Audio Engineering Society

Convention Paper

Audio Effects for Real-Time Performance Using Beat Tracking

A.M. Stark[1], M.D. Plumbley[1], and M.E.P. Davies[1]

[1]*Centre for Digital Music, Queen Mary, University of London, Mile End Road, London E1 4NS, UK*
http://www.elec.qmul.ac.uk/digitalmusic

Correspondence should be addressed to Adam Stark (`adamstark@hotmail.co.uk`)

ABSTRACT

We present a new class of digital audio effects which can automatically relate parameter values to the tempo of a musical input in real-time. Using a beat tracking system as the front end, we demonstrate a tempo-dependent delay effect and a set of beat-synchronous low frequency oscillator (LFO) effects including auto-wah, tremolo and vibrato. The effects show better performance than might be expected as they are blind to certain beat tracker errors. All effects are implemented as VST plug-ins which operate in real-time, enabling their use both in live musical performance and the off-line modification of studio recordings.

1. INTRODUCTION

Audio effects are standard tools for musicians, both in the process of studio recording and during live performance. It is often desirable for musicians to use audio effects in such a way that the resulting effect is related in some way to the tempo of the piece. An example would be setting the rate parameter of a tremolo so that it modulates the signal in time to the piece. Use of effects in this way is currently possible using external metadata, such as a MIDI click track or data from a tap-tempo pedal, to provide information about the timing of beats in the performance. While these techniques can be effective,

they either restrict the musician to a set tempo or require the musician to indicate any tempo changes during the performance.

This paper presents audio effects that employ *beat tracking*, the detection of beats in an audio signal, to control effect parameters and thereby create effects that can automatically adjust themselves to tempo changes in the signal. Based upon this principle, we present a tempo-synchronous delay effect that is able to delay audio by a number of beats, rather than a number of milliseconds. We then introduce beat-synchronous low frequency oscillator (LFO) based effects including tremolo, auto-wah

and flanger that synchronise the rate of a LFO to the beats in the input signal. The principles of all conventional implementations of the audio effects in this paper can be found in Zölzer [2, p31-92].

As the intended use of the effects is in live performance, all effects presented here are designed to be causal and implemented in real-time as software plug-ins. The audio sample rate used is 44.1kHz for all effects.

The effects presented here vary their performance depending upon changes in the input signal. Previous work on such effects include compressors and noise gates [2, p93-136] that alter the output level of a signal based upon the input level. Verfaille et al. [3] present a number of audio effects that extract features from the input audio and use them to control effect performance. In one example, an adaptive tremolo is presented that uses the fundamental frequency and sound intensity level to control, respectively, the rate and depth of the effect. The effects presented here are similar in principle to this previous work, however, we focus specifically upon the extraction of rhythmic information to control audio effect parameters.

This paper is structured as follows. In section 2 we provide a brief overview of the real-time beat tracking system. Sections 3 and 4 describe the process of augmenting conventional audio effects using the beat tracker. Section 5 describes possible applications of the effects described in previous sections. The performance of the effects is evaluated in section 6 before conclusions are made in section 7.

2.   A REAL-TIME BEAT TRACKING SYSTEM
The audio effects described in this paper employ an implementation of the beat tracking system of Davies et al. [1]. We select this system due to its comparable performance to state of the art beat tracking systems, causal design and low computational complexity. The beat tracker calculates the *beat period*, the amount of time between beats, and the *beat alignment*, a reference point in the signal from which future beats can be predicted. The following is a discussion of the issues arising from the real-time implementation of the beat tracker and the implication of these issues for the audio effects that employ it.

The beat tracking system in [1] uses an analysis buffer of 262144 audio samples (~6s) overlapping by a 65536 audio sample (~1.5s) *analysis frame*. The analysis buffer
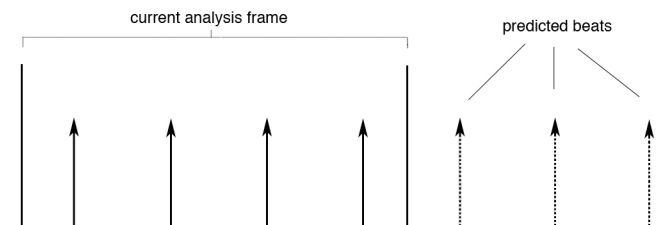

Fig. 1: A ~1.5s analysis frame containing 4 beats (shown by solid arrows) and part of a future analysis frame during which 3 beats have been predicted (shown by dotted arrows).

is used to make predictions of beat locations in the next ~1.5s analysis frame (see Figure 1). Consequently, a new estimate of the *beat period* and *beat alignment* is made approximately every 1.5 seconds enabling changes in tempo to be detected.

3.   TEMPO-SYNCHRONOUS AUDIO EFFECTS
Tempo-Synchronous audio effects do not require beat alignment information as they only make use of beat period information to automate parameter values. In this section we describe a tempo-synchronous delay effect.

3.1.   Tempo-Synchronous Delay
A delay effect is designed to create an 'echo' by delaying the input signal by a certain amount of time. A conventional delay effect uses a user-defined *delay time* parameter, represented by a number audio samples. The output signal, *y[n]*, is calculated by [3]:

$$y[n] = x[n] + \alpha \cdot x[n - Q] \qquad (1)$$

where *x[n]* is the input signal, Q is the number of audio samples by which the signal is delayed and α is the gain factor that controls the amplitude of the delayed signal.

To adapt this effect to make it tempo-synchronous, we replace the parameter Q with a value related to the beat period, τ:

$$y[n] = x[n] + \alpha \cdot x[n - (\lambda \cdot \tau)]. \qquad (2)$$

x[n] is the input signal, α is the gain factor that controls the amplitude of the delayed signal, τ is the beat period in

samples provided by the beat tracker and λ is the number of beats by which the user wishes to delay the signal.

3.2. Implementation

The tempo-synchronous delay effect is implemented in real-time using a circular array to store delayed audio samples. The value of the beat period provided by the beat tracker is re-estimated at the end of each audio analysis frame. As a result, variations in tempo can cause sharp changes in the length of the delay time. Therefore, phase mismatches in the delayed audio can occur that can be heard as unpleasant artefacts. To ensure that there are no discontinuities in the audio when the beat period is re-estimated (every ∼1.5s), the old and new delay outputs are crossfaded over 512 audio samples.

4. BEAT-SYNCHRONOUS AUDIO EFFECTS

Beat-Synchronous audio effects make use of both beat period and beat alignment information. In this section we present methods for correctly synchronising the rate of a low frequency oscillator (LFO) to the beats in an input signal. We then see the use of this beat-synchronous LFO to adapt conventional tremolo, auto-wah and flanger audio effects.

4.1. A Real-Time Beat-Synchronous Low Frequency Oscillator

Beat-synchronous low frequency oscillators can be divided into two types. A *Type I* beat-synchronous LFO has one or more cycles per beat. *Type II* beat-synchronous LFOs have cycles that last for multiple beats. In general, for a LFO to be beat-synchronous, either each beat must coincide with the end of a cycle or each cycle must end on a beat. Therefore, we need the LFO to begin on a beat and to ensure that the length of each LFO cycle is related to the beat period by a number of *cycles per beat*, Ω, where $\Omega = N$ or $\Omega = 1/M$, where N and M are positive integers. When an LFO is started at a beat location, the length of a LFO cycle, $\mathcal{T}$, in audio samples is calculated as:

$$\mathcal{T} = \frac{\tau}{\Omega} \qquad (3)$$

where τ is the beat period provided by the beat tracker.

In order to create a general algorithm, independent of any specific LFO waveform, we wish to represent the LFO as a phase value, allowing us to apply functions such as
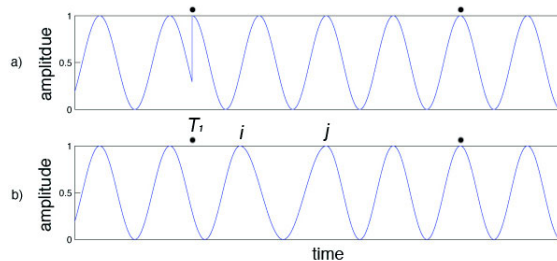

Fig. 2: a) A LFO waveform where a late beat has caused the LFO to start part of an extra cycle which has caused a phase mismatch when the first beat is reached (represented by the first black dot). b) A LFO waveform with the same situation but implementing our algorithm for smoothly locking to the new beats. Note that the waveform returns to normal 2 cycles after the first beat.

cos or *sin* to create more complex waveforms. Therefore, each cycle of the LFO is represented by a value $c[n]$ where $0 \leq c[n] < 1$. $c[n]$ climbs from 0 to 1 over $\mathcal{T}$ steps:

$$c[n] = \text{frac}(c[n-1] + \eta) \qquad (4)$$

where $\eta = 1/\mathcal{T}$ and frac$(\cdot)$ returns the fractional part of its argument. The use of frac$(\cdot)$ ensures that $c[n]$ only takes values between 0 and 1. The current phase may then be calculated by $\theta = 2\pi \cdot c[n]$.

4.1.1. Implementation of Type I Beat-Synchronous LFOs

Type I Beat-Synchronous LFOs have one or more cycles per beat. To implement Type I LFOs we might recalculate the cycle length at each beat according to equation (3). However, as beat predictions are made at the start of each audio analysis frame, at the last beat of each frame we do not know when the next beat will occur. Therefore we cannot calculate the cycle length to ensure that the LFO reaches the end of a cycle at the next beat. The result is a phase mismatch in the LFO waveform as the oscillator value is reset at the first beat (see Figure 2(a)). This behaviour is similar to that of the Phase-Reset Oscillators presented by Arora and Sethares in [4]. We shall therefore refer to this problem as the *Phase-Reset Problem*.

To prevent discontinuities due to tempo changes at the end of a frame we add an intermediate cycle to smoothly

synchronise the LFO with the new beats. We first identify the sample location, i, of the first oscillator cycle ending after the first beat. We then identify the point j, the location of the first 'correct' (i.e. had the LFO cycle been started at the first beat at the new frequency) cycle ending of the LFO after the point i. The point j is calculated by:

$$j = T_1 + (1 + \lfloor \frac{i - T_1}{\mathscr{T}} \rfloor) \cdot \mathscr{T} \qquad (5)$$

where T_1 is the sample location of the first beat and $\mathscr{T}$ is the cycle length for the current frame as defined in equation (3). In order to avoid a very short and unnatural intermediate cycle, we reassess j by:

$$j = \begin{cases} \tilde{j} + \mathscr{T}, & \text{if } (\tilde{j} - i) < \frac{\mathscr{T}}{2} \\ \\ \tilde{j}, & \text{otherwise} \end{cases} \qquad (6)$$

where $\tilde{j}$ is the value of j calculated using equation (5) and $\mathscr{T}$ is the length of the LFO cycle according to the new beat period. Finally, a single cycle is created between the point i and the point j of length $(j - i)$. For this cycle, the increment η, of $c[n]$, is calculated as:

$$\eta = \frac{1}{(j - i)}. \qquad (7)$$

When this cycle ends, the cycle length, $\mathscr{T}$, will be calculated by equation (3) and the LFO will be correctly synchronised with the predicted beats. Figure 2(b) shows a waveform overcoming the problems of 2(a) using this method.

4.1.2. Implementation of Type II Beat-Synchronous LFOs

Type II LFOs have cycles that last for many beats, therefore, the simplest implementation of Type II LFOs would be to simply recalculate the cycle length and start a new cycle after M beats, where M is the desired length of the cycle in beats. However, as there are many beats per cycle, tempo changes make it very unlikely that the cycle will end on the beat it is supposed to. Most likely, there will be a phase mismatch similar to the Type I mismatch seen in Figure 2(a).

In order to overcome this, we adjust the LFO mid cycle, at each beat, according to the current beat period, so that the LFO will still end on the correct beat. To achieve this, we first calculate δ, the number of samples remaining in the cycle:

$$\delta = \mathscr{T} - (\tau \cdot \beta) \qquad (8)$$

where $\mathscr{T}$ is the length of the cycle, β is the number of beats remaining in the cycle (including the current beat) and τ is the beat period provided by the beat tracker. Note that if the cycle is to end exactly on the correct beat, with δ samples remaining, the value of $c[n]$ should be:

$$c[n] = \frac{\delta}{\mathscr{T}}. \qquad (9)$$

However, tempo changes are likely to make this not the case. Using the value of δ, at each beat we can recalculate the increment value, η, of $c[n]$ to compensate for changes in tempo:

$$\eta = \frac{(1 - c[n])}{\delta} \qquad (10)$$

where $c[n]$ is the current LFO value and δ is the number of remaining samples in the cycle. This recalculation will increase or decrease the frequency of the LFO, depending upon its current phase position, to cause it to end on the correct beat.

There is a single case where Type II LFOs suffer from the Phase-Reset Problem. When a Type II LFO has a cycle that ends on the first beat of an analysis frame, then we have the problem that at the last beat of the preceding frame, we cannot know when the next beat will occur and cannot guarantee it will end exactly on the beat. As a result there is likely to be a phase mismatch at the first beat of the frame. We have two separate cases: that the LFO finishes before the beat (Figure 3(a)) and that the LFO finishes after the beat (Figure 3(b)). To correct these problems, we must adjust the LFO at the first beat of the new frame. We first calculate the value, c_2, the 'correct' value of $c[n]$ at the second beat:
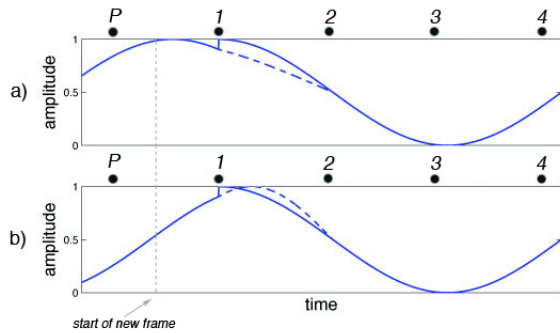
Fig. 3: Cycles that have ended before (a), and after (b), the first beat. Beats are marked by black dots. The solid line shows what the LFO would have done if it was simply restarted at the first beat, due to a lack of knowledge at the final beat of the preceding frame (marked by P). Between beats 1 and 2, the dotted line shows how the LFO adjusts itself to meet the solid line at the second beat and thus avoids a phase mismatch in the LFO.

$$c_2 = \frac{1}{M} \qquad (11)$$

where M is the number of beats in each cycle specified by the user. If the LFO has finished early and begun a new cycle (see Figure 3(a)), then between beats 1 and 2, we set the increment, η, of $c[n]$ to be:

$$\eta = \frac{c_2 - c_1}{\tau} \qquad (12)$$

where c_1 is the value that $c[n]$ has reached at the first beat and τ is the beat period provided by the beat tracker.

If the LFO has not finished its cycle before the first beat occurs (see figure 3(b)), then we must allow it to complete it's cycle and reach the correct point at the second beat. As a result, we instead calculate η by:

$$\eta = \frac{c_2 + (1 - c_1)}{\tau}. \qquad (13)$$

Using these adjustments, the LFO will smoothly reach the point at the second beat that it would have reached had it been started at the first beat. At the second beat,

adjustments will be made according to equation (10) and the LFO will return to normal operation.

4.2. Beat-Synchronous LFO-based Audio Effects

By making use of the beat-syncronous LFOs described in section 4.1, we can adapt audio effects that make use of a LFO to be beat-synchronous. We can create a beat-synchronous modulating LFO waveform, $m_\Omega[n]$, operating at Ω cycles per beat, by applying common functions, such as cos to the phase value $c[n]$, for example:

$$m_\Omega[n] = \frac{(\cos(2\pi \cdot c[n]) + 1)}{2}. \qquad (14)$$

4.2.1. Beat-Synchronous Tremolo
We can create a beat-synchronous tremolo by:

$$y[n] = x[n] \cdot m_\Omega[n] \qquad (15)$$

where $y[n]$ is the output signal, $x[n]$ is the input signal and $m_\Omega[n]$ is a Type I beat-synchronous modulating waveform operating at Ω cycles per beat.

4.2.2. Beat-Synchronous Auto-Wah
We can create a beat-synchronous auto-wah by filtering the input signal using a bandpass filter and modulating the cutoff frequency f_c using the beat-synchronous waveform:

$$f_c = f_b + (m_\Omega[n] \cdot f_r) \qquad (16)$$

where f_b and f_r are respectively the base cutoff frequency and sweep range, in Hz, of the bandpass filter. $m_\Omega[n]$ is the beat-synchronous modulating waveform operating at Ω cycles per beat. A Type I or Type II LFO could be used depending upon preference.

4.2.3. Beat-Synchronous Flanger
We can create a beat-synchronous flanger by using the modulating beat-synchronous waveform to control a variable delay line of length D:

$$D = m_\Omega[n] \cdot T_{max} \qquad (17)$$

where T_{max} is the maximum delay time (usually ~ 2 ms) and $m_\Omega[n]$ is a Type II beat-synchronous LFO operating at Ω cycles per beat. The effect is then implemented as follows:

$$y[n] = x[n] + \alpha \cdot x[n - D] \qquad (18)$$

where we choose $\alpha = 0.7$.

5. APPLICATION OF EFFECTS

The development of audio effects that adapt to changes in tempo of the input signal allow musicians to easily create many interesting effects. The tempo-synchronous delay can be used to delay a signal by a single beat, an effect, which if used upon an arpeggio, will cause each note to fall upon and harmonise with the next. The effect can also be used to delay a signal by a fraction of a beat. If used with feedback of the output into the delay line, this can cause a 'rhythmic' echo effect as each note repeats and decays in time to the music.

The beat-syncrhonous tremolo can be used to cause an instrument to have a very rhythmic sound as the effect appears to 'pulse' in time to the music. If the effect is set to operate at 3 cycles per beat, then it can give each note a 'triplet' feel where each note contains 3 'pulses' which can make a piece feel very different to how it would sound without the effect.

6. EVALUATION

The beat tracking system has been tested and evaluated (see table 1 and [1]), however, evaluating the performance of the effects described in this paper is a difficult and imprecise task. There are two reasons for this. Firstly, the performance of these effects is inherently dependent upon the performance of the beat tracker. Secondly, the result of the application of these effects is highly subjective to the listener. Therefore, to evaluate the effects, we combined an informal subjective assessment of the effects with an analysis of their robustness to beat tracker errors.

Beat predictions from the beat tracker can be simply classified as either *correct* or *incorrect*, as compared to human annotated perceived beat times. However, it is more revealing to sub-catagorise errors as our experience has shown that many errors are related to the correct beats in some way. We will consider the following types of

Beat Tracker Predictions	(%)
Correct	**78.2**
Offbeat	6.4
Half/Double Tempo	5.2
Incorrect	10.2
Audio Effect Performance	(%)
Delay	**84.6**
Type I LFO-based (even Ω)	**84.6**
Type I LFO-based (odd Ω)	**78.2**
Type II LFO-based	**78.2**

Table 1: The top section of the table shows the performance of the beat tracking system and the nature of the erratic beat predictions. The beat tracker was tested across 20 samples, half of which were pop music and half were samples of a guitar. All samples were 60 seconds in length. The lower section of the table shows the comparative performance of the tempo/beat-synchronous audio effects when their robustness to beat tracker errors is taken into consideration. Ω is the number of cycles per beat.

errors. *Offbeat* errors refer to beats that are tracked at the correct tempo, but are aligned between correct beats, approximately $\frac{\tau}{2}$ samples out of phase with correct beats, where τ is the beat period in audio samples. *Half/Double Tempo* errors are beats that are correctly aligned but are output at either half or double the correct tempo. Finally, all other errors are classed simply as *Incorrect*.

The tempo-synchronous delay effect is robust to many of these errors. Offbeat errors are erratic due to alignment, however the effect makes use of only the beat period and does not consider beat alignment information. Therefore, offbeat errors do not cause it to operate incorrectly. By using the results from table 1, we can discount offbeat errors and say that the performance of the tempo-synchronous delay is 84.6%. Half/Double Tempo errors cause the effect to delay the signal by either double or half the amount of time. While this is incorrect performance, the result is still perceptually related to the signal tempo, therefore, the error is subtle. All other types of error generally produce and undesirable result.

Beat-synchronous LFO-based effects also show some robustness to errors. For example, Type I LFOs that have
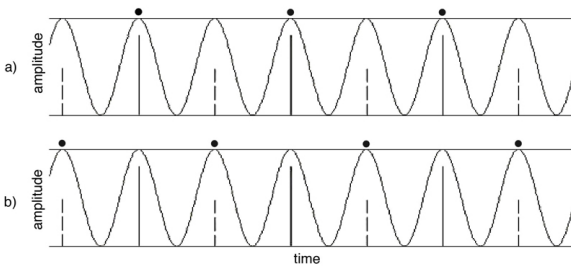
Fig. 4: a) shows the beat-synchronous LFO operating at 2 cycles per beat produced when the beats have been correctly tracked. The solid vertical lines represent beats, dotted-lines represent offbeats and black dots the beats predicted by the beat tracker. b) shows the same situation as a) except that the beats have been tracked on the offbeat. Note that the two waveforms are identical.

an even number of cycles per beat are unaffected by offbeat errors as the same waveform is produced (see Figure 4). As a result, we can again discount offbeat errors for these effects and consider the performance to be 84.6%. If an odd number of cycles per beat is used, then offbeat errors cause the waveform to operate in anti-phase to the correct waveform. For many waveforms this is a subtle error as the effect remains synchronised with beats in the signal. Half/Double Tempo errors cause the LFO to operate at a rate either half or double the correct rate. The output is still related to the beats in the signal and is therefore not an undesirable result. Finally, other error types cause the LFO not to be synchronised with the signal and can be considered simply as erratic performance.

Further work on these effects may include experimenting with other beat tracking systems to see if different problems are encountered while implementing effects or whether effect performance can be improved by certain aspects of other beat trackers. It would also be interesting to experiment with a 'sidechain' input to the effects. A sidechain input would allow a control signal to be used as the input to the beat tracker while effect processing would still be carried out upon the original input signal. This would allow an input from a percussive instrument during a live performance that encompassed several musicians. Finally, further evaluation in the form of extensive listening tests would be extremely valuable.

7. CONCLUSION

We have seen the augmentation of a number of audio effects using a beat tracker to control effect parameters.

The availability of effects that allow the setting of parameters using semantic and dynamic terms such as a beat rather than static terms such as milliseconds adds a great deal of flexibility to musicians. It allows them to create a certain effect and use it regardless of tempo. It also allows this to happen in a simple manner with no external tempo indicators or metadata needed to indicate tempo variation. We have seen that the effects show some robustness to beat tracker errors and therefore perform better than results of beat tracker testing may imply. With the type of system introduced here, a performer can avoid the restrictions currently a feature of conventional audio effects and easily create adaptive effects related to the tempo of the piece.

8. REFERENCES

[1] Davies M.E.P. and Plumbley M.D., "Context-Dependent Beat Tracking of Musical Audio", *IEEE Trans on Audio, Speech and Language Processing*, Vol. 15, Issue 3, pp. 1009-1020, March 2007

[2] Dutilleux, P., Zölzer U., "DAFX: Digital Audio Effects (Udo Zölzer ed.)", *John Wiley and Sons*, Chapters 2-5, pp. 31-136, 2002.

[3] Verfaille, V., Zölzer, U., Arfib, D., "Adaptive Digital Audio Effects (A-DAFx); A New Class of Sound Transformations", *IEEE Trans. on Audio, Speech and Language Processing*, Vol. 14, No. 5, pp. 1817-1831, September 2006

[4] R. Arora and W. A. Sethares, "Adaptive Wavetable Oscillators", *accepted for publication in IEEE Trans. Signal Processing*.