Real-time event sequencing without a visual interface

Tiago Fernandes Tavares
University of Campinas
School of Electrical and
Computer Engineering
tavares@dca.fee.unicamp.br

Adriano Monteiro
University of Campinas
Interdisciplinary Nucleus for
Sound Communication
monteiro_adc@gmail.com

Jayme G. A. Barbedo
EMBRAPA Agricultural Informatics
jbarbedo@gmail.com

Romis Attux
University of Campinas
School of Electrical and
Computer Engineering
attux@dca.fee.unicamp.br

Jônatas Manzolli
University of Campinas
Interdisciplinary Nucleus for
Sound Communication
manzolli@nics.unicamp.br

ABSTRACT

In electronic music, it is often useful to build loops from discrete events, such as playing notes or triggering digital effects. This process generally requires using a visual interface, as well as pre-defining tempo and time quantization. We present a novel digital musical instrument capable of looping events without using visual interfaces or explicit knowledge about tempo or time quantization. The instrument is built based on a prediction algorithm that detects repetitive patterns over time, allowing the construction of rhythmic layers in real-time performances. It has been used in musical performances, where it showed to be adequate in contexts that allow improvisation.

1. INTRODUCTION

Drum machines are electronic instruments frequently used to create rhythm sections in musical pieces using loops of drum notes. A generalization of this concept involves looping not simply drum notes, but generic discrete-event-related messages, allowing the periodic execution of actions such as switching timbre or triggering audio effects. To build a loop sequence, it is common to use a grid interface, like the one shown in Figure 1, which is also used to define the desired tempo and the time quantization in the excerpt.

In order to properly interact with other musicians, a drum-sequencer player must pre-define (or, at least, detect) the musical tempo and an adequate time quantization. This aspect is also present in novel interfaces that do not use the grid display but use the same paradigm, such as the Rhythmicator [2] or the Sinkapater [3]. Those interfaces are used to plan beats and

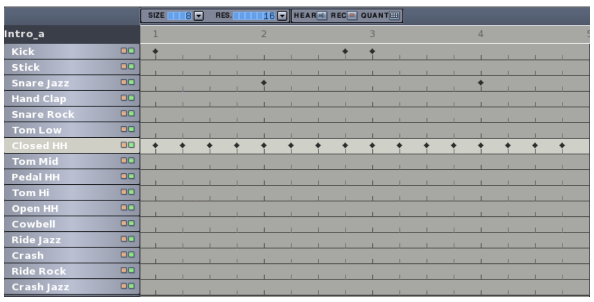

Figure 1. Drum sequencing interface in Hydrogen [1] software.

beat sequences, which is significantly different from playing drums or even tapping rhythms.

Another technique that can be used to create meaningful repetitions is to apply carefully arranged delay lines and feedback so that an audio sample is played in a pattern defined by the musician, as in RhythmDelay [4] or SDelay [5]. Using this technique, a musician can build audio loops on-the-fly, without the need of a grid interface. However, delay lines lack the symbolic-level flexibility of drum machines, as it is hard to change events (for example, changing all kick drums for cymbals) or to modify the musical tempo without affecting the timbres.

In this paper, we present a novel digital musical instrument that is capable of looping general event sequences that are played by the musician, without predefining or quantizing tempo. The instrument may be played using any interface that generates discrete events, from high-end MIDI drum interfaces to low-cost game controllers. It is implemented as a patch for PureData [6] and can be freely downloaded.

The event looping process is based on an online-learning algorithm that is used as follows. As the musician plays a sequence of events, a continuation for that is predicted by a string matching algorithm. When the user triggers the automation, the system starts yielding the predicted continuations and feeding them back into itself, creating an event sequence that corresponds to continuing the pattern played previously by the musician, without any explicit inputs regarding tempo or time-quantization.

The proposed method relies on a discrete-symbol representation for audio, which means that it possesses the flexibility of drum machines. To use it, however, the musician must employ skills that are closer to playing drums or tapping rhythms than to using visual interfaces. Also, the implementation in a free, open-source environment allows it to be used in several creative ways.

This paper is organized as follows. In Section 2, previous approaches on the automatic detection of patterns in music are presented. The event forecasting algorithm is described in Section 3, and implementation issues are discussed in Section 4. Experiments showing advantages and limitations of the proposed instrument are discussed in Section 5. Section 6 brings further discussions on the results. Last, Section 7 brings some conclusive remarks.

2. PREVIOUS WORK

The metaphor of a grid-looping environment, such as the usual drum machine [1], has been employed in many innovative systems. Two remarkable examples are the Rhythmicator [2], which uses a probabilistic model to create drum tracks directly from audio analysis, and the Sinkapater [3], which allows each drum track to use a different measure length, creating polyrhythms that are usually hard to be played by a human being [7]. In both cases, there is an inherent need to predefine tempo and beat, which, as discussed above, is a skill that is foreign to the playing of drums itself.

To avoid using these concepts – tempo and beat – it is necessary to automate the process of building the event loops. As shown in early studies by Shannon [8] and Solomonoff [9, 10], sequences of symbols can be predicted, as long as they present a certain degree of repetition. This assumption is fit for event loops, as the same pattern is, in general, repeated many times over a musical piece.

Assayag *et al.* [11] developed a system capable of forecasting continuations for melodies. In their work, a codebook built from a data corpus is used to predict plausible continuations for a given piece. The outcomes of this system were evaluated as "repetitive", which may be expected from the deterministic nature of the prediction system. Also, building a codebook is a slow process that cannot be done in real time, and the system only supports quantized tempo.

To avoid the repetition problem, there has been an effort towards developing systems capable of learning rhythms from a corpus and predicting new patterns. Techniques such as rule-based probabilistic recombination [12], artificial life [13] and genetic algorithms [14] were used in this context. This gives rise to another form of human-computer interaction, in which the machine yields unpredictable outcomes.

A characteristic that is common to all musical prediction methods cited above is that they rely on an exact timing precision. This can be achieved if strict timing quantization is used. However, the quantization implies in a pre-definition of tempo and beat.

We propose a system that receives as input a sequence of events and yields a possible continuation for these events. The yielded events may be re-inserted into the system using feedback, thus creating a loop of events that do not rely on explicit definitions of tempo or beat. The continuations are quickly learned, so the system may be used in real-time performances, and its behaviour is highly predictable, which gives the user great control of its outcomes.

A thorough explanation about the algorithm is presented in the next section.

3. PROPOSED SYSTEM

The proposed system assumes that events repeat within a particular piece or excerpt. This assumption allows simple, intuitive interactions with the system, as the process of designing a new loop becomes similar to that of showing a rhythmic pattern to a human being. Hence, the musician can have great control on the outcomes of the system, as if the usual grid interface was being used.

In the context of this work, each musical event n is represented by its *onset* s_n, that is, the time it happens, and its *label* l_n, which identifies the event. Using labels, events may be related to any description of discrete musical gestures desired by the musician, such as "play cymbal", "strongly play cymbal", "play random drum" or "activate reverb effect". As will be shown, the flexibility of the event label allows many creative uses of the system.

The system receives event-related messages through any device that yields discrete messages (such as MIDI instruments, OSC controllers or HID devices). The label and onset of the events are stored in an internal buffer of arbitrary size. The information in the internal buffer is used to predict the following event, as described below.

The forecasting algorithm is based on the assumption that the musician is playing a loop, a condition that has to be intentionally caused. When the user desires, the predictions may be used as inputs to the forecast system, creating a feedback loop and allowing the continuation of the event sequence. Preliminary tests showed that adding a reset functionality, which clears the internal buffer, made it easier to switch between different beats.

In the context of the prediction algorithm, two events n and m are considered equivalent if their label is the same ($l_n = l_m$) and their inter-onset intervals (IOIs), that is, the difference between their onset and the previous onsets, are within an allowed deviation ($\|s_n - s_{n-1} - (s_m - s_{m-1})\| \leq \alpha$). The algorithm, shown in Figure 2, aims at searching, within the last N recorded events, the longest subsequence $[M - K +$

1 ... M] whose events are equivalent to those in the subsequence $[N - K + 1 \dots N]$. After the subsequence $[M - K + 1 \dots M]$ is found, it is reasonable to assume that the continuation of the recorded excerpt (that is, event $N + 1$) will be equivalent to that of the subsequence (event $M + 1$).

```
1:  procedure FORECAST($N, \boldsymbol{s}, \boldsymbol{l}, \alpha$)
2:      $\boldsymbol{d} \leftarrow$ array of $N$ zeroes
3:      for $M = N - 1$ to $2$ do                ▷ Search for repetitions
4:          $k \leftarrow 0$
5:          while $M - k > 0$ and $l_{N-k} = l_{M-k}$ and $|(s_{N-k} - s_{N-k-1}) - (s_{M-k} - s_{M-k-1})| < \alpha$ do
6:              $k \leftarrow k + 1$
7:          $\boldsymbol{d}_M \leftarrow k$
                                   ▷ Check if a subsequence was found
8:      if MAX(D) $> 0$ then                     ▷ If found
9:          $\hat{M} \leftarrow \arg\max D$
10:     else
11:         $\hat{M} \leftarrow N$
                                                 ▷ Yield events
12:     $s_{N+1} = s_{\hat{M}+1} - s_m + s_N$
13:     $l_{N+1} = l_{\hat{M}+1}$
14:     return
```

Figure 2. Pseudo-code for event forecasting.

Figure 3 shows an example of a possible execution of the forecasting algorithm. The left column shows the internal buffer after the user has yielded a series of events, arbitrarily labeled "hit drum" and "switch reverb". After triggering event E9, the forecasting system detects that, if the buffer is delayed by three events, events E9 and E6, as well as the previous five events, are equivalent, which is a higher number of equivalent events than if any other delay was used.

In this example, event E7 is chosen as the most plausible continuation for the delayed sequence, generating the estimated E10 and is being used to build the yielded event. If event E10 is used by the system as an input, the next event to be yielded would be a "hit drum", then a "switch reverb", then a "hit drum" again, creating a cycle of repetitions. Hence, feedback may be used to create event loops in real time.

Next section discusses implementation and usability issues.

4. IMPLEMENTATION ISSUES

The proposed instrument was implemented as a patch for PureData [6]. This allows users to employ their favorite interfaces and sound designs, as well as their own compositional ideas. It is an open-source project, which means that the algorithm may be easily ported to other contexts.

The patch works in two modes: **observation**, or manual, and **prediction**, or automatic. In the observation mode, the system receives inputs from the user and tries to predict the next event that will be received. When the prediction mode is triggered, the system receives its own predictions, instead of user-generated events, as inputs.

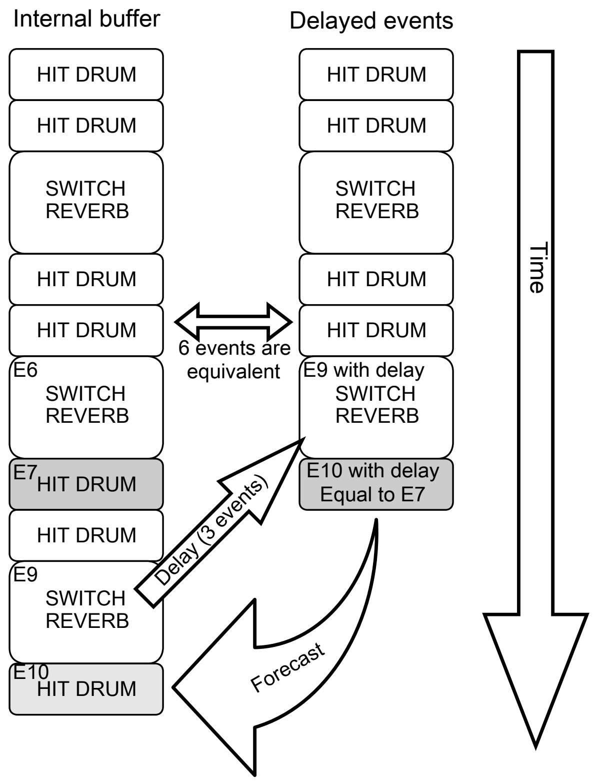

Figure 3. Example of a possible execution of the forecasting algorithm.

Additional functionality may be easily implemented by the user, but not as part of the system itself. Several predictor instances in parallel, for example, may be used to achieve polyrhythms. Also, the actions related to an event must be defined by the user: playing a drum sample or switching the configurations of an effect, for example.

5. EXPERIMENTAL PERFORMANCES

The experiments in this section aim at highlighting interesting features and also drawbacks of the proposed system, gathering information on how it can be helpful to musicians and how it may be improved. For this purpose, the system was used in a solo and an accompanied performance. The recorded audio material can be listened to at the URL `http://www.dca.fee.unicamp.br/~tavares/Looper/index.html`.

The first performance – solo – aimed at showing the main capabilities of the instrument. The piece was intentionally composed so that a drum sequence and an effect switch were independently looped, allowing another layer of percussion to be freely played. Spectrograms are used to show the most important parts.

The second performance – duo – had the goal of showing that the drum loops could be quickly arranged and played in an arbitrary tempo. The piece was an improvisation in which digital effects were manually played while drums were looped using the proposed system. The piece was analyzed based on the impression of both musicians and on auditory characteristics of the recording.

5.1 Solo performance

The first experiment was based on composing and playing a short piece using the proposed instrument. The composer and performer was one of the authors of this paper. A low-cost gamepad controller was used to tap rhythms and to control the behaviour of the prediction algorithm.

The piece is based on two drum synthesizers that yield samples to a clip-based distortion that, in turn, yields a change in timbre. The drum onsets and the use of the distortion are controlled by the musician and can be individually looped using separated instances of the proposed system. By looping drum patterns, a base rhythm for the piece can be generated, whereas looping switches in the distortion creates different electronic ambiences that give more variations to the piece.

Figure 4 shows the instant the looping process is triggered by the musician as a vertical black line – the previous events are manually controlled. After triggering the prediction mode, the musician stopped playing. As it can be seen, the algorithm successfully continued the manually-played pattern.

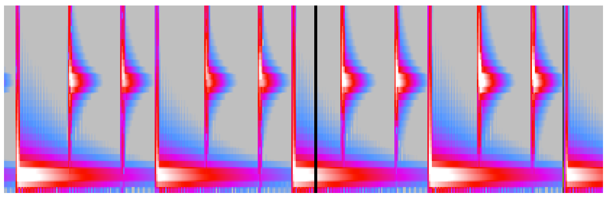

Figure 4. Spectrogram showing the prediction of a drum loop.

Figure 5 depicts the results of looping the distortion switch, which is triggered in the moment displayed as a vertical black line. As it can be seen, when the distortion is used more harmonics are present. This adds to the drum loops, creating a more complex rhythm.

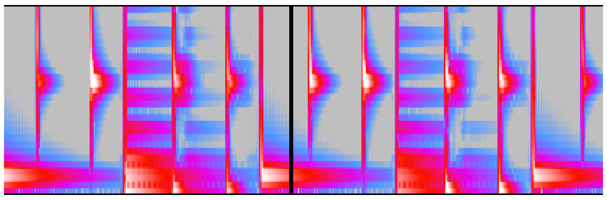

Figure 5. Spectrogram showing the prediction of distortion triggers.

Last, Figure 6 shows a particular excerpt in which both the drum and the effect loops are active. The musician manually plays a new layer of drum events. These events are not looped, but contribute to create a rhythm that would be hard to achieve by manual playing without the overlap of automated loop patterns.

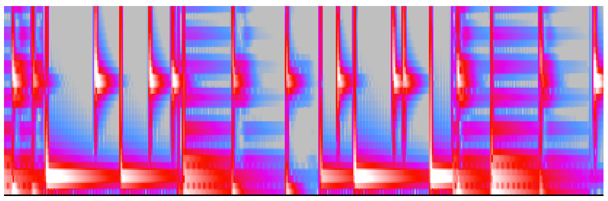

Figure 6. Spectrogram showing the use of an additional percussion layer on top of the event and the percussion loops.

In the audio recording, it could be noted that the developed rhythms tend to sound natural, despite of the lack of dynamics caused by the use of a low-cost controller. The transitions between musical sections were smooth, which is a consequence of having to play each new loop pattern that would be used – this prevented sudden transitions between complex patterns. These characteristics were also observed in the accompanied performance, as shown below.

5.2 Accompanied performance

In the second experiment, the proposed system was used in a guitar-electronics duo. The electronic percussion accompaniment, based on two tabla samples, was played over an improvised guitar. The guitar and the electronic parts were played, respectively, by an invited musician and by one of the authors of this paper, who used a low-cost gamepad controller to tap rhythms and control the behaviour of the prediction algorithm.

The guitar player stated that, while playing, there was no need to explicitly think about tempo or musical sections, because they emerged naturally from playing. According to the statement, this flexibility allowed the musician to enhance the focus on other musical aspects, such as dynamics and phrasing. Ultimately, this lead to a feel of flow, which is frequently not the case when playing with drum sequencers, as they are bounded to pre-defined tempo, swing and measures.

The electronics player observed that, as it is easy to switch between the automatic and the manual modes of operation, the percussion parts could be quickly rearranged. The interaction with the guitar player felt more natural than if a grid interface was used, because using the system with a game controller is more similar to tapping rhythms. Also, while the automatic operation was used, other actions could be performed – manually playing another percussion layer or triggering other digital effects.

In the audio recording, it was noticeable that the feeling of steadiness, often present when using drum machines, was only present in few passages. Hence the proposed instrument allows using steady beats, but not as a requirement of the instrument. In spite of the tempo flexibility, there was no variation in the loudness of the drum beats, as the physical interface (a game controller) did not capture dynamics.

In the audio recording, the feeling of steadiness, caused by the quantized tempo often present when using grid interfaces, was significantly reduced. Hence, the proposed instrument allows for a steady rhythm, but not as an imposition of the instrument (as it would be the case for a grid interface). However, due to the nature of the game controller used, no information on dynamics was gathered, so the percussion is clearly played with the same loudness at all times.

6. DISCUSSION

This section discusses characteristics of the proposed instrument, highlighting the main differences regarding the previous approaches.

A feature that must be considered is that the instrument does not pre-define a physical interface for usage nor a sound design. These parts of the interaction must be composed by the user, which allows for great flexibility, but also require the user to employ a certain level of expertise. A possible way to solve this is by developing pre-set configurations for common use cases, but it must be considered that developing own interfaces and sound designs is frequently an important part of modern musical composition processes.

The instrument allows a quick development of rhythm loops and does not require a visual interface or any pre-definition of tempo or beat. Also, the same interface that is used to develop the loops can be used to build a layer of freely improvised drums. This means that a the instrument is potentially more useful in contexts that allow for flexibility and improvisation.

Since there is no visual interface or explicit notations, there is no way to store rhythms for later use. This also indicates that the instrument is suitable for flexible and improvised scenarios. On the other hand, its applications in offline composition are limited, as there is no visual feedback regarding the stored sequences.

The instrument delivers great control of the sequence that will be played, because it simply continues patterns played by the musician. This behaviour is close, but not exactly the same as, the one yielded by the method proposed by Assayag *et al.* [11], as it is deterministic and learned from data yielded by a human being in real-time. Hence, the system does not aim at generating new sequences as the previous approaches discussed above [12–14].

However, it is possible to change the outcomes algorithm by increasing the onset tolerance α to values that are close to a typical inter-onset interval. In this case, the algorithm can yield unexpected continuations for a sequence. Although this is not the original purpose of the system, interesting sequences can emerge from this phenomenon.

Overall, the proposed system has presented great musical potential in improvised performances. It reinforces the paradigm of using multiple layers to build a piece. However, its applications to offline musical composition are limited, as there is no interface allowing the user to review the current content of the buffer.

Next section presents conclusive remarks.

7. CONCLUSION

We presented a novel digital instrument aimed at sequencing events without the need for visual interface or pre-definitions of tempo, measure or time quantization. The core of the instrument is a string matching algorithm that quickly learns patterns played by a human musician and continues them in real-time. The instrument was used in two musical performances, one solo and one duo, and was evaluated both by its user and by the accompanying musician.

The proposed instrument is shown to be adequate to improvised contexts, in which its quick response may be employed at its best. It allows the rapid generation of multi-layer rhythmic figures, as well as the automation of effect triggers. Also, allows using abilities that are closer to playing acoustic drums, such as tapping rhythms, which greatly favours its use in accompanied improvisation.

An aspect of the instrument that remains unexplored is the possibility of transforming its data, thus generating new sequences from user inputs. Doing this without using explicit time quantization and, at the same time, preserving the user's intention, is a topic that should be studied in future work.

8. REFERENCES

[1] Hydrogen. Hydrogen advanced drum machine for gnu/linux. [Online]. Available: http://www.hydrogen-music.org/

[2] G. Sioros and C. Guedes, "Automatic rhythmic performance in Max/MSP: the kin. rhythmicator," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Oslo, Norway, 2011, pp. 88–91. [Online]. Available: http://www.nime2011.org/proceedings/papers/B16-Sioros.pdf

[3] J. Harriman, "Sinkapater - an untethered beat sequencer," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, G. Essl, B. Gillespie, M. Gurevich, and S. O'Modhrain, Eds. Ann Arbor, Michigan: University of Michigan, May 21-23 2012.

[4] P. Community. Rhythmdelay. [Online]. Available: http://puredata.hurleur.com/sujet-8421-electronics-two-rhythm-delay-steroids/

[5] AudioMulch. Sdelay. [Online]. Available: http://www.audiomulch.com/help/contraption-ref/SDelay

[6] P. community. Puredata. [Online]. Available: http://puredata.info/

[7] J. J. Summers and T. M. Kennedy, "Strategies in the production of a 5 : 3 polyrhythm," *Human Movement Science*, vol. 11, no. 12, pp. 101 – 112, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016794579290053E

[8] C. E. Shannon, "Prediction and entropy of printed english," *The Bell System Technical Journal*, vol. 30, pp. 50–64, 1951.

[9] R. Solomonoff, "A formal theory of inductive inference, part i," *Information and Control*, vol. 7, no. 1, pp. 1–22, Mar 1964.

[10] ——, "A formal theory of inductive inference, part ii," *Information and Control*, vol. 7, no. 2, pp. 224–254, Jun 1964.

[11] G. Assayag, S. Dubnov, and O. Delerue, "Guessing the composer's mind: Applying universal prediction to musical style," in *Proceedings ICMC 99*, Beijing, China, 1999.

[12] M. Dahia, H. S. E. Trajano, G. Ramalho, C. Sandroni, and G. Cabral, "Using Patterns to Generate Rhythmic Accompaniment for Guitar," in *Proceedings of the SMC 2004*, Paris, France, 2004.

[13] J. M. Martins and E. R. Miranda, "Breeding Rhythms with Artificial Life," in *Proceedings of the SMC 2008*, Berlin, Germany, 2008.

[14] G. Bernardes, C. Guedes, and B. Pennycook, "Style Emulation of Drum Patterns by Means of Evolutionary Methods and Statistical Analysis," in *Proceedings of the SMC 2010*, Oslo, Norway, 2010. [Online]. Available: http://smcnetwork.org/files/proceedings/2010/26.pdf