ROBOD: a Real-time Online Beat and Offbeat Drummer

Sebastian Böck [1]*, Florian Krebs* [12]*, Amaury Durand* [3]*, Sebastian Pöll* [1]*, Raminta Balsyte* [1]

[1] Department of Computational Perception, Johannes Kepler University Linz, Austria
[2] Joanneum Research, Graz, Austria [3] Télécom ParisTech, Paris, France

ABSTRACT

This paper describes the submission of team 24717 for the IEEE signal processing cup 2017. The challenge is to develop a real-time beat tracking system operating on an embedded device. As an application, we present **ROBOD**, the **R**eal-time **O**nline **B**eat and **O**ffbeat **D**rummer.[1] ROBOD listens to a musician in a live performance, recognises beats, downbeats and rhythmic patterns, and then accompanies the musician on the drum set. Supplementary material including videos presenting the system in action can be found online at `https://gitlab.cp.jku.at/ROBOD/supplementary/wikis/home`.

1. INTRODUCTION

The automatic inference of the metrical structure in music is one of the fundamental problems in machine listening. The task of beat tracking deals with finding the most salient level of this metrical grid, the beat. The beat consists of a sequence of regular time instants which usually invokes human reactions like foot tapping. During the last years, many beat tracking algorithms have been proposed and by now are able to achieve performance on par with human tapping [1].

However, most of these systems require the audio signal to be present as a whole in order to analyse it and infer the beat positions. When processing a continuous audio stream, the problem becomes more difficult to solve, since the decision about a beat has to be made instantaneously given only the past signal. Especially the beginning of musical pieces can be challenging when no strong beat is present.

Most other challenges are of technical nature. The processing time required and the time delay introduced by the algorithm has to be kept as low as possible. In this paper we present a real-time beat tracking algorithm which is based on a state-of-the-art offline algorithm, adapted to the needs of online processing of the audio signal and the limitations of the embedded platform chosen, the Raspberry Pi.

[1] We aimed at calling the system *ROBD*, but unfortunately it sometimes prefers offbeats over beats.

2. SYSTEM DESCRIPTION

The presented system is based on the beat tracking approach presented in [1]. We modified the system to be able to cope with continuous live audio input. Therefore we had to adapt both the structure of the recurrent neural network (RNN) and the inference method of the dynamic Bayesian network (DBN).

The detected beat positions are then further processed to determine the rhythmical pattern played and the location of the detected beats inside a bar. This part combines the work presented in [2] and [3]. The beat positions and a simple spectral flux-based input feature are used to obtain beat-synchronous features which are the scored with Gaussian mixture models (GMMs) and used as observations for another DBN.

Finally, we use the inferred information to control an artificial drummer. The hardware of the drummer is able to control up to four different drums, but could be easily extended. The structure of the whole system is depicted in Figure 1 and described in the following sections.

The whole system is implemented in Python with the open source *madmom* library [4]. To be able to process live audio signals, we added real-time online processing capabilities to *madmom*. All source code of the system will be made available at the project's website at `https://gitlab.cp.jku.at/ROBOD`.

2.1. Signal pre-processing

Both the beat tracking as well as the downbeat and pattern tracking stage share a common signal pre-processing. We use the short-time Fourier transform (STFT) to obtain a time-frequency representation of the signal. The audio signal is sampled at 48 kHz and split into overlapping frames of 2048 samples length located 480 samples apart. This results in a frame rate of 100 frames per second. A frame is reported as soon as enough new samples are present at the input audio buffer.

The phase of the complex STFT is omitted and only the magnitudes are used onwards. The magnitude spectrogram is then filtered with a semitone filterbank and scaled logarithmically after adding a constant value of 1. Frequencies below

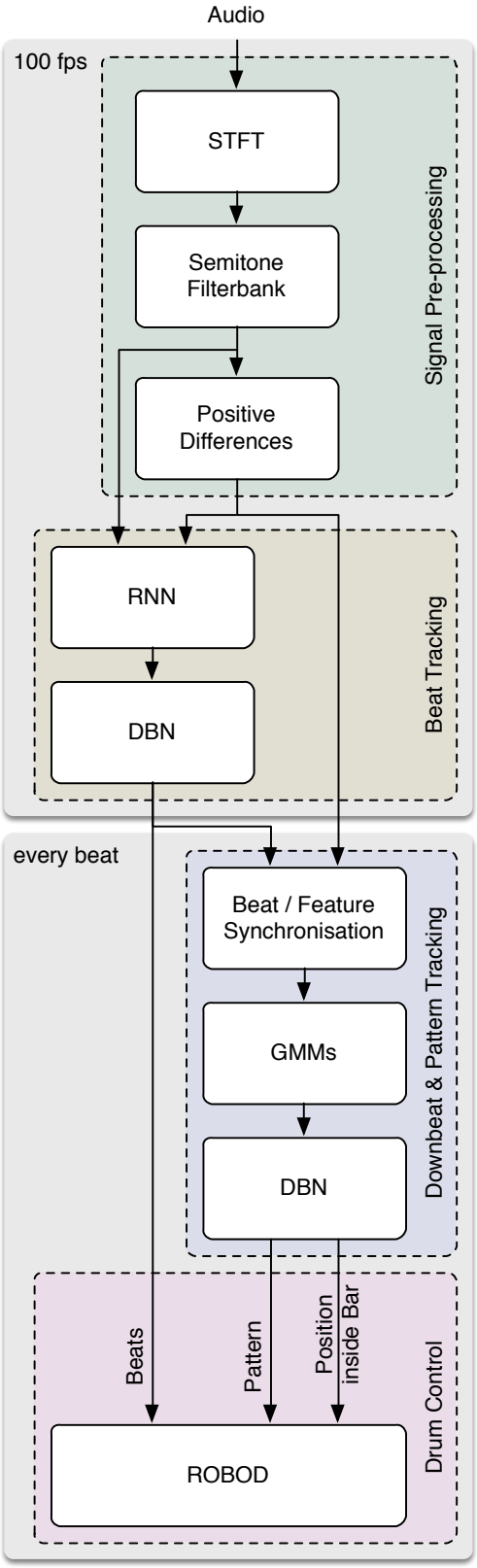

Fig. 1. ROBOD system overview.

30 Hz and above 18 kHz are cut off. This reduces the dimensionality of the features down to 81 frequency bands and puts them in a range suitable for all consecutive processing steps. These measurements are inspired the the human auditory system. Since beats occur mostly at onset positions, we additionally compute the positive first order differences to be able to spot onsets more easily. These parameters have been found to be useful especially in online scenarios, where audio signals cannot be normalised (i.e. put into a pre-defined range) and the level of the signal is not known beforehand [5].

As input to the neural network of the beat tracking stage, we use both the logarithmically filtered and scaled spectrogram as well as their positive first order differences. As feature for the downbeat and pattern tracking stage we use the spectral flux, which is computed by summing all positive first order differences.

2.2. Beat tracking

The beat tracking part is used to predict the positions of the beats. If only these are of interest, all remaining parts (the lower half of Figure 1 described in Section 2.3 to 2.4) can be omitted.

2.2.1. Recurrent neural network

The beat tracking stage is built around a recurrent neural network (RNN) which is used to determine possible beat locations in the audio signal. Contrary to the original work, which uses a bi-directional RNN to process the complete audio signal at once, we use a uni-directional network which operates on a stream of input features on a frame-by-frame basis.

The network topology closely follows [6] and [1] and uses three hidden layers with 25 long short-term memory (LSTM) units each. For training of the networks, standard gradient descent with error backpropagation and a learning rate of $1e^{-5}$ is used. We initialise the network weights with a uniform distribution with mean 0 and standard deviation of 0.1. We use early stopping with a disjoint validation set to stop training if no improvement over 20 epochs can be observed.

As training data we use the same data as in [7]. To achieve acceptable beat tracking performance we were forced to displace the targets 15 ms into the future for training. Interestingly, this does not lead to the same shift when predicting the beats. We experienced a mean shift of only 5 ms instead.

2.2.2. Dynamic Bayesian network

The output of the RNN represents the beat probability at the current time frame. This probability is used directly as a observation likelihood for the dynamic Bayesian network (DBN), which jointly infers tempo and phase of a beat sequence. The original DBN used in [1] uses a rather large state space. To reduce the computational demand, we replace the

state space by the one proposed in [8]. This has the additional advantage of increased performance.

Since we do not have the complete beat sequence at hand, we cannot use Viterbi decoding to obtain the global best solution. Instead, we use the forward algorithm to determine the most likely state given the RNN activations and the preceding state probability.

2.3. Downbeat and pattern tracking

In order to accompany a musician on the drums, we need to identify the length and position within a bar, as well as the rhythmic pattern. To this end, we combine the systems presented in [2, 3] and adapt them to our needs.

We selected six drum patterns that ROBOD should be able to play. One with a bar length of two, two with a bar length of three and four and one with a bar length of five beats.

2.3.1. Beat-synchronous features

To simplify the later inference with a DBN, the features are first synchronised to the rate of the beats determined by the beat tracking stage (cf. Section 2.2). This has the advantage that the resulting beat-synchronous features are tempo-invariant and the resolution of the downbeat tracking state space can be reduced from frames to beat sub-divisions. In this work, we discretise each beat into four sub-divisions.

The system presented in [2] uses a 2-dimensional feature. However, we found that for our application the 1-dimensional spectral flux is sufficient and thus use this.

2.3.2. Gaussian mixture models

For each pattern and each beat sub-division we fit a Gaussian mixture model (GMM) with four mixture components to the spectral flux extracted from the training set. This means that a pattern with three beats is represented by $3 \times 4 = 12$ GMMs. With six patterns and in total 21 modelled beats, the model contains 84 GMMs, whose parameters have to be learned from data. In order to learn the parameters for the GMMs, we recorded five 30-second long tracks for each pattern.

2.3.3. Dynamic Bayesian network

For inference of the downbeat positions, we use a combination of the approaches presented in [2] and [3]. In contrast to the beat DBN (Section 2.2.2), the hidden states consist only of beats here. For our model with 6 patterns and therefore 21 beats in total, the resulting DBN has 21 states. For a given pattern, the states can only be visited in cyclical left to right order (e.g. $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ in a $3/4$ bar). At the end of a pattern it is possible to change a pattern with $p = 0.001$, otherwise the pattern stays constant.

We use the forward path to compute the probability distribution over the 21 hidden states at every beat. The most probable state (which determines the pattern and position of the current beat inside the bar) is reported to the subsequent drum control stage to coordinate the drum hits.

2.4. Drum control

The purpose of this stage is mostly to control the drum hits and apply timing corrections needed because of the delay introduced by the previous stages. These timing corrections depend on the hardware used, e.g. a faster computer introduces less delay. We also smooth the tempo by using the median tempo of the last three tempi. In addition to tempo corrections, the drum control module has to know which drum pattern fits to the pattern played by the musician. In this work, we defined these drum patterns by hand.

Because of the delay of the previous stages, this stage extrapolates the next beat position from the smoothed tempo and thus hits the drum proactively rather than waiting for the next beat to be detected. We decided to always play a whole bar rather than stop playing immediately even if no more beats are detected.

3. HARDWARE / SOFTWARE IMPLEMENTATION

Since most parts of the system are already publicly available as part of the *madmom* library [4] written in *Python*, we chose to implement the system in the same language. *Python* is an interpreted language and is thus not as fast as compiled languages. However, *madmom* makes heavy use *NumPy* [9], *SciPy* [10], and speed critical parts are written in *Cython* [11].

Modern computers are able to finish all computations with delays only recognisable by professional musicians – even if the system is written in an interpreted language. But since the system is implemented on an embedded device with rather limited processor power, the individual stages contribute considerable delays which are very easy to recognise.

In order to keep these delays at a minimum, we split the two basic operating blocks processed at different sampling rates (i.e. at 100 frames per second and at every beat, respectively) into two concurrent processes. The first handles all signal pre-processing and beat tracking related stuff, whereas the second deals with the downbeat and pattern selection and controls the drums.

The beat tracking stage introduces an inherent average delay of ca. 6 ms (the process has an average load of 60% at 100 fps). Together with the 5 ms of the RNN predictions (cf. Section 2.2.1), this adds up to a shift which needs to be corrected in the drum control stage.

The downbeat and pattern tracking stage has an average processing time of up to 200 ms, thus it was inevitable to split it into a separate process to not block the processing of consecutive frames arriving every 10 ms. The scoring of the 84 GMMs contributes most and the DBN with its very few states

only to a minor degree to these 200 ms. Therefore, the computational load of the downbeat tracker depends on the tempo of a piece, as for faster songs the GMMs have to be evaluated at a higher rate. Yet, the processing can be done in real-time for tempi up to 300 bpm.

The drum control is handled with an Arduino controller connected to the serial console of the Raspberry Pi. This device translates a sequence of drum control messages into servo movements.

4. CONCLUSION

In this paper we presented the inner workings of our submission for the IEEE signal processing cup 2017. It consists of a real-time beat, downbeat and pattern tracking algorithm and the corresponding hardware control unit to accompany musicians on a drum set.

5. REFERENCES

[1] Sebastian Böck, Florian Krebs, and Gerhard Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 10 2014, pp. 603–608.

[2] Florian Krebs, Sebastian Böck, and Gerhard Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 11 2013, pp. 227–232.

[3] Florian Krebs, Sebastian Böck, Matthias Dorfer, and Gerhard Widmer, "Downbeat Tracking using Beat-Synchronous Features and Recurrent Neural Networks," In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 129–135.

[4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.

[5] Sebastian Böck, Florian Krebs, and Markus Schedl, "Evaluating the online capabilities of onset detection methods," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 10 2012, pp. 49–54.

[6] Sebastian Böck and Markus Schedl, "Enhanced Beat Tracking with Context-Aware Neural Networks," in *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx)*, Paris, France, 9 2011, pp. 135–139.

[7] Sebastian Böck, Florian Krebs, and Gerhard Widmer, "Joint beat and downbeat tracking with recurrent neural networks," In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 255–261.

[8] Florian Krebs, Sebastian Böck, and Gerhard Widmer, "An Efficient State Space Model for Joint Tempo and Meter Tracking," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, Spain, 10 2015, pp. 72–78.

[9] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, 3 2011.

[10] Eric Jones, Travis Oliphant, Pearu Peterson, et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2016-03-11].

[11] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith, "Cython: The Best of Both Worlds," *Computing in Science Engineering*, vol. 13, no. 2, pp. 31–39, 3 2011.