voicing threshold = 0.45, Octave cost = 0.01, Octave-jump cost = 0.35, Voiced / unvoiced cost = 0.15, Pitch ceiling = 600 Hz.

Parameters used in AC-ADJUSTED: Time step = 0.01 seconds, Pitch floor = 50Hz, Max. number of candidates = 15, Very accurate = off, Silence threshold = 0.03, Voicing threshold = 0.45, Octave cost = 0.1, Octave-jump cost = 0.5, Voiced / unvoiced cost = 0.5, Pitch ceiling = 700 Hz.

2.4 AC-LEIWANG

In our study we have also included the exact F0 tracker used in Lei Wang's approach for QBSH [3], which obtained the best results for most of the datasets in MIREX 2013. It is based on P. Boersma's autocorrelation method [8], but it uses a finely tuned set of parameters and a post-processing stage in order to mitigate spurious and octave errors. This F0 tracker is used in the latest evolution of a set of older methods [11, 12] also developed by Lei Wang (an open source C++ implementation is available[5]).

2.5 SWIPE'

The Swipe' algorithm was published by A. Camacho in 2007 [10]. This algorithm estimates the pitch as the fundamental frequency of the sawtooth waveform whose spectrum best matches the spectrum of the input signal. The algorithm proved to outperform other well-known F0 estimation algorithms, and it is used in the F0 estimation stage of some state-of-the-art query-by-humming systems [13]. In our study, we have used the original author's Matlab implementation[6]. The Matlab code does not provide a voiced / unvoiced classification of frames, but it outputs a strength vector `S` which has been used for it. Specifically, a frame is considered voiced if its strength is above a threshold S_{th}, otherwise they are considered unvoiced.

Parameters used in SWIPE': DT (hop-size) = 0.01 seconds, pmin = 50 Hz, pmax = 700Hz, dlog2p = 1/48 (default), dERBs = 0.1 (default), woverlap = 0.5 (default), voicing threshold S_{th} = 0.3.

2.6 MELODIA-MONO and MELODIA-POLY

MELODIA is a system for automatic melody extraction in polyphonic music signals developed by Salamon in 2012 [15]. This system is based on the creation and characterisation of pitch contours, which are time continuous sequences of pitch candidates grouped using auditory streaming cues. Melodic and non-melodic contours are distinguished depending on the distributions of its characteristics. The used implementation is MELODIA VAMP plugin[7] in Sonic Annotator tool[4]. This plugin has two default sets of parameters, adapted to deal with monophonic or polyphonic audio. We have experimented with both of them, and therefore we have defined two methods: MELODIA-MONO and MELODIA-POLY.

Parameters used in MELODIA-MONO: Program = Monophonic, Min Frequency = 55Hz, Max Frequency = 700Hz, Voicing Tolerance = 3,00, Monophonic Noise Filter = 0,00, Audio block size = 372 (not configurable), Window increment = 23 (not configurable).

Parameters used in MELODIA-POLY: Program = Polyphonic, Min Frequency = 55Hz, Max Frequency = 700Hz, Voicing Tolerance = 0,20, Monophonic Noise Filter = 0,00, Audio block size = 372 (not configurable), Window increment = 23 (not configurable).

Note that the time-step in this case can not be directly set to 0.01 seconds. Therefore, we have linearly interpolated the pitch vector in order to scale it to a time-step of 0.01 seconds.

[3] http://code.soundsoftware.ac.uk/projects/pyin
[4] http://www.vamp-plugins.org/sonic-annotator/

[5] http://www.atic.uma.es/ismir2014qbsh/
[6] http://www.cise.ufl.edu/ acamacho/publications/swipep.m
[7] http://mtg.upf.edu/technologies/melodia

3. AUDIO-TO-MIDI MELODIC MATCHERS

In this section, we describe the three considered methods for audio-to-MIDI melodic matching: a simple baseline (Section 3.1) and two state-of-the-art matchers (Sections 3.2 and 3.3).

3.1 Baseline approach

We have implemented a simple, freely available [5] baseline approach based on dynamic time warping (DTW) for melodic matching. Our method consists of four steps (a scheme is shown in Figure 2):

(1) Model building: We extract one pitch vector $\mathbf{P^k}$ (in MIDI number) for every target MIDI song $k \in 1 \ldots N_{\text{songs}}$ using a hop-size of 0.01 seconds. Then we replace unvoiced frames (rests) in $\mathbf{P^k}$ by the pitch value of the previous note, except for the case of initial unvoiced frames, which are directly removed (these processed pitch vectors are labelled as $\mathbf{P^{*k}}$). Then, each pitch vector $\mathbf{P^{*k}}$ $\forall k \in 1 \ldots N_{\text{songs}}$ is truncated to generate 7 pitch vectors with lengths [500, 600, 700, 800, 900, 1000, 1100] frames (corresponding to the first 5, 6, 7, 8, 9, 10 and 11 seconds of the target MIDI song, which are reasonable durations for an user query). We label these pitch vectors as $\mathbf{P^{*k}_{5s}}$, $\mathbf{P^{*k}_{6s}}, \ldots \mathbf{P^{*k}_{11s}}$. Finally, all these pitch vectors are resampled (through linear interpolation) to a length of 50 points, and then zero-mean normalized (for a common key transposition), leading to $\mathbf{P^{50*k}_{\mathit{Duration}}}$ $\forall \mathit{Duration} \in 5s \ldots 11s$ and $\forall k \in 1 \ldots N_{\text{songs}}$. These vectors are then stored for later usage. Note that this process must be done only once.

(2) Query pre-processing: The pitch vector $\mathbf{P^Q}$ of a given .wav query is loaded (note that all pitch vectors are computed with a hopsize equal to 0.01 seconds). Then, as in step (1), unvoiced frames are replaced by the pitch value of the previous note, except for the case of initial unvoiced frames, which are directly removed. This processed vector is then converted to MIDI numbers with 1 cent resolution, and labelled as $\mathbf{P^{*Q}}$. Finally, $\mathbf{P^{*Q}}$ is resampled (using linear interpolation) to a length $L = 50$ and zero-mean normalized (for a common key transposition), leading to $\mathbf{P^{50*Q}}$.

(3) DTW-based alignment: Now we find the optimal alignment between $\mathbf{P^{50*Q}}$ and all pitch vectors $\mathbf{P^{50*k}_{\mathit{Duration}}}$ $\forall \mathit{Duration} \in 5s \ldots 11s$ and $\forall k \in 1 \ldots N_{\text{songs}}$ using dynamic time warping (DTW). In our case, each cost matrix $\mathbf{C^{Duration,k}}$ is built using the squared difference:

$$C^{Duration,k}(i,j) = (P^{50*Q}(i) - P^{50*k}_{\mathit{Duration}}(j))^2 \qquad (1)$$

Where k is the target song index, $\mathit{Duration}$ represents the truncation level (from 5s to 11s), and i, j are the time indices of the query pitch vector $\mathbf{P^{50*Q}}$ and the target pitch vector $\mathbf{P^{50*k}_{\mathit{Duration}}}$, respectively. The optimal path is now found using Dan Ellis' Matlab implementation for DTW [16] (`dpfast.m` function), with the following allowed steps and associated cost weights $[\Delta i, \Delta j, W]$: $[1,1,1]$, $[1,0,30]$, $[0,1,30]$, $[1,2,5]$, $[2,1,5]$. The allowed steps and weights have been selected in order to penalize 0 or 90 angles in the optimal path (associated to unnatural alignments), and although they lead to acceptable results, they may not be optimal.

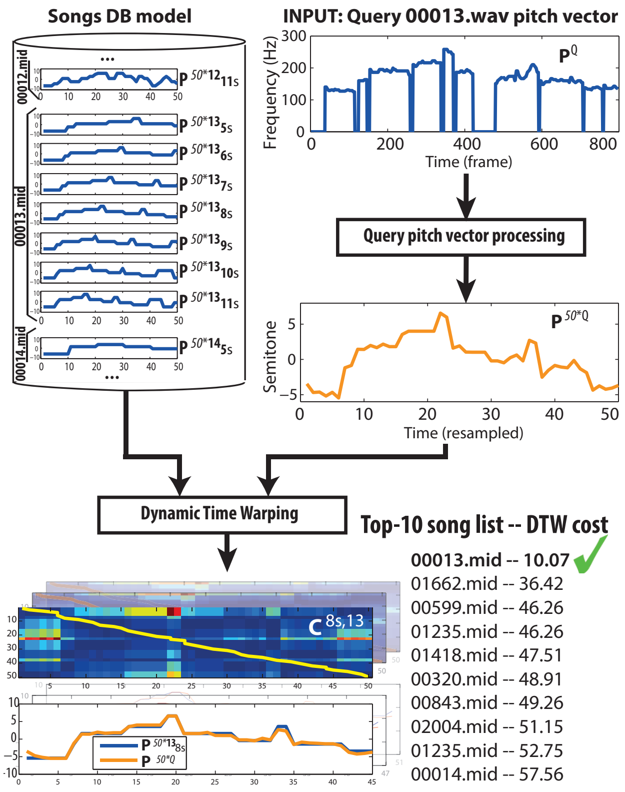

Figure 2. Scheme of the proposed baseline method for audio-to-MIDI melody matching.

(4) Top-10 report: Once the $\mathbf{P^{50*Q}}$ has been aligned with all target pitch vectors (a total of $7 \times N_{\text{songs}}$ vectors, since we use 7 different durations), the matched pitch vectors are sorted according to their alignment total cost (this value consists of the matrix `D` produced by `dpfast.m` evaluated in the last position of the optimal path, `Tcost = D(p(end),q(end))`). Finally, the 10 songs with minimum cost are reported.

3.2 Music Radar's approach

MusicRadar [3] is a state-of-the-art algorithm for melodic matching, which participated in MIREX 2013 and obtained the best accuracy in all datasets, except for the case of IOACAS [8]. It is the latest evolution of a set of systems developed by Lei Wang since 2007 [11, 12]. The system takes advantage of several matching methods to improve its accuracy. First, Earth Mover's Distance (EMD), which is note-based and fast, is adopted to eliminate most unlikely candidates. Then, Dynamic Time Warping (DTW), which is frame-based and more accurate, is executed on these surviving candidates. Finally, a weighted voting fusion strategy is employed to find the optimal match. In our study, we have used the exact melody matcher tested in MIREX 2013, provided by its original author.

3.3 NetEase's approach

NetEase's approach [13] is a state-of-the-art algorithm for melodic matching, which participated in MIREX 2013 and

[8] http://www.music-ir.org/mirex/wiki/2013:Query_by_Singing/Humming

obtained the first position for IOACAS dataset[8], as well as relevant results in the rest of datasets. This algorithm adopts a two-stage cascaded solution based on Locality Sensitive Hashing (LSH) and accurate matching of frame-level pitch sequence. Firstly, LSH is employed to quickly filter out songs with low matching possibilities. In the second stage, Dynamic Time Warping is applied to find the N (set to 10) most matching songs from the candidate list. Again, the original authors of NetEase's approach (who also authored some older works on query-by-humming [14]) collaborated in this study, so we have used the exact melody matcher tested in MIREX 2013.

4. EVALUATION STRATEGY

In this section, we present the datasets used in our study (Section 4.1), the way in which we have combined F0 trackers and melody matchers (Section 4.2) and the chosen evaluation measures (Section 4.3).

4.1 Datasets

We have used the public corpus MIR-QBSH[8] (used in MIREX since 2005), which includes 4431 .wav queries corresponding to 48 different MIDI songs. The audio queries are 8 seconds length, and they are recorded in mono 8 bits, with a sample rate of 8kHz. In general, the audio queries are monophonic with no background noise, although some of them are slightly noisy and/or distorted. This dataset also includes a manually corrected pitch vector for each .wav query. Although these annotations are fairly reliable, they may not be totally correct, as stated in MIR-QBSH documentation.

In addition, we have used the Audio Degradation Toolbox [17] in order to recreate common environments where a QBSH system could work. Specifically, we have combined three levels of pub-style added background noise (`PubEnvironment1` sound) and smartphone-style distortion (`smartPhoneRecording` degradation), leading to a total of seven evaluation datasets: (1) Original MIR-QBSH corpus (2) 25 dB SNR (3) 25 dB SNR + smartphone distortion (4) 15 dB SNR (5) 15 dB SNR + smartphone distortion (6) 5 dB SNR (7) 5 dB SNR + smartphone distortion. Note that all these degradations have been checked in order to ensure perceptually realistic environments.

Finally, in order to replicate MIREX conditions, we have included 2000 extra MIDI songs (randomly taken from ESSEN collection[9]) to the original collection of 48 MIDI songs, leading to a songs collection of 2048 MIDI songs. Note that, although these 2000 extra songs fit the style of the original 48 songs, they do not correspond to any .wav query of Jang's dataset.

4.2 Combinations of F0 trackers and melody matchers

For each of the 7 datasets, the 4431 .wav queries have been transcribed using the 8 different F0 trackers mentioned in Section 2. Additionally, each dataset also includes the 4431 manually corrected pitch vectors of MIR-QBSH as a reference, leading to a total of 7 datasets $\times$ (8 F0 trackers + 1 manual annotation) $\times$ 4431 queries = 63 $\times$ 4431 queries = 279153 pitch vectors. Then, all these pitch vectors have been used as input to the 3 different melody matchers mentioned in Section 3, leading to 930510 lists of top-10 matched songs. Finally, these results have been used to compute a set of meaningful evaluation measures.

4.3 Evaluation measures

In this section, we present the evaluation measures used in this study:

(1) Mean overall accuracy of F0 tracking ($\overline{\mathbf{Acc_{ov}}}$): For each pitch vector we have computed an evaluation measures defined in MIREX Audio Melody Extraction task: *overall accuracy* ($\mathrm{Acc_{ov}}$) (a definition can be found in [15]). The *mean overall accuracy* is then defined as $\overline{\mathrm{Acc_{ov}}} = (1/N)\sum_{i=1}^{N} \mathrm{Acc_{ov}}_i$, where N is the total number of queries considered and $\mathrm{Acc_{ov}}_i$ is the overall accuracy of the pitch vector of the i:th query. We have selected this measure because it considers both voicing and pitch, which are important aspects in QBSH. For this measure, our ground truth consists of the manually corrected pitch vectors of the .wav queries, which are included in the original MIR-QBSH corpus.

(2) Mean Reciprocal Rank (MRR): This measure is commonly used in MIREX Query By Singing Humming task[8], and it is defined as: $\mathrm{MRR} = (1/N)\sum_{i=1}^{N} r_i^{-1}$, where N is the total number of queries considered and r_i is the rank of the correct answer in the retrieved melodies for i:th query.

5. RESULTS & DISCUSSION

In this section, we present the obtained results and some relevant considerations about them.

5.1 $\overline{\mathbf{Acc_{ov}}}$ and MRR for each F0 tracker - Dataset - Matcher

In Table 1, we show the $\overline{\mathrm{Acc_{ov}}}$ and the MRR obtained for the whole dataset of 4431 .wav queries in each combination of F0 tracker-dataset-matcher (189 combinations in total). Note that these results are directly comparable to MIREX Query by Singing/Humming task[8] (Jang Dataset). As expected, the manually corrected pitch vectors produce the best MRR in most cases (the overall accuracy is 100% because it has been taken as the ground truth for such measure). Note that, despite manual annotations are the same in all datasets, NetEase and MusicRadar matchers do not produce the exact same results in all cases. It is due to the generation of the indexing model (used to reduced the time search), which is not a totally deterministic process.

Regarding the relationship between $\overline{\mathrm{Acc_{ov}}}$ and MRR in the rest of F0 trackers, we find a somehow contradictory result: the best $\overline{\mathrm{Acc_{ov}}}$ does not always correspond with the best MRR. This fact may be due to two different reasons. On the one hand, the meaning of $\overline{\mathrm{Acc_{ov}}}$ may be distorted due to annotation errors in the ground truth (as mentioned in Section 4.1), or to eventual intonation errors in the dataset. However, the manual annotations produce the best MRR, what suggests that the amount of these types

[9] www.esac-data.org/

F0 tracker	Clean dataset	25dB SNR	25 dB SNR + distortion	15dB SNR	15 dB SNR + distortion	5dB SNR	5 dB SNR + distortion
(A)	100 / 0.82 / 0.89 / 0.96	100 / 0.82 / 0.89 / 0.96	100 / 0.82 / 0.89 / 0.95	100 / 0.82 / 0.89 / 0.96	100 / 0.82 / 0.89 / 0.96	100 / 0.82 / 0.89 / 0.96	100 / 0.82 / 0.88 / 0.95
(B)	89 / **0.80 / 0.89 / 0.96**	89 / **0.80 / 0.89 / 0.96**	**88 / 0.80 / 0.88 / 0.95**	88 / **0.79 / 0.88 / 0.94**	84 / 0.71 / 0.86 / 0.94	78 / 0.50 / 0.73 / 0.85	67 / 0.33 / 0.57 / 0.73
(C)	**90** / 0.74 / 0.85 / 0.94	90 / 0.71 / 0.85 / 0.92	86 / 0.72 / 0.84 / 0.92	89 / 0.71 / 0.84 / 0.92	85 / 0.66 / 0.81 / 0.89	72 / 0.49 / 0.58 / 0.70	64 / 0.26 / 0.39 / 0.51
(D)	90 / 0.71 / 0.83 / 0.92	**90** / 0.74 / 0.85 / 0.93	85 / 0.74 / 0.85 / 0.94	**90** / 0.78 / 0.87 / 0.94	**85 / 0.77 / 0.87 / 0.94**	79 / **0.69 / 0.79 / 0.87**	72 / **0.58 / 0.69 / 0.81**
(E)	89 / 0.71 / 0.83 / 0.92	89 / 0.71 / 0.84 / 0.92	84 / 0.66 / 0.80 / 0.91	88 / 0.72 / 0.84 / 0.93	83 / 0.65 / 0.80 / 0.91	75 / 0.67 / 0.67 / 0.82	66 / 0.48 / 0.53 / 0.73
(F)	86 / 0.62 / 0.81 / 0.89	86 / 0.70 / 0.83 / 0.92	81 / 0.64 / 0.78 / 0.89	82 / 0.60 / 0.77 / 0.88	75 / 0.50 / 0.67 / 0.82	48 / 0.03 / 0.08 / 0.04	44 / 0.04 / 0.04 / 0.03
(G)	88 / 0.56 / 0.81 / 0.88	87 / 0.47 / 0.79 / 0.86	83 / 0.47 / 0.76 / 0.85	86 / 0.39 / 0.78 / 0.87	81 / 0.35 / 0.73 / 0.82	70 / 0.11 / 0.32 / 0.52	63 / 0.04 / 0.20 / 0.38
(H)	87 / 0.66 / 0.83 / 0.87	87 / 0.67 / 0.82 / 0.87	83 / 0.64 / 0.78 / 0.84	86 / 0.66 / 0.81 / 0.84	82 / 0.58 / 0.74 / 0.80	83 / 0.51 / 0.73 / 0.75	73 / 0.32 / 0.55 / 0.62
(I)	84 / 0.62 / 0.76 / 0.86	84 / 0.62 / 0.76 / 0.86	79 / 0.50 / 0.64 / 0.74	84 / 0.63 / 0.76 / 0.86	79 / 0.50 / 0.65 / 0.75	**83** / 0.60 / 0.73 / 0.83	**75** / 0.39 / 0.55 / 0.65

Table 1: F0 overall accuracy and MRR obtained for each case. F0 trackers: (A) *MANUALLY CORRECTED* (B) *AC-LEIWANG* (C) *AC-ADJUSTED* (D) *PYIN* (E) *SWIPE'* (F) *YIN* (G) *AC-DEFAULT* (H) *MELODIA-MONO* (I) *MELODIA-POLY*. The format of each cell is: $\overline{\mathrm{Acc}_{\mathrm{ov}}}(\%)$ / *MRR-baseline* / *MRR-NetEase* / *MRR-MusicRadar*.

of errors are low. On the other hand, the measure $\overline{\mathrm{Acc}_{\mathrm{ov}}}$ itself may not be totally representative of the suitability of a pitch vector for QBSH. Indeed, after analysing specific cases, we observed that two pitch vectors with same F0 tracking accuracy (according to MIREX measures) may not be equally suitable for query-by-humming. For instance, we analysed the results produced by the baseline matcher using two different pitch vectors (Figure 3) with exactly the same evaluation measures in MIREX Audio Melody Extraction task: *vocing recall* $= 99.63\%$, *voicing false-alarm* $= 48.40\%$, *raw pitch accuracy*$= 97.41\%$, *raw-chroma accuracy* $= 97.41\%$ and *overall accuracy* $= 82.91\%$. However, we found that pitch vector (a) matches the right song with rank $r_i = 1$ whereas pitch vector (b) does not matches the right song at all ($r_i \geq 11$). The reason is that MIREX evaluation measures do not take into account the pitch values of false positives, but in fact they are important for QBSH. Therefore, we conclude that the high MRR achieved by some F0 trackers (AC-LEIWANG when background noise is low, and PYIN for highly degraded signals), is not only due to the amount of errors made by them, but also to the type of such errors.

Additionally, we observed that, in most cases, the queries are matched either with rank $r_i = 1$ or $r_i \geq 11$ (intermediate cases such as rank $r_i = 2$ or $r_i = 3$ are less frequent). Therefore, the variance of ranks is generally high, their distribution is not Gaussian.

5.2 MRR vs. $\overline{\mathrm{Acc}_{\mathrm{ov}}}$ for each matcher

In order to study the robustness of each melodic matcher to F0 tracking errors, we have represented the MRR obtained by each one for different ranges of $\overline{\mathrm{Acc}_{\mathrm{ov}}}$ (Figure 4). For this experiment, we have selected only the .wav queries which produce the right answer in first rank for the three matchers considered (baseline, Music Radar and NetEase) when manually corrected pitch vectors are used (around a 70% of the dataset matches this condition). In this way, we ensure that bad singing or a wrong manual annotation is not affecting the variations of MRR in the plots. Note that, in this case, the results are not directly comparable to the ones computed in MIREX (in contrast to the results shown in Section 5.1).

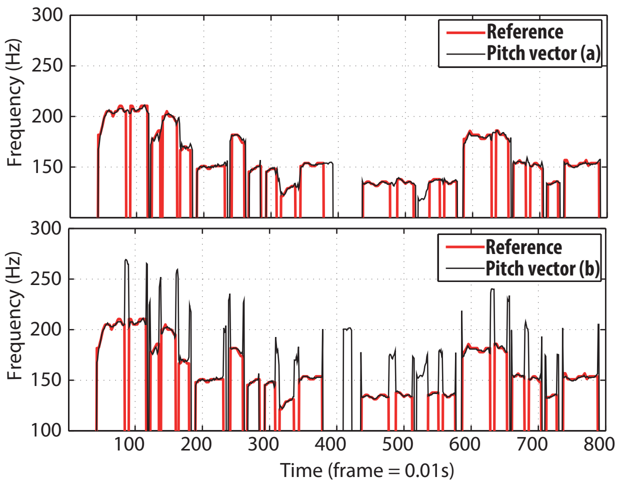

Figure 3. According to MIREX measures, these two pitch vectors (manually manipulated) are equally accurate; however, they are not equally suitable for QBSH.

Regarding the obtained results (shown in Figure 4), we observe clear differences in the robustness to F0 estimation errors between matchers, which is coherent with the results presented in Table 1. The main difference is found in the baseline matcher with respect to both NetEase and Music Radar. Given that the baseline matcher only uses DTW, whereas the other two matchers use a combination of various searching methods (see Sections 3.2 and 3.3), we hypothesise that such combination may improve their robustness to F0 tracking errors. However, further research is needed to really test this hypothesis.

6. CONCLUSIONS

In this paper, eight different state-of-the-art F0 trackers were evaluated for the specific application of query-by-humming-singing in different conditions of pub-style added noise and smartphone-style distortion. This study was carried out using three different matching methods: a simple, freely available baseline (a detailed description has been provided in Section 3.1) and two state-of-the-art matchers. In our results, we found that Boersma's AC method [8], with an appropriate adjustment and a smoothing stage

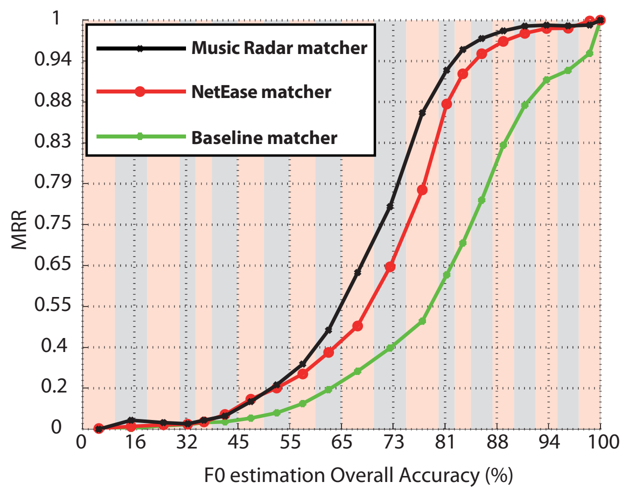

Figure 4. MRR obtained for each range of Overall Accuracy (each range is marked with coloured background rectangles). We have considered only the .wav queries which, using manually corrected F0 vectors, produce MRR $= 1$ in all matchers.

achieves the best results when the audio is not very degraded. In contrast, when the audio is highly degraded, the best results are obtained with pYIN [6], even without further smoothing. Considering that pYIN is a very recent, open source approach, this result is promising in order to improve the noise robustness of future QBSH systems. Additionally, we found that F0 trackers perform differently on QBSH depending on the type of F0 tracking errors made. Due to this, MIREX measures do not fully represent the suitability of a pitch vector for QBSH purposes, so the development of novel evaluation measures in MIREX is encouraged to really measure the suitability of MIR systems for specific applications. Finally, we observed clear differences between matchers regarding their robustness to F0 estimation errors. However, further research is needed for a deeper insight into these differences.

7. ACKNOWLEDGEMENTS

Special thanks to Doreso [1] team (especially to Lei Wang and Yuhang Cao) and to Peng Li for their active collaboration in this study. This work has been funded by the Ministerio de Economía y Competitividad of the Spanish Government under Project No. TIN2013-47276-C6-2-R and by the Junta de Andalucía under Project No. P11-TIC-7154. The work has been done at Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech.

8. REFERENCES

[1] A. D. Brown and Brighthand staff: "SoundHound for Android OS Review: 'Name That Tune,' But At What Price?", *Brighhand Smartphone News & Review*, 2012. Online: www.brighthand.com [Last Access: 28/04/2014]

[2] J. -S. Roger Jang: "QBSH and AFP as Two Successful Paradigms of Music Information Retrieval" Course in *RuSSIR*, 2013. Available at: http://mirlab.org/jang/ [Last Access: 28/04/2014]

[3] Doreso Team (www.doreso.com): "MIREX 2013 QBSH Task: Music Radar's Solution" *Extended abstract for MIREX*, 2013.

[4] A. De Cheveigné and H. Kawahara: "YIN, a fundamental frequency estimator for speech and music," *Journal of the Acoustic Society of America*, Vol. 111, No. 4, pp. 1917-1930, 2002.

[5] L. Rabiner: "On the use of autocorrelation analysis for pitch detection," *Acoustics, Speech and Signal Processing, IEEE Transactions on,*, Vol. 25, No.1, pp. 24-33. 1977.

[6] M. Mauch, and S. Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," *Proceedings of ICASSP*, 2014.

[7] P. Boersma and D. Weenink: "Praat: a system for doing phonetics by computer," *Glot international*, Vol. 5, No. 9/10, pp. 341-345, 2002. Software available at: www.praat.org [Last access: 28/04/2014]

[8] P. Boersma: "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," *Proceedings of the Institute of Phonetic Sciences*, Vol. 17, No. 1193, pp 97-110, 1993.

[9] E. Keelan, C. Lai, K. Zechner: "The importance of optimal parameter setting for pitch extraction," *Journal of Acoustical Society of America*, Vol. 128, No. 4, pp. 2291–2291, 2010.

[10] A. Camacho: "SWIPE: A sawtooth waveform inspired pitch estimator for speech and music," PhD dissertation, University of Florida, 2007.

[11] L. Wang, S. Huang, S. Hu, J. Liang and B. Xu: "An effective and efficient method for query-by-humming system based on multi-similarity measurement fusion," *Proceedings of ICALIP*, 2008.

[12] 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, set reduction and multiple similarity measurement rescoring," *Proceedings of INTERSPEECH*, 2008.

[13] P. Li, Y. Nie and X. Li: "MIREX 2013 QBSH Task: Netease's Solution" *Extended abstract for MIREX*, 2013.

[14] P. Li, M. Zhou, X. Wang and N. Li: "A novel MIR system based on improved melody contour definition," *Proceedings of the International Conference on MultiMedia and Information Technology (MMIT)*, 2008.

[15] J. Salamon and E. Gómez: "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics," *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 20, No. 6, pp. 1759–1770, 2012.

[16] D. Ellis: "Dynamic Time Warp (DTW) in Matlab", 2003. Web resource, available: www.ee.columbia.edu/ dpwe/resources/matlab/dtw/ [Last Access: 28/04/2014]

[17] M. Mauch and S. Ewert: "The Audio Degradation Toolbox and its Application to Robustness Evaluation," *Proceedings of ISMIR*, 2013.