ENHANCED BEAT TRACKING WITH CONTEXT-AWARE NEURAL NETWORKS

Sebastian Böck, Markus Schedl

Department of Computational Perception
Johannes Kepler University, Linz
Austria
sebastian.boeck@jku.at

ABSTRACT

We present two new beat tracking algorithms based on the autocorrelation analysis, which showed state-of-the-art performance in the MIREX 2010 beat tracking contest. Unlike the traditional approach of processing a list of onsets, we propose to use a bidirectional Long Short-Term Memory recurrent neural network to perform a frame by frame beat classification of the signal. As inputs to the network the spectral features of the audio signal and their relative differences are used. The network transforms the signal directly into a beat activation function. An autocorrelation function is then used to determine the predominant tempo to eliminate the erroneously detected - or complement the missing - beats. The first algorithm is tuned for music with constant tempo, whereas the second algorithm is further capable to follow changes in tempo and time signature.

1. INTRODUCTION

For humans, tracking the beat is an almost natural task. We tap our foot or nod our head to the beat of the music. Even if the beat changes, humans can follow it almost instantaneously. Nonetheless, for machines the task of beat tracking is much harder, especially when dealing with varying tempi, as the numerous publications by different authors on this subject suggest.

Locating the beats precisely opens new possibilities for a wide range of music applications, such as automatic manipulation of rhythm, time-stretching of audio loops, beat accurate automatic DJ mixing or self-adapting digital audio effects. Beats are also crucial for analyzing the rhythmic structure, and the genre of songs. In addition they help identifying cover songs or estimating the similarity of music pieces.

The remainder of this paper is structured as follows: Section 2 gives a short overview over existing methods for beat tracking. Section 3 briefly introduces the concept and different types of neural networks with a special emphasis on bidirectional Long Short-Term Memory recurrent neural networks, which are used in the proposed algorithms. Section 4 details all aspects of the newly proposed beat tracking algorithms. Results and discussion are given in Section 5 and the final section presents conclusions and an outlook to further works.

2. RELATED WORK

Most methods for beat tracking of audio signals have a working scheme like the one shown in Figure 1. After extracting features from the audio signal, they try to determine the periodicity of the signal (the tempo) and the phase of the periodic signal (the beat locations). The features can be for example onsets, chord changes, amplitude envelopes, or spectral features. The choice of a particular feature mostly depends on the subsequent periodicity estimation and phase detection stages. For periodicity estimation, autocorrelation, comb filter, histogram, and multiple agent based induction methods are widely used. Some methods also produce phase information during periodicity estimation, and therefore do not need a phase detection stage to determine the exact position of the beat pulses. [1] gives a good overview on the subject.

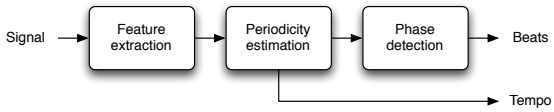


Figure 1: Basic workflow of traditional beat tracking methods.

Most of todays top performing beat tracking algorithms rely on onsets as features [2, 3, 4]. Since music signals contain much more onsets than beats, additional processing is needed to locate the beats within the onsets. By transferring this determination of beats into a neural network, less complex post-processing is needed to achieve comparable or better results.

3. NEURAL NETWORKS

Neural networks have been around for decades and are successfully used for all kind of machine learning tasks.

The most basic approach is the *multilayer perceptron* (MLP) forming a *feed forward neural network* (FNN). It has a minimum of three layers where the input values are fed through one or more hidden layers consisting of neurons with non-linear activation functions. The output values of the last hidden layer are finally gathered in the output nodes. This type of network is a strictly causal one, where the output is calculated directly from the input values.

If cyclic connections in the hidden layers are allowed *recurrent neural networks* (RNN) are formed. They are theoretically able to remember any past value. In practice however, RNNs suffer from the vanishing gradient problem, i.e. input values decay or blow up exponentially over time.

In [5] a new method called *Long Short-Term Memory* (LSTM) is introduced to overcome this problem. Each LSTM block (depicted in Figure 2) has a recurrent connection with weight 1.0 which enables the block to act as a memory cell. Input, output, and forget gates control the content of the memory cell through multiplicative units and are connected to other neurons as usual.

If LSTM blocks are used, the network has access to all previous input values.

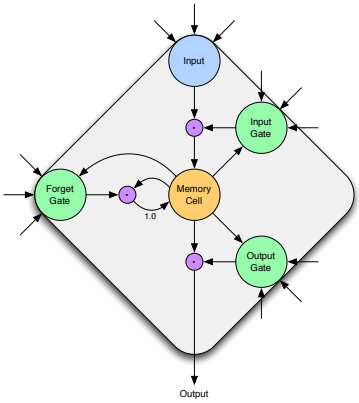


Figure 2: LSTM block with memory cell

If not only the past, but also the future context of the input is necessary to determine the output, a number of different strategies can be applied. One is to add a fixed time window to the input, another solution is to add a delay between the input values and the output targets. Both measures have their downsides as they either increase the input vector size considerably or the input values and output targets are displaced from each other.

Bidirectional recurrent neural networks (BRNN) offer a more elegant solution to the problem by doubling the number of hidden layers. The input values to the newly created set of hidden layers are presented to the network in reverse temporal order. This offers the advantage that the network not only has access to past input values but can also 'look into the future'.

If bidirectional recurrent networks are used in conjunction with LSTM neurons, a *bidirectional Long Short-Term Memory* (BLSTM) recurrent neural network is build. It has the ability to model any temporal context around a given input value. BLSTM networks performed very well in areas like phoneme and handwriting recognition and are described more detailed in [6].

4. ALGORITHM DESCRIPTION

This section describes our algorithm for beat detection in audio signals. It is based on bidirectional Long Short-Term Memory (BLSTM) recurrent neural networks. Due to their ability to model the temporal context of the input data [6], they perfectly fit into the domain of beat detection. Inspired by the good results for musical onset detection [7], the approach of this work is used as a basis and extended to suit the needs for audio beat detection by modifying the input representation and adding an advanced peak detection stage.

Figure 3 shows the basic signal flow of the proposed system. The audio data is transformed to the frequency domain via three parallel Short Time Fourier Transforms (STFT) with different window lengths. The obtained magnitude spectra and their first order differences are used as inputs to the BLSTM network, which produces a beat activation function. In the peak detection stage, first the periodicity within this activation function is detected with the autocorrelation function to determine the most dominant tempo. The beats are then aligned according to the previously computed beat interval. We propose two different peak detection algorithms, one tuned for music with constant tempo and beats (*BeatDetector*) and a second one which is able to track tempo changes (*BeatTracker*). The individual blocks are described in more detail in the following sections.

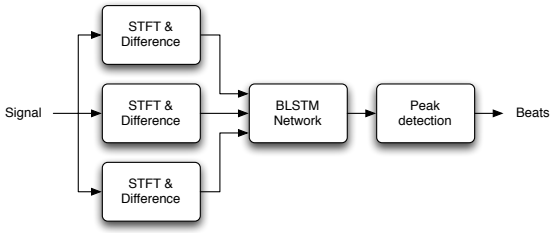


Figure 3: Basic signal flow of the presented beat detector / tracker

4.1. Feature Extraction

As input, the raw pulse code modulated (PCM) audio signal with a sampling rate of $f_s = 44.1$ kHz is used. To reduce the computational complexity, stereo signals are converted to a monaural signal by averaging both channels. The discrete input audio signal $x(n)$ is segmented into overlapping frames of W samples length. The windows with lengths of 23.2 ms, 46.4 ms, and 92.8 ms (1024, 2048, and 4096 samples respectively) are sampled every 10 ms, resulting in a frame rate $f_r = 100$ fps. A standard Hamming window $w(l)$ of the same length is applied to the frames before the STFT is used to compute the complex spectrogram $X(n,k)$

$$X(n,k) = \sum_{l=-\frac{W}{2}}^{\frac{W}{2}-1} w(l) \cdot x(l+nh) \cdot e^{-2\pi jlk/W} \qquad (1)$$

with n being the frame index, k the frequency bin index, and h the hop size or time shift in samples between adjacent frames. The complex spectrogram is converted to the power spectrogram $S(n,k)$ by omitting the phase portion of the spectrogram by:

$$S(n,k) = |X(n,k)|^2 \qquad (2)$$

Psychoacoustic knowledge is used to reduce the dimensionality of the resulting magnitude spectra. To this end, a filterbank with 20 triangular filters located equidistantly on the Mel scale is used to transform the spectrogram $S(n,k)$ to the Mel spectrogram $M(n,m)$. To better match the human perception of loudness, a logarithmic representation is chosen (cf. Figure 4(a)):

$$M(n,m) = \log\left(S(n,k) \cdot F(m,k)^T + 1.0\right) \qquad (3)$$

If large window lengths are used for the STFT, the raise of the magnitude values in the spectrogram occurs early compared to the actual beat location (cf. Figure 4(b)). Instead of calculating the simple positive first order difference as in [7], a more advanced method is used to overcome this displacement of the actual beat locations compared to the positive first order difference. First a median spectrogram $M_{median}(n,m)$ is obtained according to

$$M_{median}(n,m) = \text{median}\{M(n-l^*,m),\ldots,M(n,m)\} \quad (4)$$

with l^* being the length for which the median is calculated. This length depends on the used window size W for the STFT, and is computed as: $l^* = \lfloor W/100 \rfloor$. Both the use of the median and the length of the window were empirically determined during preliminary studies. The positive first order median difference $D^+(n,m)$ is then calculated as

$$D^+(n,m) \quad = \quad H\left(M(n,m) - M_{median}(n,m)\right) \quad (5)$$

with $H(x)$ being the half-wave rectifier function $H(x) = \frac{x+|x|}{2}$ (cf. Figure 4(c)). Using only the positive differences as additional inputs to the neural network gave better performance than omitting the differences at all or including both the positive and negative values.

4.2. Neural Network

For the neural network stage, a bidirectional recurrent neural network with LSTM units is used. As inputs to the neural network three logarithmic Mel-spectrograms $M_{23}(n,m)$, $M_{46}(n,m)$ and $M_{93}(n,m)$ (computed with window sizes of 23.2 ms, 46.4 ms, and 92.8 ms, respectively) and their corresponding positive first order median differences $D^+_{23}(n,m)$, $D^+_{46}(n,m)$, and $D^+_{93}(n,m)$ are used, resulting in 120 input units. The fully connected network has three hidden layers in each direction, with 25 LSTM units each (6 layers with 150 units in total). The output layer has two units, representing the two classes 'beat' and 'no beat'. Thus the network can be trained as a classifier with the cross entropy error function. The outputs use the softmax activation function, i.e., the output of each unit is mapped to the range $[0,1]$ and their sum is always 1. The output nodes thus represent the probabilities for the two classes.

4.2.1. Network Training

The network is trained as a classifier with supervised learning and early stopping. The used training set consists of 88 audio excerpts taken from the ISMIR 2004 tempo induction contest[1] (also known as the "Ballroom set") with lengths of 10 seconds each, the 26 training and bonus files from the MIREX 2006 beat tracking contest[2] with lengths of 30 seconds, and 6 musical pieces of the set introduced by Bello in [8] with lengths from 3 to 15 seconds. Each musical piece is manually beat annotated, marking every quarter note in case of time signature with a denominator of four (i.e., 2/4, 3/4, and 4/4), and the eighth note for all pieces (or parts of pieces) with a time signature of 5/8 or 7/8. The 120 files have a total length of 28.5 minutes and 3,595 annotated beats.

Each audio sequence is preprocessed as described above and presented to the network for learning. The network weights are initialized with random values following a Gaussian distribution with mean 0 and standard deviation 0.1. Standard gradient descent with backpropagation of the errors is used to train the network. To prevent over-fitting, the performance is evaluated after each training iteration on a separate validation set (a 15% randomly chosen disjoint part of the training set). If no improvement is observed for 20 epochs, the training is stopped and the network state with the best performance on the validation set is used onwards.

[1] http://mtg.upf.edu/ismir2004/contest/tempoContest/node5.html

[2] http://www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking

4.2.2. Network Testing

Since the network weights were initialized randomly, five different networks were trained on different sets of the training data. The beat activation functions of the 'beat' output nodes are then averaged and used as input to the following stage (cf. Figure 4(d)). For the evaluation the preprocessed music excerpts are presented to these five previously trained networks.

4.3. Peak Detection

The averaged beat activation function (cf. Figure 4(d)) gives the probability of a beat at each frame. Similar to [7], the function could be used directly to determine the beats by applying a simple threshold. However, a more sophisticated algorithm for peak picking is applied here. It is able to reduce the relatively high number of false positives and negatives even further. This method yields an F-measure value of 0.88 for a 5-fold cross validation on the complete training set, compared to 0.81 achieved using a simple threshold.

If constant tempo is assumed for (a part of) the musical piece, the predominant tempo can be used to eliminate false positive beats, or complement missing false negative ones. The two different proposed peak detection techniques differ only in the length for which a constant tempo is assumed. The *BeatDetector* assumes a constant tempo throughout the whole musical piece, whereas the *BeatTracker* considers only a moving window which covers the next 6 seconds. This modification enables the *BeatTracker* to follow tempo changes.

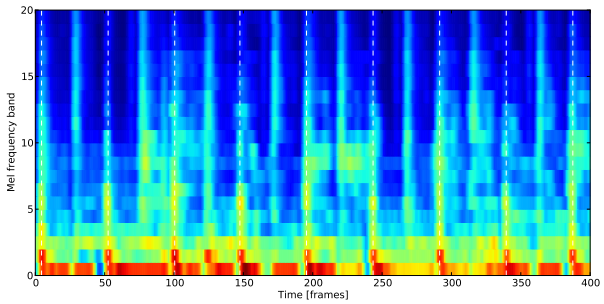
4.3.1. Autocorrelation Function

Both proposed algorithms first determine the tempo for the musical piece. The *BeatDetector* uses the entire input signal for calculation, whereas the *BeatTracker* only uses the next 6 seconds relative to the actual starting point. The most dominant beat interval of this segment is used to estimate the tempo. The autocorrelation function (ACF) is calculated on the beat activation function $a_b(n)$ as follows:

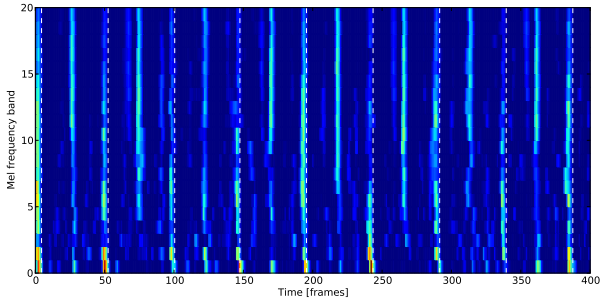
$$A(\tau) = \sum_n a_b(n+\tau) \cdot a_b(n) \quad (6)$$

The algorithm constrains the possible tempo range of the audio signal from $T_{min} = 40$ to $T_{max} = 220$ given in beats per minute. Thus only values of $A(\tau)$ corresponding to the range from $\tau_{min} = 273$ ms to $\tau_{max} = 1.5$ s are used for calculation. Since music tends to slightly vary in tempo and beats sometimes occur early or late relative to the absolute position of the dominant tempo, the resulting inter beat intervals vary as well. Therefore a smoothing function s is applied to the result of the autocorrelation function $A(\tau)$. A Hamming window with a size of $\tau_t = 150$ ms is used. The size of this window is not crucial, as long as it is wide enough to cover all possible interval fluctuations and remains shorter than the smallest delay τ_{min} used for the autocorrelation. This results in the smoothed autocorrelation function $A^*(\tau)$:

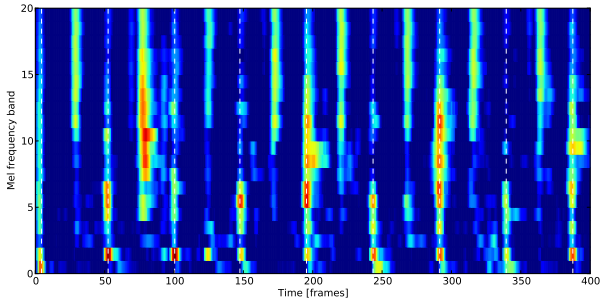
$$A^*(\tau) = A(\tau) \star s\left(\tau_t\right) \quad (7)$$



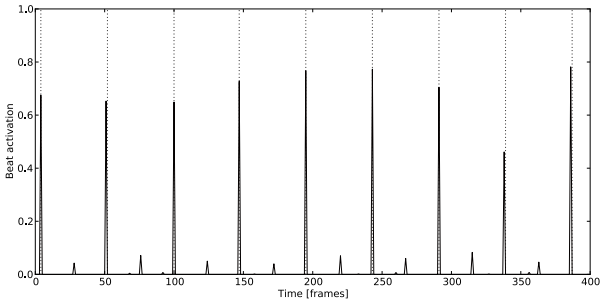
(a) Logarithmic Mel spectrogram with an STFT window of 92.8 ms



(b) Positive first order difference to the preceding frame



(c) Positive first order difference to the median of the last 0.41 s



(d) Beat activation function (output of the neural network stage)

Figure 4: Evolution of the signal through the processing steps of the algorithm. It shows a 4 s excerpt from *'Basement Jaxx - Rendez-Vu'*. Beat positions are marked with dashed/dotted vertical lines.

4.3.2. Beat Phase Detection

The dominant tempo T corresponds to the highest peak in the smoothed autocorrelation function $A^*(\tau)$ at index τ^*. This delay τ^* is used as the beat interval i. The phase of the beat p^* is computed as the highest value of the beat activation function's sum at the possible beat positions for the given interval i:

$$p^* = \max_{p=0\ldots i} \sum_k a_b(p + k \cdot i) \tag{8}$$

4.3.3. Peak Picking

Finally, the beats are represented by the local maxima of the beat activation function. Thus, we use a standard peak search around the locations given by $n_k = p^* + k \cdot i$ calculated with the previously determined p^*. To allow for small timing fluctuations, a deviation factor $d = 0.1 \cdot i$ is introduced and the final beat function $b(n)$ is given by:

$$b(n) = \begin{cases} 1 & \text{for } a_b(n_k - d) \leq a_b(n_k) \geq a_b(n_k + d) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The *BeatDetector* determines all beats in this manner. The *BeatTracker* only detects the next beat and moves the beginning of the lookahead window to that beat. Then the dominant tempo estimation and all consecutive steps (Section 4.3.1 to 4.3.3) are performed on the new section of the beat activation function.

5. EVALUATION

Beat tracking performance was evaluated during the MIREX 2010 beat tracking contest with two different datasets[3]. The first set, the McKinney collection (MCK set), has rather stable tempo. The second collection (MAZ set) consists of Chopin Mazurkas, which are in 3/4 time signature and contain tempo changes.

Both described algorithms outperformed all other contributions on the MCK set. The *BeatDetector* shows a small overall advantage over the *BeatTracker*. Depending on the used performance measure the relative performance gain compared to the next best algorithm is up to 5.7% (F-measure with a detection window of $\pm$70 ms), 6.9% (Cemgil: accuracy based on a Gaussian error function with 40 ms std. dev.), 8.2% (Goto: binary decision based on statistical properties of a beat error sequence), and 4.7 (PScore: McKinney's impulse train cross-correlation method). Table 1 summarizes the results and also includes the best result ever achieved in the MIREX competition by any algorithm as a reference to the state-of-the-art. It can be seen that our *BeatTracker* algorithm performs better or close to it (depending on the used performance measure). This shows the future potential of this approach compared to other signal based ones, given the fact that the actual peak picking algorithm is a rather simple one.

The tempo changes of the MAZ set are the main reason for the *BeatDetector* not performing better (see Table 2), as it assumes a constant tempo throughout the whole musical piece. Nonetheless the algorithm performs still reasonably well. As expected, the more flexible *BeatTracker* performs better and ranks second according to F-measure and first according to Cemgil's performance

[3]Evaluation measures described at http://www.music-ir.org/mirex/wiki/2010:Audio_Beat_Tracking#Evaluation_Procedures

MCK Set [%]	F-measure	Cemgil	Goto	PScore
BeatTracker	54.50	**41.30**	8.87	**59.19**
BeatDetector	53.16	40.28	**22.64**	57.73
GP3	50.27	37.21	20.92	56.54
LGG2	49.97	37.68	17.93	54.96
TL2	41.97	29.86	2.50	50.59
NW1	35.56	25.83	5.75	45.67
MRVCC1	25.70	18.34	0.14	38.36
ZTC1	24.64	18.61	0.41	26.07
GP1 (2009)	**54.80**	41.00	22.20	59.00

Table 1: Results for the MIREX 2010 beat tracking evaluation (MCK set). Only the best performing algorithm of other participants are shown; GP1 & GP3: Peeters, LGG2: Oliveira et. al., TL2: Lee, NW1: Wack et. al., MRVCC1: Campos et. al., ZTC1: Zhu et. al.

measure. However, the most mentionable aspect is that the neural networks were trained solely on ballroom dance and other kinds of western pop music. Neither a classical piece nor piano music was used for training. Furthermore, only one training example actually contained tempo changes. This suggest that even better performance can be expected when trained on music which has properties similar to the MAZ data set.

MAZ Set [%]	F-measure	Cemgil	Goto	PScore
TL2	**68.46**	40.42	0.00	**72.21**
BeatTracker	58.74	**51.81**	0.00	57.92
MRVCC2	49.26	39.55	0.31	51.22
GP4	48.27	36.72	0.31	50.06
BeatDetector	47.30	38.20	0.00	45.92
LGG2	41.48	30.65	0.00	43.51
NW1	27.59	19.82	0.00	31.35
ZTC1	1.16	0.94	0.00	0.94

Table 2: Results for the MIREX 2010 beat tracking evaluation (MAZ set). Only the best performing algorithm of other participants are shown; TL2: Lee, MRVCC2: Campos et. al., GP4: Peeters, LGG2: Oliveira et. al., NW1: Wack et. al., ZTC1: Zhu et. al.

6. CONCLUSIONS AND FUTURE WORK

This paper presented two novel beat tracking algorithms which perform state-of-the-art although they use a relatively simple and straight forward approach. The *BeatTracker* outperformed all other algorithms in the MIREX 2010 beat tracking contest for the McKinney dataset. Although no classical music was used for training and the training set had less then 3.5 minutes of material with a time signature of 3/4 the new *BeatTracker* performed still reasonably well on the Mazurka test set (all excerpts are in 3/4 time signature). This shows the aptitude of the BLSTM neural network for correctly modeling the temporal context and directly classifying beats. Since the *BeatTracker* shows superior performance over the more simple *BeatDetector* even for musical excerpts with constant tempo, future development will concentrate on this algorithm.

Besides training with a more comprehensive training set, future work should also investigate a possible performance boost by implementing some more advanced beat tracking algorithms in the peak detection stage. Kalman filters [9], particle filters [10], a multiple agents architecture [11] and dynamic programming [2] seem promising choices. Another possibility is the inclusion of other input features which haven proven to be effective for identifying beats [12].

7. ACKNOWLEDGMENTS

This research is supported by the Austrian Science Funds (FWF): P22856-N23.

8. REFERENCES

[1] F. Gouyon and S. Dixon, "A review of automatic rhythm description systems," *Computer Music Journal*, vol. 29, pp. 34–54, February 2005.

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

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

[4] G. Peeters, "Beat-marker location using a probabilistic framework and linear discriminant analysis," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, September 2009.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Ph.D. thesis, Technische Universität München, 2008.

[7] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long short-term memory neural networks," in *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, August 2010, pp. 589–594.

[8] 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*, vol. 13, no. 5, pp. 1035–1047, September 2005.

[9] Y. Shiu, N. Cho, Chang P.-C, and C.-C. Kuo, "Robust on-line beat tracking with kalman filtering and probabilistic data association (kf-pda)," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1369–1377, August 2008.

[10] S. Hainsworth and M. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP J. Appl. Signal Process.*, vol. 2004, pp. 2385–2395, January 2004.

[11] S. Dixon, "Evaluation of the Audio Beat Tracking System BeatRoot," *Journal of New Music Research*, vol. 36, no. 1, pp. 39–50, 2007.

[12] F. Gouyon, G. Widmer, X. Serra, and A. Flexer, "Acoustic cues to beat induction: A machine learning perspective," *Music Percepcion. UCPress*, vol. 24, pp. 177–188, 2006.