Percussive Beat tracking using real-time median filtering

Andrew Robertson[1], Adam Stark[2], and Matthew E. P. Davies[3]

[1] Centre for Digital Music, Queen Mary University of London, UK
[2] Codasign Ltd, UK
[3] INESC TEC, Portugal

Abstract. We present an efficient method for extracting the percussive component of a signal in real-time and a method for using this information to perform reliable beat tracking. The percussive signal can be used for the transcription of drum events and to learn the drum pattern for a song. We evaluate this method on several databases and thereby provide a comparison to other real-time methods.

1 Introduction

Beat tracking algorithms aim to replicate the human ability to tap in time with music. The problem has been approached using event detection [6], multiple agent hypotheses [3], comb filter resonators [7] and autocorrelation [2]. As input to the beat tracking algorithm, most methods use an onset detection function [1] – a mid-level representation that reflects the extent to which musical 'novelty' occurs in the current frame.

Since drums tend to drive the rhythmic pulse of a song, it makes sense to aim to synchronise most closely with the percussive elements of the music. Fitzgerald [5] applied median filtering to a spectrogram to separate the percussive and harmonic components of a signal. Whilst the technique is relatively simple to implement compared to other source separation methods, in a real-time system it can be computationally expensive and introduce a significant latency. In this paper, we present a method for percussive separation that can be carried out in real-time and evaluate a percussive detection function to improve real-time beat tracking. Our intended application is live synchronisation of audio and video.

2 Method

Fitzgerald's [5] technique for percussive separation is motivated by the observation that percussive features tend to consist of wide band noise across all frequencies and appear as vertical lines in the spectrogram,

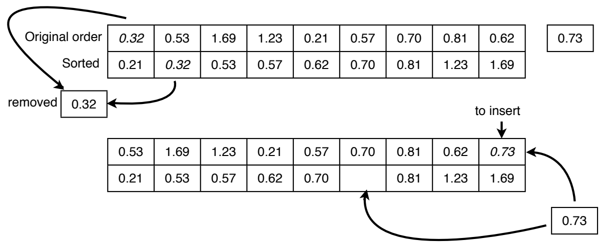

Fig. 1. Illustration showing how we add a new value (0.73) and efficiently find the median of the most recent N values.

whilst harmonic components consist of frequencies which persist, and thus appear as horizontal lines. Median filtering is applied in both the frequency and time directions to create separate percussive-enhanced and harmonic-enhanced spectrograms. These can be compared to create a percussive mask which indicates the extent to which the energy of each bin belongs to the percussive component. A percussive detection function is obtained by summing the percussive component at a given time frame across all frequencies. By summing frequency bins below 120Hz, and frequency bins between 200 Hz and 500 Hz, we can create functions which correlate to kick and snare strength respectively.

To carry out percussive separation in real-time, we require an efficient technique for median filtering. Median filters operate by replacing a given value in a signal with the median of the set of N values centred on the value under consideration. By storing the sorted set of values and the sequential set of values (preserving the order in which they arrived), we can simplify the calculation. Figure 1 illustrates how the process works in practice. Given a new value, we need to remove the oldest value, determined using the sequential array, by searching for this value in the sorted set of values and removing it from both sets. Finding the median then only requires adding the new value to the sequential set (at the most recent position) and inserting the new value at the correct location into the sorted set.

For a standard FFT of framesize 2048, we might limit the median filtering, but this still require several hundred values. Our proposed method allows us to calculate a thousand median filter values in approximately 0.7 msec rather than 2.3 msec. When processing audio in real-time, this is a significant reduction in the time taken for median filtering.

We use this percussive detection function as the input to our beat tracking algorithm. We adapted the method from Stark et al. [8], which uses a tempo following technique from Davies and Plumbley [2] and dy-

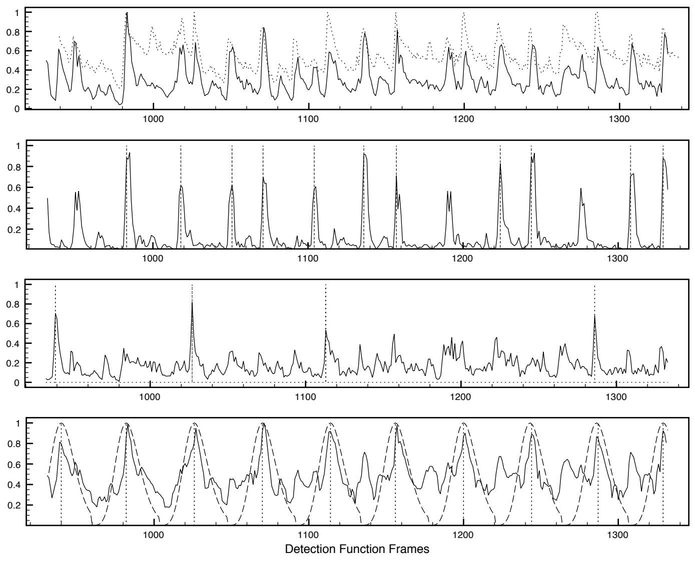

Fig. 2. The top figure shows the percussive detection function (solid) and the complex spectral difference onset detection function (dotted). The second and third figures show kick and snare strength functions with detected onsets indicated as vertical lines. The bottom plot shows the cumulative detection function (solid line) with the Gaussian window (dotted line) used when making beat predictions (dotted vertical lines).

namic programming method from Ellis [4]. Initialisation is carried by key commands. A synchronisation process adjusts for the latency inherent due to percussive separation and uses linear regression to predict the subsequent beat times. By quantising the kick and snare detection functions relative to the beat positions, we can learn a representation of the drum pattern. The inter-beat interval is divided into twelve equidistant temporal bins to include both triplets and sixteenth note events.

3 Evaluation

We evaluated the real-time C++ implementation relative to the *Btrack* algorithm [8]. Since our intended use is in live performance, our beat tracker requires initialisation. This seems a preferable design choice rather than introducing instability by having the beat tracker potentially switch to alternative tempo hypotheses. For the evaluation, we set the beat tracker to the correct beat and phase at five seconds into the song. Across the database, the beat tracker largely outperforms the *Btrack* algorithm, although providing initialisation from the ground truth inevitably introduces a bias in favour our proposed method.

We also created a Max for Live device that enables the beat tracker to control Ableton Live. A video of the the system is viewable at http://youtu.be/fT8DXecuXhg and illustrates the system's response to tempo fluctuations.

		Measure			
Database	Method	cmlC	cmlT	amlC	amlT
Beatles	Btrack	48.3	63.5	56.3	75.9
(179)	Proposed	63.2	78.3	63.2	78.3
Rock Corpus	Btrack	42.8	58.1	54.4	75.7
(200)	Proposed	56.9	75.5	57.5	76.6
Klapuri	Btrack	48.4	57.0	56.0	68.7
(474)	Proposed	60.8	70.5	60.8	70.5

Table 1. Comparison between Stark et al.'s *Btrack* algorithm and our method. The input feature for *Btrack* is the complex spectral difference onset detection function.

4 Conclusions

In this paper, we have presented a method for median filtering that enables percussive separation in real-time. We perform beat tracking using a percussive detection function and generate functions that correlate with kick and snare events. By learning the patterns of these drum events, we can form expectations for the drum patterns played in a bar.

References

1. J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5, Part 2):1035–1047, 2005.
2. M. E. P. Davies and M. D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3):1009–1020, 2007.
3. S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001.
4. D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.
5. D. Fitzgerald. Harmonic/percussive separation using median filtering. In *13th Int. Conference on Digital Audio Effects (DAFx-10), Graz, Austria*, 2010.
6. M. Goto and Y. Muraoka. A beat tracking system for acoustic signals of music. In *Proceedings of the Second ACM International Conference on Multimedia*, pages 365–372, 1994.
7. E. D. Scheirer. Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, 103(1):588–60, 1998.
8. A. M. Stark, M. E. P. Davies, and M. D. Plumbley. Real-time beat-synchronous analysis of musical audio. In *Proc. DAFX-09*, pages 299–304, 2009.