REAL-TIME BEAT-SYNCHRONOUS ANALYSIS OF MUSICAL AUDIO

Adam M. Stark, Matthew E. P. Davies and Mark D. Plumbley[*]

Centre for Digital Music
Queen Mary University of London
London, United Kingdom
adam.stark@elec.qmul.ac.uk

ABSTRACT

In this paper we present a model for beat-synchronous analysis of musical audio signals. Introducing a real-time beat tracking model with performance comparable to offline techniques, we discuss its application to the analysis of musical performances segmented by beat. We discuss the various design choices for beat-synchronous analysis and their implications for real-time implementations before presenting some beat-synchronous harmonic analysis examples. We make available our beat tracker and beat-synchronous analysis techniques as externals for Max/MSP.

1. INTRODUCTION

The automated analysis of musical performance in real-time can provide useful knowledge about the nature of that performance. This information can then be used in interactive musical systems, such as score following systems [1], to create intelligent and articulate musical responses, automatically, to human musical initiations.

Beat-synchronous analysis is the analysis of a musical signal segmented by the rhythmic and metrical events of that same signal. This is achieved through the use of a beat tracker (e.g. [2]) - a technique for automatically detecting the dominant metrical pulse, or 'beat', of a piece of music. Beat-synchronous analysis has been used widely in offline applications and has been shown to improve performance, for example in chord recognition [3] and structural segmentation [4].

Encouraged by these positive results, we seek to extend the use of beat-synchronous analysis to real-time applications. We present a new model for real-time beat tracking, showing performance comparable to state of the art offline models. We then present a methodology for beat-synchronous analysis, in particular harmonic analysis, discussing the various design choices and their implications for real-time applications.

There are several benefits of using a beat-tracker to augment harmonic analysis. Firstly, in many forms of music, harmonic changes often occur at beat locations and so segmentation by a rhythmic feature such as the beat may improve performance. Secondly, we may wish to use the harmonic analysis to infer something about the structure of the performed music, using for example some form of self-similarity analysis. In contrast to a frame by frame analysis where there are many frames per beat, beat-synchronous segmentation greatly reduces the size of the data, allowing the analysis of longer segments of audio. A further benefit of beat-synchronous analysis is that the same musical phrase or passage will be represented using the same number of data points regardless of tempo variations.

Beat-synchronous analysis has been used previously in real-time applications [5], creating a sub-beat divided matrix representation of an audio signal through beat-synchronous spectral analysis. However in this paper we present a full discussion of the merits and disadvantages of the different design choices, and their implications for real-time processing, outside of the context of the application.

This paper is structured as follows. In section 2 we present a model for real-time beat tracking. Section 3 describes the use of this beat tracker in a methodology for beat-synchronous analysis. In section 4 we present an evaluation and discussion of both the beat-tracker and the different methods for beat-synchronous analysis. In section 5 we present our conclusions.

2. BEAT TRACKING

In this section we present our real-time beat tracking model. It is a formed as a hybrid of two existing systems, drawing on the flexibility of Ellis' dynamic programming algorithm [6] for assigning beat locations and the tempo estimation stage of the Davies and Plumbley [7] method.

2.1. Input Feature

The input feature for our beat tracking system is the complex spectral difference onset detection function (DF) [8]; a continuous mid-level representation of an audio signal which exhibits peaks at likely note onset locations. The onset detection function $\Gamma(m)$ at sample m is calculated by measuring the Euclidean distance between an observed spectral frame $X_k(m)$, and a predicted spectral frame $\hat{X}_k(m)$ for all bins k,

$$\Gamma(m) = \sum_{k=1}^{K} |X_k(m) - \hat{X}_k(m)|. \quad (1)$$

Following the approach in [7] we calculate the DF with a temporal resolution of 11.6ms. For a full derivation see [8].

2.2. Beat Prediction

Our underlying model for beat tracking assumes that the sequence of beats, γ_b, will correspond to a set of approximately periodic peaks in the onset detection function. We follow the dynamic programming approach of Ellis [6]. At the core of this method is the generation of a recursive cumulative score function, $C^*(m)$.

[*] This work was supported by EPSRC Grants EP/G007144/1 and EP/E045235/1. AMS is supported by a Doctoral Training Account (DTA) studentship from the EPSRC.

whose value at m is defined as the weighted sum of the current DF value $\Gamma(m)$ and the value of C^* at the most likely previous beat location,

$$C^*(m) = (1-\alpha)\Gamma(m) + \alpha \max_v \left(W_1(v)C(m+v)\right). \quad (2)$$

We search for the most likely previous beat over the interval (into the past) $v = -|2\tau_b|, \ldots, -|\tau_b/2|$ where τ_b specifies the *beat period* – the time (in DF samples) between beats. To give most preference to the information exactly τ_b samples into the past, we multiply C^* by a log-Gaussian transition weighting,

$$W_1(v) = \exp\left(\frac{-(\eta \log(-v/\tau_b))^2}{2}\right). \quad (3)$$

The method for determining τ_b is given in section 2.3.

In terms of parameterisation of (2) and (3), the value of α sets the balance between new information in the onset detection function and existing past information in C^*. The value of η defines the *tightness* of the transition weighting W_1. By default, we set α=0.9 and η=5. We explore the effect of varying α and η in Section 4.

The calculation of $C^*(m)$ is updated at each new detection function sample $\Gamma(m)$, therefore it does not violate our real-time constraint. Ellis' implementation is non-causal because it stores the location of the best previous beat for each sample m and then recovers the beat locations via a recursive *backtrace* once the entire onset detection function has been analysed. For our real-time system we need to predict the locations of future beats in the audio, without the opportunity to observe the complete input signal.

The recursive calculation of the cumulative score function C^* means that it carries some *momentum* where by reliable beat locations (for the non-causal system [6]) can still be found in the presence of arrhythmic playing or silence. To make beat predictions in our causal system we directly exploit the latter property by continuing to generate the cumulative score, C^*, over a one-beat window into the future. Since future information in the onset detection function is unobservable, we ignore its contribution by temporarily setting α=1 in (2), (returning it to its default value once the beat prediction has been made and new DF samples arrive). Each predicted beat γ_{b+1} is made at a fixed point in time m_0 once the current beat γ_b has elapsed, $m_0 = \gamma_b + \tau_b/2$. The predicted beat itself is found as the index of the maximum value over the one-beat window

$$\gamma_{b+1} = m_0 + \arg\max_v \left(C^*(m_0+v)W_2(v)\right) \quad (4)$$

where $v = 1, \ldots, \tau_b$ specifies the future one-beat window and $W_2(v)$ is a Gaussian weighting centred on the most likely beat location $(m_0 + \tau_b/2)$,

$$W_2(v) = \exp\left(\frac{-(v-\tau_b/2)^2}{2(\tau_b/2)^2}\right). \quad (5)$$

Due to the dependence on a previous beat location in (4) the real-time beat tracker must be initialised in some way to find the first beat. In Section 4 we explore the effect on performance of providing an arbitrary first beat and a user-defined initialisation (e.g. from a "count-in"). A graphical example of the beat prediction process is shown in Figure 1. The predicted beat is shown beyond the observed signal.

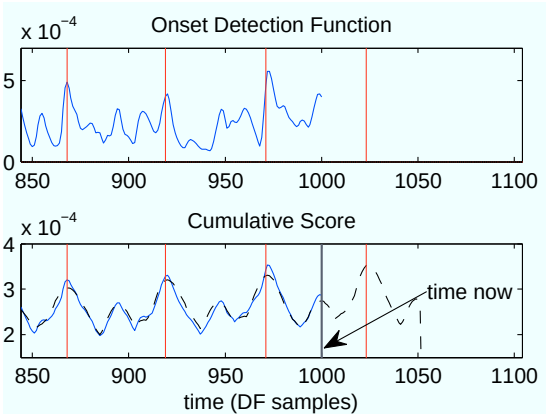

Figure 1: *Top: Onset Detection with predicted beat locations. Bottom: Cumulative score (solid line) with future cumulative score (dotted line). Current time is shown as the bold grey vertical line.*

2.3. Tempo Induction

To be able to track beats in music that varies in speed we need to regularly update the tempo estimate used by the beat tracking stage. In line with the beat prediction methodology, the tempo is re-estimated once each new predicted beat has elapsed.

The approach we adopt to estimating the tempo (and hence beat period τ_b) is based on components from the two state model of Davies and Plumbley [7]. The method can be summarised in the following five steps: i) we extract a six second analysis frame (up to m_0 from (4)) from the onset detection function $\Gamma(m)$; ii) we preserve the peaks in $\Gamma(m)$ by applying an adaptive moving mean threshold to leave a modified detection function $\tilde{\Gamma}(m)$; iii) we take the autocorrelation function of $\tilde{\Gamma}(m)$; iv) we pass the autocorrelation function through a shift-invariant comb filterbank weighted by a tempo preference curve; and v) we find the beat period as in the index of the maximum value of the comb filterbank output, $R(l)$. An example comb filterbank output is shown in the top plot of Figure 2. For a complete derivation of $R(l)$, see [7].

To minimise the common beat tracking error of switching between metrical levels [7] we restrict the range of tempi to one tempo octave from $t_{\min}$=80 beats per minute (bpm) to $t_{\max}$=160 bpm. We map the lag domain signal $R(l)$ into the tempo domain between $t_{\min}$ and $t_{\max}$ to give $R_b(t)$, using the following relationship

$$R_b(t - t_{\min}) = R(|60/(0.01161 \times t)|) \qquad t = t_{\min}, \ldots, t_{\max} \quad (6)$$

where 0.01161 (e.g. 512/44100) is the temporal resolution of the onset detection function in seconds, which is independent of the sampling frequency of the audio. More generally, the relationship between lag (in DF samples) and tempo (in bpm) is

$$l = \left|\frac{60}{0.01161 \times t}\right| \quad (7)$$

Example plots of $R(l)$ and a corresponding $R_b(t)$ are shown in Figure 2.

As in existing work (e.g. [9]) we assume that tempo is a slowly varying process. We enforce some dependence on consecutive tempo estimates by finding the current tempo t_b based on

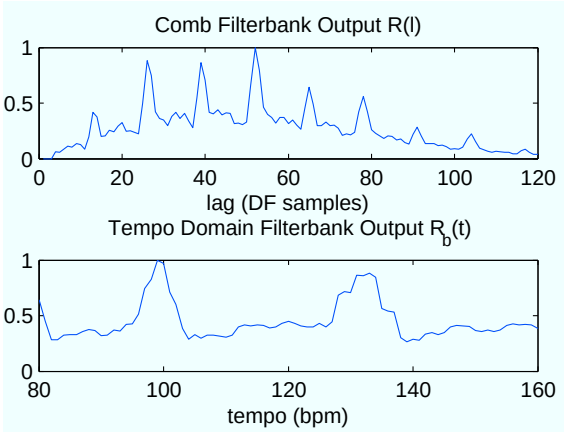

Figure 2: *Top: Comb Filterbank Output $R(l)$. Bottom: $R(l)$ mapped into the tempo domain to give $R_b(t)$.*

the previous estimate t_{b-1}. For this purpose we use a one-step "Viterbi-like" decoding. To model the slowly varying tempo, we use a transition matrix $A(t_i, t_j)$ where each column is a Gaussian of fixed standard deviation $\sigma = (t_{\max} - t_{\min})/8$,

$$A(t_i, t_j) = P(t_b - t_{\min} = t_j | t_{b-1} - t_{\min} = t_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(t_i - t_j)^2}{2\sigma^2}\right) \quad (8)$$

and $t_i, t_j = 1, \ldots, (t_{\max} - t_{\min})$.

At each new iteration, we store the maximum value of the product of each column of A with the stored state probabilities Δ_{b-1} from the previous iteration,

$$\Delta_b(t_j) = \max\left(\sum_{t_j=1}^{t_j=t_{\max}-t_{\min}} A(t_i, t_j)\Delta_{b-1}(t_j)\right). \quad (9)$$

We then update Δ_b to reflect the tempo range comb filter output for the current beat frame R_b by taking the element-wise product of the two signals,

$$\Delta_b(t_j) = R_b(t_j)\Delta_b(t_j). \quad (10)$$

To prevent Δ_b growing exponentially or approaching zero at each iteration we normalise it to sum to unity:

$$\Delta_b(t_j) = \frac{\Delta_b(t_j)}{\sum_{t_j=1}^{t_{\max}-t_{\min}} \Delta_b(t_j)} \quad (11)$$

We then find the current tempo t_b as the index of the maximum value of Δ_b

$$t_b = t_{\min} + \arg\max_{t_j}(\Delta_b(t_j)) \quad (12)$$

and convert it back to beat period τ_b using (7).

3. BEAT-SYNCHRONOUS HARMONIC ANALYSIS

The causal nature of our beat tracking system allows its real-time implementation. In this section we present a model for real-time beat-synchronous harmonic analysis and its use in the implementation of a spectrogram, chromagram and chord detection system.

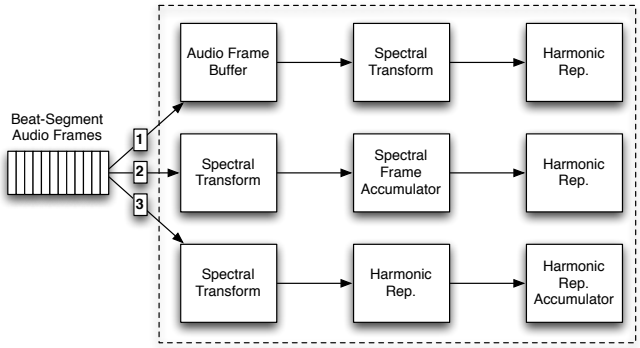

Figure 3: *Three different methods for beat-synchronous harmonic analysis. 'Rep.' stands for 'Representation'.*

3.1. A Model for Beat-Synchronous Harmonic Analysis

We define S, the length of our beat-synchronous segment in audio samples, to be related to the beat period, τ, also in audio samples, by $S = \frac{\tau}{\omega}$ where ω is an integer greater than or equal to 1. This allows us to choose the number of segments per beat and therefore perform analysis at metrical levels lower than that of the beat tracker.

Each segment of length S will contain a number of audio frames, $Q = \lfloor \frac{S}{N} \rfloor$ where N is the length of each audio frame in audio samples. To perform a beat-synchronous analysis, we present three methods which are discussed below.

3.1.1. Method 1

The first method accumulates all the audio from the frames within a beat-segment and then calculates a spectral transform followed by a harmonic representation, such as a chromagram [10]. This can be seen in the first row of Figure 3. A problem with this is that the amount of audio that it is necessary to accumulate varies, from beat to beat, with the tempo. A further difficulty relates to the level of computational complexity. Assuming N is a power of 2, computing the fast Fourier transform (FFT) of a single longer segment of length N requires more calculations than computing Q FFTs of length N/Q. This is demonstrated by:

$$O(N\log(N)) > O(Q \cdot \frac{N}{Q}\log(\frac{N}{Q})) \quad (13)$$

for $Q = 2^y$ and $1 \leq y < r$ for $N = 2^r$. This reduces to:

$$O(N\log(N)) > O(N\log(\frac{N}{Q})) \quad (14)$$

This problem is exacerbated by the fact that all processing is carried out in one step and is not distributed across time as is the case with computing multiple shorter spectral transforms. However, a benefit of this technique is that the larger size of the spectral transform would allow greater frequency resolution for analysis purposes.

3.1.2. Method 2

The second method performs the spectral transform on each frame, accumulates spectral frames and then calculates a harmonic repre-

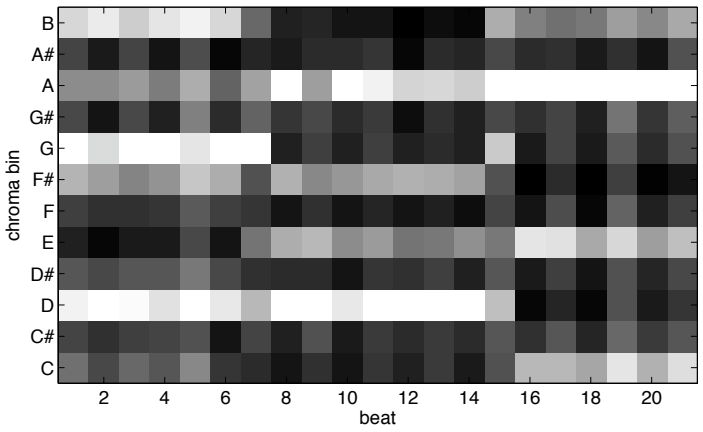

Figure 4: *A beat-synchronous chromagram.*

sentation. This distributes the calculations of the spectral transform across smaller frames and, as was shown in section 3.1.1, is more efficient than computing a spectral transform on the combined frames. Also, by only computing a single harmonic representation, the amount of processor usage is minimised. However, our harmonic analysis algorithm may benefit from several analyses, and so, depending upon the analysis in question, a single harmonic representation may not be as reliable as the accumulation of several over a number of frames.

3.1.3. Method 3

Method 3 calculates both the spectral transform and harmonic representation on each frame and then accumulates the results of the harmonic representations. The difference between methods 2 and 3 is that method 2 uses temporal smoothing of the results of the spectral transforms while the method 3 uses temporal smoothing of the results of several harmonic representations. The preferred method is determined by the nature of the analysis technique in question, given these differences in implementation. It is also possible that, should the harmonic analysis merely involve a summation over spectral bins, methods 2 and 3 will be produce identical results. However, it should be noted that the computation of both a spectral transform and a harmonic representation at each frame is less efficient than the approach of method 2.

3.2. Frame Overlap

An issue arises with some harmonic analysis algorithms as we need a frame size that is large enough to provide sufficient frequency resolution to represent low frequencies. If this frame size is large (some techniques can use a frame size of more than 0.5 seconds [10]) then we are left with very few frames per beat. A solution is to use a larger buffer and a small hop size to increase the number of analyses between beats. However, we have the problem that the overlap may cause audio from one beat to be considered in the next beat. This may contain harmonic information that is dissimilar to the audio we wish to analyse. As a result, we suggest clearing the audio buffer at each beat after the analysis by replacing it with zeros.

3.3. Beat-Synchronous Spectrogram

To calculate a beat-synchronous spectrogram, we calculate each spectral frame f using the Fourier transform:

$$X_f(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{15}$$

for $0 \leq k < N$, where $x(n)$ are the samples of the audio frame and N is the frame size. Then we calculate the Fourier transform for the beat segment, b, by:

$$X_b(k) = \sum_{f=0}^{F-1} |X_f(k)| \tag{16}$$

for $0 \leq k < N$, where F is the number of frames. For method 1, $F = 1$ and for methods 2 and 3 $F > 1$.

3.4. Beat-Synchronous Chromagram

We calculate a beat-synchronous chromagram, $\Phi_b(i)$, using the technique presented in [10], as follows:

$$\Phi_b(i) = \sum_{h=0}^{H-1} \Phi_h(i) \tag{17}$$

where i is the chroma bin index, $i = 0, 1, ..., I-1$ where $I = 12$ and Φ_h is the hth chromagram calculated from H spectral frames. For methods 1 and 2, $H = 1$, while $H > 1$ for method 3. An example beat-synchronous chromagram can be seen in Figure 4.

3.5. Beat-Synchronous Chord Analysis

We implement a beat-synchronous chordal analysis by classifying the beat-synchronous chromagram presented in section 3.4 using the technique presented in [10]. Implementations of all beat-synchronous analysis techniques and the beat tracking model presented in section 2 are available as externals for Max/MSP [1].

4. EVALUATION

4.1. Beat Tracking Performance

We measure the performance of our beat tracking algorithm on an existing annotated database [11] that has been used for the comparison of beat tracking models [7]. The database contains 222 musical excerpts (each about 60s in length) across a wide range of musical styles. We measure performance using the continuity-based evaluation metric as used in [7]. We calculate:

- $\mathrm{CML_c}$: the ratio of the longest continuously correctly tracked section to the length of the file, with beats at the correct metrical level.
- $\mathrm{CML_t}$: the total number of correct beats at the correct metrical level.
- $\mathrm{AML_c}$: the ratio of the longest continuously correctly tracked section to the length of the file, with beats at allowed metrical levels.
- $\mathrm{AML_t}$: the total number of correct beats at allowed metrical levels.

[1] http://www.elec.qmul.ac.uk/digitalmusic/people/adams/bsa/

Beat Tracker	$\mathrm{CML_c}$ (%)	$\mathrm{CML_t}$ (%)	$\mathrm{AML_c}$ (%)	$\mathrm{AML_t}$ (%)
SDP	51.7	58.7	63.3	73.6
SDP+tempo	55.0	65.3	64.0	75.7
SDP+beat	60.6	71.0	64.9	76.5
KEA (NC)	55.7	62.4	70.0	80.0
DP (NC)	54.8	61.2	68.1	78.9

Table 1: Comparison of beat tracking performance. SDP is the default real-time model. SDP+tempo has an initial tempo. SDP+beat has an initial tempo and first beat specified. KEA (NC) and DP (NC) are existing non-causal algorithms.

Beats are considered accurate if they fall within a $\pm 17.5\%$ window around each annotated beat location. Tracking at the correct metrical level means the tempo of the beats and annotations are the same, and the beats are in-phase. The allowed metrical levels permit tracking at twice and half the annotated metrical level and tapping on the off-beat at the correct tempo. For further details see [7].

We evaluate three variants of our beat tracking algorithm: the first, SDP refers to the default initialisation, where an arbitrary first beat is specified; for this we select a time instant 1.5 seconds after the start of each test excerpt. The second variant, SDP+tempo still has an arbitrary beat initialisation, but is given the annotated tempo. The third variant, SDP+beat is given the first annotated beat location *and* the annotated tempo. A summary of results is given in Table 1, where a comparison against the Klapuri et al (KEA) [9] and Davies and Plumbley (DP) [7] non-causal algorithms is also provided.

The results in Table 1 indicate that our real-time algorithm (SDP) is competitive with state of the art non-causal methods, even though our approach must predict beats solely from past data; a constraint not applied to the non-causal methods. Furthermore, when given the initialisation of a first beat and tempo (similar to a "count-in" in musical performance) our beat tracker is able to exceed the state of the art under the strictest evaluation requirement ($\mathrm{CML_c}$). We consider this level of accuracy very encouraging for potential future use in interactive musical performance.

Moving beyond these isolated accuracy values, we also address the robustness of our algorithm in terms of its parameters. We re-evaluate the SDP approach under each continuity-based criterion for $0 \leq \alpha \leq 1$ in (2) and for $1 \leq \eta \leq 10$ in (3). The resulting "accuracy surfaces" are shown in Figure 5.

If α=1, then beat tracking performance under all evaluation measures is zero. This is consistent with (2) where setting α=1 means that no information from the onset detection function is ever incorporated into the cumulative score, and hence no beat locations are predicted. The relatively flat nature of the $\mathrm{CML_t}$ and $\mathrm{AML_t}$ in comparison to the slope in the surfaces of $\mathrm{CML_c}$ and $\mathrm{AML_c}$ suggests that parameter choices can adversely affect the overall accuracy when continuity is required; which for real-time performance is important. By inspection the variation in α leads to greater changes in performance, therefore we believe this parameter to be more influential than η. In future work we intend to explore the adaptive modification of these parameters in real-time, which we consider a potential method for improving accuracy.

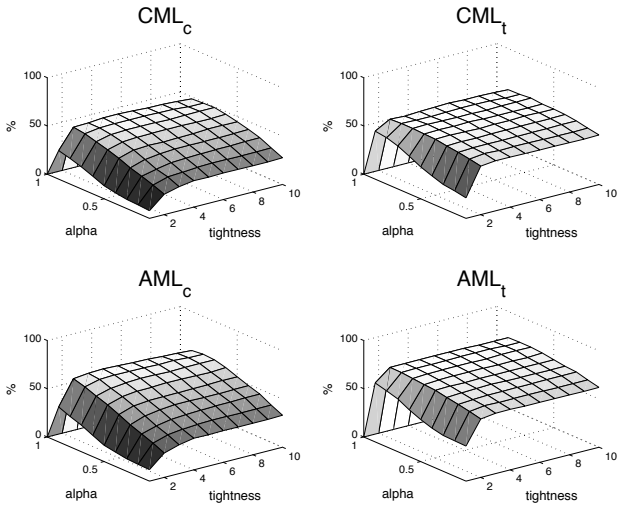

Figure 5: *Beat tracking accuracy surfaces for SDP approach. Clockwise from top left:*$\mathrm{CML_c}, \mathrm{CML_t}, \mathrm{AML_c}, \mathrm{AML_t}$.

4.2. Evaluating Beat-Synchronous Analysis Techniques

We conducted an informal analysis experiment using all three beat-synchronous methods by performing a beat-synchronous chord analysis of a polyphonic guitar performance. The algorithm attempted to label the chord at each beat as one of the 24 major and minor triads. As can be seen in Figure 6, the resulting performance was identical for all three methods. All methods correctly labelled 95.2% of the 105 beats correctly. We compared this with a frame by frame analysis of the same signal using the same chord recognition algorithm. The result was that 91.6% of the 1037 frames were correctly labelled. These preliminary results of the evaluation of the beat-synchronous analysis methods indicate that it improves performance over a frame-by-frame approach, however it is accepted that the results will vary depending upon the style of music. The similarity in performance of the three methods indicate that the choice of the preferred method should be based upon computational complexity, for which method 2 is the cheapest.

It would be desirable to perform a more thorough evaluation of the beat-synchronous analysis technique, but the process of annotating audio examples is difficult and time-consuming. We intend to undertake a more rigorous evaluation with a focus on the real-time aspects in our future work.

Some problems can occur with the compounding of errors occurring in the beat tracker and subsequent analysis algorithms. That is, if the beat tracker performs poorly then this can both exacerbate and cause problems in the harmonic analysis algorithms. For example, if the beat tracker is not calculating beats at the correct locations in the audio signal then this can lead to harmonic content from before and after a harmonic change being incorporated into the same 'beat'. This would make for poor performance in the beat-synchronous chord analysis for example.

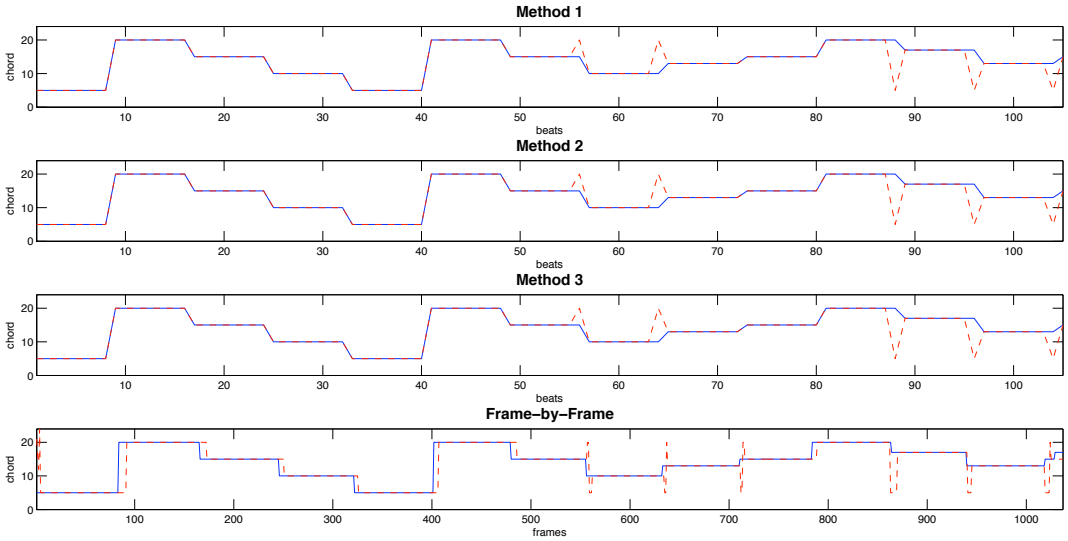

Figure 6: *The chord output of all three beat-synchronous methods is identical, indicating that there is little to distinguish the techniques in terms of their influence on the analysis. The frame-by-frame approach shows a higher percentage of errors than the beat-synchronous methods. The chord labels are 0-11 for C minor -> B minor and 12-23 for C Major -> B Major. The solid line represents the ground truth and the dotted line is the beat-synchronous analysis.*

5. CONCLUSIONS

In this paper we have addressed the topic of beat synchronous analysis towards a real-time interactive musical system. As part of our approach we have formulated a new real-time beat tracking model and have shown its performance to be competitive with state of the art offline systems. Furthermore we have illustrated the potential for beat synchronous analysis to outperform frame-based processing for real-time chord detection. Within our future work we plan to conduct a large scale evaluation of real-time beat synchronous analysis methods addressing both objective/subjective accuracy and computational complexity.

6. ACKNOWLEDGEMENTS

The authors would like to thank Stephen Hainsworth for making his beat tracking test database available for use in our evaluation.

7. REFERENCES

[1] N. Orio, S. Lemouton, and D. Schwarz, "Score following: State of the art and new developments," in *New Interfaces for Musical Expression*, 2003.

[2] D. P. W. Ellis, C. Cotton, and M. Mandel, "Cross-correlation of beat-synchronous representations for music similarity," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 57–60.

[3] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, London, UK, 2005, pp. 304–311.

[4] M. Levy and M. Sandler, "Structural segmenation of musical audio by constrained clustering," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 156, no. 2, pp. 318–326, 2008.

[5] N. Schnell, D. Schwarz, and R. Müller, "X-micks - interactive content based real-time audio processing," in *Proceedings of International Conference on Digital Audio Effects*, 2006.

[6] D. P. W. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[7] M. E. P. Davies and M. D. Plumbley, "Context-dependent beat tracking of musical audio," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 3, pp. 1009–1020, 2007.

[8] 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, 2004.

[9] A. P. Klapuri, A. Eronen, and J. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.

[10] A. M. Stark and M. D. Plumbley, "Real-time chord recognition for live performance," in *Proceedings of International Computer Music Conference*, 2009, To appear.

[11] S. Hainsworth, *Techniques for the Automated Analysis of Musical Audio*, Ph.D. thesis, Department of Engineering, Cambridge University, 2004.