PERFORMING EXPRESSIVE RHYTHMS WITH BILLABOOP VOICE-DRIVEN DRUM GENERATOR

Amaury Hazan

Music Technology Group, IUA
Universitat Pompeu Fabra, Barcelona, Spain
ahazan@iua.upf.es

ABSTRACT

In a previous work we presented a system for transcribing spoken rhythms into a symbolic score. Thereafter, the system was extended to process the vocal stream in real-time in order to allow a musician to use it as a voice-driven drum generator. Extensions to this work are the following. First we achieved a study of the system classification accuracy based on typical onomatopoeia used in western beat boxing, with the perspective of building a general supervised model for immediate use. Also, we want the user to be able to generate expressive rhythms, beyond the symbolic drum representation. Thus we considered a class-specific mapping of continuous vocal stream descriptors with either effects or synthesis parameters of the drum generator. The extraction of the symbolic drum stream is implemented in the BillaBoop VST Core plug-in. The class-specific mapping and the sound synthesis are carried out in Plogue Bidule [1] framework. All these components are integrated into a low-latency application that allows its use for live performances.

1. INTRODUCTION

We present an application for generating expressive drum rhythms controlled by voice. The aim of this work is to develop a system able to reduce the gap between the user and a device, namely keyboard, drum pad or GUI, in order to generate on-the-fly synthetic rhythms using samples or some synthesis techniques. This is relevant as many musicians have just an intuitive experience of performing rhythm and can not easily communicate to a computer a beat they have in mind. Furthermore, in both non-western music, recent western urban genres and electronic music, the oral tradition of music and especially rhythm is predominant. In this paper, we are interested in western beat-boxing signals, that refer to a recent urban tradition. In recent works we built a system able to achieve on-the-fly vocal hits categorization into 3 drum sounds (namely Bass drum, Snare Drum, and Hi Hat), but the percussive vocabulary was restricted to [Poo, Tch, Tss] utterances. Indeed, typical beat boxing signals cover a wider range of utterances, and refer not only to acoustic instruments, but also to synthesized sounds or vocal utterances. We believe that by taking into account a wider variety of examples, despite the noise added to the training data, we can improve the overall robustness of the system. Thus, a training set was built based on tutorial recordings performed by the beat box performer TyTe, available on his website[2]. A taxonomy of the vocal sounds that were collected is presented in Section 4.

The other issue raised by this work is that even though each percussive event was perfectly segregated and transcribed, the resulting drum score would lack of information describing how the performer has modulated the produced sounds, a crucial point of oral performance expressiveness. Thus, a representation of oral percussion expressive effects (e.g. energy or centroid variation) has to be defined and effective computational methods to track them have to be used. In our case the voice would act as a drum trigger, but would also be able to control different aspects of the triggered sounds. The expressive class-specific mapping can be applied either to each one of the synthesizers parameters or to drum-specific effect controls. We present the different components of the system: descriptors generator, onset/offset detector, event segregation component, class-specific descriptors mapper, and finally drum generator. Figure 1 shows how theses components are integrated in order to provide a three drum output stream from a monophonic input. Note that the components that appear inside the dashed square belong to the BillaBoop Core, implemented as a VST plug-in, while the outer elements are implemented in Plogue Bidule.

Based on these ideas the outline of this paper is the following. In section 2 we present some work related to percussive voice analysis, drum sounds identification, and voice-controlled instruments. Section 3 presents onset detection and descriptors extraction techniques that meet the real time criteria, section 4 gives details about the training data used to build the classification model and then deals with supervised machine learning techniques that are used to process this data, section 5 presents the implementation of the whole system. Finally the results are discussed and work directions are given in section 6.

2. RELATED WORK

In [1], the proposed system segregates the vocal input stream into a four drum score that can be edited in order the generate sample triggered drum rhythms, and is extended in [2] to achieve real-time three drum rhythm generation. In [3, 4], a system for on-the-fly segmentation using state of the art onset detection techniques is presented. In [4] a first real-time categorization using the single spectral centroid feature of the first detected frame is used, which allows some live beat box control application. Finally, KTDrumTrigger [3] is a VST plug-in dedicated to extracting drum symbolic information from the audio stream. Its categorization process can be seen as semi-automatic because the user has to tune several analysis parameters in order to obtain the desired system behav-

[1] www.plogue.com
[2] www.humanbeatbox.com
[3] www.koen.smartelectronix.com/KTDrumTrigger

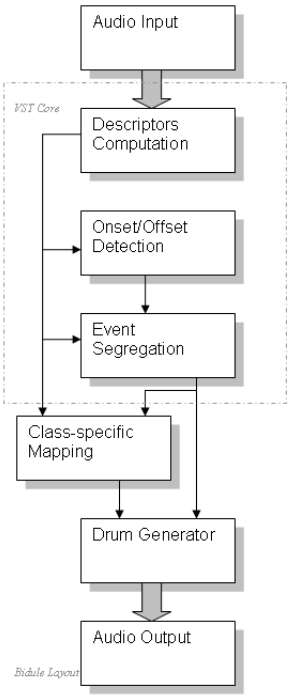

Figure 1: *Overview of Billaboop system.*

ior. Among all the works listed above the main aim is to obtain a symbolic rhythmic score without considering other aspects of the oral percussive signal. On the other hand some works consider the real-time tracking of continuous expressive features of the voice without taking into account events segregation. Kantos [4] is a commercial vintage synthesizer that can be controlled by the voice or any other acoustic signal. The user can edit the mapping between extracted acoustic features to some synthesis parameters. [5] uses novel descriptors that track [wah-wah] utterances in the vocal stream in order to drive a wah-wah effect.

3. DESCRIPTORS EXTRACTION AND ONSET/OFFSET DETECTION FOR PERCUSSIVE VOICE SIGNALS

In this section we present descriptors extraction and onset/offset detection components. We focus in this paper on voice percussive signals which are a bandwidth-limited and monophonic. Thus we designed a fast and simple component for this kind of source. In particular, we assumed that only one drum instrument could be triggered at the same time.

3.1. Descriptors

First, the input sound is analyzed frame by frame, and descriptors are computed in the time and frequency domain (after having performed a Fast Fourier Transform on the triangular-windowed temporal frame). We report here the descriptors which are used, but due to the lack of space, we do not present all of them formally. We refer to [6] for a detailed review. The only temporal descriptor we use is Zero Crossing Rate, which is calculated as the rate the signal changes sign over the frame duration. The other descriptors we center on are extracted in the frequency domain. First, High Frequency Content is the weighted area of the spectrum between 100 and 10000 Hz, which gives an increasing weight to increasing frequencies and lets high frequency variations be detected easily.

$$HFC = \sum_{k=k_{100}}^{k_{10000}} k|X(k)| \tag{1}$$

Spectral Centroid refers to the center of gravity of the spectrum, and is defined as:

$$Centroid = \sum_{k=0}^{N-1} \frac{kf_s}{N} \frac{|X(k)|}{\sum_{k=0}^{N-1}|X(k)|} \tag{2}$$

Spectral Roll-off, is the frequency in Hz for which 85% of the spectrum energy is contained below. It can be used to distinguish between harmonic and noisy sounds. On the other hand, as a results of a statistical analysis of the spectral data of the training recordings, we define three perceptual energy bands for vocal percussive sounds. E_{low} stands for the spectral energy between 100 and 2000 Hz, E_{med} refers to the spectral energy between 2000 and 6000 Hz, and E_{high} between 6000 and 10000Hz. These energies are defined as follows:

$$E(band) = \sum_{k=k_{lowfreq}}^{k_{upfreq}} |X(k)|^2 \tag{3}$$

where the indexes $upfreq$ and $lowfreq$ represent lower and upper band frequencies. For a given frequency f, we compute k_f, index of the spectral bin corresponding to f in equation *(1),(2),(3)* as follows:

$$k_f = \lceil \frac{fN}{f_s} \rceil \tag{4}$$

where N is equal to the number of points in the FFT, and f_s is the sample rate frequency. Finally we define to overall energy between 100 and 10000 Hz, $E_{tot} = E_{low} + E_{med} + E_{high}$

3.2. Onset detection

Most of state of the art onset detection algorithms are based on multi-band processing using psychoacoustic knowledge. A comparision of such algorithms can be found in [7]. We implemented an onset detection algorithm based on spectral variations between the former frame and the current one. We focus on two spectral variation features, namely High Frequency Content variation and overall Band Energy variation. They are defined as follows:

$$\Delta HFC[n] = \frac{HFC[n] - HFC[n-1]}{HFC[n-1]} \tag{5}$$

$$\Delta BandEnergy[n] = \frac{\sum_{bands} \Delta E_{band}[n]}{E_{tot}[n]} \tag{6}$$

where, for each of the three defined bands, we have:

$$\Delta E_{band}[n] = \frac{E_{band}[n] - E_{band}[n-1]}{E_{band}[n-1]} \tag{7}$$

In equations *(5),(6),(7)*, n stands for the index of the actual frame while n-1 is the index of the former frame. The onset detection algorithm outputs a state variable which can be equal to *silence*, *onset*, or *steady-state*. It proceeds as follows:

[4] http://www.antarestech.com/products/kantos.html

```
if(onset-state = silence):
    if(DeltaHFC > DeltaHFCThr AND
    DeltaBandEnergy > DeltaBandEnergyThr):
        onset-state = silence;
        end
    end
if(onset-state = onset):
    compute_label();
    onset-state = steady-state;
    end

if(onset-state = steady-state):
    if(DeltaHfc<-DeltaHFCOffThr):
        onset-state = silence;
        end
    end

if(onset-state = steady-state):
    output_descriptors_map();
else: output_descriptors_zeros();
    end

return onset-state;
```

In this algorithm *DeltaHFCThr*, *DeltaBandEnergyThr*, and finally *DeltaHFCOffThr* are thresholds defined by the user in order to modify attack and release sensibilities of the onset detector. Also, notice that we included in the algorithm the methods *compute-label*, *output-descriptors-map*, and *output-descriptors- zeros*. The first one is used to perform the event segregation task and is discussed in the next section, while the two latter are used to perform the class-specific mapping, that is, if *onset-state* equals *steady-state*, the descriptors are computed and are given as output of the VST Core plug-in, otherwise, zeros are given as output (See Section 5).

4. A SUPERVISED MODEL FOR EVENT SEGREGATION

The event segregation task has to be performed with the following restrictions: we focus on only 3 target classes, namely Bass Drum, Snare Drum, and Cymbal. The classification task only consider early descriptors of the hits to be classified, that is, the first frame peaked by the onset detector. Obviously, this task cannot be computationally expensive in order to meet the real-time restrictions. Furthermore, we look at models that can be easily translated into C programming language in order to be integrated into the system.

4.1. Collecting TyTe beat box training data

Defining an ontology of oral percussive sounds is a difficult task, because even restricting our study to western beat boxing signals, a wide range of sounds that are not formally defined is used in the performances. New sounds are introduced by successive performers and added to the cultural pool. As a first attempt, we collected sounds provided in the beat box tutorials provided by TyTe, a professional beat boxer. We report in Table 1 part the terminology of the drum sounds that can be found in this tutorial. Three subsets containing different vocal hits categories were chosen, and each one was labeled with a target class. In some cases sounds that do not refer to the same instrument (e.g. Splash Cymbal and Closed Hi-Hat) are labeled with a same class, here Cymbal.

Bass Drum	Dry Kick, Classic Kick, 808 Kick, Roll Kick
Snare Drum	Classic Snare, Brushed Snare, 606 Snare, 909 Snare, 808 Snare, Cough Snare, 808 Snare Roll, 808 Rimshot Snare
Cymbal	Splash Cymbal, Brushed Cymbal, Fast Hi-Hat, Slow Hi-Hat, Open Hi-Hat, Closed Hi-hat

Table 1: A subset of the vocal hits categories present in the training set with their assigned labels

C4.5	C4.5 w/boost	C4.5 w/bag
77.85	79.64	81.71

Table 2: 10 fold Cross Validation results of the data for the TyTe data set

Actually, among the presented sounds, some refer to acoustic instruments (e.g. Dry Kick), others refer to drum synthesizers (e.g. 606 snare), or are more directly related to the vocal aparatus (e.g. Cough Snare).

For each of these categories, some examples are available as isolated sounds, but for most of them the vocal hits of a given category are present in short beat box excerpts, involving different sounds. We prefer to study vocal hits that come from a vocal rhythm because the data is closer to real-world situations than the data that comes form isolated sounds. Consequently, we labeled manually each of the collected vocal hits. We collected a total of 321 vocal drum sounds, namely 132 hits for Bass Drum, 80 hits for Snare Drum, and 109 vocal hits for Cymbal. The descriptors used for the training vectors are those presented in subsection 3.1. except E_{tot} because we do not want to build a segregation model sensitive to the loudness of the hits.

4.2. Decision Tree Algorithms

As pointed out in the introduction, the data which is used to train the model is rather noisy, because the sounds to be labeled with the same class can be produced with different vocal effects. Tree induction algorithms build a IF-ELSE tree model by selecting at each node the most relevant attribute in a divide and conquer manner. This algorithm is well known for the efficiency of its pruning process and because it handles noisy data accurately. We compare the results of C4.5, C4.5 with boosting, and C4.5 with bagging. Other supervised algorithms were tested, but are not reported here due to the lack of space.

4.3. Classification results

Table 2 presents 10-fold cross validation results of the TyTe data set presented above. We compare the classification accuracy of C4.5, C4.5 with boosting , and C4.5 with bagging . The results appear to be encouraging compared with the baseline of 33%, however they need to be improved if we want to obtain a system able to analyze automatically all the percussive vocabulary of TyTe in real-time. Improvements are discussed in the last section.

5. IMPLEMENTATION

First, the components described in Section 3 and 4 are integrated into the BillaBoop VST Core plug-in. We used as a basis the im-

plementation presented in [5]. Through the VST interface, the user can adjust onset detection thresholds (namely *DeltaBandEnergyThr*, *DeltaHFCThr* and *DeltaHFCOffThr*). We implemented the segregation components using a C translation of the C4.5 decision tree induced in the Weka Framework[5]. We show in Figure 2 how the VST Core and the class-specific mapping components communicate.

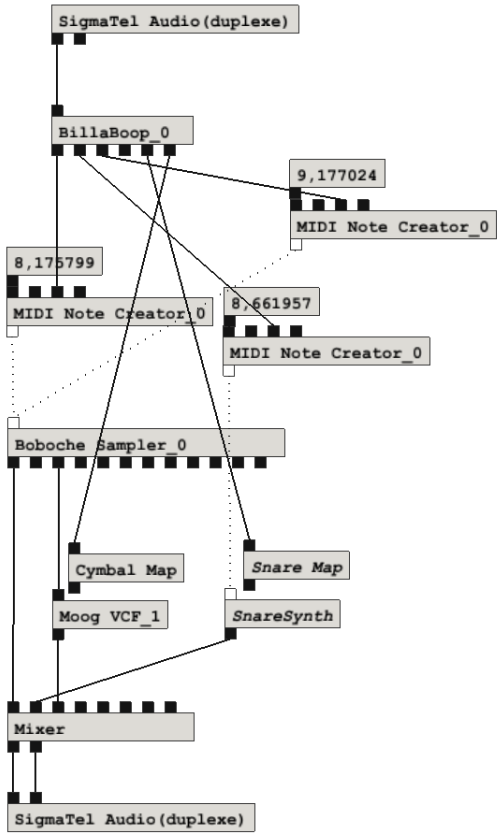

Figure 2: *A Possible layout in Plogue Bidule.*

The layout organization is the following. First, the incoming stream (here form Sigmatel sound card driver), goes to Billaboop VST. Here descriptors extraction, onset detection and event segregation tasks are achieved. Billaboop VST plug-in outputs the extracted symbolic information using the multi-channel audio output in the following way. Output ports 1 to 3 output 0s and 1s according to the current performed drum. For instance, a 1 is on the first output if the state is *steady-state* and the label is BD, 0 otherwise. Output port 4 to 6 contain descriptors of the vocal hit during onset and steady state (here Centroid and Roll-off), otherwise they output 0s. Port 1 to 3 goes to separate *Midi Note Creator* modules. Here Bass Drum and Hi-Hat are triggered with *Boboche Sampler*, while the Snare Drum is controlled by an analogue synthesizer. Ports 5 and 6 contain continuous descriptors values , the Centroid is mapped to the *Snare Synth* Cutoff synthesis parameter via *SnareMap* module, while the Roll-off is mapped to the *Moog VCF* Resonance parameter processing the Cymbal sample via *CymbalMap* module. Note that the links between mappers outputs and generator or effects do not appear in Bidule interface. The latency due to the hop size is 11 ms using a 1024 samples frame size with an overlap of 50%. Adding the latency of the Edirol FA-101 sound card we use (10 ms), we obtain 21 ms. Obviously, we should also add the processing time of the BillaBoop VST Core and the Plogue Bidule Host, which were not evaluated at the time of writing this paper. Nevertheless, the whole response time is fast enough to let the user interact naturally with the system.

[5] www.cs.waikato.ac.nz/∼ ml/weka/

6. DISCUSSION AND CONCLUSIONS

We designed and presented the components of a real-time application for expressive voice-driven drum generation. Although at early stages of implementation, the system already provides encouraging results and stimulating experiments, that are of interest for the design of future musical interfaces. Nevertheless, the choice to make onset detection and classification in one analysis frame is a strong constraint, and enhancing the accuracy of system for this setting may be a hard task. Further work includes a formal evaluation of both onset detection and classification components for different settings and performers. We also want to increase the amount of descriptors to be included in our model. For instance, descriptors derived from [5] concentrating on slope integrals of perceptually relevant frequencies of speech seems promising, but the information of more than one frame will be needed. We believe that if we reach these goals an expressive interface for the vocal percussive performer can be designed. The system and sound examples are available on: www.iua.upf.es/∼ahazan/BillaBoop.

7. ACKNOWLEDGEMENTS

This work was supported by the Spanish TIC project ProMusic (TIC-2003-07776-C02-01)

8. REFERENCES

[1] A. Hazan, "Towards automatic transcription of oral expressive performances," in *Proceedings of the Intelligent User Interfaces Conference (IUI 2005)*, 2005.

[2] A. Hazan, "Billaboop: Real-time voice-driven drum generator," in *Proceedings of Audio Engineering Society, 118th Convention*, 2005.

[3] P. Brossier, J. Bello, and M. Plumbley, "Fast labelling of notes in music signals," in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[4] N. Collins, "On onsets on-the-fly: Real-time events segmentation and categorization as a compositional effect," in *Proceedings of the First Sound and Music Computing Conference (SMC'04)*, 2004.

[5] A. Loscos and T. Aussenac, "The wahwactor: a voice controlled wah-wah pedal," in *Proceedings of New Interfaces for Musical Expression (NIME 2005)*, 2005.

[6] P. Herrera, A. Dehamel, and F. Gouyon, "Automatic labeling of unpitched percussion sounds," in *Proceedings of the 114th Audio Engineering Society Convention*, 2003.

[7] N. Collins, "A comparision of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions," in *Proceedings of the 118th Audio Engineering Society Convention*, 2005.