Rethinking Reflexive Looper for structured pop music

Marco Marchini
UPMC - LIP6
Paris, France
marco.marchini@upmc.fr

François Pachet
Sony CSL
Paris, France
pachet@csl.sony.fr

Benoît Carré
Sony CSL
Paris, France

ABSTRACT

Reflexive Looper (RL) is a live-looping system which allows a solo musician to incarnate the different roles of a whole rhythm section by looping rhythms, chord progressions, bassline and more. The loop pedal, is still the most used device for those types of performances, accounting for many of the cover songs performances on youtube, but not all kinds of song apply. Unlike a common loop pedal, each layer of sound in RL is produced by an intelligent looping-agent which adapts to the musician and respects given constraints, using constrained optimization. In its original form, RL worked well for jazz guitar improvisation but was unsuited to structured music such as pop songs. In order to bring the system on pop stage, we revisited the system interaction, following the guidelines of professional users who tested it extensively. We describe the revisited system which can accommodate both pop and jazz. Thanks to intuitive pedal interaction and structure-constraints, the new RL deals with pop music and has been already used in several in live concert situations.

Author Keywords

Live-looping, musical human-computer interaction, realtime music system, HCI design

ACM Classification

H.5.5 [Information Interfaces and Presentation] Sound and Music Computing, J.5 [Arts and Humanities] Performing arts (e.g., dance, music), I.2.11 [Artificial Intelligence] Multiagent systems.

1. INTRODUCTION

Live-looping is a technique for performing live music that originated from magnetic tape recorders. It consists in capturing audio (on stage or previously) using a device and replaying it back in loops. Terry Riley was the first musician to use tape loops and tape delay/feedback systems in his performances in the 60s [7]. The technique was later used in *minimal* music to create endless repetitions of rhythms. Live-looping became popular in the 90s after the commercial release of several looping devices (notably Roland and Digitech loop pedals) affordable and easy to use. Loop pedals offer the ability for a single musician to create multiple layers to their live music, thus creating complex musical textures such that of a *full band*. In pop/folk they were typically used to enrich instrumental music with repeated rhythmic patterns or to create songs with static harmony.

A search for the keyword "loop pedal cover" on Youtube produces, at the time of this paper, about 145k matching videos, many of which are recent amateur performances. Typical performance consists of a *build-up* part where the musician records multiple layers of sound on a fixed chord sequence (usually four chords). The chord sequence repeats throughout all the performance being used by the musician as an accompaniment over which s/he sings or plays solos. However, musicians rarely include a *break-down* section where tracks are muted and a new chord sequence is introduced. This occurs because the common loop pedal makes it difficult to produce transitions between sections with different durations and/or chords. As a consequence, only songs with static harmony are performed with a pedal loop. For example, whereas we can find plenty of covers for Beatles songs with static harmony, we could not find any solo performance of *I feel fine* with a loop pedal, as this song is not with static harmony. In *I feel fine* chords change from verse to chorus, and the two sections have different durations.

Reflexive Looper (RL), proposed by Pachet et al. [6], brings the potential to revive the creative use of live-looping. Looping with RL goes beyond the play-back of sound recorded live, as the recorded input is transformed in several complex ways before being played-back (slicing, pitch-shifting and concatenative synthesis, see Sec. 2.4). Whereas a common loop pedal determines the playback material depending on the position in the loop and the button pressed by the user, each looping-agent in RL (chords, bass, drum, and voice) determines the playback material by solving an optimization problem, which is updated every few milliseconds.

We argue that the potentials of RL were not fully exploited yet because of its focus on *jazz* improvisation, and the lack of planning device for guiding the interaction. Several human-computer improvisation systems allow for plans that can guide the performance in a strict or loose manner [4]. Similarly to the Continuator [5], RL accumulates an ever growing database of musical phrases from which to draw from in fabricating the response, making them less predictable as the performance evolves. Not only such behavior contradicts most of Pop music, but also any music containing a begin, an evolution and an end.

This article describes a new software architecture that solves the problem of performing structured pop music. Pop music, as other genres, is based on structured repetition, which requires a significant degree of planning. We show how the proposed system allows a solo musician to play *I feel fine* by The Beatles live, something that was impossible with the common loop pedal. We show that using RL the musician

NIME'17, May 15-19, 2017, Aalborg University Copenhagen, Denmark.
.

can easily produce seven tracks in real-time that respect the transitions between sections with few explicit instructions given by the musician during the performance. Nonetheless, the interaction that we designed for pop is also applicable to structuring improvisation in other contexts (e.g. jazz). The new system is therefore usable in a multitude of live scenarios with timing constraints: stage performance, piano bar, jazz improvisations, etc.

2. REDESIGNING REFLEXIVE LOOPER

We have implemented the new system as a VST plug-in and we also improved each of the constituent blocks of the system which we review in this section: playing-mode tracking (Sec. 2.1), chord substitution (Sec. 2.2), and chord grid (Sec. 2.3). Lastly, in Sec. 2.4 we summarize the typical interaction with RL through a short example, which shows the potentials of implicit interaction.

2.1 Playing-mode tracking

Playing-mode tracking is a key aspect of RL allowing it to *pro-actively listen to* the musician through an intelligent real-time analysis of the audio input. The analysis aggregates a custom frame-based playing-mode recognition, with standard silence detection and onset detection algorithms to track and tag musical phrases according to playing-modes. Note that the current system does not need to recognize chords from the audio input as those can be easily deduced from the chord grid over which the musician is playing (this will be explained in more details in Sec. 2.3).

In the previous prototype (see [6]) a MIDI-guitar was required for playing-mode recognition, and this was not accepted by most guitarists, as they wanted to use their own guitar without adding extra sensors. Playing-mode recognition allows the system to discriminate between musician playing *chord* (strumming, arpeggiated, etc.) from musician playing *melody*, *drums*, or *bassline* . In the original system, this was done by classifying features extracted from the MIDI-events received from the guitar output [6].

In our new implementation we free the musician from the need of MIDI-guitar by adopting another approach based on extracting features directly from the guitar audio-signal as explained in [2]. This approach makes RL compatible with any electric or electro-acoustic guitar, as long as sufficient training data for each playing mode is provided for the specific guitar[1].

For each playing mode that the musician wants to loop (generally chords, drums and bassline, but not melody), the system instantiates a *looping-agent*. This is a real-time algorithm that runs on a dedicated thread, that keeps track of all the musical phrases played by the musician so far in the specific mode, labels them with the appropriate chord names, and prepares loops to be played on the upcoming bars. Each looping agent schedules loops in one particular mode, according to the musical phrases collected so far, and is subject to real-time scheduling conditions (as described in [1]). We tune the parameters of each agent (the parameters are described in Sec. 2.3 as constraints to the constraint optimization) according to their mode: e.g. unlike chord, drum loops do not respect harmonic constraints and are never transposed.

2.2 Chord substitutions

The previous system, described in [6], uses the concept of *chord substitution* coming from harmony theory. This allows a loop to be reused in harmonic contexts which are similar to the one where the loop was recorded. For example, the chord A minor is an acceptable substitution of the chord C major, so if the musician has played an A minor but no C major, the system can use the first in the place of the second. When loading a new chord grid, the system obtains all the possible substitutions in advance by calculating a harmonic distance between all chord pairs. The pairs for which the distance is below a given threshold (adjustable by the user) are allowed substitutions each with a cost proportional to the harmonic distance.

To calculate the harmonic similarity between two chords we first transform each into a histogram over the pitch classes. We give a weight of one to each pitch class in the chord (e.g. the chord C7 corresponds to the pitch classes {C, E, G, Bb}) and two to the third of the chord, and zero to the remaining pitch classes in the octave. Then we compute the cosine similarity between the two deriving normalized histogram as a similarity between chords. From the harmonic similarity x between chords, we derive the harmonic distance as $d = x - 1$.

Chord substitution allows the system to recycle loops for different harmonic contexts. In the new prototype we combined it with *audio-transposition* for a maximal flexibility of the system. With audio-transposition musicians are not required to play many chords, and songs can also modulate to different tonalities. We introduced audio-transposition into RL by integrating already existing pitch-shifting algorithms (we use the Rubber Band Library[2] as it is an open source project, also other commercial pitch shifter software can be set for better quality). To handle transposition, when loading the chord grid, the system also calculates the cost of substitution for each transposed pair in an interval range defined by the user. We consider transposition as an additional cost that is combined to the cost of the substitution, therefore promoting small transpositions over big ones.

The user can also set a maximal allowed transposition interval (in semitones) for each looping agent, and each agent will autonomously take decisions about the optimal transposition and substitution to use at each moment, as explained in the following section.

2.3 Chord grid

Before starting to play, the user selects a *chord grid* over which s/he wants to improvise. RL does not make use of any music material coming from previously recorded sessions or any external backtrack (although in a future implementation it might be interesting to allow that). Instead, as any live looper, RL records audio material during the session and plays it back in real-time without musical breaks. The saved chord grid does not include any prepared backtrack, nor any pre-defined transpose instruction, serving for the only purpose of defining harmonic constraints for the selection of material to play-back.

Once the chord grid and a BPM have been selected by the user, RL constructs and constantly maintains a list of future beats $(B_i)_{i=0,\dots,n}$ ordered in time, where the B_0 is the current beat, B_1 is the next beat, and B_n is the further away in time. The number of beats n in the list is determined by the BPM, so as to include the beats in the next 10 seconds (or another duration chosen by the user) at least (e.g. if BPM$=$ 120 then $n = 20$). At each beat B_i corresponds a chord N_i in the grid, and a vector M_i containing other contextual information about the beat (metrical position of the beat, section number, and more).

All of the musical phrases that have been recorded so far are sliced into chunks c of duration one beat and stored in the collection $\mathcal{C}$. Each chunk c played by the musician inherits

[1]Our training set contain at least 10 minutes of recording for each playing mode.

[2]See `http://breakfastquay.com/rubberband/`

the information about the beat over which it was played. This allows us to define $N(c)$ and $M(c)$, the chord content and the contextual information of the chunk respectively. We then define $\mathcal{C}^*$ as the extended collection of all possible transpositions of chunks in $\mathcal{C}$ (the transposition is the range of integer semitones between -7 and +7 to account for a full octave, but can be restricted or incremented by the user). For a chunk $c* \in \mathcal{C}^*$ transposed from $c \in \mathcal{C}$ we define $M(c*) = M(c)$ and $N(c*) = \text{transp}(N(c))$, where we transpose the chord of the same amount as the chunk. Lastly, we define $\mathcal{C}_i$ as the set of acceptable candidate chunks at beat B_i, this consists of all chunks in $\mathcal{C}^*$ that have harmonic distance lower than a given threshold to the target chord N_i, for which M_i is compatible to $M(c)$. The latter condition is verified if the metrical position in M_i is the same as $M(c)$, and also if other conditions depending on the settings chosen by the user apply.

At any given instant, each *looping-agent* in RL determines the playback material by solving the optimization problem:

$$\underset{(c_i)_{i=0,\ldots,n}}{\arg\min} \left(\sum_{i=0}^{n} \text{replCost}(c_i) + \sum_{i=0}^{n-1} \text{discCost}(c_i, c_{i+1}) \right)$$

under the constraints:

$$c_i \in \mathcal{C}_i$$

where $\text{replCost}(c_i)$ is the cost of replacing the chord N_i with the chord $N(c_i)$ given by the harmonic distance and the transposition, and $\text{discCost}(c_i, c_{i+1})$ is a cost of transition which depends on $M(c_i)$ and $M(c_{i+1})$. The transition cost penalizes discontinuities in time and transposition, especially within the bar, thus favoring the reuse of long sequences of audio over short ones. The problem of finding an optimal sequence of chunks $(c_i)_{i=0,\ldots,n}$ is efficiently solved using a dynamic programming algorithm that can find a solution, with computational complexity bounded by the square of $\max_i \|\mathcal{C}_i\|$.

2.4 Illustration of an implicit interaction

We illustrate how the system works with the short music example represented in Figure 1. In the example, the musician has chosen to play over an eight bar long chord grid, at a tempo of 90 BPM. After hitting the *start* button a pre-count of one bar announces the grid; this is the sole button to be used as the rest of the interaction is completely implicit. In this example the musician plays during the first four bars while leaving the system continue alone on the remaining four. More precisely, in bars 1-2 he plays an arpeggio, and in bars 3-4 plays a bassline.

During bars 1-2 the system provides no back-loops as it is launched from an empty state. The *chord-agent* records the arpeggios and fabricates a musical continuation, which gets updated continuously as the audio material gets in. Such continuation stays however inhibited as long as the musician continues playing the arpeggios, so is not released immediately.

At bar 3, the shift to a new mode of playing (bassline) is detected by the system, this removes the inhibition of the chord-agent which is thus faded-in immediately. During bars 3-4 the same process is repeated for the *bass-agent*, which records the bassline in an inhibited state until bar 5 where the musician stops playing altogether, thus the bass agent continuation of the bassline is also released.

These examples show how to produce a bass and chord accompaniment with RL within four bar, without any explicit interaction. At this point the musician could already play a melody or sing on top of the accompaniment which would continue playing even on different chord sequences. We show

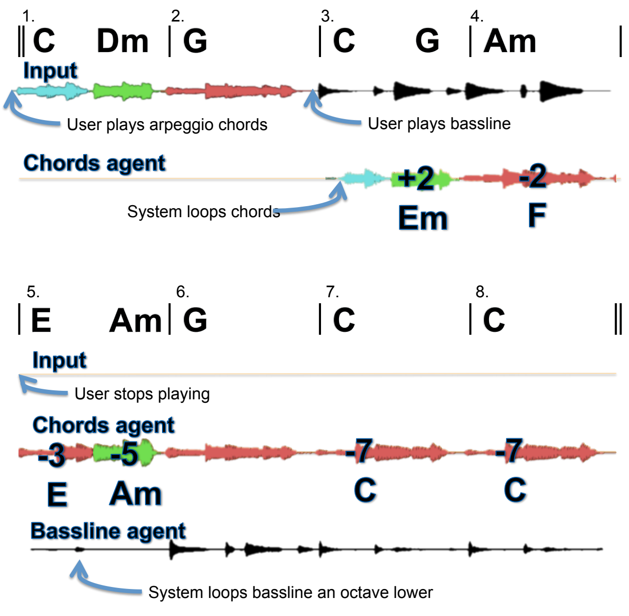

Figure 1: This eight-bar long example illustrates the basic use of RL. It shows how implicit interaction defines events when each agent is activated.

in the next section how this implicit interaction, combined with structure-constraints, can build up the arrangement of a complete pop song.

3. EXTENDING THE SYSTEM FOR POP

In the new interface design, we attempted to create not only a tool for engaging stage performance but also a creative music tool for home performance with an intuitive interaction. As RL makes it easy to build full band styles by recording few bars, musicians often completely reshape their performance within the first few tests. RL is described as a creative tool by professional musicians (see [3]). We argue that improving the interface design reinforces such creative potentials, by avoiding disruptions of creative flow.

In this sense, our goal is to enable musicians to quickly explore the musical possibilities given by the system (see [8]). With this goal in mind, we believe adopting the appropriate metaphors and UI-feedback leads to the most intuitive interface, which allows controlling the system with minimal effort. The second author has been a jazz tester of the system since he invented it. Other occasional tester have given important feedback. During the last two years, we iterated numerous tests with third author of this article and pop musician Benoît Carré. This collaboration led to numerous minor-improvement and adaptations that summed up into a completely new system. The result of the collaboration is two-fold. From one side, we developed the UI-interface metaphors that are relevant to musicians, which helped us designing an appropriate interface. On the other side, we learned from him which are the specificity of pop music that require specific attention.

Most musicians testing the system wanted to gain control over the looping-agents in an explicit way by pausing, disabling or forcing them when desired. We have tested a commercial MIDI pedal interface to do so (see Sec. 3.2). We noticed that our musicians would use this feature extensively in a testing session to test results on-the-fly but in concert situations they would tend to use this resource minimally because in that case structure constraints are generally suffi-

Table 1: Chord sequence in the song *I feel fine* by the Beatles.

Section	Chord sequence			
intro:	A	Asus4	G	Gsus4
	D	Dsus4	D	Dsus4
verse:	D	Dsus4	D	Dsus4
	A	Asus4	A	G
	D	Dsus4		
chorus:	D	F#m	G	A
	D	F#m	Em	A
bridge:	C	Csus4	C	Csus4
impro:	C			

Table 2: Structure constraint table used for the song *I feel fine*. On the left the table for the *chord* agent, on the right the analogous tables for *bass*, *drums* and *voice* agents are represented in compact form.

Chords:	intro	verse	chorus	bridge	impro
intro	✔	✔	✗	✔	✗
verse	✔	✔	✗	✔	✗
chorus	✗	✗	✔	✗	✗
bridge	✔	✔	✗	✔	✗
impro	✗	✗	✗	✗	✔

Bass:

■	■	■	□	□
■	■	■	□	□
■	■	■	□	□
□	□	□	■	□
□	□	□	□	■

Drums:

■	■	■	□	□
■	■	■	□	□
■	■	■	□	□
□	□	□	■	□
□	□	□	□	■

Voice:

■	□	□	□	□
□	■	□	□	□
□	□	■	□	□
□	□	□	■	□
□	□	□	□	■

cient and more convenient. We also added the possibility to create additional textures with manually activated agents which can go beyond the pre-trained playing modes. This feature made good use of the pedal interface and was used by musicians more extensively in concert situations as well.

3.1 Enforcing musical structure

The chord grid may be structured into different sections such as intro, verse, chorus, bridge, etc. In pop music structure is important and the transitions between one section to another are often highlighted by variations in instrumentation. We want RL to realize these transitions properly without the need of extra buttons during the performance. For this reason, we introduced *structure-constraints*, which constrains the agents to use only sound material originating from specified sections.

A typical structure-constraint is defined in the RL interface by marking check-boxes in a matrix where the rows i represent the *target* sections and the columns j the *source* sections. Marking the (i, j)-cell as ON signifies that the system can use audio recorded during section j when generating loops for section i. An example matrix used for the song *I feel fine* is shown in Tab. 2.

Thanks to configurable structure constraints the pop musician realized a cover of *I feel fine* by the Beatles. The chord sequence for each song section is reported on Tab. 1: intro, verse, chorus, bridge, and impro. The musician repeats each section several times in a row and also alternates verse and chorus, depending on the duration of the version he wants to realize. In Fig. 2 we represented an example performance that we have recorded. The top three rows represent the performance of the musician, in terms of button pressed, instrumental phrases played, and vocals. The rest of the rows show the loops played by the system with incoming arrows from their origin phrase. Throughout the three minute performance, the musician only uses four explicit controls: start, fx, mute-instruments, stop.

Despite the few explicit controls, all of the orchestrating actions (activate loops, stop loops etc.) that are showed in Fig. 2 result as a consequence of the structural constraints chosen by the musician and his playing. Tab. 2 shows the structural constraints chosen by the musician for chords, bass, drums, and voice. The tables indicate whether recorded material from one section can be used in a second. For example, the table for the chords indicates that during the chorus only the audio patterns recorded in the chorus can be used. But in the verse, it can also use audio recorded from the intro, and so on.

3.2 Pedal control

In early prototypes of RL, the controls were limited to a *start* button and a *stop* button: a mouse click was thus sufficient for demoing the system. In the last two years, we have tested several MIDI devices for controlling the system, we still have not found the ideal controller. It is a common habit from musicians to ask for "just a button more" to control the latest aspect of their song that has become suddenly important. The risk of following the requests too literally is that we end up with plenty of unrelated controls that are too specific and are not remembered by the users. This is why we concluded that, before deciding what device we are going to use, we needed to define appropriate UI metaphors that would allow organizing the controller and provide useful feedback to the users. Following such metaphors also allows us to state which features the ideal device should possess.

The most important metaphor in controlling the system is that of a *looping-agent* which we have already introduced in this paper. Besides being a component of our software, a looping-agent is also intuitively understood by musicians as a track that automatically assumes a *state* over which we might act on: on, off, recording, automatic, or manual. This also seems a natural extension of the traditional loop pedals. For this reason, organizing the interface by looping-agents results in the most consistent interface as each of them is represented by a single controller: be it a button or a switcher or another type of controller.

The device we are thus looking for should allow switching between states, but also provide feedback from the software when it is auto-piloted by the software. The simplest realization of this principles is given by a button with a LED that can change colors to indicate the state. We use a single press of the button to switch between states: *automatic*, for using the implicit interaction, *on*, to force the playback of the loop, or *off* to mute the loop. When in automatic-state, the LED can also provide a feedback to indicate when the loop is ready to play music, signifying that the looping-agent has recorded enough material to provide a response. We found few commercial pedals that would allow this possibility to give LED feedback, and we have been using at the moment the Keith McMillen Softstep pedal for the purpose[3].

3.3 Video installation in concert

The current RL system was used in several concert scenario in the last two years and participated in a competition of new music instruments. In concert scenarios, musicians come with a prepared performance, which means having prepared the chord grid and the structure constraints; there

[3]See https://www.keithmcmillen.com/products/softstep/

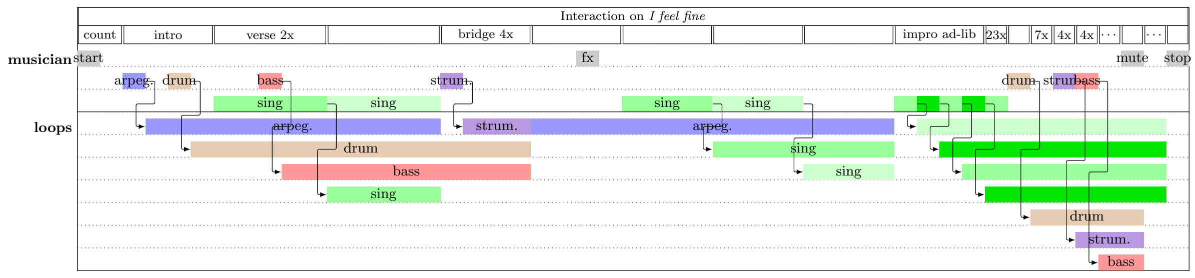

Figure 2: Tracks played on the song *I feel fine*. The first three rows contain: the pedal actions performed by the musician, the instrumental track played, and singing. The remaining rows show the loops generated by the system connected to their source.

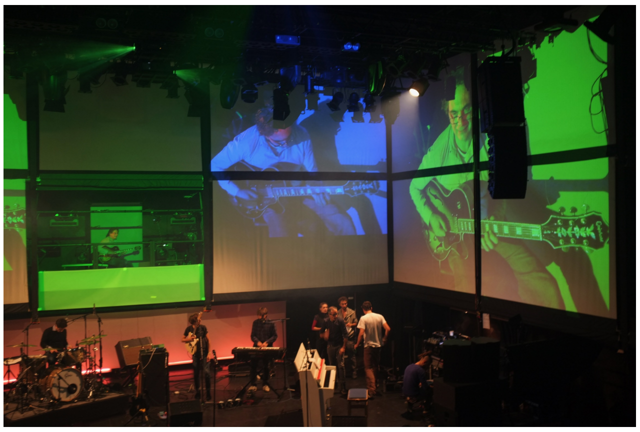

Figure 3: Musician playing with RL on stage. The video projections, in different colors, show the musical agents playing with the musician in real-time.

is, however, space yet for some improvisation. We generally received positive feedback from the audience, although some people commented that it was hard to follow all the complex transformations that RL applies to the playback material and, because of this, some people thought that the musician was playing over pre-recorded backtracks.

Pop music is also a genre where visuals are becoming increasingly important during concerts. RL offers a possibility for creating engaging visuals of the performances by looping videos. To best enrich the experience of the audience from a didactic and entertaining point of view, we implemented a video installation that displays the system status by playing back live video transformed similarly to the audio and in sync with the music. This allowed the audience to follow the performance in a way that was not possible before. Each looping-agent is visible as a looped video that is synced with the music.

Fig. 3 shows the use of the video installation during a concert. To realize the installation, we implemented an application with openFrameworks[4], connected to a video camera capturing the performer live and recording the stream of video. The application communicates via OSC to the VST plug-in which, in turn, sends a clock-signal and the instructions for reproducing the video loops in sync with the looping-agents.

4. DISCUSSION

Our new system solves the problem of performing structured pop music with a looper. Structure is given by the section-structure of the song, which is set by the musician, and enforced by specifiable structure-constraints. Each phrase learned by the system is labeled according to the section it was recorded from. In each section the musician can specify structure constraints that limit the section of origin of samples used to generate the responses.

In addition, we have improved the interface with the system by simplifying it for the user. We implemented the system as a VST instrument, which allows the musician to work within his own DAW. By classifying playing modes directly from audio, we removed the necessity for a MIDI instrument, thus allowing musicians to use any instrument. We introduced a MIDI pedal with LED feedback allowing to extend the basic interaction with the system such as start and stop buttons with additional controls over the looping-agents. This allowed control over sound and textures which makes RL a creative tool for experimenting with sound.

We showed, with the example of *I feel fine*, how structure-constraints allow creating contrasts in rhythmic style between alternating sections of the piece (e.g. verse, chorus, bridge). We showed that this method requires using very few explicit controls during the performance *and* recording only few bars of instrumental performance to generate all the tracks, something that was impossible with a traditional looper.

5. CONCLUSION

Reflexive Looper (RL) is a live-looping system inspired by jazz combos which allows a solo musician to incarnate the different roles of a whole rhythm section by looping rhythms, chord progressions, bassline and voice. The system works without pressing any button by exploiting implicit interactions for guiding intelligent live-looping agents. Implicit interactions, embodied in the music performance itself, are possible thanks to a real-time AI-engine listening to the audio produced by the musician. The AI continuously classifies music input based on its playing-mode (e.g. strumming, bass-line, or melody) and feeds databases of musical phrases in different playing-modes to their respective music-playing agents. Each agent decides what to loop back at any moment depending on what the musician is currently playing. Using concatenative-synthesis and transposition, the agents are also able to respond with new musical phrases derived from the ones previously learned.

The original implementation of RL was applied to jazz improvisations over a chord grid with interesting musical results. To cope with pop music structure, we introduced specifiable structure-constraints limiting the system response inside some parts of the song (e.g. verse, chorus, or bridge). The user can specify such constraints in advance to enforce a predefined structure. In addition, we introduced a set of additional music-agents that the user can activate manually

[4]See `http://openframeworks.cc/`

with a pedal interface to enrich the musical texture. We designed the interaction with the pedal so that it is as simple as it can get, by responding pro-actively to the commands of the musician in a context-aware fashion. This allows a smooth coexistence of both implicit interactions and explicit interactions.

6. ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n. 291156).

We thank Fiammetta Ghedini for assistance with video editing, and Pierre Roy for the numerous discussions on code optimization.

7. REFERENCES

[1] R. B. Dannenberg. Real-time scheduling and computer accompaniment. 1989.

[2] R. Foulon, P. Roy, and F. Pachet. Automatic classification of guitar playing modes. In *International Symposium on Computer Music Modeling and Retrieval*, pages 58–71. Springer, 2013.

[3] J. McCormack and M. d'Inverno. Designing improvisational interfaces. In *Title: Proceedings of the 7th Computational Creativity Conference (ICCC 2016). Universite Pierre et Marie Curie*, 2016.

[4] J. Nika. *Guiding human-computer music improvisation: introducing authoring and control with temporal scenarios*. PhD thesis, Université Pierre et Marie Curie Paris 6 - Ircam, 2016.

[5] F. Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

[6] F. Pachet, P. Roy, J. Moreira, and M. d'Inverno. Reflexive loopers for solo musical improvisation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2205–2208. ACM, 2013.

[7] M. Peters. The birth of loop. *Retrieved May*, 25:2004, 1996.

[8] R. W. White, B. Kules, S. M. Drucker, and m.c. schraefel. Supporting exploratory search, introduction, special issue, communications of the acm. April 2006.