AN OPEN-SOURCE DRUM TRANSCRIPTION SYSTEM FOR PURE DATA AND MAX MSP

Marius Miron, Matthew E.P. Davies, Fabien Gouyon

INESC TEC, Sound and Music Computing Group, Porto, Portugal

ABSTRACT

This paper presents a drum transcription algorithm adjusted to the constraints of real-time audio. We introduce an instance filtering (IF) method using sub-band onset detection, which improves the performance of a system having at its core a feature-based K-nearest neighbor classifier (KNN). The architecture proposed allows for adapting different parts of the algorithm for either bass drum, snare drum or hi-hat cymbals. The open-source system is implemented in the graphic programming languages Pure Data (PD) and Max MSP, and aims to work with a large variety of drum sets. We evaluated its performance on a database of audio samples generated from a well known collection of midi drum loops randomly matched with a diverse collection of drum sets. Both of the evaluation stages, testing and validation, show a significant improvement in the performance when using the instance filtering algorithm.

Index Terms— drum transcription, feature-based classification, real-time audio, Pure Data, Max MSP

1. INTRODUCTION

We propose an algorithm for automatic transcription of drum performances and drums loops. Our goal is to automatically extract symbolic notation [1] which would play a vital role in the creation of tools for musicians, for example, in expressive transformation of drum loops, recombination or accompaniment. Moreover, a real-time implementation could be used in live music situations, where it could enhance the musician's performance.

Automatic transcription of percussion sound events assumes detecting and labeling specific drum events. There are a few problems which can arise when dealing with this task, which relate to the acoustic diversity of the drum sounds we want to label, as they can be produced by a variety of drum sets, to the difference in loudness of different loops, and to the fact that two or more sounds can be simultaneous or overlap.

This work is financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundacao para a Ciencia e a Tecnologia (Portuguese Foundation for Science and Technology) within project Shake-it, Grant PTDC/EAT-MMU/112255/2009. Part of this research was supported by the CASA project with referencePTDC/EIA-CCO/111050/2009 (FCT).

Regarding existing methods for drum loops transcription, Fitzgerald et al [2] used prior subspace analysis and independent component analysis to deal with mixture of drum sounds and label them in drum loops. Tzanetakis et al [3] proposed two methods for transcribing kick and snare, using bandpass filters and wavelet analysis. Gillet and Richard discussed three approaches based on Hidden Markov Models and Support Vector Machines [4], and noise subspace projection [5]. Battenberg et al [6] introduced a technique for decomposing drum audio into spectral templates which are learned using a probabilistic Gamma Mixture Model. Additionally, with respect to real-time audio and Pure Data, an algorithm for isolated percussion sounds classification using a KNN classifier was described by Brent [7].

Labeling drums in polyphonic audio is a difficult challenge because of the presence of other instruments. This task was the subject of a MIREX contest which took place in 2005 [8]. Tanghe et al [9] used an onset detector, a feature extractor and a SVM classifier to label either kicks, snares and hi-hats. Paulus and Virtanen [10] combine principal component analysis and non-negative matrix factorization to decompose the spectrogram into spectrograms of the targeted drum sounds. Yoshii et al [11] proposed an approach which computes an adaptive spectrogram template for each drum class.

Despite the number of methods published, very little code has been made available and most systems are not targeted towards processing data in real-time, or even causally. Hence, we propose an open-source algorithm that works in real-time and offline. We are building on the previous approaches as we are using an onset detector for percussive events, a feature computation step, and a trained classifier for each drum class, labeling an event as a member or a non-member of that class. We are introducing a pre-classification stage, a method based on sub-band onset detection, with the purpose of filtering the instances which are fed to each class. The results show that the instance filtering (IF) and a KNN classifier, give better results combined than evaluated separately.

The remainder of this paper is structured as follows: in the Section 2, we describe the architecture of the system, the onset detection, instance filtering, feature selection and classification, followed by the implementation in PD and Max MSP. In the Section 3, we present the results of the two evaluation stages, testing and validation, with the proper discussion.

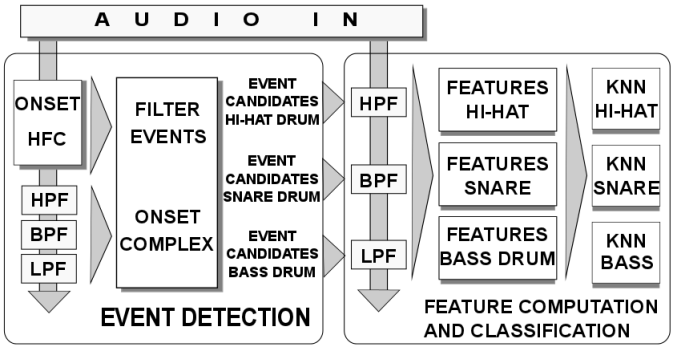

Fig. 1. The architecture of the drum transcription system

2. METHOD

Our algorithm takes as input stereo live audio, sampled at 44100 Hz, generated by acoustic or electronic drum kits, and detects bass drum, snare drum and hi-hat cymbals. The architecture of the system involves an event detection stage, and a feature extraction and classification stage. The first step is to detect possible drum onset candidates for classification. Secondly, a set of features is calculated from a salient part of the audio, after the onset has been detected. Finally, a KNN classifier assigns a label to the instance candidate.

Because we want to filter incorrect instances that might be seen by the classifier, we apply an instance filtering algorithm based on sub-band onset detection, which occurs at a pre-classification stage, after the initial onset detection. As the first onset detection has to be quick, the second one needs to have a high resolution to detect changes in the specific subbands.

We start with the general assumption that we can have all combination of kick, snare and hi-hat along with tom-toms at the same time. Thus, for making our system able to detect simultaneous events, we opted for labeling each onset as kick or non-kick, snare or non-snare, hi-hat or non-hi-hat. This leads to the use of three binary classifiers, as in [9], which have to be trained separately and function in parallel. The feature computation also functions separately because we are computing the feature after filtering the signal in certain frequency bands. Similarly, we are using three instance filtering, trying to filter events for each drum class, as described in Figure 1.

2.1. Event detection

2.1.1. Onset detection

We first need to detect the onsets. Because we are working with real-time audio, the onset detection algorithm has to be fast enough to capture a significant part of drum sound, as the analysis part is only triggered by this onset. We are using the high frequency onset detection (HFC) as it was reported to be

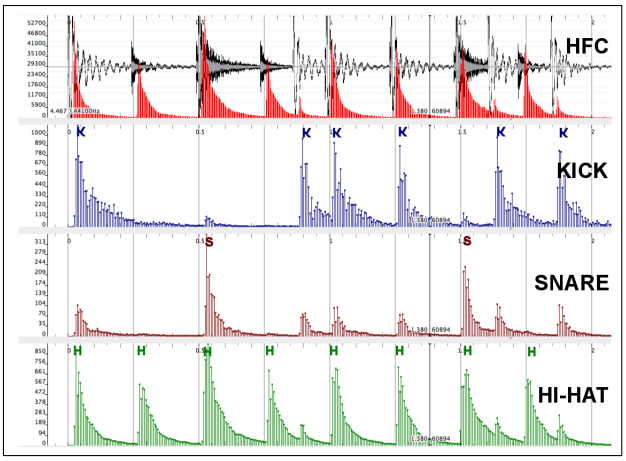

Fig. 2. The HFC onset detection function and complex the onset detection functions for each drum class, as a part of the event detection and instance filtering stage.

the best for percussive onsets [12]. In order to improve the responsiveness and decrease the delay time from the actual onset, we set the window size to 512 samples (11.6 ms) and hop size 128 samples (2.9 ms).

2.1.2. Instance filtering

Our assumption is that, since it uses a smaller window size because it needs to be fast, the HFC onset detection could detect a number of false positives, which can be fed to the classifier, resulting in a drop in performance. Furthermore, as the classifying stage occurs, some instances seen by the classifier are not characteristic to the represented data (noise) [13, p 134]. We propose a pre-classification algorithm for filtering the instances which are inputted into the KNN classifier, thus eliminating the false positives, in order to improve performance.

Because we need to filter the events based on three frequency bands for the three classifiers, we are using a second onset detector, with a higher resolution frequency-wise, that can detect onsets on specific bands. Three onset detectors work on the filtered signal and have to react to changes in the spectrum in those specified bands and not only in the high frequency band (Figure 2). We will use the complex domain onset detector with double window (1024 samples, 23.2 ms) and hop size (256 samples, 5.8 ms) because it tracks the spectral differences across the frames [12].

The number of frames taken into consideration and the big window size introduce a higher delay for the complex onset, hence it will always come after the HFC one. If no detection is made in a certain frequency band, then that instance is not sent to the classifier. We can adjust each onset detector to act particularly to a specific drum type.

2.2. Feature Computation and Classification

Since different drum strokes can occur at the same time, we have three processing streams running in parallel and we will compute all the features on the filtered audio, separately, for each classifier. We will low pass (LP) filter the audio for the bass drum, band pass (BP) filter it for the snare drum and high pass (HP) filter it for the hi-hat. Studying the acoustic properties of each class and their variance, we pick empirically the values for the central/cut-off frequency (Hz), as it follows: 90 Hz for BD, 280 Hz (20 Hz bandwidth) for SD, 9000 Hz for HH.

Another reason for implementing such filters is the classifier's sensitivity to noisy features [13, p 208], when just a few features carry important information. In this case, filtering the signal for a specific drum class before computing the actual features, it also diminishes the weight of the noisier features that are not relevant for that class.

Taking into account the inter-onset interval at the fastest tempo and the time envelope for each class, the salient part of the sound is set to the first approximately 100 ms (10 frames of 2048 samples with an overlapping factor of 75%) after the detected onsets [1]. In this way we will capture the most important part of the decay of the instrument, on which we will compute a set of features for either bass drum, snare or hi-hat, and their means over these frames. If another onset occurs earlier than this interval, the salient part is considered to be the inter-onset interval between the two successive onsets. The window size is larger to attain higher spectral resolution.

Because a certain drum stroke can last less than 100 ms, and because it is likely to have a new onset overlapping with the decay tail of onset before, we choose to weight the features with the value of the root mean square (RMS) [1] in order to give less weight to silent frames.

A set of spectral features, energy in bark bands and bark spectrum cepstral coefficients (BFCC) were reported to achieve good performance when classifying unpitched percussion [14]. The following features are chosen: energy in 23 bark bands, 23 bark frequency cepstrum coefficients, spectral centroid, spectral rolloff, as described in [7].

The classification stage uses a KNN classifier where an object is classified by a majority of votes of it's neighbors [13, p 209]. For each drum category, the classifier takes as input the feature vector, seeks the closest instances in the training data, and computes the most likely class, along with a measure of confidence.

2.3. Implementation

We choose to implement our system in the PD-extended version of PD, using the libraries described below. For the onset computation stage we are using a wrapper for the `aubio` library [12]. The chosen values for the HFC onset detection [12] parameters are -80 dB for the silence threshold and 0.3 for the onset threshold. For the complex onsets, we set these parameters to -50 dB and 0.8 for bass drum, -70 dB and 0.7 for snare, and -120 dB and 0.5 for hi-hat. The peak threshold is particularly high because we ought to detect very clear peaks in specific frequency bands, and the silence threshold is low because we need to have a higher tolerance for the peaks that do not fall in the bands we apply the filtering.

The three filters are implemented using the PD's `ggee` library to generate coefficients for the correspond biquad filters. For the KNN classifier and the feature computation we are using the patches included in the `timbreID` library as described in [7].

The PD implementation allows for modularity, every audio-processing patch working with its own window size and hop size, independently of the ones of the environment, adapting them for the requirements of each task. The open-source code and patches are available online on git repositories for PD [15] and Max MSP [16].

3. EVALUATION

The evaluation involves three stages: training, testing and validating. First, we train our system on a set of data. In the second stage, we test on a smaller collection, adjusting every setting to achieve optimal results, and then we validate the obtained configuration on a larger collection. We also evaluate onset detection stage, and the system with and without instance filtering, in order to know what influence has each stage on the final results. To assess the performance we are using an F-measure F, precision p, and recall r as described in [13, p 270]. The window used for evaluation is 35 ms.

3.1. Conditions

We opted for a training database made of 884 instances of which 284 bass drum, 284 snare drum, 272 open/closed hi-hat or ride, 44 toms gathered from different drum sets, most of them acoustic. We train three classifiers using this data, for bass/non-bass, snare/non-snare and hi-hat/non-hi-hat.

The audio for the test and validation data sets is generated using Timidity++[17], by randomly matching midi drum loops to a various set of drum kits soundfonts. For testing, we are using 177 drum loops (5204 instances) of different genres from the Groove Monkee [18] collection and 50 drum kits. For the validation stage, we randomly matched 242 loops (13712 instances) from the Loop Loft Drumatic Beats collection with 75 drum kits. Both audio and midi are independent between the data sets. The system is evaluated on the PD implementation.

3.2. Testing and Validation

3.2.1. Event detection evaluation

The HFC onset detection gives the following results for the testing dataset: $F = 0.93$, $p = 0.95$, and $r = 0.91$. The

	Testing			Validation		
	F	p	r	F	p	r
BD IF	0.76	0.68	0.86	0.65	0.54	0.81
BD C	0.75	0.70	0.81	0.52	0.40	0.74
BD C+IF	**0.85**	0.91	0.80	**0.72**	0.71	0.73
SD IF	0.57	0.43	0.85	0.60	0.46	0.83
SD C	0.74	0.75	0.72	0.74	0.73	0.74
SD C+IF	**0.79**	0.88	0.71	**0.79**	0.86	0.74
HH IF	**0.88**	0.84	0.93	0.77	0.66	0.93
HH C	0.85	0.94	0.78	0.79	0.72	0.89
HH C+IF	0.85	0.95	0.77	**0.88**	0.89	0.87
Overall	0.84	0.93	0.76	0.81	0.83	0.79

Table 1. The comparison of the results when testing the system with no classifier: IF, using just the classifier C, and combining the classifier with instance filtering C+IF

class-wise accuracy for kick is 0.89, for snare 0.87, and for hi-hat 0.94. The delay from the actual onset depends on the window and hop sizes, and the majority of onsets are detected within 10ms-15ms time frame.

We evaluate how good the instance filtering part performs in order to asses the performance of the event filtering stage. We test the system using just the output of the filtering stage without the classifier, as seen in Table 1. The precision of the instance filtering is high for hi-hat (0.84), lower for bass drum (0.68) and the lowest for the snare drum (0.43), where we can't exclude for example tom toms, high pitched bass strokes or low pitched hi-hats. Additionally, we compute the specificity [13, p 270], a measure of how filtering rejects the instances of other classes, obtaining 0.86 for BD, 0.70 for SD and 0.84 for HH.

3.2.2. Testing and Validation

In order to asses the performance and stability of our system after the instance filtering step, we evaluate it with and without this component. The original testing configuration is the one using 23 BFCC, energy in 23 Bark bands and the spectral descriptors mentioned in Section 2.2. Furthermore, we optimised the parameters of the classifier for testing. We determined a optimal value $k = 5$, the number of neighbors, and we used Euclidean distance [13, p 25] to find the closest neighbors.

We add the instance filtering on the initial configuration, and this gives a clear improvement, as seen in Table 1. Just using the instance filtering, with no classifying stage, shows better results for the hi-hats, mainly because their inter-onset interval is too short to allow computing feature and they are segregated at the high frequencies in the spectrum. On the other hand, for bass and snare drums, there is an increase in the precision p, which leads to the conclusion that the instance filtering helps to decrease the numbers of false positives.

After testing the data, picking the best settings for each class, we want to validate on the large validation set. As seen in Table 1, the performance is similar to the one of the first dataset, with a slight drop in performance for the bass drum. This is related to the poor accuracy of the HFC onset detection in detecting the BD onsets (84%) and to the presence of low frequency tom drums, which results in low precision. However, the hypothesis that the algorithm performs better for the hi-hat just with IF is not confirmed. The classifier holds an important role in separating a large variety of sounds.

The last row of Table 1 presents the overall performance of the C+IF system. Additionally, we compute the standard deviation F_{std} [13, p 39] and the mean F_{mean} of the overall F-measure. We obtain $F_{std} = 0.14$ and $F_{mean} = 0.82$ for testing and $F_{std} = 0.17$ and $F_{mean} = 0.77$ for validation.

In conclusion, a system that relies only on sub-band onset filtering (IF) is unable to discriminate well between classes, and detects a large number of false positives. On the other hand, a system that uses only a KNN classifier is unable to react to the dynamics of a drum performance, especially to overlapping drum strokes. Using the IF stage along with the classifier, increased the precision and overall performance for all the classes.

4. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a real time drum transcription system, implemented in Pure Data. Our transcription system highlights that, under the constraints of a real-time audio, simple sub-band filtering can increase the system's performance by reducing the number of false positives. Moreover, we proposed the used of a class specific approach in order to maximize detection for each drum class. We tested and validated on a diverse collection of drum loops created with a variety drum kits in order to prove its robustness.Our approach

Our algorithm works in real-time as [6], [9] and [7], but it doesn't require adapting the model for a particular drum kit. The variety of the model (training, testing and validation data sets) is what distinguish our approach from the [19] (one drum kit for training and two for testing). We are not building decoy tail models as in [6], but we are using an instance filtering stage to filter these tails which overlap to the next sound.

A future step will be to train the model with additional sounds, because it is not adapted to detect other percussion instruments. Moreover, the instance filtering stage could adapt the parameters of the filters to the incoming signal, rather than using the pre-defined values. We can improve the robustness of onset detection by passing the incoming signal through a compressor and dc-blocking module. We can adjust the system's features by studying how intra-class correlation is influenced by the IF stage and by the selected features. Furthermore, we can look at the relation between inter-class microtiming and class-wise accuracy.

5. REFERENCES

[1] Marchini M. and H. Purwins, "An unsupervised system for the synthesis of variations from audio percussion patterns," in *7th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, 2010, pp. 277–278.

[2] D. Fitzgerald, R Lawlor, and E. Coyle, "Drum transcription using automatic grouping of events and prior subspace analysis," *Digital Media Processing for Multimedia Interactive Services - Proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services*, pp. 306–309, 2003.

[3] G. Tzanetakis, A. Kapur, and R. McWalter, "Subband-based Drum Transcription for Audio Signals," *2005 IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1–4, Oct. 2005.

[4] O. Gillet and G. Richard, "Automatic transcription of drum loops," *International Conference on Acoustics, Speech, and Signal Processing ICASSP04, Montral, Qubec, 2004*, pp. 269–272, 2004.

[5] O. Gillet and G. Richard, "Drum track transcription of polyphonic music using noise subspace projection," *Proceedings of the 6th International Conference on Music Information Retrieval*, pp. 92–99, 2005.

[6] E. Battenberg, V. Huang, and D. Wessel, "Toward live drum separation using probabilistic spectral clustering based on the itakura-saito divergence," in *AES 45th Conference: Applications of Time-Frequency Processing in Audio*, Helsinki, Finland, 2012.

[7] W. Brent, "Cepstral analysis tools for percussive timbre identification," *Proceedings of the 3rd International Pure Data Conference*, 2009.

[8] MIREX Contest Results, "Mirex contest results - audio drum detection, 2005," `http://www.music-ir.org/evaluation/mirex-results/audio-drum/index.html`.

[9] K. Tanghe, S. Degroeve, and B. De Baets, "An algorithm for detecting and labeling drum events in polyphonic music," in *Proceedings of the first Music Information Retrieval Evaluation eXchange (MIREX)*, 2005.

[10] J. Paulus and T. Virtanen, "Drum transcription with non-negative spectrogram factorisation," *13th European Signal Processing Conference*, 2005.

[11] K. Yoshii, M. Goto, and H. Okuno, "Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 1, pp. 333–345, Jan. 2007.

[12] P. Brossier, *Automatic annotation of musical audio for interactive applications*, Ph.D. thesis, Queen Mary, University of London, 2006.

[13] D. Hand, *Principles of data mining*, vol. 30, The MIT Press, Jan. 2001.

[14] F. Gouyon, P. Herrera, and A. Dehamel, "Automatic labeling of unpitched percussion sounds," in *AES 114th Convention*, 2003.

[15] M. Miron, M.E.P. Davies, and F. Gouyon, "An open-source drum transcription system for pure data and max msp, pure data source code and patches," `http://github.com/SMC-INESC/drumtranscription_pd/`.

[16] M. Miron, M.E.P. Davies, and F. Gouyon, "An open-source drum transcription system for pure data and max msp, max msp source code and patches," `http://github.com/SMC-INESC/drumtranscription_maxmsp/`.

[17] Timidity++, "Timidity++ software syntesizer," `http://timidity.sourceforge.net/`.

[18] Groove Monkee, "Groove monkee midi drum loops collection," `http://www.groovemonkee.com/en/`.

[19] O. Gillet and G. Richard, "Supervised and unsupervised sequence modelling for drum transcription," *Proceedings of the International Conference on Music Information Retrieval 2007*, pp. 219–224, 2007.