Fast implementations for perceptual tempo extraction

Matthew Davies
Centre for Digital Music
Queen Mary, University of London
Mile End Road, United Kingdom
matthew.davies@elec.qmul.ac.uk

Paul Brossier
Centre for Digital Music
Queen Mary, University of London
Mile End Road, United Kingdom
paul.brossier@elec.qmul.ac.uk

ABSTRACT

We provide an overview of two algorithms submitted for the Audio Tempo Extraction contest within MIREX 2005: A non-causal Matlab implementation (Davies submission) [4] and a real-time C implementation (Brossier submission) [3]. Both algorithms extract a primary and secondary tempo, associated phase values for each tempo and the relative salience of the slowest tempo. Results indicate that in comparison to other algorithms in the evaluation our approaches placed 9th and 12th out of 13 submissions, and were the two fastest implementations.

1 Non-causal Implementation

1.1 Overview

- An onset detection function (DF) is generated from the audio
- The autocorrelation function (ACF) of the DF is taken
- The ACF is passed through a shift-invariant comb filterbank to extract the primary tempo
- Given this tempo value, the time signature is estimated
- A secondary tempo is derived from the primary tempo and time-signature, and tempo saliences are calculated
- The phase of each tempo is found by cross-correlating the DF with a weighted impulse train

A graphical overview showing the extraction of a primary beat period from an audio input signal is given in Fig. 1.

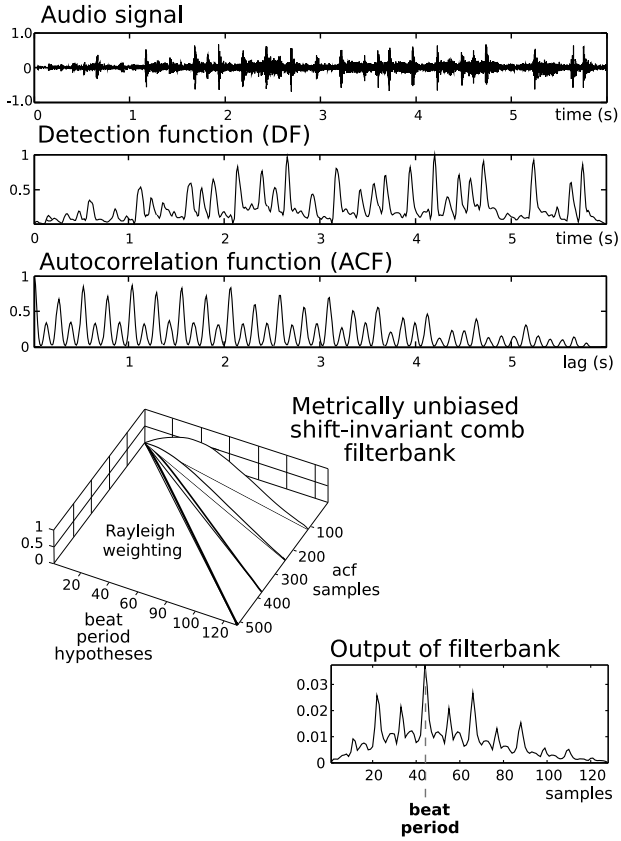

Figure 1: Overview of tempo extraction algorithm. The a detection function (DF) is generated from the audio signal. We take the autocorrelation function (ACF) of the DF, and pass through a shift-invariant comb filterbank to identify the beat period. This is then converted to a tempo estimate.

1.2 Pre-Processing

The first stage in the algorithm is to transform the input audio signal into a suitable representation on which to perform the tempo analysis. We choose the complex spectral difference onset detection function (DF) [2]. The DF is generated by measuring spectral difference (in the complex domain) between target and observed STFT frames, exploiting phase inconsistencies to emphasise tonal onsets and energy changes for percussive onsets. The DF is calculated using a window length of 1024 samples with a 512 sample hop size, giving a temporal resolution of 11.6ms. Before undertaking any tempo analysis, the DF is further processed. It is low pass filtered, after which an adaptive median threshold is generated (16 sample window, 15 sample overlap), which is subtracted from the smoothed DF to give the final input to the system.

1.3 Tempo extraction

The unbiased autocorrelation function (ACF) of this modified DF is then found, from which only the first 512 samples (equivalent to 6 seconds of audio) are retained. The ACF is then passed through a shift-invariant comb filterbank to extract the primary beat period (from which the tempo can be inferred). The filterbank (implemented in matrix form) is comprised of 4 comb elements, where each column represents a different beat period hypothesis (increasing in periodicity with a lower tempo bound of 40bpm) and each row of the matrix is weighted by a tempo preference curve (derived from the Rayleigh distribution function) to emphasise beat periods equivalent to tempi in the range of 75 to 150 bpm. An estimate of primary beat period, τ, is found as the index of the maximum of the output of the filterbank (see Figure 1). This value is then refined by averaging the indices of local maxima at integer multiples of the beat period within the ACF and converted to tempo (in beats per minute) using the following equation:

$$\text{tempo} = \frac{60}{(\tau * 0.0116)} \tag{1}$$

where 0.0116 represents the resolution of the detection function.

1.4 Finding a secondary tempo

Given this primary tempo, we then seek to find another tempo hypothesis consistent with the metrical structure of the input as a likely candidate at which human subjects would tap in time. We first attempt to classify the time signature, as either duple or triple, by evaluating the sum of energy within the ACF at multiples of 2 and 3 of the beat period. The secondary tempo is then derived using a simple rule based approach, and is a function of the time signature and a fixed tempo threshold of 120bpm, taken as an approximate mean tempo value irrespective of musical genre [5].

- If primary tempo $>$ 120 bpm, and time-sig is duple, then secondary tempo is primary * 1/2
- If primary tempo $<$ 120 bpm, and time-sig is duple, then secondary tempo is primary * 2
- If time-sig is triple, then secondary tempo is primary * 1/3

If the initial tempo is quicker than 120 bpm, we assume it is most likely that a secondary tapping rate would be half this initial rate (e.g. a pairing of 130 bpm and 65 bpm seems intuitively more likely than 130 bpm and 260 bpm to describe human tapping behaviour). Conversely, if the primary tempo is slower than 120bpm, then we should double the tempo hypothesis. These two rules assume that the time signature is duple, if we find it to be triple, then we attempt to find the bar length tempo as the second metrical level (i.e. in triple cases, we always try to find a slower tempo).

1.5 Tempo Salience

We choose to measure the salience of a tempo hypothesis as a function of the distance of the tempo to the set threshold of 120 bpm. The salience is found as:

$$\text{salience} = 1 - |\text{tempo} - 120|/120 \tag{2}$$

The salience values for the primary and secondary tempi are normalised such that their sum is equal to 1. We retain the salience of the slower tempo, to be used in the evaluation of our algorithm.

1.6 Beat Phase

The phase of each tempo hypothesis is found by cross-correlating the first 512 samples of the DF with an impulse train (impulses at beat period intervals) and finding the index of the maximum output. In each case the search is only as far as 1 beat period from the start of the file. To give most emphasis to the start of the file, the impulses are linearly weighted, with the first being the strongest and the last being the weakest.

2 Real-time Implementation

The real-time implementation proposes a fast and causal version of the Matlab algorithm. In contrast to the non-causal model which finds a single phase for each tempo hypothesis, the real-time algorithm identifies all beat locations by repeating the processes for beat period induction and phase alignment on an overlapping frame basis across the length of the file. At the beginning of a file or during a tempo change, the prediction mechanism may require several seconds to settle on a fixed value, depending on the salience of the tempo and the nature of the music signal.

For the purpose of the contest, two lists of tempo candidates are derived from the beat locations, one containing the fast candidates, the other ones at slower tempi. The selection of a secondary tempo is entirely based on the primary candidate, using halving and doubling around a split tempo of 95 bpm, regardless of the signature and ignoring triple cases – the time signature was not readily available in the real-time implementation at the time of the submission. From each detected beat location, a tempo candidate is computed, and its fast and slow versions are stored in each list. The final two tempi T1 and T2 are derived from the most frequent occurrences in both candidate lists. Given the causal nature of the algorithm, all beat locations are predicted solely from past evidence. To estimate the phase of the first beats within each excerpt the phase values are mapped back to the beginning of the file from the beat locations found after a given delay and when a fixed number of identical tempo have been detected consecutively. The salience of each candidate is derived from the likelihood found in each tempo list.

The real time implementation of the beat tracking algorithm is available as a shared library of functions, *aubio*, licensed under the GNU General Public License, at http://aubio.piem.org.

3 Results

3.1 Training Data

A small annotated database of 20 files (each 30 seconds in length) was provided to each participant upon which their tempo extraction algorithms could be trained. We tested our algorithms on a 2.8GHz Intel PC running Debian GNU/Linux 3.1. The Matlab (version 7) algorithm completed execution of the 20 training files in 76 seconds - averaging 3.8 seconds per 30 second input file. The real-time C implementation (compiled using GCC) completed execution in 16 seconds - averaging 0.8 seconds per 30 second input file.

3.2 Test Data

The accuracy measure used to evaluate the algorithms was calculated as a combination of the success of the algorithm in the following tasks:

- TT1: The primary tempo within 8% of the ground truth annotation
- TT2: The secondary tempo within 8% of the ground truth annotation
- TT1I, TT2I: An integer multiple of each tempo within 8%
- ST1: The relative salience of the slower tempo, compared to the ground truth, ST1GT
- TP1: The phase of T1 within 15%
- TP2: The phase of T2 with 15%

which were combined to give a P-score for each file:

$$\begin{aligned} P = {} & 0.25 * TT1 + 0.25 * TT2 \\ & + 0.10 * TT1I + 0.10 * TT2I \\ & + 0.20 * (1.0 - |ST1 - ST1GT| \\ & / \max(ST1, ST1GT)) \\ & + 0.05 * TP1 + 0.05 * TP2 \end{aligned}$$

The test database contained 140 excerpts, from which the overall P-score was defined as the mean of all individual values. The time taken to process the database was also recorded to indicate the differences in speeds of the evaluated algorithms.
An overview of results[1] for our algorithms compared to the winning algorithm, (that of Miguel Alonso [1]) are given in Table 1, and include the P-score, Runtime and the percentage of files where at least one tempo was correctly identified. Of the 13 submissions to the evaluation, our algorithms placed 9th (Davies) and 12th (Brossier), and were the two fastest implementations.

[1] A more detailed breakdown of results can be found at the following url: http://www.music-ir.org/evaluation/mirex-results/audio-tempo/index.html

Algorithm	P-score	One Tempo Correct (%)	Runtime (s)
Davies	0.628	95.00	1005
Brossier	0.583	86.43	180
Alonso	0.689	80.71	2875

Table 1: Results table showing P-score, the percentage of files for which at least one tempo was correctly identified and the Runtime, for the Davies, Brossier and Alonso algorithms

ACKNOWLEDGEMENTS

Special thanks to the entire MIREX team for overseeing the evaluation, and to Martin McKinney for proposing the contest.
This research has been partially funded by the EU-FP6-IST-507142 project SIMAC (Semantic Interaction with Music Audio Contents). More information can be found at the project website http://www.semanticaudio.org.

References

[1] M. Alonso, B. David, and G. Richard. Tempo and beat estimation of musical signals. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, pages 158–163, Barcelona, Spain, October, 2004.

[2] 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*, 11(6):553–556, July 2004.

[3] M. E. P. Davies, P. M. Brossier, and M. D. Plumbley. Beat tracking towards automatic musical accompaniment. In *Proceedings of the 118th Convention of the AES*, Barcelona, Spain, May 28–31, 2005.

[4] M. E. P. Davies and M. D. Plumbley. Beat tracking with a two state model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2005)*, volume 3, pages 241–244, Philadelphia, USA, March 18–23, 2005.

[5] L. van Noorden and D. Moelants. Resonance in the perception of musical pulse. *Journal of New Music Research*, 28(1):43–66, March 1999.