Journal of New Music Research
2007, Vol. 36, No. 1, pp. 1 – 16

Routledge
Taylor & Francis Group

Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms

M. F. McKinney[1], D. Moelants[2], M. E. P. Davies[3] and A. Klapuri[4]

[1]Philips Research Laboratories, Eindhoven, The Netherlands; [2]Ghent University, Belgium; [3]Queen Mary University of London, UK; [4]Tampere University of Technology, Finland

Abstract

This is an extended analysis of eight different algorithms for musical tempo extraction and beat tracking. The algorithms participated in the 2006 Music Information Retrieval Evaluation eXchange (MIREX), where they were evaluated using a set of 140 musical excerpts, each with beats annotated by 40 different listeners. Performance metrics were constructed to measure the algorithms' abilities to predict the most perceptually salient musical beats and tempi of the excerpts. Detailed results of the evaluation are presented here and algorithm performance is evaluated as a function of musical genre, the presence of percussion, musical meter and the most salient perceptual tempo of each excerpt.

1. Introduction

Beat tracking and tempo extraction are related tasks, each with its own specificity and applications. Tempo extraction aims at determining the global speed or tempo of a piece of music, while beat tracking attempts to locate each individual beat. The tempo can be extracted without the knowledge of every single beat, thus tempo extraction could be considered an easier task. On the other hand, the result of tempo extraction is a single (or small number of related) value(s), which makes it vulnerable to error. Another difference between the two tasks is how they handle fluctuating tempi: the primary challenge of many beat-tracking systems is following the changing tempo of a piece of music, while for tempo extractors, it does not make much sense to notate a changing tempo with a single value. For music with a constant tempo, beat trackers do not provide us with much extra information than tempo extractors, except for the phase of the beat. Due to these differences, both tasks lead to different applications. Tempo extraction is useful for classifying and selecting music based on its overall speed, while beat tracking allows one to synchronize music to external elements, e.g. gestural control or live accompaniment.

Despite the differences between beat tracking and tempo extraction, both problems have been historically connected. The first attempt to do some kind of automatic pulse detection can be found in the 1970s. In a study of meter in Bach's fugues, Longuet-Higgins and Steedman (1971) derived meter and tempo from a symbolic (score-based) representation of the notes. Later, this led to rule-based systems that built up an estimate of the beat based on the succession of longer and shorter rhythmic intervals (Longuet-Higgins & Lee, 1982, 1984; Lee, 1985). These systems tried to model the process of building up a beat based on the start of a rhythmic sequence. Povel and Essens (1985) also started from purely symbolic rhythmic patterns (not taking into account aspects like dynamic accents or preferred tempo) and analysed them as a whole, searching for the metric structure that fit best with the foreground rhythm. Similarly, Parncutt (1994) analysed short repeating rhythmic patterns, however, he incorporated knowledge about phenomenological accent and preferred tempo to

Correspondence: M. F. McKinney, Digital Signal Processing, Philips Research Laboratories, Eindhoven, The Netherlands.
E-mail: martin.mckinney@philips.com

DOI: 10.1080/09298210701653252

make an estimation of tempo and meter. Miller et al. (1992) proposed a different approach, not starting from a set of rules, but from the response of a bank of oscillators to the incoming signal. The basic idea here was that oscillators start resonating with the incoming rhythm, so after a while the oscillator corresponding to the dominant periodicities should get largest amplitude. Introducing sensitivity related to human tempo preferences and coupling oscillators with related periodicities led to a more accurate detection of tempo and metric structure, while the resonance characteristics of the oscillators enabled them to deal with small tempo fluctuations (Large & Kolen, 1994; McAuley, 1995; Gasser et al., 1999).

All these approaches start from a theoretical viewpoint, rooted in music psychology. In music performance there was a need to find ways to coordinate the timing of human and machine performers. This led to systems of score following, where a symbolic representation of music was matched with the incoming signal (Dannenberg, 1984; Baird et al., 1993; Vantomme, 1995; Vercoe, 1997). Toiviainen (1998) developed a MIDI-based system for flexible live-accompaniment, in which he started from an oscillator-based model related to the Large and Kolen (1994) model. Toiviainen (1998), as well as Dixon and Cambouropoulos (2000), used MIDI, which allowed them to use the advantages of symbolic input to follow tempo fluctuations and locate the beats. However, if one wants to apply tempo detection or beat tracking on music databases or in an analogue performance, techniques have to be developed to extract relevant information from the audio signal. Goto and Muraoka (1994, 1998) solved this problem by focusing on music with very well determined structural characteristics. Searching for fixed successions of bass and snare drums in a certain tempo range, they obtained good results for a corpus of popular music. However it is hard to generalize this method to other musical styles. The first techniques to create a more general approach to beat tracking and tempo detection came from Scheirer (1998), who calculated multi-band temporal envelopes from the audio signal and used them as input to banks of resonators, and from Dixon (1999, 2000), who used onset detection as the first stage followed by a traditional symbol based system. Since then new signal processing techniques have been developed, most of which will be illustrated in this issue.

In the next section, summaries of several state-of-the-art beat tracking and tempo extraction systems are presented. These algorithms participated in the 2006 Music Information Retrieval Evaluation eXchange (MIREX 2006c), an international contest, in which systems dealing with different aspects of Music Information Retrieval are evaluated. Two of the proposed contests, tempo extraction and beat tracking, are summarized here. Further details of four of the participating algorithms can be found in separate articles in the current issue while two others are described in more detail in appendices to this article. Details about the ground-truth data and the evaluation procedure will be given in Section 3 and evaluation results are provided in Section 4.

2. Algorithm descriptions

In general, the algorithms described here consist of two stages: a first stage that generates a *driving* function from direct processing of the audio signal; and a second stage that detects *periodicities* in this driving function to arrive at estimates of tempo and/or beat times. While it is perhaps a crude oversimplification to describe the algorithms in terms of such a two-step process, it facilitates a method for meaningful comparison across many different algorithm structures. Thus, at the end of this algorithm overview, we conclude with a general algorithm classification scheme based on these two stages.

Most of the algorithms presented here were designed for both beat tracking and tempo extraction and are evaluated for both of these tasks. One algorithm (see Section 2.5) was designed mainly (and evaluated only) for beat tracking. Two algorithms (see Sections 2.1 and 2.2) were designed and evaluated only for tempo extraction.

Most of the algorithms are described in detail in other publications (four in this same issue), so we limit our description here to the essential aspects.

2.1 Algorithm summary: Alonso, David & Richard

The algorithm from Alonso et al. (2006) was designed for tempo extraction only and comes in two variants, the second with an improved onset detection method. If we apply the two-stage descriptive schema outlined above, the *driving function* here is a pulse train representing event onsets, detected by thresholding the spectral energy flux of the signal. In the second variant of this algorithm, onset detection is improved by using spectral-temporal reassignment to improve the temporal and spectral resolution in the initial stages. The *periodicity detector* here is a two-stage process, where candidate periodicities are first calculated using three methods, autocorrelation, spectral sum, and spectral product. Dynamic programming is then employed to calculate the optimal path (over time) through the derived periodicities.

Parameters of the driving function derivation include: audio downsampled to 22 kHz, spectral processing in eight bands, a processing frame of $\sim$34 ms with a hop size of 5 ms, resulting in a driving function with a 5-ms temporal resolution.

Further details on this algorithm can be found in a separate article in this issue (Alonso et al., 2007).

2.2 Algorithm summary: Antonopoulos, Pikrakis & Theodoridis

Antonopoulos et al. (2006) developed an algorithm for tempo extraction that derives a driving function from an audio self-similarity measurement. The self-similarity metric is calculated from audio features similar to Mel-Frequency Cepstral Coefficients (MFCC) but with a modified frequency basis. Periodicity in this driving signal is detected through the analysis of 1st-order intervals between local minima, which are plotted in histograms as a function of interval size. These intervals are assumed to correspond to the beat period in the music and thus the largest peaks in the histograms are taken as the most salient beat periods.

Parameters of the driving signal include: 42 frequency bands between 110 Hz and 12.6 kHz, 93-ms temporal windows with a 6-ms hop size, resulting in a driving signal with 6-ms temporal resolution.

Further details of this algorithm can be found in a separate article in this issue (Antonopoulos et al., 2007).

2.3 Algorithm summary: Brossier

Brossier (2006b) developed an algorithm for beat tracking and tempo extraction for the 2006 MIREX. The driving function for his beat tracker is a pulse train representing event onsets, derived from a spectral difference function through adaptive thresholding. The phase and magnitude of periodicities in the onsets were extracted using an autocorrelation function, which in turn were used to calculate beat times. Tempo was then calculated from the most prominent beat periods.

Parameters of Brossier's driving function derivation include: 44.1 kHz sampling rate, linear frequency analysis across the complete spectrum, a 1024 sample analysis frame with a hop size of 512 samples, yielding a 5.6 ms temporal resolution.

Further details of this algorithm can be found in Brossier's PhD thesis (Brossier 2006a).

2.4 Algorithm summary: Davies & Plumbley

Davies and Plumbley (2007) submitted algorithms for the tempo and beat tracking evaluations. Three separate driving functions (spectral difference, phase deviation and complex domain onset detection functions) are used as the basis for estimating the tempo and extracting the beat locations. The autocorrelation function of each driving function is passed through a perceptually weighted shift-invariant comb filterbank, from which the eventual tempo candidates are selected as the pair of peaks which are strongest in the filterbank output function and whose periodicities are most closely related by a factor of two.

The beat locations are then found by cross-correlating a tempo-dependent impulse train with each driving function. The overall beat sequence is taken as the one which most strongly correlates with its respective driving function.

Parameters of the driving functions include: 23.2 ms analysis frames with an 11.6-ms frame hop for audio sampled at 44.1 kHz, yielding driving functions with 11.6-ms temporal resolution.

Further details of the algorithms can be found in Appendix A of this article and in Davies and Plumbley (2007).

2.5 Algorithm summary: Dixon

Dixon (2006) submitted his BeatRoot algorithm to the MIREX 2006 beat tracking evaluation. The driving function of BeatRoot is a pulse train representing event onsets derived from a spectral flux difference function. Periodicities in the driving function are extracted through an all-order inter-onset interval (IOI) analysis and are then used as input to a multiple agent system to determine optimal sequences of beat times.

Parameters of the BeatRoot driving function derivation include: linear frequency analysis covering the entire spectrum, a 46-ms analysis frame with a 10-ms frame hop, yielding a driving function with 10-ms temporal resolution.

Further details of this algorithm can be found in another article in this issue (Dixon 2007).

2.6 Algorithm summary: Ellis

Ellis (2006) developed an algorithm for both the beat tracking and the tempo extraction evaluations. The driving function in his algorithm is a real-valued temporal "onset" envelope obtained by summing a half-wave rectified auditory-model spectral flux signal. The periodicity detector is an autocorrelation function scaled by a window intended to enhance periodicities that are naturally preferred by listeners. After candidate tempi are identified, beat tracking is performed on a smoothed version of the driving function using dynamic programming to find the globally optimal set of beat times. The beat-tracking algorithm uses backtrace and is thus intrinsically non-real-time and it relies on a single global tempo, making it unable to track large ($>10\%$) tempo drifts.

Parameters of the driving function derivation include: 40-band Mel-frequency spectral analysis up to 8 kHz, a 32-ms analysis window with a 4-ms hop size, yielding a driving function with a 4-ms time resolution.

Further details of this algorithm can be found in a separate article in this issue (Ellis 2007).

2.7 Algorithm summary: Klapuri

The beat tracking algorithm submitted by Klapuri to the 2006 MIREX is identical to that described in Klapuri et al. (2006). The algorithm was originally implemented in 2003 and later converted to C++ by Jouni Paulus in 2004. The method and its parameter values have been untouched since then.

The method analyses musical meter jointly at three time scales: at the temporally atomic *tatum* pulse level, at the beat (aka *tactus*) level, and at the musical *measure* level. Only the tactus pulse estimate was used in the MIREX task. The time-frequency analysis part calculates a driving function at four different frequency ranges. This is followed by a bank of comb filter resonators for periodicity analysis, and a probabilistic model that represents primitive musical knowledge and uses the low-level observations to perform joint estimation of the tatum, tactus, and measure pulses.

Both causal and non-causal versions of the method were described in Klapuri et al. (2006). In MIREX, the causal version of the algorithm was employed. The 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 near the beginning of an input signal, but on the other hand, the causal version is more appropriate for on-line analysis. Further details of this algorithm can be found in Appendix B.

2.8 Algorithm summary overview

Table 1 shows a summary of all algorithms entered in the beat-tracking and tempo-extraction evaluations.

3. Evaluation method

For the beat-tracking task, the general aim of the algorithms was to identify beat locations throughout a musical excerpt. To test the algorithms we used a set of 160 excerpts from which we collected beat annotations using a pool of listeners. We tested the algorithms by comparing their estimated beat locations to the annotated beat locations from every excerpt and listener to arrive at an overall measure of accuracy.

The aim of the tempo extraction task was to identify the two most perceptually salient tempi in a musical excerpt and to rate their relative salience. The same annotations used for the beat-tracking evaluation were used to calculate the perceptual tempi of the excerpts.

The beat-tracking and tempo-extraction evaluations were carried out by the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) at the Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign. The evaluations were part of the 2006 MIREX, which included a number of other music information retrieval evaluations as well (MIREX, 2006c).

Details on the excerpts, annotations and evaluation method are given in the following sections.

3.1 Evaluation data

The ground truth data used in both the tempo-extraction and beat-tracking evaluations was collected by asking a number of listeners to tap to the perceived beats of musical excerpts, each 30 s long. In total, we used data for 160 excerpts[1], each tapped to by 40 annotators. The collection of excerpts was selected to give a representative overview of music with a relatively stable tempo. It contains a broad range of tempi (including music especially collected for representing extreme tempi), a wide range of western and non-western genres, both classical and popular, with diverse textures and instrumentation, with and without percussion and with about 8% non-binary meters. Due to this variety the set should be fit to test the flexibility of the automatic detection systems, both in terms of input material and of performance over the whole tempo range.

The tapping data were collected by asking annotators to tap along to the musical excerpts using the space bar of a computer keyboard. Data was collected over two sessions using 80 annotators in total, with approximately equal groups of musicians and non-musicians as well as of male and female participants. The output of this large set of annotators, with varying backgrounds, gives us a representative view of the perceptual tempo (McKinney & Moelants, 2006) of each excerpt. Distributions of these tapped tempi for individual excerpts often show two or even three modes, indicating that different annotators perceived the most salient musical beat at different metrical levels. In the evaluations that follow, we take into account all tapped data for a given excerpt and treat them collectively as the global perception of beat times and their respective tempi. For the beat-tracking evaluation, we use all individual tapping records in the evaluation metric, while for the tempo-extraction evaluation, we summarize the perceptual tempo by taking the two modes in the tempo distribution with the largest number of annotators. The idea is that these two modes represent the two most perceptually-relevant tempi while the relative number of annotators at each

[1]The original collection (cf. McKinney & Moelants, 2006) contained 170 excerpts, but 10 of them were left out due to irregularities in the beat structure (mainly having a fluctuating tempo), which made them inappropriate for the tempo extraction task.

Table 1. Algorithm summary. Algorithm: ALO - Alonso, Richard and David; ANT - Antonopoulos, Pikrakis & Theodoridis; BRO - Brossier; DAV - Davies & Plumbley; DIX – Dixon; ELL – Ellis; KLA – Klapuri. Application: BT – Beat Tracking; TE – Tempo Extraction. Driving Function Type: ON – Detected Onsets; SF – Spectral Flux; SR – Spectral Reassignment; SSF – Self-Similarity Function; PD – Phase Difference; CSF – Complex Spectral Flux; TED – Temporal Envelope Difference. Periodicity Detection: ACF – Autocorrelation Function; SSP – Spectral Sum and Product; DP – Dynamic Programming; PW – Perceptual Weighting; IMI – Inter-Minima Interval; CFB – Comb Filter Bank; IOI – Inter-Onset Interval; MA – Multiple Agent System; HMM – Hidden Markov Model. Implementation Language:* The C/C++ code for the ANT algorithm was generated directly using the MATLAB compiler and thus does not provide the typical complexity advantage gained from manually optimizing the C/C++ code.

Algorithm		ALO1	ALO2	ANT	BRO	DAV	DIX	ELL	KLA
Application		TE	TE	TE	BT & TE	BT & TE	BT	BT & TE	BT & TE
Driving Function	Type	SF ON	SR, SF ON	SSF	SF ON	SF, PD CSF	SF ON	SF	TED
	Time Resolution	5 msec	5 msec	6 msec	5.6 msec	11.6 msec	10 msec	4 msec	5.8 msec
	Number of Channels	8	8	1	1	1	1	1	4
Periodicity Detection		ACF, SSP DP, PW	ACF, SSP DP, PW	IMI	ACF	ACF CFB, PW	IOI MA	ACF DP, PW	CFB HMM, PW
Implementation Language		MATLAB	MATLAB	C/C++*	C/C++ Python	MATLAB	Java	MATLAB	C/C++

mode represents the relative salience of the two tempi. More details about the stimuli, annotators and procedure can be found in McKinney and Moelants (2006).

3.2 Beat-tracking evaluation

The output of each algorithm (per excerpt) was a list of beat locations notated as times from the beginning of the excerpt. These estimated beat times were compared against the annotated times from listeners. In order to maintain consistency with the tempo evaluation method (see Section 3.3) we treat each excerpt annotation as a perceptually relevant beat track: we tested each algorithm output against each of the 40 individual annotated beat tracks for each excerpt.

To evaluate a single algorithm, an averaged ''P'' score was calculated that summarizes the algorithm's overall ability to predict the annotated beat times. For each excerpt, 40 impulse trains were created to represent the 40 annotated ''ground-truth'' beat tracks, using a 100 Hz sampling rate. An impulse train was also generated for each excerpt from the algorithm-generated beat times. We ignored beat times in the first 5 s of the excerpt in order to minimize initialization effects, thus the impulse trains were 25 s long, covering beat times between 5 and 30 s. The P-score (for a given algorithm and single excerpt) is the normalized proportion of beats that are correct, i.e. the number of algorithm-generated beats that fall within a small time-window, W_s of an annotator beat. The P-score is normalized by the number of algorithm or annotator beats, whichever is greatest, and is calculated as follows:

$$P = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{NP}\sum_{m=-W_s}^{+W_s}\sum_{n=1}^{N} y[n]\cdot a_s[n-m], \qquad (1)$$

where $a_s[n]$ is the impulse train from annotator s, $y[n]$ is the impulse train from the algorithm, N is the sample-length of impulse trains $y[n]$ and $a_s[n]$, W_s is the ''error'' window within which detected beats are counted as correct, and NP is a normalization factor defined by the maximum number of impulses in either impulse train:

$$NP = \max\left(\sum y[n], \sum a_s[n]\right). \qquad (2)$$

The ''error'' window, W_s was one-fifth the annotated beat, derived from the annotated taps by taking the median of the inter-tap intervals and multiplying by 0.2. This window, W_s, was calculated independently for each annotated impulse train, a_s.

The overall performance of each beat-tracking algorithm was measured by taking the average P-score across excerpts.

3.3 Tempo-extraction evaluation

For each excerpt, the histogram analysis of the annotated beat times, yielded two ground-truth peak tempi, GT_1 and GT_2, where GT_1 is the slowest. In addition, the

strength (salience) of GT_1 in comparison to GT_2 was also derived from the tempi histograms and is denoted as GST_1. GST_1 can vary from 0 to 1.0.

Each tempo-extraction algorithm generated two tempo values for each musical excerpt, T_1 and T_2, and its performance was measured by its ability to estimate the two tempi to within 8% of the ground-truth tempi. The performance measure was calculated as follows:

$$P = GST_1 \cdot TT_1 + (1 - GST_1) \cdot TT_2, \tag{3}$$

where TT_1 and TT_2 are binary operators indicating whether or not the algorithm-generated tempi are within 8% of the ground-truth tempi:

$$TT = \begin{cases} 1 & \text{if } |(GT - T)/GT| < 0.08, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Thus, the more salient a particular tempo is, the more weight it carries in the calculation of the *P*-score. The average *P*-score across all excerpts was taken as the overall measure of performance for each tempo extraction algorithm.

4. Results

4.1 Beat-tracking results

Overall results of the beat-tracking evaluation are shown in Figure 1 (upper plot). The results show that Dixon's algorithm performs best, however its average *P*-score is significantly higher than only that from Brossier's algorithm. Looking at the absolute range of performance across the algorithms shows that, with the exception of Brossier's algorithm, they all perform equally well, with *P*-scores differing by no more than 0.023.

To develop better intuition for the absolute value of the *P*-score, we calculated *P*-scores for each of our annotators by cross-correlating a single annotator's beat track for a given excerpt with the beat tracks from every other annotator (see Equation (1)). Average *P*-scores for each annotator are shown in Figure 1 (lower plot). While some individual annotator *P*-scores are lower than averaged algorithm *P*-scores, the average human annotator *P*-score (0.63) is significantly higher than that from any single algorithm ($p < 0.001$, bootstrapped equivalence test, see e.g. Efron & Tibshirani, 1993). However, if we take the best-performing algorithm on each excerpt and average those *P*-scores, we get an average score that is significantly higher than the average annotator *P*-score (see Figure 2). If we also take the best performing human annotator on each excerpt, we see an even higher average score. Together, these results suggest that an optimal combination of the current beat-tracking algorithms would perform better than the average human annotator but not an optimal human annotator.

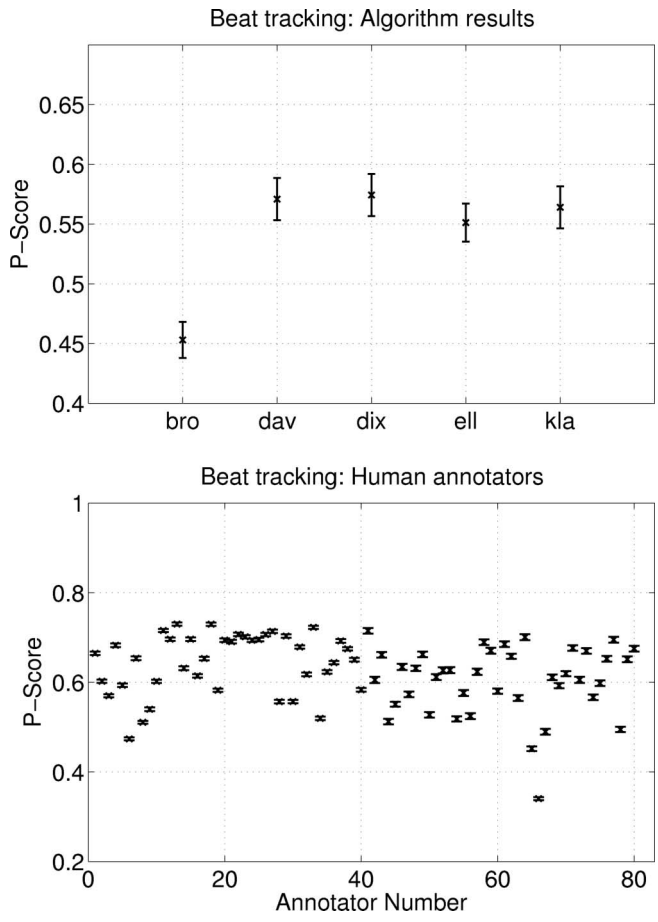

Fig. 1. Beat tracking evaluation results. Average *P*-scores for each algorithm are plotted (upper plot). Average *P*-scores for individual annotators are plotted in the lower plot. Error bars indicate standard error of the mean, estimated through bootstrapping across *P*-scores from individual excerpts. Note the different ordinate scales on the two subplots.

We also examined the algorithm *P*-scores as a function of a number of musical parameters, including excerpt genre, meter, the presence of percussion, and the most salient perceptual tempo. We used a coarse genre classification with the following general definitions: *Classical*: Western classical music including orchestral and chamber spanning eras from Renaissance to 20th century; *Hard*: loud and usually fast music, using mainly electric guitars (often with distortion) and drums, e.g. punk, heavy metal; *Jazz*: improvisational music with a strong meter, syncopation and a "swing" rhythm, including the sub-styles swing, vocal, bebop and fusion; *Pop*: light music with a medium beat, relatively simple rhythm and harmony and often a repeating structure; *Varia*: popular music genres that do not fall into the main categories and have in common that they can be considered as "listening music", e.g. folk, chanson, cabaret; *World*: non-Western music, typically folk and often poly-rhythmic, including African, Latin and Asian music. Results of this analysis are shown in Figure 3.

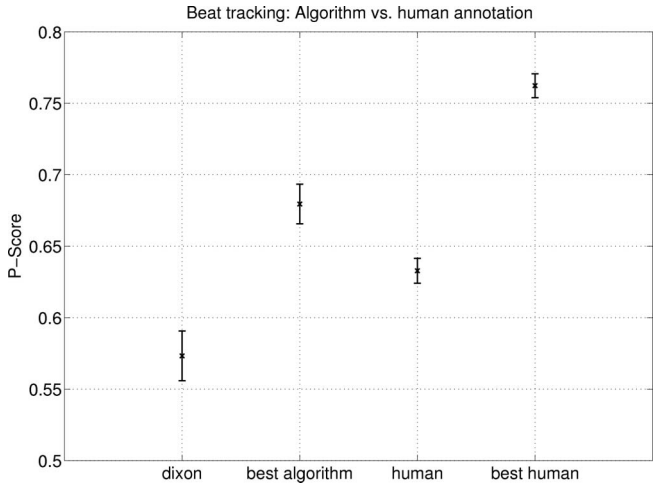

Fig. 2. Algorithm versus human-annotator beat tracking results. Average *P*-scores are shown for (1) the best-performing single algorithm (Dixon), (2) the best-performing algorithm on each excerpt, (3) all human annotators, and (4) the best-performing human annotator on each excerpt. Error bars indicate standard error of the mean, estimated through bootstrapping (Efron & Tibshirani, 1993) across *P*-scores from individual excerpts.

The top plot in Figure 3 reveals a number of differences in performance depending on the genre of the music:

- Algorithms differed in their sensitivity to genre: Davies' and Klapuri's algorithms show large performance variation across genre while Brossiers' and Ellis' algorithms show virtually no performance difference across genre.
- Algorithms sensitive to genre (Davies, Dixon, and Klapuri) performed best on Pop and World music, perhaps because of the straight, regular beat of Pop music and the strong rhythmic nature of World music.
- Brossier's, Davies' and Klapuri's algorithms performed worst on Hard music. Informal analyses showed that these algorithms often locked to a slower metrical level and/or to the upbeat when presented with this style of music, characterized by up-tempo and off-beat drums and guitars.
- Of the four top performing algorithms, Ellis' is the most stable across genre. It performs significantly worse than the other three on Pop music and worse than Davies' on World music, but it performs significantly better than Davies' and Klapuri's on Hard music and significantly better than Dixon's on Classical music.

Figure 3(b) shows the effect of percussion on the algorithms' beat-tracking ability. All algorithms show better performance on percussive music, although the

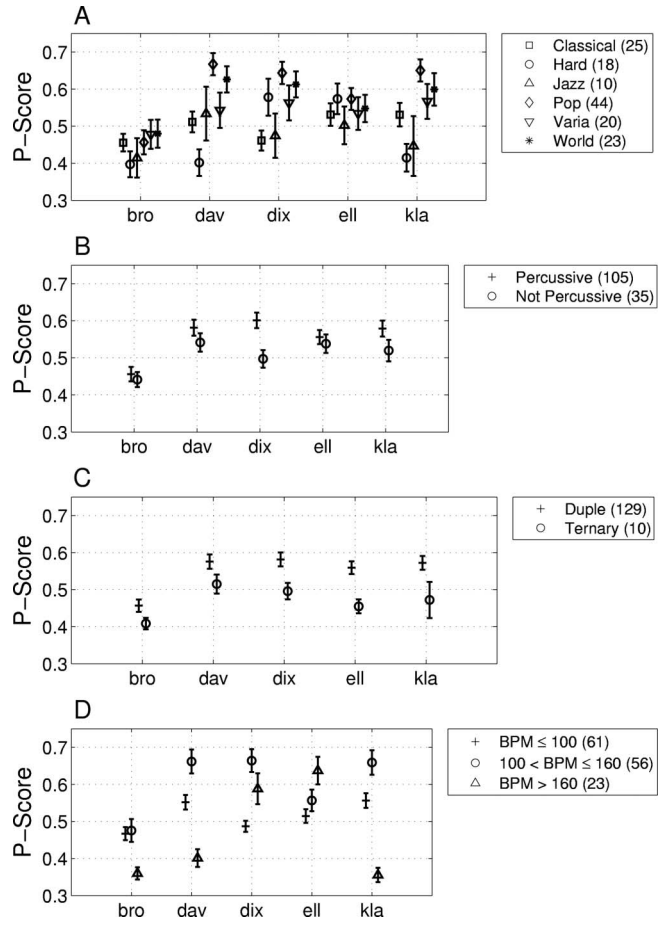

Fig. 3. Beat-tracking evaluation results as a function of (a) genre, (b) percussiveness, (c) meter and (d) most-salient ground-truth tempo. Average *P*-scores for each algorithm are plotted for each condition. Error bars indicate standard errors of the mean, estimated through bootstrapping across *P*-scores from individual excerpts. The total number of excerpts used in the effect of meter analysis (c) was 139 because one of the 140 test excerpts had a meter of 7/8 (not duple or ternary).

difference is significant only for Dixon's and Klapuri's algorithm. The three algorithms that showed the greatest sensitivity to music genre (Davies, Dixon, and Klapuri) also show the greatest sensitivity to the presence/absence of percussion. Dixon's algorithm shows the largest sensitivity to the presence of percussion with a *P*-score differential of 0.10 between the two cases.

Figure 3(c) shows that all algorithms perform significantly better on excerpts with duple meter than on excerpts with ternary meter. Ellis' algorithm shows the largest difference in performance, with *P*-score differential of 0.11 between the two cases.

Finally, Figure 3(d) shows beat-tracking performance as a function of the most salient perceived tempo (taken from the ground-truth data for each excerpt). Most algorithms perform best at mid-tempi (100 – 160 BPM) but Ellis' algorithm does best at higher tempi

(>160 BPM). Ellis' algorithm is also the most consistent, overall, across the three tempo categories. In contrast, the algorithms from Davies and Klapuri perform relatively poorly at high tempi and perform very differently in the different tempo categories. At low tempi (<100 BPM), Davies' and Klapuri's algorithms perform best, while Dixon's and Brossier's algorithms perform worst.

In addition to the overall *P*-score, we also evaluated the performance of each algorithm using a "partial" *P*-score, assessing them against only those annotated beat tracks for which the tempo (metrical level) was the same as that from the algorithm-generated beat track. Specifically, an annotation was used in the evaluation only if the tapped tempo was within 8% of the algorithm-generated tempo (the same criteria used for the tempo-extraction evaluation). The rationale for this analysis is that we wanted to see how well the algorithms beat-track at their preferred metrical level, with no penalty for choosing a perceptually less-salient metrical level. Figure 4 shows the results of this analysis for the algorithms (upper plot) as well as for individual annotators (lower plot). As one would expect, most algorithms show an elevated average score here in comparison to the normal *P*-scores (Figure 1). Brossier's algorithm, however, shows a slight decrease in score here, although the difference is not significant. In terms of this partial *P*-score, Ellis' algorithm does not perform as well (statistically) as the three other top-performing algorithms. The partial *P*-scores of individual annotators (lower plot) show an even greater increase, on average, than do the algorithms, in comparison to the normal *P*-scores. The plot shows that the scores from annotators 1–40 are higher, on average, than those from annotators 41–80. It should be noted that the two groups of annotators worked on separate sets of the musical excerpt database and that the second group (41–80) annotated a set of excerpts chosen for their extreme tempo (fast or slow). More information on the musical excerpt sets and annotators can be found in McKinney and Moelants (2006).

Another aspect of algorithm performance worth examining is computational complexity, which can be grossly measured by the time required to process the test excerpts. The IMIRSEL team has posted basic results of this beat-tracking evaluation on their Wiki page, including computation time for each algorithm (MIREX 2006a). The computation times of each algorithm are displayed here in Table 2 and should be interpreted with knowledge of each algorithm's implementation language, as displayed in Table 1. Generally, a MATLAB implementation of a particular algorithm will run slower than its optimized C/C++ counterpart. The algorithms were run on two different machines (differing in operating system and memory), however the processors and the processor speeds were identical in both machines.

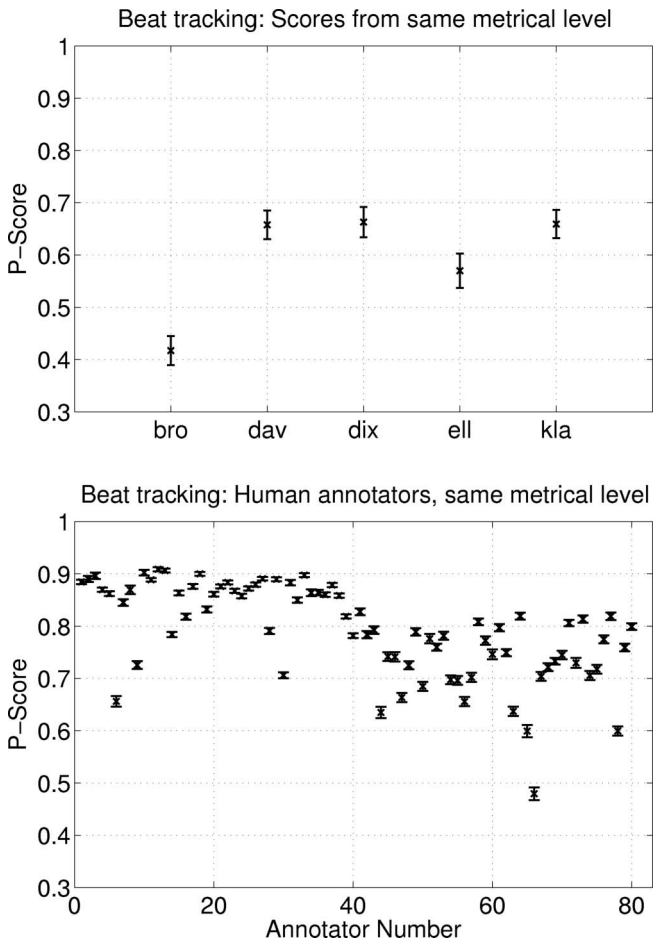

Fig. 4. Beat-tracking evaluation based on annotated beat tracks with the same tempo (and metrical level) as that from the algorithm-generated beat track. Average *P*-scores for each algorithm are shown in the upper plot and average *P*-scores for individual annotators are shown in the lower plot. Error bars indicate standard errors of the mean, estimated through bootstrapping across *P*-scores from individual excerpts.

Table 2. Computation time required for beat tracking. Computation times are for processing the entire collection of 140 30-s musical excerpts. Algorithms: BRO – Brossier; DAV – Davies & Plumbley; DIX – Dixon; ELL – Ellis; KLA – Klapuri. Results taken from MIREX (2006a).

	Algorithm				
	BRO	DAV	DIX	ELL	KLA
Computation time (s)	139	1394	639	498	1218
Implementation language	C/C++ Python	MATLAB	Java	MATLAB	C/C++

The numbers show that Dixon's algorithm, while performing the best, is also reasonably efficient. Brossier's algorithm is the most efficient, but it also performs the

worst. Ellis' algorithm has the second to shortest runtime despite being implemented in MATLAB, and thus, if optimized, could be the most efficient algorithm. In addition, his algorithm performed statistically equivalent to the best algorithms in many instances. The two slowest algorithms are those from Davies and Klapuri, however it should be noted that Davies' algorithm is implemented in MATLAB, while Klapuri's in C/C++.

4.2 Tempo extraction results

Overall results of the tempo-extraction evaluation are shown in Figure 5. In general, the algorithm *P*-scores here are higher and their range is broader than those from the beat-tracking task (see Figure 1). These differences may come from differences in how the two *P*-scores are calculated, but it is also likely that the task of extracting tempo *and* phase (beat-tracking) is more difficult than the task of extracting tempo alone.

The data in Figure 5 show that the algorithm from Klapuri gives the best overall *P*-score for tempo extraction, however it does not perform statistically better than the algorithm from Davies. Klapuri's algorithm, however, performs statistically better than all the other algorithms, while Davies' algorithm performs better than all but Alonso's (ALO2), statistically.

The overall results also show that Alonso's addition of spectral reassignment in his second algorithm (see Section 2.1) helps to improve the *P*-score, but not significantly in the mean across all excerpts.

As in the beat-tracking evaluation, we examined algorithm performance as a function of a few musicological factors, namely, genre, the presence of percussion, meter and most-salient perceptual tempo. Figure 6 shows a breakdown of the tempo-extraction *P*-scores according to these factors.

For the tempo task, there is not a single genre for which all tempo-extraction algorithms performed best or worst but a number of remarks can be made regarding the effect of genre:

- Classical tended to be the most difficult for most algorithms, with Varia also eliciting low *P*-scores. Both genres contain little percussion.
- The Hard genre provided the highest *P*-scores for most algorithms, while World also showed relatively high scores.
- Ellis' algorithm showed the least sensitivity to differences in genre, with average *P*-scores for the different genres clustered tightly together.
- Despite performing worst overall, Brossier's algorithm performed statistically equivalent (in the mean)

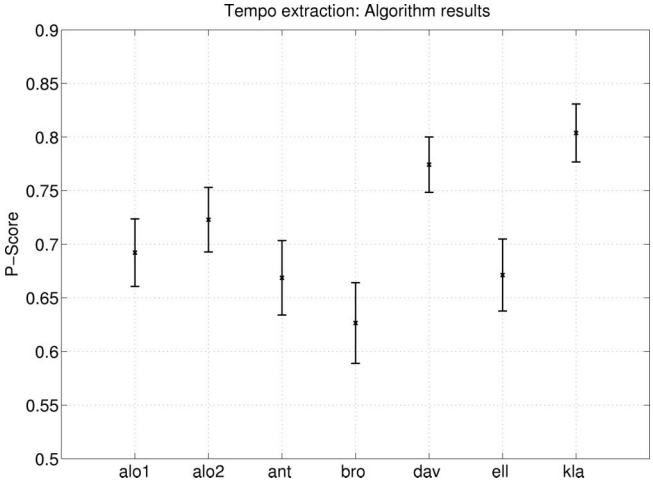

Fig. 5. Tempo extraction evaluation results. Average *P*-scores for each algorithm are plotted. Error bars indicate standard errors of the mean, estimated through bootstrapping across *P*-scores from individual excerpts.

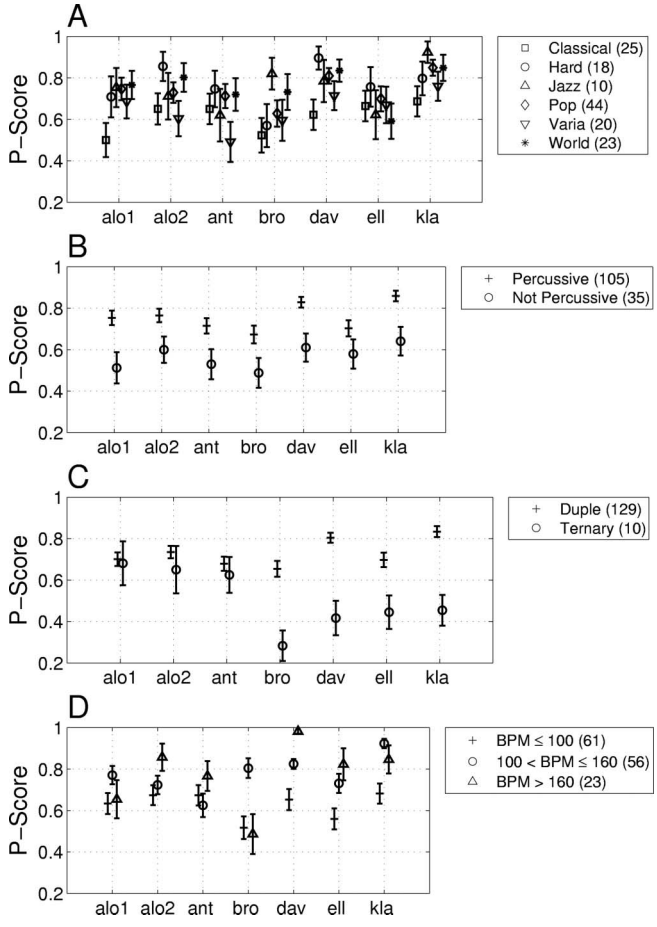

Fig. 6. Tempo extraction evaluation results as a function of (a) genre, (b) percussiveness, (c) meter and (d) most-salient ground-truth tempo. Average *P*-scores for each algorithm are plotted for each condition. Error bars indicate standard errors of the mean, estimated through bootstrapping across *P*-scores from individual excerpts.

Table 3. Computation time required for tempo extraction. Computation times are for processing the entire collection of 140 30-s musical excerpts. Algorithms: ALO – Alonso, Richard & David; ANT – Antonopoulos, Pikrakis & Theodoridis; BRO – Brossier; DAV – Davies & Plumbley; ELL – Ellis; KLA – Klapuri. Results taken from MIREX (2006b). *The C/C++ code for the ANT algorithm was generated directly using the MATLAB compiler and thus does not provide the typical complexity advantage gained from manually optimizing the C/C++ code.

	Algorithm						
	ALO1	ALO2	ANT	BRO	DAV	ELL	KLA
Computation time (s)	2126	4411	14500	1486	1389	445	1216
Implementation language	MATLAB	MATLAB	C/C++*	C/C++ Python	MATLAB	MATLAB	C/C++

to best algorithm (Klapuri) for the genres Jazz and World.

The effect of percussion is, in general, greater for the tempo-extraction task than it was for beat tracking. Figure 6(b) shows that every algorithm performs significantly worse on music without percussion than on music with percussion. It is likely the sharp transients associated with percussive instruments, which in turn elicit sharper driving functions, aid in the automatic extraction of tempo. For music without percussion, Klapuri's algorithm still shows the best mean performance, but is not significantly better than any of the other algorithms.

The effect of meter (Figure 6(c)) was large for four of the seven algorithms and was larger, for the effected algorithms, in the tempo-extraction task than in the beat-tracking task. The data show that these four algorithms (BRO, DAV, ELL, and KLA) perform significantly worse for ternary than for binary meters. Both Brossier (2006b) and Davies and Plumbley (2007, see also this article, Appendix A) make the explicit assumption that the two most salient tempi are related by a factor of two, thus it is not surprising that they perform worse on excerpts with ternary meter. The algorithms from Ellis (2007) and Klapuri et al. (2006, see also this article, Appendix B) do not contain any explicit limitations to duple meters, however they both seem to have implicit difficulty in extracting the perceptual tempi with ternary meters. Finally, the algorithms from Alonso et al. (2007) and Antonopoulos et al. (2007) do not contain assumptions regarding duple versus ternary meter and perform equally well (statistically) in both cases across our range of excerpts.

Figure 6(d) shows tempo extraction performance as a function of the most salient groundtruth tempo. Most algorithms perform best at high tempi (>160 BPM) while the rest perform best at mid-tempi ($100-160$ BPM). Almost all algorithms perform worst at low tempi (<100 BPM). Klapuri's algorithm performs significantly better than all other algorithms at mid-tempi while Davies' algorithm performs significantly better than the others at high tempi. Of all the conditions, Davies' algorithm at high tempi is the best-performing combination, with a near-perfect *P*-score.

As in the evaluation of beat tracking, we also looked at the overall algorithm run time of the tempo extraction algorithms as a measure of computational complexity. The results from the IMIRSEL team are posted on the MIREX Wiki page and include the same processor used for the beat-tracking evaluation (MIREX 2006b). It appears from their results, presented here in Table 3, that the algorithm from Antonopoulos et al. (2007) is by far (nearly an order of magnitude) more complex than all the other algorithms. It is likely that this computational load comes from a number of factors including their self-similarity-based driving function, their multi-pass approach to periodicity detection, the iterative method for periodicity voting as well as non-optimized C/C++ code. Ellis' algorithm is by far the most efficient, processing the excerpts in less than half the time as the next fastest algorithm (despite being implemented in MATLAB). It is interesting to note that the additional computation (spectral reassignment) in Alonso's second entry, ALO2, increased the computation time relative to ALO1 by more than a factor of two, but the performance remained statistically the same (see Figure 5). Again, these results need to be interpreted with knowledge of the implementation language of each algorithm (see Table 3).

5. Discussion

We have evaluated a number of algorithms for automatic beat tracking and tempo extraction in musical audio using criteria based on the population perception of beat and tempo. The main findings of the evaluation are as follows:

- Human beat trackers perform better, on average, than current beat-tracking algorithms, however an optimal combination of current algorithms would outperform the average human beat tracker.

- Algorithms for beat tracking and tempo extraction perform better on percussive music than on non-percussive music. The effect was significant across all tempo-extraction algorithms but not across all beat-tracking algorithms.
- Algorithms for beat tracking and tempo extraction perform better on music with duple meter than with ternary meter. The effect was significant across all beat-tracking algorithms but not across all tempo-extraction algorithms.
- The best performing tempo-extraction algorithms run simultaneous periodicity detection in multiple frequency bands (ALO and KLA) or on multiple driving functions (DAV).
- The best performing beat-tracking algorithms (DIX and DAV) use relatively low-resolution driving functions (10 and 11.6 ms, respectively).
- Overall computational complexity (measured in computation time) does not appear to correlate with algorithm performance.

This work extends a summary of an earlier tempo evaluation at the 2004 MIREX in which a different database of music was used, notated only with a single tempo value (Gouyon et al., 2006). In order to accommodate a single ground-truth tempo value for each excerpt in that evaluation, two types of tempo accuracies were measured: one based on estimating the single tempo value correctly and a second based on estimating an integer multiple of the ground-truth tempo (thus finding any metrical level). Here, we chose to treat the ambiguity in metrical level through robust collection of perceptual tempi for each excerpt. We took the dominant perceptual tempi, characterized through the tempi distribution of the listener population, as the ground-truth tempi for each excerpt. The use of perceptual tempi in this study is advantageous in that it inherently deals with the notion of metrical ambiguity and for many applications, including music playlisting and dance, it is the *perceptual* tempo that counts. However, in other applications, such as auto-accompaniment in real-time performance, notated tempo is the desired means of tempo communication. For these applications, a separate evaluation of notated-tempo extraction would be useful.

Our evaluation shows that the beat-tracking algorithms come close but do not quite perform as well, on average, as human listeners tapping to the beat. Additionally, while it is not exactly fair comparing the *P*-scores between the tempo-extraction and beat-tracking evaluations, it appears that beat-tracking performance, in general, is poorer than the performance of the tempo-extraction algorithms. Apparently the additional task of extracting phase of the beat proves difficult.

Looking at the various parameters of the algorithms and their performance, we can postulate on a few key aspects. It appears from the tempo-extraction results that algorithms that process simultaneous driving functions, either multi-frequency bands or different types of driving functions, perform better. The best performing tempo extractors (KLA, DAV, ALO) all contain multiple frequency bands or driving functions. The same advantage does not seem to hold for beat-tracking, where Dixon's algorithm processes a single broad-band driving function.

About half of the algorithms presented here calculate explicit event onsets for the generation of their driving functions. Two of the best performing algorithms for both beat tracking and tempo extraction (DAV and KLA), however, do not calculate explicit onsets from the audio signal but instead rely on somewhat more direct representations of the audio. The fact that they perform as well as they do supports previous work that suggests one does not need to operate at the "note level" in order to successfully extract rhythmic information from a musical audio signal (Scheirer, 1998; Sethares et al., 2005).

Several of the algorithms (ALO, DAV, ELL, KLA) use a sort of perceptual weighting on their final choice of tempi, emphasizing tempi near 120 BPM while deemphasizing higher and lower tempi. This type of weighting could adversely effect algorithm performance at high and low tempi in that the algorithm could track the beats at the wrong metrical level. It is interesting to note, however, that all four of these algorithms are the top-performing tempo-extractors at high tempi (>160 BPM) and that Ellis' beat-tracker performs best in the same category. Also of interest is the fact that Davies' and Klapuri's beat-trackers perform relatively poorly at high tempi, but their tempo-extractors are the best and third-best in the same tempo range. It is likely that, at high tempi, the beat-alignment portions of their algorithms are not robust or their algorithms switch to tracking lower metrical levels.

Finally, it appears that the time resolution of the driving function, at least for beat-tracking, does not need to be ultra-high. The best performing beat trackers (DIX and DAV) use time resolutions of 10 and 11.6 ms and outperform other algorithms with higher time resolutions. The best performing tempo extractor (KLA) has a time resolution of 5.8 ms, while the second best (DAV) has a time resolution of 11.6 ms, outperforming others with higher time resolutions. Of course it is the complete combination of parameters and functions that dictate overall performance, but this type of analysis can help constrain design guidelines for future algorithm design.

Acknowledgements

We would like to thank J. Stephen Downie and other members of the IMIRSEL team, who planned, facilitated and ran the MIREX algorithm evaluations. Andreas

Ehmann, Mert Bay, Cameron Jones and Jin Ha Lee were especially helpful with the set-up, processing, and analysis of results for both the Tempo Extraction and Beat Tracking evaluations.

We would also like to thank Miguel Alonso, Iasonas Antonopoulos, Simon Dixon, Dan Ellis and Armin Kohlrausch for valuable comments on an earlier version of this article.

Matthew Davies was funded by a College Studendship from Queen Mary University of London and by EPSRC grants GR/S75802/01 and GR/S82213/01.

References

Alonso, M., David, B. & Richard, G. (2006). Tempo extraction for audio recordings. From the Wiki-page of the Music Information Retrieval Evaluation eXchange (MIREX). Retrieved 1 May 2007 from http://www.music-ir.org/evaluation/MIREX/2006_abstracts/TE_alonso.pdf

Alonso, M., Richard, G. & David, B. (2007). Tempo estimation for audio recordings. *Journal of New Music Research*, *36*(1), 17–25.

Antonopoulos, I., Pikrakis, A. & Theodoridis, S. (2006). A tempo extraction algorithm for raw audio recordings. From the Wiki-page of the Music Information Retrieval Evaluation eXchange (MIREX). Retrieved 1 May 2007 from http://www.music-ir.org/evaluation/MIREX/2006_abstracts/TE_antonopoulos.pdf

Antonopoulos, I., Pikrakis, A. & Theodoridis, S. (2007). Self-similarity analysis applied on tempo induction from music recordings. *Journal of New Music Research*, *36*(1), 27–38.

Baird, B., Blevins, D. & Zahler, N. (1993). Artificial intelligence and music: Implementing an interactive computer performer. *Computer Music Journal*, *17*(2), 73–79.

Bello, J.P., Duxbury, C., Davies, M.E. & Sandler, M.B. (2004). On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, *11*(6), 553–556.

Brossier, P. (2006a). *Automatic Annotation of Musical Audio for Interactive Applications*. PhD thesis, Queen Mary, University of London, London, August.

Brossier, P. (2006b). The aubio library at MIREX 2006. From the Wiki-page of the Music Information Retrieval Evaluation eXchange (MIREX). Retrieved 1 May 2007 from http://www.music-ir.org/evaluation/MIREX/2006_abstracts/AME_BT_OD_TE_brossier.pdf

Dannenberg, R. (1984). An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference*, San Francisco, pp. 193–198. Computer Music Association: San Francisco, CA.

Davies, M.E.P. & Plumbley, M.D. (2005). Comparing mid-level representations for audio based beat tracking. In *Proceedings of the DMRN Summer Conference*, Glasgow, Scotland, pp. 36–41.

Davies, M.E.P. & Plumbley, M.D. (2007). Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech and Language Processing*, *15*(3), 1009–1020.

Dixon, S. (1999). A beat tracking system for audio signals. In *Proceedings of the Conference on Mathematical and Computational Methods in Music*, pp. 101–110, Wien. Austrian Computer Society: Vienna.

Dixon, S. (2000). A beat tracking system for audio signals. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, Melbourne, pp. 778–788.

Dixon, S. (2006). MIREX 2006 audio beat tracking evaluation: BeatRoot. From the Wiki-page of the Music Information Retrieval Evaluation eXchange (MIREX). Retrieved 1 May 2007 from http://www.music-ir.org/evaluation/MIREX/2006_abstracts/BT_dixon.pdf

Dixon, S. (2007). Evaluation of the audio beat tracking system BeatRoot. *Journal of New Music Research*, *36*(1), 39–50.

Dixon, S. & Cambouropoulos, E. (2000). Beat tracking with musical knowledge. In W. Horn (Ed.), *Proceedings of the 14th European conference on artificial intelligence* (pp. 626–630). Amsterdam: IOS Press.

Efron, B. & Tibshirani, R.J. (1993). *An introduction to the bootstrap. Monographs on statistics and applied probability*. New York: Chapman & Hall.

Ellis, D.P.W. (2006). Beat tracking with dynamic programming. From the Wiki-page of the Music Information Retrieval Evaluation eXchange (MIREX). Retrieved 1 May 2007 from http://www.music-ir.org/evaluation/MIREX/2006_abstracts/TE_BT_ellis.pdf

Ellis, D.P.W. (2007). Beat tracking with dynamic programming. *Journal of New Music Research*, *36*(1), 51–60.

Gasser, M., Eck, D. & Port, R. (1999). Meter as mechanism: a neural network that learns metrical patterns. *Connection Science*, *11*, 187–216.

Goto, M. & Muraoka, Y. (1994). A beat tracking system for acoustic signals of music. In *Proceedings of the second ACM international conference on multimedia* (pp. 365–372). ACM: San Francisco, CA.

Goto, M. & Muraoka, Y. (1998). Musical understanding at the beat level: real-time beat tracking for audio signals. In D.F. Rosenthal and H.G. Okuno (Eds.), *Computational auditory scene analysis* (pp. 157–176). Mahwah, NJ: Lawrence Erlbaum Associates.

Gouyon, F., Klapuri, A., Dixon, S., Alonso, M., Tzanetakis, G., Uhle, C. & Cano, P. (2006). An experimental comparison of audio tempo induction alorighms. *IEEE Transactions on Audio, Speech and Language Processing*, *14*(5), 1832–1844.

Klapuri, A., Eronen, A. & Astola, J. (2006). Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(1), 342–355.

Large, E.W. & Kolen, J.F. (1994). Resonance and the perception of musical meter. *Connection Science*, *6*(1), 177–208.

Lee, C.S. (1985). The rhythmic interpretation of simple musical sequences: towards a perceptual model. In R. West, P. Howell and I. Cross (Eds.), *Musical structure and cognition* (pp. 53–69). London: Academic Press.

Lerdahl, F. & Jackendoff, R. (1983). *A generative theory of tonal music*. Cambridge, MA: MIT Press.

Longuet-Higgins, H.C. & Steedman, M.J. (1971). On interpreting Bach. In B. Meltzer and D. Michie (Eds.), *Machine intelligence 6* (pp. 221–241). Edinburgh: Edinburgh University Press.

Longuet-Higgins, H.C. & Lee, C.S. (1982). Perception of musical rhythms. *Perception*, *11*, 115–128.

Longuet-Higgins, H.C. & Lee, C.S. (1984). The rhythmic interpretation of monophonic music. *Music Perception*, *1*(4), 424–441.

McAuley, J.D. (1995). *Perception of time as phase: toward an adaptive-oscillator model of rhythmic pattern processing*. PhD thesis, Indiana University.

McKinney, M.F. & Moelants, D. (2006). Ambiguity in tempo perception: What draws listeners to different metrical levels? *Music Perception*, *24*(2), 155–166.

Miller, B.O., Scarborough, D.L. & Jones, J.A. (1992). On the perception of meter. In M. Balaban, K. Ebcioglu and O. Laske (Eds.), *Understanding music with AI: perspectives on music cognition* (pp. 428–447). Cambridge: MIT Press.

MIREX Audio Beat Tracking Results. (2006a). From the Wiki-page of the Music Information Retrieval Evaluation eXchange (MIREX). Retrieved 1 May 2007 from http://www.music-ir.org/mirex2006/index.php/Audio_Beat_Tracking_Results

MIREX Audio Tempo Extraction Results. (2006b). From the Wiki-page of the Music Information Retrieval Evaluation eXchange (MIREX). Retrieved 1 May 2007 from http://www.music-ir.org/mirex2006/index.php/Audio_Tempo_Extraction_Results

MIREX Music Information Retrieval Evaluation eXchange. (2006c). Retrieved 1 May 2007 from http://www.music-ir.org/mirex2006/index.php/Main_Page

Moore, B.C.J. (1995). *Hearing. Handbook of perception and cognition*, 2nd ed. New York: Academic Press.

Parncutt, R. (1994). A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception*, *11*(4), 409–464.

Povel, D.-J. & Essens, P. (1985). Perception of temporal patterns. *Music Perception*, *2*(4), 411–440.

Scheirer, E.D. (1998). Tempo and beat analysis of acoustical musical signals. *Journal of the Acoustical Society of America*, *103*, 588–601.

Sethares, W.A., Morris, R.D. & Sethares, J.C. (2005). Beat tracking of musical performances using low-level audio features. *IEEE Transactions on Speech and Audio Processing*, *13*(2), 275–285.

Toiviainen, P. (1998). An interactive MIDI accompanist. *Computer Music Journal*, *22*(4), 63–75.

Vantomme, J.D. (1995). Score following by temporal pattern. *Computer Music Journal*, *19*(3), 50–59.

Vercoe, B. (1997). Computational auditory pathways to music understanding. In I. Deliège and J. Sloboda (Eds.), *Perception and cognition of music* (pp. 307–326). Hove: Psychology Press.

Appendix A: Extended summary of the Davies & Plumbley algorithm

The Davies and Plumbley submissions to the Audio Tempo Extraction and Audio Beat Tracking tasks within MIREX 2006 are based on their existing beat tracking system, the full details of which may be found in Davies and Plumbley (2007). Within this summary we provide an overview of this beat tracking system highlighting those areas which have been modified from the originally published approach. In particular two main changes have been implemented: (i) in line with the tempo extraction task, the algorithm has been updated to provide two perceptual tempo estimates; and (ii) for the beat tracking task, the algorithm has been adapted to generate three parallel sequences of beats. Each sequence is derived from a different onset detection function, from which the eventual output is determined via a post-processing confidence measure.

Beat tracker overview

The basic operation of the Davies and Plumbley beat tracking system, shown in Figure 7(a), can be broken down into four discrete stages:

- **Input representation**: The first stage in the beat tracking algorithm is the transformation of the audio signal into a representation more suited to identifying beat locations. The chosen input representation is the *onset detection function* – a continuous signal which exhibits peaks at likely onset locations. The onset detection function is calculated by measuring the spectral difference between short term (11 ms) analysis frames and is referred to as the *complex domain* method (Bello et al., 2004). An example audio signal and onset detection function are shown in Figures 7(b) and (c).
- **Beat period**: The process of extracting the beat period (the time between consecutive beats) centres on the analysis of the autocorrelation function of the onset detection function. As can be seen from the example plot in Figure 7(d) many peaks are present in the autocorrelation function. The majority correspond to periodicities which are too long to be considered reasonable candidates for the tempo. To extract a meaningful estimate of the beat period, the autocorrelation function is passed through a *shift-invariant* comb filterbank. The comb filterbank is implemented in matrix form, where the columns of

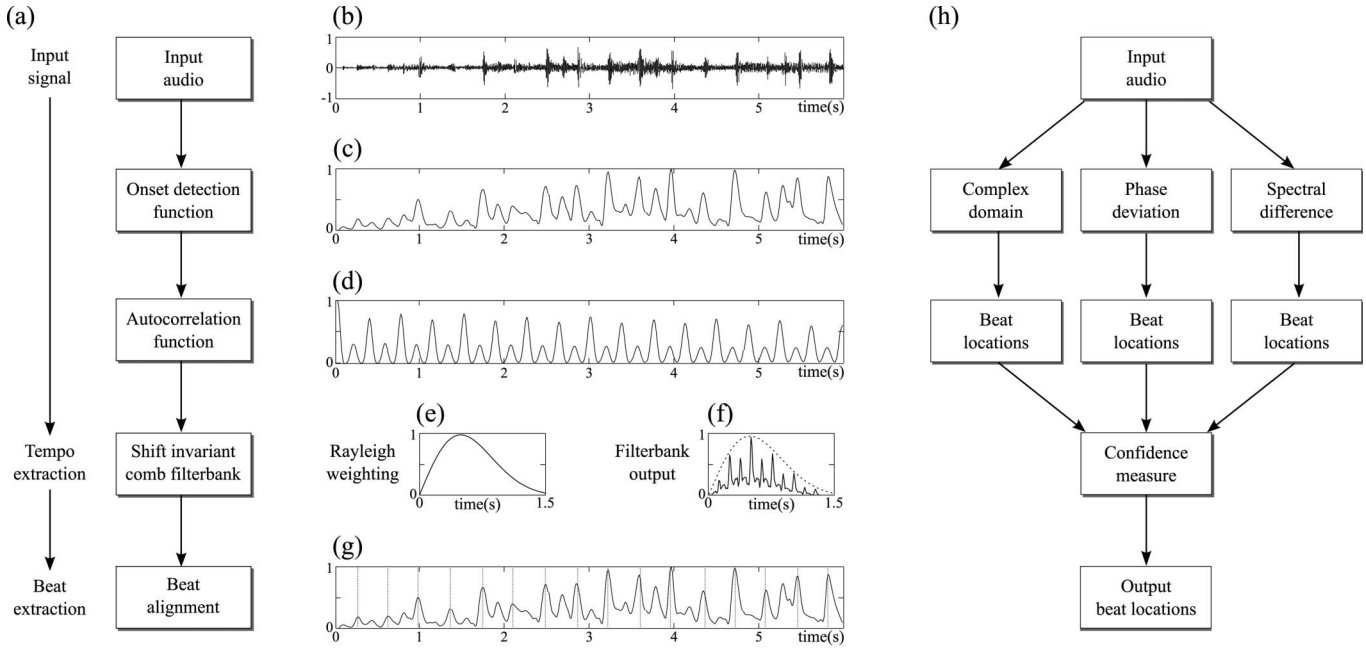

Fig. 7. (a) Overview of the Davies and Plumbley beat tracking system. (b) Audio signal. (c) Onset detection function. (d) Autocorrelation function. (e) Rayleigh weighting function. (f) Comb filterbank output. (g) Onset detection function with beat locations. (h) Beat tracking with multiple onset detection function inputs.

the matrix contain comb-like impulse trains covering a range of periodicities to an upper limit of 1.5 s.

To indicate an approximate perceptual weighting over these periodicities, the relative strength of each column is set by a Rayleigh distribution function (as shown in Figure 7(e)). This weighting gives most emphasis to periodicities close to 500 ms. The comb filterbank output function (in Figure 7(f)) is calculated as the sum of the product of each column of the matrix with the autocorrelation function. The beat period is taken as the index of the maximum value of the output function.

For the beat tracking system, only a single beat period estimate is necessary, however within the tempo extraction task, two are required (one for each perceptual tempo candidate). To identify two periodicities from the comb filterbank output it may not be sufficient to extract the two most significant peaks, whose height may be distorted by the Rayleigh weighting function. Instead, the pair of peaks, which are most closely related by a factor of two and are the strongest in the filterbank output signal, are extracted. The periodicity of the higher of the two peaks is taken as the primary tempo estimate and the lower as the secondary. The salience between the two tempi is calculated as the ratio of the height of the stronger peak to the sum of the heights of both peaks.

- **Beat phase**: Once an estimate of the beat period has been identified the next task within the beat tracking system is to find the locations of the beats. To represent the known beat period an impulse train with impulses at beat period intervals is formed. Over-lags ranging one beat period, this impulse train is cross-correlated with the onset detection function. The lag at which the impulse train most strongly correlates with the onset detection function is taken to represent the phase of the beats. Subsequent beat locations can be predicted at beat period intervals from this point. An example onset detection function with beat locations is given in Figure 7(g).
- **High-level processing**: Within the tempo extraction task only global estimates of the two tempi are required, therefore the tempo estimates can be derived from one global autocorrelation function calculated across the length of the onset detection function. However, for the beat tracking task, the beat tracker should be reactive to changes in the phase or tempo of the beats. To enable this sensitivity, both the beat period and beat phase processes are repeated on overlapping onset detection function frames. Each frame is 6 s in length with a 1.5 s step increment.

A potential problem with the repeated estimation of beat period and phase is that no effort is made to enforce continuity. This is resolved by measuring the consistency between sequential beat period estimates. If three consecutive beat period estimates are close then the Rayleigh weighting (within the shift-invariant comb filterbank) is replaced with a tighter Gaussian weighting centred on the consistent beat period estimate. This greatly limits the range of observable

periodicities, forcing the beats to be predicted at approximate beat period intervals. The downside of this restriction is that it can leave the beat tracker blind to any global changes in tempo, i.e. those beyond the limited range of the Gaussian weighting.

To address the compromise between sensitivity to tempo change and consistency of beats, a *Two State Model* for beat tracking is implemented. The first state, referred to as the *General State*, uses the Rayleigh weighting to find an initial tempo estimate and to track tempo changes while the second, known as the *Context-dependent State* maintains continuity within regions of approximately constant tempo with the Gaussian weighting. Further details of the complete beat tracking system can be found in Davies and Plumbley (2007).

Beat tracking with multiple inputs

The beat tracking system presented in Davies and Plumbley (2007) employs a single onset detection function (the complex domain method) as the basis for finding beat locations. Informal experiments by Davies and Plumbley (2005) revealed that overall beat tracking performance could be improved by choosing between multiple onset detection functions. In these initial tests, the selection was made by hand, however for the beat tracking task, an automatic approach is presented. In all, three onset detection functions are calculated for a given audio signal: (i) *complex domain*; (ii) *phase deviation* and (iii) *spectral difference*, each of which is described in Bello et al. (2004).

The beat tracking system operates independently on each onset detection function, which results in three sequences of beat times. To make the decision over which beat sequence to take as the sole output from the system, each sequence is rendered as an impulse train at the temporal resolution of the onset detection function. A time-limited (100 ms) cross-correlation between each beat impulse train and its respective onset detection function is used to evaluate each beat tracking submission. The beat sequence which is most strongly correlated with a detection function is then taken as the output. An overview of the beat tracking process with multiple onset detection functions is shown in Figure 7(h).

Appendix B: Extended summary of the Klapuri algorithm

This appendix describes the meter analysis method of Klapuri et al. (2006). Figure 8 shows an overview of the method. The different parts are now explained one at a time.

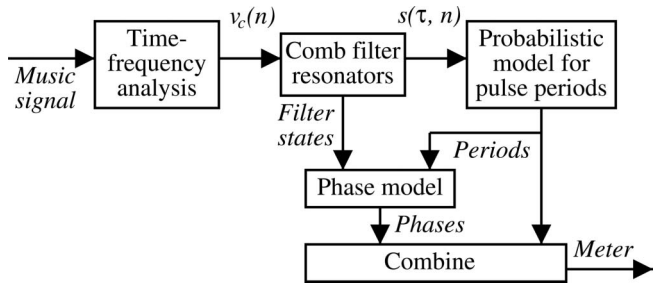

Fig. 8. Overview of the meter analysis method of Klapuri et al. (2006). (Reprinted, with permission, from Klapuri et al. (2006). © 2006 IEEE.)

Time-frequency analysis

The time-frequency analysis front-end aims at measuring the degree of musical accent (or, stress) as a function of time at four different frequency ranges. Acoustic input signals are sampled at 44.1 kHz rate and 16-bit resolution. Discrete Fourier transforms are calculated in successive 23 ms time frames with 50% overlap. In each frame, 36 triangular-response bandpass filters are simulated that are uniformly distributed on a critical-band scale between 50 Hz and 20 kHz (Moore 1995, p. 176). The power at each band is calculated and stored to $x_b(n)$, where n is the frame index and $b = 1, 2, \ldots, 36$ is the band index. The signals $x_b(n)$ are interpolated to obtain a time resolution of 5.8 ms.

The task of musical accent estimation is then recasted as measuring the amount of incoming spectral energy as a function of time. From the human hearing viewpoint, it makes sense to operate on a logarithmic magnitude scale, therefore the subband envelopes are log-compressed as $y_b(n) = \log(x_b(n) + \epsilon)$, where ϵ is a small constant. A low-pass filter with a 10-Hz cutoff frequency is then applied to smooth the compressed power envelopes. The resulting smoothed envelopes are differentiated and half-wave rectified (constraining negative values to zero) to obtain the signal $u_b(n)$ which measures the degree of incoming energy at band b in frame n in a perceptually meaningful way.

Finally, each 9 adjacent bands are linearly summed to get 4 accent signals at different frequency ranges ("channels") c:

$$v_c(n) = \sum_{b=9(c-1)+1}^{9c} u_b(n), \quad c = 1, \ldots, 4. \qquad \text{(B1)}$$

Periodicity analysis

Periodicity of the bandwise accent signals $v_c(n)$ is analysed in order to measure the salience of different pulse candidates. This is done using a bank of comb filter

resonators similar to those employed by Scheirer (1998). Each individual comb filter corresponds to a pulse candidate. The comb filters have an exponentially-decaying impulse response where the *half-time* refers to the delay during which the response decays to a half of its initial value. The output of a comb filter with delay τ for input $v_c(n)$ is given by

$$r_c(\tau, n) = \alpha_\tau r_c(\tau, n-\tau) + (1-\alpha_\tau)v_c(n), \qquad \text{(B2)}$$

where the feedback gain α_τ determines the half-time (here 3 s).

A bank of such resonators is applied, with τ getting values from 1 to $\tau_{\max}$, where the maximum period $\tau_{\max}$ corresponds to four seconds. Short-time energies $s_c(\tau, n)$ of each comb filter in channel c at time n are calculated by squaring and averaging the filter outputs and by applying a normalization procedure (see Klapuri et al., 2006, for details).

Finally, the salience $s(\tau, n)$ of a metrical pulse with period-length τ at time n is obtained by summing across channels:

$$s(\tau, n) = \sum_{c=1}^{4} s_c(\tau, n). \qquad \text{(B3)}$$

Probabilistic model for pulse periods

The comb filters serve as feature extractors for two probabilistic models. One model is used to decide the period-lengths of the tatum, tactus, and measure pulses and the other model is used to estimate the corresponding phases (see Figure 8).

For period estimation, a hidden Markov model is used that describes the simultaneous evolution of the three pulses. The observable data consists of the instantaneous energies of the resonators, $s(\tau, n)$, denoted $\boldsymbol{s}_n$ in the following. The unobservable hidden variables are the tatum period τ_n^{A}, tactus period τ_n^{B}, and measure period τ_n^{C}. The vector $\boldsymbol{q}_n = [j, k, l]$ is used to denote a "meter state", equivalent to $\tau_n^{\mathrm{A}} = j$, $\tau_n^{\mathrm{B}} = k$, and $\tau_n^{\mathrm{C}} = l$. The hidden state process is a first-order Markov model which has an initial state distribution $P(\boldsymbol{q}_1)$ and transition probabilities $P(\boldsymbol{q}_n|\boldsymbol{q}_{n-1})$. The observable variable is conditioned on the current state so that we have state-conditional observation densities $p(\boldsymbol{s}_n|\boldsymbol{q}_n)$.

The joint probability density of a state sequence $Q = (\boldsymbol{q}_1\boldsymbol{q}_2 \ldots \boldsymbol{q}_N)$ and observation sequence $O = (\boldsymbol{s}_1\boldsymbol{s}_2 \ldots \boldsymbol{s}_N)$ can be written as

$$p(Q, O) = P(\boldsymbol{q}_1)p(\boldsymbol{s}_1|\boldsymbol{q}_1)\prod_{n=2}^{N} P(\boldsymbol{q}_n|\boldsymbol{q}_{n-1})p(\boldsymbol{s}_n|\boldsymbol{q}_n), \qquad \text{(B4)}$$

where the term $P(\boldsymbol{q}_n|\boldsymbol{q}_{n-1})$ can be decomposed as

$$P(\boldsymbol{q}_n|\boldsymbol{q}_{n-1}) = P\left(\tau_n^{\mathrm{B}}|\tau_{n-1}^{\mathrm{B}}\right)P\left(\tau_n^{\mathrm{A}}|\tau_n^{\mathrm{B}}, \tau_{n-1}^{\mathrm{A}}\right) P\left(\tau_n^{\mathrm{C}}|\tau_n^{\mathrm{B}}, \tau_{n-1}^{\mathrm{C}}\right). \qquad \text{(B5)}$$

Above, it has been assumed that the tatum period τ_n^{A} at time n depends only on the tatum period at time $n-1$ and on the beat period at time n, but not on the measure period. Similarly, τ_n^{C} depends only on τ_{n-1}^{C} and τ_n^{B}, but not on τ_n^{A} directly.

Klapuri et al. (2006) proposed a structured and parametric form for the term $P(\boldsymbol{q}_n|\boldsymbol{q}_{n-1})$ in order to impose musical constraints on the estimated meter. In practice, the period lengths were constrained to be slowly-varying, and binary or ternary integer relationships were preferred between the concurrent pulses. The optimal meter, that is, the optimal state sequence $Q = (\boldsymbol{q}_1\boldsymbol{q}_2 \ldots \boldsymbol{q}_N)$ can be found using the Viterbi algorithm. In a causal model, the meter estimate $\boldsymbol{q}_n$ at time n is determined according to the end-state of the best partial path at that point in time.

The period length of the tactus pulse alone suffices for tempo estimation. In the MIREX-2006 contest, the median period length over the latter half of the analysed signal was used. It should be noted, however, that joint estimation of the three pulses improves the robustness of the tactus analysis. In order to conform to the MIREX tempo extraction task, another tempo value is obtained by doubling or halving the first estimate towards a mean tempo of 109 BPM.

Phase estimation

For beat tracking, also the temporal locations of each individual beat are required. This task is called beat phase estimation and is here based on the latest outputs of the comb filter corresponding to the estimated tactus period $\hat{\tau}_n^{\mathrm{B}}$. There are four channels, therefore there are four output signals $r_c\left(\hat{\tau}_n^{\mathrm{B}}, j\right)$, where $c = 1, \ldots, 4$ is the channel index and the phase index j takes on values between $n - \tau + 1$ and n when estimation is taking place at time n. The four signals are summed in order to get an observation vector $h_n(j)$ for phase estimation:

$$h_n(j) = \sum_{c=1}^{4} (6-c)r_c\left(\hat{\tau}_n^{\mathrm{B}}, j\right), \qquad \text{(B6)}$$

where $j = n - \tau + 1, \ldots, n$. Note that the lower-frequency channels are given a larger weight in the sum. This is is motivated by the "stable bass" rule of Lerdahl and Jackendoff (1983) and improves the robustness of phase estimation in practice.

The observation vector (B6) is used in another hidden Markov model which describes the phase evolution of the tactus pulse. Joint estimation with the other two pulses was not attempted.