MIREX 2006

The Second Annual Music Information Retrieval Evaluation eXchange

Abstract Collection

Edited by:
The International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL)

Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign

Table of Contents

Audio Music Similarity and Retrieval Task

Audio Onset Detection Task

Audio Tempo Extraction Task

Query-by-Singing/Humming Task

Symbolic Melodic Similarity Task

Score Following Task

THE AUBIO LIBRARY AT MIREX 2006

Paul M. Brossier
Centre for Digital Music
Queen Mary University of London
Mile End Road, London
piem@altern.org

ABSTRACT

For the 2006 edition of the Music Information Retrieval Evaluation eXchange (MIREX), we presented the latest enhancements to the aubio library, participating this year in four different tasks: Audio Melody Extraction, Audio Tempo Extraction, Audio Beat Tracking and Audio Onset Detection. The algorithms we submitted are tailored for causal operation and interactive systems. We describe here the outline of these methods with pointers to the relevant references. The evolution of the algorithm performances from last year results is discussed, and this year's results are compared to that of other systems.

Keywords: MIREX, aubio, onset, melody, tempo.

1 INTRODUCTION

In (Brossier, 2006), we describe our investigations on the automatic annotation of musical signals for interactive systems. This study includes the evaluation of several methods for onset detection, pitch extraction and beat tracking. Different algorithms were implemented to work in real time and evaluated on large databases of hand annotated data.

Our implementation, the aubio library, was designed to use these annotation routines in interactive systems. The library, along with the programs we submitted to the 2006 edition of MIREX, is available under the GNU General Public License (Brossier, 2003).

2 ALGORITHMS

We give here a brief description of the algorithms we submitted for the different tasks of MIREX. For a detailed description of these methods, see (Brossier, 2006).

2.1 Onset detection

In (Bello et al., 2005), a number onset detection methods were reviewed, several of which based on a phase vocoder. For our causal implementation, we have implemented different onset detection methods with several modifications for real time operations (Brossier et al., 2004).

The onset detection algorithms consist of a phase vocoder, a detection function built from consecutive spectral frames obtained from the phase vocoder, and a peak picking algorithm tailored to find peaks in the detection function corresponding to actual onsets and within a short delay. A silence gate is also used to avoid spurious detections in areas of low energy.

We submitted four different functions, each to be run with different pick peaking threshold values: high frequency content (Masri, 1996), spectral difference function (Foote and Uchihashi, 2001), complex domain function (Duxbury et al., 2003), and a dual function, built using both HFC and Kullback Liebler function (Hainsworth and Macleod, 2003). A detailed study of this implementation is given in (Brossier, 2006, Chap. 2).

2.2 Beat tracking

The algorithm submitted for beat tracking was first described in (Davies and Plumbley, 2004), with further improvements for its real-time implementation in aubio in (Davies et al., 2005). The approach makes use of an onset detection function built on a phase vocoder, similar to that used in Section 2.1. A modified autocorrelation of the onset detection function is then computed to determine the beat period and phase alignment. Based on the detected period and phase, beats are predicted for the following frames, so that the beats can be tapped along live audio streams. A detailed study of this implementation is given in (Brossier, 2006, Chap. 4).

2.3 Tempo extraction

The method we use for the extraction of tempo is based on beat tracking algorithm described in Section 2.2. The MIREX evaluation metrics used for tempo extraction requires two different tempo values and their corresponding phases. The primary tempo values is computed by our algorithm as the median of all beat periods found in the file. In order to produce the two tempi output required by the evaluation metrics, we use a set of simple rules: if the primary tempo period is greater than 95 beats per minute (BPM), the secondary tempo period is set to half the primary period; if instead the primary tempo period is found smaller than 95 BPM, the secondary period is set to twice the primary period. The value of 95 was determined empirically based on the training set provided for this task.

2.4 Melody extraction

The algorithm we have submitted to the melody extraction task is designed for fundamental frequency estimation. The method is derived from the YIN algorithm (de Cheveigné and Kawahara, 2002), and was detailed in (Brossier, 2006, Chap. 3). A major modification to the YIN algorithm is the use of a tapered squared difference function to build the cumulative normalised sum. This modification was found to facilitate the selection of the fundamental period and to increase the robustness of the method in the presence of noise and other instruments.

3 SOFTWARE IMPLEMENTATION

The aubio library consists in a collection of C routines. The programs submitted for MIREX make use of the Python external for aubio. For code availability, see (Brossier, 2003).

Using the Python interface is convenient for rapid prototyping and debugging. However, using Python causes an increase of the run times of the algorithms, since the Python environment has be loaded and unloaded at each run, unlike for their C equivalent. The actual computation times required for each algorithms were measured in (Brossier, 2006), where detailed comparisons of the run times obtained for different onset and pitch detection methods are given.

4 RESULTS DISCUSSION

4.1 Onset detection

The Precision-Recall graphs used to display the onset detection results give an interesting comparison of the performance of the different algorithms on different classes of sounds. These graphs highlight the different performances achieved by different methods on different sound classes. The ranking of the algorithms are indeed very dependant of the type of sounds, suggesting that the final ranking is very dependant on the overall distribution in the database.

Only minor changes were applied to the onset detection routines since our submission to the 2005 edition of MIREX, so that the optimal results we obtain are very similar to that of last year. The study of the threshold parameter, with values in the range 0.1 to 0.9, demonstrates the ability of our adaptive thresholding approach to move from under-segmentation to over-segmentation, regardless of the function we use.

4.2 Beat tracking

For its first edition, the beat tracking task has seen five different submissions. The P-score evaluation measure gave results in the range 0.407 to 0.391, where our algorithm ranked last. Given the small differences between submissions, it would be interesting to have more detailed results. However, our algorithm is not designed to extract beat locations at the very beginning of the file, and the P-score we obtained is coherent with the one of M. E. Davies submission, which interpolates beat locations at the beginning of the file, and which ranked 4^{th} with a P-score of 0.394.

The aubio submission obtained the shortest run time amongst submissions, with 139 seconds to run through the test set, while other submissions were executed in between 498 seconds and 1394 seconds.

4.3 Tempo extraction

With an average of 78.57% of files annotated with the correct tempo period, results showed that our algorithm actually performed worst than last year (80.71%), which seems to indicate that parameters are over fitted to the test data set. However, the empirical rule we used to determine the second tempo value seem fairly efficient, with 50.71% of the files annotated with both correct tempi, whereas M. E. Davies submission detected both correct tempi in 45.71% of the files.

4.4 Melody extraction

The results we obtained for this year MIREX edition are encouraging, since no post-processing was done on the pitch track, and the output of the pitch detection were used directly for evaluation. As we have no mechanism to decide whether or not a frame is voiced, the voicing false alarm rate is not relevant for our algorithm.

ACKNOWLEDGEMENTS

Many thanks to the MIREX organisation team for another successful edition of MIREX, and especially to Andreas F. Ehmann for pointing out several issues in the initial submissions.

References

Juan-Pablo Bello, Laurent Daudet, Samer Abdallah, Christopher Duxbury, Mike P. Davies, and Mark B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions in Speech and Audio Processing*, 13 (5):1035–1047, September 2005.

Paul M. Brossier. Aubio, a library for audio labelling. `http://aubio.piem.org`, 2003. Last visited: Fri Octobre 6 2006.

Paul M. Brossier. *Automatic annotations of musical audio for interactives systems*. PhD thesis, Centre for Digital Music, Queen Mary University of London, London, UK, 2006. submitted for publication.

Paul M. Brossier, Juan-Pablo Bello, and Mark D. Plumbley. Real-time temporal segmentation of note objects in music signals. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 458–461, Miami, Florida, USA, November 2004.

Matthew E. P. Davies and Mark D. Plumbley. Causal tempo tracking of audio. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 164–169, Barcelona, Spain, October 2004.

Matthew E. P. Davies, Paul M. Brossier, and Mark D. Plumbley. Beat tracking towards automatic musical accompaniment. In *Proceedings of the Audio Engeenir-ing Society 118th Convention*, Barcelona, Spain, May 2005.

Alain de Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

Christopher Duxbury, Mike E. Davies, and Mark B. Sandler. Complex domain onset detection for musical signals. In *Proceedings of the International Conference on Digital Audio Effects (DAFx-03)*, pages 90–93, London, UK, 2003.

Jonhatan Foote and Shingo Uchihashi. The beat spectrum: a new approach to rhythm analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2001)*, pages 881–884, Tokyo, Japan, August 2001.

Stephen Hainsworth and Malcom Macleod. Onset detection in music audio signals. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 163–166, Singapore, 2003.

Paul Masri. *Computer modeling of Sound for Transformation and Synthesis of Musical Signal*. PhD dissertation, University of Bristol, UK, 1996.

Tempo estimation and beat tracking with adaptive input selection

M. E. P. Davies and M. D. Plumbley
Centre for Digital Music
Queen Mary University of London
Mile End Road, London, E1 4NS, United Kingdom
matthew.davies@elec.qmul.ac.uk

Abstract

We present details of our submissions to the Audio Tempo Extraction and Audio Beat Tracking contests within MIREX 2006. The approach we adopt makes use of our existing beat tracking technique with a modified tempo extraction stage, and with the provision of three different onset detection functions to act as input. For each onset detection function we extract potential beat locations and then employ a confidence measure to find the most appropriate input representation for a given audio signal. The beats which yield the highest confidence are extracted as the output of the system.

Keywords: Beat tracking, Tempo extraction, onset detection, MIREX

1. Approach

In this paper we address two aspects of computational rhythmic analysis: i) finding the underlying tempo of a piece of music; and ii) the related topic of extracting beat locations. The output of each algorithm has been compared against the performance of other submitted algorithms with the MIREX 2006 Audio Tempo Extraction [1] and Audio Beat Tracking [2] contests.
A detailed description of the operation of our beat tracking system which incorporates both the extraction of tempo and beat locations may be found in [3]. Therefore within this extended abstract, we merely provide an overview of the operation of each algorithm, in particular addressing those aspects which differ from our previous approach.

1.1. Beat Tracking System

The beat tracking system [3] upon which both MIREX submissions are based can be broken down into four discrete stages:

- The transformation of the input audio signal into an onset detection function (DF).
- The extraction of the beat period by passing the autocorrelation function (ACF) of the DF through a shift-invariant comb filterbank.
- The extraction of the phase of the beats by cross-correlating the DF with an impulse train with impulses at beat period intervals.
- The use of a two-state switching model to track tempo changes and enforce contextual continuity within constant tempo regions.

1.2. Tempo Estimation

The Audio Tempo Extraction contest requires the identification of two perceptual tempi consistent with the performance of multiple human annotators and a salience value describing the strength of the primary metrical level over the secondary level. Within our submission to the MIREX 2005 Audio Tempo Extraction contest (for which the test data and annotations were identical) we extracted the single tempo which most strongly resonated with the periodicities within the input ACF, and inferred the secondary tempo and salience using a simple rule-based approach.
To provide a more robust method, we now apply a peak picking algorithm to the comb filterbank output function. Given a range of possible beat periods with corresponding peak heights, we find the $\log_2$ ratio of all combinations of peak heights, normalising and scaling the results such that a perfect 2:1 ratio will have a minimum absolute score, s_1 of zero,

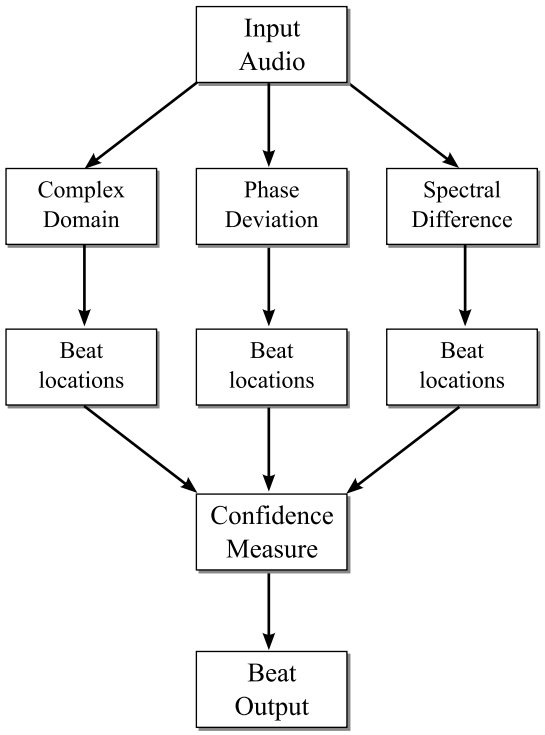


Figure 1. Overview of beat tracking system

$$s_1(i,j) = |(1 - |\log_2(\frac{P(i)}{P(j)})|)| \quad (1)$$

where $P(i)$ and $P(j)$ are the respective periodicities for peaks i and j. To each $s_1(i,j)$ we then add the weighted reciprocal of the sum of the peak heights $A(i)$ and $A(j)$ to give a second score, s_2

$$s_2(i,j) = s_2(i,j) + \frac{\delta}{A(i) + A(j)} \quad (2)$$

where δ is empirically set to 0.01. We then extract the pair of periodicities $P(i)$ and $P(j)$ with the lowest s_2 as those which represent the perceptual tempi of the input, taking the periodicity with the higher peak height as the primary metrical level and the lower as the secondary level. The normalised ratio of peak heights then provides the salience,

$$\text{salience} = \frac{\text{A(i)}}{\text{A(i)} + \text{A(j)}} \quad (3)$$

where in this example $A(i) > A(j)$.

1.3. Beat Tracking with an adaptive input

The task of beat tracking can be considered analogous to the human ability of foot-tapping in time to music. While seemingly intuitive for humans, beat tracking remains a complex task in computational rhythmic understanding. A particular failure of many approaches (for a review see [3]) is the inability to maintain equivalent performance across a wide range of musical genres.

Preliminary experiments in [4] demonstrated that the overall beat tracking performance of our system could be significantly improved by knowing a priori which of several possible onset detection function to use as input, while leaving all other aspects constant. We now extend this concept by attempting to automatically identify the onset detection function most suited to the input signal. The approach we adopt first involves the calculation of three related onset detection functions: i) complex spectral difference; ii) phase deviation; and iii) spectral difference [5].

We then calculate the beat locations for each detection function and then a use *confidence* measure to select which beats are to be taken as the output to the system. The confidence measure is related to the beat tracking evaluation function [2], which we calculate by cross-correlating an impulse train representing the predicted beat locations with each detection function, selecting the beats arising from the strongest correlation as the output. An overview of our approach is shown in fig. 1

2. Results

The results for the Audio Tempo Extraction contest are shown in Table 1 and the results for the Audio Beat Tracking Contest are in Table 2. For each table the overall *P-score* is taken to represent the performance. Further details of the competing algorithms may be found on the respective MIREX web-pages, for Tempo Extraction [1] and for Beat Tracking [2].

Contestant	P-score
Dixon	0.407
Ellis	0.401
Klapuri	0.395
Davies	**0.394**
Brossier	0.391

Table 2. Results table for Audio Beat Tracking

Results indicate that for Tempo Extraction our approach placed 2nd out of 7 entries and was 4th out of 5 entries for Beat Tracking.

Contestant	At least 1 tempo correct	Both tempi correct	P-score
Klapuri	94.29%	61.43%	0.806
Davies	**92.86%**	**45.71%**	**0.776**
Alonso 2	89.29%	43.57%	0.724
Alonso 1	85.71%	45.71%	0.693
Ellis	79.29%	42.86%	0.673
Antonopoulos	84.29%	47.86%	0.669
Brossier	78.57%	50.71%	0.628

Table 1. Results table for Audio Tempo Extraction

Acknowledgements

This research has been partially funded by EPSRC grants GR/S75802/01 and GR/S82213/01. Many thanks also to the MIREX team for overseeing the operation of the contests.

References

[1] "Audio Tempo Extraction - MIREX 2006," [Web site] 2006, [2006 Aug 31], Available: http://www.music-ir.org/mirex2006/index.php/Audio_Tempo_Extraction

[2] "Audio Beat Tracking - MIREX 2006," [Web site] 2006, [2006 Aug 31], Available: http://www.music-ir.org/mirex2006/index.php/Audio_Beat_Tracking

[3] M. E. P. Davies and M. D. Plumbley, "Context-dependent beat tracking of musical audio," *IEEE Transactions on Audio, Speech and Language Processing*,to appear 2007

[4] M. E. P. Davies and M. D. Plumbley, "Comparing mid-level representations for audio based beat tracking," in *DMRN 2005 Digital Music Research Network Summer Conference.*, 2005, pp 36-40.

[5] J. P. Bello, C. Duxbury, M. E. Davies and M. B. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Processing Letters*, vol. 11, no. 6, pp. 553-556, July, 2004

Beat Tracking with Dynamic Programming

Daniel P.W. Ellis
LabROSA, Dept. of Electrical Engineering
Columbia University, New York NY 10027 USA
dpwe@ee.columbia.edu

Abstract

There are many applications for which we would like to be able to track the 'beat' of a piece of recorded music – analogous to a listener's foot-tapping. This paper describes our beat-tracking system, which operates by first estimating a global tempo (via autocorrelation of an 'onset strength' signal), then using dynamic programming to find the best sequence of beat times through the whole piece that both places beats on moments of high 'onset strength', as well as maintaining a spacing between beats that agrees with the global tempo. This system has been submitted to the 2006 MIREX Audio Tempo Extraction and Audio Beat Tracking competitions.

Keywords: Tempo Extraction, Beat Tracking, Autocorrelation, Dynamic Programming

1. Introduction

Finding the beats in a musical recording is an interesting challenge and can form the basis of a number of applications, such as automatic accompaniment, transcription, computer-assisted audio editing, and music similarity. In this paper we describe our beat tracking system, which was in fact developed as part of our cover song detection system (since beat-synchronous features are a good way to normalize away tempo variations between different versions of a song) [1].

Evaluating systems for beat tracking (and hence tempo extraction) is complicated by the fact that different 'levels' in a hierarchy of beats may be considered as the main beat by different listeners. For MIREX, this problem has been neatly solved by collecting actual human tapping data for the test database [3]. Our system has been tuned with the 20 training samples released for the MIREX-06 tempo and beat evaluation. Each sample consists of 30 s of audio (from a range of styles and genres) along with the tap-instants of 40 different subjects who were played the samples. There are usually two different beat periods represented in the user data, where one is 2 or 3 times faster than the other. Beat trackers are evaluated by their ability to match the *entirety* of the subjective ground truth data, which means in practice choosing one tempo, and accepting that matches will not be high for subjects who choose a different tempo.

The following section describes our system, then section 3 reports performance on the evaluation data.

2. System Overview

This section describes each module in our system.

2.1. Onset Strength Signal

The first stage of processing is to convert the audio into a one-dimensional function of time at a lower sampling rate that reflects the strength of onsets (beats) at each time. We based this on the front-end on the one described in [2]. A log-magnitude 40-channel Mel-frequency spectrogram is calculated for 8 kHz downsampled mono versions of the original recording with a 32 ms window and 4 ms hop between frames. The first-order difference along time in each frequency channel is half-wave rectified (to leave only onset information) then summed across frequency. This 'onset strength' envelope is high-pass filtered with a 3 dB point at 0.01 rad/samp to remove d.c. offset (corresponding to global gain variations in the original signal, prior to the log operation).

2.2. Tempo Estimation

The onset strength for the entire signal is autocorrelated out to a maximum lag of 4 s (i.e. 1000 samples at our 4 ms sampling period). This raw autocorrelation is then scaled by a window to capture the intrinsic bias of listeners towards a particular range of tempi; in this way, the multiple peaks typical of the autocorrelation of a period signal can be resolved to a single dominant peak. Our window is a Gaussian on a log-time axis, and is characterized by its center (the BPM at which it is largest), and its half-width (the sigma of the Gaussian, in units of octaves on the BPM scale, since the axis is in fact logarithmic). We tuned these parameters by hand to give the best agreement with the subjective data provided for the MIREX competition; the best center was at 120 BPM (agreeing with perceptual results for the preferred tapping rate of subjects), and the width was 1.4 octaves (i.e. the window has fallen to 60.7% of its peak value at 323 BPM and at 44.6 BPM). The lag corresponding to the largest value in this autocorrelation was reported as the strongest tempo.

The competition also requires a second tempo, which is scored against the second-most popular tempo observed in

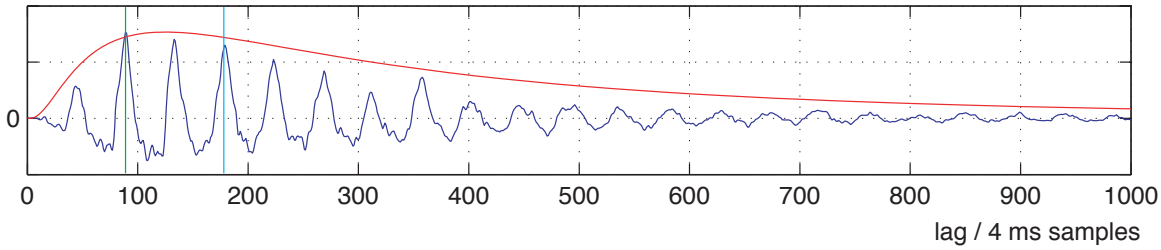


Figure 1. Autocorrelation of the 30 s Bragg excerpt. The log-time Gaussian weighting window is shown overlaid, and the primary (89 samples = 356 ms = 168.5 BPM) and secondary (178 samples = 712 ms = 84.3 BPM) periods are shown by vertical lines.

the subject studies. To find this, we searched the autocorrelation peaks closest to 0.33, 0.5, 2, and 3 times the strongest tempo. Whichever of these was largest was reported as the secondary tempo. The weight of the stronger tempo, also requested for the evaluation, was simply taken as the value of the largest autocorrelation peak divided by the sum of the peaks at both reported tempi. This value does not affect the evaluation metric as defined, so no effort was made to match it more closely to the ground-truth values in the training set.

Figure 1 shows the autocorrelation for example 2 of the training set, a 30 s excerpt from "New England" by Billy Bragg (vocals and guitar only).

2.3. Beat Tracking

The best BPM is passed to the beat tracking module, which attempts to find a sequence of beat times that all correspond to large values in the onset waveform, The onset signal is first smoothed by convolving with a Gaussian window whose half-width is 1/32 of the specified beat period. Then the best cumulative score is found for beat sequences ending at every possible time sample. This is done efficiently with dynamic programming: for each time point, a search is done over a range 0.5 to 2 beat periods into the past. The best cumulative score at each time in that window is scaled by a 'transition weight', which is another log-time gaussian, centered on the ideal time (one beat into the past), and with a width specified as a parameter of the system – a narrower width makes it harder for any beat to deviate far from the specified target period. The largest scaled value is chosen as the best predecessor beat for the current time, and added to the current onset signal value to give the best cumulative score for this time. The time of the preceding beat is also recorded. At the end of the excerpt, the best cumulative score within a couple of beats of the end is chosen, then traced back through all the preceding-time records to get the entire sequence of beats that gave rise to that best score.

In order to keep a balance between past scores and local match, the best score at the preceding beat is actually scaled by a constant a little smaller than 1 before being added to the current beat's score. This constant is a second parameter to the system: the smaller it is, the more weight is placed on achieving a good local match versus choosing a good history. This is a second parameter to the algorithm.

Figure 2 shows an example of the beats found in the first 15 s of the Bragg excerpt. The advantage of dynamic programming is that it effectively searches all possible sets of beat instants, since it is guaranteed to find the best-scoring sequence up to any point. This allows the best global beat sequence to be found, even if it involves some locally-poor matching, for instances beats that occur during silence or uninflected sustained notes.

3. Evaluation

The MIREX-06 audio tempo and beat evaluations make available 20 training excerpts, for which ground-truth tempi and beat times are given, as described above. The principal evaluation metrics are also defined. Thus we were able to optimize some of the tuning parameters of our system to maximize performance on this set.

For tempo extraction, the middle tempo of 120 BPM and the window width of 1.4 octaves were chosen this way. With these settings, using the weighted both-tempo matching score defined for the evaluation (which rewards identifying either or both of the two tempo levels in proportion to their observed prevalence among subjects) our system achieves a score of 77% correct.

For beat tracking, we tuned the transition window width and forgetting factor to maximize the evaluation score on the training data. The optimal window width weighted the extreme values in the preceding beat window (at 0.5 and 2 periods earlier) both at $0.17\times$ the central peak. The best forgetting factor decayed the history by a factor of 0.8 at each step. With these settings, and using the defined primary metric, our system scored 56.6% correct. Note, however, that 100% is not obtainable due to inconsistencies between the ground-truth listeners – no single beat tracker output can agree well with both of the two levels of beats typically found among the training data.

Since beats tracked at a slower tempo run the risk of being one half-cycle out of phase (i.e. picking beats 2 and 4 instead of 1 and 3), we thought that always using the faster of the two identified tempo (i.e. picking beats 1, 2, 3, and 4)

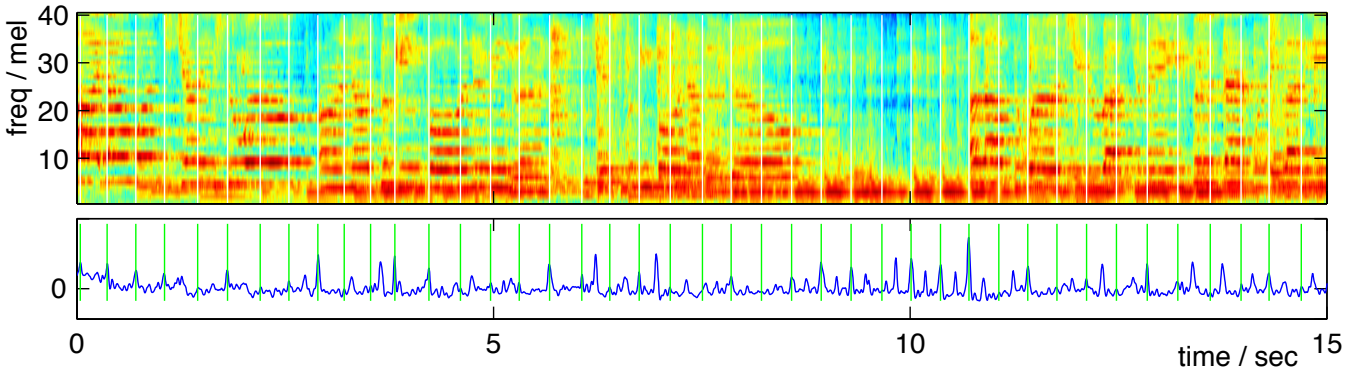


Figure 2. Excerpt showing the Mel-scale spectrogram (top pane), and the smoothed onset strength evelope (lower pane) for the first 15 s of the Bragg excerpt. Chosen beats are shown as vertical divisions. Notice the extensive syncopation (strong onsets midway between perceived beats).

might be a safer option. However, we found we got better scores by using the primary BPM value (i.e. the largest peak in the weighted correlation), whether or not it was faster than the alernative.

4. Conclusions

Using a relatively simple onset detection scheme, and assuming more-or-less fixed tempo throughout a piece, we find simple autocorrelation, suitably weighted to simulate a perceptual bias, does well at predicting perceived tempo. Beat tracking that uses dynamic programming to search all possible beat sequences does well, even when there are voids where beats have to be filled in, since the location of future as well as past beats can affect their position. Future work includes modifying the beat tracker to take account of slow but systematic changes in tempo, and perhaps a system that extracts multiple

Acknowledgments

This work was supported by the Columbia Academic Quality Fund, and by the National Science Foundation (NSF) under Grant No. IIS-0238301. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF. We are very grateful to Martin McKinney and colleagues for collecting and making available the ground-truth data for this evaluation.

References

[1] D. P. W. Ellis. Identifying 'cover songs' with beat-synchronous chroma features. In *MIREX-06 Abstracts*, 2006.

[2] T. Jehan. *Creating Music by Listening*. PhD thesis, MIT Media Lab, Cambridge, MA, 2005.

[3] M. F. McKinney and D. Moelants. Extracting the perceptual tempo from music. In *Proc. Int. Conf. on Music Info. Retr. ISMIR-04*, pages 146–149, Barcelona, 2004.

Sweepline and Recursive Geometric Algorithms for Melodic Similarity

Kjell Lemström, Niko Mikkilä, Veli Mäkinen and Esko Ukkonen
C-BRAHMS Group, Department of Computer Science
P.O.Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, FINLAND
{klemstro,mikkila,vmakinen,ukkonen}@cs.helsinki.fi

Abstract

This extended abstract gives an overview on two content-based retrieval algorithms for symbolic music, developed earlier in the C-BRAHMS group [1], that took part in the Symbolic Melodic Similarity and Query by Singing/Humming tasks of the MIREX 2006 contest [3, 4]. Given two excerpts of symbolically encoded monophonic or polyphonic music, the *query pattern* and the *target music*, the purpose of these algorithms is to find musically relevant occurrences of the query pattern within the target music.

Keywords: MIREX 2006, Melodic Similarity, Geometric Matching

1. Introduction and Background from MIREX 2005

In the previous MIREX contest organized in 2005 we submitted both a basic monophonic string-matching algorithm and the same geometric algorithm that is described in this abstract. Last year the Symbolic Melodic Similarity (SMS) task only included monophonic music from the RISM A/II collection of incipits and the results were compared to a human-generated ground truth. Both our string-matching algorithm and the more complex geometric algorithm seemed to work equally well, with the geometric algorithm performing only slightly better. One of the reasons for that performance was probably the fact that the string-matching algorithm does not use any rhythmic information at all and the geometric method is not time-scale (tempo) invariant; it requires that both the query and target melodies are played at the same speed.

This year in MIREX 2006 there were two Symbolic Melodic Similarity subtasks: a monophonic task based on the RISM collection and both exact (quantized in pitch and rhythm) and hummed queries; and a similar polyphonic task based on MIDI files harvested from the Internet [3]. We were especially interested in the polyphonic task and therefore we submitted the same geometric P3 algorithm that we submitted last year, only this time as two entries: the original one (P3) and a brute force tempo scaling version (ScaledP3) that runs the original algorithm multiple times with the pattern scaled in time by predefined constant factors and retrieves the best match across the runs. An overview of these algorithms is given in section 2.

We also intended to submit ScaledP3 to the Query by Singing/Humming (QBSH) task but after testing various approaches with the data set, we decided to use a recursive exhaustive search algorithm instead. The MIDI data available represented some challenges and the alternative would have been to process the queries with our own melody extraction method which we do not have yet. The ES algorithm is described in section 3.

2. Sweepline Algorithm

Our submission to the SMS task is based on a geometric sweepline technique that is applied on a piano-roll type representation of the musical score [2]. The intuition behind this algorithm is to slide the bar-lines representing the query over the piano-roll representation of target and to find the position that gives the maximal common shared time (see Figures 1 and 2).

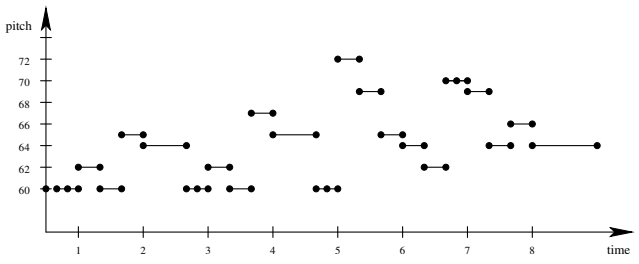


Figure 1. Query in piano-roll representation.

To this end, the piano-roll representations of the query and the target music are given to the algorithm as lexicographically ordered turning points that are calculated based on the start and end points of the bar-lines representing the query and the target.

The algorithm first populates a priority queue with two translation vectors for each turning point in the pattern. In the beginning the vectors point to first starting and ending point in the target. After this initialization the algorithm loops through all possible translation vectors between the

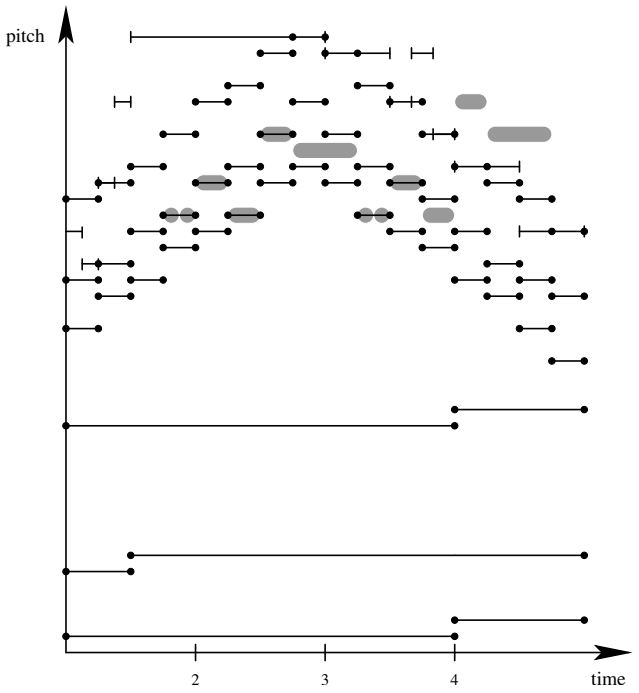


Figure 2. Target in piano-roll representation. The first twelve notes of the query in Figure 1 are shown by shading in a translated position such that the total length of the overlapping is six quarter notes.

query and the target. At each iteration the first vector in lexicographic order is retrieved from the queue and replaced by the next corresponding vector (a vector for the same turning point in the query) that has not yet been inserted into the queue.

When the algorithm iterates over translation vectors, it counts common time between the query and the target on each vertical translation level to ensure transposition invariance. The time is counted by using a linear slope that the translation vectors adjust, and the maximal overlapping is simply checked for at each iteration. Finally, normalizing the maximal overlapping by the combined length of query or target notes (whichever is smaller) results in a value that expresses the similarity of the query and the target.

The brute force pattern-scaling version of the algorithm simply uses the method described above multiple times with the notes in either the pattern or the target scaled in time. We used scaling factors 0.5, 0.667, 0.8, 1.0, 1.25, 1.5 and 2.0. The selection of these values is not based on extensive experimentation, so they are probably not the optimal factors for the task. Heuristics could perhaps be used to select a suitable scale or to at least limit the range.

The overall best match is chosen simply by comparing normalized overlap lengths of the best matches returned from the runs with each scaling factor. With such a low number of factors, good matches usually stand out from the rest. Using more scaling factors would allow smoother matching, but the algorithm execution time would also quickly increase unbearably. This brute force method should give slightly better results than the original one whenever note timing in the query is not expected to be the same as timing in the potential matches, just like in the SMS and QBSH tasks. Of course, this will not help much with tempo changes that may occur within hummed queries.

P3 runs in $O(mn \log m)$ time where m and n denote the number of musical events (notes) in query and target, respectively. The scaled version of the algorithm increases the execution time by a constant factor of 7.

3. Recursive Algorithm

The ES algorithm that we submitted to the QBSH task also uses a piano-roll representation of music, but it does not maximize the overlapping in the same way as P3 does. Instead it performs an exhaustive depth-first search trying to scale the pattern note-by-note to 'fit' the target song, with costs applied to local time-scaling, note duration changes and pitch-shifting. Clearly irrelevant branches are cut with simple heuristics while searching, which keeps the average running time in an usable range, although the worst case time complexity is $O(n^m)$.

First the algorithm divides the piano-roll representation into tiles that have a height of one MIDI pitch level and a width chosen so that there would not be many notes in one tile. A pointer to the first note that starts in each tile is stored to a table and subsequent notes at the same pitch level are linked together. This tile table is used for hashing: quickly finding notes that start within a specific range in the target music. The tile table size is a compromise between quick lookups and space consumption. We used a static tile length of 100 ms but a more optimal value for each pitch level could be chosen by scanning through the target music.

Next the target music is searched for the best occurrance of the pattern by checking recursively for a match at each note in the music, starting with each note in the pattern. For note T_i in the target music and note P_j in the pattern, the recursive check is started by calculating the expected pitch and starting time interval of the next matching note, or multiple notes when gaps are allowed in the matches. All potentially matching notes are looked up from the tile table, match score is updated and the same check is executed recursively for each of the notes, starting at the next position in the pattern.

The most adjustable part of the algorithm is the way how the following potentially matching notes are picked and scored at each recursion level. This procedure can be weighted by the already matched part of the pattern or it can be done independently for each position. To calculate the expected pitch level, we simply take the pitch interval between P_{j+1} and P_j, and add that to the pitch of T_i. Similarly, the difference between start times of the consecutive notes in the

pattern is scaled in proportion to previously matched notes and added to the start time of T_i.

Pitch and tempo shifts are handled by retrieving all notes within a certain range from the expected position: $\pm$ 2 pitch levels and the delta time scaled by 0.5 – 2.0. Notes that start outside this area are not considered further at that point of recursion. Each melody line that continues from the retrieved notes is checked recursively and the match scores are updated. Notes that are closest to the expected note position and have similar duration to the corresponding note in the pattern receive the best score. This is done by multiplying together factors derived from all these differences. 1.0 is a perfect match of a note and 0 is a complete mismatch. Therefore the whole pattern has a maximal score of $m - 1$, and match scores are normalized by dividing them with this value.

4. Results and Analysis

In this section we analyze results from the two MIREX 2006 tasks that our algorithms competed in. More information about the tasks and evaluation methods can be found through task descriptions in the MIREX Wiki. Abstracts from all the participants are published in the result pages. [3, 4]

4.1. Symbolic Melodic Similarity

Our algorithms were at the tail of the competition in all the Symbolic Melodic Similarity tasks (see Figures 3, 4 and 5). Comparing the results of our two P3 variations and looking at the results from last year might suggest that there was a problem within our implementation this year that did not surface as strongly last time. The difference may certainly come from different task setup, but since the RISM A/II collection was used both times and other algorithms performed much better, we might have mistuned something.

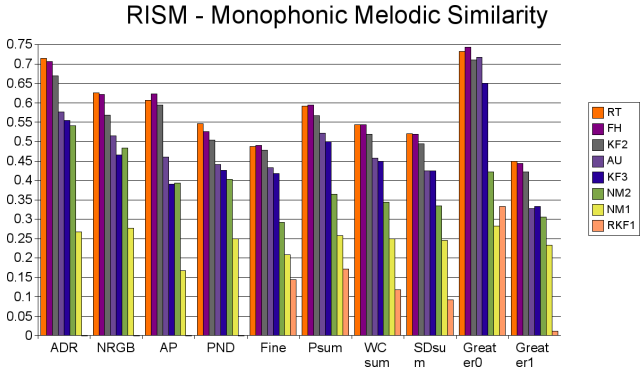


Figure 3. Summary of MIREX 2006 Symbolic Melodic Similarity Monophonic (RISM) task results. The teams and algorithms are FH: Pascal Ferraro and Pierre Hanna, NM1: our original P3 algorithm, NM2: scaled P3, RT: Rainer Typke, Frans Wiering and Remco C. Veltkamp, KF: Klaus Frieler and AU: Alexandra Uitdenbogerd. We thank Rainer Typke for providing these graphs.

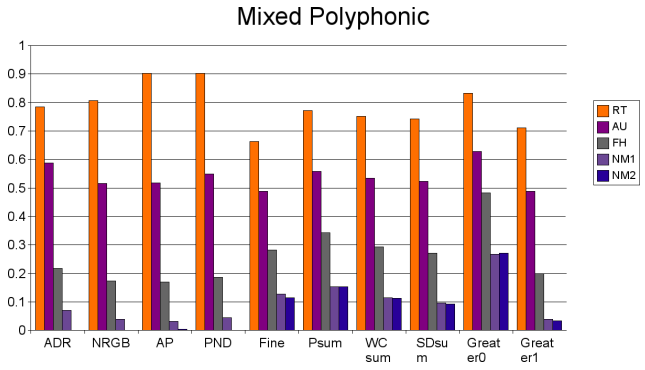


Figure 4. SMS Mixed Polyphonic task results.

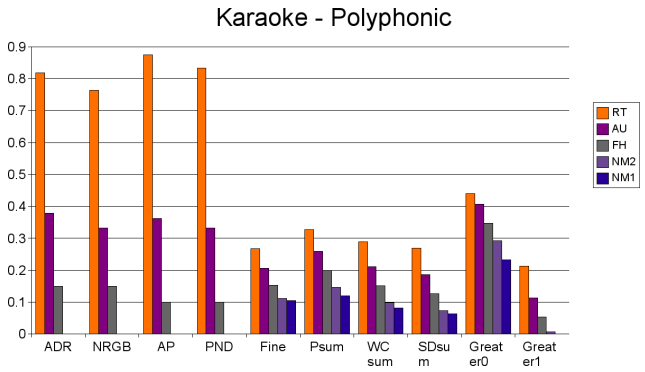


Figure 5. SMS Karaoke Polyphonic task results.

One possible source of problems with the RISM data is the normalization of common time in a match which in our current implementation allows incipits with only a few notes to match the queries better than longer incipits that humans would consider best matches. We first used this normalization scheme last year, when it seemed to work better than simply dividing the common time by pattern duration. After the MIREX 2006 results were published, we compared these two normalization approaches with the evaluation data, and there is a clear difference in favor of the simple normalization by pattern duration. This only affects the results when matching short queries in a database of potentially even shorter incipits, such as the RISM collection. For straight similarity comparison of two songs, the normalization we used is a valid approach.

The poor performance of our submissions in the polyphonic SMS tasks (Figures 4 and 5) was most likely caused by our decision to ignore all track and channel information available in the MIDI files in the algorithm implementation. With the track information in place, the task can be reduced to almost monophonic similarity comparison where simple dynamic programming methods work well. Of course it could be argued that in the real world the track information is usually available, but then again, there are cases where

	XW1	XW2	RJ	RL	**NM**	CS1	RT2	CS3	AU2	CS2	AU1	RT1
Task I (MRR)	0.926	0.900	0.883	0.800	**0.688**	0.568	0.390	0.348	0.288	0.283	0.205	0.196
Task II (Mean Prec.)	-	-	0.926	-	**0.722**	0.587	0.401	0.415	0.238	0.649	0.163	0.468

Table 1. Query by Singing/Humming task results. We refer the reader to the task description for more information about the test collections and the evaluation method [4]. The teams are AU: Alexandra Uitdenbogerd, CS: Christian Sailer, FH: Pascal Ferraro and Pierre Hanna, NM: us with the ES algorithm, RJ: J.-S. Roger Jang, Nien-Jung Lee and Chao-Ling Hsu, RL: Ernesto Lopez and Martin Rocamora, RT: Rainer Typke, Frans Wiering and Remco C. Veltkamp and XW: Xiao Wu and Ming Li.

the searched pattern is not constrained to one track – here it was in majority of the relevant files considering the queries used. There are also sources such as automatic polyphonic transcriptions of audio recordings, where the instrument information may not even be available at all.

The P3 algorithm is computationally efficient but it does not use any indexing, so searching large databases can be slow. In MIREX 2005, parsing the short MIDI incipits probably took most of the time the algorithm was measured to run for and therefore the execution time of the actual algorithm was not clear. Overall it was the fastest one along with our DP algorithm. This year there were much longer pieces of music in the polyphonic task and the indexing and query execution times were separated for those algorithms that support indexing. It is clear that an indexing scheme is necessary in most applications that require searching massive music collections, even though fast on-line algorithms have their uses as well. Indexing polyphonic music fully and efficiently is still a big challenge.

4.2. Query by Singing/Humming

The ES algorithm that we submitted to the QBSH task is an experimental brute force implementation of ideas that may be useful in symbolic melodic similarity in general when implemented more efficiently. Currently it is too slow for polyphonic matching with patterns longer than 10-20 notes, although it could be used for n-gram searches or index construction. The results from QBSH (see Table 1) are quite encouraging since this algorithm performed fairly well even with the imperfect MIDI queries supplied, while the other purely symbolic algorithms (AU, FH and RT) had more difficulties with them.

Overall, these two MIREX tasks were a teaching experience on music information retrieval from large databases. We will have to consider richer approaches for weighing the potentially matching notes, like the ES algorithm does, but hopefully with much lower time complexity. If that proves to be impossible or difficult, we could use a multi-level approach where a simple and fast algorithm retrieves a long list of potential matches and then a slower algorithm filters those results to pick the best matches. A similar approach might be applicable to polyphonic indexing.

5. Acknowledgments

Our best thanks to IMIRSEL for organizing MIREX 2006, to Rainer Typke and Anna Pienimäki for their work on the Symbolic Melodic Similarity task and to J.-S. Roger Jang and J. Stephen Downie for the Query by Singing/Humming task. We much appreciate the opportunity to compare various music information retrieval methods in a controlled environment and hope that MIREX will continue for years to come.

References

[1] K. Lemström, V. Mäkinen, A. Pienimäki, M. Turkia and E. Ukkonen, "The C-BRAHMS Project," in *ISMIR 2003 Fourth Int. Conf. on Music Inf. Retr. Proc.*, Oct. 2003, pp 237-238, See: http://www.cs.helsinki.fi/group/cbrahms/

[2] E. Ukkonen, K. Lemström and V. Mäkinen, "Sweepline the Music!." *Computer Science in Perspective — Essays dedicated to Thomas Ottmann*, vol 2598, pp 330-342, Springer-Verlag, 2003

[3] R. Typke and Anna Pienimäki, "Symbolic Melodic Similarity," [Web site] 2006, Available: http://www.music-ir.org/mirex2006/index.php/Symbolic_Melodic_Similarity

[4] J. S. Downie, J.-S. Roger Jang and R. Typke, "Query by Singing/Humming," [Web site] 2006, Available: http://www.music-ir.org/mirex2006/index.php/QBSH:_Query-by-Singing/Humming

MIREX 2006 Audio Music Similarity and Retrieval Submission (1st version)

Thomas Lidy **Andreas Rauber**
Vienna University of Technology
Department of Software Technology and Interactive Systems
Favoritenstrasse 9-11/188, A-1040 Vienna, Austria
lidy,rauber@ifs.tuwien.ac.at

Abstract

This paper describes our approach to the MIREX 2006 Audio Music Similarity and Retrieval evaluation task. We use a new implementation of the Statistical Spectrum Descriptor feature extractor and compute a distance matrix for similarity retrieval in Matlab.

1. Audio Feature Extraction

We have completely re-implemented the feature sets used in the MIREX 2005 Audio Genre Classification in Java, and thus participate in this years evaluation exchange with the new Java implementation. The Java feature extraction can now extract Rhythm Patterns features, Statistical Spectrum Descriptors and Rhythm Histograms from au, wav and mp3 files, with either 11, 22 or 44 kHz. It enables the recursion of arbitrary directories containing any number of audio files and also the mixed usage of different file formats and even sampling rates within one feature extraction process. (As the MIREX 2006 Audio Music Similarity and Retrieval task however uses only wav format as input, we omit the submission of the library for reading mp3 files). It should be more robust than the Matlab implementation and has also been tested with silence in audio. If an error occurs with some file, the program outputs a meaningful message, skips the file and continues with the next audio file. (In this case that audio file will not be included in the distance matrix).

From pre-liminary tests we found that the Statistical Spectrum Descriptors (SSD) deliver reasonable results when employed for retrieval with similarity rankings. As the computational cost for extracting SSD is also lower than as for both Rhythm Patterns and Rhythm Histograms, and the dimensionality is lower than the one of Rhythm Patterns (which reduces the cost for distance calculations), we decided to base our submission solely on the SSD features.

The following paragraphs describe the implementation of the SSD feature extraction. For the other feature sets we refer to [1].

Statistical Spectrum Descriptors are derived from a psychoacoustically transformed spectrogram and comprise several statistical moments, which are intended to describe fluctuations on a number of critical frequency bands.

Before the calculation of the features, the audio file is segmented into chunks of approx. 5.9 seconds. The first and the last segment are skipped, from the remaining segments, every 3rd one is processed. An SSD feature vector is the calculated for every segment.

First the spectrogram is computed using the short time Fast Fourier Transform (STFT), with a window size of 1024 (@ 44 kHz), 512 (@ 22 kHz) or 256 (@ 11 kHz), respectively, and 50 % overlap.

The Bark scale, a perceptual scale which groups frequencies to critical bands according to perceptive pitch regions, is applied to the spectrogram. The Bark scale is defined by limits within the audio frequency region, partitioning the frequency spectrum into 24 critical bands. Using 22 kHz audio as input, the number of bands is 23 only. Frequency bands from the spectrogram are aggregated to the bands defined by the Bark scale [2].

The Bark scale spectrogram is then transformed into the decibel scale. Subsequently, the values are transformed into Sone values, in order to approximate the loudness sensation of the human auditory system.

From this representation of a segment's spectrogram a number of statistical moments is computed, in order to describe fluctuations within the critical bands: mean, median, variance, skewness, kurtosis, min- and max-value are computed for each critical band, forming the SSD feature set. The feature vector for an audio file is then constructed as the median of the SSD features of the extracted file segments.

2. Distance Matrix Calculation

As the the implementation of similarity ranking and according distance measures in our Java software has not yet been completed, we run the distance matrix calculation for the Audio Music Similarity and Retrieval task in Matlab. The feature vector file written by the Java Audio Feauture Extraction program is read into matlab. The distance matrix is calculated using the cityblock metric. The Matlab function then writes the distance matrix to the file format specified

on the MIREX task Wiki. Several types of evaluations are to be carried out from this distance matrix.

3. Performance

The Audio Feature Extraction roughly estimated takes about 17 hours on a 3.0 Ghz CPU for 5000 wav files. Distance Matrix Computation is estimated to take about 15 to 30 minutes.

Disk space requirements:

- Feature vector file: 15 MB
- Distance Matrix: 260 MB

NB: The Audio Feature Extractor writes quite a lot of useful logging to standard output, e.g. the audio file processed, the estimated remaining time and also which segment of the file is currently processed.

References

[1] T. Lidy and A. Rauber, "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification," in *Proceedings of the Sixth International Conference on Music Information Retrieval*, London, UK, September 11-15 2005, pp. 34–41.

[2] E. Zwicker and H. Fastl, *Psychoacoustics - Facts and Models*, ser. Springer Series of Information Sciences. Berlin: Springer, 1999, vol. 22.

Post Processing Music Similarity Computations

Tim Pohle
Department of Computational Perception
Johannes Kepler University Linz, Austria
tim.pohle@jku.at

Abstract

Today, among the best-performing algorithms for music similarity computations are algorithms based on Mel Frequency Cepstrum Coefficients (MFCCs). In these algorithms, each music track is modelled as a Gaussian Mixture Model (GMM) of MFCCs. The similarity between two tracks is computed by comparing their GMMs. As pointed out in [1, 2, 3], the distance space obtained this way has some undesirable properties. In this MIREX'06 submission, a technique has been implemented that aims to correct such anomalies to a certain extent [1]. The described algorithm ranked second (out of six) in the MIREX evaluation based on human listeners (note that the differences between the top-five ranked algorithms are not statistically significant). There is indication that it works better for artist identification than the other submitted algorithms.

1. Feature Extraction and Basic Distance Computation

The basic feature extraction process is quite similar to the one in [5]. It was chosen because its good tradeoff between runtime and quality, and because algorithms based on related techniques yielded good results in MIREX'05.

- The input wave files (22.050 Hz sampling rate, mono) are divided into frames of 512 samples length, with 256 samples overlap, disregarding the first and last 30 seconds.
- The number of frames corresponding to 2 minutes (i.e. 20.672 frames) are used for feature extraction. In the submitted algorithm, these frames are not chosen to be consecutive. Instead, the length of the wave data is divided into 20.672 fragments of equal length, and from each of those fragments, randomly 512 consecutive samples are chosen for feature extraction. By randomly choosing the frames possible aliasing effects with respect to the track's meter are reduced. It seems that this approach yields better results than choosing the frames in a fully random manner, or taking all frames from the two minutes in the middle of the track.
- From the chosen frames, 25 MFCCs are computed.
- A song is represented as the overall mean of the MFCCs, and the full covariance matrix.

The feature extraction process was implemented using the MA-Toolbox ([6]). Two songs are compared by the Kullback-Leiber (KL) distance. If the inverse of a song's covariance matrix can not be found, it is assumed that it is dissimilar to all other songs.

One drawback of this technique is that it does not take into consideration the temporal order of frames, thus aspects related to time are not modelled. An approach to add time-dependent features is propsed in [2]. However, the version used here it is a good starting point for the post-processing step described in the next section.

2. Post Processing

As pointed out in [1, 2, 3], the distance space obtained with such an algorithm has some undesired properties. Some tracks may be very similar to many other tracks (so-called "hubs" [3], e.g. in a collection containing about 2.500 tracks, one track may appear in the ten nearest neighbours of 250 other tracks). Also, there may be tracks that are *not* similar to other tracks. Reducing the effects of these properties may improve the quality of the algorithm's output. One could think of various ways to approach this, including those described below.

2.1. Calculations on the Distance Matrix

After computing the distance matrix of all tracks in a collection, all values in a column and all values in a row are divided by the distance of the (e.g.) 25th nearest neighbour of the track that corresponds to the index of this row or column, respectively. On in-house collections, this typically improved the leave-one-out 5 nearest neighbour (NN) genre classification by one or more percent points, depending on the collection.

In the MIREX'06 audio similarity contest, it is not allowed to use such knowledge about the whole collection for which the distances should be computed (the *test collection*). Thus, it would be necessary to provide the models of another collection (the *reference collection*), that serves for reference during the (pairwise) distance computations of the contest.

As it turned out in preliminary experiments, using a reference collection for this approach did not work satisfactorily

[1] For more detailed evaluations, please refer to [4]

in all cases. Thus, no further investigations into this directions were made[2].

2.2. Counting the Number of Appearances in k-NN Sets

Another possible approach is to determine for each piece A of the test collection a measure which is called k-*occurrences*. For calculating this, it is assumed that A is part of the reference collection. Count how often it appears in the set of k nearest neighbours (e.g. $k = 10$) of tracks in the reference collection. The k-occurrence may then be used to either filter out those pieces with a high k-occurrence, or to accordingly modify the distances of track A to other tracks in the test collection. However, this approach has not been tested yet because the approach described in the next section showed to be effective in preliminary experiments. Thus, an evaluation of this approach is left as future work[2].

2.3. Proximity Verification

The basic idea behind this approach is to replace the absolute distance (obtained by computing the KL distance) by a relative distance based on the ranking of tracks. Thus, the divergence of track A and B, denoted $D(A, B)$, is k, where B is the k nearest neighbour of A (i.e. there are $D(A, B)-1$ other tracks in the collection that are more similar to A than B, measured by the KL distance).

As in general $D(A, B) \neq D(B, A)$, a symmetric distance measure is obtained by defining

$$D_{PV} = D(A, B) + D(B, A)$$

This approach is called *proximity verification* here, as D_{PV} has a low value only if both A is a close neighbour of B *and* B is a close neighbour of A. Obviously, the average value of the divergence between track A and all tracks T_i in the collection $\frac{1}{N}\sum_{i=1}^{N} D(A, T_i)$ is the same, regardless if a track A is determined by initial similarity algorithm as being similar to many other tracks, or if it even is regarded as being dissimilar to all other tracks. Thus, these effects are reduced.

Again, in the MIREX contest, as it is not allowed to use knowledge about the test collection, $D_{PV}(A, B)$ has to be determined on the models of a reference collection, with the models of tracks A and B being the only information available from the test collection. In the submitted implementation, this results in values for $D(A, B)$ that are usually not integers, as the rank of B is interpolated.

2.3.1. Preliminary Evaluation

Preliminary experiments on in-house test collections and on the ISMIR'04 Genre Classification Contest Training Collection indicate that this approach also is beneficial when using a reference collection for post processing instead of using the test collection itself. Another tendency seems to be that merits are larger when using a larger reference collection. Examples of evaluation results with the largest possible reference collection consisting of all available tracks (more than 8.000, including the tracks in the test collection, and some tracks multiple times) are given below[3]. The test collection consists of 2447 tracks from 22 genres.

Tables 1 and 2 show that the percentage of the closest tracks that are in the same genre is improved by applying proximity verification. These results are quite promising; however, on the ISMIR'04 Genre Classification Contest Training Collection, the corresponding 5-NN value was only improved by approximately 1.4% before artist filtering.

No Artist Filter	5	10	20	50
Basic	67.9%	60.8%	51.8%	39.1%
ProxiVeri	73.2%	66.5%	56.1%	42.8%

Table 1. Percentage of closest n tracks of each track that are in the same genre for $n = \{5, 10, 20, 50\}$, *without* artist filter. $Basic$ is the basic algorithm described in Section 1, $ProxiVeri$ is the same algorithm with additional proximity verification.

With Artist Filter	5	10	20	50
Basic	28.6%	26.9%	25.0%	21.3%
ProxiVeri	31.2%	29.8%	27.8%	24.1%

Table 2. Percentage of closest n tracks of each track that are in the same genre for $n = \{5, 10, 20, 50\}$, *with* artist filter. Same algorithms as in Table 1.

Figure 1 indicates that proximity verification has a positive effect both on the number of tracks that are considered as being similar to many other tracks, and on the number of tracks that are similar to only few other tracks in the collection. Also, there is a positive effect on the number of track triples where the triangle inequality is fulfilled (e.g. the number of track triples where it is not fulfilled dropped from about 41% to about 32%).

3. Conclusions

The main motivations for submitting this algorithm to MIREX are to compare the properties of the resulting similarity space to the other submissions, and also to get feedback about how human evaluators assess its performance. Future work includes an in-depth evaluation of the proposed post processing approaches, and – most importantly – their application to other feature extraction routines and distance measures, most notably such that model time aspects of the music signal.

[2] In the meantime, further work is available [4].

[3] More detailed evaluations are future work. In particular, evaluations with reference collections that are smaller than the test collection are important.

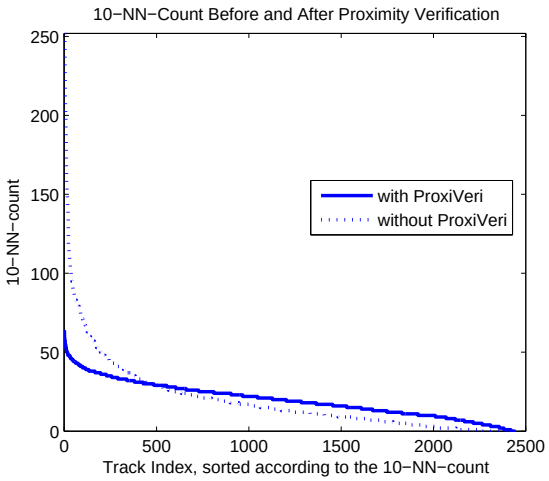


Figure 1. 10-occurrences before and after proximity verification. The y-axis is cut off at 252, corresponding to the second highest value before proximity verification. The highest value is 359. The highest value after proximity verification is 64.

4. Comments on the MIREX results

This final section contains a brief discussion of the MIREX results. The most important performance measure is the listening test, as the other measures do not directly take into account how the songs that were rated "most similar" actually sound.

4.1. Listening Test Results

The algorithm described in this abstract ("TP") ranked second in the listening tests. However, from the six submitted algorithms, the differences of the top five ranked algorithms were not statistically significant. This was the first MIREX AudioSim with a listening test, so there were no previous results that could be used to design the experimental setup. Knowing that the state-of-the-art algorithm have close scores in this listening test, for future evaluations a modified setup (e.g. more queries, a more diverse music collection) could be considered to improve the significance of the results.

4.2. Distance Matrix Statistics

Several metrics were calculated on the distance matrices that were produced by each of the submitted algorithms. Many of them use the genre labels of the songs. Unfortunately, the number of tracks per genre were very skewed, and the genre labels probably are not assigned very carefully. For example, Britney Spears was classified as Rock, and the music from the genre Rap & Hip-Hop and Rock constituted more than 70% of the collection. Thus, the results of these metrics should be regarded very cautiously.

In micro-averaged 5-NN genre classification, TP ranked first when no artist filter was used. If the genre labels are considered near noise, then this result may indicate that TP is better suited for artist identification than the other submissions, which is consistent with the results of last year's MIREX, and the observation that TP produced the lowest Artist / Genre ratio (i.e., the ratio of the distance between tracks of the same artist and the distance of tracks of the same genre).

4.2.1. Always Similar

One of the motivations for using proximity verification was to remove hubs. To my knowledge, only one of the other submitted algorithms (G1C) makes use of a technique that is known to potentially produce hubs. Depending on if the cover song tracks were also considered for calculating the k-occurrences, the statistics were different. When they are considered, the highest 50-occurrence of G1C is 2058, indicating the presence of a hub. It turns out that when not regarding the cover song tracks, there seems to be no hub for G1C. In the latter case, the highest 50-occurrence of G1C is 434, which is within the range of the other algorithms. The corresponding values of TP were 557 and 554, respectively. These values seem acceptable for non-hub tracks, although for a final decision, it would be necessary to listen to the tracks with the highest k-occurrences.

5. Acknowledgements

This research is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project number L112-N04, and by the EU 6th FP project S2S2 ("Sound to Sense, Sense to Sound", IST-2004-03773).

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*, vol. 1, no. 1, 2004.

[2] Elias Pampalk, *Computational Models of Music Similarity and their Application in Music Information Retrieval*, Ph.D. thesis, Technische Universität Wien, 2006.

[3] J.-J. Aucouturier and F. Pachet, "A scale-free distribution of false positives for a large class of audio similarity measures," 2006, Submitted.

[4] Tim Pohle, Peter Knees, Markus Schedl, and Gerhard Widmer, "Automatically Adapting the Structure of Audio Similarity Spaces," in *Proc. 1^{st} Workshop on Learning the Semantics of Audio Signals (LSAS)*, Athens, Greece, December 2006.

[5] Michael Mandel and Dan Ellis, "Song-Level Features and Support Vector Machines for Music Classification," in *Proc. International Symposium on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.

[6] Elias Pampalk, "A Matlab Toolbox to Compute Music Similarity From Audio," in *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, October 10-14 2004.

MIREX SYMBOLIC MELODIC SIMILARITY AND QUERY BY SINGING/HUMMING

Rainer Typke
Universiteit Utrecht
Padualaan 14
3584 CH Utrecht, Netherlands
rainer.typke@musipedia.org

Frans Wiering
Universiteit Utrecht
Padualaan 14
3584 CH Utrecht, Netherlands
frans.wiering@cs.uu.nl

Remco C. Veltkamp
Universiteit Utrecht
Padualaan 14
3584 CH Utrecht, Netherlands
remco.veltkamp@cs.uu.nl

Abstract

This submission to the Music Information Retrieval Evaluation eXchange in the Symbolic Melodic Similarity task uses ideas from the system that used the Earth Mover's Distance (EMD) in MIREX 2005. The total weight sums are normalized before applying the EMD, which makes it possible to use a vantage index. A novel way of segmenting is used. Response times are shortened from 14 hours for searching 581 short monophonic incipits to 3 seconds for searching 1000 complete polyphonic pieces of music. This speedup made it possible to put more effort into searching accurately by searching multiple segment sizes at the same time. Therefore, the new method should not only be faster but also more effective.

We submit this algorithm not only for the Symbolic Melodic Similarity task, but also for Query by Singing/Humming. For the latter, we rely on the MIDI files provided by J.-S. Roger Jang (張智星) instead of splitting the given pitch vectors into notes ourselves or working with the wave files.

Keywords: MIREX, symbolic melodic similarity, query by humming/singing.

1. Tasks

At the MIREX competition [1], algorithms from different researchers are compared by letting them solve the same tasks, using the same data. This extended abstract describes a submission to two out of the nine tasks at MIREX 2006.

1.1. Symbolic Melodic Similarity

The task is to retrieve MIDI files that contain material which is melodically similar to a given MIDI query. Half the queries are quantized in rhythm and pitch, and half are only quantized in pitch but not in rhythm. The latter half was created by singing melodies.

[1] MIREX 2006: http://www.music-ir.org/mirex2006

There are three subtasks that differ mainly in the collection of data to be searched:

- Approximately 16,000 incipits from the UK subset of the RISM collection, almost exclusively monophonic. Six queries (three of them quantized).
- 1000 polyphonic Karaoke files. Five queries (two of them quantized; the three sung queries include two versions of the same melody).
- 10,000 randomly chosen MIDI files that were harvested from the Web, most of them polyphonic. Six queries (three of them quantized).

Due to the size of the collections, no ground truth was known in advance. From every participating algorithm, the top ten matches are put into a pool, and human graders judge the relevance.

1.2. Query by Singing/Humming

The Query by Singing/Humming task is split into two subtasks which are strongly influenced by the character of available data. For a collection of 48 MIDI files containing quantized melodies, 2719 sung versions are available as Wave files. For every recording, a pitch vector and a MIDI file were derived. Participants can choose whether to use the audio, pitch vector or MIDI files for answering queries.

1.2.1. Subtask 1: Known-Item Retrieval Task

Based on Prof. Jang's original idea to test for the ability to find the "ground-truth" needle in a collection "haystack".

- Test database: 48 ground-truth MIDIs + $\sim$ 2000 Essen Collection MIDI noise files.
- Queries: 2719 sung queries.
- Evaluation: Mean Reciprocal rank, over top X returns, of the "know-item" ground-truth file for each sung query

1.2.2. Subtask 2: Variants Retrieval Task

Based on Prof. Downie's idea that queries are variants of ground-truth.

- Test database: 48 ground truth MIDIs + $\sim$ 2000 Essen MIDI noise files + 2719 sung queries.
- Queries: 2719 sung queries + 48 ground truth MIDI files.
- Evaluation: Classic precision and recall over X top returns.

2. Indexing

2.1. Splitting polyphonic files into voices

As a preparation for indexing, every MIDI file is split into channels and tracks. It is assumed that every voice is stored in either its own track or channel. In a second step, a skyline algorithm is applied to make each of these extracted voices monophonic.

Even in cases where a voice jumps back and forth between channels or tracks, it can still be searched as long as it stays in one channel or track for at least the minimum segment length (see the next section for a description of segmenting).

2.2. Segmenting

Depending on the collection to be searched, the monophonic voices are split into overlapping segments of varying lengths. If the index size is not a limiting factor (because the collection size is sufficiently small or the computer to be used has enough memory), the segment sizes vary from 5 to 16 consecutive notes. For the collection of 10,000 MIDI files, space on the testing machines was so tight that we could only create segments of sizes 5, 6, and 7. At every note in the voice, a segment of every size begins (unless there are fewer notes left in the piece than necessary for creating a segment of the desired length).

Note that in both cases, the space complexity is $O(N)$, where N is the total number of notes in all pieces to be indexed.

2.3. Vantage indexing

For every segment, the distance to each of a small, fixed set of vantage objects [3] is calculated. As distance measure, the "Proportional Transportation Distance" [5] is used (this is the "Earth Mover's Distance", preceded by a weight normalization such that both weighted point sets have the same total weight).

Later, to answer queries with the vantage index, the distances between the query and each vantage object are calculated, and objects with similar distances to the vantage objects are retrieved from the database. This can be done efficiently with range queries that are supported by B-trees, and it does not involve any expensive EMD calculations except for the comparisons of the query with the small, fixed set of vantage objects.

3. Searching

3.1. Searching an index with many segment lengths

If the index contains segments of length 5 to 16, the query is truncated to 16 notes (if it is longer), and all segments are searched for this possibly truncated query using the vantage index [3]. By not only searching segments with the same length as the query, added or dropped notes, as well as grace notes and other embellishments do not necessarily lead to mismatches but just increase the distance a bit.

In a second step, for the top 50 returned items, the real distance is calculated instead of the estimate that is based on the vantage index. Finally, the top 10 items demanded in the task description are returned.

3.2. Searching an index with just three different segment lengths

If the index only contains segments of length 5, 6, and 7, the query is cut into segments of length 6. For each query segment, the vantage index is searched. Again, for the top 50 returned segments, the real distances are calculated (for each query segment). Finally, these partial results are combined as described in [5]. That is, an optimum combination of query segments is found that match a piece in the same relative positions as their positions within the query, such that the segments' average distance is minimized and the coverage of the query is maximized.

3.3. Considerations for matching documents that are shorter than the query

For the RISM UK collection of musical incipits, one might want to retrieve documents that are shorter than the query, while this case is very unlikely for a collection consisting of complete pieces. To support this possibility for the RISM subtask, even though the incipits are split into segments of 5 to 16 consecutive notes, the query is not just capped at 16 notes as described above, but split into segments of varying sizes from 5 consecutive notes up to either 16 or the length of the query, whatever is lower. That way, shorter incipits can be matched to parts of the query, but the whole query can still be matched to longer incipits in one comparison.

Currently, perfect matches of a whole incipit that is shorter than the query lead to a distance of zero, and so do perfect matches of the whole query with parts of a longer incipit. Assigning different distances (a lower distance to the latter case, where more notes match), would be an improvement, but one would need to investigate how big such a difference should be in order to agree with human ideas of similarity.

3.4. The Query by Singing/Humming task

We submit our algorithm not only for Symbolic Melodic Similarity but also for the Query by Singing/Humming task. However, we do not analyze the wave or pitch vector files, but instead work only with the MIDI files provided by J.-S. Roger Jang along with the collection of queries. By doing

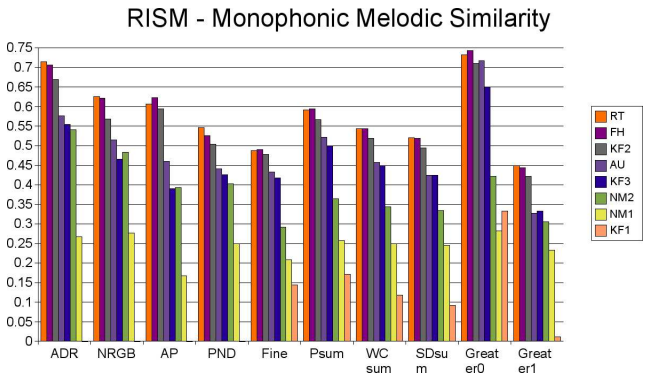


Figure 1. Task I: RISM Overall Summary. See Section 4.1.2 for an explanation of the measures. The methods are: RT - the method described in this paper; FH - an editing distance for quotiented trees by Pascal Ferraro and Pierre Hanna [2], KF is a hybrid distance measure by Klaus Frieler and Daniel Müllensiefen, AU is Alexandra Uitdenbogerd's Start-Match Alignment technique, and NM is the geometric "P3" algorithm by Kjell Lemström, Niko Mikkilä, Veli Mäkinen and Esko Ukkonen. For a more detailed description of these methods, see the MIREX abstracts, available from http://www.music-ir.org/mirex2006/index.php/Symbolic_Melodic_Similarity_Results

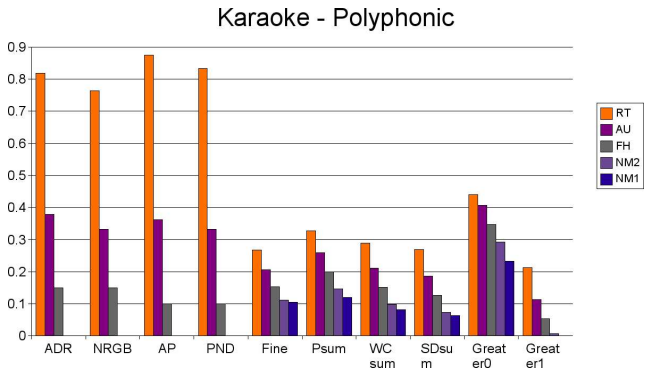


Figure 2. Task IIa: Karaoke Overall Summary

so, we avoid the need to change the algorithm at all, but any error in his conversion from Wave to MIDI will reduce our performance.

For this task, we submit the algorithm with both indexing variants – with segments of lengths 5 to 16 and with segments of lengths 5, 6, and 7. We treat the queries the same way as for the Symbolic Melodic Similarity task, that is, if we have many segment lengths, we cut the query at 16 notes, while for just three different segment lengths, we segment the query into segments of length 6. Since it is known that in Jang's collection, the queries match the database items only at the beginning, we index only the first 25 notes of every item.

The variant with fewer segment lengths needs less space for the index (only three segments start at every note instead of eleven), but requires more computing time for answering queries since there are multiple segment searches, and their results need to be consolidated into one overall result.

4. Results, Analysis

4.1. Symbolic Melodic Similarity

4.1.1. Building a ranked list from relevance scores

The raw ground truth data consisted of a rough and a fine relevance score for every item that was returned by an algorithm. For the rough score, a scale of "very similar", "somewhat similar", and "not similar" was used, while the fine score was just a number between 0 and 10. For the polyphonic tasks, where algorithms returned excerpts of MIDI files but not whole MIDI files, separate relevance scores were collected for each excerpt, even if there were multiple excerpts from the same MIDI file.

For some measures, an ordered list of relevant items is necessary. These ordered lists were created as follows from the collected relevance scores:

- Calculate average scores for every MIDI file; these averages were taken from three human graders if a MIDI file was returned by only one algorithm, or by a multiple of three people if multiple algorithms returned the same polyphonic MIDI file.
- For each query, order the matches first by the rough and, in case of ties, by the fine score.
- Group together matches with the same average rough score.
- Only include items with average rough scores of better than "somewhat similar".
- If the resulting list is longer than 10, remove whole groups at the end until at most 10 items remain; there was one exception where the top group had 11 "very similar" items.

4.1.2. Measures

The following measures were used (these abbreviations are used in Figures 1, 2, and 3):

- ADR = Average Dynamic Recall [4].
- NRGB = Normalized Recall at Group Boundaries.
- AP = Average Precision (non-interpolated).
- PND = Precision at N Documents.
- Fine = Sum of fine-grained human similarity decisions (0-10).
- PSum = Sum of human broad similarity decisions: NS=0, SS=1, VS=2.

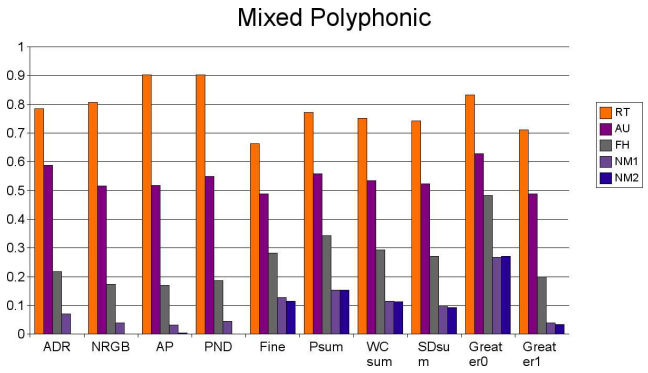


Figure 3. Task IIb: Mixed Polyphonic Overall Summary

- WCsum = 'World Cup' scoring: NS=0, SS=1, VS=3 (rewards "very similar").
- SDsum = 'Stephen Downie' scoring: NS=0, SS=1, VS=4 (strongly rewards "very similar").
- Greater0 = NS=0, SS=1, VS=1 (binary relevance judgement).
- Greater1 = NS=0, SS=0, VS=1 (binary relevance judgement using only "very similar").

All measures are normalized such that they lie in the range from 0 to 1.

4.1.3. Monophonic Task

For the monophonic task, there were no significant performance differences between our method and the editing distance for quotiented trees by Pascal Ferraro and Pierre Hanna [2], [1]. When looking at the actual result lists – they are available at http://rainer.typke.org/mirex06.0.html – the main difference between the results of these two methods seems to be that the latter is more likely to retrieve rather short matches (compare, for example, http://rainer.typke.org/qr3-fh.0.html and http://rainer.typke.org/qr3-rt.0.html). In some cases, this might have lead to a lower average precision or average dynamic recall, like for example in the case of http://rainer.typke.org/qr6-fh.0.html, where short matches pushed nice longer ones down to lower ranks.

The other methods performed worse than ours and Ferraro's/Hanna's, no matter which measure is used.

4.1.4. Polyphonic Tasks

For both polyphonic tasks, our method outperforms the other methods.

Besides the obvious difference in the number of notes that can sound at the same time, the polyphonic collections also differ in other ways from the monophonic RISM collection:

- Both polyphonic collections were random sets of files that were harvested from the Web. Because of this, the encoding quality was not as homogeneous as for the RISM collection. Some files in the polyphonic collection were not syntactically correct MIDI files.
- While the RISM collection was created from plaine&easie code and therefore rhythmically quantized, the polyphonic collections contained both quantized music and renditions of performances, where neither onset times nor note durations were exactly the same as what one would find in a written score.

When looking at Figures 2 and 3, it is very noticeable that the six rightmost measures, which were not based on the ranked lists described in Section 4.1.1, indicate a much worse performance for the Karaoke task for all algorithms. The reason is that the Karaoke collection was much smaller (1000 items instead of the 10,000 items in the mixed collection) and therefore contained fewer good matches to begin with. Even an ideal algorithm can therefore not reach a score of 1 for measures such as "Fine" for the Karaoke collection. The four measures on the left side compare the algorithms' outputs with the ground truth lists.

4.2. Query by Singing/Humming

No algorithm that relied on the provided MIDI files turned out to be very successful. These MIDI files contained many incorrectly recognized additional notes with more or less random pitches.The most successful algorithm that used the MIDI files provided by Jang, NM, performed an exhaustive depth-first search trying to scale the pattern note-by-note to fit the song, with costs applied to local time-scaling, note duration changes and pitch-shifting (note that this is not the same algorithm as the "symbolic" NM algorithm). Our algorithm performed relatively poorly because it looked for matches for *all* query notes, including the random added notes, without the possibility of reducing the importance of individual query notes. The NM algorithm is able to selectively ignore query notes that do not fit any note in a matching piece. For a similar effect with our method, we would need to skip the weight normalization step and therefore work with a pure EMD. However, this would reduce the effectiveness of vantage indexing – it would remove the guarantee that indexing does not lead to false negatives.

Many of the successful algorithms used symbolic approaches; for example, Xiao Wu and Ming Li use a transcription to notes for filtering out candidates for matches, followed by a final scoring on the frame level. Christian Sailer as well as Ernesto Lopez and Martín Rocamora do all matching in the symbolic domain after transcribing the audio signal to notes. An important reason for such hybrid approaches outperforming our method was that they used their own elaborate methods for transcribing the audio signal into notes.

	Task I (MPR)	Task II (Mean Precision)	Uses audio or MIDI
XW1 (Xiao Wu and Ming Li)	0.926	no entry	audio
XW2	0.900	no entry	audio
RJ (Roger Jang and Nien-Jung Lee	0.883	0.926	audio
RL (Ernesto Lopez and Martín Rocamora)	0.800	no entry	audio
NM	0.688	0.722	MIDI
CS1 (Christian Sailer)	0.568	0.587	audio
RT2	0.390	0.401	MIDI
CS3	0.348	0.415	audio
AU2	0.288	0.238	MIDI
CS2	0.283	0.649	audio
FH	0.218	0.309	MIDI
AU1	0.205	0.163	MIDI
RT1	0.196	0.468	MIDI

Table 1. Overall results for QBSH. NM, RT, AU, and FH use Jang's MIDI files, while the other methods do their own note transcription (except for RJ's pure audio approach). RT2 segments the queries, while RT1 does not. There seems to be a strong correlation between using the provided MIDI files and poor overall performance. For detailed descriptions of the various methods, see the MIREX abstracts at http://www.music-ir.org/mirex2006/index.php/QBSH:_Query-by-Singing/Humming_Results

Maybe one can learn from this experience that symbolic approaches can be valuable for searching large audio databases because they can make searches very efficient without necessarily ruining the quality of results, especially if the last scoring step is again done in the audio domain and therefore avoids the problems that note transcription introduces. After reducing the number of candidates for matches by using an efficient symbolic method, one can afford more expensive, but effective audio analyses since they only need to be done for a small number of items.

5. Acknowledgments

Many thanks to IMIRSEL for the huge amount of effort spent in running the evaluation, to Anna Pienimäki for co-leading the Symbolic Melodic Similarity task and creating the queries, to Maarten Grachten for his help with converting RISM incipits to MIDI, to RISM UK for allowing us to use their incipits, and to Niall O'Driscoll from Alexa for providing Musipedia.org with free access to Alexa's web search product (the test collection of polyphonic MIDI files is a subset of the Musipedia web search data set). For the Query by Singing/Humming task, we are grateful for Stephen Downie and J.-S. Roger Jang (張智星) leading the task, and for the data collection of wave files, pitch vectors, and MIDI files provided by the latter.

References

[1] P. Ferraro and C. Godin. "An Edit Distance Between Quotiented Trees", *Algorithmica*, 36:1–39, 2003.

[2] P. Ferraro and P. Hanna. "Symbolic Melodic Similarity," *MIREX 2006 abstract*.

[3] R. H. van Leuken, R. C. Veltkamp, and R. Typke. "Selecting vantage objects for similarity indexing," *International Conference on Pattern Recognition (ICPR)* 2006, Hong Kong.

[4] R. Typke, R. C. Veltkamp, and F. Wiering. "A measure for evaluating retrieval techniques based on partially ordered ground truth lists", *International Conference on Multimedia & Expo (ICME)* 2006, Toronto, Canada.

[5] R. Typke, F. Wiering, and R. C. Veltkamp. "Transportation distances and human perception of melodic similarity," *ESCOM Musicæ Scientiæ (to appear)*.

Variations on Local Alignment for Specific Query Types

Alexandra L. Uitdenbogerd
RMIT University
Melbourne, Victoria
Australia
sandrau@rmit.edu.au

Abstract

For this submission to MIREX, we again provided a simple baseline for comparison with other submissions. For short incipit queries we used a form of dynamic programming on melody strings that enforces matching from the start of the strings — a technique we call *Start-Match Alignment*. There is some evidence that this technique is better than local alignment for short queries and incipits.

For the query by humming track we used both local alignment and Start-Match Alignment. It seems that for this collection, query set and relevance set, Start-Match Alignment works better than local alignment on average. Our algorithms were the fastest of those submitted for the symbolic search tasks, and for the polyphonic symbolic task had very good effectiveness.

Keywords: melody matching, music retrieval

1. Introduction

Our entry in the Symbolic Melodic Similarity (SMS) and the Query by Singing/Humming (QBSH) tracks of the 2006 round of MIREX used one of the techniques shown in currently unpublished work to be more effective than local alignment under some circumstances. We call this particular matching technique "Start-Match Alignment", as it finds the best match between two strings, where the match is enforced to commence at the start of the strings.

The start-match technique is applied atop the basic melody extraction and melody standardisation techniques of our three-stage melody matching model [2]. It is our belief that this combination of techniques leads to robust matching. With the application of current efficient programming methods, the approach can be used to produce a practical system for purely symbolic matching. Our techniques had yet to be tested on hummed queries, so it was interesting to see that the techniques don't appear to work as well as those that are optimised for such queries.

2. Techniques

The family of music matching techniques our team have developed are based around the 3-stage melody matching model: melody extraction, melody standardisation, and melody matching. For monophonic queries this simplifies to just the second and third stage of the model.

In this submission we use the melody extraction technique *allmono* originally developed and tested in earlier work [3, 4, 5]. Similarly, we use the *directed modulo-12* approach to standardisation, which is a convenient simplification of pitch interval strings into a smaller representation that can be mapped into alphabetic characters (a concept used by Hawley in earlier work [1]). Since the representation is purely based on pitch, variations in tempo between a sung query and the target piece of music will not affect the ranking.

The melody matching technique used for this submission is a little different to our earlier work. Based on work that is currently in submission, the approach enforces matching of strings from the start of the strings until a best matching length is found. This technique, which we have named *Start-Match Alignment*, initialises and fills the array in the manner of global alignment, but, in the manner of local alignment, returns the highest score within the matrix. The equation used to calculate each cell's value is the same as for global alignment.

$$a[i,j] = max \begin{cases} a[i-1,j]+d & i \geq 1 \\ a[i,j-1]+d & j \geq 1 \\ a[i-1,j-1]+e & p(i) = t(j) \text{ and } i,j \geq 1 \\ a[i-1,j-1]+m & p(i) \neq t(j) \text{ and } i,j \geq 1 \\ 0 & i,j = 0 \end{cases} \quad (1)$$

where d is the cost of an insert or delete, e is the value of an exact match, m is the cost of a mismatch, i and j are non-negative integers, $p(i)$ represents the ith symbol in the "pattern" or query, and $t(j)$ represents the jth symbol in the "text", or potential answer string. The weights we used were 1 for a match, -1 for a mismatch, and -2 for an insert or delete (indel).

Each melody string in the collection is compared to the query string, with ranking based on the computed alignment score. The higher the alignment score, the more similar the

two strings are assumed to be.

For QBSH we chose to use the above algorithm in addition to standard local alignment of the standardised melody strings. Local alignment locates the best substring match between two strings, regardless of where the match occurs and its length.

Due to a slip-up in the script preparation, our submission for the SMS tasks normalised the score by dividing by the log of the length of the track (in number of symbols plus one) against which the query was being compared. Results on our collection suggest that this is slightly less effective than not normalising. See the Discussion section for possible reasons.

3. MIREX Tasks and Results

SMS task 1 consisted of incipit queries matched against the RISM collection of incipits. Half of the queries were transcriptions of a hummed or whistled source. Both collection and queries were monophonic. In this task our technique was ranked in the middle of the set of algorithms. It was one of three techniques that used an "indexing" phase. The query time was the fastest, being 31 seconds, with the next shortest being 59 seconds. At 64 seconds, the sum of the indexing and querying times was 4 seconds shorter than the fastest query-phase-only submission.

SMS task 2 involved search for melodies in a polyphonic collection. Of the five algorithms, Start-Match was ranked second across various measures of effectiveness. The query time was faster than all other submissions, as was the indexing time.

The QBSH tasks used a very large collection of sung queries against a monophonic collection consisting of 48 ground truth quantised melodies in addition to the Essen collection of folk songs. For this task our best algorithm performed poorly compared to other submissions (eighth out of thirteen for task 1 and second last for task 2). However, it is clear that the Start-Match algorithm was more effective than local alignment. Once again our submission was very fast compared to other entrants.

4. Discussion

This particular submission was not optimised for speed. Retaining the collection strings in memory, or using a compressed form for matching would be much faster, as would the use of a heap for retaining the top 10 results. Despite the above, the submission was the fastest at answering queries in both tasks of the SMS track.

As mentioned earlier, the SMS submission inadvertently applied normalisation to the alignment scores. When tested on our own collection this leads to slightly worse effectiveness. On spending considerable time examining the retrieved answers to the queries, it became clear that a reason normalisation may be unhelpful is that the length of the *track* was used, and not the length of the entire piece. This can lead to short bits of accompaniment getting much higher scores than warranted. Despite this liability, our submission was in the middle range for effectiveness in SMS task 1, and the second most effective algorithm for task 2.

For the QBSH tasks, there was an obvious difference in effectiveness results between those algorithms working directly with the WAV files, and those using the provided MIDI. There may have been errors introduced into the MIDI files at the time of manual transcription (as suggested by the organiser), which made the task more difficult. While our technique was the fastest for these tasks, the use of MIDI files instead of WAV meant that less processing was required than some of the other submissions, making some comparisons meaningless.

The lack of success in task 2 of QBSH for Start-Match may in part be attributed to the types of errors found in sung queries. The alignment approach we used causes two symbols to be incorrect when a single note error occurs in a query, such as a wrong note substituted for a correct note. This can cause the penalty to be too great when matching some query-melody pairs. There are more robust techniques for this alignment problem that we intend to apply in the future.

An issue that seems apparent is that algorithms work best on collections and queries most similar to those that researchers have used for their development. Our algorithms were developed on monophonic symbolic queries against a polyphonic collection, and they worked best on these. Typke et al.'s development has largely been on incipit collections, and this led to excellent results for that domain. Jang et al.'s familiarity with the use of sung queries appeared to pay off in the QBSH track. Rather than being an issue of fair-play, it highlights what is known about training and test collections. It also makes clear that the assumptions about the nature of the collection and queries results in different choices for optimal algorithms.

5. Conclusion

MIREX 2006 evaluation has shown that a simple alignment technique applied to pitch interval strings is still competitive for some symbolic query and collection types, but doesn't handle sung queries quite as well as other approaches that were submitted. Future work for our team should include experimentation with sung query sets.

6. Acknowledgments

We thank Iman S. H. Suyoto for providing his MidiPerl code from MIREX 2005.

References

[1] M. Hawley. The personal orchestra, or audio data compression by 10,000:1. *Computing Systems*, 3(2):289–329, 1990.

[2] A. L. Uitdenbogerd and J. Zobel. An architecture for effective music information retrieval. *J. American Society of Information Science and Technology (JASIST)*, 55(12):1053–1057.

[3] A. L. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In B. Smith and W. Effelsberg, editors, *Proc. ACM International Multimedia Conference*, pages 235–240, Bristol, UK, September 1998. ACM, ACM Press.

[4] A. L. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In D. Bulterman, K. Jeffay, and H. J. Zhang, editors, *Proc. ACM International Multimedia Conference*, pages 57–66, Orlando Florida, USA, November 1999. ACM, ACM Press.

[5] A. L. Uitdenbogerd and J. Zobel. Music ranking techniques evaluated. In M. Oudshoorn, editor, *Proc. Australasian Computer Science Conference*, Melbourne, Australia, January 2002.

MIREX 2006 Audio Beat Tracking Evaluation: BeatRoot

Simon Dixon
Austrian Research Institute for Artificial Intelligence
Freyung 6/6, Vienna 1010, Austria
simon.dixon@ofai.at

Abstract

BeatRoot is an interactive beat tracking system which has been used for several years in studies of performance timing. In this new version, some of the weaknesses of the original system have been addressed. The original simple onset detection algorithm, which caused problems for beat tracking music without prominent drums, has been replaced with a more robust onset detector. Several new features have been added, such as annotation of multiple metrical levels and phrase boundaries, and improvements in the user interface. Also, the new version has been written entirely in Java, so that it runs on all major platforms. The beat tracking algorithm remains largely unchanged: BeatRoot uses a multiple agent architecture which simultaneously considers several different hypotheses concerning the rate and placement of musical beats, resulting in accurate tracking of the beat, quick recovery from errors, and graceful degradation in cases where the beat is only weakly implied by the data.

Keywords: MIREX, tempo induction, beat tracking.

1. Introduction

Compared with complex cognitive tasks such as playing chess, beat tracking (identifying the basic rhythmic pulse of a piece of music) does not appear to be particularly difficult, as it is performed by people with little or no musical training, who tap their feet, clap their hands or dance in time with music. However, while chess programs compete with world champions, no computer program has been developed which approaches the beat tracking ability of a good musician.

As a fundamental part of music cognition, beat tracking has practical uses in performance analysis, perceptual modelling, audio content analysis (such as for music transcription and music information retrieval), and the synchronisation of musical performance with computers or other devices. The previous version of BeatRoot [1, 2] was used in a large scale study of interpretation in piano performance [3, 4] to create symbolic metadata from audio CDs for automatic analysis of performance timing.

In this paper we describe the new version of BeatRoot, a system which models the perception of beat by two interacting processes: the first finds the rate of the beats (*tempo induction*), and the second synchronises a pulse sequence with the music (*beat tracking*). A clustering algorithm finds the most significant metrical units, and the clusters are then compared to find reinforcing groups, and a ranked set of tempo hypotheses is computed. Based on these hypotheses, a multiple agent architecture is employed to match sequences of beats to the music, where each agent represents a specific tempo and alignment of beats with the music. The agents are evaluated on the basis of the regularity, continuity and salience of the onsets corresponding to hypothesised beats, and the highest ranked beat sequence is returned as the solution. The user interface presents a graphical representation of the music and the extracted beats, and allows the user to edit and recalculate results based on the editing. More complete descriptions of the algorithms can be found in [1, 5].

2. BeatRoot Architecture

BeatRoot takes digital audio as input, and processes the data off-line to detect salient rhythmic events. The timing of these events is then analysed to generate hypotheses of the tempo at various metrical levels. The stages of processing are shown in Figure 1, and will be described in the following subsections.

2.1. Onset Detection

Initial processing of the audio signal is concerned with finding the onsets of musical notes, which are the primary carriers of rhythmic information. Earlier versions of BeatRoot used a time-domain onset detection algorithm, which finds local peaks in the slope of a smoothed amplitude envelope. This method is particularly well suited to music with drums, but less reliable at finding onsets of other instruments in a polyphonic setting. In the current version it has been replaced with an onset detector which finds peaks in the spectral flux. This method is described fully in [5].

Spectral flux sums the change in magnitude in each frequency bin where the change is positive, that is, the energy is increasing. First, a time-frequency representation of the signal based on a short time Fourier transform using a Hamming window $w(m)$ is calculated at a frame rate of 100 Hz. If $X(n, k)$ represents the kth frequency bin of the nth frame,

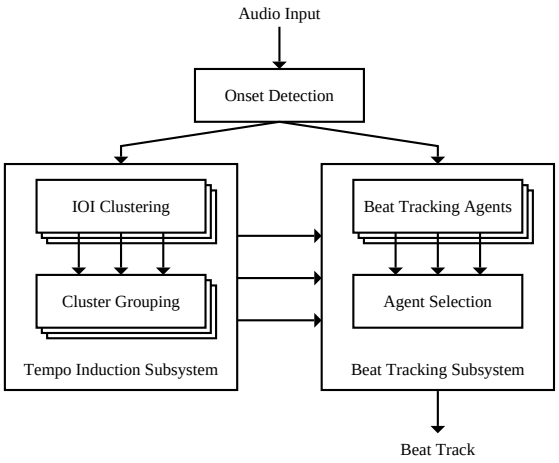


Figure 1. System architecture of BeatRoot

then:

$$X(n,k) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(hn+m)\, w(m)\, e^{-\frac{2j\pi mk}{N}}$$

where the window size $N = 2048$ (46 ms at a sampling rate of $r = 44100$ Hz) and hop size $h = 441$ (10 ms, or 78.5% overlap). The spectral flux function SF is then given by:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|)$$

where $H(x) = \frac{x+|x|}{2}$ is the half-wave rectifier function. Empirical tests favoured the use of the L_1-norm here over the L_2-norm used in [6, 7], and the linear magnitude over the logarithmic (relative or normalised) function proposed by Klapuri [8].

2.2. Tempo Induction

The tempo induction algorithm uses the calculated onset times to compute clusters of inter-onset intervals (IOIs). An IOI is defined to be the time interval between any pair of onsets, not necessarily successive. In most types of music, IOIs corresponding to the beat and simple integer multiples and fractions of the beat are most common. Due to fluctuations in timing and tempo, this correspondence is not precise, but by using a clustering algorithm, it is possible to find groups of similar IOIs which represent the various musical units (e.g. half notes, quarter notes).

This first stage of the tempo induction algorithm is represented in Figure 2, which shows the events along a time line (above), and the various IOIs (below), labelled with their corresponding cluster names (C1, C2, etc.). The next stage is to combine the information about the clusters, by recognising approximate integer relationships between clusters.

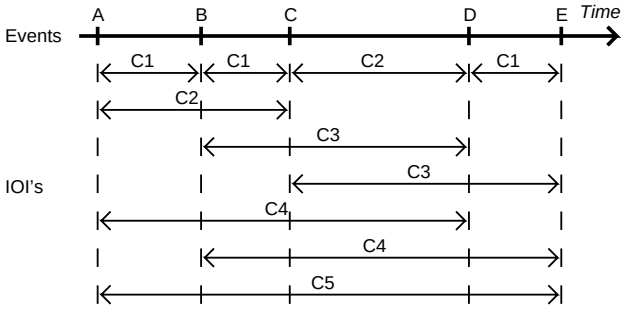


Figure 2. Clustering of inter-onset intervals: each interval between any pair of events is assigned to a cluster (C1, C2, C3, C4 or C5)

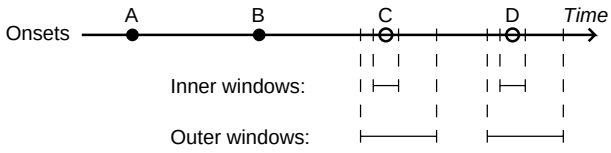


Figure 3. Tolerance windows of a beat tracking agent predicting beats around C and D after choosing beats at onsets A and B

For example, in Figure 2, cluster C2 is twice the duration of C1, and C4 is twice the duration of C2. This information, along with the number of IOIs in each cluster, is used to weight the clusters, and a ranked list of tempo hypotheses is produced and passed to the beat tracking subsystem.

2.3. Beat Tracking

The most complex part of BeatRoot is the beat tracking subsystem, which uses a multiple agent architecture to find sequences of events which match the various tempo hypotheses, and rates each sequence to determine the most likely sequence of beat times. The music is processed sequentially from beginning to end, and at any particular point, the agents represent the various hypotheses about the rate and the timing of the beats up to that time, and make predictions of the next beats based on their current state.

Each agent is initialised with a tempo (rate) hypothesis from the tempo induction subsystem and an onset time, taken from the first few onsets, which defines the agent's first beat time (phase). The agent then predicts further beats spaced according to the given tempo and first beat, using tolerance windows to allow for deviations from perfectly metrical time (see Figure 3). Onsets which correspond with the inner window of predicted beat times are taken as actual beat times, and are stored by the agent and used to update its rate and phase. Onsets falling in the outer window are taken to be possible beat times, but the possibility that the onset is not on the beat is also considered.

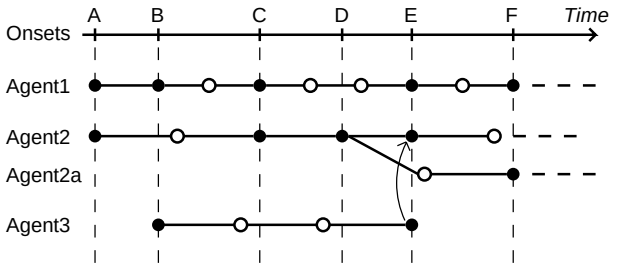


Figure 4. Beat tracking by multiple agents (see text for explanation)

Figure 4 illustrates the operation of beat tracking agents. A time line with 6 onsets (A to F) is shown, and below the time line are horizontal lines marked with solid and hollow circles, representing the behaviour of each agent. The solid circles represent predicted beat times which correspond to onsets, and the hollow circles represent predicted beat times which do not correspond to onsets. The circles of Agent1 are more closely spaced, representing a faster tempo than that of the other agents.

Agent1 is initialised with onset A as its first beat. It then predicts a beat according to its initial tempo hypothesis from the tempo induction stage, and onset B is within the inner window of this prediction, so it is taken to be on the beat. Agent1's next prediction lies between onsets, so a further prediction, spaced two beats from the last matching onset, is made. This matches onset C, so the agent marks C as a beat time and interpolates the missing beat between B and C. Then the agent continues, matching further predictions to onsets E and F, and interpolating missing beats as necessary.

Agent2 illustrates the case when an onset matches only the outer prediction window, in this case at onset E. Because there are two possibilities, a new agent (Agent2a) is created to cater for the possibility that E is not a beat, while Agent2 assumes that E corresponds to a beat.

A special case is shown by Agent2 and Agent3 at onset E, when it is found that two agents agree on the time and rate of the beat. Rather than allowing the agents to duplicate each others' work for the remainder of the piece, one of the agents is terminated. The choice of agent to terminate is based on the evaluation function described in the following paragraph. In this case, Agent3 is terminated, as indicated by the arrow. A further special case (not illustrated) is that an agent can be terminated if it finds no events corresponding to its beat predictions (it has lost track of the beat).

Each agent is equipped with an evaluation function which rates how well the predicted and actual beat times correspond. The rating is based on how evenly the beat times are spaced, how many predicted beats correspond to actual events, and the salience of the matched events, which is calculated from the spectral flux at the time of the onset. At the end of processing, the agent with the highest score outputs its sequence of beats as the solution to the beat tracking problem.

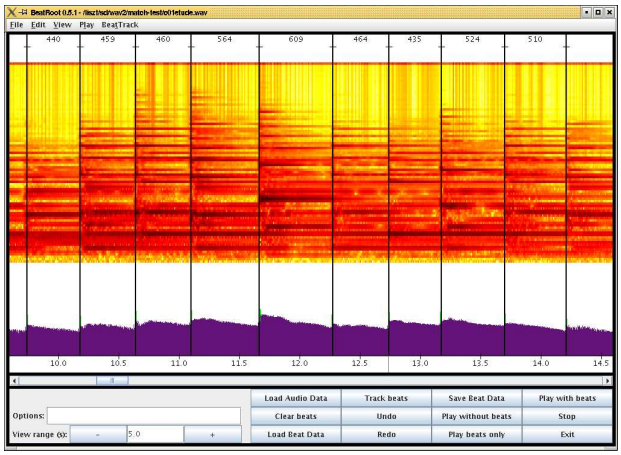


Figure 5. Screen shot of BeatRoot showing a 5-second excerpt from a Chopin piano Etude (Op.10, No.3), with the inter-beat intervals in ms (top), calculated beat times (long vertical lines), spectrogram (centre), amplitude envelope (below) marked with detected onsets (short vertical lines) and the control panel (bottom)

2.4. Implementation

The system described above has been implemented with a graphical user interface which allows playback of the music with the beat times marked by clicks, and provides a graphical display of the signal and the beats with editing functions for correction of errors or selection of alternate metrical levels. The audio data is displayed as a waveform and spectrogram, and the beats are shown as vertical lines on the display (Figure 5).

BeatRoot is written in Java and is available from:
`http://www.ofai.at/~simon.dixon/beatroot`

3. Results

3.1. Testing

BeatRoot was tested on a range of different musical styles, including classical, jazz, and popular works with a variety of tempi and meters. The following results were obtained with the previous version of BeatRoot, using test data consisting of a set of 13 complete piano sonatas, a large collection of solo piano performances of two Beatles songs and a small set of popular, jazz and latin songs. In each case, the system found an average of over 90% of the beats [1], and compared favourably to another (then) state-of-the-art tempo tracker [9]. Tempo induction was in most cases correct, with the most common error being the choice of a musically related metrical level such as double or half the subjectively chosen primary rate. The calculation of beat times is also quite robust; when the system loses synchronisation

Contestant	P-Score (average)	Run-time
Dixon	0.407	639
Ellis	0.401	498
Klapuri	0.395	1218
Davies	0.394	1394
Brossier	0.391	139

Table 1. Results of the MIREX 2006 Audio Beat Tracking Evaluation

with the beat, it usually recovers quickly to resume correct beat tracking, despite the fact that the system has no high level knowledge of music to guide it. Some audio examples are available at:
`http://www.ofai.at/~simon.dixon`

3.2. MIREX 2006 Results

BeatRoot performed best of the 5 systems submitted for the MIREX 2006 Audio Beat Tracking Evaluation, as shown in Table 1. The test data consisted of 140 files from a wide range of musical styles, which had been annotated by around 40 people per file by tapping in time with the music. Although this is a slightly different task than off-line beat tracking (see [10] for a discussion), it is a reasonable approach for this evaluation, especially considering the difficulty of creating or obtaining ground-truth data.

3.3. Discussion

Since the results have been summarised as a single score, we do not know if the difference in performance between systems is significant, nor whether the systems' choice of metrical levels was a deciding factor in these results. BeatRoot is not programmed to select the metrical level corresponding to the perceived beat, nor to a typical tapping rate; it tends to prefer faster rates, because they turn out to be easier to track, in the sense that the agents achieve higher scores. More detailed results and analysis would be very interesting. An interesting task for future years would be to test beat tracking performance for a given metrical level (e.g. given the first two beats or the initial tempo). It would also be interesting to know the P-scores of the annotators (tappers), measured on the basis of the other tappers' data, to see how close this year's entries are to human beat tracking ability.

4. Acknowledgements

This work was supported by the Vienna Science and Technology Fund, project CI010 *Interfaces to Music*, and the EU project S2S^2. OFAI acknowledges the support of the ministries BMBWK and BMVIT. Thanks to the MIREX team for conducting the MIREX evaluation.

References

[1] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[2] S. Dixon, "An interactive beat tracking and visualisation system," in *Proceedings of the International Computer Music Conference*, 2001, pp. 215–218.

[3] G. Widmer, "Machine discoveries: A few simple, robust local expression principles," *Journal of New Music Research*, vol. 31, no. 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, no. 3, pp. 111–130, 2003.

[5] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, 2006, pp. 133–137.

[6] C. Duxbury, M. Sandler, and M. Davies, "A hybrid approach to musical note onset detection," in *Proceedings of the 5th International Conference on Digital Audio Effects*, 2002, pp. 33–38.

[7] J.P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in musical signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[8] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Phoenix, Arizona, 1999.

[9] S. Dixon, "An empirical comparison of tempo trackers," in *Proceedings of the 8th Brazilian Symposium on Computer Music*. 2001, pp. 832–840, Brazilian Computing Society.

[10] S. Dixon, W. Goebl, and E. Cambouropoulos, "Perceptual smoothness of tempo in expressively performed music," *Music Perception*, vol. 23, no. 3, pp. 195–214, 2006.

Audio Beat Tracking Algorithm for MIREX 2006

Anssi Klapuri
Institute of Signal Processing, Tampere University of Technology
Korkeakoulunkatu 1, 33720 Tampere, Finland
anssi.klapuri@tut.fi

Abstract

This paper describes an audio beat tracking algorithm submitted to the MIREX 2006 contest. The algorithm has been described in detail in the article "Analysis of the Meter of Acoustic Musical Signals" published in IEEE Trans. Audio, Speech and Language Processing, 14(1), 2006. In summary, the method analyses musical meter jointly at three time scales, of which only the beat (tactus) level is used in this MIREX task.

Keywords: Musical meter analysis, beat tracking.

1. Introduction

The beat tracking algorithm employed here is identical to that described in [1]. [1] The algorithm was originally implemented by the author in 2003 and converted to C++ by Jouni Paulus in Spring 2004. The algorithm and its parameter values have been untouched since then.

In [1], both causal and non-causal versions of the method were described. The practical difference between the two is that the causal version generates beat estimates based on past samples, whereas the non-causal version does (Viterbi) backtracking to find the globally optimal beat track after hearing the entire excerpt. The backtracking improves accuracy especially during the beginning of an analysis signal.

Here we employed the causal version of the algorithm. The non-causal version would be slightly more accurate especially for short analysis signals like those used in this contest, but also less realistic for on-line applications.

2. The meter analysis algorithm

The aim of the method proposed in [1] is to estimate the meter of acoustic musical signals at three levels: at the tactus, tatum, and measure-pulse levels. An overview of the method is shown in Fig. 1.

For the time-frequency analysis part, a technique is employed which aims at measuring the degree of spectral change, or, "accent" in music signals. In brief, preliminary time-frequency analysis is conducted using a quite large number of subbands and by measuring the degree of spectral change at these channels. Then, adjacent bands are combined to arrive at four bandwise accent signals, for which periodicity analysis is carried out.

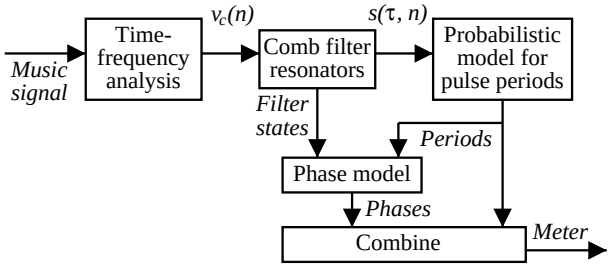


Figure 1. Overview of the meter analysis method.

Periodicity analysis of the bandwise accent signals is performed using a bank of comb filter resonators very similar to those used by Scheirer in [2]. Before we ended up using comb filters, four different period estimation algorithms were evaluated. A bank of comb filter resonators was chosen because it was the least complex among the three best-performing algorithms.

The comb filters serve as feature extractors for two probabilistic models. One model is used to estimate the period-lengths of metrical pulses at different levels. The other model is used to estimate the corresponding phases (see Fig. 1). The probabilistic models encode prior musical knowledge regarding well-formed musical meters. In brief, the models take into account the dependencies between different pulse levels (tatum, tactus, and measure) and, additionally, implement temporal tying between successive meter estimates.

3. Acknowledgements

The author is grateful to Jouni Paulus who wrote a C++ implementation of the method based on the Matlab (and some C) code of the author.

References

[1] 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.

[2] E. Scheirer, "Tempo and beat analysis of acoustical musical signals," *Journal of the Acoustical Society of America*, vol. 103, pp. 588–601, Jan. 1998.

[1] Available at www.cs.tut.fi/sgn/arg/klap/sapmeter.pdf.

Identifying 'Cover Songs' with Beat-Synchronous Chroma Features

Daniel P.W. Ellis
LabROSA, Dept. of Electrical Engineering
Columbia University, New York NY 10027 USA
dpwe@ee.columbia.edu

Abstract

Large music collections, ranging from thousands to millions of tracks, are unsuited to manual searching, motivating the development of automatic search methods. When two musical groups perform the same underlying song or piece, these are known as 'cover' versions. We describe a system that attempts to identify such a relationship between music audio recordings. To overcome variability in tempo, we use beat-tracking to describe each piece with one feature vector per beat. To deal with variation in instrumentation, we use 12-dimensional chroma feature vectors that collect spectral energy supporting each semitone of the octave. To compare two recordings, we simply cross-correlate the entire beat-by-chroma representation for two tracks and look for sharp peaks indicating good local alignment between the pieces. Evaluation on a small set of 15 pairs of pop music cover versions identified within the USPOP2002 collection achieves a performance of around 60% correct.

Keywords: Music Similarity, Cover Songs, Chroma Features, Beat Tracking

1. Introduction

Immediate access to large music collections is now commonplace – be they the thousands of songs on the MP3 player in your pocket, or the millions of songs available at online music stores. But finding music within such collections can be very problematic, leading to the current interest in automatic music similarity estimation. In this paper, we address a slightly different problem: rather than trying to find music whose genre, style, or instrumentation match particular query examples, we are trying to find versions of the *same piece* of music, despite the fact that they may be performed with very different styles, instrumentation, etc. These alternate versions of the same underlying piece of music are known as 'cover versions'.

Cover versions will typically retain the essence of the melody and the lyrics (for a song) but may vary greatly in other dimensions. Indeed, in pop music, the main purpose of recording a cover version is typically to investigate a more-or-less radically different interpretation of a song (although in different recordings of classical music the variations may be more subtle). Thus, to solve this problem, we must devise representations and matching schemes that are robust to changes in tempo, instrumentation, and general musical style.

2. Overview

Our representation has two main features: We use a beat tracker to generate a beat-synchronous representation with one feature vector per beat. Thus, variations in tempo are largely normalized as long as the same number of beats is used in each phrase. The representation of each beat is a normalized chroma vector, which sums up spectral energy into twelve bins corresponding to the twelve distinct semitones within an octave, but attempting to remove any distinction between different octaves. Chroma features capture both melodic information (since the melody note will typically dominate the feature) and harmonic information (since other notes in chords will result in secondary peaks in a given vector).

To match two tracks represented by such beat-vs-chroma matrices, we simply cross-correlate the entire pieces. Long sequences of beats with similar tonal structure will result in local maxima at the appropriate lags in the cross-correlation, with the size of the peak increasing both with the degree of similarity in the chroma features, and the length of matching sequences. To distinguish between genuine matches and incidental high cross-correlations, we emphasize rapid variations in the cross-correlation (i.e. particular lags at which alignment is high despite being low at neighboring lags) through high-pass filtering. To accommodate transposition between versions (performances in different keys), we cross-correlate between all twelve possible semitone transpositions of the chroma vectors.

3. Beat tracking

Our beat-tracker is based on the description of Jehan [5]. A log-magnitude 40-channel Mel-frequency spectrogram is calculated for 8 kHz downsampled mono versions of the original recording with a 32 ms window and 8 ms hop between frames. The first-order difference along time in each frequency channel is half-wave rectified (to leave only onset information) then summed across frequency. This "onset envelope" is high-pass filtered with a 3 dB point at 0.01

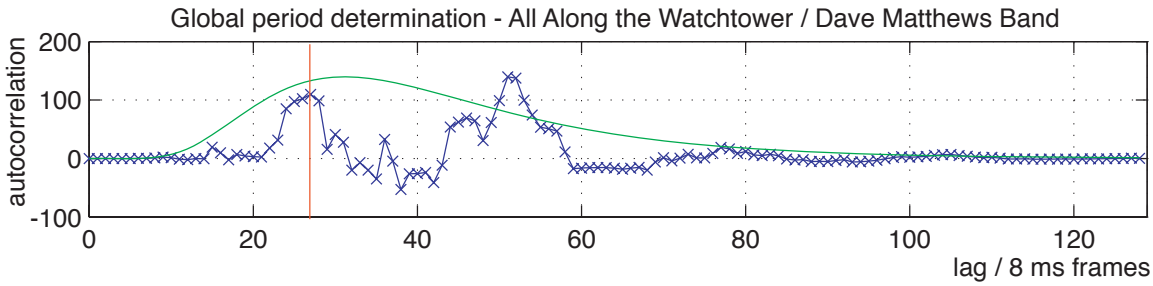


Figure 1. Autocorrelation of the first 90 s of a piece, used to choose the global target tempo. The Gaussian weighting window is shown overlaid, and the chosen period (27 samples = 278 bpm) is shown by a vertical line.

rad/samp to remove d.c. offset, and the first 90 s of the piece are autocorrelated out to a lag of 128 points (1.024 s). This autocorrelation is windowed with a Gaussian in log-period centered on 240 bpm, with a half-width of 1.5 octaves. Then the shortest lag that is a local maximum with a value at least 0.4 times the largest maximum in the windowed autocorrelation is taken as the global target period. This favors the multiple of the basic beat of the piece that is closest to 240 bpm i.e. closer to the tatum (shortest melody note duration) than what would be the notated tempo of the piece. Figure 1 shows an example of the global autocorrelation

The onset envelope is then filtered by a periodicity enhancing smoothing window composed of cos^8 at the global target period, Hann-windowed out to ± 3 periods. Beats are then chosen as the local maxima of this enhanced onset function within each beat-length window centered one beat on from the last marked beat. However, if no maxima reaches 0.25 of the magnitude of the last-picked maxima, the default predicted beat position is used instead, and the search continues forward. This allows the tracker to continue through short stretches of weak or absent beat.

4. Chroma features

To the extent that the beat tracking can identify the same main pulse in different renditions of the same piece, representing the audio against a time base defined by the detected beats normalizes away variations in tempo. We choose to record a single feature vector per beat, and use twelve element 'chroma' features to capture both the dominant note (typically melody) as well as the broad harmonic accompaniment [4, 1]. The idea of calculating harmonic features over beat-length segments appears to have been developed several times; we first became aware of it in [6].

Rather than using a coarse mapping of FFT bins to the chroma classes they overlap (which is particularly blurry at low frequencies), we use the phase-derivative within each FFT bin both to identify strong tonal components in the spectrum (indicated by spectrally-adjacent bins with close instantaneous frequencies) and to get a higher-resolution estimate of the underlying frequency [2]. We found that using only components up to 1 kHz in our chroma features worked best. Figure 2 shows an example of the chroma features alongside the beat-tracked mel spectrum of the fragment they describe.

5. Matching

From the processing so far, we have each recording represented by a matrix of 12 chroma dimensions by however many beats are detected in the entire piece. We expect cover versions to have long stretches (verses, choruses, etc.) that match reasonably well, although we don't particularly expect these to occur in exactly the same places, absolutely or relatively, in the two versions. We initially experimented with chopping one piece up into multiple fragments and looking for the best cross-correlation of each fragment in the test piece, but in addition to being very slow it was difficult to choose the best length of fragment size. In the end, the simpler approach of cross-correlating the entirety of the two matrices gave us the best results. Although this is unable to reward the situation when multiple fragments align but at different relative alignments, it does have the nice property of rewarding both a good correlation between the chroma vectors and a long sequence of aligned beats, since the overall peak correlation is a product of both of these. Chroma vectors are intrinsically non-negative; we scaled them to have unit norm at each time slice. The cross-correlation is further normalized by the length of the shorter segment, so the correlation results are bounded to lie between zero and one. We perform the cross-correlation twelve times, once for each possible relative rotation (transposition) of the two feature matrices.

We observed, however, a number of spurious large correlations from relatively long stretches dominated by a single chroma bin; this occurs in many tracks. We found that genuine matches were indicated not only by absolutely large cross-correlations but also by sharp local maxima in cross-correlations that fell off rapidly as the relative alignment changes from its best value. To emphasize these sharp local maxima, we choose the transposition that gives the largest peak correlation then high-pass filter that cross-correlation function with a 3 dB point at 0.1 rad/sample. The 'distance' reported for the evaluation is simply the reciprocal

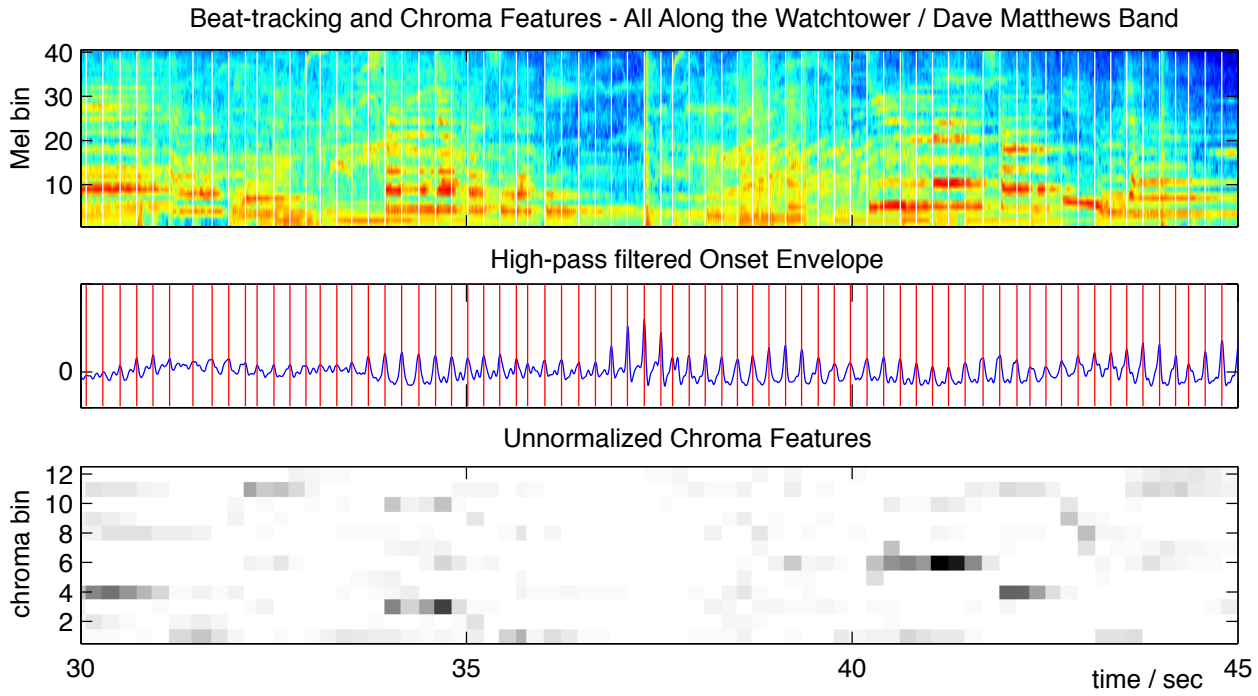


Figure 2. Excerpt showing the Mel-scale spectrogram (top pane), the periodicity-enhanced onset envelope (middle pane, with chosen beat instants indicated), and the unnormalized per-beat chroma feature vectors (bottom pane).

of the peak value of this high-pass filtered cross-correlation; matching tracks typically score below 20, whereas unrelated tracks are usually above 50.

Matching will fail if the feature extraction is based on beats with different relations to the music i.e. if one version tracks twice as many beats per song phrase. To accommodate this, we experimented with including two representations of each track, the original plus one using double the beat length (i.e. around 120 bpm) but this did not offer any advantage in our experiments.

6. Evaluation

We developed the system on set of 15 pairs of pop-music tracks that were alternate versions of the same song. They were extracted from the USPOP2002 dataset [3] by making a list of all tracks from the total set of 8764 tracks that had the same name, then listening to each pair to see if they were in fact the same piece; about 20% were. We stopped after we had found 15 pairs. Interestingly, it was often hard to tell if two tracks were the same until the verse began, at which point the lyrics quickly indicated matching tracks.

We made two lists of tracks, each containing one of the two versions of each track. In the evaluation, each track in the A list was compared to every track in the B list, and called a cover version of the track that it was most similar to; thus, the task was to identify the cover version knowing that one exists, rather than deciding if two songs were similar enough to be considered covers. Our best system (over variations in parameters such as filter breakpoints for the chroma features and matching) correctly identified 10 of 15 tracks; typical performance varied between 6 and 9 correct (where guessing would give one). Four of the pairs were clearly difficult for our representation and were almost never correctly identified. The test set is detailed in table 1.

7. Conclusions

Identifying cover tracks is an interesting new direction for content-based search of music audio databases. However, it is much more computationally expensive than the time-insensitive feature-distribution models typically used in genre and artist classification: our initial experiments took up to 30 s to compare each pair of tracks, making search in large databases completely intractible; we managed to speed this up by a factor of 100, but this still limits the size of database that we can afford to search by such direct means.

Our plan is to use these techniques to identify a dictionary of smaller fragments that can provide the most efficient coverage of large music databases. These can then be used as (possibly redundant) 'index terms' to permit the use of more rapid indexing schemes, as well as potentially revealing interesting repeated motifs and shared structure within music collections.

Table 1. Cover version test set from uspop2002, along with typical system performance.

Title	"A" artist	"B" artist	comments
Abracadabra	Steve Miller Band	Sugar Ray	easy
Addicted to Love	Robert Palmer	Tina Turner	hard
All Along the Watchtower	Dave Matthews Band	Jimi Hendrix	hard
America	Paul Simon (live)	Simon and Garfunkel	
Before You Accuse Me	Creedence Clearwater Revival	Eric Clapton	
Blue Collar Man	REO Speedwagon	Styx	easy
Caroline No	Beach Boys	Brian Wilson (live)	easy
Cecilia	Paul Simon (live)	Simon and Garfunkel	very hard
Claudette	Everly Brothers	Roy Orbison	easy
Cocaine	Eric Clapton	Nazareth	
Come Together	Aerosmith	Beatles	easy
Day Tripper	Beatles	Cheap Trick	easy
Faith	George Michael	Limp Bizkit	
God Only Knows	Beach Boys	Brian Wilson (live)	hard
Gold Dust Woman	Fleetwood Mac	Sheryl Crow	easy

Acknowledgments

This work was supported by the Columbia Academic Quality Fund, and by the National Science Foundation (NSF) under Grant No. IIS-0238301. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF.

References

[1] M. A. Bartsch and G. H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, New York, October 2001.

[2] F. J. Charpentier. Pitch detection using the short-term phase spectrum. In *Proc. ICASSP-86*, pages 113–116, Tokyo, 1986.

[3] D. Ellis, A. Berenzweig, and B. Whitman. The "uspop2002" pop music data set, 2003. `http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html`.

[4] T. Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proc. ICMC*, pages 464–467, Beijing, 1999.

[5] T. Jehan. *Creating Music by Listening*. PhD thesis, MIT Media Lab, Cambridge, MA, 2005.

[6] 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*, pages 112–119, New York NY, 2004.

Identifying Cover Songs from Audio Using Harmonic Representation

Kyogu Lee
Center for Computer Research in Music and Acoustics
Department of Music, Stanford University
kglee@ccrma.stanford.edu

Abstract

This extended abstract describes in detail a submission to the task on Audio Cover Song in the Music Information Retrieval eXchange in 2006. The system uses as feature set a chord sequence identified by an HMM trained with audio-from-symbolic data, and computes a distance between two chord sequence pair using the Dynamic Time Warping algorithm to find the minimum alignment cost. The rational behind the system is that cover songs largely preserve harmonic content even if they vary in other musical attributes such as instrumentation, tempo, key, and/or melody.

Keywords: MIREX, Cover Song, Chord Sequence, Dynamic Time Warping

1. Introduction

A cover song is defined as a song performed by an artist different from the original artist [1]. Identifying cover songs given an original as a seed/query or finding the original given a cover version from the raw audio is a challenging task, and it has recently drawn attention in a Music Information Retrieval society. Cover songs are different from its original in terms of many musical attributes such as duration, tempo, dynamics, instrumentation, timbre, or even genre. Therefore, the raw audio in the time-domain or its frequency-domain representation like spectrogram is very different from each other. Such diversity found in cover songs requires a robust feature set that remains largely unchanged under various musical changes mentioned above.

Harmonic progression is a robust mid-level representation that is largely preserved under such musical variations. While other musical details such as melody, tempo, and/or timbre may vary from one to another, their harmonic progression over time undergoes minor changes compared with the others.

2. System Overview

Our system consists of two main blocks – (1) feature set is first extracted from the raw audio, and (2) distance measures are computed between a song pair based on the extracted feature set. Figure1 illustrates the overview of the system.

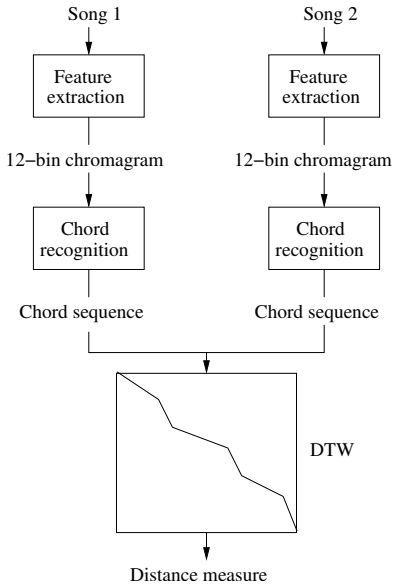


Figure 1. System overview

2.1. Chord recognition

Automatic chord recognition algorithm is decomposed of two parts. The first part computes a quantized 12-bin chromagram from the raw audio [1]. After a chromagram is obtained, automatic chord recognition algorithm based on the HMM is applied to get the chord sequence [2, 3], which has just one value per frame. Figure2 shows the overview of the automatic chord recognition system.

Transposition from one key to another is not rare in cover songs, and it may cause a serious problem in computing the distance because chord-to-chord distance becomes larger even though relative chord progression between the two sequences might be alike. To avoid this problem, key identification must precede. Instead of designing a sophisticated key identification algorithm, we simply estimated the key of a song to be the most frequent chord in the chord sequence, and transposed every song to a C major key before sending it to a distance computing algorithm (algo1). As an alternative, we also used the first chord to be the key (algo2).

2.2. Distance computing

[1] http://www.secondhandsongs.com/wiki/Guidelines/Cover

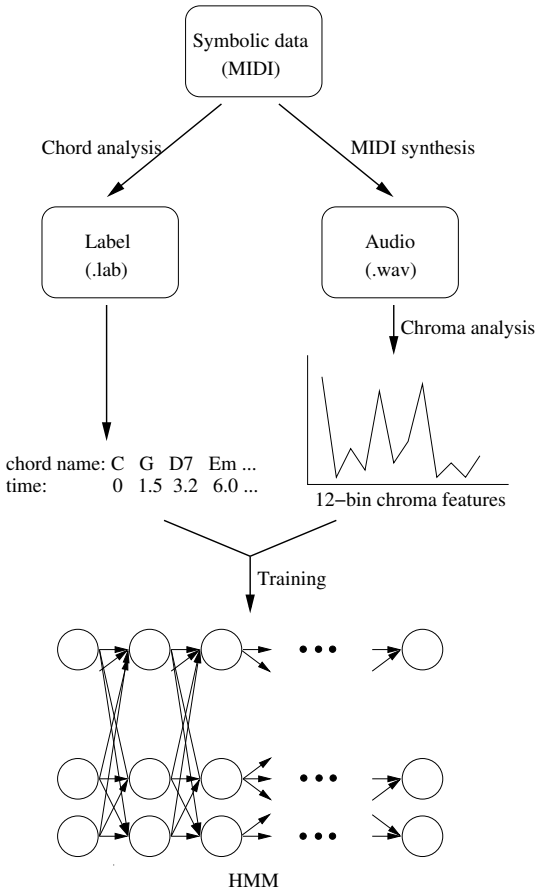


Figure 2. Overview of the chord recognition system.

After frame-level chord sequence is obtained for each song, we used the dynamic time warping algorithm (DTW) to find the minimum alignment cost between the two songs. DTW algorithm has been successfully used in the automatic speech recognition system to identify a word by finding the minimum-cost path between the input word and the word templates.

In order to use the DTW algorithm, we first need to obtain two-dimensional cost matrix from the two inputs. We defined a cost of being in aligned states by computing the chord-to-chord distance from the HMM parameters obtained above. In addition, we also defined a transition cost from the transition probability matrix in the HMM. Therefore, a combined cost at time step k for input a_k and b_k is given by

$$d(a_k, b_k) = d_S(a_k, b_k) + d_T(a_k, b_k | a_{k-1}, b_{k-1}), \qquad (1)$$

where d_S is a cost of being in aligned states, and d_T is a transition cost from the previous states to the current states. The total cost of alignment between the two sequences a and b is then given by

$$D(a, b) = \sum_{k=1}^{K} d(a_k, b_k). \qquad (2)$$

Figure3 displays the examples of the DTW for a cover-pair (on the left) and for a non-cover pair (on the right).

3. Results and analysis

3.1. Test material

Test data was composed of 30 queries with each query having 11 different cover versions including themselves. Therefore, total collection contains 30x11 = 330 songs. The collection includes a wide range of music from classical to hard rock.

3.2. Evaluation

Eight algorithms including the two by the author were submitted to the MIREX task on cover song identification. Four measures were used to evaluate the performance of the algorithms – (1) total number of cover songs identified; (2) mean number of cover songs identified; (3) mean of maxima; and (4) Mean reciprocal rank (MRR) of first correctly identified cover. Table1 shows the results using these measures.

As shown in Table1, two algorithms described in this paper are ranked at 2nd and 3rd places, respectively, using all four measures. Raw results reveal that some songs are difficult to identify for all systems while other songs are system-specific. In addition, the top four algorithms were specifically designed only for cover song identification whereas the bottom four were originally used in the similarity finding task as well. This proves that the two tasks are quite different from each other.

4. Conclusions

We proposed a system that identifies the cover songs from the raw audio using harmonic representation as a feature set and the dynamic time warping algorithm to score an alignment between the two songs abstracted through chord sequences. The rationale behind this idea was harmonic content would remain largely intact under various acoustical changes found in different versions of cover songs.

To this end, we first extracted a chord sequence from the chromagram at the frame rate using the HMM, and anchored the whole sequence to the most frequent chord or to the first chord to avoid the problem of transposition of keys. We then used the dynamic time warping algorithm to find the minimum alignment cost between a pair of chord sequences. In computing the total cost, we not only used a cost of being in aligned states but also a transition cost from chord to chord to reflect the theory of harmonic progression in Western tonal music.

Although we used our system only to recognize the cover songs, we believe it can be also used to find musical similarity since cover songs are extreme examples of similar music. Therefore, even if some songs which appear high in the list are not relevant, they might be evaluated similar to the query by human subjects, especially harmonic content is a key criterion in evaluating musical similarity.

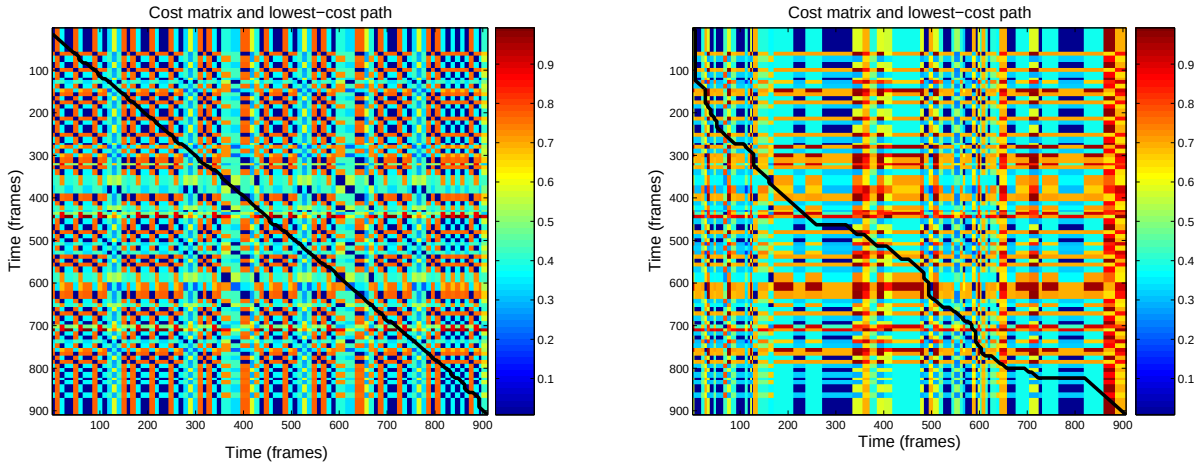


Figure 3. Examples of dynamic time warping.

Table 1. Summary results of eight algorithms.

Measure	Total number of covers identified	Mean number of covers identified	Mean of maxima	MRR of first correctly identified cover
1	D.P.W. Ellis (761)	D.P.W. Ellis (2.31)	D.P.W. Ellis (4.53)	D.P.W. Ellis (0.49)
2	**K. Lee [1] (365)**	**K. Lee [1] (1.11)**	**K. Lee [1] (2.50)**	**K. Lee [1] (0.22)**
3	**K. Lee [2] (314)**	**K. Lee [2] (0.95)**	**K. Lee [2] (2.27)**	**K. Lee [2] (0.22)**
4	Sailer & Dressler (211)	Sailer & Dressler (0.64)	Sailer & Dressler (2.13)	Sailer & Dressler (0.21)
5	Lidy & Rauber (149)	Lidy & Rauber (0.45)	Lidy & Rauber (1.57)	Lidy & Rauber (0.12)
6	K. West [1] (117)	K. West [1] (0.35)	T. Pohle (1.50)	K. West [1] (0.10)
7	T. Pohle (116)	T. Pohle (0.35)	K. West [1] (1.30)	K. West [1] (0.10)
8	K. West [2] (102)	K. West [2] (0.31)	K. West [2] (1.23)	T. Pohle (0.09)

In the future, we plan to include a melodic description in the feature set, which is another robust musical attribute that doesn't change much from cover to cover. In addition, we believe that applying the DTW to the most representative part of music will help not only increase the identification performance but also decrease the computation time to a great degree.

References

[1] C. A. Harte and M. B. Sandler, "Automatic chord identification using a quantised chromagram," in *Proceedings of the Audio Engineering Society*. Spain: Audio Engineering Society, 2005.

[2] K. Lee and M. Slaney, "Automatic chord recognition using an HMM with supervised learning," in *Proceedings of the International Symposium on Music Information Retrieval*, Victoria, Canada, 2006.

[3] ——, "Automatic chord recognition from audio using a supervised HMM trained with audio-from-symbolic data," in *Audio and Music Computing for Multimedia Workshop*, 2006.

Finding Cover Songs by Melodic Similarity

Christian Sailer and Karin Dressler
Fraunhofer IDMT
Langewiesener Str. 22
98693 Ilmenau/Germany
{sar,dresslkn}@idmt.fraunhofer.de

Abstract

This paper describes the submission to the MIREX 06 (Music Information Retrieval EXchange) Audio Cover Song task delivered by Fraunhofer IDMT. A method to detect cover songs by comparing the most salient melodies of musical pieces is proposed, based on the assumption that cover versions of one song contain the same melody. To this end, the predominant melody of the musical pieces to be considered is extracted and characteristic parts are sought for. The melodic similarity between the pieces is calculated and derived from these values, a distance matrix is constructed.

Keywords: MIREX, Audio Cover Song, melodic similarity, melody extraction, melody alignment

1. Introduction

The basic idea behind this approach is that a cover song is by definition a new rendition of a previously recorded song. As it is a version of the same song, it should carry the same melody as one of the conceptual features inherent to the musical entitiy of song. Considering this, it is even more probable that the covered song has the same melody (or some variation of it) as the original version, than having a similar sound or instrumentation. This may easily be seen by comparing e.g. Tori Amos' version of Nirvana's "Smells Like Teen Spirit" to the original.

Thus, a method to find cover versions by comparing the salient melodies of musical pieces is proposed.

2. Implementation Overview

The algorithm is implemented in C++ and is available for Linux and Windows platforms. It consists of two parts, an indexing program (a) and the actual retrieval tool (b).

(a) extracts the melody (predominant voices) from audio files, processes them to relevant pieces and stores them in a data base. The algorithm works sample rate agnostic and reads wav, aiff and mp3 files. The run time is linear with the duration of audio input and is about 10-12 times faster than real time [1] .

(b) reads the data base generated by (a) and calculates the distance matrix as is described in the requirements for the Audio Musical Similarity, respectively the Audio Cover Song task on the MIREX 2006 site [2] . As this is a quadratic matrix where each piece is compared to each other, the runtime for N pieces is $O(N^2)$.

3. Indexing

3.1. Pitch Detection

The first step of the melody extraction is using a multi resolution FFT and the algorithm proposed by Dressler in [1, 2]. It yields a voiced/non voiced detection in conjunction with a pitch line (pitch in cent over time) for the voiced parts.

3.2. Melody Quantization

In a process derived from the melody segmentation works of Heinz [3], note boundaries are estimated from the pitch line and further spectral information. The resulting note candidates are then subjected to a plausibility test where objects of too short duration or insufficient loudness are discarded. Finally, a discrete pitch is estimated for each note candidate, providing melody information on a note basis.

3.3. Relevance Weighting

There are several reasons why storing the melody over the entire length of the song is not desirable. To begin with, a lot of processing time is used for the data base search if both the query and the reference are very large. Furthermore, it is more probable to receive relevant results by matching several short excerpts of one piece against another melody instead of matching entire melodies, as structural differences within a musical piece or between versions may lead to confusing results when comparing whole melodies.

For these reasons, the extracted melodies are split into relevant pieces. In order to find such pieces, parts of the extracted melody are sought that fulfil the requirements to be a musical theme: They must be between 3 and 8 seconds long, and consist of neither too few nor too many notes [4]. Following the idea that at least in western popular music,

[1] All runtimes are measured on a 3GHz Intel Pentium IV
[2] See http://www.music_ir.org/mirex2006

important themes (like verse, chorus, etc.) are usually repeated, these theme candidates are looked up within the same piece of music and weighted according to their number of appearances within the piece.

4. Evaluation

To evaluate the indexed data, up to three excerpts of the melody are matched against all entries in the data base of indexed data, using a melody alignment algorithm that has been developed for a Query by Humming system [5].

4.1. Melody Alignment

The look-up is carried out as a string alignment process (see [6] for a general explanation), which has been adapted for melody search [7]. As basic search alphabet, the relative change of a melody over time is used. Thus, not notes but descriptions of note transitions represented by note intervals and the ratio of inter-onset intervals are considered. This frees the search algorithm from a dependency on absolute tempo and pitch – attributes that may vary between different versions of a piece of music.

The alignment is carried out as a semi-local alignment, meaning that the whole query string must match any part of the reference string. It returns a value which is the higher, the better the query matches to the reference.

In a post processing step, the contours of the matching part of the reference are matched to the contour of the query, and a correction of the alignment value is carried out.

As the resulting alignment values depend on the size of the query string (the longer the string, the greater the maximum possible value), the values are finally normalized to the range of $[0..1]$.

4.2. Distance Matrix

The melody alignment yields a melodic similarity measure s_{ab} between two melodies a and b ranging from 0 (no similarity at all) and 1 (equality). To get the distance d_{ab} as it is required by the Audio Cover Song task, all similarities where $s_{ab} = 0$ are set to $s_{ab} = 10^{-7}$, and then the distance is calculated as $d_{ab} = \frac{1}{s_{ab}} - 1$, thus being a non-negative number between 0 and 10^7.

5. Results and Discussion

An overview of the results [3] of the Audio Cover Song evaluation can be found in table 5. The test bed consists of a set of about 5000 songs, embedded into which are 30 songs along with 10 cover versions of each of these song.

Each of these 330 versions is used as query and the ten most similar songs returned are evaluated

As can easily be seen, the submitted algorithm yields a higher success rate on dectecting cover versions than the musical similarity algorithms evaluated for this task. Considering the fact that cover versions do not necessarily have the same sound, and often are intentionally rearranged, this may not be a big surprise. On the other hand, it yielded lower success rates than algorithms that have been carefully crafted to detect cover versions based on an extensive structural and musical analysis of the audio tracks.

Table 1. Overview of the results of the Audio Cover Song task, this submission in bold. Entries marked with * are musical similarity algorithms. The first result line shows total cover versions found out of 3300 possible, the second line shows inverse average rank of best cover version

CS	DE	KL1	KL2	KWL*	KWT*	LR*	TP*
211	761	365	314	117	102	149	116
0.21	0.49	0.22	0.22	0.10	0.10	0.12	0.09

Several problems could be identified for this algorithm: Considering just one feature, in this case melody, makes the algorithm vulnerable to extraction errors as well as to cases where melody is not the key feature of a song or has been altered in the cover version. Additionally, the melody extraction, though state of the art, is far from being perfect. The note segmentation required for the qbh algorithms to work is highly dependent on the quality of pitch extraction and furthermore adds its own errors. Finally, the segmentation of the melody in theme like pieces is based solely on the melody information previously extracted in this approach, which leads to spurious results in some cases. Recapitulating, it becomes obvious that error propagation from the first steps, promoted by a strong dependency on exact inputs by later stages, diminishes the overall performance of the system.

6. Conclusive Remarks

With this submission, it could be shown that melody is an important feature for detecting cover versions, and that automatically extracted melodies can be a handy tool for this task. However, it also became clear that automatically extracted melodies alone are not sufficient to conduct this task satisfactorily. Besides optimising the melody extraction and segmentation, an interesting prospect will be the integration of such a melody similarity assessment into a larger, multi feature system, using melody as one feature as well as using a song segmentation for detecting the relevant musical pieces.

7. Acknowledgements

We would like to thank the ISMIRSEL team for the organization and running of this evaluation.

Work on the Query by Humming algorithms used in this approach was partially funded by the EU-Project Semantic

[3] Please see http://www.music-ir.org/mirex2006/index.php/MIREX2006_Results for full results and further information on contestants

Hifi (FP6-507913) [4] . Karin Dressler's work on melody extraction was partially funded by a scholarship of the state of Thuringia.

References

[1] K. Dressler, "Sinusoidal extraction using an efficient implementation of a multi-resolution FFT," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 247–252.

[2] ——, "Audio melody submission," in *MIREX 06*, 2006.

[3] T. Heinz, "Ein physiologisch gehörgerechtes verfahren zur automatisierten melodietranskription," Ph.D. dissertation, Technische Universität Ilmenau, 2006.

[4] D. Temperley, *The Cognition of Basic Musical Structures*. The MIT Press, 2001.

[5] C. Sailer, "Two note based approaches to query by singing/humming," in *MIREX 06*, 2006.

[6] D. Gusfield, *Algorithms on Strings, Trees and Sequences*. Cambrigde University Press, 1997.

[7] C. Sailer, "Using string alignment in a query-by-humming system for real world applications," Talk at 150th ASA Fall Meeting, October 2005.

[4] See http://shf.ircam.fr

An Auditory Streaming Approach on Melody Extraction

Karin Dressler
Fraunhofer Institute for Digital Media Technology
Langewiesener Str. 22
98693 Ilmenau, Germany
dresslkn@idmt.fraunhofer.de

Abstract

The MIREX (Music Information Retrieval Evaluation eXchange) framework provides a common set of data to evaluate and compare a vast variety of MIR systems. This paper describes our submission to the audio melody extraction evaluation addressing the task of identifying the melody pitch contour from polyphonic musical audio. It shall give an overview about the used methods and a discussion of the evaluation results. The presented algorithm is a derivative of our submission to MIREX'05. Therefor we will outline changes between the two versions and discuss the impact of the further developments.

The MIREX 2006 evaluation results show that our algorithm performs best in pitch detection and melody extraction.

Keywords: MIREX 2006, audio melody extraction.

1. METHOD

1.1. Spectral Analysis

A multi resolution spectrogram representation is obtained from the audio signal by calculating the Short-Term Fourier Transform (STFT) with different factors of zero padding using a Hann window. Thereby we make use of a Multi Resolution FFT – an efficient technique used to compute STFT spectra in different time-frequency resolutions [1]. For all spectral resolutions – assuming audio data sampled at 44.1 kHz – the resulting STFT frame size and the hop size of the analysis window are 2048 and 256 samples, respectively. This processing step is followed by the computation of the magnitude and phase spectra.

To gain a better frequency discrimination, the instantaneous frequency (IF) is estimated from successive phase spectra. We apply the well-known phase vocoder method proposed by [2] for the IF extraction.

1.2. Peak Selection

Sinusoidal components of the audio signal contain the most relevant information about the melody. Yet, it is a challenge to reliably identify sinusoidal partials in polyphonic music. Of course a consistent and moderate change in magnitude and frequency of the examined spectral peaks is a good criterium for the identification of sinusoidals. However, this requires a continuous tracking of partials with time, a demand which cannot be implemented easily for polyphonic audio signals.

Charpentier found that you can identify sinusoidals by distinct spectral features in one FFT frame alone [3]. We developed his method further and this way improved the performance and robustness of the adjacent pitch estimation noticeably. Nevertheless this efficient method is not adequate for audio signals with a dense spectrum, because it relies on a non distorted phase spectrum around spectral peaks. This is not the case for closely adjoining partials, which will be erroneously identified as noise and will be discarded from further analysis. For this reason we employed a psychoacoustic model in this year's application in contrast to the local sinusoidality criterion we applied in our submission to MIREX'05 [4].

Unlike the before-mentioned sinusoidality criterion, psychoacoustic masking is a method to exclude non audible peaks - sinusoidal or not - from further processing. We use a very simplified implementation of simultaneous and temporary masking, which by far does not reach the complexity of models used in modern lossy audio coders. However, this way many unprofitable peaks can be erased from the spectrum in order to speed up the further processing.

1.3. Pitch Estimation

The magnitude and instantaneous frequency of the sinusoids are evaluated by a pitch estimation method, as the frequency of the strongest harmonic may not be the perceived pitch of a periodic complex tone. At first, the pitch estimator performs a magnitude weighting and then it analyzes the harmonic structure of the polyphonic signal. The algorithm covers four octaves – computing pitch frequencies and an approximate prediction of the pitch salience in a frequency range between 80 Hz and 1280 Hz. A variable number of pitch candidates at each frame (about five pitches on average) is used to track tone objects.

1.4. Auditory Streaming

At the same time the frame-wise estimated pitch candidates are processed to build acoustic streams. Tones which have

a sufficient magnitude and are located in an adequate frequency range are assigned to the corresponding streams. Anyhow, every stream may possess only one active tone at any time. So in competitive situations the active tone is chosen with the help of a rating method that evaluates the tone magnitude and the frequency difference between the pitch of the tone and the actual stream position. Conversely, a tone is exclusively linked to only one stream.

This is a new concept compared to the method we used in last year. There, all tone objects lasting longer than 100 ms were grouped according to their frequency range and stored in different registers. Then all tone objects belonging the most energetic frequency range gained an additional weight in the concluding comparison. Yet, essentially any tone from any frequency region – even outside the most energetic frequency range – could win the final comparison and become part of the melody.

This is not the case in this year's algorithm where only tones from the most salient stream are considered to be melody tones. Therefor, the correct identification of the melody stream is very important for the success of the method!

1.5. Identification of the Melody Stream

Finally, the melody stream must be chosen. In general the most salient stream is identified as the melody. Of course it may happen that two ore more streams have about the same magnitude and thus no clear decision can be taken. In this case, the stream magnitudes are weighted according to their frequency. Streams from the bass region receive a lower weight than streams from the mid and high frequency regions. If no clear melody stream emerges during a short time span, the most salient weighted stream is chosen.

2. Implementation

The algorithm is implemented in C++ and is available for Windows and Linux platforms. The performance of the algorithm varies slightly depending on the complexity of the audio input. The reported execution time for the MIREX 2006 test sets, which consist of 45 audio pieces with an overall length of 1053 seconds, is 75 seconds. Thus the audio analysis is approximately 14 times faster than real-time on an AMD Athlon XP 2600+1.9GHz CPU system with 2 GB RAM – the fastest runtime among all submissions. The implementation is suitable for the instant processing of an audio stream, although with a latency of 250ms up to 4s this implementation is not suitable for real-time processing. However, the allowed latency may be decreased to a minimum value of about 25ms. Of course such a small latency will noticeably decrease the overall accuracy of the algorithm.

3. MIREX Evaluation

3.1. Evaluation Overview

The aim of the MIREX Audio Melody Evaluation is to extract melodic content from polyphonic audio. Two datasets were available for the evaluation this year. The MIREX 2005 dataset contains 25 phrase excerpts of 10-40 seconds length from the following genres: Rock, R&B, Pop, Jazz, Solo classical piano. The same data was used for the MIREX 2005 audio melody contest. This way a direct comparison between the evaluation 2005 and this year's evaluation is possible. For the ISMIR 2004 Audio Description Contest, the Music Technology Group of the Pompeu Fabra University assembled a diverse set of 20 polyphonic musical audio pieces and corresponding melody transcriptions including MIDI, Jazz, Pop and Opera music as well as audio pieces with a synthesized voice. Each file has an approximate duration of 20 seconds [1] .

The audio excerpts are provided as single channel PCM data in CD-quality (16-bit resolution, 44.1 kHz sample rate). The corresponding reference annotations of the predominant melody include a succession of pitch frequency estimates at discrete time instants (5.8/10 ms grid). Zero frequencies indicate periods without melody. The estimated frequency was considered correct whenever the corresponding ground truth frequency is within a range of 50 cents.

To maximise the number of possible submissions the transcription problem was divided into two subtasks, namely the melody pitch estimation and the distinction of melody and non-melody parts (voiced/unvoiced detection). It was possible to give a pitch estimate even for those parts, which have been declared unvoiced. Those frequencies are marked with a negative sign. Moreover, each dataset was divided into a vocal and a non-vocal melody voice subset.

3.2. Results

The evaluation results show that our algorithm performs best in pitch detection and melody extraction [2] . As indicated by the excellent runtime of our algorithm, the implemented methods allow a very efficient computation of the melody pitch contour.

Table 1 shows that the overall accuracy varies significantly among the submissions. However, we must not forget that quite different approaches are compared. In contrast to the other transcription systems, Poliner and Ellis present a classification-based system that uses no assumptions about the physical nature of sound [5]. Brossier aims at real-time processing with a very short latency [6]. Sutton et al have built a system that is only suitable for singing voice extraction [7]. So naturally their system performs better for the vocal pieces. Ryynänen and Klapuri use a general approach with a parameter setting especially tuned for the transcription of the singing voice [8]. For the given vocal examples,

[1] The data set including the reference annotations can be found on the contest web page http://ismir2004.ismir.net/melody_contest/results.html

[2] Detailed evaluation results can be found at http://www.music-ir.org/mirex2006/index.php/Audio_Melody_Extraction_Results

Table 1. 2006 MIREX Audio Melody Extraction Results

Dataset	Participant	Voicing Recall	Voicing False Alm	Voicing d-prime	Raw Pitch	Raw Chroma	Overall Accuracy	Runtime (s)
ISMIR 2004	Dressler	90.9%	10.5%	2.58	82.9%	84.0%	82.5%	27
	Ryynänen & Klapuri	84.4%	12.6%	2.16	80.6%	82.3%	77.3%	440
	Poliner & Ellis	89.9%	36.3%	1.63	73.2%	76.4%	71.9%	-
	Sutton et al	73.2%	24.9%	1.30	62.6%	65.4%	58.2%	5014
	Brossier	99.7%	88.4%	1.61	57.4%	68.7%	49.6% *	30
MIREX 2005	Dressler	89.3%	28.8%	1.80	77.7%	82.0%	73.2%	48
	Dressler (2005)	*81.8%*	*17.3%*	*1.85*	*68.1%*	*71.4%*	*71.4%*	*32*
	Ryynänen & Klapuri	78.2%	16.5%	1.75	71.5%	75.0%	67.9%	773
	Poliner & Ellis	93.5%	45.1%	1.64	66.2%	70.4%	63.0%	-
	Sutton et al	64.5%	13.8%	1.46	56.4%	60.1%	53.7%	8195
	Brossier	99.5%	98.2%	0.46	41.0%	56.1%	31.9% *	58

Note: * Brossier did not perform voiced/unvoiced detection, so the overall accuracy cannot be meaningfully compared to other systems.

no significant difference can be noted between the accuracy of this implementation and our submission.

All resubmitted algorithms have improved the overall accuracy compared to the results of MIREX'05. As we can see in table 1 (where the submission of last year is marked by italic font), our melody extraction algorithm has gained 1.8% in overall accuracy for the MIREX 2005 dataset. The improvements for the raw pitch and raw chroma estimation seem even more pronounced. Yet, a part of this increased accuracy has to be attributed to the use of the negative frequency output, which has not been used last year.

4. Acknowledgements

Many thanks to the IMIRSEL team at the at the University of Illinois at Urbana-Champaign for running the MIREX evaluations.

This research was partially founded by the state of Thuringia, Germany (*Landesgraduiertenförderung Thüringen*).

References

[1] K. Dressler, "Sinusoidal extraction using an efficient implementation of a multi-resolution FFT," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Quebec, Canada, Sept. 18–20, 2006, pp. 247–252.

[2] J. L. Flanagan and R. M. Golden, "Phase vocoder," *Bell System Technical Journal*, pp. 1493–1509, 1966.

[3] F. J. Charpentier, "Pitch detection using the short-term phase spectrum," in *Proc. of ICASSP 86*, 1986, pp. 113–116.

[4] K. Dressler, "Extraction of the melody pitch contour from polyphonic audio," in *1st Music Information Retrieval Evaluation eXchange (MIREX)*, 2005, available online http://www.musicir.org/evaluation/mirex-results/articles/melody/dressler.pdf.

[5] G. E. Poliner and D. P. W. Ellis, "Classification-based melody transcription," *Machine Learning Journal*, 2006.

[6] P. Brossier, J. P. Bello and M. D. Plumbley, "Real-time temporal segmentation of note objects in music signals," in *Proceedings of the International Computer Music Conference (ICMC 2004)*, Florida, USA, 2004.

[7] C. Sutton, E. Vincent, M. D. Plumbley and J. P. Bello, "Transcription of vocal melodies using voice characteristics and algorithm fusion," in *2nd Music Information Retrieval Evaluation eXchange (MIREX)*, 2006, available online http://www.music-ir.org/evaluation/MIREX/2006_abstracts/AME_sutton.pdf.

[8] M. P. Ryynnen and A. P. Klapuri, "Transcription of the singing melody in polyphonic music," in *Proc. 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006.

Transcription of the Singing Melody in Polyphonic Music (MIREX 2006)

Matti Ryynänen and Anssi Klapuri
Institute of Signal Processing, Tampere University Of Technology
P.O.Box 553, FI-33101 Tampere, Finland
{matti.ryynanen, anssi.klapuri}@tut.fi

Abstract

We introduce a method for the MIREX 2006 "Audio Melody Extraction" task in which the goal is to estimate fundamental frequency (F0) trajectory of the main melody within polyphonic music. The introduced method is based on multiple-F0 estimation followed by acoustic and musicological modeling. The acoustic model consists of separate models for melody notes and for no-melody segments. The musicological model uses key estimation and note bigrams to determine the transition probabilities between notes. Viterbi decoding produces a sequence of notes and rests as a transcription of the melody. The method details are published in ISMIR 2006 proceedings. As an extension to this method, we use a simple F0 estimate selection to produce the required F0 trajectory for the task evaluation. Although the method was developed for the automatic transcription of singing melodies in polyphonic music, it is also applicable in general melody transcription tasks.

1. Introduction

Singing melody transcription refers to the automatic extraction of a parametric representation (e.g., a MIDI file) of the singing performance within a polyphonic music excerpt. A melody is an organized sequence of consecutive notes and rests, where a note has a single pitch (a note name), a beginning (onset) time, and an ending (offset) time.

Recently, melody transcription has become an active research topic. The conventional approach is to estimate the F0 trajectory of the melody within polyphonic music, such as in [1], [2], [3], [4]. Another class of transcribers produce discrete notes as a representation of the melody [5], [6]. The introduced method belongs to the latter category, and it is published in [7]. Here the method is, however, extended with a post-processing step of F0 selection so that the required output format is produced for the MIREX evaluation.

Figure 1 shows a block diagram of the proposed method. First, an audio signal is frame-wise processed with two feature extractors, including a multiple-F0 estimator and an accent estimator. The acoustic modeling uses these features

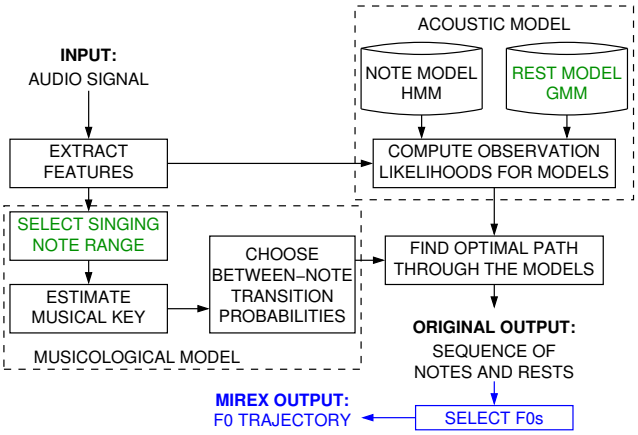


Figure 1. The block diagram of the transcription method. The blue section indicates an extension to the method for the MIREX style output. The green sections indicate the main changes to the system compared to our MIREX 2005 method.

to derive a hidden Markov model (HMM) for note events and a Gaussian mixture model (GMM) for singing rest segments. The musicological model uses the F0s to determine the note range of the melody, to estimate the musical key, and to choose between-note transition probabilities. A standard Viterbi decoding finds the optimal path through the models, thus producing the transcribed sequence of notes and rests. The decoding simultaneously resolves the note onsets, the note offsets, and the note pitch labels. The proposed method resembles our polyphonic music transcription method [8] and our MIREX 2005 method but now it has been tailored for singing melody transcription and includes improvements, such as an acoustic model for rest segments in singing and singing note range selection. These are indicated in green in Fig. 1.

As an extension to the system, we use a simple selection of F0s in the vicinity of each transcribed note to produce the required output. This is indicated with the blue section in Fig. 1.

2. Method Description

We briefly introduce the method in the following. For more details, please see [7].

2.1. Feature Extraction

The front-end of the method consists of two frame-wise feature extractors: a multiple-F0 estimator and an accent estimator. We use the multiple-F0 estimator proposed in [9] in a fashion similar to [8]. The estimator applies an auditory model where an input signal is passed through a 70-channel bandpass filterbank and the subband signals are compressed, half-wave rectified, and lowpass filtered. STFTs are computed within the bands and the magnitude spectra are summed across channels to obtain a summary spectrum for subsequent processing. Periodicity analysis is then carried out by simulating a bank of comb filters in the frequency domain. F0s are estimated one at a time, the found sounds are canceled from the mixture, and the estimation is repeated for the residual.

There was room for improvement in the note-onset transcription of [8], and the task is even more challenging for singing voice. Therefore, we add the accent signal feature which has been successfully used in singing transcription [10]. We apply the accent estimation method proposed in [11].

2.2. Acoustic and Musicological Modeling

Our method uses two different abstraction levels to model melodies: low-level acoustic modeling and high-level musicological modeling. The acoustic modeling aims at capturing the acoustic content of singing whereas the musicological model employs information about typical melodic intervals.

2.2.1. Acoustic Models

Note events are modeled with a 3-state left-to-right HMM. The model allocates one note HMM for each MIDI note in the estimated note range (see Fig. 2). We use a GMM for modeling the time segments where no singing-melody notes are sounding, that is, rests. For training the note and rest models, we use the RWC (Real World Computing) Popular Music Database which consists of 100 acoustic recordings of typical pop songs with annotated melodies [12].

2.2.2. Musicological Modeling

The note range estimation aims at constraining the possible pitch range of the transcribed notes. Since singing melodies usually lie within narrow note ranges, the selection makes the system more robust against spurious too-high notes and the interference of prominent bass line notes. This also reduces the computational load due to the smaller amount of note models that need to be evaluated. The note range is determined from the estimated F0s.

The musicological model controls transitions between the note models and the rest model in a manner similar to that used in [8]. The musicological model first finds the most probable relative-key pair using a musical key estimation method [10]. The relative-key pair is then used to choose the note bigram probabilities estimated from a large database of monophonic melodies.

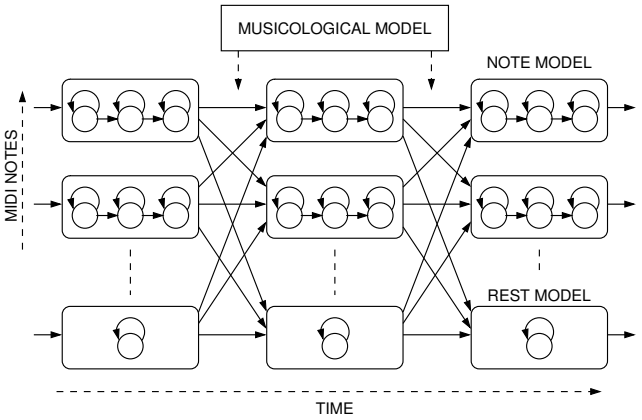


Figure 2. The network of note models and the rest model.

2.2.3. Finding the Optimal Path

The note event models and the rest model form a network of models where the note and rest transitions are controlled by the musicological model. This is illustrated in Figure 2. We use the Viterbi algorithm to find the optimal path through the network to produce a sequence of notes and rests, i.e., the transcribed melody. Notice that this simultaneously produces the note pitch labels, the note onsets, and the note offsets.

2.3. Determining the F0 Trajectory for MIREX 2006

The remaining task is to determine F0s in every frame of a 10 ms grid. This is done based on the note-level transcription. Since we have the transcribed MIDI notes, we simply select the frame-wise F0s which were associated to the note during the transcription. If the absolute difference between the note model and an associated F0 is more than two semitones, we use the center frequency of the MIDI note instead. Then we use linear interpolation to output the F0s at every 10 ms. For the rest segments, we output the most prominent F0 estimates which lie on the estimated note range as the *unvoiced* F0 estimates (i.e., negative F0 values in the output), or zero if no such value is found.

Transcribed notes may end slightly too early due to salience decrease typical during note endings. If some unvoiced F0s immediately continue the F0 trajectory of the transcribed note, those unvoiced F0s are converted into voiced estimates. By starting from the end of a note, unvoiced F0 estimates are frame-by-frame converted to voiced if the absolute difference between consecutive estimates is less than 0.5

Figure 3 shows the method output for an excerpt in MIREX 2004 dataset. The green circles indicate the annotated F0s (in MIDI note numbers). The grey boxes shows the transcribed sequence of notes and rests. The actual output of the method is the voiced F0 estimates (the blue dots). In addition, the unvoiced F0 estimates (negative F0 values in

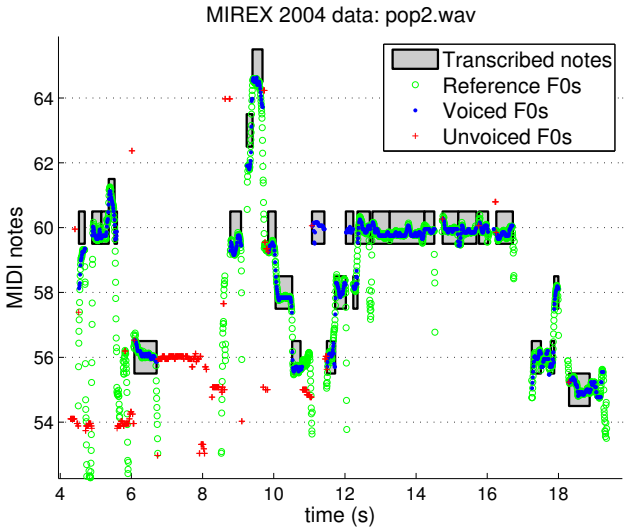


Figure 3. The method output for `pop2.wav` in MIREX 2004 dataset. See text for details.

the output file) are shown in red. The last note shows an example of note extension where some unvoiced F0s have been changed to voiced since they naturally continue the F0 trajectory of the note.

3. About the Implementation

The method has been implemented as Matlab M-files and MEX-files, and it should run in Linux Matlab versions 6.5 and 7.2. The execution time on a 1.7 GHz Linux PC is about twice the real-time without any particular optimizations.

4. Evaluation Results

The method performed second best in the evaluations. Since the method was developed for singing transcription, it performed better for vocal melodies than non-vocal melodies. Table 1 compares the performance of our methods in 2005 and 2006 for the "MIREX 2005 Dataset - All". The new method works better than the old one with these criteria. In particular, the rest modeling improves the voicing detection, clearly indicated by "Voicing d-prime" and "Vx False Alarm" rates. In addition, the 2006 method is considerably faster due to a faster multiple-F0 estimation method.

The method was developed for singing note transcription (not for singing F0-estimation), and the difference is more explicit with the criteria for discrete note events used in [7]. See Table 1 in [7] for comparison.

Table 1. Comparison of our methods from 2005 and 2006 for "MIREX 2005 Dataset - All".

Criterion	2005	2006
Overall Accuracy	64.3	67.9
Raw Pitch Accuracy	68.6	71.5
Raw Chroma Accuracy	74.1	75.0
Vx Recall	90.3	78.2
Vx False Alarm	39.5	16.5
Voicing d-prime	1.56	1.75
Runtime (s) (suggestive)	10970	773

References

[1] J. Eggink and G. J. Brown, "Extracting melody lines from complex audio," in *Proc. 5th International Conference on Music Information Retrieval*, Oct. 2004.

[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*, vol. 43, no. 4, pp. 311–329, 2004.

[3] M. Marolt, "Audio melody extraction based on timbral similarity of melodic fragments," in *Proc. EUROCON 2005*, Nov. 2005.

[4] K. Dressler, "Extraction of the melody pitch contour from polyphonic audio," in *Proc. 6th International Conference on Music Information Retrieval*, Sept. 2005. MIREX05 extended abstract, available online http://www.music-ir.org/evaluation/mirex-results/articles/melody/dressler.pdf.

[5] G. E. Poliner and D. P. W. Ellis, "A classification approach to melody transcription," in *Proc. 6th International Conference on Music Information Retrieval*, pp. 161–166, Sept. 2005.

[6] R. P. Paiva, T. Mendes, and A. Cardoso, "On the detection of melody notes in polyphonic audio," in *Proc. 6th International Conference on Music Information Retrieval*, pp. 175–182, Sept. 2005.

[7] M. Ryynänen and A. Klapuri, "Transcription of the singing melody in polyphonic music," in *Proc. 7th International Conference on Music Information Retrieval*, (Victoria, Canada), Oct. 2006.

[8] M. P. Ryynänen and A. Klapuri, "Polyphonic music transcription using note event modeling," in *Proc. 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 319–322, Oct. 2005.

[9] A. Klapuri, "A perceptually motivated multiple-F0 estimation method," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 291–294, Oct. 2005.

[10] M. Ryynänen, "Singing transcription," in *Signal Processing Methods for Music Transcription* (A. Klapuri and M. Davy, eds.), pp. 361–390, Springer Science + Business Media LLC, 2006.

[11] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 342–355, Jan. 2006.

[12] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical, and jazz music databases," in *Proc. 3rd International Conference on Music Information Retrieval*, Oct. 2002.

Transcription of vocal melodies using voice characteristics and algorithm fusion

Christopher Sutton, Emmanuel Vincent, Mark D. Plumbley
Centre for Digital Music
Queen Mary, University of London
Mile End Road, London E1 4NS, UK
christopher.sutton@cantab.net

Juan P. Bello
The Steinhardt School
New York University
35 W 4th Street, New York NY 10012, USA
jpbello@nyu.edu

Abstract

This paper deals with the transcription of vocal melodies in music recordings. The proposed system relies on two distinct pitch estimators which exploit characteristics of the human singing voice. A Hidden Markov Model (HMM) is used to fuse the pitch estimates and make voicing decisions. The resulting performance is evaluated on the MIREX 2006 Audio Melody Extraction data.

Keywords: Melody, singing voice, algorithm fusion.

1. Introduction

A key goal of digital music research is the automatic transcription of polyphonic music recordings. Systems seeking to perform full transcription have met with limited success so far. Higher transcription accuracy has been obtained by systems seeking to perform only a partial transcription consisting of the chord sequence, the drum track or the melody.

The melody of a piece of music is generally defined as the sequence of notes played by the lead instrument, but this leaves considerable ambiguity since the factors determining which instrument is the "lead" to a human listener are somewhat subjective and ill-defined. The fact that the raw pitch accuracy scores reported in the MIREX 2005 Audio Melody Extraction evaluation were considerably lower than for monophonic recordings suggests that the systems entered struggled to consistently identify the lead instrument.

In this paper, we aim to avoid this ambiguity by focusing on the case where melody is carried by the main vocal line, which is better defined objectively. Unlike standard transcription systems based on a single pitch estimator, the proposed system relies on two distinct pitch estimators which exploit characteristics of the human singing voice. A HMM is used to produce the final transcription by fusing the pitch estimates and making voicing decisions.

Useful voice characteristics are described in Section 2, followed in Section 3 by details of the system's design. The resulting performance is evaluated in Section 4 and conclusions are given in Section 5.

2. Characteristics of the human singing voice

To avoid transcribing non-vocal instruments, the proposed system exploits two salient characteristics of singing voice: *pitch instability* and *high-frequency dominance*.

2.1. Pitch instability

Pitch instability refers to the property of the singing voice that its pitch varies considerably over time compared with other pitched instruments. This is mostly due to the fact that *vibrato* typically exhibits an extent of $\pm$60–200 cents for singing voice and only $\pm$20–35 cents for other instruments [1]. Also, vocalists almost always sing *legato*, changing pitch smoothly during note attacks and transitions.

This characteristic has been exploited recently by a vocal detection system [2]. After identification of the musical key, the system filters the input audio by an inverse comb filter which attenuates all the harmonic partials of the seven notes in the key. Since vocal notes are rarely at exactly the intended pitch, their partials survive this process while other pitched instruments are attenuated.

2.2. High-frequency dominance

High-frequency dominance refers to the property of the singing voice that the power of its upper partials is larger than with other instruments. This has been observed in a study on vocal melody transcription [3], where the high frequency (over 800Hz) channels of a correlogram led to more accurate vocal pitch estimates than the low frequency channels.

We further investigated this effect in [4]. Figure 1 shows the minimum, mean and maximum reliability of correlogram channels for the estimation of vocal pitch over a range of recordings, where reliability is defined as the proportion of resulting pitch estimates within 50 cents of the ground truth. The recordings used were the nine training files for the MIREX 2005 Audio Melody Extraction evaluation featuring singing voice as lead instrument. The figure demonstrates that channels in the 3–15kHz range provide more reliable vocal pitch estimates than other channels.

3. Proposed System

Experimentally, the voice characteristics described above are difficult to combine into a single standard pitch estimator. Therefore we adopt a novel approach for melody transcription, in which multiple transcriptions produced by parallel estimators are fused into a single transcription, hope-

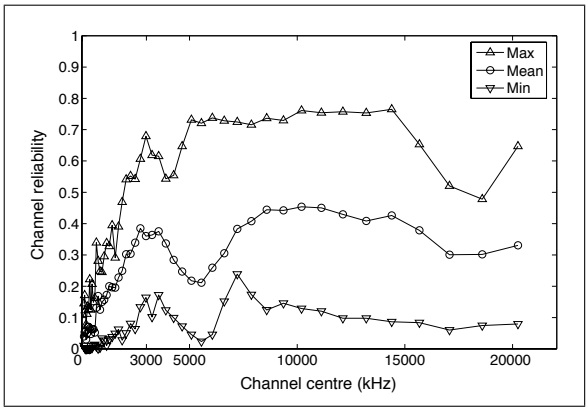


Figure 1. Reliability of correlogram frequency channels

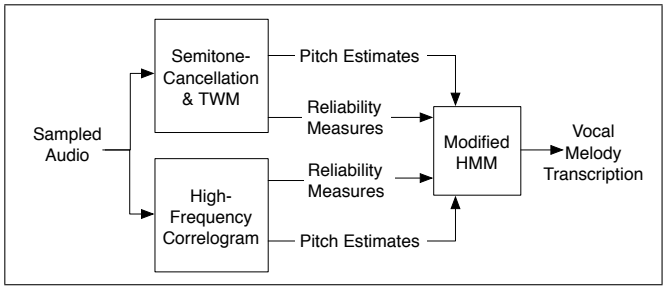


Figure 2. Diagram of the proposed system

fully more accurate than using any one of the estimators. In the following, two pitch estimators are used, but the system design and the fusion method generalise to a larger number of estimators.

The system diagram is shown in Figure 2. The input audio is processed by two pitch estimators, each producing a series of pitch estimates and associated *reliability measures* at 10 ms intervals. These values are then input to a HMM system to produce a single series of pitch estimates, with unvoiced segments represented by 0 Hz estimates.

3.1. Semitone-cancellation & TWM

The first vocal pitch estimator consists of a pre-processing stage in which a semitone-cancellation procedure emphasises the vocals, followed by the standard Two-Way Mismatch (TWM) [5] monophonic pitch transcription algorithm.

3.1.1. Semitone-cancellation procedure

Experimentally, we found that the non-vocal cancellation procedure proposed in [2] was too destructive of vocal pitch and did not allow accurate pitch estimation. Thus, instead of eliminating all the harmonic partials of interfering notes, we eliminate fundamental frequencies only. Since most music contains notes not in the musical key, the key detection stage is discarded and all semitone notes are eliminated. Based on the relative *vibrato* extent of vocals and other instruments (see Section 2.1), the bandwidth of the cancellation filters is set to ± 20 cents. This process is implemented in the frequency domain by zeroing suitable FFT bins [4]. Since most

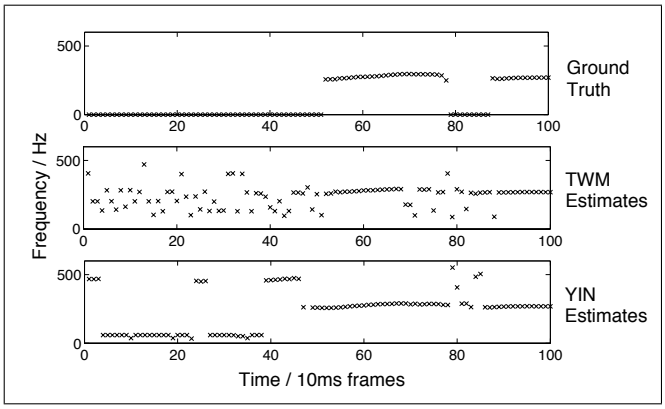


Figure 3. Pitch estimates after semitone-cancellation

partials of non-vocal notes survive this procedure, the output is finally bandpass-filtered to 300–2000Hz, which roughly corresponds to the pitch range of the human singing voice.

3.1.2. Pitch estimation by TWM

Informal listening tests show that the output of the semitone cancellation procedure is generally dominated by vocals, with components from other instruments being unpitched or much quieter. Thus it is feasible to transcribe vocal pitch by passing this output to a monophonic transcription algorithm. This algorithm should favour predominant partials on voiced frames to achieve high pitch accuracy. Since the fusion system (see Section 3.3) favours pitch continuity, it should also produce scattered pitch estimates on unvoiced frames to achieve high voicing detection accuracy. The TWM algorithm was chosen, as it offers a good compromise between these two objectives [1] . Other algorithms were found to generally transcribe weak instrumental notes on unvoiced frames [4], as illustrated in Figure 3.

3.1.3. Reliability measure

In order to assess which TWM pitch estimates are likely to be correct, each estimate is further associated with a reliability measure. This measure is obtained simply by mapping the TWM error [5] linearly to the interval $[0, 1]$.

3.2. High-frequency correlogram

The second vocal pitch estimator consists of a correlogram-based monophonic pitch transcription algorithm using only certain channels where the voice is likely to be predominant.

3.2.1. Correlogram design

The input audio is filtered by a 50-channel gammatone filterbank spanning the range 100Hz–22kHz [2] . The unbiased autocorrelation function (ACF) of each channel is computed in 50ms frames at 10ms intervals. The predominant pitch is then estimated in each channel and each frame by summing

[1] We used the implementation described in U. Zölzer, editor. *DAFX : Digital Audio Effects*. Wiley, 2002.

[2] This filterbank was implemented using the Auditory Toolbox available at http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/

the ACF value at the first three multiples of each integer lag in the singing voice range (1–12.5 ms) and picking the lag resulting in the largest sum. We found this method more reliable than full harmonic comb matching of the ACF.

3.2.2. High-frequency bias

Based on the channel reliability measures computed in Section 2.2, only 19 correlogram channels in the range 3–15kHz are used. The vocal pitch is then estimated for each time frame by clustering together channel-wise pitch estimates within 50 cents of each other and selecting the cluster with largest population. Experimentally, this approach provides the desired behaviour of accurate pitch estimates on voiced frames and scattered estimates on unvoiced frames. Other transcription algorithms applied to the input audio bandpass-filtered to 3–15kHz also produced scattered estimates on unvoiced frames, but achieved lower pitch accuracy [4].

3.2.3. Reliability measure

As above, each estimate is associated with a reliability measure. In this case, we wish to mark estimates as reliable when there is a strong consensus among correlogram channels. Thus reliability is defined as the proportion of channel-wise estimates belonging to the selected cluster.

3.3. Modified HMM

The fusion system is based on a HMM in which the hidden states represent the exact pitch sung, and the observed data are the pitch estimates and reliability measures from the two estimators described above. The Viterbi algorithm is used to produce the output transcription.

3.3.1. Dynamic state generation

Rather than defining an infinite number of hidden states to model continuous frequency, the states of the HMM are defined dynamically based on the input pitch estimates. With K pitch estimates $\{e_{k,t}\}_{1\le k\le K}$ at time t, the set of $(K+1)$ states is defined by $\Omega_t = \{\omega_{j,t}\}_{0\le j\le K}$ where

$$\omega_{j,t} = \begin{cases} \text{unvoiced} & \text{if } j = 0, \\ e_{j,t} & \text{if } 1 \le j \le K. \end{cases} \quad (1)$$

The notations $\omega_{j,t}$ and $e_{j,t}$ refer both to states and observations and to their assigned frequency values. The proposed system uses two pitch estimators, and so $K = 2$.

To avoid transcription errors when both estimators briefly fail, an additional *dummy state* is generated at time t for each state at $t-1$ for which there is no nearby estimate at t. More precisely, a state with frequency $\omega_{j,t-1}$ is added to Ω_t if there is no k for which $e_{k,t}$ is within 50 cents of $\omega_{j,t-1}$. A *pruning* process is introduced in the Viterbi algorithm to prevent such states persisting indefinitely [4].

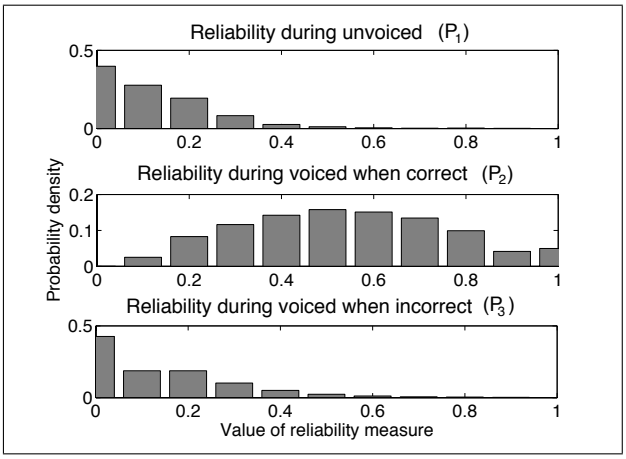


Figure 4. Distributions of High-Frequency Correlogram reliability measures

3.3.2. Observation probabilities

Unlike previous post-processing HMM systems [6], the proposed system considers all pitch estimates $\{e_{k,t}\}_{1\le k\le K}$ and reliability measures $\{r_{k,t}\}_{1\le k\le K}$ when calculating the observation probability of a given state $\omega_{j,t}$, defined by

$$P(\{e_{k,t}\}_{1\le k\le K}, \{r_{k,t}\}_{1\le k\le K} \mid \omega_{j,t}) = \prod_{k=1}^{K} P(e_{k,t}, r_{k,t} \mid \omega_{j,t}). \quad (2)$$

Each per-estimate observation probability $P(e_{k,t}, r_{k,t} \mid \omega_{j,t})$ is calculated using one of three probability distributions specific to the corresponding estimator k, depending on whether the state is voiced or unvoiced and whether the difference $d = 1200 \times |\log_2(\omega_{j,t}/e_{k,t})|$ between state and estimate frequencies is larger than 50 cents:

$$P(e_{k,t}, r_{k,t} \mid \omega_{j,t}) = \begin{cases} P_{1,k}(r_{k,t}) & \text{if } j = 0, \\ P_{2,k}(r_{k,t}) & \text{if } j \ne 0 \text{ and } d \le 50, \\ P_{3,k}(r_{k,t}) & \text{if } j \ne 0 \text{ and } d > 50. \end{cases} \quad (3)$$

These distributions were learnt by applying the two pitch estimators to the nine training recordings mentioned in Section 2.2 and forming histograms of the resulting reliability measures. The distributions for the high-frequency correlogram estimator are shown in Figure 4 and those for the semitone-cancellation-TWM estimator have similar shape. It can be seen that the reliability measures are generally low on unvoiced frames and for incorrect pitch estimates, but higher for correct pitch estimates.

3.3.3. Transition probabilities

Transition probabilities between voiced states are modelled using a combination of Gaussians with variances of 50 and 100 cents representing the variation in pitch during a note

Table 1. Summary evaluation for 19 30-second test recordings

System	Voicing Recall	False Alarm	d-prime Measure	Raw Pitch Accuracy	Raw Chroma Accuracy	Overall Accuracy
HF Corr.	58%	17%	1.20	59%	63%	63%
SC/TWM	68%	29%	1.00	56%	67%	58%
Proposed	71%	24%	1.25	71%	77%	67%

and between successive notes respectively. Other transition probabilities are estimated from the ground truth transcriptions for the training set. The transition probability from state $\omega_{i,t-1}$ to state $\omega_{j,t}$ is therefore defined as

$$P(\omega_{j,t} \mid \omega_{i,t-1}) = \begin{cases} 0.97 & i=0, j=0, \\ 0.03 \times \frac{1}{|\Omega_t|-1}, & i=0, j\neq 0, \\ 0.014 & i\neq 0, j=0, \\ c_{i,t} \times (0.936 \times \mathrm{e}^{\frac{-d^2}{100}} + 0.05 \times \mathrm{e}^{\frac{-d^2}{200}}), & i\neq 0, j\neq 0 \end{cases} \quad (4)$$

where $d = 1200 \times |\log_2(\omega_{j,t}/\omega_{i,t-1})|$ denotes the pitch difference in cents and $c_{i,t}$ is a normalisation factor chosen such that the transition probabilities sum to one.

4. Evaluation

The proposed system was first tested on 19 30-second extracts covering a wide range of genres and instrumentations, and evaluated according to the criteria used in the MIREX 2005 Audio Melody Extraction task. For comparison, the two pitch estimators were tested individually by running the HMM with a single set of pitch estimates. The results for the three systems are shown in Table 1. It can be seen that the proposed system considerably outperforms either single-estimate system, with a better d-prime value for voicing detection and substantially higher pitch accuracy. This demonstrates that there is a benefit to using multiple pitch estimators in parallel, and that the modified HMM system is a suitable fusion method.

The system was also entered for the MIREX 2006 Melody Extraction Task, with results being compiled for vocal melodies, non-vocal melodies, and all melodies. In the case of vocal melodies (see Table 2), the system achieved a similar transcription accuracy as above, ranking it third out of five in both categories. The same test set was used in 2005 and when vocal melody results are compiled for the 2005 systems also, the proposed system ranks 4/15 for raw pitch accuracy and 5/15 for overall accuracy.

The voicing performance was better than during previous testing, achieving a d-prime measure of 1.74, compared with the top-scoring system's d-prime measure of 1.75. The system's specialisation for vocal melodies is demonstrated well by the results for non-vocal melodies, where both voicing and pitch estimation performance fall considerably, and overall accuracy drops to around 30%.

Table 2. MIREX 2006 results - Vocal Melodies

System	Voicing Recall	False Alarm	d-prime Measure	Raw Pitch Accuracy	Raw Chroma Accuracy	Overall Accuracy
Dressler	85.5%	28.7%	1.62	78.5%	81.6%	73.7%
Ryynänen	77.0%	15.6%	1.75	75.7%	76.9%	72.5%
Sutton	71.8%	12.3%	1.74	70.7%	71.6%	67.3%
Poliner	93.7%	44.3%	1.68	69.1%	70.6%	65.0%
Brossier	99.6%	97.9%	0.63	42.7%	53.5%	30.7%

5. Conclusion

It was hoped that by narrowing the melody transcription task to vocal melodies only, a higher accuracy of transcription would be achievable. Though results do not yet show this, the pitch accuracy obtained is promising for a system which has not yet been extensively developed. The pitch estimation results also demonstrate the potential of using multiple pitch estimators in parallel. The benefit of specialising in vocal melodies is shown by the strong voicing performance, where a relatively simple method achieves voicing performance similar to the top-scoring system. More information about the proposed system and a discussion of potential improvements are available in [4].

6. Acknowledgments

We would like to thank the staff of UIUC for all their hard work organising and running the 2006 MIREX competition. We are especially grateful for the extra work involved in producing separate results for vocal melodies.

E. Vincent is funded by EPSRC grant GR/S75802/01.

References

[1] R. Timmers and P. W. M. Desain. Vibrato: The questions and answers from musicians and science. In *Proc. Int. Conf. on Music Perception and Cognition (ICMPC)*, 2000.

[2] A. Shenoy, Y. Wu, and Y. Wang. Singing voice detection for karaoke application. In *Proc. Int. Symp. on Visual Communications and Image Processing (VCIP)*, 2005.

[3] Y. Li and D. L. Wang. Detecting pitch of singing voice in polyphonic audio. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages III–17–20, 2005.

[4] C. Sutton. Transcription of vocal melodies in popular music. Master's thesis, Dept. of Electronic Engineering, Queen Mary, University of London, 2006.

[5] R. C. Maher and J. W. Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America*, 95(4):2254–2263, 1994.

[6] M. P. Ryynänen and A. P. Klapuri. Polyphonic music transcription using note event modeling. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 319 – 322, 2005.

Audio-Based Music Similarity and Retrieval: Combining a Spectral Similarity Model with Information Extracted from Fluctuation Patterns

Elias Pampalk*
National Institute of Advanced Industrial Science and Technology (AIST)
IT, AIST, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan

October 7, 2006

Abstract

This paper describes the implementation submitted by the author to the MIREX'06 (Music Information Retrieval eXchange) evaluation track on audio-based music similarity and retrieval. In addition, this paper summarizes the optimization of this implementation and its evaluation prior to submission. Finally, a detailed analysis and discussion of the MIREX results is presented. Overall, this implementation performed slightly better in terms of quality and computation time than the other implementations. However, the measured differences were not significant.

1. Introduction

The perception of music similarity is subjective, context-dependent, and multi-dimensional (including instrumentation, harmony, melody, rhythm etc.). Nevertheless, the basic approach of this implementation is one-size-fits-all. In particular, given any two songs, without any further context, one number is computed.

There are mainly two reasons for focusing on such obviously over-simplistic approaches. First, there are applications where one-size-fits-all can be applied such as automatic playlist generation. Second, evaluating models which change their similarity rankings depending on the respective context is significantly more complex.

1.1. Evaluation

The optimal approach to compare the performance of computational models of music similarity is to evaluate them within the context of their application. For example, one approach could be to ask users which similarity model generates the best playlists.

However, empirical results suggest that even without a specific application context similarity ratings can be evaluated consistently by human listeners. For example, Logan & Salomon [1] presented results from a listening test where two subjects were asked to judge if a given song is similar to another (yes/no). The subjects disagreed in only 12% of the cases.

A similar consistency of judgments is reported in [2]. In particular, a listening test was conducted where the subjects were given a song (X) and asked to rate its similarity to two other songs (A and B) on a scale from 1 to 9. Thus, each subject was asked for two numbers given three songs: the similarity of AX and the similarity of BX. The consistency of the ratings from different subjects were compared in terms of the difference $d_i = \mathrm{AX}_i - \mathrm{BX}_i$ which was computed for each subject i. The results showed that this difference was surprisingly consistent. In 26% of the cases two subjects i and j had the same values ($d_i = d_j$). In 32% of the cases the difference was only 1 point ($|d_i - d_j| = 1$). In about 19% of the cases the difference was 2 points ($|d_i - d_j| = 2$). Only in a few cases the listeners truly disagreed (in 15% of the cases ($|d_i - d_j| \geq 4$).

A simpler alternative to conducting listening tests is to use a genre classification approach (e.g. [1, 3, 4]). The basic assumption is that given a piece of music, very similar pieces can be found within the same genre. However, special precautions need to be taken such as applying an "artist filter" and using different collections with different taxonomies (for a detailed discussion see [2]).

1.2. Related Work

All details of this implementation are given in [2], where the implementation (Matlab source code), optimization, and evaluation (including a listening test) are described.

The approach of combining a spectral similarity model with information from fluctuation patterns is based on previous work presented in [4]. A similar version which was optimized with respect to computation time was submitted to the genre classification track of MIREX'05 [5] where it (despite only using a nearest neighbor classifier) outperformed a number of powerful classifiers (such as support vector machines).

The spectral similarity part of this implementation is based on the work of Mandel & Ellis [6]. However, alternative approaches developed by Logan & Salomon [1] or Aucouturier & Pachet [3] could be applied as well. The main advantage of the approach used by Mandel & Ellis is that it is computationally very fast.

The fluctuation patterns were presented in [7, 8] and are based on previous work by Früwirth & Rauber [9, 10]. The "gravity" descriptor extracted form the fluctuation patterns was presented in [4]. The "bass" descriptor is based on

*) Part of this work was done while the author was working at the Austrian Research Institute for Artificial Intelligence (OFAI).

work presented in [7] and was slightly modified as described in [2]. The specific implementation of the fluctuation patterns (e.g. using MFCCs) used for this implementation to compute the fluctuation patterns was first described in [5].

2. Implementation of G1C

G1C stands for "Single Gaussian Combined". The implementation submitted to MIREX is described in detail in [2] including the Matlab code. In this section the techniques are only roughly summarized.

First, for each piece of music the MFCCs are computed for a maximum of two minutes from the center of the piece. In particular, a 19-dimensional MFCC vector for every 23ms of the signal is computed. The distribution of these vectors is summarized using a single Gaussian (G1) with full covariance matrix. The distance between two Gaussians is computed using a symmetric version of the Kullback Leibler divergence.

The fluctuation pattern (FP) describes the modulation of the loudness amplitudes per frequency bands. To some extent it can describe periodic beats. A FP is a two-dimensional matrix where each row corresponds to a frequency-band and each column to a modulation frequency (in the range of 0-10Hz). The values of this matrix describe how strong the fluctuation of the loudness amplitude is within a specific frequency band and at a specific modulation frequency.

To compute the fluctuation patterns the Mel spectrogram (with loudness in dB) is used. The Mel spectrogram is obtained in an intermediate step when computing the MFCCs. In the next step the energy in each frequency bands is regrouped into 12 bands. This re-grouping is done such that variations in lower frequency bands are emphasized. The Mel spectrogram is then chopped into 3 second segments. For each segment, the loudness modulation in each frequency band is computed using an FFT. The modulation frequencies are analyzed in the range of 0-10Hz. The modulation amplitudes are weighted to emphasize modulation around 4Hz based on a model of fluctuation strength [11, 12]. Specific modulation patterns are emphasized using smoothing and edge detection filters. All fluctuation patterns computed for each 3 second window are combined by computing the median of all patterns. The patterns of two pieces of music are compared by computing the Euclidean distance (and first converting the matrix into a vector by concatenating the rows).

From the FP of each song two features are extracted. One is the "gravity" (FP.G). It is the center of gravity of the FP along the modulation frequency dimension. It roughly corresponds to the perception if a piece is slow or fast. The other is the "bass" (FP.B). It is computed as the fluctuation strength of the lower frequency bands at higher modulation frequencies. The distance of two songs for each of these descriptors is computed as the absolute difference of values.

Given the four distance values (for G1, FP, FP.B, and FP.G) the overall similarity of two pieces is computed as a weighted linear combination. The normalization and weights used are described in detail in [2]. If the inversion of the covariance necessary for G1 leads to numerical problems for any song, then the combination weight is set to zero when it is compared to any other song. The optimization and evaluation of the weights is briefly described in Section 3.

2.1. Computational Resources

The following computation times are measured running Matlab code (Windows XP) on a Pentium M 2GHz processor. Given an audio file in WAV format extracting the features for one piece of music takes about 2 seconds. Computing the spectral similarity of two songs takes about 0.1 milliseconds. The FP part of the similarity computation is much faster as it only requires computing the Euclidean distance of two vectors with 362 elements each. For each song a total of $362 + 2 \times 19 \times 19 + 19$ (=1103) values need to be stored. (In addition to the G1 covariance matrix also the inverted covariance matrix is stored so it does not need to be recomputed for each similarity computation.) For a comparison of computation times see Section 4.5 and Figure 6.

3. Optimization and Evaluation

To optimize the combination weights and to evaluate G1C a genre-based evaluation procedure was used. In particular, given a music collection containing pieces for which the genre and artist is known the following steps were computed: First, for each piece (query) all pieces from the same artist in the collection are removed. Second, the piece most similar to this query is found (according to the similarity model). If this piece is from the same genre, the score for the query piece is 1, otherwise 0. Finally, the average score for all queries is computed. This is identical to nearest neighbor genre classification, measuring the performance using leave-one-out cross-validation, and using an artist filter to ensure that training and test set contain non-overlapping sets of artists.

The combination weights of G1C were optimized using two music collections (DB-MS and DB-L described in [2]). The parameter space was evaluated using a grid search in combination with the genre-based evaluation approach. The combination which performed best in average on the two music collections, was evaluated using 4 collections (DB-S, DB-ML, DB-30, DB-XL). In addition a listening test was conducted to analyze how the genre-based results are related to judgments made by human listeners. The results of this test confirmed that improvements measured with the genre-based procedure are also measurable using a listening test. (For details see [2].)

4. MIREX'06 Results

The raw data and details of the evaluation procedure can be found on the MIREX pages.[1] This section briefly describes the listening test setup and analyzes the results.

[1] http://www.music-ir.org/mirex2006/index.php

4.1. Listening Test Setup

To evaluate the performance of the algorithms a user study was conducted by IMIRSEL.[2] First, each of the 6 algorithms submitted to the contest retrieved the 5 most similar songs form the database given the query. Songs from the same artist as the query were filtered. The main reason for filtering songs from the same artist is that the intention was not to evaluate how well the submissions perform in artist identification.[3] Overall 60 queries were used for the listening test. Each candidate and query pair was rated by three subjects.

Only researchers working on related topics participated in the listening test. This restriction was necessary due to legal issues.[4] The participants were given a query song and a list of 30 candidate songs and asked to rate the similarity of each candidate song on a scale from 0-10 using a slider (with a resolution of 0.1) where 0 corresponds to not similar and 10 corresponds to very similar (fine scale). Instead of using the slider the participants could directly enter a number into a text box. Overall, 32% of the fine score ratings the participants entered are whole numbers (0,1,2,...) and 16% are half numbers (0.5,1.5,...). In addition, the participants were asked to rate each of the 30 songs on a broad scale with the options: not similar (NS), somewhat similar (SS), and very similar (VS).

4.2. Official Ranking

The official ranking of the algorithms was computed using the data from the slider (fine scale). For each query and algorithm a score was computed. This score is the mean of the 15 (=3×5) similarity ratings (each in the range 0-10) associated with each query/algorithm pair. In the next step the Friedman test is computed, the results of the Friedman test are then post-processed to find significant differences between algorithms. The corresponding Matlab code is:

```
[p,table,stats] = friedman(M);
multcompare(stats, ...
              'ctype', 'tukey-kramer', ...
              'estimate', 'friedman', ...
              'alpha', 0.05);
```

where `M` is a matrix with 60 rows (corresponding to the queries) and 6 columns (corresponding to the algorithms). The Friedman test was chosen because it is a non-parametric test which does not assume a normal distribution of the data.

For each query the Friedman test ranks the algorithms with respect to their scores. If an algorithm is consistently ranked higher than another one, then it is significant better. On the other hand, if the algorithm A scores better for half of the queries and algorithm B for the other half, then the difference is not significant. (Note that this is the case even if algorithm B has much higher scores in average.) Estimating the significance of differences is very important as some difference are very small and wrong conclusions could easily be drawn.

[2] http://www.music-ir.org/evaluation/

[3] For a discussion on the relationship between artist identification and music similarity see [5].

[4] For details see the email by Stephen Downie on the MIREX list on September 2, 2006.

	Mean	Median
EP	4.30	4.05
TP	4.23	4.05
VS	4.04	3.50
LR	3.93	3.70
KWT*	3.72	3.40
KWL*	3.39	2.95

Table 1. Statistics of the average fine score per algorithm. The minimum value is 0 (not similar) and the maximum is 10 (very similar).

The results are shown in Figure 1 (left side). Note that EP marks the G1C algorithm and "*" marks all submissions that contain bugs according to their authors. Most of the measured differences between algorithms are not significant according to the Friedman test (at p-level 0.05). Only KWL* performs significantly worse than some others.

4.3. Other Fine Scale Results

The right side of Figure 1 shows the results when the median is computed instead of the mean of the 15 ratings to obtain the score for each query/algorithm pair. For some query/algorithm pairs the differences between the two approaches can be very large. For example, the in contrast to the mean the median would not distinguish between 1,2,9 and 1,2,3. The advantage of using the median is that it is less sensitive to outliers. However, as can be seen the significance of the measured differences remains the same.

Figure 2 shows the results when using a balanced two-way ANOVA instead of the non-parametric Friedman test. Although the necessary assumptions are not perfectly met the results are very similar to those of the Friedman test.

Figure 3 shows the distribution of scores per query for each algorithm. As can be seen, when each of the 60 scores is computed as the mean of the respective 15 observations the normal distribution is a better approximation than in the case where the median is used.

Table 1 shows statistics of all ratings. The average ratings are lower than those reported in [2] where the mean ratings was 6.37 (one a similar scale from 1-9). The most likely reason for the difference is the that a different music collection was used. As shown in [2] the performances vary largely depending on the collection. In any case, the average values indicate that the overall quality of the similarity measures might not be satisfactory for users. To evaluate this would require tests within the application context.

4.4. Broad Scale Results

Figure 4 (left side) shows how the broad scale ratings are distributed per algorithm. Overall, G1C got the largest num-

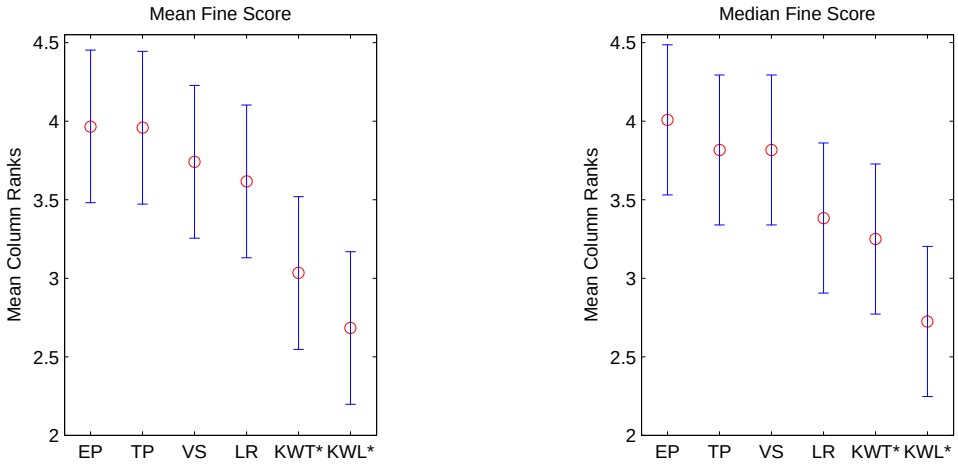


Figure 1. Evaluation results using the Friedman test. The left side shows the final ranking of the contest. The red circles mark the mean ranks computed using the Friedman test. The blue lines mark the significance bounds using a level of p=0.05. The right side differs from the left side with respect to how the score was computed per query/algorithm pair. In particular, the median was used instead of the mean.

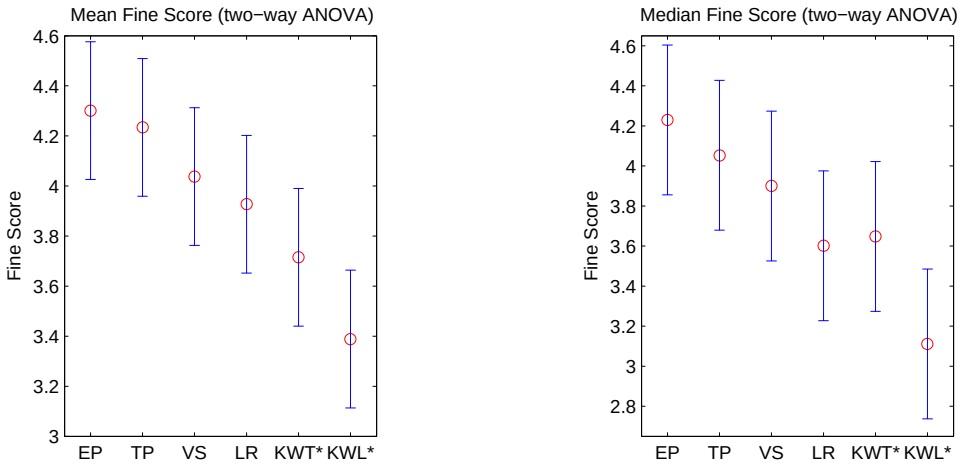


Figure 2. Results using a balanced two-way ANOVA instead of the Friedman test (see Figure 1). The left side uses the mean and the right side uses the median to compute the score for each query/algorithm pair given the 15 observations.

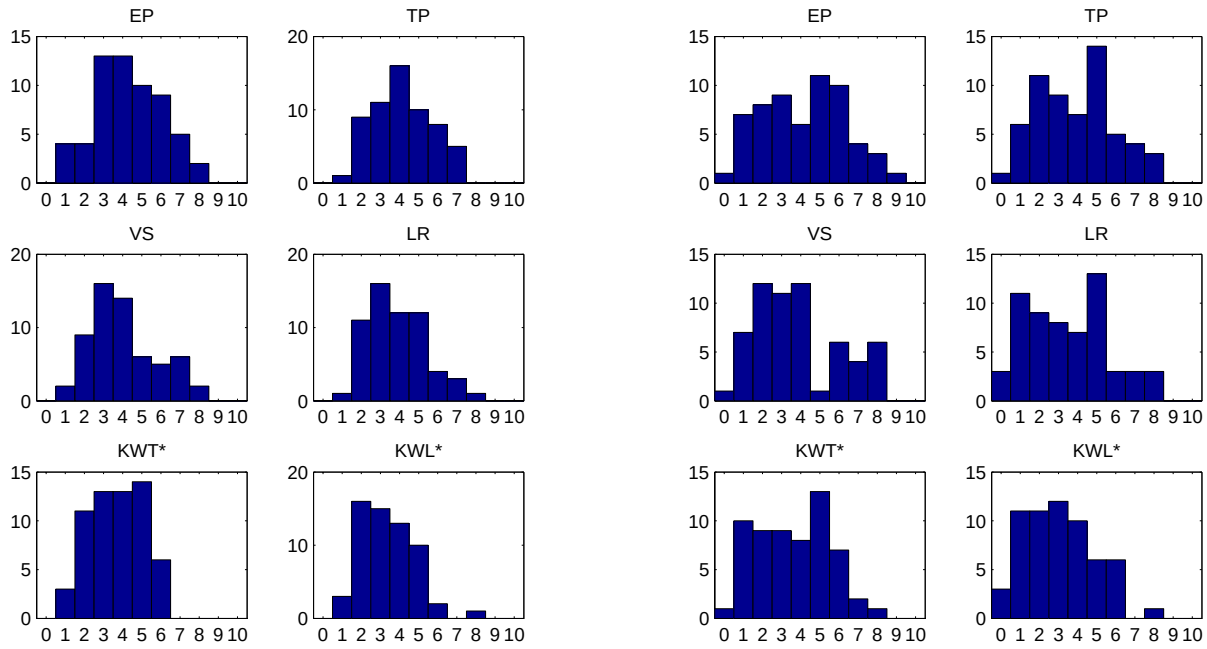


Figure 3. Histogram of the 60 scores per algorithm. On the left side the scores are computed as the mean of all 15 observations (subject ratings) and on the right side as the median.

ber of very similar ratings (VS) and about the same number of not similar ratings (NS) as TP. The right side of Figure 4 shows a similar visualization using the fine scale data. As can be seen, both scales have produced very similar results except that the fine scale resulted in more fine grained distinctions.

The MIREX webpage lists a number of statistics computed based on the broad scale data. There are a number of different ways to interpret the broad scale data. Each interpretation assigns a different number of points to each similarity category. The optimal interpretation depends very much on the final application. In the same way the broad categories are assigned to different points the fine scale could be mapped to a non linear scale to emphasize certain areas of the scale differently.

Given a specific interpretation the average of points assigned to an algorithm per query is used for further analysis. The statistics are listed in Table 2 and Figure 5 shows the corresponding significance boundaries using the Friedman test (and the mean to compute the scores).

4.5. Other Statistics

In addition to the results of listening test a number of other statistics were computed. These are based on a collection of 5000 pieces which included 330 cover songs. For some of the statistics reported here, the cover songs were filtered from the collection. In general the statistics are based on computing the compute distance matrix (5000×5000 values).

4.5.1. Computation Times

Figure 6 shows the computation times for some of the submissions. When interpreting the results it is necessary to consider that some of the algorithms have not been optimized with respect to computation times. However, the measured computation times are rough indicators of the complexity of each algorithm.

Overall G1C is fastest to extract the features from the 5000 songs and compute the complete distance matrix. G1C requires only about half the time to extract the features than the next fastest submission. Lidy & Rauber (LR) submitted by far the fastest algorithm with respect to the distance computation time. Depending on the task this can be a significant advantage. However, to improve the distance computation time of G1C and TP, for example, M-trees [13] and other indexing strategies could be used. In particular, for most applications it is not necessary to compute a complete distance matrix.

4.5.2. Characteristics of the Similarity Spaces

Most similarity space indexing algorithms make assumptions about the similarity space. To study the properties of the similarity space three aspects were analyzed for those algorithms which allowed computation of the complete dis-

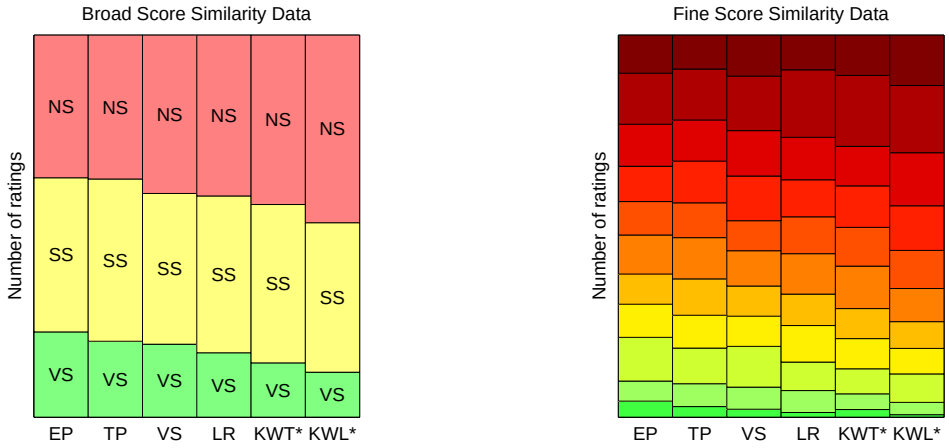


Figure 4. The left side shows the number of times per algorithm the songs were rated with each of the broad similarity scale categories. The categories on the broad scale are: not similar (NS), somewhat similar (SS), and very similar (VS). The right side uses the data from the fine scale. For each algorithm the lowest block (green) corresponds to a rating of 10, the highest block (red) to a rating of 0. All ratings are rounded to the nearest whole number.

	Points per Category			Scores					
Abbreviation	NS	SS	VS	EP	TP	VS	LR	KWT*	KWL*
Greater0	0	1	1	62.7	62.3	58.6	57.9	55.7	50.9
Psum	0	1	2	42.5	41.1	38.8	37.4	34.9	31.3
Wcsum	0	1	3	35.8	34.0	32.3	30.6	28.0	24.8
Sdsum	0	1	4	32.4	30.5	29.0	27.1	24.6	21.6
Greater1	0	0	1	22.3	19.9	19.1	16.9	14.2	11.8

Table 2. Evaluation scores using the broad scale data for different interpretations of the data. The values are normalized so that the scale ranges from 0 (complete failure) to 100 (perfect). The general tendency is the same regardless of the points per category.

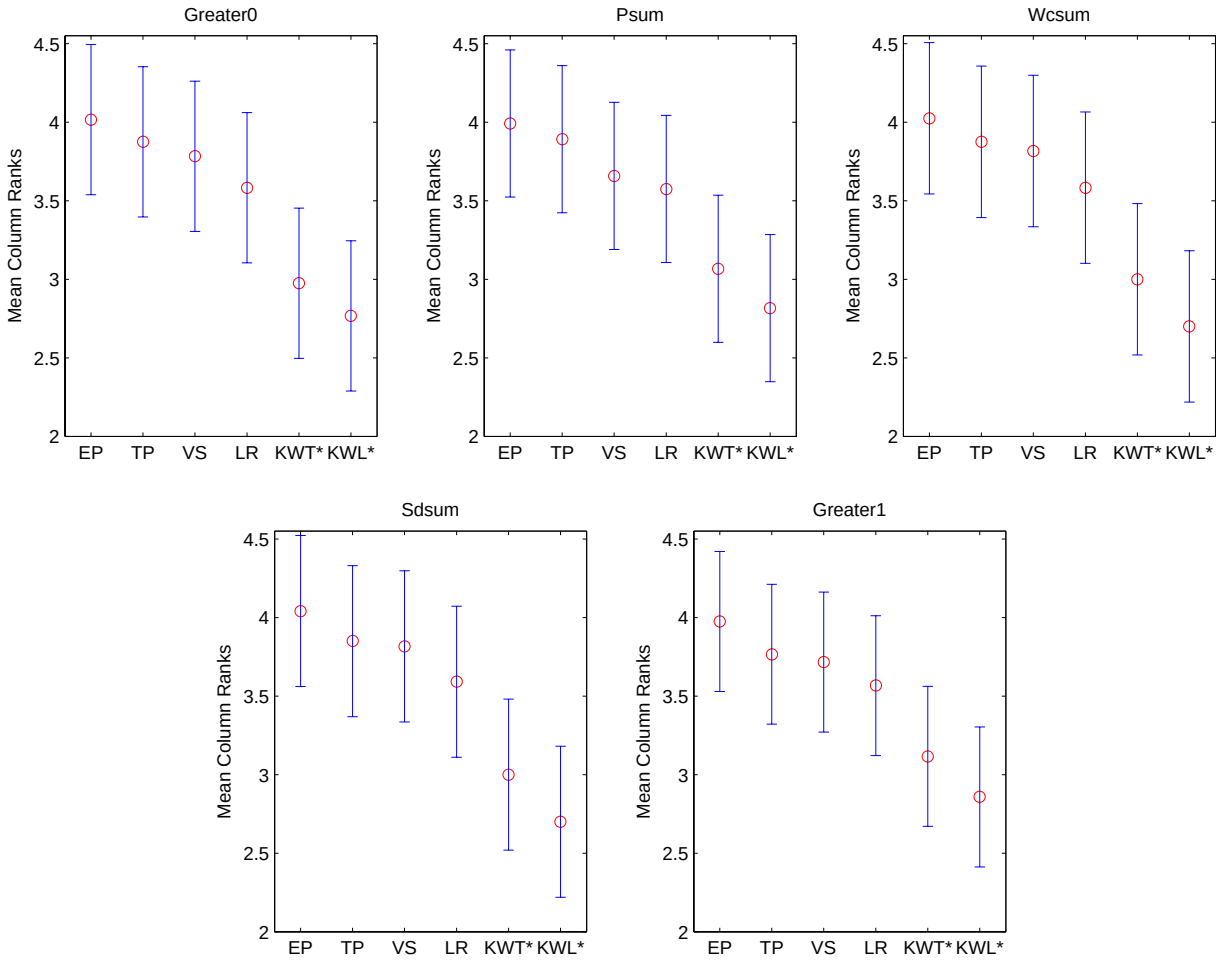


Figure 5. The same evaluation results as shown in Figure 1 except that the broad score (with different interpretations) is used instead of the fine score. In terms of the insignificance of the differences the results are similar to those using the fine score data.

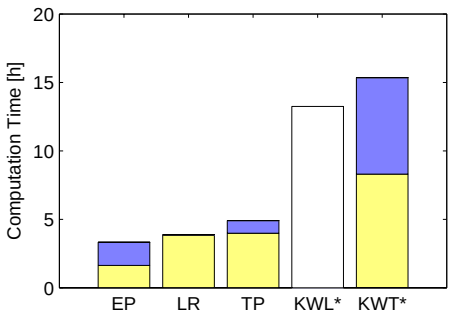


Figure 6. Computation times for some of the submissions. The lower part of each bar (yellow) is the feature extraction time (for 5000 songs). The upper part (blue) is the distance computation time for the complete distance matrix (which requires computing the distance of 12.5 million song pairs). For KWL* the times for the individual parts were not recorded. The VS submission was not able to compute the full distance matrix within a reasonable amount of time. The times were measured on a machine with: Dual AMD Opteron 64, 1.6 GHz, 4 GB RAM, running Linux (CentOS).

tance matrix within reasonable time. [5]

First, the problem of "always similar songs" (also known as "hubs") was analyzed. Hubs in music collections were first reported by Auctouturier & Pachet [14]. A detailed discussion can be found in [15]. A hub is a song that (according to the computational model of similarity) is very similar to a large number of other songs. However, this computational similarity does not correspond to perceptual similarity.

Some similarity measures (such as those based on spectral similarity) are affected. However, the number of these hubs is usually very low, and in a collection of several thousand pieces only few can be observed. Hubs are easy to detect when analyzing a distance matrix, however, they are difficult to detect when only computing a number given two songs. Tim Pohle's submission uses an interesting approach to prevent extreme hubs.

Table 3 shows the maximum number of times a song appeared in the top 5 ranks. The main observation is that in the subset of cover songs one "always similar" song appeared for G1C but not for the other submissions. The existence of "always similar" outliers for G1C was also documented in [2].

Very closely related to the analysis of "always similar" songs is the question if there are any songs which never occur in the top 5 rankings. In the collection used for this contest, no cases of "always dissimilar" songs were found. However, as shown in [2] songs which are dissimilar to almost all songs in the collection (including songs which sound similar) can occur using G1C.

Of interest is also if the triangular inequality holds in the

[5] The analysis was implemented by Kris West.

Collection Size	EP	TP	LR	KWT*	KWL*
4670	48	62	42	61	24
5000	1753	62	42	61	24

Table 3. Maximum number of times a song was ranked in the top 5 most similar songs of all songs. The lowest possible value is 5. The highest possible value equals the size of the collection minus 1.

similarity space the submission defines. The triangular inequality states that the sum of the distances AX and BX is larger or equal to the distance AB (in our case A, B, and X are songs). This inequality is an important characteristic of metric spaces and a number of algorithms (especially indexing algorithms) rely on it. To measure the degree to which the submissions fulfill the inequality a random sample of triangles is drawn from the distance matrix. Each of these triangles is tested whether the inequality is fulfilled. For the submissions G1C, KWT*, and LR the inequality held for all samples drawn. For TP the inequality held in about 32% of the cases, and for KWT* in about 54%.

In [2] a different music collection is used and G1C fulfills the inequality only in 36% of the cases. One possible explaination could be that the submitted version of G1C is not exactly the same as the one used on [2]. The only difference is that the contribution of the spectral similarity is set to zero if nummerical problems occur when computing the inverse covariance matrix. However, nummerical problems only occur very seldomly.

4.5.3. Genre-based Evaluation

Previous work has shown that evaluations based on genre data correspond to evaluations based on similarity ratings gathered in listening tests [2]. However, the genre data available for the music collection used in this contest was not reliable. For example, Britney Spears and Depeche Mode are assigned to the same genre (rock). Furthermore, the distribution of songs per genre is very unbalanced. This is reflected in the results computed using an artist filter which do not reflect the results from the listening test.

The results without using an artist filter are more useful, because they basically measure the artist identification performance. However, identifying artists and finding similar pieces are not the same tasks. For example, an algorithm that can identify recording environments or other production effects might perform very well for artist identification but not for music similarity. for a more detailed discussion on this see [5, 2].

5. Discussion

There are two reasons why the measured differences (using the listening test data) were not significant. First, the differences between the algorithms are very small. Second, the evaluation procedure was not adequate to measure these

small differences.

5.1. Glass Ceiling

Aucouturier and Pachet pointed out a glass ceiling for spectral similarity [14]. The results of this listening test have confirmed this ceiling. In particular, G1C is only marginally better than the author's submission to the MIREX 2005 and ISMIR 2004 genre classification contests [2]. The main improvements from 2004 and 2006 have been in terms of computation time which has been reduced by several magnitudes.

Thus, it is not surprising that some of the submissions have reached the same glass ceiling. In particular, the overall difference between G1C and TP is very difficult to measure.

5.2. Evaluation Procedure

A very straightforward approach to increase the power of the test (and allow us to measure significant differences) would have been to use a larger number of queries. To reduce the overall load on the subjects, fewer subjects per rating and fewer candidates per algorithm and query could be used. Thus, more than 10 times as many queries could have been used in the evaluation without increasing the effort per subject.

Using fewer candidates per algorithm and query would also have the advantage that the size of the local context would be reduced. The local context is the context in which the subjects rate the songs. This context consists of the query, and the 30 candidates to be rated. A smaller local context is likely to lead to more consistent ratings.[6] For example, in [2] the local context consisted of only 3 songs and the consistency was higher. However, when using a larger context (e.g. when evaluating several algorithms) the subjects should be given tools to help them apply ratings consistently. One such option would be to implement a sort function for the ratings.

Figures 7 and 8 visualize the consistency of the ratings. Figure 9 shows the consistency of the ratings from the listening test described in [2]. The consistency is computed as follows. For each query and candidate pair (60×30) there are 3 ratings (by three different subjects). We compute the absolute differences between these ratings. In total this results in $60\times30\times3$ absolute differences. In an ideal case all of these would be zeros. In the worst case (worse than random) a large proportion of these values would be 10 (which is the maximum possible disagreement on the fine scale from 0-10).

The consistency can be quantified and compared using the ratios of pairs with a very high consistency. In particular, the first quarter of bins (using the histograms in Figures 7-9) is considered to be highly consistent. In case of the broad

[6] It is important to note that splitting the candidates per query into two or more sets is not a solution. If the candidates are rated in a different local context than the ratings are not comparable. As a result a test such as the Friedman test could not be used to evaluate the results.

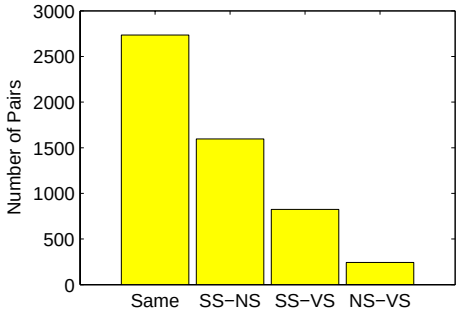


Figure 7. Histogram of absolute rating differences (broad score).

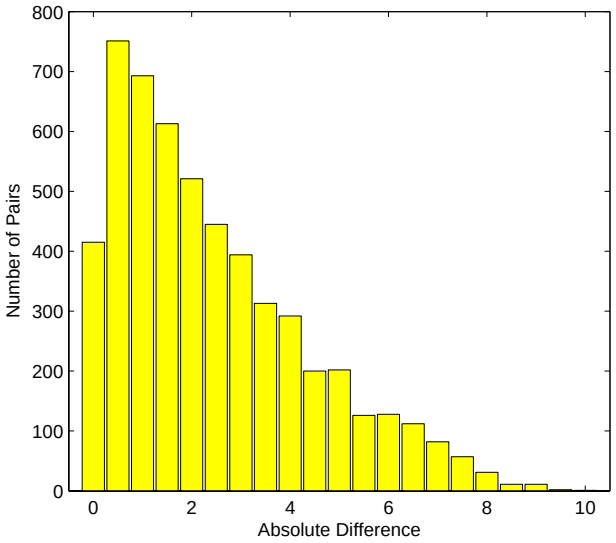


Figure 8. Histogram of absolute rating differences (fine score).

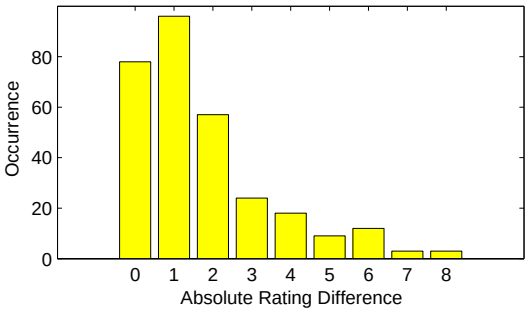


Figure 9. Histogram of absolute rating differences for the listening test reported in [2] where a scale from 1-9 was used.

Evaluation Procedure	Consistent Pairs
Broad Scale	51%
Fine Scale	55%
Evaluation reported in [2]	58%

Table 4. Percentage of very consistent pairs of ratings for different evaluation procedures.

score data this means the ratio of exact same ratings compared to the number of all pairs. For the fine score data the first 5 bins are used. For the listening test conducted in [2] the first 2 bins are used. The results are given in Table 4. The results question the use of the broad scale and support the argumentation for using a smaller local context.

6. Conclusions

Future listening tests should use a larger number of queries. A fine scale is preferable to a broad scale because the ratings are more consistent. Furthermore, the size of the local context should be reduced to increase the consistency of the ratings.

However, even with improved evaluation procedures the differences between algorithms which have reached the "glass ceiling" are very marginal and might not be relevant for most applications. Evaluations within the context of applications are clearly desirable. A question of particular interest is if the similarity measures in their current form (despite their obvious limitations) can be successfully applied in any applications.

6.1. Conclusions of the Evaluation Results

G1C was fastest overall and achieved the highest score. However, the measured differences were not significant. The submission by Lidy & Rauber is the only algorithm which uses a vector space. This results in extremely fast distance computations and also allows the application of their submission to a larger number of problems. Their submission is probably the most suitable for extremely large collections (containing millions of pieces). Tim Pohle presented a very interesting approach which can also be applied to G1C to reduce the number of "always similar" outlier songs. Furthermore, the distance computation time for his submission is only about half of that for G1C which can be a major advantage for some applications.

References

[1] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'01)*, 2001.

[2] E. Pampalk, "Computational models of music similarity and their application in music information retrieval," Docteral dissertation, Vienna University of Technology, Austria, March 2006.

[3] J.-J. Aucouturier and F. Pachet, "Music Similarity Measures: What's the Use?" in *Proc. of ISMIR*, 2002.

[4] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *Proceedings of the ISMIR International Conference on Music Information Retrieval*, 2005.

[5] E. Pampalk, "Speeding Up Music Similarity," in *Proceedings of the MIREX Annual Music Information Retrieval eXchange*, 2005.

[6] M. I. Mandel and D. P. W. Ellis, "Song-Level Features and Support Vector Machines for Music Classification," in *Proc. of ISMIR*, 2005.

[7] E. Pampalk, "Islands of Music: Analysis, Organization, and Visualization of Music Archives," MSc thesis, Vienna University of Technology, Department of Software Technology and Interactive Systems, 2001, http://www.ofai.at/˜elias/music/thesis.html.

[8] E. Pampalk, A. Rauber, and D. Merkl, "Content-Based Organization and Visualization of Music Archives," in *Proceedings of the ACM Multimedia*, 2002.

[9] M. Frühwirth, "Automatische Analyse und Organisation von Musikarchiven (Automatic Analysis and Organization of Music Archives)," MSc thesis, Vienna University of Technology, Austria, 2001.

[10] M. Frühwirth and A. Rauber, "Self-Organizing Maps for Content-Based Music Clustering," in *Proceedings of the Twelfth Italian Workshop on Neural Nets (WIRN01)*, 2001.

[11] E. Terhardt, "Über akustische Rauhigkeit und Schwankungsstärke (On the acoustic roughness and fluctuation strength)," *Acustica*, vol. 20, pp. 215–224, 1968.

[12] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, 2nd ed., 1999.

[13] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," in *Proceedings of the International Conference on Very Large Data Bases*, M. Jarke, M. Carey, K. R. Dittrich, F. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, Eds., 1997.

[14] 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, http://journal.speech.cs.cmu.edu/articles/2004/3.

[15] J.-J. Aucouturier, "Ten experiments on the modelling of polyphonic timbre," Docteral dissertation, University of Paris 6, Paris, France, May 2006. [Online]. Available: http://www.jj-aucouturier.info/papers/PHD-2006.pdf

Simple Spectrum-Based Onset Detection

Simon Dixon
Austrian Research Institute for Artificial Intelligence
Freyung 6/6, Vienna 1010, Austria
simon.dixon@ofai.at

Abstract

In a recent empirical study, various methods for detecting the onset times of musical notes in audio signals were evaluated [1]. The study focussed on published methods based on spectral features such as the magnitude, phase and complex domain representations, and compared existing methods (spectral flux, phase deviation and complex difference) with proposed improvements to these methods (weighted phase deviation, normalised weighted phase deviation and rectified complex difference). Two test sets were used: a set of short excerpts from a range of instruments (1060 onsets), plus a much larger data set of piano music (106054 onsets). Results showed a similarly high level of performance with a magnitude-based (spectral flux), a phase-based (weighted phase deviation) or a complex domain (complex difference) onset detection function. For MIREX 2006, the following five onset detection functions were submitted: spectral flux, complex domain, rectified complex domain, weighted phase deviation and normalised weighted phase deviation.

Keywords: MIREX, spectral flux, phase deviation, complex domain

1. Introduction

Recent reviews and evaluations of onset detection methods can be found in [2, 3, 4, 1]. The onset detection functions described in this document are more fully described and compared in [1]. Although it is clear that different methods are suitable for different data sets, we focus on simple, general-purpose methods of finding onsets. All methods presented here share the same peak picking algorithm, which limits the closeness of successive onsets. For polyphonic music, this might penalise the algorithms, depending on how the evaluation is performed.

2. Onset Detection Functions

An onset detection function is a function whose peaks are intended to coincide with the times of note onsets. Onset detection functions usually have a low sampling rate (e.g. 100Hz) compared to audio signals; thus they achieve a high level of data reduction whilst preserving the necessary information about onsets. Most onset detection functions are based on the idea of detecting changes in one or more properties of the audio signal.

If an audio signal is observed in the time-frequency plane, an increase in energy (or amplitude) within some frequency band(s) is a simple indicator of an onset. Alternatively, if we consider the phase of the signal in various frequency bands, it is unlikely that the frequency components of the new sound are in phase with previous sounds, so irregularities in the phase of various frequency components can also indicate the presence of an onset. Further, the phase and energy (or magnitude) can be combined in various ways to produce more complex onset detection functions. These ideas form the basis of the onset detection functions described in this paper.

All of the methods presented here make use of a time-frequency representation of the signal based on a short time Fourier transform using a Hamming window $w(m)$, and calculated at a frame rate of 100 Hz. If $X(n,k)$ represents the kth frequency bin of the nth frame, then:

$$X(n,k) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(hn+m)\, w(m)\, e^{-\frac{2j\pi mk}{N}}$$

where the window size $N = 2048$ (46 ms at a sampling rate of $r = 44100$ Hz) and hop size $h = 441$ (10 ms, or 78.5% overlap).

2.1. Spectral Flux

Spectral flux measures the change in magnitude in each frequency bin, and if this is restricted to the positive changes and summed across all frequency bins, it gives the onset function SF [5]:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|)$$

where $H(x) = \frac{x+|x|}{2}$ is the half-wave rectifier function. Empirical tests favoured the use of the L_1-norm here over the L_2-norm used in [6, 2], and the linear magnitude over the logarithmic (relative or normalised) function proposed by Klapuri [7].

2.2. Phase Deviation

The rate of change of phase in an STFT frequency bin is an estimate of the instantaneous frequency of that component. This can be calculated via the first difference of the phase of $X(n,k)$. Let $\psi(n,k)$ be the phase of $X(n,k)$, that is:

$$X(n,k) = |X(n,k)|\, e^{j\psi(n,k)}$$

where $-\pi < \psi(n,k) \leq \pi$. Then the instantaneous frequency is given by the first difference $\psi'(n,k)$:

$$\psi'(n,k) = \psi(n,k) - \psi(n-1,k)$$

mapped onto the range $(-\pi, \pi]$. The change in instantaneous frequency, which is an indicator of a possible onset, is given by the second difference of the phase:

$$\psi''(n,k) = \psi'(n,k) - \psi'(n-1,k)$$

which is also mapped onto the range $(-\pi, \pi]$. Large discontinuities in the unwrapped phase or its derivatives can wrap around to 0, but the onset detection function based on phase deviation, PD, takes the mean of the absolute changes in instantaneous frequency across all bins [8, 2], which reduces the chance of a missed detection:

$$PD(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |\psi''(n,k)|$$

2.3. Weighted Phase Deviation

Phase deviation performs poorly because of "noise introduced by components with no significant energy" [2]. That is, the function considers all frequency bins k equally, although the energy of the signal is concentrated around the bins containing the partials of the currently sounding tones. The *weighted phase deviation* (WPD) function takes this into account by weighting the phase deviation values by the magnitude of the corresponding frequency bin:

$$WPD(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k)\ \psi''(n,k)|$$

The *normalised weighted phase deviation* ($NWPD$) function is similar, except that the sum of the weights is factored out, to give a weighted average phase deviation:

$$NWPD(n) = \frac{\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k)\ \psi''(n,k)|}{\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k)|}$$

2.4. Complex Domain

Another way of jointly considering amplitude and phase is to search for departures from "steady-state" behaviour in the complex domain, by calculating the expected amplitude and phase of the current bin $X(n,k)$, based on the previous two bins $X(n-1,k)$ and $X(n-2,k)$. The target value $X_T(n,k)$ is estimated by assuming constant amplitude and rate of phase change:

$$X_T(n,k) = |X(n-1,k)|\, e^{\psi(n-1,k)+\psi'(n-1,k)}$$

and therefore a complex domain onset detection function CD can be defined as the sum of absolute deviations from the target values:

$$CD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X(n,k) - X_T(n,k)|$$

This formulation is simpler but equivalent to the complex domain detection function in [2, 9].

2.5. Rectified Complex Domain

One problem with the CD method is that it does not distinguish between increases and decreases in amplitude of the signal, so that onsets are not distinguished from offsets. The rectified complex domain (RCD) onset detection function uses half-wave rectification to preserve the complex differences only in spectral bins where energy is increasing:

$$RCD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} RCD(n,k)$$

where

$$RCD(n,k) = \begin{cases} |X(n,k) - X_T(n,k)|, & \text{if} |X(n,k)| \geq \\ & |X(n-1,k)| \\ 0, & \text{otherwise} \end{cases}$$

3. Onset Selection

The onsets are selected from the detection function by a peak-picking algorithm which finds local maxima in the detection function, subject to various constraints. The thresholds and constraints used in peak-picking have a large impact on the results, specifically on the ratio of false positives to false negatives. For example, a higher threshold generally reduces the number of false positives and increases the number of false negatives. The best values for thresholds are dependent on the application and the relative undesirability of false positives and false negatives.

Peak picking is performed as follows: each onset detection function $f(n)$ is normalised to have a mean of 0 and standard deviation of 1. Then a peak at time $t = \frac{nh}{r}$ is selected as an onset if it fulfils the following three conditions:

$$f(n) \geq f(k) \text{ for all } k \text{ such that } n - w \leq k \leq n + w$$
$$f(n) \geq \frac{\sum_{k=n-mw}^{n+w} f(k)}{mw + w + 1} + \delta$$
$$f(n) \geq g_\alpha(n-1)$$

	PN	PP	NP	CM	Sonatas	Error
SF	0.952	0.984	0.967	0.882	0.964±0.017	8.8
WPD	0.947	0.912	0.966	0.836	0.912±0.028	9.6
NWPD	0.938	0.971	0.958	0.879	0.944±0.021	10.3
CD	0.946	0.978	0.936	0.876	0.966±0.015	12.8
RCD	0.963	0.981	0.963	0.877	0.955±0.018	9.3

Table 1. *Results of onset detection tests for 5 onset detection functions (SF, WPD, NWPD, CD and RCD). The first four columns show the maximum of the F-measure for the four subsets of data set 1: pitched non-percussive (PN), pitched percussive (PP), non-pitched percussive (NP) and complex mixture (CM). The last 2 columns show results for data set 2 (Sonatas): the F-measure with standard deviation of F-measures across sonatas and average absolute error in ms.*

where $w = 3$ is the size of the window used to find a local maximum, $m = 3$ is a multiplier so that the mean is calculated over a larger range before the peak, δ is the threshold above the local mean which an onset must reach, and $g_\alpha(n)$ is a threshold function with parameter α given by:

$$g_\alpha(n) = \max(f(n),\ \alpha g_\alpha(n-1) + (1-\alpha)f(n))$$

Experiments were performed with various values of the two parameters δ and α, and it was found that best results were obtained using both parameters, but the improvement in results due to the use of the function $g_\alpha(n)$ was marginal, assuming a suitable value for δ is chosen.

4. Results

Before submission, two data collections were used for testing the onset detection functions. The data from Bello et al. [2], consists of 4 sets of short excerpts from a range of instruments, classed into the following groups: NP — non-pitched percussion, such as drums (119 onsets); PP — pitched percussion, such as piano and guitar (577 onsets); PN — pitched non-percussion, in this case solo violin (93 onsets); and CM — complex mixtures from popular and jazz music (271 onsets). The second data collection consists of about 4 hours of Mozart Piano Sonatas (106054 onsets) — two orders of magnitude more than that used in other evaluations — and includes complex passages such as trills, fast scale passages with pedal and arpeggiated chords. The level of complexity is such that a human annotator would not be able to mark all the onsets precisely.

Table 1 shows the results across these two data sets. In each case, the results are shown for the point on the ROC curve which gives the maximum value of the F-measure. That is, the ground-truth data was used to select optimal values of δ and α. Further issues involving evaluation are discussed in [1].

In these results, the spectral flux, weighted phase deviation and complex domain methods all achieved a similar level of performance on this data, so that the choice of a suitable algorithm could be based on other factors such as simplicity of programming, speed of execution and accuracy of correct onsets (right column), which all speak for the spectral flux onset detection function (SF).

Entry	Precision	Recall	F-measure
roebel-3	0.836	0.779	0.788
roebel-2	0.831	0.769	0.780
roebel-1	0.861	0.746	0.777
du	0.797	0.799	0.762
brossier-hfc	0.752	0.774	0.734
dixon-sf	0.736	0.790	0.726
brossier-dual	0.769	0.735	0.724
brossier-complex	0.780	0.725	0.721
dixon-rcd	0.735	0.765	0.716
dixon-cd	0.709	0.776	0.710
brossier-specdiff	0.764	0.701	0.707
dixon-wpd	0.663	0.786	0.685
dixon-nwpd	0.524	0.908	0.620

Table 2. *Average precision, recall and F-measure for the best parameter setting for each of the MIREX 2006 entries, sorted by F-measure.*

The results from the MIREX 2006 competition are shown in Table 2. The performance of the onset detection functions is much lower than in Table 1. There are a number of reasons why this is the case: first, the parameter settings used in Table 1 were refined with knowledge of the onset times, allowing some amount of overfitting to the data. From the results in Table 2, it is clear that the range of parameter settings for the submitted onset detection functions was too narrow, so that the optimal point on the ROC curve was not reached in each case. This is particularly clear for the case of the NWPD function, where a bug in the submitted code led to wrong parameter values being used. It is also worth noting that the data used in Table 1 are relatively easy for onset detection; the first part consists of simple music, and the second part consists of complex music played on a simple-to-detect instrument, the piano. Further analysis of the results will yield insights into the specific strengths and weaknesses of the individual algorithms.

5. Acknowledgements

This work was supported by the Vienna Science and Technology Fund, project CI010 *Interfaces to Music*, and the EU project S2S^2. OFAI acknowledges the support of the ministries BMBWK and BMVIT. Thanks to Juan Bello for providing test data, and to the MIREX team for conducting the MIREX evaluation.

References

[1] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, 2006, pp. 133–137.

[2] J.P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in musical signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[3] N. Collins, "A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions," in *118th Convention of the Audio Engineering Society*, Barcelona, Spain, 2005.

[4] J.S. Downie, "2005 MIREX contest results - audio onset detection," www.music-ir.org/evaluation/mirex-results/audio-onset, 2005.

[5] P. Masri, *Computer Modeling of Sound for Transformation and Synthesis of Musical Signal*, Ph.D. thesis, University of Bristol, Bristol, UK, 1996.

[6] C. Duxbury, M. Sandler, and M. Davies, "A hybrid approach to musical note onset detection," in *Proceedings of the 5th International Conference on Digital Audio Effects*, 2002, pp. 33–38.

[7] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Phoenix, Arizona, 1999.

[8] C. Duxbury, J.P. Bello, M. Davies, and M. Sandler, "A combined phase and amplitude based approach to onset detection for audio segmentation," in *Proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS-03)*, 2003, pp. 275–280.

[9] 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 Processing Letters*, vol. 11, no. 6, pp. 553–556, 2004.

MIREX 2006: Spectral-flux based Musical Onset Detection

Yunfeng Du
Institute of Acoustics,
Chinese Academy of Sciences
ydu@hccl.ioa.ac.cn

Ming Li
Institute of Acoustics,
Chinese Academy of Sciences
mli@hccl.ioa.ac.cn

Jian Liu
Institute of Acoustics,
Chinese Academy of Sciences
jliu@hccl.ioa.ac.cn

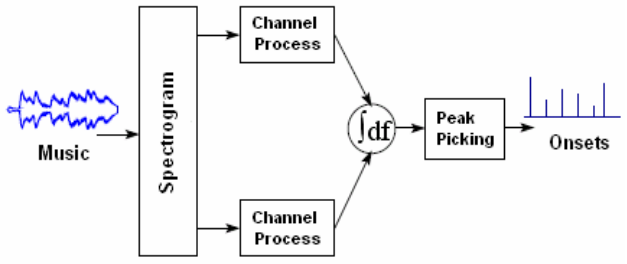
Abstract

This abstract describes a submission of the ThinkIT Speech Lab at Institute of Acoustics, Chinese Academy of Sciences to the onset detection contest of the Music Information Retrieval Evaluation eXchange (MIREX) 2006. This submission presents an algorithm using spectral flux to detect musical onsets. Firstly, a detection function is generated via spectral flux. Then, a peak picking procedure is applied on this function to extract the onset points. Finally, the evaluation result of the submitted algorithm is presented with discussion and analysis.

Keywords: MIREX, onset detection, spectral flux

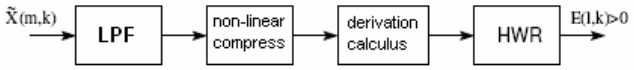
1. System Overview

The submitted onset detection algorithm mainly uses the spectral flux [1] to generate detection function for measuring the musical onset regions, followed by a peak picking procedure to extract the onset points in the audio stream.



Onset detection system overview

The detection function is generated via the flux of spectral energy, the main steps include (1) FFTs based on time sequence to build a spectrogram for the input audio, (2) low pass filtering, μ-law compression, Canny-operator based differentiating and half-wave-rectification for each frequency channel of the spectrogram, (3) summating all frequency channels together to form the detection function. The information of spectral phase is ignored due to the tiny contribution to the overall performance.



Processing performed at each frequency channel

The peak picking procedure is mainly achieved by thresholding, the main steps include (1) picking up all local maxima, (2) computing threshold based on the standard deviation of the detection function, (3) picking up all the local maxima which are beyond the threshold, (4) merging the onset points which are too close to each other. All the surviving onset points make up of the final result.

2. Description of Algorithm

The components of the presented algorithm are described detailedly in this section. Firstly, the components used in generating the detection function are introduced, and then all the arts utilized in the peak picking procedure to extract onset points are presented. The input audio stream is down-sampled into 16 kHz with uniform format of 16 bits, mono channel.

2.1 Generating detection function

Firstly, a computation of spectrogram is achieved by applying FFT on each frame of the audio stream. The frame length is 16ms, and the FFT size is set as 512 which is a double size of the frame length to promote the spectral resolution. Further processing is applied on each frequency channel of the spectrogram, which is represented as a function of spectral energy at a certain FFT bin with respect to time. After each frequency channel being processed, all channels are integrated together to form the detection function. Below are the detailed processes applied in each frequency channel.

2.1.1 Low pass filtering

A low pass filtering operation is first applied on each frequency channel to extract the spectral energy envelope.

This procedure is achieved by convolving the frequency channel function with a Half-Hanning window [2, 4] which has a low-pass characteristic. The length of the Half-Hanning window is set as 50ms in our algorithm.

2.1.2 μ-law compression

After low pass filtering, a μ-law compression [3] is applied on each filtered channel to achieve a non-linear compression effect. This procedure together with the following differentiating procedure is a psycho-acoustic relevant process that can catch a tiny but perceptible spectral energy change more precisely. The compression factor μ which determines the degree of compression is set as 100 in our algorithm.

2.1.3 Differentiating

After μ-law compression, a differentiating procedure is applied on each filtered and compressed channel to transform the sudden rises of spectral energy into narrow peaks. The differentiating is achieved by using Canny-operator which is widely used in image processing [4]. The σ, which controls the shape of the operator, is set as 1 in our algorithm.

2.1.4 Half-wave rectification (HWR)

In a similar way, after differentiating, there will be negative peaks exhibited in each channel when the spectral energy drops, marked as an offset. Since we are only interested in onsets, a half-wave rectification (HWR) is applied to only preserve the positive peaks in each channel.

2.2 Extracting onset points

When the detection function is generated, a peak picking procedure is applied on this function to search out all the onset points. Below are the detailed steps.

2.2.1 Local maxima searching

Local peaks' positions and heights are detected in the detection function with a running window method: local maxima are detected at the indexes whose values are higher than those of their neighbours within 25ms.

2.2.2 Thresholding

The threshold is computed as the product of an adjustable coefficient and the standard deviation of the detection function, whose value is computed only based on the non-zero values in the function. Then all the local maxima beyond the threshold are picked as onset candidates.

2.2.3 Candidates merging

The minimum duration between every two reasonable onsets is set as 100ms. Therefore, all the candidate points, which are too close to each other, are merged together by preserving the one with greater magnitude and deleting the weaker one. All the surviving onset points make up of the final result.

3. Implementation

The implementation of this algorithm is achieved by C++ and built in Win32 environment with Intel C++ Complier 9.0. The implementation consists of a front-end utilizing the program of "sox" [5] to achieve resample for the input audio steam, whose format is required to be the Windows PCM with WAV header.

4. Evaluation Result

5. Acknowledgement

This work is supported by Chinese National 973 program (2004CB318106) and National Natural Science Foundation of China (10574140, 60535030). Many thanks to the IMIRSEL for running the evaluation.

References

[1] M. Alonso, "Extracting Note Onsets from Musical Recordings", *Proc. IEEE International Conference on Multimedia and Expo*, 2005

[2] E. Scheirer, "Tempo and Beat Analysis of Acoustic Music Signals," J. Acoust. Soc. Am., vol. 103, no. 1, pp. 588-601, Jan. 1998

[3] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the Meter of Acoustic Musical Signals", *IEEE Trans. Audio, Speech, and Language processing*, vol.14, no.1, Jan. 2006

[4] Lie Lu, Hong-Jiang Zhang. "Automated Extraction of Music Snippets ", *Proc. ACM Multimedia 03*, pp.140-147, Berkeley CA, Nov. 2-8, 2003

[5] http://sox.sourceforge.net/

Onset Detection in Polyphonic Signals by means of Transient Peak Classification MIREX 2006 – Onset Detection

A. Röbel
IRCAM-CNRS –STMS
1, pl Igor-Stravinsky
75004 Paris, France
axel.roebel(at)ircam.fr

Abstract

The article describes an onset detection algorithm that is based on a classification of spectral peaks into transient and non-transient peaks and a statistical model of the classification results to prevent detection of random transient peaks due to noise. This article describes the algorithmic changes compared to last years submission and discusses the conclusions drawn from the evaluation results.

Keywords: Onset detection. Peak classification.

1. INTRODUCTION

In the following article we are going to describe a transient detection algorithm that has been developed for a special application, the detection of transients to prevent transformation artifacts in phase vocoder based (real time) signal transformations [6, 7]. This application requires a number of special features that distinguishes the proposed algorithm from general case onset detection algorithms: The detection delay should be as short as possible, frequency resolution should be high such that it becomes possible to distinguish spectral peaks that are related to transient and non transient signal components, for proper phase reinitialization the onset detector needs to provide a precise estimate of the location of the steepest ascend of the energy of the attack. In contrast to this constraints the application does not require the detection of soft onsets, where a soft onset is characterized by time constants equal to or above the length of the analysis window. This is due to the fact that such onsets are sufficiently well treated by the standard phase vocoder algorithm. False positive detections are not very problematic as long as they appear in noisy time frequency regions. A major distinction is that a single onset may be (and very often is) composed of multiple transient parts, related either to a slight desynchronization of polyphonic onsets or due to sound made during the preparation of the sound (gliding fingers on a string). While these desynchronized transients are generally not considered as independent onsets they nevertheless constitute transients which should be detected for the intended application.

The evaluation of the transient detection algorithm for onset detection and music segmentation tasks has revealed that the detection results are comparable with existing algorithms for onset detection or signal segmentation tasks [8]. Therefore, it is now the major means for signal segmentation and onset detection in IRCAM's AudioSculpt application [1]. Since MIREX 2005 a number of improvements have been added which should improve the performance with respect to onset detection and which we are interested to evaluate on the MIREX database.

In the following article we will describe the algorithm only briefly, and we refer the reader to the article published during MIREX 2005 [8]. Besides that we will discuss the improvements of the original algorithm since MIREX 2005.

2. Fundamental Strategy

There exist many approaches to detect attack transients. For a number of current approaches see [2, 5, 4, 9]. In contrast to the evaluation of energy evolution in integral frequency bands, a criterion that most of the approaches are relying on, the following article proposes a two stage strategy which first classifies the spectral peaks in a standard DFT spectrum into peaks that potentially may be part of an attack transient and those that are not. Based on this classification a statistical model of background transient peak activity is employed to detect transient events. The advantage of this two stage approach is that the transient components of the signal are classified with rather high frequency resolution, allowing a precise distinction between transient and non transient signal components.

The basic idea of the proposed transient detection scheme is straightforward. A peak is detected as potentially transient whenever the center of gravity (COG) of the time domain energy of the signal related to this peak is at the far right side of the center of the signal window. Note, that it can be shown [8] that the COG of the energy of the time signal and the normalized energy slope are two quantities with qualitatively similar evolution and, therefore, the use of the COG of the energy for transient detection instead of the energy evolution appears to be of minor importance.

3. From transient peaks to onsets

Unfortunately not every spectral peak detected as transient indicates the existence of an onset. Further inspection reveals that spectral peaks related to noise signals quite often have a COG far of the center of the window. In contrast to spectral peaks related to signal onsets these false transient peaks in noise are not synchronized in time with respect to each other. The synchronization of a sufficient number of transient peaks is the final means to avoid detection of noise peaks as onsets.

3.1. Transient energy ratio

Last years MIREX has shown that the normalized energy variation of the transient events is a further means to effectively distinguish random transient events from onsets.

As normalized energy variation (NEV) we define the maximum of the ratio between total signal energy in a transient frame and the transient energy in the same frame, where the maximization is done over the whole duration of the onset. As defined the NEV is bound between 0 and 1. In practical applications the NEV threshold is adapted interactively by the user who can adapt the threshold with direct feedback about the transients that persist. For this years evaluation we used the NEV as control parameter that will be changed for the different runs of the algorithm to create the precision/recall performance curves. We select the NEV threshold to cover the range $NEV = [0 - 0.36]$.

For last years evaluation the NEV was selected to be NEV$= 0.35$. Note, however, that due to the changes of the other parameter settings and the statistical model that was used, the NEV thresholds are not directly comparable.

4. Algorithmic improvements

4.1. Evaluation time grid

A detailed inspection of the algorithm has shown that transient conditions in the statistical model may be limited to only very short time ranges. This is especially true for weak onsets or onsets that appear within a large amount of background noise. An improvement can be easily obtained by means of decreasing the inter frame offset of the analysis frames of the underlying STFT. Compared to MIREX 2005 the frame step has been reduced from an 8th part of the window to an 24th part of the analysis window.

4.2. Limit onset time distance

The results of last years MIREX have shown that one of the major problems of the algorithm are double detections that may occur if multiple instruments have onsets with only slight delays. It is generally desirable to have a means to control the time density of onsets. Accordingly, the algorithm has been changed to allow the user to control the required distance between two detected onsets. If there is more than a single transient event that occurs in the allowed time distance then only the strongest one will be output. The strength of the transients are evaluated according to the NEV criterion described above.

4.3. Transient peak detection

The transient peak detector described in last years MIREX uses the center of gravity (COG) of the energy related to a single peak to determine whether the peak is part of a transient. As explained above, the peak is classified as transient, if the COG is sufficiently to the right of the center. Obviously, for a real transient peak, the signal duration should at the same time be shorter than the duration of the analysis window. Especially for noise peaks it sometimes happens, that the COG is far to the right and the duration is large. This cannot happen in reality because it would indicate that the signal related to the peak would extend outside of the analysis window.

Because the analysis window is fixed, the only way to explain this situation is by means of cancellation. If the part of the signal that lies outside of the analysis window is canceled by other peaks the overall signal stays within the analysis window. This cancellation does happen especially for noise peaks. To detect these transient peak artifacts and to prevent an impact on the transient detector a new mode of the transient peak detector has been developed. This mode requires a transient peak to have an COG offset that is above the user defined COG threshold, and at the same time requires the duration of the signal related to the peak to be smaller then the duration of the analysis window. This mode is enabled in submission 3 of the onset detector algorithm. Note, that the time duration of the signal related to the peak can be calculated directly from the peak spectrum [3].

4.4. Statistical model

The detection of an onset event requires that a sufficient number of synchronized transient peaks are detected. To establish a reasonable condition for the sufficient number we rely on a statistical model of the transient background activity that is due simply to random transient events in the background noise. The background activity is derived by means of a short time history of the detected transient peaks. The history is calculated independently for overlapping bands covering a time range of approximately the 3/4 of the analysis window. For each band the relative number of observed peaks that exceeded the transient threshold is used to determine the average transient probability in the frame history, which in turn is compared to the transient peak probability in the future time range covering approximately 1/4 of the analysis window.

The exact statistical model that is used to describe the transient peak events has been described in [8] and will not repeated here. We address here, the problem of the selection of the bands that are used to monitor the transient events. In the previous version of the algorithm a fixed size band has been used, the bandwidth of which was a priori given by the user. The major problem with the fixed bandwidth of the

Subm. Id	M[ms]	K	G	N_E	H[kHz]	A[dB]	T[ms]	Duration filter
1	36.3	1.7	2.6	15	10	-46	50	off
2	45.4	2.0	2.4	14	11	-46	50	off
3	45.4	1.9	2.4	14	10.5	-46	50	on

Table 1. Parameter settings of the three different submissions to the MIREX 2006 onset detection evaluation. M window size, K COG threshold factor, G transient confidence threshold, N_E minimum bandwidth of statistical model, H upper frequency limit of the spectrum to be used in the detector, A minimum amplitude level of a transient, T minimum distance between two onsets.

statistical model is the fact, that onset events may produce transient peak events with a large scale of different bandwidths. If the bandwidth of the statistical model is much smaller than the bandwidth of the event, the confidence calculated from the model is too small. However, if the bandwidth of the model is much larger than the bandwidth of the event we may not detect a narrowband transient event due to the fact that the variation compared to the background transient activity is too small. To resolve this problem the current version of the algorithm uses a statistical model with different bandwidths. The smallest possible bandwidth is given by the user and the algorithm uses the given bandwidth and all integer multiples of this bandwidths to monitor the background probability. Due to the fact that the models for the larger bandwidths can be calculated from the narrow bands, the calculation of the hierarchic models does not require a significant computational cost. However, it allows us to select the confidence threshold with respect to the optimal bandwidth such that the setup of the threshold is less signal dependent.

5. Parameter selection

There remain a number of user selectable parameters for the transient detector. The first one is the analysis window size M. With respect to this parameter there exist contradicting demands because on one hand attack transients of sinusoids that mix with stationary sinusoids will not be correctly detected such that frequency resolution should be high and window size large. On the other hand we can not detect more than one attack transient within a single window such that window size should be small. This is a variant of the well known time resolution/frequency resolution trade off for time frequency analysis.

The second parameter is the COG threshold. A simple theoretical investigation shows that for the noise free case the maximum COG normalized by the analysis window is 0.5 and for maximum robustness C_s should be close to this value. Due to background noise or preceding notes, however, part of the transient may be covered such that the maximum value of the observed COG will generally be lower than 0.5. As limiting case for a transient condition we consider a linear ramp that start at the very left end of the analysis window. Signals with COG smaller than this will not be detected. The parameter K is a multiplying factor of the COG of the linear ramp and it is used to control the COG threshold. The smaller K the more sensitive the detector is but at the same time the more random transient peaks may be detected in noise.

The third parameter is the confidence threshold G that is the confidence of the statistical model that the transient probability did change. The lower the confidence threshold the more sensitive the algorithm will be, again running the risk of false detections in noise.

The fourth parameter is the minimum bandwidth of the frequency bands that are used to obtain the statistical model for background transient activity. As explained in section **4.4** the statistical model will monitor the transient probability in a hierarchical manner. Therefore, the bandwidth parameter is not as important as in the previous version. The bandwidth N_E is specified in terms of the mainlobe width of the analysis window.

The fifth parameter is the highest frequency H that will possibly be included in the transient peak detection process. The sixth parameter, the minimum distance between detected transients T, has been discussed in section **4.2**. As last parameter we consider the minimum amplitude an onset needs to have to be accepted as onset event. This minimum amplitude A is expressed in terms of the full scale amplitude of the signal.

The parameters have been optimized using a set of hand labeled sound files containing mostly sharp attacks related to drum, bars, or plucked string onsets. Three sets of parameters have been selected to be part of the MIREX evaluation. The parameter settings used in the MIREX evaluation are given in table (4.3).

6. Discussion of the results

While we are happy to see that our algorithm did compare rather favorable with the other contributions, we don't believe that the evaluation can be used to compare the different algorithms.

The main problem is the fact that all algorithms have been trained and adapted using different data sets. Therefore, it appears questionable to draw any conclusions with respect to the ranking of the algorithms. The analysis of the different sound classes reveals that for the different classes different algorithms are "winners". This could be related to the fact that one algorithm is better, or it could be re-

lated to the fact that the algorithm has been trained with a training set that contained more examples for that class. As mentioned above our algorithm has been adapted using especially drums, plucked strings, bells and the like. Accordingly the algorithm should work best with these examples - which is the case. For all these classes the algorithm is rather successful. As a surprise we note that the algorithm works rather successfully well for the singing voice.

Therefore, in the following I will discuss only the ranking of the three versions of our algorithm and the version we send last year.

The first thing to remark is that the recall rate was rather low in all but the bars and bells classes. This indicates that the filtering due to the COG threshold and the confidence level was to strong for some of the data classes. Accordingly, even for a minimum value of the normalized transient energy threshold not all transients passed. This may be due to the fact that our training data base does not contain any sustained strings or wind instruments. We hope to be able to extend the training database to try adapting to a larger scope of sound classes for next years MIREX.

As a second point we note, that the duration filter used in submission three of the algorithm did not always improve the results. The filtering seems to have created an advantage especially for the wind instruments, which may be related to the blowing noise.

As a last remark we mention that the shorter window length of submission 1 seems to be appropriate especially for the voice example. We can only conjecture the reason for this fact. It may be related to vibrato in the singing voice.

Comparing the algorithm with last years version we find that the new versions are better for nearly all classes. Exceptions are the wind and the brass instruments for which the average F-measure have slightly lowered. This result is disturbing especially due to the fact that last years version had a fixed parameter set.

7. Acknowledgments

We would like to thank the team at IMIRSEL for their efforts to create this valuable opportunity to test the algorithms and different parameter settings on an independent data set.

References

[1] N. Bogaards, A. Röbel, and X. Rodet. Sound analysis and processing with audiosculpt 2. In *Proc. Int. Computer Music Conference (ICMC)*, 2004.

[2] J. Bonada. Automatic technique in frequency domain for near-lossless time-scale modification of audio. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 396–399, 2000.

[3] L. Cohen. *Time-frequency analysis*. Signal Processing Series. Prentice Hall, 1995.

[4] C. Duxbury, M. Davies, and M. Sandler. Improved time-scaling of musical audio using phase locking at transients. In *112th AES Convention*, 2002. Convention Paper 5530.

[5] P. Masri and A. Bateman. Improved modelling of attack transients in music analysis-resynthesis. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 100–103, 1996.

[6] A. Röbel. A new approach to transient processing in the phase vocoder. In *Proc. of the 6th Int. Conf. on Digital Audio Effects (DAFx03)*, pages 344–349, 2003.

[7] A. Röbel. Transient detection and preservation in the phase vocoder. In *Proc. Int. Computer Music Conference (ICMC)*, pages 247–250, 2003.

[8] Axel Röbel. Onset detection in polyphonic signals by means of transient peak classification. In *MIREX Online Proceedings (ISMIR 2005)*, London, Great Britain, September 2005. avail. at `http://www.music-ir.org/evaluation/mirex-results/articles/onset/roebel.pdf`.

[9] X. Rodet and F. Jaillet. Detection and modeling of fast attack transients. In *Proc. Int. Computer Music Conference (ICMC)*, pages 30–33, 2001.

TEMPO EXTRACTION FOR AUDIO RECORDINGS

Miguel Alonso, Bertrand David and Gaël Richard
GET–Télécom Paris
46 rue Barrault, Paris 75634
cedex 14, France
{miguel.alonso,bertrand.david,gael.richard}@enst.fr

ABSTRACT

Nowadays, the problem of estimating the tempo of audio recordings receives a large amount of attention from the automatic audio processing community. Applications for this task include automatic playlist generation, synchronization of audio tracks, computer based music transcription, music information retrieval... This paper briefly presents a technique for estimating and tracking the tempo of audio recordings. This approach relies on a front end that estimates phenomenal accents (onsets) and their respective time location. The second step consists in a periodicity detection block that calculates the beat rate of the audio signal and it is followed by a dynamic programming stage that performs tempo tracking. This

Keywords: energy flux, phenomenal accents, dynamic programming.

1 Description of the first algorithm

This year we submit two algorithms. The first one corresponds to the same proposal submitted last year for Mirex'05 with some minor modifications. In addition, we also submit another algorithm based on a similar principle, but using more recent versions of the corresponding system components.

It is assumed that the beat of the audio signal is relatively constant, at least during the duration of the tempo analysis window.

The system that we proposed can be divided into four major steps:

- *phenomenal accent detection:* also called onset detection, refers to locating discrete temporal events in an audio stream where there is a marked change in any of the perceived psychoacoustical properties of sound: loudness, timbre and pitch (Lerdahl and Jackendoff, 1983);

- *periodicity estimation:* consists in detecting the rate at which phenomenal accents appear;
- *periodicity tracking:* this part is carried out by a dynamic programming algorithm that finds the best periodicity path through succesive tempo analysis frames;
- *Path selection and tempo salience estimation:* this module selects the two main tempi from the various periodicity paths found by the tracking algorithm, then it estimates the relative salience of the lower tempo.

The general overview of the system is presented in Figure 1. The algorithm works as follows: the input signal is resampled at a lower frequency ($f_s = 22,050$ Hz) to obtain $x(n)$. This is done in order to reduce the computational burden, knowing that the rhythmic properties of the original audio remain unaltered. Next, $x(n)$ is decomposed into eight-uniform non-overlapping subbands using a maximally decimated polyphase filter bank.

A musical stress profile indicating the potential location of phenomenal accents is computed for each subband signal. This was done using the system presented in (Alonso et al., 2005), where a perceptually plausible power envelope is calculated. Then its derivative is computed using an efficient differentiator filter and a detection function that bears onsets as peaks is obtained.

The periodicity of the detection function is obtained using three methods widely employed in pitch estimation: the summary autocorrelation function, the spectral sum and the spectral product. The procedure followed is explained in (Alonso et al., 2004). After processing each tempo analysis window, the output of every method is stored in its respective time–frequency matrix. Then, a dynamic programming algorithm is used to determine and track the optimum paths of tempo candidates in each analysis frame. For the Mirex contest, to compute the tempo ($\mathbb{T}$) of the excerpt under analysis, the optimum paths found by every periodicity method are first time-averaged followed by an inter-algorithm average.

In the last part, the two main tempi are selected from the set of paths computed by the tracking algorithm. The selection criteria combines path energy (salience) and *a priori* information about the human preferences concerning the beat period as suggested by Moelants (2002) and McKinney and Moelants (2004).

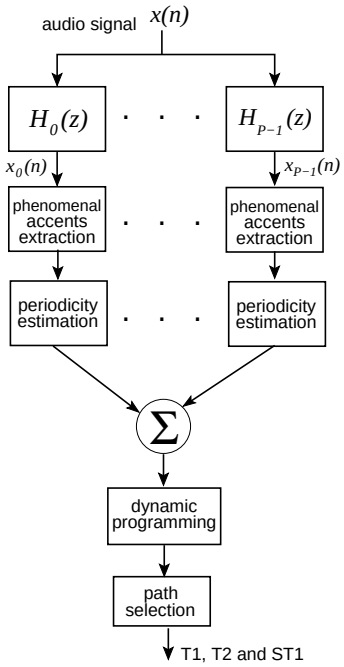


Figure 1: Overview of the system.

2 Description of the second algorithm

This approach bears close resemblance to the algorithm described above. In this variant, the musical stress estimation block has been improved. Instead of computing the traditional spectrogram, this variant uses a reassigned version to enhance simultaneously the resolution in time and frequency.

This algorithm uses only the spectral sum for periodicity estimation, instead of the three technies used in the previous approach. Additionally, the periodicity tracking block has also been upgraded to improve its robustness to tempo variations. Finally, the tempi selection block uses a slightly different scheme for the selection of periodicity paths since only period estimations from one are available.

References

M. Alonso, B. David, and G. Richard. Tempo and beat estimation of music signals. In *Proc. Int. Symposium on Music Inf. Retriev (ISMIR)*, 2004.

M. Alonso, G. Richard, and B. David. Extracting note onsets from musical recordings. In *Proc. IEEE Int. Conf. on Multimedia & Expo (ICME)*, 2005.

F. Lerdahl and R. Jackendoff. *A generative theory of tonal music*. MIT Press, Cambridge, Massachusetts, 1983.

M. F. McKinney and D. Moelants. Extracting the perceptual tempo from music. In *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, 2004.

D. Moelants. Preferred tempo reconsidered. In *Proc. of the 7th Int. Conf. on Music Perception and Cognition*, pages 580–583, 2002.

A Tempo Extraction Algorithm for Raw Audio Recordings

Iasonas Antonopoulos, Aggelos Pikrakis and Sergios Theodoridis
Department of Informatics and Telecommunications
University of Athens
Panepistimioupolis, 15784, Athens, Greece
{jantonop, pikrakis, stheodor}@di.uoa.gr

Abstract

This paper presents a tempo extraction algorithm for raw, polyphonic audio recordings, assuming that music meter and tempo remain constant throughout the recording. The method was submitted at MIREX 2006, in the context of the Audio Tempo Extraction evaluation task. Our approach is based on the self-similarity analysis of the audio recording and does not assume the presence of percussive instruments. In order to account for the contest's requirements, which emphasizes on perceptual tempo tracking, the proposed method returns two tempi, ranging from 40bpm to 320bpm, along with their relative strength.

Keywords: Tempo Extraction, Self Similarity Analysis

1. Description of the Algorithm

1.1. Feature Extraction

The proposed algorithm is a variant of previously published work by the authors ([1]). At a first step, each raw audio recording is divided into overlapping long-term segments, each of which has a duration equal to 6 seconds (long-term step has been set equal to 0.5 seconds). For each long-term segment, a short-term moving window generates a sequence of feature vectors. Approximate values for the length of the short term window and overlap between successive windows are 100ms and 95ms respectively. The adopted feature is similar with the standard MFCCs, however the filter bank consists of overlapping triangular filters, whose center frequencies align with the chromatic scale of tones (starting from 110Hz and reaching up to approximately 7KHz).

Let $\mathbf{F} = \{f_1, f_2, \ldots, f_N\}$, be the feature sequence that is extracted from a long-term segment. Sequence F serves as the basis to calculate the Self Similarity Matrix (SSM) of the segment, using the Euclidean function as a distance metric. Since the SSM is symmetric around the main diagonal, in the sequel it suffices to focus on its lower triangle.

At a next step, the mean value of each diagonal of the SSM is calculated. If B_k stands for the mean value of the kth diagonal, then:

$$B_k = \frac{1}{N-k} \sum_{l=k}^{N} ||f_l, f_{l-k}||$$

where $N-k$ is the length of the k-th diagonal and $||.||$ is the Euclidean distance function. In the sequel, we will refer to the k-th diagonal as the k-th lag. If B is treated as a function of k, then its plot against k exhibits certain local minima (valleys) for a number of ks. Each valley can be interpreted as corresponding to a periodicity, that is inherent in the long-term segment being examined. In addition, the difference of lags between any two valleys can also reveal inherent periodicities.

1.2. Tempo Estimation

Our method is based on the assumption that the perceived tempi correspond to periodicities that appear as valleys of B. In order to extract a pair of dominant periodicities (i.e., two perceived tempi), each long-term segment is processed as follows:

1. All "dominant" valleys of B are detected. A valley is considered to be "dominant" if it is the deepest one in a neighborhood of lags. The width of the neighborhood is a predefined constant. The lags of detected valleys are sorted in ascending order and the difference of lags between successive valleys is calculated. All differences are placed in a histogram. This histogram exhibits peaks at certain values.

2. In the sequel, the width of the neighborhood for calculating dominance is increased and all dominant valleys are again detected for this new width. The differences of lags of detected valleys are placed in a new histogram.

3. The peaks of the above two histograms are then detected and all possible pairs of lags corresponding to these peaks are formed. The ratio of lags in each pair is then examined in order to decide whether it approximates sufficiently one of the following music meter ratios, namely: $\{\frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}, \frac{8}{8}, \frac{9}{8}\}$, or $\{\frac{2}{4}, \frac{3}{4}, \frac{4}{4}, \frac{5}{4}\}$, depending on the value of the lag of the first component of the pair under consideration. The best pair is selected by also taking into account the height of

peaks corresponding to the lags of the pair in the respective histograms.

The above procedure yields one pair of lags per long-term segment. At a final stage the lags appearing in all winner-pairs are again placed in a histogram. This last histogram exhibits certain peaks. These peaks are examined in pairs and the procedure of step 3 is repeated to yield a winning pair. The two lags of this pair are returned by the algorithm as the two most dominant periodicities (perceived tempi).

The proposed method took the 6th place in the MIREX-2006 tempo extraction contest. The performance of the algorithm is summarized in Table 1. The method was implemented in Matlab (v7.0) and was cross-compiled to generate the submitted executable.

At least one tempo correct	**Both tempi correct**	**P-score**
84.29% (5th place)	47.86% (3d place)	0.669 (6th place)

Table 1. Performance of the proposed method

References

[1] Aggelos Pikrakis, Iasonas Antonopoulos and Sergios Theodoridis, "Music Meter and Tempo Tracking from raw polyphonic audio", in *Proc. of ISMIR 2004*, Barcelona, Spain, Oct. 2004.

Audio Tempo Extraction Algorithm for MIREX 2006

Anssi Klapuri
Institute of Signal Processing, Tampere University of Technology
Korkeakoulunkatu 1, 33720 Tampere, Finland
anssi.klapuri@tut.fi

Abstract

This paper describes an audio tempo extraction algorithm submitted to the MIREX 2006 contest. The algorithm is identical to the one submitted to MIREX contest in 2004, and has been described in detail in the article "Analysis of the Meter of Acoustic Musical Signals" published in IEEE Trans. Audio, Speech and Language Processing, 14(1), 2006. In summary, the method analyses musical meter jointly at three time scales, of which only the tactus (beat) level is used here. The output tempo is obtained as the median inter-beat-interval during the latter half of the analysed signal.

Keywords: Tempo estimation, musical meter analysis, beat tracking.

1. Introduction

The tempo extraction algorithm described here is identical to that submitted to MIREX contest in 2004. It is based on the meter analysis method originally described in [1]. [1]

The employed algorithm estimates only one tempo value for the analysed signal. This is calculated as a median of the inter-beat-intervals during the latter half of the analysed signal. In order to conform to the task description in MIREX 2006 contest, another tempo value is obtained by doubling or halving the first estimate towards the mean tempo of 109 beats per minute (BPM). The latter estimate is assigned a small weight value.

2. The underlying meter analysis algorithm

The aim of the method proposed in [1] is to estimate the meter of acoustic musical signals at three levels: at the tactus, tatum, and measure-pulse levels. An overview of the method is shown in Fig. 1.

For the time-frequency analysis part, a technique is employed which aims at measuring the degree of spectral change, or, "accent" in music signals. In brief, preliminary time-frequency analysis is conducted using a quite large number of subbands and by measuring the degree of spectral change at these channels. Then, adjacent bands are combined to arrive at four bandwise accent signals, for which periodicity analysis is carried out.

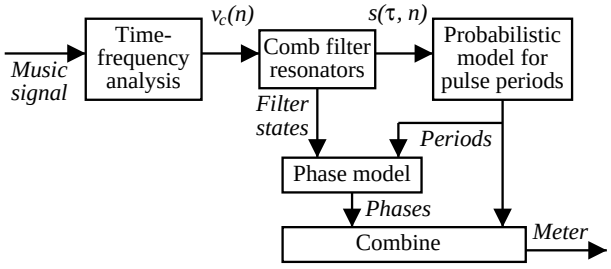


Figure 1. Overview of the meter analysis method, on which the tempo extraction algorithm is based.

Periodicity analysis of the bandwise accent signals is performed using a bank of comb filter resonators very similar to those used by Scheirer in [2]. Before we ended up using comb filters, four different period estimation algorithms were evaluated. A bank of comb filter resonators was chosen because it was the least complex among the three best-performing algorithms.

The comb filters serve as feature extractors for two probabilistic models. One model is used to estimate the period-lengths of metrical pulses at different levels. The other model is used to estimate the corresponding phases (see Fig. 1). The probabilistic models encode prior musical knowledge regarding well-formed musical meters. In brief, the models take into account the dependencies between different pulse levels (tatum, tactus, and measure) and, additionally, implement temporal tying between successive meter estimates.

3. Acknowledgements

The author is grateful to Jouni Paulus who wrote a C++ implementation of the method based on the Matlab (and some C) codes of the author.

References

[1] 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.

[2] E. Scheirer, "Tempo and beat analysis of acoustical musical signals," *Journal of the Acoustical Society of America*, vol. 103, pp. 588–601, Jan. 1998.

[1] Available at www.cs.tut.fi/sgn/arg/klap/sapmeter.pdf.

Simple But Effective Methods for QBSH at MIREX 2006

J.-S. Roger Jang, Nien-Jung Lee, and Chao-Ling Hsu
Department of Computer Science
National Tsing Hua University, Taiwan
{jang, qmonster, leon}@wayne.cs.nthu.edu.tw

Abstract

This extended abstract describes a submission to the QBSH (Query by Singing/Humming) task of MIREX (Music Information Retrieval Evaluation eXchange) 2006. The methods used for both subtasks 1 and 2 are introduced together with the evaluation results Comments and suggestions for further QBSH task are also addressed in the paper.

Keywords: MIREX, Query by Singing/Humming (QBSH), LS (Linear Scaling), DTW (Dynamic Time Warping).

1. Overview of QBSH Task

The goal of QBSH (Query by Singing/Humming) task at MIREX 2006 is to evaluate MIR systems that take sung or hummed query input from real-world users. QBSH task consists of two subtasks:

- Subtask 1: Known-Item Retrieval
  - Input: 2797 sung/hummed queries of 8 seconds.
  - Test database: 48 ground-truth MIDIs + 2000 Essen Collection MIDI noise files.
  - Evaluation: Mean reciprocal rank (MRR) of the ground truth computed over the top-20 returns.
- Subtask 2: Queries as Variations
  - Input: 2797 sung/hummed queries + 48 ground-truth files of 8 seconds
  - Test database: 48 ground-truth MIDIs + 2000 Essen MIDI noise files + 2797 sung/hummed queries.
  - Evaluation: The precision based on the number of songs within the same ground-truth class of the query calculated from the top-20 returns for each of the 2845 queries.

2. QBSH Corpus

The QBSH corpus provided by Roger Jang [1] consists of recordings of children's songs from students taking the course "Audio Signal Processing and Recognition" over the past 4 years at CS Dept of Tsing Hua Univ., Taiwan. The corpus consists of two parts:

1. MIDI files: 48 monophonic MIDI files of ground truth.
2. WAV files: 2797 singing/humming clips from 118 subjects, with sampling rate of 8 KHz and bit resolution of 8 bits.

For each of the WAV file, the corpus provides another two files distinguished by their file extensions, including PV (files of pitch vectors, derived with a frame size of 256 and zero overlap), and MID (midi files). PV files are pitch vectors labelled manually by the student who recorded the clip. MIDI files were generated from the PV files through a simple note segmentation algorithm. The participants may choose any one of the formats as the input to their systems.

The recording count of each MIDI file is shown in Figure 1.

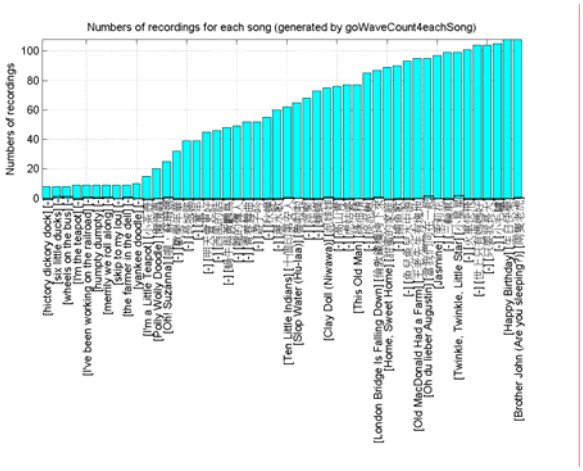


Figure 1. Recording count of each MIDI file.

3. Our Approaches

Since all the query data are available, we have to choose a simple but effective distance measure, which do not run into the potential problem of over fitting/training. Under this guideline, our primary candidates are

- DTW: Dynamic Time Warping [2, 3]
- LS: Linear Scaling [4]
- LS+DTW: LS plus DTW [5]

Then we need to decide which files to be used as the input to our system. Apparently, WAV and PV should be better ones since MID is derived from PV. In order to decide to use WAV or PV, we performed an evaluation similar to subtask 1, where 2000 MIDIs are selected from the Internet as a replacement for Essen Collection. When WAV files are used, we employed a robust pitch tracking algorithm based on dynamic programming to extract the pitch vectors. The result is shown in Figure 2.

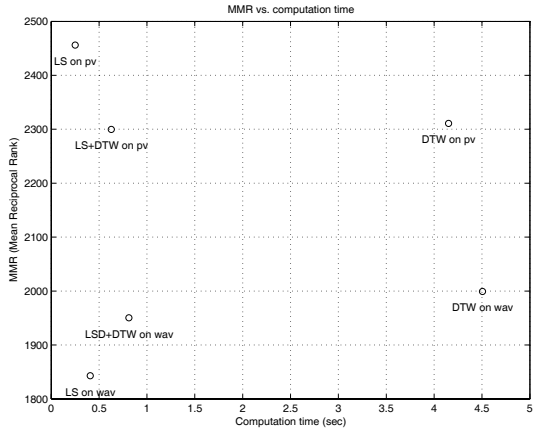


Figure 2. MMR vs. computation time for several methods on PV and WAV files.

Our evaluation demonstrates that PV can always achieve better performance than WAV, as shown in Figure 2, since PV files contain pitch vectors labelled manually. In fact, we still found some mistakes in PV, which should be corrected later in order to make the QBSH corpus more trustworthy. The performance on WAV files is not as good, primarily due to the fact that the WAV files are recorded by 126 subjects at different PCs with different microphone setups, hence it is hard to do both endpoint detection and pitch tracking accurately.

We did not try MID files as the query set since our algorithm is based on pitch vectors (frame-based) instead of music notes.

Since computation time is not really an issue in this task, we used only DTW and LS in our evaluation. Based on the evaluation criteria for both subtasks, we found that LS is the best method for subtask 1 and DTW is the best method for subtask 2.

4. Results

Out simple distance measures do prove to be effective in both subtasks. In subtask 1, an MRR (mean reciprocal rank) of 0.883 is achieved, ranked 3 among 13 participants. For subtask 2, an MP (mean precision) of 0.926 is achieved, ranked 1 among 10 participants. Figure 3 demonstrates the evaluation results for both subtasks.

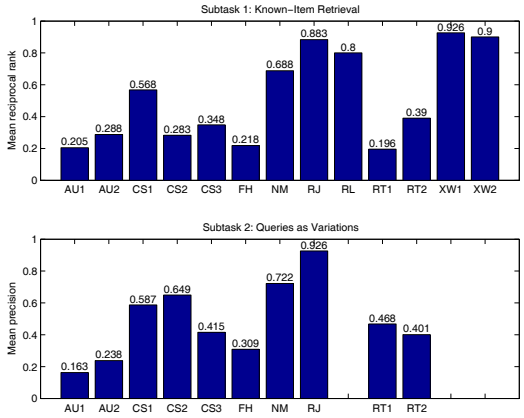


Figure 3. Evaluation results of two subtasks of QBSH

5. Comments on QBSH Task

For a comprehensive evaluation of QBSH in the coming year, we have several comments/suggestions:

Preparation of a test set

Ideally, the test set should not be accessible to any participant beforehand. One way to achieve this is to require every participant to submit a set of recordings to IMIRSEL team to be used as the test set. The test set should be released after the evaluation results are publicized. By following this convention, we should be able to increase our QBSH corpus year by year and new effective methods can be identified accordingly.

More participation

For this year, we have only 13 participants for subtask 1, 10 participants for subtask 2. We should try to encourage more participants since there are much more people working on QBSH.

Variations of QBSH Task

1. Use WAV exclusively as the query input: This is closer to the real-world situation where a QBSH system has to deal with acoustic input to pitch vector conversion using a pitch tracking algorithm.
2. Use MP3 as the test database: This is far more practical then using monophonic MIDIs as the test database. Of course, this is also far more challenging since audio melody extraction is well-known as a tough task in MIREX.

References

[1] J.-S. Roger Jang, "QBSH: A Corpus for Designing QBSH (Query by Singing/Humming) Systems", available at the "QBSH corpus for query by singing/humming" link at the organizer's homepage at http://www.cs.nthu.edu.tw/~jang.

[2] J.-S. Roger Jang and Ming-Yang Gao, "A Query-by-Singing System based on Dynamic Programming", International Workshop on Intelligent Systems Resolutions (the 8th Bellman Continuum), PP. 85-89, Hsinchu, Taiwan, Dec 2000.

[3] J.-S. Roger Jang, Hong-Ru Lee, "Hierarchical Filtering Method for Content-based Music Retrieval via Acoustic Input", The 9th ACM Multimedia Conference, PP. 401-410, Ottawa, Ontario, Canada, September 2001.

[4] J.-S. Roger Jang, Hong-Ru Lee, Ming-Yang Kao, "Content-based Music Retrieval Using Linear Scaling and Branch-and-bound Tree Search", IEEE International Conference on Multimedia and Expo, Waseda University, Tokyo, Japan, August 2001.

[5] Jyh-Shing Roger Jang, Hong-Ru Lee, Jiang-Chuen Chen, and Cheng-Yuan Lin, "Research and Development of an MIR Engine with Multi-modal Interface for Real-world Applications", Journal of the American Society for Information Science and Technology, 2004.

[6] J.-S. Roger Jang, Chao-Ling Hsu, Hong-Ru Lee, "Continuous HMM and Its Enhancement for Singing/Humming Qurey Retrieval", International Symposium on Music Information Retrieval 2005, London, UK, Sept 2005.

[7] J.-S. Roger Jang, Jiang-Chun Chen, Ming-Yang Kao, "MIRACLE: A Music Information Retrieval System with Clustered Computing Engines", 2nd Annual International Symposium on Music Information Retrieval 2001, Indiana University, Bloomington, Indiana, USA, October 2001.

Tararira: Query By Singing System

Ernesto López, Martín Rocamora
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería de la Universidad de la República
Julio Herrera y Reissig 565 – (598) (2) 711 09 74, Montevideo, Uruguay
elopez,rocamora@fing.edu.uy

Abstract

This extended abstract details a submission to the Music Information Retrieval Evaluation eXchange in the Query by Singing/Humming task. The problem of query by singing consists of building a machine capable of simulating the cognitive process of identifying a musical piece from a few sung notes of its melody. In this work, the algorithms of pitch tracking, onset detection and melody matching used in the system Tararira [1] are briefly described. Much effort has been put on automatic transcription of singing voice as it is a key factor in the overall performance. A novel way of combining note by note matching with the approach based on pitch time series matching is introduced.

Keywords: QBH, MIREX, melody matching.

1. Introduction

Through the last decade, different approaches to face the query by singing problem were considered. In all the proposals, the database consist of music in symbolic notation, generally MIDI, instead of raw or compressed audio as there is no sufficiently robust automatic way to extract the melody directly from a recording to compare it with the query.

The systems proposed can be divided, from its representation and matching technique, basically into two approaches. The traditional approach is based on note by note comparison [2][3], whereas a more recent approach utilizes the comparison of fundamental frequency time series [4][5]. The first approach consist of transcribing the voice signal into a sequence of notes and searching for the best occurrences of this pattern on database of melodies. Due to the performance decrease produced by transcription errors, the other approach avoids the automatic transcription, comparing melodies as fundamental frequency time series. Unfortunately, this implies working with long sequences (very long compared to sequences of notes) therefore computational time becomes prohibitive. Moreover, it is necessary to require the user to sing a previously defined melody fragment [4][5].

In the system Tararira [1] a novel way of combining both approaches is introduced, that preserves the advantages of each of them. Firstly, the system selects a reduced group of candidates from the database, using note by note matching. Then, the selection is refined using fundamental frequency time series comparison.

The system architecture is divided in two main stages, as depicted in figure 1. The first one is the transcription of the query into a sequence of notes. In the second one, this sequence is matched to the melodies stored in the database, and a list of musical pieces is retrieved, in a similarity order.

The transcription stage involves the following tasks:

- To estimate the fundamental frequency contour to set the note pitches.
- To segment the audio signal in order to establish the onset time and duration of notes.
- To perform a melodic analysis to adjust the note pitches to the equal tempered scale.

The tasks of the matching stage are:

- To codify the note sequence so as to obtain key and tempo transposition independence in the matching.
- To set flexible similarity rules to take into account query ornaments or mistakes, and automatic transcription errors.
- To refine the candidates selection, avoiding automatic transcription errors, by comparing fundamental frequency time series.

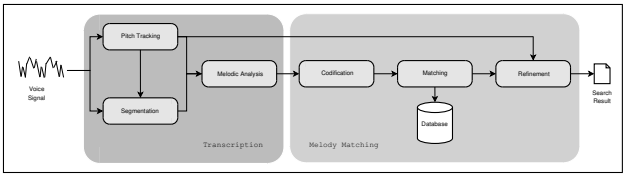


Figure 1: Block diagram of the system.

Work partly supported by Comisión Sectorial de Investigación Científica (CSIC).

2. Singing voice transcription

The goal of the automatic transcription is to extract from the audio signal the sequence of notes that best represents the sung melody. To do that, the events with greater probability to correspond to notes are identified. Each event is characterized by three values: pitch, onset time and duration.

In this work, much effort is put on the automatic transcription of the singing voice, as it remains an unsolved problem. The singing voice is one of the most difficult musical instruments to deal with.

2.1. Pitch tracking

To establish the notes pitch, the evolution of the fundamental frequency (F0) of the singing voice must be estimated. There are very well known techniques to do that. The implemented algorithm utilizes the Difference Function [6], a variation of the Autocorrelation Function.

2.2. Audio segmentation

The transcription of the query requires to establish each note onset time and duration. This problem is known as automatic audio segmentation into notes, and is the most difficult part of the automatic transcription. The singing voice has a set of features that make the note boundaries diffuse, and so, hard to identify.

It is possible to distinguish different note onsets in a singing voice signal. For notes sung with syllables starting with occlusive phonemes (such as, /ta/), the sudden energy release produces hard onsets, that are shown as a big signal energy increments. When a note starts with a gradual energy increase (for example, a nasal consonant), the onset is softer and therefore difficult to establish.

The algorithm implemented, in order to get a robust segmentation, looks for signs of events in the amplitude envelope as well as in the fundamental frequency contour. In the first stage, events corresponding to energy changes are detected. The algorithm calculates energy envelopes from different frequency bands [7], as generally the events appear more clearly in some band than in the envelope of the signal. The most salient ones are considered genuine note onsets. On a second stage, the weaker events are validated if they show a pitch change. Finally evident pitch changes that do not show an energy increment are identified (e.g. legato). However, this is not an easy task because the expressiveness of the performance and the lack of training of inexperienced singers introduce a set of features in the frequency contour that can be wrongly considered as additional notes (soft transition, spikes, instabilities, vibrato) [8].

2.3. Adjustment to the equal tempered scale

To assign a pitch value to each note, first of all it is necessary to approximate the fundamental frequency contour to a single frequency value. Then, a note pitch from the equal tempered scale is associated to this frequency value. Singers, specially untrained ones, are unable to sing according to a tuning system, so the query does not respect the reference and intervals of the equal tempered scale. It becomes necessary to adjust the natural deviation between the sung melody and the equal tempered scale.

The adjusting technique used assumes the hypothesis that when singing a melody a reference tone is held in mind, and that tempered note intervals relative to this reference are sung [8]. The method consists in estimating the reference tone through the most frequent deviation from the equal tempered scale, and in this way adjusting the pitch of the sung notes. Estimating the reference tone through the most frequent deviation takes into account that besides the deviation from the absolute equal tempered scale, in some note intervals an additional error may exist that is related to the difficulty of singing them.

3. Melody matching

A melody can be identified in spite of being performed at different pitch and at different tempo. However, some changes modify the melodic line but still allow the melody to be recognized, like sporadic pitch and duration errors or expressive features. The independence to the specific pitch and tempo is carried out in the note's encoding. By means of flexible similarity rules in the matching stage it is possibly to achieve tolerance to modifications due to ornaments or mistakes of the query, and automatic transcription errors.

In the system developed, a two stage approach is performed: firstly, a small group of candidates is selected based on notes comparison and then the search is refined using pitch time series.

3.1. Note sequences matching

Working with note sequences, the melody matching problem is basically an approximate string matching problem.

Encoding

The pitch transposition invariance is obtained by encoding the pitches sequence $A = (a_1, a_2, \ldots, a_n)$ as the sequence of intervals $\overline{A} = (a_2 - a_1, a_3 - a_2, \ldots, a_n - a_{n-1})$. It is evident that a sequence A' transposition of A has the same interval representation.

Ideally, tempo invariance should be obtained by normalizing the notes duration to a tempo invariant reference duration, for example, the duration of a beat, as in written notation. Unfortunately, it is not always possible to automatically estimate the tempo from a sung melody. A simple substitute for the beat is the duration of the previous event [9]. Given the duration sequence, $D = (d_1, d_2, \ldots, d_n)$, the tempo invariant representation utilized is the relative duration sequence $\overline{D} = (\frac{d_2}{d_1}, \frac{d_3}{d_2}, \ldots, \frac{d_n}{d_{n-1}})$. This sequence is quantized to a discrete alphabet. Due to the gross approximations in duration that are committed when singing carelessly, the inter onset interval is used as a more consistent representation of durations.

Matching

The matching step consists in finding good occurrences of the codified query in the database. For this task, the Edit Distance is calculated using the algorithm called Dynamic Programming [10], combining duration and pitch information. In this combination, pitch is considered more important because it is more discriminative than duration. Moreover, duration information is less trustful when singing carelessly.

Once all the database elements were compared with the encoded query, the best occurrences of the pattern are selected according to the Edit Distance (see figure 2).

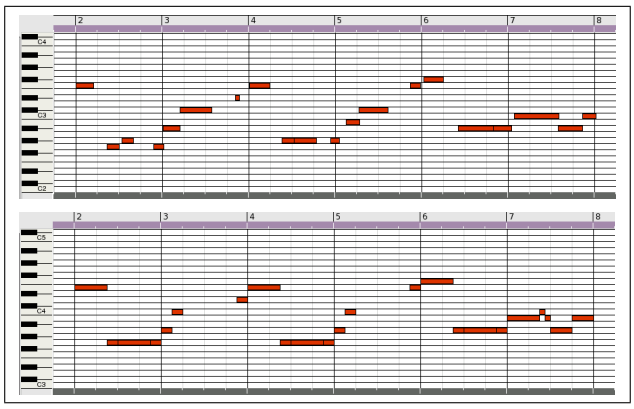


Figure 2: Transcription of the query (top) and an occurrence in the database (bottom).

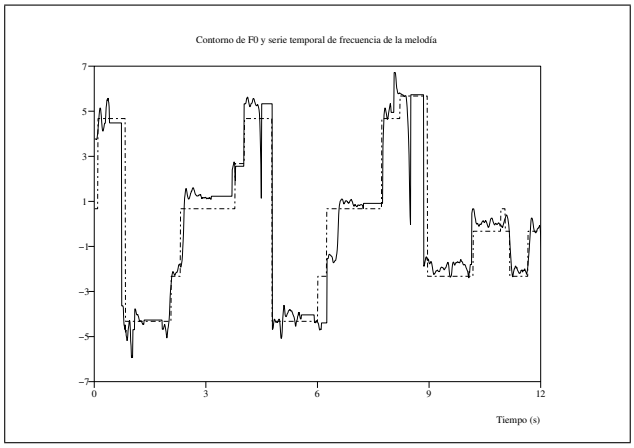


Figure 3: The corresponding pitch time series of the query and the occurrence in figure 2, normalized and aligned by the system.

3.2. Pitch time series matching

As a way of avoiding the automatic transcription errors, a recent approach compares the F0 contour of the query with melodies codified as pitch time series, by means of Local Dynamic Time Warping (LDTW).

However, this approach has some restrictions. Besides it high computational cost, an element of the database must match exactly the query, as it is not possible to search subsequences into sequences providing pitch and tempo invariance [4][5]. For this reason, the database building process is troublesome, because it is necessary to identify from the original melody those fragments likely to be sung.

In the note sequence matching stage, fragments similar to the query are identified in the melodies of the database. Then pitch time series of this fragments are build and are compared to the F0 contour of the query (see figure 3). In this way LDTW is applied to a small group of candidates, without imposing constrains to the query.

4. Evaluation Task

The system was submitted to the MIREX Evaluation eXchange in the Query by Singing/Humming Task 1, known as Known-Item Retrieval. In this task, submitted systems take a sung query as input and return a list of songs from the test database. Mean reciprocal rank (MRR) of the ground truth is calculated over the top 20 returns. The test database consists of 48 ground-truth MIDIs + 2000 Essen Collection MIDI noise files. The query database consists of 2797 sung queries. The sung queries were represented by audio (.wav), pitch vector (.pv) and MIDI (.mid) files transcribing the pitch vectors. Participants could choose whether to use the audio, pitch vector or MIDI files for querying. The system submitted uses the audio queries in .wav format.

4.1. Results

In the table of figure 4 the main results of the MIREX QSBH Task 1 are presented: Mean Reciprocal Rank, machine type [1] and runtime in seconds. Runtime is specified separately for the index and query stage or as a unique value representing both parts. More details of the results are available online [2] .

4.2. Analysis

The submitted system achieved a good performance, placing 4th among 13 evaluated systems. Significance tests show that the results of the first four algorithms are grouped closely.

The data set used in the evaluation has the remarkable characteristic that every query starts from the beginning of the corresponding song. Although this information could be used as prior knowledge, the submitted system does not take it into account (to the best of our knowledge at least one of the three best ranked algorithms do it). The hypothesis of match from the beginning would increase the performance of any system, as it reduces false positives. With regards to the assumption of this hypothesis in a real world system, it can be claimed that as long as each melody can be cut into phrases in advance, every query can be considered to match from the beginning. However, there is no guarantee that an

[1] Machine type A is a Dual AMD Opteron 64 1.6GHz, 4GB RAM, CentOS. Machine type B is an Intel P4 3.0GHz, 3GB RAM, XP.

[2] http://www.music-ir.org/mirex2006/index.php/MIREX2006_Results

Participant	MRR	CPU type	Runtime (s) index	query
Wu, Li (1)	0.926	B	63	2502
Wu, Li (2)	0.900	B	63	2817
Jang, Lee	0.883	B	25637	
López, Rocamora	**0.800**	**A**	**6**	**20604**
Lemström et al.	0.688	A	8302	
Sailer (1)	0.568	A	3	56560
Typke et al. (2)	0.390	A	23442	4629
Sailer (3)	0.348	A	3	4618
Uitdenbogerd (2)	0.288	A	8	140
Sailer (2)	0.283	A	3	608
Ferraro, Hanna	0.218	A	89239	
Uitdenbogerd (1)	0.205	A	8	166
Typke et al. (1)	0.196	A	23442	2034

Figure 4: QBSH Task 1 results.

arbitrary query starts at the beginning of a phrase. Moreover, to do this an automatic melody segmentation system is needed, introducing another source of error. In general, imposing this kind of constrains to the problem limits the scope of the system.

Regarding the runtime, although no particular attention was paid on efficiency, the processing time performed was reasonable. Further work should consider improving it.

Finally, an interesting conclusion that can be drawn from the results of this contest is that the state of the art in the query by humming problem shows that, although being still an open problem, it is feasible to face real world situations. Much effort has to be put on automatically building the database. In this sense, results on the MIREX Audio Melody Extraction Task are promising.

References

[1] E. López and M. Rocamora, "Tararira: Sistema de búsqueda de música por melodía cantada," *Proc. of the X Brazilian Symposium on Computer Music*, pp. 142–153, 2005.

[2] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: Musical information retrieval in an audio database," *Proc. ACM Multimedia*, pp. 231–236, 1995.

[3] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham, "Towards the digital music library: Tune retrieval from acoustic input," *Proc. of the ACM Digital Libraries*, pp. 11–18, 1996.

[4] R. B. Dannenberg and N. Hu, "A comparison of melodic database retrieval techniques using sung queries," *JCDL*, pp. 301–307, 2002.

[5] D. Shasha and Y. Zhu, "Warping indexes with envelope transforms for query by humming," *Proc. of the 2003 ACM SIGMOD Conference on Management of Data*, pp. 181–192, 2003.

[6] A. de Cheveignè and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *JASA*, vol. 111, pp. 1917–1930, 2002.

[7] A. P. Klapuri, "Sound onset detection by applying psichoacoustic knowldege," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1999.

[8] E. Pollastri, *Processing Singing Voice for Music Retrieval*. PhD thesis, Università Degli Studi Di Milano, 2003.

[9] B. Pardo and W. Birmingham, "Encoding timing information for musical query matching," *ISMIR*, pp. 267–268, 2002.

[10] K. Lemström, *String Matching Techinques for Music Retrieval*. PhD thesis, Department of Computer Science, University of Helsinki, 2000.

Two Note Based Approaches to Query by Singing/Humming

Christian Sailer
Fraunhofer IDMT
Langewiesener Str. 22
98693 Ilmenau/Germany
sar@idmt.fraunhofer.de

Abstract

This paper describes the submissions to the MIREX 2006 Query by Singing/Humming task delivered by Fraunhofer IDMT. The approach presented here is based on extracting the pitch out of monophonic singing (or humming), and hereafter segmenting and quantising it into a melody composed of discrete notes. Finally this melody is compared to a database of indexed melodies, using an error tolerant similarity search. Two algorithms have been submitted that differ in the melody extraction method, roughly characterised by the trade-off between accuracy of transcription (and therefore recall) and computing time needed. A third version accepting queries in midi format has also been submitted.

Keywords: MIREX, Query by Humming, Query by Singing, similarity search

1. Introduction

The term Query by Singing/Humming usually describes the retrieval of a musical piece containing a certain melodic theme by singing or humming the same. Several tasks are to be solved to tackle this problem: The corpus of melodies in which the query will be searched has to be acquired, the singing input has to be processed into a format that can be handled by the search algorithm, and melodies similar to the query input have to be spotted in the melody database. The last step requires a high grade of discrimination whilst being tolerant against either input errors or errors propagating from previous processing steps. As this results in many parameters to be defined in a QBSH evaluation, assessing and comparing such systems can be a tedious problem.

By presenting a data corpus in a defined format, and setting up two retrieval tasks, the MIREX 2006 QBSH task presents a frame work allowing comparison of different systems

2. Implementation Overview

The submission consists of an indexing tool (a), and three different query tools (b), (c), and (d). The query tools all use the same melody search algorithm and the same melody database, but feature different mechanisms to acquire the melody information to search for. The database is read from the file created by (a) and stored in the memory. The tools are able to read the query input as wav, aiff, mp3 or MIDI files from a given directory, match them against the database and output a list of the most similar melodies for each query file.

All algorithms are implemented in C++ and available for Windows and Linux. Approximate run times [1] are shown in table 1.

3. Indexing

For indexing, monophonic MIDI files are used. They are read by the indexer (a) and transformed into a database file that can be accessed by the query tools. This comprises just a transformation of the midi files into suitable format. Modification that may happen to the data are the elimination of polyphony, if overlapping notes lead to MIDI files that are slightly polyphonic. The behaviour for massively polyphonic midi files is not defined.

4. Query

The general approach to querying by audio files is extracting the melody from the audio itself. In this case, discrete note representation of the melody are extracted, as can for example be found in MIDI files. Two different algorithms to extract melodies are submitted separately. The third submitted algorithm reads in pre-extracted melodies from monophonic MIDI files.

In a subsequent step, the extracted melodies are compared to the melodies in the indexed database. The look-up

[1] All run times are measured on a 3GHz Intel Pentium IV system.

Table 1. Run times for the different algorithms. N denotes number of indexed songs, l the length of the query

Algorithm	Run Time	Scaling
Indexing	1-2s/1000 songs	$O(N)$
DB Look Up	2s/1000 songs	$O(lN)$
Extraction Warp	about $\frac{1}{5}$ Realtime	$O(l)$
Extraction Ear	about $1.5 - 2$ Realtime	$O(l)$

Table 2. Overview of the results of QBSH task. See text for explanation, entries discussed here in bold.

	AU1	AU2	**ear**	**midi**	**warp**	FH	NM	RJ	RL	RT1	RT2	XW1	XW2
Task 1	0.205	0.288	**0.568**	**0.283**	**0.348**	0.218	0.688	0.883	0.800	0.196	0.390	0.926	0.900
Task 2	0.163	0.238	**0.587**	**0.649**	**0.415**	0.309	0.722	0.926	n.e.	0.468	0.401	n.e.	n.e.

part is exactly the same for all query by singing/humming variants presented here.

4.1. Melody Extraction using a Physiological Ear Model

The first algorithm uses a physiological ear model to achieve a transcription that is as close to human hearing as possible. This yields a very precise melody transcription of monophonic audio, and therefore the best accuracy in querying that is achievable, but due to its high complexity, computing time exceeds real time.

The algorithm used here is based on the implementation of Heinz [1, 2] and has undergone minor modifications for bug fixes and stability improvement.

4.2. Melody Extraction using a Warped FFT

A second, much faster algorithm uses a warped FFT [3] to transform the time signal into a spectrogram with sufficient temporal resolution and sub half-note bin width throughout the spectrum. With an algorithm inspired by PreFest [4, 5], a salient pitch line is extracted from the spectrogram. In a melody segmentation process that has been developed based on the works of Heinz [2], a sequence of temporally discrete note objects is derived from the pitch line in conjunction with further spectral information. These note objects are then quantised to a discrete 12-tone note grid, resulting in a sequence of discrete musical notes.

4.3. Melody Similarity Search

The look-up is carried out as string alignment process [6], which has been adapted for melody search [7] on discrete note representations of the melodies. As basic search alphabet, the relative change of a melody over time, i.e. not notes, but descriptions of note transitions are used, represented by the note intervals and ratio of inter-onset intervals. This makes the search algorithm independent from absolute tempo and pitch.

As further investigations have shown, human individuals tend to render melodies in about the correct tempo, so the absolute note length has been added to the evaluation criteria of the melody search [8].

The alignment is carried out as a semi-local alignment, meaning that the whole query string must match any part of the reference string, and returns a value that increases with the similarity of the query to the reference.

In a post processing step, the contours of the M best matching melodies are compared to the contour of the query, and a correction of the alignment values is carried out. In the current implementation, $M = 50$ is used.

As the resulting alignment values depend on the size of the query string (the longer the string, the greater the maximum possible value), the values have to be normalised to allow an assessment of the alignment quality.

5. Results and Discussion

The data used consists of about 2000 MIDI noise files, 48 ground truth MIDI files and 2797 renditions of these 48 melodies as wave [2], pitch vector and MIDI files. Task 1 uses the noise files and the ground truth files as database and the vocal renditions as queries and measures reciprocal rank of the matching ground truth file. Task 2 uses the noise files, the ground truth files and the renditions as database and the renditions as query and measures the recall rate of versions of the query song among the top 10 results.

An overview [3] of the results of the QBSH tasks can be found in table 2.

As could be expected, the physiological ear model outperformed the warped FFT extraction in both tasks, while both are no match for some of the best algorithms. The warped FFT extraction algorithm is also known for having some problems with distorted files, and may have had some problems with the 8-bit quantisation of the query files.

Surprisingly, the version using midi files performed very weak in task 1, which may be explained by the mediocre quality of the query midis, which where generated automatically from pitch vector files [4]. This assumption is also supported by the increase of performance in task 2 – possibly the query MIDI files were more similar to each other than to the respective ground truth files.

One main problem of the described algorithms is probably the similarity search engine that has only been developed and optimised until 2004. Recent developments on other features and more elaborate search strategies proved to be more successful on this task.

6. Conclusive Remarks

The algorithms solely depending on note-quantised melodies are clearly outperformed by algorithms using multiple stage search algorithms and/or pitch vectors to represent singing queries. This allows at least the assumption that pitch vectors are a better representation for sung inputs than quan-

[2] 8kHz, 8bit, mono

[3] See http://www.music-ir.org/mirex2006/index.php/QBSH:_Query-by-Singing/Humming_Results for full results and information on participants

[4] See QBSH discussion page on http://www.music-ir.org/mirex2006 for details

tised melodies. This may prove to be an important result for further development on QBSH systems.

7. Acknowledgements

We would like to thank the ISMIRSEL team for their effort put into the organisation and the running of this task.

Work on the Query by Humming algorithms used in this approach was partially funded by the EU-Project Semantic Hifi (FP6-507913) [5] .

References

[1] T. Heinz and A. Brückmann, "Using a physiological ear model for automatic melody transcription and sound source recognition," in *Proceedings of the 114th Audio Engeering Society's Convention*, 2003.

[2] T. Heinz, "Ein physiologisch gehörgerechtes verfahren zur automatisierten melodietranskription," Ph.D. dissertation, Technische Universität Ilmenau, 2006.

[3] A. Härmä, M. Karjalaunen, L. Savioja, V. Välimäki, U. K. Lane, and J. Huopainiemi, "Frequency-warped signal processing for audio applications," *Journal of the Audio Engineering Society*, vol. 48, no. 11, pp. 19–22, November 2000.

[4] M. Goto, "A robust predominant-f0 estimation method for real-time detection of melody and bass lines in cd recordings," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Istanbul, 2000, ordner-Nr: A-13.

[5] ——, "A predominant-f0 estimation method for cd recordings: Map estimation using em algorithm for adaptive tone models," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, 2001, pp. 3365–3368, ordner-Nr: A-14.

[6] D. Gusfield, *Algorithms on Strings, Trees and Sequences*. Cambrigde University Press, 1997.

[7] C. Sailer, "Using string alignment in a query-by-humming system for real world applications," Talk at 150th ASA Fall Meeting, October 2005.

[8] M. Dittrich, "Untersuchung und Optimierung eines musikalisch fundierten Suchverfahrens zur Melodieerkennung," Master's thesis, Technische Universität Ilmenau, 2003.

[5] see http://shf.ircam.fr

QBSH System for MIREX

Xiao Wu
HCCL Lab, Institute of Acoustics,
Chinese Academy of Sciences
xwu@hccl.ioa.ac.cn

Ming Li
HCCL Lab, Institute of Acoustics,
Chinese Academy of Sciences
mli@hccl.ioa.ac.cn

Abstract

This paper describes HCCL lab's submission to the Query-by-Singing/Humming(QBSH) task of Music Information Retrieval eXchange(MIREX) 2006. As we do not participate in the second sub-task, this paper will only deal with the Known-Item Retrieval sub-task. In the submitted system, we apply a novel algorithm called Recursive Alignment(RA) to compute the similarity score between query and candidates. We also employ the multilevel filter strategy to reduce the running time. Finally, we give the evaluation results of the presented system.

Keywords: MIREX, QBSH, Music Information Retrieval

1. System Overview

This section gives a overview of the submitted system.Figure1 presents the framework of the submitted system which originates from [1]. The system consists of four stages: (1) feature extraction (2) filters using part of the query (3) filters using the whole query and (4) final rescoring. Inspired by Viola who introduces cascade filters to detect human faces [2], the system employs seven level filters to efficiently eliminate unlike candidates. Basically the former filters are more efficient but less accurate than the latter ones. Top-down fashioned similarity measure algorithms are selected for the final rescorer and most of the filters. We believe such category of algorithms are more robust to local mismatches caused by note-segmentation erorrs, inaccurate singing and grace notes in the reference. The following sections will describe these stages in detail.

2. Database Preprocess

The database are constructed with monophonic midi files. Common used information such as pitch value, note duration and onset time is included in the database. Besides, we also perform note compression, music phrase segmentation and pentanotes clustering while building the database.

2.1. Note Compression

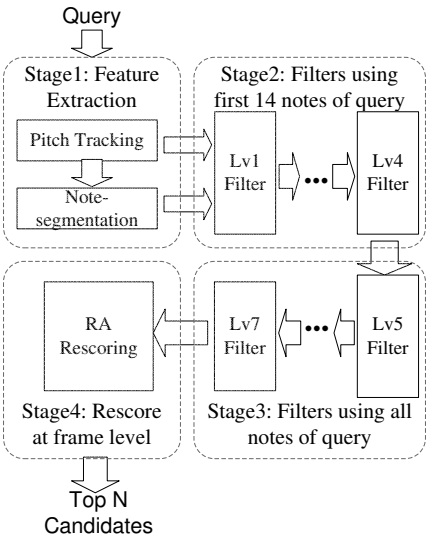


Figure 1. System Framework

As we find in practice that most people shorten the long notes when they sing, we compress the duration d which is longer than the song average duration $\overline{d}$ to

$$\overline{d} + log\left[1 + \alpha(d - \overline{d})\right]/\alpha$$

where α is a predefined constant.

2.2. Music Phrase Segmentation

Our analysis on real world queries shows that more than 98% of them start from the beginning of music phrases. Thus we give each note a weight value to tell how probable this note could be the beginning of certain music phrase. While computing the similarity score, this weight will be considered. The value of the weight is decided by the context such as neighboring rest notes, duration of the previous note and repeating pattern.

2.3. Pentanotes Clustering

We cluster all the neighboring 5 notes into 128 classes using K-Mean algorithm. Meanwhile a forward table and a inverse table are constructed to provide indexing between classes and pentanotes.

3. Features Extraction

Feature extraction includes pitch tracking and note segmentation. The input query is 16bit encoded linear PCM with 8kHz sample rate.

3.1. Pitch Tracking

Pitch value is computed for each window of 25ms where adjacent windows overlap by 15ms. Improved sub-harmonic summation is adopted as the pitch tracking method[3]. Spectral energy is normalized by the average energy around the frequency, which lowers gross errors. Post processing such as median filtering, linear filtering are adopted too.

3.2. Note Segmentation

We segment notes by using an energy-based approach [4]. It is processed as the following procedure. Firstly, voiced sections and unvoiced sections are discriminated apart by adaptive energy threshold. Secondly, notes are segmented by the fluctuation of the harmonic energy and wave energy. Thirdly, notes are split within which pitch fluctuation is beyond a semitone. Finally, notes whose duration is too short are deleted or merged to its adjacent notes.

4. Filtering and Similarity Measure

The system introduces 7-level filters to efficiently eliminate unlike candidates. Each filter keeps very high recall. The former filters are more efficient but less accurate than the latter ones. Candidates which survive cascade filters are passed to rescorer to re-compute the similarity score. A novel algorithm called Recursive Alignment(RA) [1], which outperforms all other competitor algorithms in our experiment for its high precision, is applied in pitch contour for rescoring. Variations of RA which run much faster at the expense of less accuracy are selected by some of the filters.

As is shown in Figure 1, the QBSH system has two filtering stages. In stage 2 only the first few notes of the query (usually the first 14 segmented notes) are used to generate 6000 most probable candidate melody sections. Then the whole query is used to select 500 survivors out of the 6000 candidates in stage 3. The reason we do not use the whole query in stage 2 comes from the consideration of runtime efficiency. Table 1 lists all the filters used in the submitted system.

4.1. Key Detection

Since the query and the candidate are usually from the different keys, we always subtract their own mean pitch during the similarity computation. Furthermore, finer tuning is made in the final rescore stage to determine the best key transposition.

4.2. Pentanote Indexing

Pentanotes indexing is performed before all other filters. Firstly part of the whole query are selected for stage 2. Then all pentanote clusters are compared with the head and tail of the part-query using frame-based RA algorithm. The 25% most similar clusters are kept. With the index table we can map these clusters to pentanotes in the database and construct melody candidates. Usually several million candidates are constructed after this filter.

Table 1. Cascade filters

LV	Feature(s)	Algorithm
Stage2: using part of the query		
1	pitch contour	pentanote indexing
2	pitch contour	RA VarⅢ
3	variance, highest pitch, etc.	linear classifier
4	segmented notes	RA VarⅡ
Stage3: using the whole query		
5	pitch histogram distance	linear classifier
6	segmented notes	RA VarⅡ
7	segmented notes	RA VarⅡ

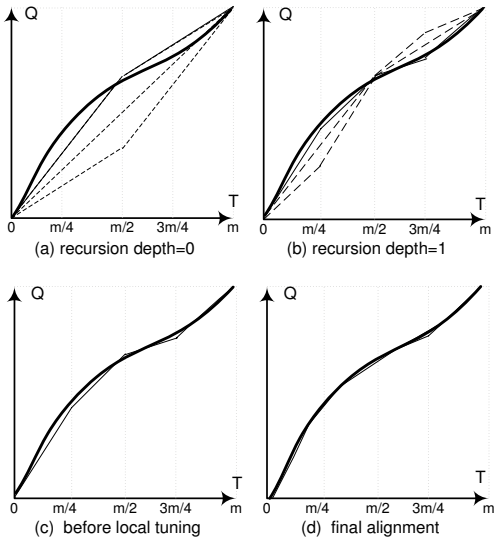


Figure 2. RA Alignment with 1 Recursion and 3 Possible Scale Tries

4.3. Linear Classification with Simple Features

The second and the fifth filters are linear classifiers using simple features such as pitch variance, highest pitch value and pitch histogram etc. Compared with either pitch contour or note sequence, they are one dimension features which need few computation to compute or to classify. And also compared with N-grams, they are global view features which seem to be robust to local errors. Here Manhattan distance is adopted to calculate the similarity between two histograms. The classification thresholds are predefined constants.

4.4. RA and RA variations

Recursive Alignment(RA) [1] inspired by J.Jang's LS [5] is a top-down algorithm to match query and melody candidate at frame level. The basic idea of RA comes from the fact

that the query and the candidate are similar if and only if they roughly share the same shape in the global view. The algorithm divides the candidate melody into 2^N parts recursively and each part uses a linear alignment scale. After that local tuning is applied to get the final alignment path. Figure2 gives an example of RA. The main difference between RA and other frame level alignment algorithms such as DTW is RA's top-down fashion. In RA higher level decision is always made ahead of lower ones, that is, global scale factor which is determined before local ones will restrict the local scales within a reasonable range. We believe such top-down style can handle long-distance information (rhythm and duration for example) better.

RA variations are employed by some filters. RA VarⅡ uses segmented notes instead of frames while computing score which reduces complexity magnitude from frame order to note order. RA Ⅲ divides frequency space into several bands and binarizes the pitch value in each sub-band, so the score of several frames can be computed in parallel utilizing the 32-bit bandwidth of computers.

5. System Implementation

The system is implemented with C++ and is built in Win32 environment with Intel C++ Compiler 9.0. We submit two executable files based on different assumptions. The first one assumes that the queries are always from the beginning of the targets, which fits the case of the evaluation. The second one allows the user start from any position of the target song, which is relatively slower and less accurate but we think it has more practical value.

6. Evaluation Result

The queries are 2797 wave files and the database containing 48 ground truth MIDI files along with 2000 Essen Collection noise MIDI files. Before recognition starts, we convert all queries into 16bit linear encoded format with 8K Hz sample rate.

We achieved the best results among all contestants in the subtask we participated. For the system "match from beginning", the Mean Reciprocal rank (MMR) is 0.926. For the system "match from anywhere", the MMR is 0.900. We think the submitted system mainly benefits from two things: firstly, introducing top-down fashioned RA algorithm which considers long-distances shape of pitch contour while optimizing the alignment; secondly, employing carefully selected filters which greatly reduce the search space while keeping high recall. The system gives a good performance even if there is no assumption of singing from beginning because it is designed for this intention. In the future, we may focus on revising the MIDI database to make the reference more similar to human singing. Perhaps some statistical technologies are needed.

7. Acknowledgments

This work is supported by National Natural Science Foundation of China (10574140, 60535030), Chinese 973 program (2004CB318106). Many thanks to IMIRSEL for their great effort in the organization and evaluation of MIREX2006. We are also grateful to Stephen Downie and Jyh-Shing Jang who lead the QBSH task and prepare the evaluation data.

References

[1] Wu,X., Li,M., Liu,J., Yang,J., Yan,Y., "A top-down approach to melody match in pitch contour for query by humming," in *Proc of International Conference of Chinese Spoken Language Processing*, 2006.

[2] Viola,P., Jones,M., "Robust real-time object detection," in *International Journal of Computer Vision*, 2002.

[3] Li,M., Wen,Y., Yu,T., "High efficient pitch tracking method for tonal feature extraction," in *Proc of International Conference of Chinese Computing*, 2001.

[4] Li,M., Yan,Y., "An humming based approach for music retrieval," in *Proc of National Conference on Man-Machine Speech Communication*, 2005.

[5] Jang,J., Hsu,C. and Lee,H., "Continuous hmm and its enhancement for singing/humming query retrieval," in *Proc of ISMIR*, 2005.

MIREX Symbolic Music Similarity

Pascal Ferraro
LaBRI - Université de Bordeaux 1
351 cours de la Libration
F-33405 Talence cedex, France

pascal.ferraro@labri.fr

Pierre Hanna
SCRIME - LaBRI - Université de Bordeaux 1
351 cours de la Libration
F-33405 Talence cedex, France

pierre.hanna@labri.fr

Abstract

This extended abstract presents a submission to the Music Information Retrieval Evaluation eXchange (MIREX) in the symbolic Melodic Similarity task. This submission is an implementation of an extension of the edit-distance technique to the polyphonic context. Melodies are represented by quotiented sequences. A quotiented sequence is a sequence graph defined with an additional equivalent relation on its vertices and such that the quotient graph is also a sequence graph. The core of the method relies on an adaptation of edit-distance metrics, regularly applied in bioinformatic context. Results are presented and discussed in the monophonic and polyphonic contexts.

Keywords: symbolic melodic similarity, edit-distance, quotiented sequence, polyphony.

1. Similarity Scores

1.1. Problem formalization

To take into account the polyphonic nature of musical sequences, we propose to use a quotiented sequence representation. Formally, a quotiented sequence is a sequence graph with an equivalence relation defined on the set of vertices, and such that the resulting quotient graph is also a sequence. By definition, in a quotiented sequence, quotient graph and support sequence are both sequences. A quotiented sequence can thus be considered as a self-similar structure represented by sequences on two different scales. Note that a quotiented structure can also be viewed as a tree graph.

In this paper we propose to extend the representation of monophonic melodies as sequences of pitches and durations [2]. In the context of polyphonic music, notes that occur at the same time are grouped to form a quotiented sequence. Each vertex of the quotiented sequence is labelled by the pitch and the duration of each note. The pitch is coded according to the difference with the tonic (in semitones), so that the algorithms proposed are transposition invariant. The tonic of the piece considered has to be known beforehand. The duration is coded in sixteenth note values.

The score s depends on the pitch range and on the duration difference between two notes x_1 and x_2:

$$s_Q = s_{\text{pitch}} + \mathbb{K} s_{\text{duration}}$$

where s_{pitch} is the score due to the difference of pitch, and s_{duration} is the score due to the difference of duration. The parameter $\mathbb{K}$ determines the relative weight of the pitch difference with the duration difference. The score s_{duration} is simply the difference of duration (in sixteenth note values) between notes x_1 and x_2. The score s_{pitch} is determined according to consonance: the fifth (7 semitones) and the third major or minor (3 or 4 semitones) are the most consonant interval in Western music. The score associated with the empty symbol is computed according to the score between a note and a rest. This score has been fixed.

This approach leads us to consider any polyphonic sequence as a series of ordered pairs. As for monophonic sequence, each pair is defined by the pitch of the note and its length (coded in sixteenth note values). The pitch of the note is coded as the difference (in semitones) with the tonic. This difference is determined modulo an octave and is thus defined in the range $[0, 11]$ semitones.

1.2. Edit Score

We propose to consider only three operations (substitution, deletion and insertion), that are usually used to compare musical sequences. Ferraro and Godin [1] have recently introduced an edit distance between unordered quotiented trees based on a comparison of support graph and edit operations that preserves equivalence relations. We propose here a symmetric approach by comparing quotiented sequences at the more macroscopic scale. Basically, quotiented sequences refer to sequences whose nodes are also sequences. Edit score related to quotient vertices is then defined as an edit score computation between the support subsequences of these vertices.

The main difference between this recursive relation and the computation of global edit score between sequences lies in the computation of the score of the edit operations between quotient vertices. They are computed as the edit score between the support subsequences corresponding to the quotient vertices. The complexity of this algorithm is $O((|S_1|+$

$|Q_1|) \times (|S_2| + |Q_2|)$, where $|Q_1|$ and $|Q_2|$ represent the respective number of chords in polyphonic musical sequences S_1 and S_2.

1.3. Local Alignment

In many applications, two strings may not be highly similar in their entirety but may contain regions that are highly similar. This is particularly true when long streches of anonymous sequences are compared, since only some internal sections of those strings may be related. In this case, the task is to find and extract a pair of regions, one from each of the two given strings, that exhibit high similarity. This is called *local alignment or local similarity problem* [3] and is defined as : Given two strings S_1 and S_2, find substrings ρ_1 and ρ_2 of S_1 and S_2, respectively, whose similarity is maximum over all pairs of substrings from S_1 and S_2.

The computation of a local similarity allows to detect local conserved areas between both sequences. The solution of such a problem is based on the notion of suffix mapping between sequences.

The local suffix mapping problem for a given pair x_1, x_2 of vertices is to find a (possibly empty) suffix ρ_1 of $S_1[x_1]$ and a (possibly empty) suffix ρ_2 of $S_2[x_2]$ such that the score of the optimal sequence of edit operations transforming ρ_1 into ρ_2 is the maximum over all scores of sequences of edit operations between suffixes of $S_1[x_1]$ and $S_2[x_2]$.

Similarly to the computation of an optimal score between quotiented sequences, the complexity of solving the local suffix mapping problem between quotiented sequences is $O((|S_1|+|Q_1|)\times(|S_2|+|Q_2|)$. All optimal local alignments of two quotiented sequences can be represented in two dynamic programming tables and can be found by tracing any pointers back from any cell with the optimal value.

2. Experimentations and Results

3. Conclusion

References

[1] P. Ferraro and C. Godin. An Edit Distance Between Quotiented Trees. *Algorithmica*, 36:1–39, 2003.

[2] M. Mongeau and D. Sankoff. Comparison of Musical Sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[3] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

Extended abstract for submissions to MIREX 2006: Three algorithms for symbolic similarity computation

Klaus Frieler
Universität Hamburg
Neue Rabenstrasse 13
20354 Hamburg, Germany
kf (at) omniversum.de

Daniel Müllensiefen
Goldsmiths College, University of London
New Cross Road, New Cross
London, SE14 6NW
d.mullensiefen (at) @gold.ac.uk

Keywords: Symbolic Similarity, Similarity of Melodies.

1. Introduction

In this abstract we describe very briefly the three algorithms that we submitted to this year's MIREX competition for symbolic similarity. All three algorithms only work for the comparison of monophonic melodies, and thus only entered the RISM task of the symbolic similarity competition.

2. Constructing hybrid algorithms from the SIMILE toolbox

In the past we have explored different methods for abstracting information within different musical dimensions from melodies and for comparing, i.e. measuring the similarity, two abstract sequences which may represent monophonic melodies. All abstraction and similarity computation methods are implemented in our software toolbox SIMILE. The individual methods are described in greater detail in [7].

2.1. Melodic dimensions and abstraction methods

The abstraction methods we implemented so far for representing data in different musical dimensions of single line melodies are:

- Pitch: MIDI quantisation, leap/step-quantisation, Parsons Code, see [7].
- Rhythm: Categorisation to five duration classes, representation as 'gaussified' values, see [7], [2].
- Contour: Different methods for smoothing coarse directional movements, Fourier transform, see e.g. [10].
- Implied tonality: Categorisation according to harmonic content as based on Krumhansl's tonality vector, e.g. [5].
- Accent structure: Combinations of Gestalt-like accent rules from psychological literature, e.g. [4].

2.2. Similarity algorithms

Data from any abstraction method can be combined with most of the following similarity algorithms that are employed for doing the actual comparision.

- Edit Distance: e.g. [6]
- n-grams: e.g. [1]
- Geometric distance: [10]; [8]
- Correlation coefficient: e.g. [10]

3. The employed algorithms

We tried three different algorithms on the competition items (RISM incipits) to learn a little bit about the behaviour of different approaches with this particular melodic material (short beginning phrases of classical melodies).

3.1. KF1

KF1 was the name for the algorithm we submitted to last year's MIREX competition. This algorithm was optimised on a set of human experts ratings of short melodic phrases mainly from pop tunes. A detailed description can be found in the extended abstract of our last year's submission [3]

3.2. KF2

KF2 stands for an algorithm which proofed to be the best hybrid combination of abstraction methods and algorithms in our study with pop music tunes [7]. There it was termed `opti3`, and it consists of three different individual algorithms:

$$\begin{aligned} KF2 &= 0.505 \cdot \texttt{nGrUkkon} + 0.417 \cdot \texttt{rhythFuzz} \\ &\quad +0.24 \cdot \texttt{harmCorE} - 0.146 \end{aligned}$$

where

- `nGrUkkon:` measure based on the Ukkonen distance of pitch intervals 3-gram.
- `rhythFuzz:` Edit distance of sequences of categorised rhythm values.
- `harmCorE:` Harmonic measure: The main tonality of the melody is calculated according to Krumhansl's algorithm and the two tonalities are compared with the edit distance.

3.3. KF3

KF3 was one of our new and still very experimental approaches to melodic similarity measurement. It is based on the calculation of melodic accents. To this end, for every note of the two melodies, we calculate binary accents weights according to the following rules:

- `phrasend:` Ending of melodic phrase.
- `phrasbeg` Beginning of melodic phrase.
- `beat13:` Note on beat 1 or 3 of bar.
- `shortpr:` Accent of second note of 2-note phrase.
- `longmod:` Duration longer than mode of all durations in phrase.
- `pextrem:` Melodic contour turning point.
- `jumpaft5:` Note after a jump of at least 5 semitones.

For every rule that evaluates to true an accent value of 1 is allocated to that specific note. All accent values are summed for each note. This general approach and the individual rules in particular are described in greater detail in [9].

For the two melodies we receive therefore two resulting number sequences, which are then compared using the Edit Distance algorithm where each number is treated as a separate symbol and costs for deletion, insertions, and substitutions is always 1.

4. Contest results

As last year's MIREX test set hasn't been published and we weren't able to learn from the particular type of melodic phrases, we submitted three very different algorithms. The failure of KF1 in last year's and this year's MIREX shows clearly the detrimental effects of overfitting. KF1 performed very well in predicting human listener judgements regarding the similarity of a specific set of pop music phrases, when it was tested on a separate test set comprising short phrases from the same repertoire. But its internal complexity and its very bad performance on the RISM incipits suggest that it is heavily overfitted to the specific type of data set that is was constructed from.

We are surprised by the still acceptable performance of KF3, which ignores all specific pitch and duration information and instead relies on some quite abstract melodic accents computation. This could mean that accent structure is indeed an important aspect of a melody's identity. In a future optimisation round, this algorithm should be combined with similarity approaches that consider the melody data (pitch and duration) with less abstraction.

KF2 is the algorithm that has proven to be reliable in a variety of situations in the past (e.g. comparing pop music tunes and folk song phrases). We therefore are quite pleased that without any further modification it worked also well on the competition data set and delivered results which are close behind the two algorithms that scored best.

References

[1] Downie, J.S. (1999). Evaluating a simple approach to music information retrieval. *Conceiving melodic n-grams as text*. PhD Thesis, University of Western Ontario.

[2] Frieler, K. (2004). Beat and Meter Extrraction Using gaussified onsets. *ISMIR Proceedings Barcelona, 2004*

[3] Frieler, K., and Müllensiefen, D. (2005). The Simile algorithm for melodic similarity *Abstract MIREX 2005*. http://www.music-ir.org/evaluation/mirex-results/articles/similarity/frieler.pdf

[4] Jones, M.R. (1987). Dynamic pattern structure in music: Recent theory and research. *Perception & Psychophysics*, 41, p. 621-634.

[5] Krumhansl, C. L. (1990). *Cognitive foundations of musical pitch*. New York: Oxford University Press.

[6] Mongeau, M., and Sankoff, D. (1990). "Comparison of musical sequences". *Computers and the Humanities* 24, 161-175.

[7] Müllensiefen, D., and Frieler, K. (2004). "Cognitive Adequacy in the Measurement of Melodic Similarity: Algorithmic vs. Human Judgements". *Computing in Musicology* 13, 147-176.

[8] O'Maidin, D. (1998). A Geometrical Algorithm for Melodic Difference in Melodic Similarity. Melodic Similarity: Concepts, Procedures, and Applications. *Computing in Musicology* 11. Ed. Walter B. Hewlett & Eleanor Selfridge-Field. Cambridge: MIT Press.

[9] Pfleiderer, M., and Müllensiefen, D. (2006). "The perception of accents in pop music melodies. *Proceedings of the 9th International Conference on Music Perception and Cognition, Bologna, 2006*. http://www.escom-icmpc-2006.org/pdfs/513.pdf.

[10] Steinbeck, W. (1982). *Struktur und Ähnlichkeit. Methoden automatisierter Melodieanalyse.* Kassel: Bärenreiter.

[11] Uitdenbogerd, A. L (2002). *Music Information Retrieval Technology*. PhD thesis. RMIT University Melbourne Victoria, Australia

Score Following at Ircam

Arshia Cont
Ircam / UCSD
1 Place Igor Stravinsky
75004 Paris, FRANCE.
cont@ircam.fr

Diemo Schwarz
Ircam, Realtime Applications Team
1 Place Igor Stravinsky
75004 Paris, FRANCE
schwarz@ircam.fr

Abstract

This paper describes the submission to the MIREX'06 (Music Information Retrival Evaluation eXchange) first score following tasks.

1. Overview

Score following is the key to an interaction with a written score/song based on the metaphor of a performer with an accompanist or band. For a historical review of score follower systems we refer the reader to [1, 2].

The Score Following Player submitted accepts monophonic audio and MIDI input from the performer. The audio following modules uses a core algorithm based on Hidden Markov Models (HMM). Figure 1 shows a block diagram overview of the follower algorithm.

The problem of matching a performance with a score can be considered a special case of sequence alignment, which has been extensively addressed in other research areas, notably in speech recognition and in molecular genetics. In both these domains, HMMs have become extremely popular due to their outstanding results. The HMM in Figure 1 can also be viewed as a sequential model of the *score* where the states (score events) can not be directly observed. What is observed by the system is the probabilities assigned to each state of the score model which are used consequently by a decoding algorithm to match the realtime audio to an event in the score. In the following section we describe the methodology for each block in Figure 1.

2. Score model

The music score is modeled as HMMs where each state represents an event in the score. The topology of the HMM is left-to-right, in accordance with the temporal precedence of score events. Each event in the score is modeled as a sequence of states. These states take into account, for each score event, the features related to the attack, the sustain, and the possible silence at the end. Figure 2 shows a sample

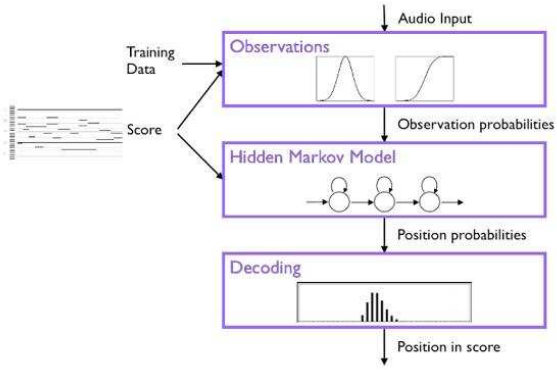


Figure 1. Overview of Score Follower system

note model used in our system. A *silence* model is essentially the same but with *rest* states instead of *sustains* and *attacks*.

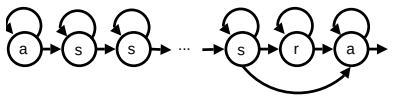


Figure 2. Note model used in Score HMM. `a` stands for *attack*, `s` for *sustain* and `r` for *rest*.

The number of states (n) and transition probabilities (p for forward and $1-p$ for self) at each low level event is determined by solving for n and p in a binomial distribution given the mean (np) as the duration of the event according to the given score and a fixed variance ($np(1-p)$) (50% of the duration here).

Given this note model in terms of HMMs, a score can be represented by accumulating all the note and rest events according to the score and in a sequential manner. Figure 2 shows a sample score and its corresponding score model.

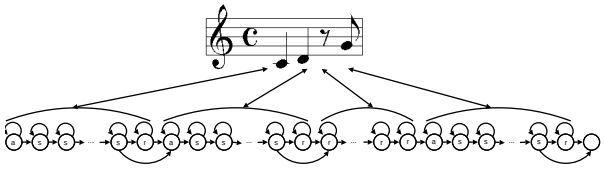


Figure 3. Sample score and corresponding HMM model.

3. Observation Modeling

Observation in the context of our system consists of calculating features from the audio spectrum in real-time and associate the desired probabilities for low-level HMM states. Low-level states in our system are *attack, sustain* and *rest* for each note in the score. Spectrum features are Log of Energy, Spectral Balance and Peak Structure Match (PSM). We will not go into implementation details of the mentioned features which are described in [1, 3, 2]. The observation process can be seen as a dimension reduction process where a frame of our data, or the FFT points, lies in a high dimensional space $\Re^J$ where J is 2048. In this way, we can consider the features as vector valued functions, mapping the high dimensional space into a much lower dimensional space, or more precisely to $2 + N$ dimensions where N is the number of different notes present in the score for the PSM feature. Another way to look at the observation process is to consider it as a probability mapping between the feature values and low-level state probabilities. A diagram of the observation process is demonstrated in Figure 4. In this model, we calculate the low-level feature probabilities associated with each feature which in terms are multiplied to obtain a certain low-level state feature probability. As an example, the Log of Energy feature will give three probabilities Log of Energy for Attack, Log of Energy for Sustain and Log of Energy for Rests. In order to calculate probabilities from features, each of the 8 low-level state feature probabilities is using probability mapping functions from a database of stored trained parameters. They are derived from Gaussians in forms of cumulative distribution functions (CDFs), inverse cumulative distribution functions or PDFs depending on the heuristics associated with each feature state. Note that the dimension of each model used is one at this time. By this modeling we have assumed that the low-level states' attributes are global which is not totally true and would probably fail in extreme cases. However, due to a probabilistic approach, training the parameters over these cases would solve the problem in most cases we have encountered. Another assumption made is the conditional independence among the features, responsible for the final multiplication of the feature as in Figure 4.

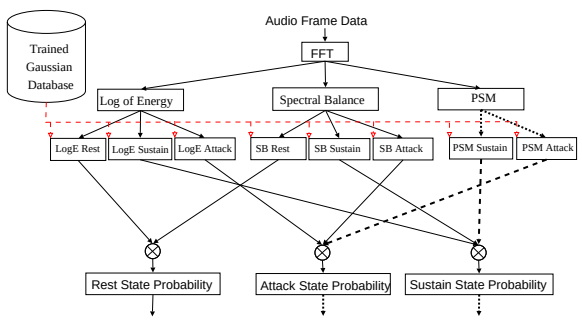


Figure 4. Probability Observation Diagram

4. Decoding and Alignment

Once the observation probabilities are calculated for states in the score model, they are used by a decoding scheme to decide what is the appropriate current high-level state using present and past information. The Bayesian framework in this submission considers observation probabilities as $P(y_t|x_t^k)$ where y_t is a realtime audio observation at time t and x_t^k would be the (hidden) state k in the score. Using this scheme, the current *belief* of the system is computed as in Equation 1 where Z is a normalizing constant, $P(x_t^k|x_{t-1}^{k^*})$ is the transition prior from the score model and $P(x_{t-1}^k|y_{1:t-1})$ is the previous belief of the system. This way, the current high-level state can be decoded by Equation 2.

$$P(x_t^k|y_{1:t}) = \frac{1}{Z} P(y_t|x_t^k) P(x_t^k|x_{t-1}^{k^*}) P(x_{t-1}^k|y_{1:t-1}) \quad (1)$$

$$k^* = \underset{k}{\operatorname{argmax}}\, P(x_t^k|y_{1:t}) \quad (2)$$

5. Conclusion

In this paper, we briefly described our submission to MIREX'06 Score Following task. In a real-world application, our observation model accepts piece-specific and instrument-specific trained data as parameters of each CDF described above. For this submission, since training was not considered for the contest, we use default parameters. Extensions to the system described in this paper as well as more material can be found in [4] and references herein.

References

[1] Arshia Cont, "Improvement of observation modeling for score following," Dea atiam, University of Paris 6, IRCAM, Paris, 2004.

[2] Nicola Orio, Serge Lemouton, Diemo Schwarz, and Norbert Schnell, "Score Following: State of the Art and New Developments," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, Montreal, Canada, May 2003.

[3] Nicola Orio and Diemo Schwarz, "Alignment of Monophonic and Polypophonic Music to a Score," in *Proceedings of the ICMC*, Havana, Cuba, 2001.

[4] Arshia Cont, "Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical hmms," in *IEEE ICASSP*. May 2006, Toulouse.