MODEL-BASED EVENT LABELING IN THE TRANSCRIPTION OF PERCUSSIVE AUDIO SIGNALS

Jouni Paulus, Anssi Klapuri

Institute of Signal Processing, Tampere University of Technology
P.O.Box 553, FIN-33101 Tampere, Finland
{paulus,klap}@cs.tut.fi

ABSTRACT

In this paper we describe a method for the transcription of percussive audio signals which have been performed with arbitrary non-drum sounds. The system locates sound events from the input signal using an onset detector. Then a set of features is extracted from the onset times. Feature vectors are clustered and the clusters are assigned with labels which describe the rhythmic role of each event. For the labeling, a novel method is proposed which is based on metrical (temporal) positions of the sound events within the measures. The system is evaluated using monophonic percussive tracks consisting of non-drum sounds. In simulations, the system achieved a total error rate of 33.7%. Demo signals are available at URL:<http://www.cs.tut.fi/~paulus/demo/>.

1. INTRODUCTION

The aim of this paper is to propose a method for transcribing percussive rhythms from audio signals into a symbolic representation. In particular, the idea is that the input rhythms can be performed using an arbitrary set of percussive sounds, for example by hand-tapping or pencil-clicking on different materials, or even by scat singing (imitating drum instruments by speech sounds). The output of the system consists of a sequence of time-stamped labels which can be further converted e.g. to a MIDI file. Due to the degrees of freedom in regard to the sound sets allowed, only three rhythmically different sounds are attempted to be recognized. These are referred to as *rhythmic role labels*, and are denoted as *B* (bass drum), *S* (snare drum) and *H* (hi-hat), according to the instruments that are usually used to play the roles of these labels in real world music.

A system of the described kind acts as a *user interface* for presenting musical rhythms to a computer. Specific musical instruments or musical education (e.g. score writing ability) is not necessary. Such a user interface has several applications. For example, it can be used to enter a query string to a music information retrieval system. A musician may use it to input a drum track to a score-writing program. A non-musician may want to create music by tapping a rhythm (and simultaneously singing a melody), and to have a computer program which listens and accompanies according to a particular musical style. The most intuitive method for presenting the rhythms would be tapping them with e.g. fingers and record the produced sound with a microphone. If some special hardware is needed for presenting the desired rhythmical patterns to the computer, the usability of the system degrades. Also, the extra hardware may be intimidating for the user. The task for the system is to somehow recognize and label the percussive sound events.

A rather constraining model with only three rhythmically different labels is proposed. Although this drops all timbral nuances of the audio signals, the basic rhythmic percept can be retained for a large body of music from different genres. As has been shown by Zils *et al.*, a drum track of popular music can be presented with very few actual elements occurring and still it will produce the same rhythmic percept as the original track [1]. Their system attempts to extract a drum track consisting of bass and snare drum occurrences. Here, the label *H* was added, because even though the rhythmic percept generated by only bass and snare drums are close to the original one, still something is missing. This missing part is in popular music often played by hi-hats, hence the label. The initial intuition predicts that this kind of model for rhythms suits for the genres pop, rock, blues and hip hop, and to some degree for electronic music.

To our knowledge, transcription of percussive tracks performed with arbitrary non-drum sounds has not been attempted before. Transcription of percussive tracks performed with actual drum instruments has been attempted e.g. by FitzGerald *et al.* who used sub-band independent subspace analysis in transcribing drum tracks consisting of kick drums, snare drums and hi-hats [2]. Their system used manually set rules in determining the correct naming of the found components. Performance of their system was quite reasonable (total success rate 89.5%) considering that the material was polyphonic. The weak point is that their system needed human interference and so the system can be used with one kind of a drum set only, not with arbitrary sounds. Virtanen used a data-adaptive sparse coding approach, where the cost function took temporal continuity into account in [3] when trying to separate different sound sources from a mixture. He tested the system in automatic drum transcription, trying to separate kick and snare drums from polyphonic mixtures. The total error rate for his system was 34%. The weakness in that system was that it needed spectral templates to identify the separated sources. Also hi-hats were not separated due to their relatively weak energy.

When normal drum sounds are used, an acoustic-model based approach can be used. Earlier we presented a system for transcribing polyphonic drum tracks of real drums with the aid of simple

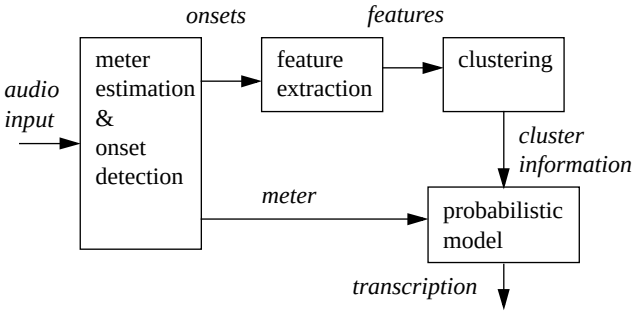

Figure 1: *System overview.*

acoustic models and *N*-gram based language models in [4]. But when using arbitrary sounds in the input, acoustic information it is not enough to identify the sounds. In [5] Patel and Iversen have studied the North Indian tabla drumming tradition in which a system of nonsense syllables is used to name drum sounds. They wanted to know if there exists an acoustic and perceptual basis for the mapping from drum sounds to these syllables. As a result they found that there exists acoustic features, such as spectral centroid and decay time, that can be used in the mapping when used in relation to each other. In their perceptual tests, people unfamiliar with tabla drumming were able to create the mapping quite well when sets of two syllable sounds and corresponding tabla sounds were presented to them. This method works when individual sounds need to be labeled, but the metrical information information tends to overrule the acoustic information when dealing with rhythmical patterns. This is partly due to the diversity of the possible sound sets.

2. PROPOSED METHOD

Overview of the proposed system is shown in Figure 1. The bottom-up clustering part of the system resembles that of Herrera *et al.* where drum tracks consisting of hi-hats, bass and snare drums are automatically labeled. Their system analyses the signal using a constant temporal grid, extracts features at each grid point and finally clusters the extracted features [6].

Another system which uses percussive sound clustering is that of Wang *et al.* which detects percussive sounds in a musical piece and then clusters them into as many clusters as needed according to their perceptual similarity. The obtained cluster information is then used in reconstructing acoustic signal in the case of a packet loss in the transmission of an encoded signal [7].

In our system, features are extracted only at the beginnings of detected sound events, similarly to [7]. For this purpose, onset detection is performed using the mid-level representation of the system presented in [8]. At each onset location a small frame of the signal is extracted and analysed with a method similar to [6]. The analysis result, i.e. the clustering information, is then inserted to a grid of tatum pulses according to the timing information resulting to a symbolic representation. As the crucial step, this information is fed to the labeling system, which then uses a simple probabilistic model in determination the labeling. The term *labeling* is used to refer to the naming of the created clusters using the limited set of rhythmic role labels available. If the clustering was not accomplished totally correct, a simple algorithm for post-labeling cluster assignment changes is applied.

2.1. Meter estimation and sound onset detection

Temporal segmentation in the proposed system is done using the musical meter estimator described in [8]. *Meter* refers to the temporal regularity of music signals, consisting of pulse sensations at different levels. The applied meter estimator analyses meter at three different time scales. *Beat* (foot tapping rate) is the most prominent level. *Tatum* (time quantum) refers to the shortest durational values that are still more than incidentally encountered. The other durational values, with few exceptions, are integer multiples of the tatum. Musical *measure* is related to the harmonic change rate and to the length of rhythmical patterns in music. The accuracy of the meter estimator has been evaluated in [8] and it was found to be applicable in music from different genres.

Here, information about the temporal structure is used for two purposes. First, the bottom-up feature extraction takes place only at the instants of detected *onsets*. The feature vectors are then clustered further as will be described in Sec 2.2 and 2.3. Secondly, a subsequent probabilistic model uses the metrical positions of the sound events belonging to each cluster to infer the rhythmical role (label) of each cluster. This is described in more detail in Sec. 2.4. A discrete time grid of equidistant tatum pulses is created using the information extracted from the signal.

2.2. Feature extraction

From the location of each detected sound onset, a part of the signal is extracted using a Hanning window. Since the sound events are limited in time by their nature, a rectangular window could also be used. The length of the window is to the next detected onset, but 100 ms at maximum. From each frame temporal centroid, signal crest factor, signal energy, spectral kurtosis and six MFCCs (Mel-frequency cepstral coefficients) are extracted. The zeroth MFCC is not used. Finally the features are normalized to have zero mean and unity variance over time.

2.3. Clustering of sound events

The normalized feature vectors are clustered with fuzzy K-means algorithm. Like with its crisp version, the number of desired clusters is set manually. The term *crisp* is used here as the opposite of fuzzy. In addition to the normal cluster information, the algorithm also calculates for each data point degrees of membership to each cluster. The data point is assigned to the cluster to which it has the largest membership value. When left to this state, the clustering result is similar to the one produced by the crisp version. The membership values are used in the post-labeling enhancement.

The result after clustering is a sequence of time stamped cluster numbers $c_t \in \{1, 2, \ldots, K\}$, where K denotes the total number of clusters and t is the continuous time index over the signal. Each cluster number c_t is assigned to the nearest grid point. If there is no cluster number assigned to a certain grid point, it will contain the number 0. The resulting grid contains the cluster numbers $c_i \in \{0, 1, 2, \ldots, K\}$, where i is discrete time index over the signal in steps of one tatum. In further steps, when using

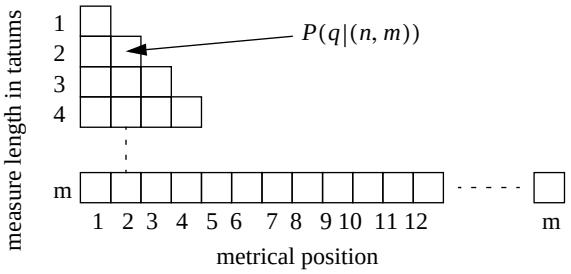

Figure 2: *Data representation of the probabilistic model. q denotes the label, n metrical position, m length of the measure in tatums, and P the probability of the label q to be present at the position* (n, m).

the term cluster numbers, the set $\{0, 1, 2, \ldots, K\}$ is referred.

The degrees of cluster memberships after time quantization, are store in a matrix U, where the element $(U)_{i,k}$ is the degree of membership of the data point at the time i in the cluster number *k*. On the locations *i* where no onsets was assigned to, the matrix U contains the value 1 in location corresponding to the cluster number $k = 0$, and 0 on the locations corresponding to the other clusters. Similarly, the value 0 is set to the location corresponding to the cluster number $k = 0$ on the locations *i* where any other cluster number was assigned.

2.4. Probabilistic model

The remaining problem is to find a way for labeling the clusters, i.e. to find a mapping $L:\{0, 1, 2, \ldots, K\} \rightarrow \{\varnothing, B, H, S\}$ from cluster numbers k to labels $q \in \{\varnothing, B, H, S\}$. The label $\varnothing$ is directly mapped to the cluster number representing silence, i.e. $k = 0$. The rest of the mapping could be created with the methods of acoustic pattern recognition, e.g. a Gaussian mixture model. The main problem with acoustic models is that they cannot generalize to extreme cases like the sound set variation considered here. If the input signals are performed with drum instruments, an acoustic model based recognition system can be constructed, as we demonstrated in [4]. As the aim was to be able to handle and label any percussive tracks, independently of the used instruments, acoustic models need to be set aside. Instead, a method relying on the timing of sound events within patterns is introduced.

The model estimates the probability of a certain rhythmic label *q* to be present at a certain time index within a measure, when the time is discretized to steps of one tatum. An illustration of the data structure of the matrix of probability values is in Figure 2. The measure length in tatum units is *m* and $n \in \{1, 2, \ldots, m\}$ denotes the position of the sound event within the measure. Since there exists numerous different musical time signatures, the measure lengths from 1 to 48 were modeled.

2.4.1. Model estimation

Probabilities for each label *q* to occur at different metrical positions (n, m) were estimated using a commercially available MIDI database Drumtrax 3.0. The database covers most of the western musical genres containing 359 performances in total.

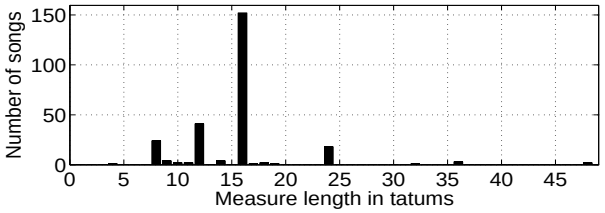

Figure 3: *Number of songs in the training set having a certain length of measure in tatums.*

The pieces are organized in 14 different categories, of which 13 are different genres and the last one is a kind of a tool box. Varied subset of 26 pieces (two from each genre) was left for the test set and the rest were used in training of the model. The division was done for ten times and the presented results are averaged over all tests. In probability estimation the musical grid of songs was annotated by hand and notes are distributed to this discrete time grid. Then each MIDI drum instrument is assigned to belong to one label q. The grid is divided to measures and they are handled individually. The number of occurrences of each label in each metrical position are calculated to a data structure seen in Figure 2, and the resulting probabilities estimated.

Though the used training set contained quite an extensive range of pieces, not all probabilities could be determined due to the lack of proper data. The number of songs in the training subset of the database, having a certain measure length in tatums can be seen in Figure 3. This zero occurrence problem was handled by applying Witten-Bell smoothing. An example of the produced probabilities can be seen in Figure 4.

2.4.2. Model usage

Each of the cluster numbers *k* is mapped to one of the rhythmic role labels *q*. After assigning a mapping, the resulting probability can be calculated with

$$P(L) = \prod_i P(q_i|(n_i, m)), \quad (1)$$

where *i* denotes the discrete time index, q_i the label assigned to that position using the mapping L, n_i the metrical position of the time index *i* within the measure of the length *m*. Because of the small number of the clusters and possible labels, every possible mapping permutation can be calculated in *brute force* and the optimal one found. The optimal labeling is defined to be the one having the largest total probability. This labeling strategy is based on the assumption that a majority of events are correctly clustered.

2.5. Post-labeling cluster changes

Since it is more than likely that not all of the data points are clustered correctly in a realistic situation, a simple method for enhancing the final result is presented. The basis for the changes is the cluster membership values from the fuzzy K-means. It is assumed that most of the data points is assigned to a correct cluster and hence only small changes are allowed. Also, it is assumed that the chosen mapping L' is the correct one and it is not changed in the algorithm. It should be noted that the algorithm can change only the data points that contained an onset, i.e. not

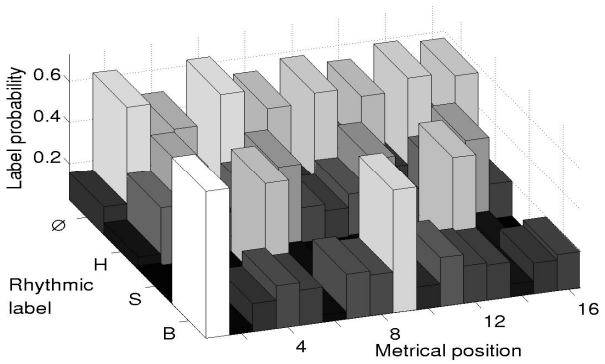

Figure 4: *An example of calculated label probabilities. The measure length is 16.*

the ones containing the label $\varnothing$. It proceeds as follows:

```
calculate base line probability P_B(L') with Eq. (1)
for i ∈ {grid points in piece}
  for k ∈ {0, 1, …, K}
    if (U)_{i,k} > u_min
      assign label q_i to be the one mapped from cluster k
      calculate total probability P_T(L') using the Eq. (1)
      if P_T(L') > P_B(L')
        retain the change
      else
        discard the change
      endif
    endif
  endfor
endfor
```

The membership value limit u_{min} is in the range $0 \ldots 1$. It has the effect that the smaller it is, the less the system trusts the clustering result. If it set to 0, the system can change any cluster assignment. When the total number of clusters is 4, including the silence, the best working value for u_{min} was found to be ca. 0.10. Value this small allows the algorithm to change ca. 40% of label assignments, where a non-silence label was set. The number of actually changed labels is very small, less than 5% in most cases.

3. SYSTEM EVALUATION

The system was evaluated with simulations. The input to the system, audio tracks with percussive sounds was produced by synthesizing 30 seconds from each of the MIDI pieces left to the test set. The test and train set division was done randomly for ten times and the presented results are the average of all divisions. The evaluation could be done automatically since the reference was obtained from the MIDI piece.

In all simulations, the metrical information was annotated by hand instead of using the musical meter estimator. This was done because of the need for automatic transcription evaluation. Since the reference data was annotated in MIDI files, it was decided to use the annotated metrical information. The accuracy of the musical meter estimator was evaluated in [8].

3.1. Audio synthesis

Since the aim was away from the more traditional drum tracks towards tracks performed with arbitrary percussive sounds, a specific synthesizer is needed. It was constructed by recording total of 68 different sounds that can be thought to be used in a real situation. For each sound 15 repetitions were recorded to guarantee some degree of variation in the synthesis. The sounds were distributed so that 48 of them were produced by tapping with hands or pen to tables, books, coffee mugs etc., or by foot tapping with different footwear. The remaining 20 sounds were speech sounds by two persons, both performing the same 10 sounds. From the recorded samples, five sets of percussive sounds were constructed. Two of these set consist only sounds produced with speech and the rest consist of clicks and tapping sounds. It should be noted that since no acoustic modeling was done, there was no need to do any division to train and test sets.

The synthesis produces monophonic signal, i.e. only one sound is playing at a time. In a case where more than one sound was to be played simultaneously, the one to be played was chosen by a simple priority scheme where the sound mapped to from MIDI notes to label *S* have the highest priority, *B* the second highest, and *H* the lowest.

3.2. Simulation setups

Since the total system performance depends heavily on the successive sub-blocks, it was decided to test it in four individual steps:

1. The performance of *onset detection and clustering accuracy.* In this setup, the system performs onset detection, feature extraction and clustering. The clusters are assigned with labels manually, so that the error rate is minimized.
2. Test if the *rhythmic role labeling* is theoretically possible based on the metrical position of events. In this setup, the system was given the reference transcription except that the sound labels were hidden and had to be inferred using the method described in Sec. 2.4.
3. The performance of the *whole system without post-labeling cluster changes*. Here, the system is given only acoustic signal and the metrical information and it needed to detect onsets, extract features, perform clustering and infer the labeling.
4. The performance of the *whole system with post-labeling cluster changes*. This setup is similar to the step 3, but the post-labeling cluster change algorithm is also used.

Each of these cases are evaluated using the same test material and it can be determined which parts of the system may need further development. The used error rate measure was

$$e = \frac{\sum_i f(q_i^{\text{ref}}, q_i^{\text{trans}})}{\sum_i 1}, \tag{2}$$

where *i* is the discrete time index over the whole signal and

$$f(q^1, q^2) = \begin{cases} 1, \text{ if } q^1 \neq q^2 \\ 0, \text{ otherwise} \end{cases}. \tag{3}$$

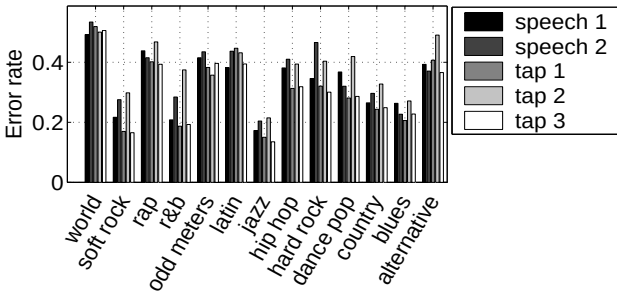

Figure 5: *Genre error rate averages for each of the used five sound sets.*

3.3. Simulation results

The simulation results for the four individual steps are in the Table 1. In addition to the total error rate over all sound sets, there is also the average error rates of speech based sound sets and tapping based sound sets. More detailed error rate statistics for each of the 13 genres and five sets can be seen on Figure 5.

As from the results can be seen, it seems that sounds produced with speech are more difficult for the system to cope with. When inspecting the error rates for each genre, seems that soft rock, r'n'b and jazz are the easiest ones for the system while alternative and world genres are the most difficult. The within genre variations were large, mostly caused by the random choice of the part of the pieces to be analysed. The post-labeling cluster change algorithm turned out to have a very small effect to the total performance. This is due to the fact that errors in labeling step cause high total error rate and minor changes can not fix enough errors.

Some demo signals from the simulations are available at URL:<http://www.cs.tut.fi/~paulus/demo/>.

Table 1: *Error rates for different test steps for speech sounds (average of two sets), tapping sounds (average of three sets) and the average of all systems.*

test step	speech ER	tapping ER	total avg ER
1. clustering	15.28%	13.12%	13.98%
2. role labeling	27.91%	27.91%	27.91%
3. whole w/o post fix	34.91%	33.15%	33.85%
4. whole w/ post fix	35.68%	33.01%	33.67%

4. DISCUSSION

In generation of the *tatum grid*, the tempo is assumed to be constant over the whole signal. Similarly, it is assumed that the length of the measure is constant. However, it is very likely that these assumptions do not hold when operating with real world signals. Musicians may vary the tempo when playing, creating their own versions of the piece and the time signature may be different in the verse and chorus of the piece.

When *constructing the models*, fixed assignment from MIDI notes to the rhythmic role labels turned out to generate problems. Though in many cases the rhythmic roles are operated by the specified drum instruments, this was not the case with all genres. Especially with jazz, latin and world music genres, more exotic percussions were used or the instrument was in a different role (e.g. cymbal in rock is usually in the "snare" role whilst in jazz it may be in the "bass" role). This caused problems in automatic handling of the test data. It turned out that on most of the cases where labeling after "perfect clustering" was erroneous, the time signature was such that there was only few other pieces available for training material.

5. CONCLUSIONS

This paper has introduced a method for transcribing percussive audio signals consisting of arbitrary sounds. It uses the statistical dependencies of metrical positions of rhythmical elements in labeling of the events. The efficiency of the method was evaluated with simulations. The transcription for arbitrary sound sets is a difficult task, and especially errors in label assignment increase the error rate significantly compared to the errors in the clustering step. Based on the results, using metrical positions in the rhythmic role labeling is possible to some degree.

6. REFERENCES

[1] Zils A., Pachet F., Delerue O., Gouyon F., "Automatic Extraction of Drum Tracks from Polyphonic Music Signals", in *Proc. 2nd Int. Conference on Web Delivering of Music (WedelMusic2002)*, Darmstadt, Germany, pp. 179-183, 2002.

[2] FitzGerald D., Coyle E., Lawlor B., "Sub-band Independent Subspace Analysis for Drum Transcription", in *Proc. 5th Int. Conference on Digital Audio Effects (DAFX-02)*, Hamburg, Germany, pp. 65-69, 2002.

[3] Virtanen T., "Sound Source Separation Using Sparse Coding with Temporal Continuity Objective", in *Proc. of International Computer Music Conference (ICMC2003)*, Singapore, 2003, to appear.

[4] Paulus J., Klapuri A., "Conventional and Periodic N-grams in the Transcription of Drum Sequences", in *Proc. of IEEE International Conference on Multimedia and Expo (ICME03)*, Baltimore, USA, pp. 737-740, 2003.

[5] Patel A. D., Iversen J. R., "Acoustic and Perceptual Comparison of Speech and Drum Sounds in the North Indian Tabla Tradition: An Empirical Study of Sound Symbolism", in *Proc. 15th International Congress of Phonetic Sciences*, Barcelona, Spain, 2003, to appear

[6] Gouyon F., Herrera P., "Exploration of techniques for automatic labeling of audio drum tracks' instruments", in *Proc. MOSART: Workshop on Current Directions in Computer Music*, Barcelona, Spain, 2001.

[7] Wang Y., Tang J., Ahmaniemi A., Vaalgama M., "Parametric Vector Quantization for Coding Percussive Sounds in Music", in *Proc. IEEE International Conference on Acoustics, Speech & Signal Processing (ICASSP03)*, Hong Kong, pp. 652-655, 2003.

[8] Klapuri A., "Musical Meter Estimation and Music Transcription", Paper presented at the Cambridge Music Processing Colloquium, Cambridge University, UK, 2003.