A MUSIC LOOP EXPLORER SYSTEM

Sebastian Streich  
YAMAHA Corporation  
Center for Advanced Sound Technologies  
203 Matsunokijima, Iwata  
Shizuoka 438-0192, JAPAN  
{sstreich,beesuan}@beat.yamaha.co.jp

ABSTRACT

Music loops, seamlessly repeating segments of audio, are an important ingredient for remixes and mash-ups. By recombining loops taken from complete tracks or from loop libraries not only professional DJs but even musical laypersons can enjoy the experience of music creation. One key aspect is to identify what sounds good together. To facilitate this selection process, we present a system for exploring collections of music loops through a graphical user interface that allows playful interaction with the content. The system first extracts loop segments from a selection of music tracks. The loops are then visualized as graphical objects in a GUI. Depending on their needs, the users can switch between various criteria for the visualization of the objects which is based on a set of manually or algorithmically provided features. Interaction with the objects triggers playback and simple effects on the loops.

1. INTRODUCTION

Loop centred music software is around now for several years already; the pioneering “ReCycle” was released by Propellerhead Software as early as 1994. With many followers from other music software companies, this type of application enjoys high popularity until today. Improvements and extensions of functionality were introduced over the years, for example to have a more reliable beat slicing or to allow smooth switching between loops in live performance situations. With our proposed system we want to demonstrate possible improvements in two areas that have seen only modest advancement since the early days. The first one consists in the automatic extraction of loops from entire music tracks. This functionality enables users to utilize material from their entire music collection for remixing without the need for laborious manual cutting. As we have reported the underlying technology in previous publications (see [6]), we will cover this part only very briefly in this publication.

The second improvement, which is the main topic of this paper, concerns the loop selection or file browsing section of the above mentioned family of programs. While modern programs of said type often include some library management functionalities, they fail in providing a compact overview of the entire loop collection. Also, even when pre-views are featured, it is not very straightforward to quickly try many different combinations of loops and thus identify interesting candidates for building up a new track. Here, we combine concepts for collection visualization known from the domain of music information retrieval (e.g., [1] or [4]) with targeted functionalities in order to allow playful exploration of a loop collection.

![Figure 1. Overview diagram of the main components in our loop explorer system.](image)

2. SYSTEM DESCRIPTION

The loop explorer system presented here has four main units. These are the loop segment extraction, the feature extraction, the visualization in the GUI, and the playback management. Figure 1 shows an overview diagram. As can be seen, the loop segment extraction block is an optional component as long as a loop collection is already available. However, this functionality opens up very interesting perspectives by giving access to a multitude of loop segments taken from a user’s entire music collection or just from his or her favourite tracks.

At current state the system is not yet integrated into one single application. The loop segment extraction is so far realized in form of a MATLAB script, the feature extraction is done with a separate executable, and the GUI and playback component are programmed in the TCL/TK scripting language. Ultimately, the idea is of course that the system either forms an independent stand-alone application or, even better, is integrated as part of the library management inside a loop sequencer software.
2.1. Loop Segment Extraction

The loop segment extraction is based on an adaptation of a music structure detection system described in [7]. We will discuss the extraction algorithm only very briefly in this paper and refer to [6] for a more in depth explanation.

2.1.1. Loop Region Detection

Similar to the original approach, we start with computing a similarity matrix based on a chroma feature, the harmonic pitch class profile (HPCP, see [2]). Put simply, this feature captures information about the sounding pitches and some of their harmonics all mapped into the range of one octave. It is computed at discreet time intervals on short excerpts of the signal (~90 ms), so that changes can be tracked along time. The similarity matrix is composed of the similarity values for all possible pairs of these time intervals. In our case, the similarity is determined by the cosine distance (see [7] for details) between the respective chroma feature vectors.

When visualized, this matrix representation has the nice property of revealing repetitions in the signal by displaying them as line elements parallel to the diagonal. While the original method in [7] identifies repetitions of structural units (e.g., verse and chorus) throughout the entire track, our adopted algorithm focuses on regions in the signal where multiple consecutive and regular repetitions of shorter segments occur. The rationale behind this is that such a repetition pattern points us to parts of the music where loops or loop-like material is used. We refer to these parts as "loop regions". The spacing between the line segments in the similarity matrix also tells us the approximate duration of a single loop cycle for each loop region.

2.1.2. Loop Instance Extraction

Apart from the right loop cycle duration, it is also important to find a suitable position for cutting the audio, so it really repeats seamlessly. By basing the loop region detection on tonal similarity, we have already avoided problems related to harmonic discontinuities. Now we also need to take care of rhythm and timbre properties. We therefore first locate the beat positions within each loop region [5]. We then compute the averages of the lower order Mel-frequency Cepstral Coefficients (MFCC) for each beat interval as a timbre descriptor. Based on the detected loop duration from the previous step, we then compare the MFCC vector of each beat with that of its corresponding beat in the following loop cycle using again the cosine distance measure. We sum up the similarity scores of each group of three consecutive beats. Finally, we cut out the loop instance that starts with the group of beats yielding the highest similarity value, as this indicates the smoothest timbre continuity upon wrap around.

The loop instances are stored as separate audio files in the loop collection. The list of detected beats could be preserved as well to serve as slicing point information.

With our test material, that did not contain excessive swing rhythms, we found it sufficient to simply keep the average inter-beat interval (IBI) as the reference.

2.2. Feature Extraction

The features we extract from the loops belong to the category of so-called low-level descriptors. Such descriptors are rather straightforward to compute in a frame-by-frame fashion and are commonly found in audio or music classification systems [3]. They reflect specific physical or statistical properties of the audio signal, like for example the number of zero-crossings or the signal energy per frame.

In our current system the results of the feature extraction process for the entire loop collection are stored in a so-called collection file. Each line of this ASCII file contains the path and filename of a loop instance followed by the list of descriptor values. This solution is not very efficient, but it allows an easy separation of the feature extraction and the GUI application. Ideally, this data should be kept in a proper database or it could be stored directly inside the audio files in a separate chunk.

It has to be noted that the loops in the loop collection come from two different sources, as figure 1 clearly shows. While the loops that were extracted from the music collection had no metadata attached to them, the content from the loop library contained manually annotated tags. These consist of the tempo, the root key, the instrument category, and the musical style.

2.3. GUI and Playback Management

As mentioned before, the GUI and the playback management component of our system have been realized using the TCL/TK scripting language. TCL/TK allows for a very quick and easy implementation of various GUI functionalities, but has rather limited capacities in terms of data processing when compared to higher level languages. Thanks to the SNACK package however it is very straightforward to realize audio playback in TCL/TK. We utilized version 8.4 of the freely available ActiveTcl distribution1.

Our GUI (figure 2) is divided into two main parts, a frame with various control elements on the top and a display area below, where the loop collection elements are visualized in form of coloured circles. The main user interaction happens with the mouse inside the display area, as we will describe later in section 2.3.2.

2.3.1. Collection Visualization

The GUI control frame contains several buttons and sliders for the user to trigger basic functionalities but most importantly, to adjust the way the loop collection is visualized. When starting the application the user is first required to load a loop collection by selecting a collection file as described in section 2.2. The

1 http://www.activestate.com/Products/activetcl/
application then parses this file line by line. It creates a
SNACK sound object with a unique ID for each line and
links it to the audio file on the hard disk. This has the
advantage of low memory usage even when dealing
with large collections of more than thousand loops. The
numerical descriptor values are stored in a descriptor
matrix (list of lists). In order to normalize the data for
visualization we also check the minimum and maximum
value for each descriptor during parsing. We create a
graphical circle object with a unique tag for each line.
Any non-numeric descriptor, like the instrument class, is
stored in form of a tag for the corresponding circle
object.

Figure 2. GUI screen shot. The display shows a zoom on
a collection sub-set tagged as “Guitar/Plucked”.

Once a collection is loaded the user can specify how
it should be visualized. For this purpose he can select
any of the available numerical descriptors to control any
of the four visualization parameters for each object:
size, colour, horizontal and vertical position. The
descriptor data is shifted and scaled according to the
minimum and maximum values that appear in the
collection, so all elements are always guaranteed to fit
inside the display window.

The distribution of descriptor values is almost never
flat on the entire range but rather often severely skewed.
This causes the circle objects to cluster together in some
areas on the screen while other areas are only sparsely
populated. To compensate this effect the user can apply
transformations to individual descriptors. Our
implementation currently supports a logarithm and a
square function.

The GUI also has a filtering function. The user can
select any available value from the non-numeric
descriptors in a drop down menu button. This hides all
items which do not have the selected value as a tag.
Especially with large collections, this helps in getting a
better overview. The current implementation only
makes use of the instrument category descriptions here,
but it is straightforward to use other non-numeric
descriptors or to combine several of them.

2.3.2. Interacting with the GUI

The user can interact with each circle object in the
display area by using the mouse. By hovering the mouse
pointer over a circle object the corresponding wav-file’s
name will pop up in a balloon help style. Double-
clicking anywhere on the screen area will zoom in and
centre the view on the clicked point. The control frame
contains a button to zoom out to the full view again.

A left-click on a circle object starts or stops playback
of the corresponding loop. The GUI operates in two
different playback modes: browsing and loop mixing. In
browsing mode, loops are played in their original tempo
and without repetition upon selection by the user. This
mode is useful for a basic orientation on how the loops
are organized on the screen. It is also suitable when
looking for a particular item in the collection. In the
loop mixing mode, the user sets a master tempo with the
BPM slider. Based on this tempo setting, a beat clock is
triggered that serves as the reference for loop playback.
Upon selection a loop will start infinite playback, beat
by beat along with the clock. In order to adjust the
length of the original beats to the clock beats, we play
them back at an adjusted sampling rate. This resembles
the effect that would be achieved by adjusting the speed
of a record player to reach a target tempo for the
recorded music upon playback.

Multiple loops can be selected for playback at the
same time. They will automatically be synchronized on
the beat no matter when exactly the user clicks on a
circle to start a loop. By clicking the synchronization
button in the control frame, the user can trigger all
currently playing loops to jump to their first beat.

In either of the two modes the user can activate
dragging. This allows him to move the circle objects in
the display area with a drag and drop operation.
Dragging functionality can serve several goals. In the
first place it is an easy way to deal with occlusions, but
it also allows the user to sort and arrange the items
according to his preference. Finally, this functionality
could be used as an annotation tool to modify the results
of algorithmic feature computation according to
subjective impression.

In our implementation we assume a 3-button mouse.
Using the middle or right button on a circle object
doubles or halves the tempo of the corresponding loop.
This functionality can either be used as an effect or to
correct octave errors that might occur in the tempo
detection process. The system is designed in a way that
allows playback to continue (particularly in loop mix
mode) no matter what changes the user makes to
visualization settings, filtering, dragging, or
modification of tempo.
3. DISCUSSION

To the authors’ knowledge the presented system is the first of its kind addressing loop collections in particular. There have been quite a few publications on GUIs for browsing and exploring collections of entire music tracks ([4] is just one example). Usually these systems follow a slightly different approach for the visualization. Rather than mapping individual descriptors directly to parameters of the plot, it is more common to apply a clustering algorithm to the data instances first. A popular method is to train a self-organizing map (SOM) on the descriptor data of the entire collection. When the map is visualized the distances between items in terms of their descriptor values are to some extend reflected in the distances on the screen. The advantage of this method is that it can easily map high-dimensional data into a 2-dimensional representation. On the other hand, the contribution of an individual descriptor can become completely concealed in this type of visualization.

For browsing song collections, the overall similarity among items (possibly resembling the notion of musical genre) is probably a more useful property than concrete values of individual low-level descriptors. In other contexts, however, the situation might be different. The system presented in [1], for example, uses a visualization very similar to ours with the goal to facilitate audio collage creation. For the case of loop collection browsing we imagine that the user might rather want to have specific properties of the items displayed at a glance instead of seeing whether two items are very similar or very different in their overall properties. It might for example be interesting to find an item with a very noisy sound characteristic for mixing it with a clean, tonal loop. Or one might look for a loop with the energy concentrated in the bass region in order to add some “punch” to a mix.

The number of descriptors that can be made available for visualization is not limited to the low-level features defined in MPEG-7. Quite a few other interesting candidates have been and are still developed in the MIR community. It is however a remaining problem that only some of those possess a clear correspondent in perceptual sound impression, while others are rather unintuitive and therefore less useful for direct visualization. Dedicated user tests might reveal a reduced set of relevant and intuitive descriptors for the target application.

Our system can currently visualize descriptors with up to four dimensions only. However, when playing with the system we found it already hard to keep these four attributes in mind simultaneously. It was easier for us to explore the collection by focusing on the spatial location of the objects. We used color only occasionally as a secondary criterion and mostly ignored the size. Since the assigned descriptors can be freely changed by the user while exploring the collection we consider the limitation in dimensions not very problematic. We therefore abstained from the use of dimension reduction techniques like principal component analysis (PCA), because they cannot guarantee an intuitive mapping.

4. FUTURE WORK

The loop explorer system we demonstrate here is still in an early stage of development. It has not been evaluated in an objective user study. While we believe that it is already fun to use and serves the purpose of facilitating the exploration process, there is a list of features that could further improve its usefulness.

Especially when dealing with large collections, we can often observe occlusions in the visualization. While zooming and dragging are the current remedies, more elaborate mechanisms like “liquid browsing” [8] could be implemented. Another very noticeable shortcoming of the current system is the absence of tempo independent pitch shifting, which makes it difficult to mix loops with strong tonal content, since they will often cause unpleasant dissonances. A real-time application based on TCL/TK and SNACK is reaching its limits here. However, a high-level implementation, possibly integrated into a full-featured loop mixing software, would allow more advanced processing. The user could set a master key along with the master tempo and all playing loops could be adjusted in pitch accordingly on the fly.

5. REFERENCES


