Real-time Manipulation of Syncopation in Audio Loops

Diogo Cocharro
INESC TEC
Porto, Portugal
dcocharro@gmail.com

George Sioros
Faculdade de Engenharia
da Universidade do Porto
Porto, Portugal
gsioros@gmail.com

Marcelo Caetano
INESC TEC
Porto, Portugal
mcaetano@inescporto.pt

Matthew E.P. Davies
INESC TEC
Porto, Portugal
mdavies@inescporto.pt

ABSTRACT

In this work we present a system that estimates and manipulates rhythmic structures from audio loops in real-time to perform syncopation transformations. The core of our system is a technique for the manipulation of syncopation in symbolic representations of rhythm. In order to apply this technique to audio signals we must first segment the audio loop into musical events using onset detection. Then, we use the symbolic syncopation transformation method to determine how to modify the rhythmic structure in order to change the syncopation. Finally we present two alternative methods to reconstruct the audio loop, one based on time scaling and the other on resampling. Our system, *Loopalooza*, is implemented as a freely available *MaxForLive* device to allow musicians and DJs to manipulate syncopation in audio loops in real-time.

1. INTRODUCTION

Rhythm transformations encompass a wide range of techniques to modify rhythmic patterns. Musically, such rhythm transformations hold great creative potential for live performance and automatic remixing. However, the manual manipulation of rhythmic patterns in audio signals can be a long and cumbersome process and often requires a high level of technical skill in specific audio editing software. Automating rhythmic transformations can therefore allow users to focus more on the creative rather than technical aspects of the process and encourage greater experimentation.

There is a number of systems for automatic transformation of rhythm in audio signals. These systems have been designed towards a specific goal, e.g. changing the swing [1, 2], the tempo or the meter [2], or matching the rhythm of the audio input to a model [3–5]. Depending on the transformation algorithm, the order of the segments is preserved [1, 2] or not [3–5]. To the best of our knowledge, none of these audio-based rhythm transformation systems can allow users to explicitly manipulate the syncopation present in audio.

Within the symbolic music domain (i.e. using MIDI signals) Sioros et al. [6–8] proposed several algorithms to generate rhythmic patterns in real-time which allowed users to control specific qualities of the generated rhythm. In particular, the generative algorithm described in [6, 7] allows for the manipulation of the amount of syncopation, the density of events, and the metrical strength of the generated patterns, while the system described in [8] allows users to control the complexity of the drum pattern that are produced by recombining short MIDI loops.

Recent studies about syncopation [9–11] have shown it stands out as one of the main musical properties that contributes to the perception of groove, rhythm complexity and music tension. In this work, we present a system that allows users to manipulate syncopation in audio loops in real time. Our system assists DJs and music producers to explore the creative potential of syncopation. This work is based on a previous formalized algorithm, and corresponding application *Syncopalooza* [12]. *Syncopalooza* manipulates the syncopation in symbolic data, i.e. MIDI. Our system extends the previous application by applying the same syncopation algorithm to manipulate audio loops.

Our approach to modifying the syncopation in audio loops is built around three steps: analysis, transformation and reconstruction. In the analysis stage we use onset detection to identify the start times of musical events in the audio loop and hence to extract the rhythmic structure in the context of a known time signature and tempo. Next, the transformation step applies the symbolic approach to syncopation manipulation of Sioros [12] to determine how the rhythmic structure of the audio loop can be modified. Finally, reconstruction consists of implementing this transformation in the audio domain. For this reconstruction step we propose two approaches, one based on time scaling and the other using sampling techniques. We compare the properties of these two reconstruction techniques in terms of their musical and audio quality-preserving properties.

In order for musicians and DJs (as well as researchers) to interact with our audio syncopation manipulation system, we have developed a *MaxForLive* device called *Loopalooza* which is freely available online in [13]. *Loopalooza* has a simple interface, which offers users the ability to control the syncopation in the audio loop in real-time via a slider. To demonstrate the kinds of musical effects which can be created with our system we provide online demo videos in [13].

The remainder of this paper is structured as follows. In Section 2 we provide an overview of the symbolic syncopation manipulation method. In Section 3 we describe our proposed system for manipulating syncopation in audio

signals. Then, we present details of the usage of the *Loopalooza* device in Section 4. Finally we present discussion and conclusions in Sections 5 and 6.

2. SYNCOPATION TRANSFORMATIONS

Recently, Sioros et al. proposed an application for the manipulation of syncopation in music performances, named *Syncopalooza*. The core of *Syncopalooza* uses an algorithm that can remove or generate syncopation in rhythmic patterns by displacing the start times of musical events (onsets) with respect to a metrical template [12]. After the metrical template is created according to the meter and tempo, the onsets are aligned to the grid by snapping their time positions to the closest pulse boundary present in the metrical grid. Next, the continuous stream of onsets is converted into a binary pattern. Finally the transformation is performed.

The metrical template used to compute syncopation transformations describes the meter in which the rhythmic patterns are produced. It is constructed automatically for the user defined meter and tempo. As seen in the *Transformation* section of **Figure 1**, the metrical template consists of pulses that represent the alternating strong and weak beats commonly found in a 4/4 meter. Each pulse initiates a metrical level within the meter according to its position, e.g. pulse 0 in **Figure 1** initiates a quarter note, while pulse 2 initiates an eight note. The slower the metrical level is the stronger the pulse is considered. The *fastest metrical level* corresponds to the fastest subdivision where syncopation can be perceived, therefore it also represents the quantization grid, i.e. the shortest possible inter onset interval for events within the meter.

The transformation is performed by displacing the onsets so that their order is always preserved. Onsets are shifted forward to metrically stronger positions to *remove* the existing syncopation, or backward to earlier weaker metrical positions to *generate* syncopation. The transformations are applied gradually, shifting a single onset at each step. This process can generate many binary patterns with various degrees of syncopation ranging from no syncopation to maximum syncopation (a pattern that cannot be further syncopated).

To apply these types of transformations to audio signals in real-time as we propose in this paper, we require two additional elements: a means to identify the onsets of musical events, and a technique for reconstructing the audio signal after the syncopation transformation (while best preserving the original characteristics of the audio loop). Our proposed approaches for these components are described in the following section.

3. THE LOOPALOOZA SYSTEM

Most systems that automatically manipulate rhythmic features in audio [1–5] commonly comprise three stages. First, the audio input is segmented into events in an analysis stage. Then the temporal relationship among the events is manipulated during the transformation stage and finally the audio result is produced in the reconstruction stage.

Figure 1 presents an example that illustrates how our method performs these three stages. The top panel shows the analysis stage, the middle the transformation, and the bottom the reconstruction stage. The metrical template shown in the *Transformation* block imposes a quantization grid to both the input rhythmic pattern and the reconstructed one. After the analysis, it quantizes the onsets to the grid positions. During re-construction, the events will be triggered using the grid as a reference. Reconstruction of the audio loop is performed either by means of time scaling or resampling. Next we describe each stage of our system.

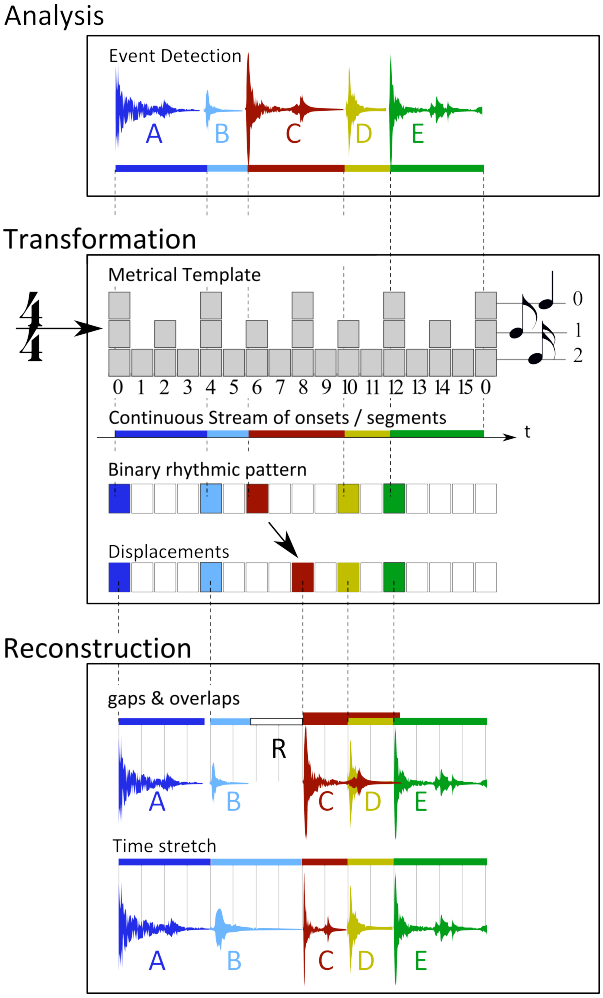

Figure 1. Example of rhythm transformation removing syncopation from the audio loop. **Analysis:** the audio input is analyzed for onsets and sliced into short segments (shown in different colors: A, B, C, D and E) forming a rhythmic pattern. Each segment represents a musical event. **Transformation:** The onsets positions are quantized into a binary pattern according to the metrical template. The onset of event C is desyncopated by displacing it from pulse 6 to pulse 8. **Reconstruction:** The new inter-onset interval durations can result in silent gaps (R between B—C) or overlapping segments (C—D). In this example, the selected transformation is reconstructed using the time scaling mode.

3.1 Analysis

In the analysis stage, the audio input (assumed in this text to be sampled at 44.1 kHz) is segmented using onset detection, thus creating a rhythmic pattern. The pattern consists of the events formed by the short segments between the onsets as shown at the *Analysis* block of **Figure 1**.

The generic analysis stage for rhythm transformation might also include beat tracking and the bar length estimation, depending on the end application. However, in our proposed system, the tempo and meter are user-defined parameters. Thus, the goal of the analysis stage is to determine the onset locations of the audio loop in real-time.

The system keeps track of the onset position in the audio buffer and the position within the musical bar. The onset detector is set with an analysis vector size of 512 samples (11.6 ms at 44100 samples per second), which defines the maximum delay required to detect an onset. To compensate for the onset detection delay, we subtract the duration of 512 samples from the stored positions.

While any real-time onset detection algorithm could be used, we use *fzero~*, a native *MaxMSP external* [14] that reports onsets by detecting if the peak amplitude or the pitch changes more than a specified amount. Thus *fzero~* is suitable to detect onsets corresponding to melodic or percussive events. This external has the advantage of being compatible across operating systems and of being continuously supported by *Cycling'74*.

3.2 Transformations

The transformation consists of displacing the time positions of the detected onsets, manipulating the rhythmic features of the pattern. The transformation algorithm is directly taken from [12]. As the main contribution of this paper is in the analysis and reconstruction stages, the transformation algorithm is described in section 2. Finally, the audio output is reconstructed with the segments in their new positions. The reconstruction process is described in the following section.

3.3 Reconstruction

The displacement of the onsets during the transformation stage changes the original inter-onset intervals resulting in potential silent gaps or overlapping segments. The reconstruction of the events in the audio output aims at generating a smooth transition between the audio segments. **Figure 1** shows an example with a silent gap R between segments B and C. Such silent segments can have a disruptive effect in the audio output.

Gouyon et. al. [1] state that it is possible to fill these gaps by generating a tail using reverberation. The applied reverberation increases the length of the decay phase of the audio segment, thus filling up the gap. However, reverberation is usually a spatial characteristic and should not change abruptly between segments. Applying reverberation only to one segment and not to the entire audio output can thus produce an unrealistic effect.

In our system, a sequencer plays back the transformed pattern by triggering the events found in the corresponding pulses. The reconstruction is performed during playback while the event is reproduced. In light of the difficulty to fill the silent gaps produced, our system provides two audio reconstruction methods, a time scaling algorithm and a resampling technique.

3.3.1 Time Scaling

The time scaling method automatically adjusts the duration of the events to the transformed inter onset intervals. Time scaling can result in a reduction in audio quality when applied to percussive sounds [2], where the sharpness of percussive onsets might be blurred. One way of avoiding artifacts due to time scaling is to use an algorithm that preserves audio transients and only operates on the stable part of the sound.

More importantly, time scaling can also distort the rhythmic pattern. A segment might contain rhythmic elements besides the initial onset. For example, quieter onsets that go undetected during the segmentation of the input or other temporal structures such as tremolo and vibrato (see **Figure 1**, segments A, C or E). Time scaling changes the tempo of the internal rhythmic structure.

The artifacts introduced by time scaling depend on the scale factor, which, in turn, depends on the displacement of the onsets. The syncopation transformations typically result in inter-onset intervals that range between half and twice the length of the original segment.

Whenever the new inter onset interval is greater than the original event duration, the event duration is stretched to fit the new interval (see segment B in **Figure 1**). When the new inter onset interval is less than the original event duration, two options are available, time compression and cropping. Time compression matches the event duration to the new inter onset interval, while cropping reproduces the event at the original speed discarding the overlapping section. A user-defined parameter determines which option is used.

3.3.2 Real-Time Resampling

Resampling fills the gaps between events by duplicating portions of the original audio (see **Figure 2** for an example). After an event is triggered, it is played back at its original speed. If the playback of the event finishes in between two pulses, the playback direction is reversed and playback continues until the next pulse. A fade out envelope is applied on the reversed portion.

If a new event is not triggered by the sequencer at the next pulse, an audio snippet with the duration of a single pulse is used to continue playback until the following pulse. The process repeats until the sequencer triggers a new event. The same snippet is used in each repetition with the amplitude attenuated by half each time.

The audio snippet is extracted from either the current event or the following one in the rhythmic pattern. This way, the two events are bridged by either creating a tail that "echoes" the last event or by anticipating the following one. In any case, the newly introduced audio material

does not affect the order of the musical events found in the original pattern.

There are three options available for the extraction of the audio snippets as a user defined parameter, i) the beginning of the last triggered event, ii) the end of the last triggered event, or iii) the beginning of the following event. Snippets from the beginning of events (options i and iii) always contain onsets. Snippets taken from the end of events (option ii) contain the decay section of the event. Since the snippets are always placed on the metrical grid, they become rhythmic elements naturally merged into the rhythmic pattern. Fade in and fade out curves are used to crossfade smoothly across snippets.

We illustrate the above process with the example of event B found in **Figure 1** and **Figure 2**. In particular, **Figure 2** shows the details of the resampling used to bridge the gap R between events B and C. Event B is triggered on pulse 4. The end of the event is reached just before pulse 6. At this point, the playback direction is reversed, so that the portion of the event noted as B1 fills the gap until pulse 6. The following event to be triggered by the sequencer is event C on pulse 8. Therefore, a gap of two 16th notes remains. The last 16th note of event B (noted on the figure as B2) is used to fill it. It is repeated twice, in pulses 6 and 7, each time with attenuated amplitude. Finally, on pulse 8, a new event (C) is triggered.

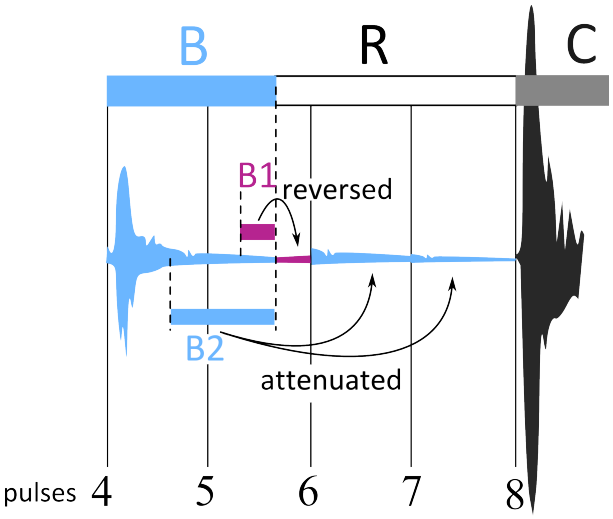

Figure 2. Reconstruction by resampling event B and the gap R of the example of Figure 1. The solid vertical lines represent the quantization grid. B1 illustrates padding the event duration to the quantization grid, which consists in reverse playback of the final part of segment B. B2 represents the audio snippet used to fill the remaining gap until the next event onset (C).

4. THE MAX FOR LIVE DEVICE

In this section, we provide an overview of the main functions and the interface of the *Loopalooza MaxForLive* audio device that implements the algorithms described in Section 3. In **Figure 3**, the interface is shown. *Loopalooza* shares some of the building blocks found in the *Syncopalooza* application described in [12]. Both applications have a similar interface and parameters, but differ in terms of their input. *Syncopalooza* only processes MIDI events in its input. On the other hand, *Loopalooza* processes a real-time audio stream.

Loopalooza is loaded in an audio track in the *Ableton Live* sequencer as an audio *MaxForLive* device. The audio stream of the track (audio input or clips playing) is then fed automatically to the audio input of the device. The continuous audio stream is constantly stored in a cyclic buffer in the device.

However, the system only processes audio as single bar loops. For this reason, each bar is processed and reproduced separately after it is recorded in the buffer. This introduces a one bar delay between the input and the transformed output. Such a delay is commonly found in audio loopers. Since the transformations are computed in real-time, there is no additional delay. Changes to parameters via the interface take effect immediately. The user can interrupt the recording of the input stream by pressing the "toggle recording" button on the right side of the device. In that case, the output continues using the last recorded bar as a loop.

The time signature and tempo is automatically retrieved from *Ableton Live*. This information is then used to build the metrical template and define the quantization grid. The global transport of *Ableton Live* controls the playback and determines the metrical position within the bar. The metrical position is used in the transformation of the audio input and during the reconstruction and playback of the audio output.

The principal control of *Loopalooza* is the syncopation slider. Each step of the slider corresponds to a different rhythmic variation of the input. At the bottom, a totally de-syncopated pattern is generated. As the slider is moved towards the top, the musical events of the input are shifted to syncopating positions. The top most pattern cannot be further syncopated. A pattern with the original syncopation but with quantized onset positions is found in one of the slider positions (shown as a red square on the slider, see **Figure 3**).

In the center of the device, a display area is found. It displays a graphical representation of the audio input and the transformations applied. At the top of the display a representation of the audio input is found. Pulses containing onsets are marked as red bars. At the bottom, black bars indicate the displaced onsets. In addition a playback indicator shows which event of the original audio loop is being played back.

The sensitivity of the onset detection can be adjusted by controlling the "onset threshold" value found on the upper right corner of the device. This parameter controls the sensitivity of the onset detection. It affects the number of detected events in the input pattern. Hence, different rhythmic patterns and transformations can be generated with the same audio input.

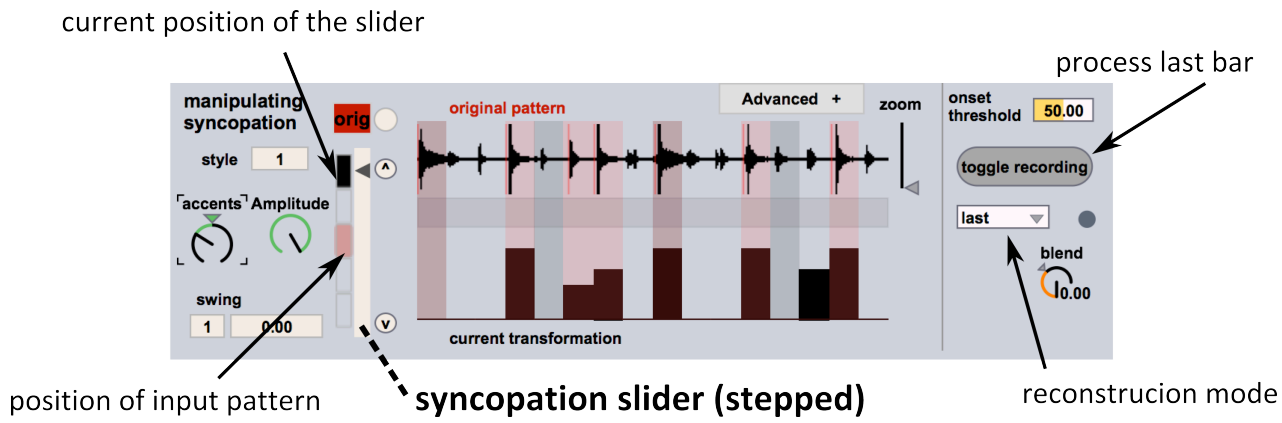

Figure 3. The *Loopalooza* MaxForLive device user interface

Loopalooza works with audio drum loops as well as audio containing melodic instruments or complex musical mixtures. To demonstrate the usage of the *Loopalooza* interface and to provide examples of the musical results it can create, demonstration videos are available here [13]. Additionally, the *MaxForLive* can be downloaded here [13] for *Ableton Live* users to experiment and use it in their music projects.

5. DISCUSSION

Automatic rhythm manipulations rely on information about the onsets, offsets and durations of rhythmic events. In audio loops, especially those comprised of complex mixtures, the retrieval of such information can be a challenging task since many sounds overlap, and hence precise temporal boundaries may be difficult to determine.

However, in our system, manipulating the syncopation requires only the most salient events to be detected. The syncopation transformations produce a larger number of variations with input patterns of a moderate number of events (fewer than 50% of the number of pulses) as discussed in [12]; too few or too many events result in fewer displacements and hence fewer variations.

Within the context of *Loopalooza*, we allow users to have control over the onset detection sensitivity and consequently over the number of detected events. Often interesting results can be obtained using a lower sensitivity for onset detection, i.e. under-detecting the number of onsets in the audio loop, and hence retaining only the most prominent events. In contrast, setting a high sensitivity and detecting a high number of onsets could be counter-productive with respect to the syncopation transformations.

Whenever the number of detected events fills up the metrical grid, this blocks possible displacements. In this case, the original transformation algorithm [12] employs dynamic accents to generate syncopation. A similar approach could be implemented in the audio domain. First, the perceived loudness of each event should be estimated. Second, a corresponding attenuation should be applied on each event resulting in stressing certain events compared to others. Increasing the syncopation would then correspond to displacing the accents to off-beat positions.

Our system does not attempt beat tracking or bar estimation like other systems [1-5]. We do not consider this a critical limitation since this information can be retrieved from the *Ableton Live* environment. However, if the input is not aligned to the tempo and meter of the environment, the loop will not be aligned with the metrical grid. This can result in the system interpreting the loop as a highly syncopated pattern rather than an unaligned pattern.

Another limitation of the *Loopalooza* system is that it does not segregate multi-timbral audio into separate streams. This functionality would require the segregation and manipulation of each instrument in real-time, which is a significant technical challenge. However, *Ableton Live* allows the user to have each instrument in a seperate track. The user can run several *Loopaloza* devices in parallel in different audio tracks, and in this way, he can manipulate each instrument input independently.

The resampling reconstruction does not employ lossy signal processing, thus the original audio quality is preserved. However, the method would be greatly improved by automatically evaluating and choosing an appropriate audio snippet for filling each particular gap.

The time scaling method is problematic in certain cases with respect to rhythm alignment and audio quality. Syncopation transformations can be drastic, resulting in high time scaling ratios. This introduces two problems, alterations in the tempo of particular segments and loss of audio quality. On this basis, we recommend the resampling approach as the default method for reconstruction.

6. CONCLUSIONS

In this paper we have presented *Loopalooza*, a system that allows users to create a multitude of rhythmic variations from a single audio input. Our contribution combines techniques and algorithms to estimate and manipulate the syncopation in rhythmic structures from audio in

real-time. It uses a simple but effective interface. A single slider allows the user to explore the musical possibilities of syncopation. *Loopalooza* is implemented as a freely available *MaxForLive* device to be used by DJs and musicians alike in live music performances and remixing. In future work we plan to improve the onset detection stage and explore more techniques for rhythmic reconstruction which best preserve the original audio quality.

Acknowledgments

This work was partially funded by the FCT post-doctoral grant (SFRH/BPD/88722/2012) and by the Media Arts and Technologies project (MAT), NORTE-07-0124-FEDER-000061, financed by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, "Fundação para a Ciência e a Tecnologia (FCT)".

7. REFERENCES

[1] F. Gouyon, L. Fabig, and J. Bonada, "Rhythmic Expressiveness Transformations of Audio Recordings: Swing Modifications," in *6th Int. Conference on Digital Audio Effects*, 2003, pp. 94–99.

[2] J. Janer, J. Bonada, and S. Jordà, "Groovator - an implementation of real-time rhythm transformations," in *121st Convention of the Audio Engineering Society*, 2006.

[3] J. A. Hockman, J. P. Bello, M. E. P. Davies, and M. D. Plumbley, "Automated rhythmic transformation of musical audio," in *11th Int. Conference on Digital Audio Effects (DAFx-08)*, 2008, pp. 177–180.

[4] E. Ravelli, J. P. Bello, and M. Sandler, "Automatic Rhythm Modification of Drum Loops," *IEEE Signal Process. Lett.*, vol. 14, no. 4, pp. 228–231, Apr. 2007.

[5] J. P. Bello, E. Ravelli, and M. B. Sandler, "Drum Sound Analysis for the Manipulation of Rhythm in Drum Loops," in *International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2006, vol. 5, pp. 8–11.

[6] G. Sioros and C. Guedes, "Automatic Rhythmic Performance in Max/MSP: the kin.rhythmicator," in *International Conference on New Interfaces for Musical Expression*, 2011, no. June, pp. 88–91.

[7] G. Sioros and C. Guedes, "A Formal Approach for High-Level Automatic Rhythm Generation," in *Bridges 2011 – Mathematics, Music, Art, Architecture, Culture Conference*, 2011.

[8] G. Sioros and C. Guedes, "Complexity Driven Recombination of MIDI Loops," *Int. Soc. Music Inf. Retr. Conf.*, pp. 381–386, 2011.

[9] G. Madison, G. Sioros, M. E. P. Davies, M. Miron, D. Cocharro, and F. Gouyon, "Adding syncopation to simple melodies increases the perception of groove," in *Conference of the Society for Music Perception and Cognition*, 2013.

[10] G. Sioros, A. Holzapfel, and C. Guedes, "On Measuring Syncopation to Drive an Interactive Music System," in *International Society for Music Information Retrieval Conference*, 2012, pp. 283–288.

[11] F. Gómez, E. Thul, and G. Toussaint, "An experimental comparison of formal measures of rhythmic syncopation," *Proc. Int. Comput. Music Conf.*, pp. 101–104, 2007.

[12] G. Sioros, M. Miron, D. Cocharro, C. Guedes, and F. Gouyon, "Syncopalooza : Manipulating the Syncopation in Rhythmic Performances," in *International Symposium on Computer Music Multidisciplinary Research*, 2013, vol. 10th, pp. 454–469.

[13] "Loopalooza," 2014. [Online]. Available: http://smc.inescporto.pt/technologies/loopalooza/.

[14] M. Zbyszynski, D. Zicarelli, and R. Collecchia, "fzero~ - Fundamental Estimation for MAX 6," in *International Computer Music Conference (ICMC)*, 2013.