UNSUPERVISED GENERATION OF PERCUSSION SOUND SEQUENCES FROM A SOUND EXAMPLE

Marco Marchini
Music Technology Group
Universitat Pompeu Fabra
marco.marchini3@gmail.com

Hendrik Purwins
Music Technology Group
Universitat Pompeu Fabra
hendrik.purwins@upf.edu

ABSTRACT

In this paper we present a system that learns rhythmic patterns from drum audio recording and synthesizes music variations from the learnt sequence. The procedure described is completely unsupervised and embodies the transcription of a percussion sequence into a fuzzy multilevel representation. Moreover, a tempo estimation procedure identifying the most regular subsequence is used to guarantee that the metrical structure is preserved in the generated sequence. The final synthesis is performed, recombining the audio material derived from the sample itself. Some examples of generations along with a descriptive evaluation are provided.

1. INTRODUCTION

During the last two decades much effort has been devoted to build computational architectures of musical sequence learning [1, 2]. The result of this research in musical intelligence has often inspired music psychology experiments. One example is the Continuator [3] that has been used to study childhood flow-experience [4]. Moreover, these systems naturally lead to the philosophical question about the nature of ”style” in music. The problem has been attacked from many perspectives but the debate among musicologists remains open. Mayer [5] arrives at the conclusion that style is not only a complex concept originating from the interplay of different description levels of a musical piece, but also it is impossible to separate ”style” from the social context in which the music has grown.

Many studies have been conducted for the analysis and the generation of music sequences. In particular, in [3], a MIDI-based system for real-time musical interaction was developed, yielding good jazz style music generation.

The handling of memory is a core challenge in music modeling [6]. Whereas the widely used bag-of-features approach neglects any sequential relations between musical events, common n-gram based methods for the representation of musical sequences usually set a maximal fixed length of context. This leads to exponentially growing storage needs to allow the model to account for more complex structures. A solution to this dilemma is offered by the different length Markov model [7]. This model determines the needed context length for each musical sequence individually, therefore maximizing storage economy, without the requirement of excessive storage.

Focusing on the question opened by machine listening systems from an information theory point of view suggests improvements to MIR techniques. In the bag-of-frames approach the distance between two audio signals is independent of the order of the notes. Since most of the musical content generally resides on the temporal organization of the sound material, essential information about the music is lost in the derived descriptors. In fact, the goal of musical intelligence systems is to learn music excerpts in a similar way as the brain does in the first auditory scene analysis process [8]. Thus, these systems indirectly define an operative notion of style.

From a statistical point of view, ”style” can be defined as a source of symbols [9]. Equivalently, we can say that, in this context, understanding the style means to find a way to compress [1] the message [10]. Another approach is in the framework of information dynamics (see [11]) in which the analysis of music is related to music cognition. Moreover in [12, 13], causal systems are proposed to capture the emergence of musical categories during the listening process.

Employment of machine learning techniques in generating musical events is crucial to achieve flexibility with respect to different musical contexts. In its architecture, our system is inspired by cognitive principles. In addition, it can be used as a validator for many of the results in music analysis in the way that the quality of the synthesis reveals if the analysis methods use to generate the synthesis have been adequate.

First, we define the system design and the interaction of its parts. Starting from low-level descriptors, we translate them into a “fuzzy score representation”, where two sounds can either be discretized yielding the same symbol or yielding different symbols according to which level of interpretation is chosen (Section 2). Then we perform skeleton subsequence extraction and tempo detection to

[1] This means to find a concise representation of the signal without loosing information from the original (lossless data compression). In this way we can store a message (a sequence of symbols) using less bits and then rebuild the original signal by *uncompressing* its shorter version. The complexity of the message turns out to be the key concept that determines the compression ratio (the ratio between the bits occupied by the original message and the ones occupied by the compressed message). The Lempel-Ziv algorithm is an example of such a compressor.

align the score to a grid. At the end, we get a homogeneous sequence in time on which we perform the prediction. For the generation of new sequences we reorder the parts of the score, respecting the statistical properties of the sequence while at the same time maintaining the metrical structure (Section 3). In Section 4, we give a descriptive evaluation of the generated result.

2. UNSUPERVISED SOUND ANALYSIS

The system architecture consists of the following processing stages (cf. Figure 1):

- Segmentation
- Symbolization
  - Feature extraction
  - Feature clustering
  - Sequence structure analysis
  - Temporal alignment
- Generation of audio
  - Adaptive cluster level determination

We will now describe each step of the process in detail.

2.1 Segmentation

First, the audio input signal is analyzed by an onset detector that segments the audio file into a sequence of musical *events*. Each event is characterized by its position in time (onset) and an *audio segment*, the audio signal starting at the onset position and ending at the following contiguous onset. In the further processing, these events will serve two purposes. On one side, the events are stored as an indexed sequence of audio fragments which will be used for the re-synthesis in the end. On the other side, these events will be compared with each other to generate a reduced score-like representation of the percussion patterns to base a tempo analysis on (cf. Fig. 1 and Sec. 2.2).

We used the onset detector implemented in the MIR toolbox [14] that is based only on the energy envelope, which proves to be sufficient for our purpose of analyzing percussion sounds.

2.2 Symbolization

We will employ segmentation and clustering in order to transform the audio signal into a discrete sequence of symbols (as shown in Fig. 3), thereby facilitating statistical analysis. However, some considerations should be made.

As we are not restricting the problem to a monophonic percussion sequence, non-trivial problems arise when one wants to translate a sequence of events into a meaningful symbolic sequence. One would like to decide whether or not two sounds have been played by the same percussion instrument (e.g. snare, bass drum, open hi hat…) and, more specifically, if two segments *contain* the same sound in case of polyphony. With a similarity distance we can derive a value representing the similarity between two sounds but when two sounds are played simultaneously a different sound may be created. Thus, a sequence could exist that allows for multiple interpretations since the system is not able to determine whether a segment contains one or more sounds played synchronously. A way to avoid this problem directly and to still get a useful representation is to use a *fuzzy representation* of the sequence. If we listen to each segment very detailedly, every segment may sound different. If we listen very coarsely, they may all sound the same. Only listening with an intermediate level of refinement yields a reasonable differentiation in which we recognize the reoccurrence of particular percussive instruments and on which we can perceive meaningful musical structure. Therefore, we propose to maintain different levels of clustering refinement simultaneously and then select the level on which we encounter the most regular non-trivial patterns. In the sequel, we will pursue an implementation of this idea and describe the process in more detail.

2.2.1 Feature Extraction

We have chosen to define the *salient part* of the event as the first 200 ms after the onset position. This duration value is a compromise between capturing enough information about the attack for representing the sound reliably and still avoiding irrelevant parts at the end of the segment which may be due to pauses or interfering other instruments. In the case that the segment is shorter than 200 ms, we use the entire segment for the extraction of the feature vector. Across the salient part of the event we calculate the Mel Frequency Cepstral Coefficient (MFCC) vector frame-by-frame. Over all MFCCs of the salient event part, we take the weighted mean, weighted by the RMS energy of each frame. The frame rate is 100 frame for second, the FFT size is 512 samples and the window size 256.

2.2.2 Sound Clustering

At this processing stage, each event is characterized by a 13-dimensional vector (and the onset time). Events can thus be seen as points in a 13-dimensional space in which a topology is induced by the Euclidean distance.

We used the *single linkage* algorithm to discover event clusters in this space (cf. [15] for details). This algorithm recursively performs clustering in a bottom-up manner. Points are grouped into clusters. Then clusters are merged with additional points and clusters are merged with clusters into super clusters. The distance between two clusters is defined as the shortest distance between two points, each in a different cluster, yielding a binary tree representation of the point similarities (cf. Fig. 2). The leaf nodes correspond to single events. Each node of the tree occurs at a certain height, representing the distance between the two child nodes. Figure 2 (top) shows an example of a clustering tree of the onset events of a sound sequence.

The height threshold controls the (number of) clusters. Clusters are generated with inter-cluster distances higher than the height threshold. Noting that two thresholds lead to the same cluster configuration if and only if their values are both within the range delimited by the previous lower node and the next upper node in the tree. It is therefore

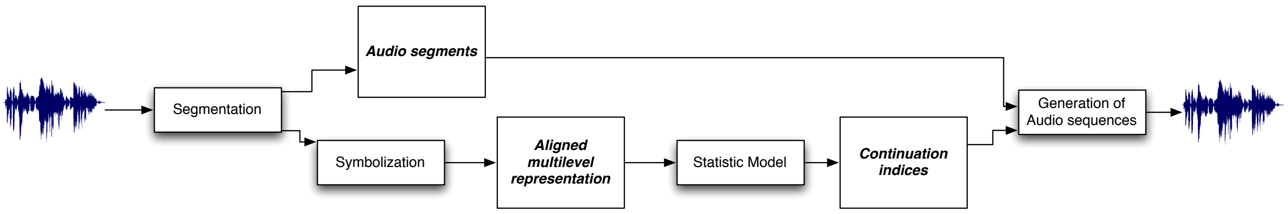

Figure 1. General architecture of the system.

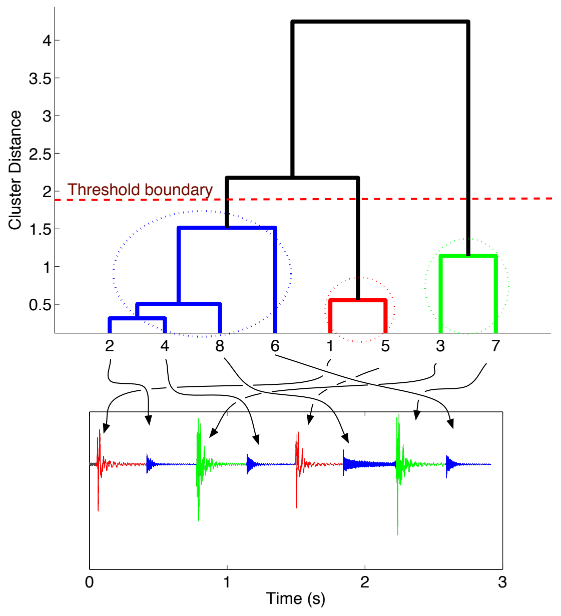

Figure 2. A tree representation of the similarity relationship between events (*top*) of an audio percussion sequence (*bottom*). The threshold value chosen here leads to a particular cluster configuration. Each cluster with more than one instance is indicated by a colored subtree. The events in the audio sequence are marked in the colors of the clusters they belong to. The height of each node is the distance (according to the single linkage criterion) between its two child nodes. Each of the leaf nodes on the bottom of the graph corresponds to an event.

evident that by changing the height threshold, we can get as many different cluster configurations as the number of events we have in the sequence. Each cluster configuration leads to a different symbol alphabet size and therefore to a different symbol sequence representing the original audio file. We will refer to those sequences as *representation levels* or simply *levels*. These levels are implicitly ordered. On the leaf level at the bottom of the tree we find the lowest inter-cluster distances, corresponding to a sequence with each event being encoded by a unique symbol due to weak quantization. On the root level on top of the tree we find the cluster configuration with the highest inter-cluster distances, corresponding to a sequence with all events denoted by the same symbol due to strong quantization. Given a particular level, we will refer to the events denoted by the same symbol as the *instances of that symbol*. We do not consider the implicit inheritance relationships between symbols of different levels.

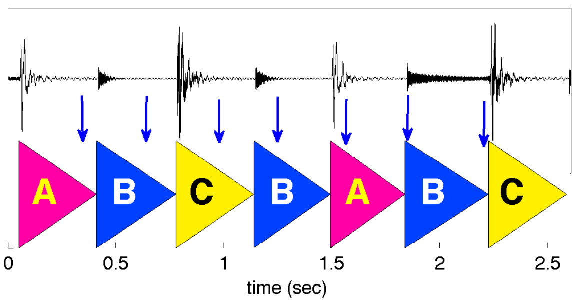

Figure 3. A continuous audio signal (*top*) is discretized via clustering yielding a sequence of symbols (*bottom*). The numbers inside the colored triangles denote the cluster index of the event, related to the type of sound, i.e. bass drum, hi-hat, or snare.

2.3 Level Selection

Handling different representations of the same audio file in parallel enables the system to make predictions based on fine or coarse context structure, depending on the situation. As explained in the previous section, if the sequence contains n events the number of total possible distinct levels is n (see Fig. 4). As the number of events increases, it is particularly costly to use all this levels together because the number of levels also increases linearly with the number of onsets. Moreover, as it will be clearer later, this representation will lead to over-fitted predictions of new events.

This observation leads to the necessity to only select a few levels that can be considered representative of the sequence in terms of structural regularity.

Given a particular level, let us consider a symbol σ having at least four instances but not more than 60% of the total number of events and let us call such a symbol an *appropriate symbol*. The instances of σ define a subsequence of all the events that is supposedly made of more or less similar sounds according to the degree of refinement of the level. Let us just consider the sequence of onsets given by this subsequence. This sequence can be seen as a set of points on a time line. We are interested to quantify the degree of temporal regularity of those onsets. Firstly, we

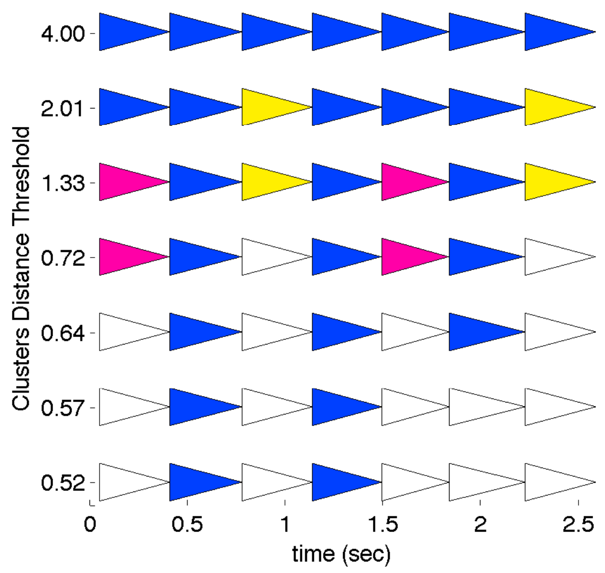

Figure 4. A sequence is displayed in a multi-level representation. Each color represents a unique symbol, a non-trivial cluster, whereas the singletons not belonging to a non-trivial cluster are drawn in white. Note how the number of different colors increases from top to bottom, indicating that the sounds are represented in greater refinement by a larger number of clusters.

computed the histogram [2] of the time differences (CIOIH) between all possible combinations of two onsets (*middle* Fig. 5). What we obtain is a sort of harmonic series of peaks that are more or less prominent according to the self-similarity of the sequence on different scales. Secondly, we compute the autocorrelation ac(t) (where t is the time in seconds) of the CIOIH which, in case of a regular sequence, has peaks at multiples of its tempo. Let t_{usp} be the positive time value corresponding to its upper side peak. Given the sequence of m onsets $x = (x_1, \ldots, x_m)$ we define the *regularity* of the sequence of onsets x to be:

$$\text{Regularity}(x) = \frac{\text{ac}(t_{usp})}{\int_0^{t_{usp}} \text{ac}(t)dt} \log(m)$$

This definition was motivated by the observation that the higher this value the more equally the onsets are spaced in time. The logarithm of the number of onsets was multiplied to the ratio to give more importance to symbols with more instances.

Then we extended, for each level, the regularity concept to an overall *regularity of the level*. This simply corresponds to the mean of the regularities for all the appropriate symbols of the level. The regularity of the level is defined to be zero in case there is no appropriate symbol.

After the regularity value has been computed for each level, we yield the level where the maximum regularity is reached. The resulting level will be referred so as the *regular level*.

We also decided to keep the levels where we have a local maximum because they generally refer to the levels

[2] We used a discretization of 100 ms for the onset bars.

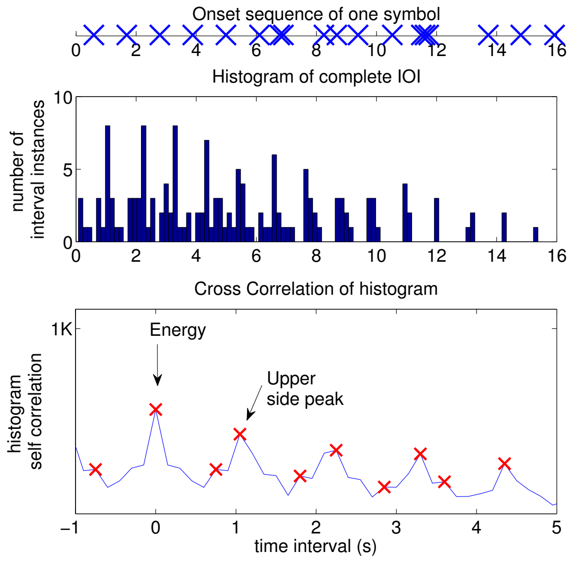

Figure 5. The procedure applied for computing the regularity value of an onset sequence (*top*) is outlined. *Middle:* the histogram of the complete IOI between onsets. *Bottom:* the autocorrelation of the histogram is shown for a subrange of IOI with relevant peaks marked.

where a partially regular interpretation of the sequence is achieved. In the case where a sequence of consecutive levels share the same regularity only the higher one is kept. Figure 6 shows the regularity of the sequence for different levels.

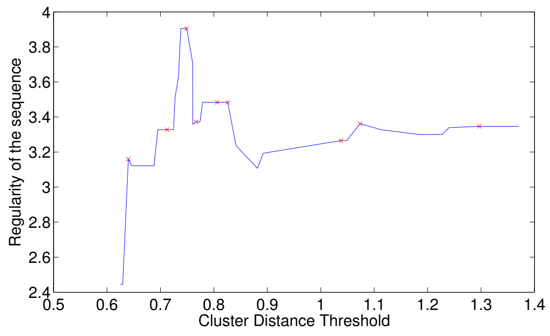

Figure 6. Sequence regularity for a range of cluster distance thresholds (x-axis). An ENST audio excerpt was used for the analysis. The regularity reaches its maximum value in a central position. Towards the right, regularity increases and then remains constant. The selected peaks are marked with red crosses implying a list of cluster distance threshold values.

2.4 Beat detection

In order to predict future events without breaking the metrical structure we use a tempo detection method and introduce way to align onsets to a metrical grid.

Our starting point is the regular level that has been found with the procedure explained in the previous subsection. On this level we select the appropriate symbol with the highest regularity value. The subsequence that carries this symbol will be referred to as the skeleton subsequence since it is like an anchor structure to which we relate our metrical interpretation of the sequence.

2.4.1 *Tempo Detection (Inter Beat Interval)*

Once the skeleton subsequence is found, the inter beat interval is estimated with the procedure explained in [16]. The tempo is detected considering the intervals between all onset pairs of the sequence using a score voting criterion. This method tends to give higher scores to the intervals that have more instances and that share many integer ratios with other intervals.

Then the skeleton subsequence onsets are parsed in order to detect a possible alignment of the grid to the sequence. A tolerance of 6% the duration of the inter beat interval is allowed for the alignment of an onset to the grid position. We chose the interpretation that aligns the highest number of instances to grid. After discarding the onsets that are not aligned we obtain a preliminary skeleton grid. In Fig. 7 the procedure is visually explicated.

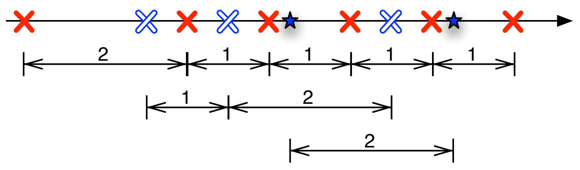

Figure 7. A skeleton sequence is represented in a timeline. *Below*, some possible alignments of the sequences are given based the measure duration provided by the Dixon method. Each phase interpretation catches some onsets (represented with its own graphical marker) and discards some others. The phase that allows to catch more onsets (the filled red crosses) is selected and the remaining onset are removed from the skeleton grid.

2.4.2 *Creation of Skeleton Grid*

The preliminary skeleton grid is a sequence of onsets spaced in multiples of a constant time interval. But, as shown in the case of Fig. 7, it can still have some gaps (due to missing onsets). The missing onsets are, thus, detected and, in a first attempt, the system tries to align the missing onsets with one of the onsets of the entire event sequence (not only from the onsets of a certain symbol). A tolerance value of 6% determines whether there is no onset to be aligned and, in this case, the system creates a grid bar in the expected beat position.

At the end of this completion procedure, we obtain a *skeleton grid* that will be considered to be a sequence of beats or, more generally, a sequence of events sharing the same metrical position (the same phase).

Because of the tolerance used for building such a grid it could be noticed that sometimes the effective measure duration could be slightly longer or slightly shorter. This fulfills the idea that the grid should be elastic in the sense that, up to a certain degree, it adapts to the timing of the actual sequence.

The skeleton grid catches a part of the complete list of onsets, but we would like to built a grid where most of the onsets are aligned. Thereafter, starting from the skeleton grid, the intermediate point between every two subsequent beats is found and aligned with an onset (if it exists in a tolerance region otherwise a place-holding onset is added). The procedure is recursively repeated until at least 80% of the onsets are aligned to a grid position or the number of created onsets exceeds the number of total onsets.

In Fig. 8, an example is presented along with the resulting grid where the skeleton grid, its aligned, and the non-aligned subdivisions are indicated by different line markers.

Note that, for the sake of simplicity, our approach assumes that the metrical structure is binary. This causes the sequence to be eventually split erroneously. However, we will see in a ternary tempo example that this is not a limiting factor for the generation because the statistical representation somehow compensates for it even if less variable generations are achieved. A more general approach could be implemented with little modifications.

The final grid is made of *blocks* of time of almost equal duration that can contain none, one, or more onset events. It is important that the sequence given to the statistical model is almost homogeneous in time so that a certain number of blocks corresponds to a defined time duration.

We used the following rules to assign a symbol to a block (cf. Fig 8):

- blocks starting on an aligned onset are denoted by the symbol of the the aligned onset,
- blocks starting on a non-aligned grid position are denoted by the symbol of the previous block.

Finally, a phase value is assigned to each block describing the number of grid positions passed after the last beat position (corresponding to the metrical position of the block). For each level of representations the new representation of the sequence will be the Cartesian product of the instrument symbol and the phase.

3. STATISTICAL MODEL LEARNING

Now we statistically analyze the structure of the symbol sequence obtained in the last section.

We employ variable length Markov chains (VLMC) for the statistical analysis of the sequences. In [7, 17], a general method for inferencing long sequences is described. For faster computation, we use a simplified implementation as described in [3]. We construct a suffix tree for each level based on the sequence of that level. Each node of the tree represents a specific context that had occurred in the past. In addition, each node carries a list of continuation indices corresponding to block indices matching the *context*.

For audio, a different approach has been applied in [18]. This method does not require an event-wise symbolic representation since it employs the factor oracle algorithm. VLMC has not been applied to audio before, because of the absence of an event-wise symbolic representation we presented above.

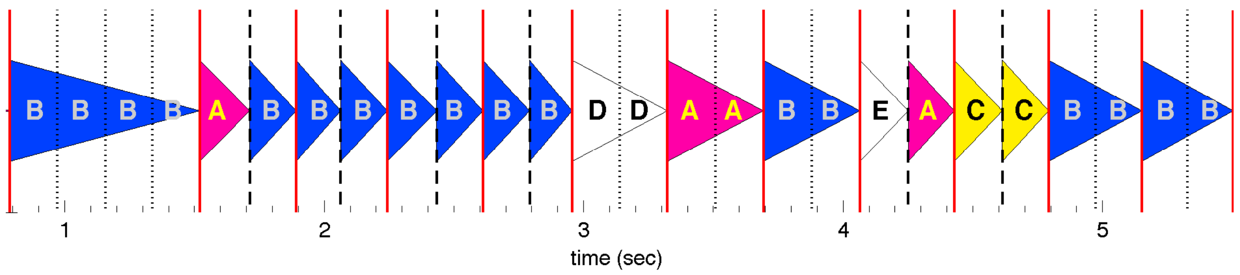

Figure 8. The event sequence derived from a segmentation by onset detection is indicated by triangles. The vertical lines show the division of the sequence into blocks of homogeneous tempo. The red solid lines represent the beat position (as obtained by the skeleton subsequence). The other black lines (either dashed if aligned to a detected onset or dotted if no close onset is found) represent the subdivisions of the measure into four blocks.

3.1 Generation Strategies

If we fix a particular level the continuation indices are drawn according to a posterior probability distribution determined by the longest context found. But which level should be chosen? Depending on the sequence, it could be better to do predictions based either on a coarse or a fine level but it is not clear which one should be preferred. First, we selected the lower level at which a context of at least $\hat{l}$ existed (for a predetermined fixed $\hat{l}$, usually $\hat{l}$ equal 3 or 4). This works quite good for many examples. But in some cases a context of that length does not exist and the system often reaches the higher level where too many symbols are provided inducing too random generations. On the other side, it occurs very often that the lower level is made of singleton clusters that have only one instance. In this case, a long context is found in the lower level but since a particular symbol sequence only occurs once in the whole original segment the system replicates the audio in the same order as the original. This behavior often leads to the exact reproduction of the original until reaching its end and then a jump at random to another block in the original sequence.

In order to increase recombination of blocks and still provide good continuation we employ some heuristics taking into account multiple levels for the prediction. We set p to be a recombination value between 0 and 1. We also need to preprocess the block sequence to prevent arriving at the end of the sequence without any musically meaningful continuation. For this purpose, before learning the sequence, we remove the last blocks until the remaining sequence ends with a context of at least length two. We make use of the following heuristics to generate the continuation in each step:

- Set a maximal context length $\hat{l}$ and compute the list of indices for each level using the appropriate suffix tree. Store the achieved length of the context for each level.

- Count the number of indices provided by each level. Select only the levels that provide less than 75% the total number of blocks.

- Among these level candidates, select only the ones that have the longest context.

- Merge all the continuation indices across the selected levels and remove the trivial continuation (the next onset).

- In case there is no level providing such a context and the current block is not the last, use the next block as a continuation.

- Otherwise, decide randomly with probability p whether to select the next block or rather to generate the actual continuation by selecting randomly between the merged indices.

4. EVALUATION OF EXAMPLES

As a descriptive evaluation, we asked a professional percussionist to judge several examples of continuations as if they were performances of a student. Moreover, we asked him to record two beat boxing excerpts trying to push the system to the limits of complexity and to assess critically the sequences that the system had generated from these recordings. The examples are available on the web site [19] along with some graphical animations explaining the analysis process.

Let us briefly explain what can be seen in these animations. In each video, we see the original sound fragment and the generation derived from it. Each video shows an animated graphical representation where each block is represented by a triangle. The horizontal axis corresponds to the time in seconds and the vertical axis to the clustering quantization resolution. In the beginning, the original sound is played and the animation shows the discovered block representation. At each moment, the currently played block is represented by an increased colored triangle and highlighted by a vertical dashed black line. The other colored triangles highlight all the blocks from the starting point of the measure to the current block. In the second sequence, only the skeleton subsequence is played. In the last sequence, the generation is shown. The colored triangles represent the current block and the current context. The size of the colored triangles decreases monotonically from the current block backwards displaying the past time window considered by the system. The colored triangles are represented only on the levels selected by the generation strategy. The colors correspond to a symbol in a one-to-one manner.

Four examples were taken from the ENST database (see [20]), one from FreeSound.org and two examples were recorded with the percussionist. From the ENST database, we have selected medium/high complexity examples that we numbered according to our collection list. They are Examples no. 15, 21, 28, and 31 corresponding to the following files of the ENST database:

```
053_phrase_afro_complex_slow_sticks
072_phrase_shuffle-blues_complex_slow_sticks
079_phrase_hard-rock_complex_medium_sticks
088_phrase_waltz_simple_medium_brushes
```

From FreeSound we have selected the popular "Amen Break" loop, because of its common use and manipulations during improvisation sets.

Starting from the latter, according to the percussionist,

> ≪As the starting material is relatively rich the continuation is very good considering the length of the original. Especially interesting is the small looping part in 0.58s. It is very similar to what I would do as a percussionist≫.

It is possible to note how the metrical structure is preserved in those examples due to the introduced tempo restrictions. Moreover, an important feature is that it creates relatively original variations given the short length of the learned examples. Example 28 is commented by the percussionist in the following way:

> ≪Very good. Meter is kept perfectly, and the "drum fills" are provided in appropriate times. A problem is that all fills are played as they appear in the song, and not extended or slightly more complex≫.

For Example 31, the percussionist expressed surprise for the realism of the generation and he referred to Example 15, saying:

> ≪The starting material is very good and rich in this case, so the continuation is rich too. Some fills are expanded and more complex which is very good, although other sequences appear exactly as they did in the original≫.

Referring to the beatboxing examples, he pointed out that the metrical structure is kept correctly but the beats do not vary too much in terms of accent. Then, he added:

> ≪Especially good is the looping of a single beat at 1.20s of the first example, although normally the looping shouldn't be repeated too much to maintain a phrase balance≫.

Finally, as an overall consideration, he pointed out an interesting application of the system:

> ≪If these continuations are used as an accompaniment, they are excellent since they, firstly, maintain a steady rhythm but at the same time evolve and, secondly, they more or less keep the time signature (i.e. strong beats usually land on the strong part of the meter, meters sound conceptually as distinct units)≫.

However, he also mentioned several missing features in comparison to a human percussionist solo.

From our point of view, it is worth mentioning that in Examples 21 and 31 even if the tempo is ternary the generation still preserves the metrical structure. In this case, three ternary beats constitute an event together, causing a bad representation of the sequence (a sort of swing subdivision). The statistical model is still able to select a good block position each time.

The behavior of the system depends on the correct behavior of all its parts. In particular, see [16] for a systematic evaluation of the tempo detection. Nevertheless, some examples show that even when the computed symbolic representation of the audio does not respect directly the underlying musical sequence (e.g. the ternary tempo) the statistic model tends to generate sequences that do not break the metric structure.

5. DISCUSSION

Our system effectively generates sequences respecting the structure and the tempo of the original sound fragment for medium to high complexity rhythmic patterns.

The descriptive evaluation of a professional percussionist confirmed that the metrical structure is correctly managed and that the statistical representation generates musically meaningful sequences. He noticed explicitly that the *drum fills* (short musical passages which help to sustain the listener's attention during a break between the phrases) were handled adequately by the system.

The critics by the percussionist were directed to the lack of dynamics, agogics and musically meaningful long term phrasing which we did not address in our approach.

Part of those feature could be achieved in the future by extending the system to the analysis of non-binary meter. To achieve musically sensible dynamics and agogics (*rallentando, accelerando, rubato...*) of the generated musical continuation for example by extrapolation [21] remains a challenge for future work.

6. ACKNOWLEDGMENTS

Many thanks to Panos Papiotis for his patience during lengthy recording sessions and for providing us with beat boxing examples, the evaluation feedback, and inspiring comments. Thanks to Ricard Marxer for his helpful support. The first author (MM) expresses his gratitude to Mirko Degli Esposti and Anna Rita Addessi for their support and for motivating this work. The second author (HP) was supported by a Juan de la Cierva scholarship of the Spanish Ministry of Science and Innovation.

7. REFERENCES

[1] S. Dubnov, G. Assayag, and R. El-Yaniv, "Universal classification applied to musical sequences," in *Pro-*

ceedings of the International Computer Music Conference, pp. 332–340, 1998.

[2] D. Cope, *Virtual Music: Computer Synthesis of Musical Style*. Cambridge, Massachusetts: MIT Press, 2004.

[3] F. Pachet, “The continuator: Musical interaction with style,” in *Proceedings of ICMC* (ICMA, ed.), pp. 211–218, ICMA, September 2002. best paper award.

[4] A. Addessi, L. Ferrari, S. Carlotti, and F. Pachet, “Young children’s musical experiences with a flow machine,” in *Proceedings of the 9th International Conference on music perception and cognition*, 2006.

[5] L. Meyer, *Style and music: Theory, history, and ideology*. University of Chicago Press, 1996.

[6] H. Purwins, M. Grachten, P. Herrera, A. Hazan, R. Marxer, and X. Serra, “Computational models of music perception and cognition II: Domain-specific music processing,” *Physics of Life Reviews*, vol. 5, pp. 169–182, 2008.

[7] P. Buhlmann and A. J. Wyner, “Variable length markov chains,” *Annals of Statistics*, vol. 27, pp. 480–513, 1999.

[8] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. The MIT Press, June 1996.

[9] C. E. Shannon, “A mathematical theory of communication,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, pp. 3–55, January 2001.

[10] M. J. Weinberger, J. J. Rissanen, and M. Feder, “A universal finite memory source.,” *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 643–652, 1995.

[11] S. Abdallah and M. Plumbley, “Information dynamics: patterns of expectation and surprise in the perception of music,” *Connect. Sci*, vol. 21, no. 2-3, pp. 89–117, 2009.

[12] A. Hazan, R. Marxer, P. Brossier, H. Purwins, P. Herrera, and X. Serra, “What/when causal expectation modelling applied to audio signals,” *Connection Science*, vol. 21, pp. 119 – 143, 2009.

[13] R. Marxer and H. Purwins, “Unsupervised incremental learning and prediction of audio signals,” in *Proceedings of 20th International Symposium on Music Acoustics*, 2010.

[14] O. Lartillot, P. Toiviainen, and T. Eerola, “A matlab toolbox for music information retrieval,” in *Annual Conference of the German Classification Society*, 2007.

[15] R. Duda, P. Hart, and D. Stork, *Pattern classification*. Citeseer, 2001.

[16] S. Dixon, “Automatic extraction of tempo and beat from expressive performances,” *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[17] D. Ron, Y. Singer, and N. Tishby, “The power of amnesia: learning probabilistic automata with variable memory length,” *Mach. Learn.*, vol. 25, no. 2-3, pp. 117–149, 1996.

[18] S. Dubnov, G. Assayag, and A. Cont, “Audio oracle: A new algorithm for fast learning of audio structures,” in *Proceedings of International Computer Music Conference (ICMC)*, pp. 224–228, 2007.

[19] “www.youtube.com/user/audiocontinuation.”

[20] O. Gillet and G. Richard, “Enst-drums: an extensive audio-visual database for drum signals processing,” in *ISMIR*, pp. 156–159, 2006.

[21] H. Purwins, P. Holonowicz, and P. Herrera, “Polynomial extrapolation for prediction of surprise based on loudness - a preliminary study,” in *Sound and Music Computing Conference*, (Porto), 2009.