B-Keeper: A Beat-Tracker for Live Performance

Andrew Robertson
Centre for Digital Music
Department of Electronic Engineering
Queen Mary, University of London
andrew.robertson@elec.qmul.ac.uk

Mark Plumbley
Centre for Digital Music
Department of Electronic Engineering
Queen Mary, University of London
mark.plumbley@elec.qmul.ac.uk

ABSTRACT

This paper describes the development of **B-Keeper**, a real-time beat tracking system implemented in Java and Max/MSP, which is capable of maintaining synchronisation between an electronic sequencer and a drummer. This enables musicians to interact with electronic parts which are triggered automatically by the computer from performance information. We describe an implementation which functions with the sequencer Ableton Live.

Keywords

Human-Computer Interaction, Automatic Accompaniment, Performance

1. INTRODUCTION

In this paper, we describe an interactive system which enables synchronisation between electronic parts from an audio and Midi sequencer and a drummer. Previous automatic accompaniment systems, such as Chris Raphael's *Music Plus One*[7] and IRCAM's *Suivi* [6] successfully performed the task of score following for a monophonic instrument with a classical score. Our system, B-Keeper, addresses the problem of automatic accompaniment for contemporary rock music by using input from percussive instruments, in particular the signal from the kick drum. Previous beat trackers [4] [3] estimate the tempo and beat positions. This system is specifically designed for live performance, incorporating specialised functions and adjustable parameters to customise it for individual pieces.

B-Keeper is an interactive system in the sense that it responds to input from the user in order to generate an output, but also it is interactive in a wider sense since there is a musical interaction that takes place during performance. The drummer is free to to set and change the tempo, whilst also responding to the parts played by B-Keeper. These parts are currently pre-arranged since we have concentrated on the problem of real-time beat tracking and synchronisation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NIME07, New York, USA
Copyright 2007 Copyright remains with the author(s).

Figure 1: Performing with B-Keeper at LAM

Our aim in designing this system is to create a basis for the development of larger musical interactive systems which can allow musicians to make use of the audio processing and event scheduling capabilities of the computer within live performance, whilst preserving and even enhancing the freedom of interaction they are used to.

Our initial use of the system has been for automatic accompaniment in order to test the ability of the system to perform accurate synchronisation in real-time. By implementing the system in Max/MSP, we are able to make use of the structural information provided by B-Keeper when designing performance systems for drums and other instruments.

2. PHILOSOPHY: FOR PERFORMANCE

Often, in order to integrate electronically generated parts into live performance, bands are forced to make the concession of having the drummer wear headphones and play to a pre-generated click track. This limits the dynamic possibilities of the piece since the tempo must remain constant. Another alternative is the triggering of such parts manually using MIDI sensors.

B-Keeper has been developed to enable a computer to interact musically with a contemporary rock band without such compromises.

We use the audio signal from a microphone in the kick drum as input and output a continuously varying tempo which is fed to a software sequencer. In performance, we

have used Ableton's *Live* [1] which offers time-stretching capabilities and MIDI and audio sequencing in real-time. We provide adjustable parameters such as thresholds and variables which alter the shape of functions used within the processing of the data so that the performer is able to control the responsiveness of the algorithm.

The nature of a real-time performance system places an emphasis on adaptability, reliability and responsiveness rather than the ability to perform the task independently on a wide-range of styles. It is important for the beat tracker to be accurate in its synchronisation as any errors will manifest themselves audibly to the audience. It is also permissable to add functionality for the user to aid the system in its judgement and vary parameters during the performance to guide the system.

Different sections of a song might be played in a specified order with cues given for the transition from one section to the next. The system need not use the same cues as humans, for instance eye contact or physical gesture, but the idea that such information could be communicated, perhaps means of a trigger or prior agreement, is useful and important when designing the system for live performance.

3. IMPLEMENTATION

In keeping with our performance philosophy, the following assumptions were made to simplify the design of the Beat-tracker:

- The piece has an approximately steady tempo which may fluctuate but does not change suddenly by any considerable degree.
- An approximation of the initial tempo is known before the performance begins.
- The bass drum signal is rhythmic and onsets are intended to occur on metrical divisions of the beat. Hence, timing deviations are assumed to be the result of human error, not deliberate stylistic expression. [1]

These assumptions have been made with a view to using B-Keeper in the rock, pop and dance genres, with the intention of extending the capabilities of the system at a later stage. The assumption that an approximation to the initial tempo is known beforehand is made since it is a reasonable assumption which simplifies the task (and hence the algorithm) and is the case for many performances involving human musicians, since they gain an idea of the tempo from rehearsals.

B-Keeper has been programmed in Max/MSP and Java. In order to synchronise the sequencer with the drummer, a constant MIDI click is sent from Live to Max/MSP, which we map to onsets from the kick drum. The system maintains an estimate for the underlying tempo of the piece and also synchronises the MIDI click as close as possible to strong onsets, equivalent to matching the correct phase.

3.1 Tempo Tracking

The smallest perceptual time unit in a rhythmic signal is known as the *tatum*. The term represents the 'temporal atom' [2] and refers to the high frequency pulse that listeners infer from the timing of rhythmic events and is the lowest metrical level of a piece of music.

B-Keeper works by detecting the tatum interval which defines the underlying metrical structure of the kick drum's rhythmic pattern. Although there may exist timing deviations in the playing of particular beats, it is assumed that the onsets are intended to occur on the beat. The timing of all other musical events can be described relative to the metrical structure deduced from the kick drum pattern.

The input is processed by the onset detector, *bonk* [5], and put through a threshold so that only strong onsets are processed. When an onset is detected, we calculate the inter-onset interval (in milliseconds) between the new onset and other recent onsets. The interval which corresponds closest to an expected regular metrical division of the beat results in an update of our expected tempo.

Given an existing estimate for the current tempo, τ, the beat tracker updates its estimate every time it detects an onset from the kick drum. This uses the following method:

1. Find the current CPU time and calculate the inter-onset interval in msecs for recent onsets.
2. Categorise each interval as lasting for a regular metrical division (e.g. one and a half beats) and calculate the performance error according to that categorisation.
3. Find the winning inter-onset interval which most strongly supports the current tempo estimate.
4. If its accuracy exceeds our threshold, update the tempo towards this new value.

In step 2, the user is able to bias the system towards metrical divisions which are more common within the user interface. Typically this would be onsets separated by an even number of tatum intervals, such as a beat or a bar.

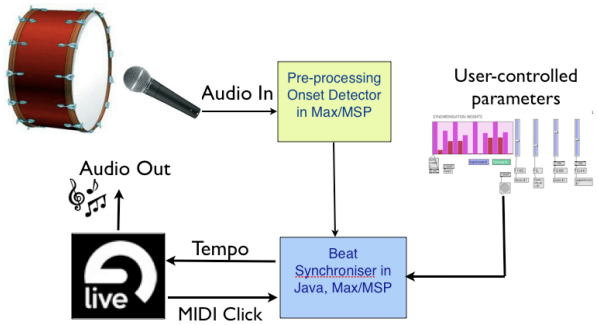

Figure 2: System Architecture

3.2 Synchronisation Method

When an onset occurs close to a regular metrical division, the system makes an extra adjustment of the tempo in order to synchronise more accurately with the onset. This is equivalent to adjusting the phase of the beat tracker.

In order to quantify the accuracy of an onset relative to an expected onset time, we used a Gaussian. [2]

[1]Syncopation (the accenting of a normally unstressed beat) is accounted for by setting synchronisation weights for each different beat of the bar.

[2]The onset time is variable due to the limitations of human coordination and timing. We will model this using a simple Normal distribution (Gaussian or bell-shaped curve) as seen in Figure 3.

For a new onset at time x_n, the Gaussian is given by:

$$G(x_n, E[x_n]) = e^{\frac{-(x-E[x_n])^2}{\sigma^2}}$$

The expected onset time, $E[x_n]$, is the CPU time of the closest MIDI click received from Ableton Live. These will occur at regular metrical divisions of the beat in integer multiples of the tatum from the bar boundary.

We ascribe a weight to the metrical position of each MIDI click which represents the likelihood of an onset occuring at that position in the bar. Typically, onsets are more likely for the *'One'* (zero tatums from the bar boundary) or the *'Three'* (four tatums from the boundary) and less likely a half-beat in. Hence, we can multiply by our likelihood function:

$$L[x_n] \in [0, 1].$$

This is a simple probability measure set by the user via a graphical interface in Max/MSP. In practice, it can be used to interesting effect within performance and determines which bar positions are used by the algorithm in order to synchronise to the sequencer.

Given a synchronisation threshold, ξ, we update according to the following rule:
If $G(x_n, E[x_n]).L(x_n)$ is greater than ξ, then the tempo estimate is adjusted by a proportion of the error, $x_n - E[x_n]$, in order to synchronise the click track to the new onset.

4. SUPERVISOR-ADJUSTABLE PARAMETERS

There are several parameters set by the supervisor (the user controlling B-Keeper in a performance), which determine the system's behaviour. The advantage of incorporating these features is to enable the supervisor to find the best settings for the performance of individual pieces. These are:

- An initial starting tempo must be set, giving the value for the tatum, τ.
- A scalar α determines the responsiveness of the system to changes in tempo. It may be desirable that the system respond quickly to tempo changes in a piece, whereas by using a smaller value of α, the system would maintain a steadier underlying tempo but be less responsive.
- Values for the update thresholds are set by the supervisor. The higher the threshold, the greater the playing accuracy required by the system in order for it to interpret the onsets as relevant to its tempo hypothesis. These can be set to *self-adjust mode*, so that they constantly update to values that match the player's accuracy.
- The size of the Gaussian window in which the system looks for new onsets is determined by the standard deviation σ for both the tempo and synchronisation procedures. The user is also able to set these to self-adjust mode, so when the drummer plays a steady beat accurately, the window narrows to increase the precision of the computer's synchronisation (illustrated in Figure 3). When the timing varies relative to the underlying tempo, then the window widens in order to maintain synchronisation with the fluctuations of the beat.
- Synchronisation weights, corresponding to a bass drum pattern, are specified which bias the system to synchronise onsets to a particular beat within the bar. Similarly, a bias towards certain more regular inter-onset intervals is set using inter-onset weighting within the tempo tracking process.

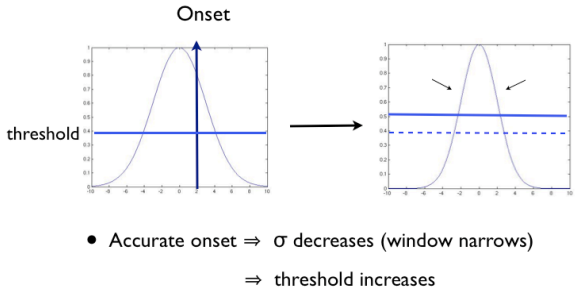

Figure 3: Self-Adjusting Window

In particular, by selecting the self-adjust mode for some parameters, the system seeks out a state of equilibrium, independent of the starting state, that is appropriate for the style of playing of that particular musician and piece.

5. ADDITIONAL MANUAL CONTROLS

An automatic system is never perfect and for performance, we have found it necessary to incorporate features that enable a human controller to guide the system in its synchronisation and on occasion, rescue it from erroneous judgements.

The way that the beat-tracker is designed means it relies on being approximately correct in order to rectify any timing differences between the computer and performer. Occasionally, it might stray too far from the beat and begin to synchronise with the performer a beat or a half-beat out. We have two options: *either* (a) the drummer re-adjusts to the B-Keeper *or* (b) a supervisor makes use of manual controls, such as the *nudge* function to bring the B-Keeper back in line.

In practice, we have made use of both available options. The nudge function works by calculating the amount it needs to slow down or speed up by in order to synchronise with the adjacent metrical beat. B-Keeper makes the appropriate tempo change to accomplish this over the next few beats. The system displays where synchronisation is taking place within the bar and so the supervisor can tell when intervention is necessary.

There is also a potential problem of latency if there is a discrepancy between the time it takes for the MIDI click signal to arrive in Max/MSP from Live and the time for a kick onset to be detected. We have incorporated a latency adjustment which can be used to counteract such an effect.

The human controller can take part in the performance by introducing parts and samples or controlling audio effects and other parameters such as volume or panning. It is also possible to record live audio or MIDI parts which can be immediately looped in Live. These can then be replayed at later points in the piece, whilst B-Keeper makes sure they are still synchronised with the drums.

6. CURRENT WORK

We are currently extending the system within Max/MSP. Having implemented the means to follow the metrical structure of the drum pattern in real-time, we are now investigating ways to process other performance data, such as harmonic information from other instruments, within that framework. This enables us to learn from previous performances, since musical events happen relative to the global structure.

By mapping the abstract structure of the piece, the system can record performance data, deduce appropriate synchronisation weights from rehearsals and automate event sequencing, from the large scale transition changes to modulation of parameters which control sound. The data structure makes it possible to implement mean-corrected event analysis, used effectively in Barry Vercoe's score following system *Synthetic Performer* [8]. We are also researching the integration of creative algorithms that make use of performance data when responding to the musicians.

7. RESULTS

7.1 Performance at LAM

We have performed with B-Keeper at the Live Algorithms for Music (LAM) conference at Goldsmiths College, University of London, on 19th December 2006.

David Nock, professional audio engineer and drummer, performed with the system, playing a piece composed for the occasion. Entitled *'Ode to Jack Kerouac'*, it has a linear structure and is in 4/4 time. The drums begin the piece, triggering B-Keeper to start the sequencer, which immediately begins playing a synthesized bass-line. Other prearranged samples, including a spoken-word extract, a violin motif and electronic percussion, are introduced at will by the controller. B-Keeper maintains synchronisation with the drummer, who is free to change tempo and vary the rhythm whilst interacting with the parts that are playing.

7.2 Performance Experience

Human intervention was required during the performance. The nudge function was used by the supervisor when it became apparent that the system had synchronised a beat out from the drums - the situation which this function is intended to remedy.

However, once B-Keeper had found the correct synchronisation, it was possible to focus on the musical process of bringing in the respective parts. Fills and complex drum patterns can present difficulties for the system unless its synchronisation is already accurate. This can be explained by the fact that in a fill we have a fast succession of onsets, so if the synchronisation is imperfect, the likelihood of an onset being mis-interpreted is higher.

David Nock, the drummer and collaborator for the performance, described the experience of playing with the system. Despite early doubts as to whether the system *"was following him or he was following it"*, he began to gain a sense of how it responded. *"It has elements of a human - almost an early learning one - you have to guide them through, accomodate them a little."* He described the interaction as *"very good. Once it gets in sync, you can feel where its boundaries are. It's able to accelerate and deaccelerate. Like with a real player, it's a two way process."*

An excerpt of this performance can be viewed on the internet at
`http://www.elec.qmul.ac.uk/digitalmusic/b-keeper/`.

8. CONCLUSION

We have designed B-Keeper, a real-time beat-tracking system for drums in live performance. We have taken a pragmatic approach, allowing the user to change many parameters which affect its behaviour, thereby customising the system for individual pieces. It can synchronise with Ableton Live, enabling use of Ableton's powerful time-stretching algorithms, virtual instruments and audio sequencer for automatic accompaniment. David Nock, the drummer who performed live with B-Keeper, reported a strong sense of *musical interaction* as the system responded to his playing.

9. ACKNOWLEDGMENTS

AR is supported by a studentship from EPSRC. The authors would like to thank Nick Bryan-Kinns for his help and advice on this research. We would also like to thank David Nock for performing with B-Keeper at LAM and for his feedback on the experience.

10. REFERENCES

[1] `http://www.ableton.com`.

[2] J. Bilmes. Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1993.

[3] M. Davies and M. D. Plumbley. Beat tracking with a two state model. In *Proc. IEEE International Conference on Acoustics, Speech and Signal In Processing (ICASSP), Philadeliphia, USA*, volume 3, pages 241–245, 2005.

[4] M. Goto. An audio-based real-time beat tracking system for music with or without drum sounds. *Journal of New Musical Research*, 30:159–171, 2001.

[5] T. A. M. Puckette and D. Zicarelli. Real-time audio analysis tools for Pd and MSP. In *Proc. International Computer Music Conference, San Francisco.*, pages 109–112, 1998.

[6] N. Orio, S. Leemouton, and D. Schwarz. Score Following: State of the Art and New Developments. In *Proc. of 2003 Conference on New Interfaces for Musical Expression (NIME), Montreal, Canada*, pages 36–41, 2003.

[7] C. Raphael. Synthesizing Musical Accompaniments with Bayesian Belief Networks. *Journal of New Musical Research*, 30(1):59–70, 2001.

[8] B. Vercoe and M. Puckette. Synthetic Rehearsal, Training the Synthetic Performer. In *Proc. International Computer Music Conference (ICMC)*, pages 275–278, 1985.