GUIDAGE: A FAST AUDIO QUERY GUIDED ASSEMBLAGE

Arshia Cont
UCSD, Music Dept. and
Ircam - UMR CNRS STMS.
cont@ircam.fr

Shlomo Dubnov
UCSD, Music Department
La Jolla, CA.
sdubnov@ucsd.edu

Gérard Assayag
Ircam - UMR CNRS STMS
Paris, France.
assayag@ircam.fr

ABSTRACT

In this article, a method is proposed for fast and automatic retrieval of factors of audio content in a large audio database based on user's audio query. The proposed method, unlike most existing systems, takes explicit considerations of temporal morphology of audio content. This work touches upon several existing approaches and technologies for sound manipulations, such as sound texture synthesis, music and audio mosaicing on the synthesis side, and audio matching, query by audio and audio structure discovery on the analysis side. Destined for creative applications, the proposed method is modular by allowing interactive choice of search criteria. The analysis side of the proposed model features a new audio structure discovery algorithm called Audio Oracle that describes the temporal morphologies of the underlying sound as a compact state-space model. The search engine, and the main focus of this paper, features a fast and novel algorithm based on dynamic programming called *Guidage* that is capable of reassembling the query audio by concatenating subclips of target audio files. Demonstrated results suggest a degree of semantic-driven control for query guided applications. The article concludes with examples of two immediate applications of audio matching using Guidage on music, speech and natural sounds and a discussion on further development and use of such methods in interactive and creative environments.

1. INTRODUCTION

In this paper we describe a unique combination of two main technologies - concatenative texture synthesis and audio query - used together in order to achieve audio content-based control over sound synthesis. The system uses a sound source (a query) to create new content from a variety of sounds over a large database of target sounds. The unique property of the system is its capability to match variable length clips of the query input to new audio materials, which are used for recombination and concatenation from the target database. Matching of the query sound clips is done over a compact state-space representation of the target audio, which models the audio structure by taking into account inter-similarities within each audio. This state-space representation is learned using Audio Oracle algorithm [1] and can be computed on-line or accessed as metadata. The search algorithm proposed here is a fast dynamic programming algorithm called *Guidage* specifically designed for Audio Oracle structures that is capable of matching the query by concatenating factors or subclips in the target audio. The resulting audio material preserves naturalness and captures temporal morphologies of the input query due to its capability to find longer phrases and flexibility of the matching criteria, as explained hereafter.

In this work we describe two fast audio content retrieval applications that work in the following manner: Given a database of target sounds, the user / composer / producer enters an audio signal whose properties he would like to replicate by finding clips in the target sounds that match to different user-defined features. The system automatically identifies segments in the target sounds that match parts of the query. Segments in the query that do not pass a user-defined matching threshold in the target sound are left silent, resulting in a collection of possible excerpts from the target database, matching to different portions of the query that are obtainable using a simple resynthesis algorithm. In other words, the search algorithm *guides* the resynthesis engine to parts of the target audio pertaining to the query audio and user-defined search parameters, and allows *reassembly* of these factors to replicate the given query. Using this search procedure, we define an immediate application: Query guided mixing and editing.

Results and the degree of the control of the synthesis engine provided by the algorithm and control interfaces suggest some degree of semantic-driven control over large corpuses of audio guided by user's audio query without any explicit segmentation or transcription of audio material. To this extent, the proposed schema is different from similar systems by its explicit consideration of temporal morphologies during audio structure discovery and taking them into account during search operations.

The work presented here touches upon several existing techniques for sound manipulations: sound texture synthesis, and audio mosaicing on the synthesis side; and audio matching, query by audio and audio structure discovery on the analysis side. We start the paper in section 2 by reviewing the most relevant works in the literature in these fields and their comparison to the proposed methods. Section 3 provides the general framework of the proposed algorithm, followed by a discussion and definition of the meta-data used by the algorithm in section 4 as today's large-scale audio data necessitates. One of the main reasons for the success of our system is the explicit description of temporal morphologies of the audio structure as meta-data using Audio Oracle algorithm which is also presented in section 4. In section 5 we define the proposed search algorithm *Guidage* that uses Audio Oracle structures to find occurences or partial-occurences of a given audio query in search target(s). The resynthesis engine is briefly described in section 6. Two application frameworks are then introduced in section 7 followed by examples and discussions on further research and development along the proposed algorithms.

2. RELATED WORKS

2.1. Concatenative Approaches to Sound Synthesis

On the synthesis part, different systems undergo various definitions and approaches to treatment of sound recording using re-assemblage of the original audio or an external audio corpus. Mosaicing usually refers to combination of larger chunks of sound in a "creative" manner", drawing upon re-mix cultures and other ways of composing by assembling samples [2, 3]. Both of the above might have different levels of control over the generative process, from uncontrolled manner of creating texture variants, to constrained selection designed to match a compositional design. Concatenative synthesis usually refers to the use of recordings for audio synthesis driven by non-audio data, such as note sequences (MIDI) for music or text in the case of speech [4]. This method mostly deals with finding the right recorded sounds in order to create a required pitch or phonetic sequence, trying to obtain an optimal tradeoff between the length of recopied clips and the amount of editing operations (pitch shifting, time stretching and etc.). Creative applications of concatenative synthesis usually undergo feature matching on local or short-term time scales [5]. Other creative applications include granular synthesis, and iterated non-linear functions, to mention a few.

One of the main drawbacks of most of the creative applications proposed using these methods is the lack of intuitive control over the produced contents. Some of the existing systems [4, 3] provide parametric control interfaces and in the case of the second, an audio content retrieval scheme for accessing target specific contents. In this work we present a new and fast way of controlling audio assemblage by explicitly considering the audio structure and its temporal morphologies into account with aid of a fast audio structure discovery module. The search algorithm then takes into account this structure and guides the synthesis engine towards sequences within the search target that can replicate the given structure as query. Moreover, the level of match or precision versus tolerance in similiarity between the query source and the candidate targets is controlled by a threshold parameter.

2.2. Audio Matching and Structure Discovery

On the matching and analysis part, there had been several approaches to finding similarity between sound textures. Because of the complexity and the random nature of sound textures, the matching methods usually rely on different features than those used for matching of speech or musical instrument sounds. For a high-quality audio content retrieval system, it is essential to consider temporal morphologies and the spectral dynamics of the audio into account. This is where audio structure discovery becomes important. Audio structure learning relies heavily in most present implementations on signal segmentation and can be viewed from two main perspectives: *model-based* approaches that incorporate certain musical or cognitive knowledge into the system in order to obtain structure boundaries, and *model-free* approaches where the structure is learned directly from the audio itself without any incorporation of a priori knowledge of musical structures [6]. For this work, we are interested in a model-free approach where the user can specify (or not) which aspect of the audio information she needs for her inquiry. Among various models that have been proposed, we focus our attention on how the temporal structure of an audio stream is derived given a set of audio features, whether it be a simple similarity matrix [7] or a combination of different features [8, 9]. Chai [10] has proposed *dynamic programming* to perform music pattern matching for finding repetitions in music and later discovering the structure of music. The dynamic programming scheme provides a score matrix that uses a normalized Euclidean distance between two multi-dimensional feature vectors. In computing the score matrix, the author uses a finite-window over time. Actual matching alignment occurs by backtracking over the score matrix and repetitions can be detected by finding local minima of trackback. In another approach, Peeters et al. [8] use Hidden Markov Models (HMM) and a multi-pass approach combining segmentation and structure discovery together. In each pass, different time-order similarity measures are run over audio feature vectors where the results estimate the number of classes and states for a K-means clustering that provides early parameters for learning of an ergodic HMM using Baum-Welch algorithm. For audio matching and query-by-example, perhaps the most relevant example is the *Freesound* project [1] where the focuses of the project are short samples (with mean duration of 3.25 seconds). Within the *Freesound* project there is a content-based search capability that describes the microsound structure using audio feature vectors using a Nearest Neighbor search. On top of the audio structure match is an ontology based on an English lexical model that accesses labels associated with each sample and enhances semantical capabilities of the results [11].

A more relevant framework to our work is the system described by Casey in [12]. In the described system, spectral dynamics are modeled using a 40-state hidden Markov model with parameters inferred by machine learning over a large corpus of audio training data. Time dependence between audio features is modeled by a first-order Markov chain process where each state is considered as a generator for feature vectors through a multi-dimensional Gaus-

[1] `http://www.freesound.org`

sian kernel. In Casey's terminology, these states are called *acoustic lexemes*. Matching can be achieved using the learned HMM and N-gram models to achieve longer sequences where the author considers N-gram models ranging from 1 to 8 states for the specific applications described in [12].

The work presented in this paper features a novel and fast audio structure discovery algorithm. Besides its fast operation, it provides a compact state-space model over time that would not have the finite time-window limitation met by most systems (fixing history window, limiting the number of states, etc.).

3. GENERAL FRAMEWORK

The search algorithm that is proposed in this article differs from the ones mentioned above mostly in terms of the search domain it uses during operation. In our application, this domain is a previously learned state-space representation of each audio in the database that represents the underlying *audio structure* based on a given similarity criteria. In this way, the search algorithm accesses states and transitions that are representative of the audio structure under search procedure as *meta data*. This meta data is provided using a novel algorithm called *Audio Oracle*, an automaton that is capable of learning repeating factors of audio structure in linear time and space (therefore feasible in realtime). Audio Oracle representation of audio structures can be learned either online during the search procedure or can be stored as meta data representation of audio files in a database under consideration. Audio Oracle is presented in details in [1] and is demonstrated briefly in section 4.

The *Guidage* search algorithm presented in section 5 takes an audio input as query. It is a fast dynamic programming type algorithm that browses all the audio files in a given database (or their associated metadata) to achieve highest reconstruction of the query audio by concatenating factors or subclips within each audio in the search database. More precisely, it provides paths over each Audio Oracle structure to achieve the longest possible similar structural sequence to the query audio. Moreover, when the application requires, microscopic search results are also accessible within each audio file as demonstrated in section 7.

The general schema of the *Guidage* algorithm is modular in the sense that the user can specify the search criteria that is used in both meta data construction and search procedure, and moreover a threshold parameter can control the degree of structure similarity that the user desires to assess during resynthesis. Since the Audio Oracle is independent of the input vector presented to the algorithm, the modularity amounts to the choice of the user in terms of what representation she desires to use as input to the algorithm. Using Audio Oracle structures as meta data would allow us to learn and store Audio Oracles for different sets of audio features that can be easily recalled and changed in the search user interface and utility of the system. In this paper, we have chosen to experiment with a limited set of audio features and combinations thereof which will be represented in the next section. This choice of audio feature set, is the authors' choice for the specific application domains that are presented here and in actual utility, the algorithms presented here are applicable to any set of features given by users.

4. SEARCH DOMAIN AND META DATA

4.1. Audio Oracles for Search Domain

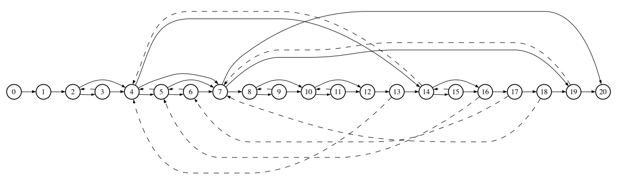
Audio Oracle is a fast automaton learning procedure that was motivated by a similar technique used for fast indexing of symbolic data such as text called *Factor Oracle* [13]. Audio Oracle is described in detail in [1]. Here, we present the description of the output and its relevance to the applications presented in this paper. Audio Oracle operates on audio feature vectors, that correspond to successive analysis frames of a given audio stream. The notion of similarity is thus determined by the type of features chosen to represent the audio signal. A sample result of Audio Oracle learning procedure is represented in figure 1 where figure 1(a) shows the learned automaton of Audio Oracle that corresponds to the waveform of figure 1(b) (two occurrences of a *Blue Jay* bird sound) where the automaton is constructed over analysis frames that correspond to MFCC features demonstrated in figure 1(c).

Interpretation of the learned Audio Oracle is straightforward. Each numbered state has a one-to-one correspondence with an audio analysis frame. There is always a forward transition from state i to state $i + 1$. Following these states and resynthesizing the audio would amount to resynthesizing the original audio. Eventually, there are other forward transitions from a state i to $j(i < j)$. These forward transitions correspond to similarity of patterns as continuations of states i and j. There are also backward transitions of suffix linkes (dashed lines in figure 1(a)) that point to a previous state that shares the *largest similar subclip* precedent to both frames.

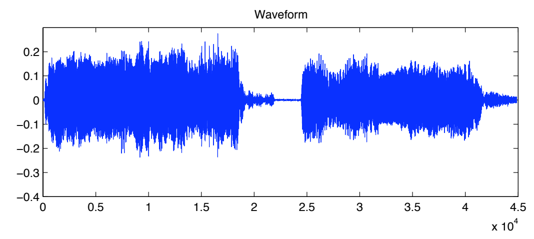
Note that Audio Oracle is a *deterministic* automaton, and is therefore described by simple transition lists where each list element $Tran(i)$ contains state indexes when there is a detected transition for state i. Moreover, each index of Audio Oracle has a one-to-one correspondence with an analysis window and its feature vector set in the audio. Thus, Audio Oracle provides a compact and efficient representation of the audio structure that immediately reduces the search domain for any search algorithm with explicit access to internal details and structures of the sound and at the same time, maintains time indexing and continuity of audio features. This is the main reason behind choosing Audio Oracle as the meta data representation used for our audio query guided search algorithm.

4.2. Audio Oracle as Meta Data

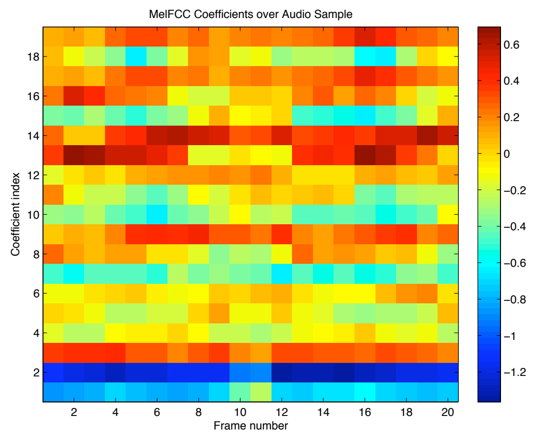
When dealing with databases of audio files and in musical application, there are several issues that are raised



(a) Learned Audio Oracle



(b) Sample Audio Waveform



(c) Mel-Frequency Cepstral Coefficients features

Figure 1. Audio Oracle Example

which we would not necessarily encounter when dealing with small sets of sound files. First of all, we would most likely be in need to access only parts of the source information for specific application and not all. In most audio applications this amounts to choosing a desired set among many audio features that are available to describe different sound properties. For example, a user that is working mostly on speech queries would be mostly in favor of Mel-Frequency Cepstral Coefficients (MFCC) and their first derivatives whereas a user that is working with music files would probably favor MFCC, pitch profiles, pitch contours, etc. or a combination thereof. Secondly, despite the increase of Random Access Memory in recent personal computers, they are still not quite sufficient when dealing with long duration audio files and large corpuses of audio. Therefore, it is desirable to store latent information needed during search operation somewhere on the hard-drive, in an interchangeable format that can be easily accessed whenever needed using a third-party application.

In our experiment, we store learned Audio Oracle structures, along with their corresponding audio features and analysis information in meta data information using the Sound Description Interchangeable Format (SDIF) [14]. Besides its universal accessibility and interchangeable format, SDIF allows fast and efficient access to any desired set of analysis frames during execution. Moreover, SDIF allows personalized type-definitions which are used here to describe Audio Oracle structures, along parameters that are used during audio re-assemblage once the search algorithm has finished the search task.

For this experiment, each audio file is analyzed and described by MFCC, Δ MFCC, pitch information, pitch contour and some combinations of these features. For each feature set, an Audio Oracle is learned and stored as a separate stream in the corresponding SDIF file. Along with this information, time-domain samples and further analysis information is stored in the SDIF file to facilitate the resynthesis procedure.

5. *GUIDAGE* ALGORITHM

We define the problem context for our proposed search algorithm as follows: Given a query audio and a search target audio file, find an assemblage of sub-clips within the target audio file that can replicate the query audio. In other words, we are aiming at reconstructing a new audio similar to the query by concatenating sub-clips of the target audio file. The search criteria is defined by the user and corresponds to the type of audio feature set (and its corresponding Audio Oracle) used for analysis and similarity comparison.

The search algorithm proposed here is based on Dynamic Programming, an algorithm paradigm in which a problem is solved by identifying a collection of subproblems and tackling them one by one, smallest first. Dynamic Programming uses the "answers" to small problems to help figure out larger ones, until the whole of them is solved. In the context of our problem, the "small" problem set amounts to finding audio chunks (or audio analysis frames in this case) in the search target audio file, that are *similar* to a corresponding chunk in the query *and* can be considered a *continuation* of the previously obtained chained based on the Audio Oracle structure of the target audio. We call this step of the procedure the *forward pass*. The "larger" problem, then, becomes finding the *best path* among all recognized that best meets the search criteria when all the small-set problems are solved. This step is referred to as *backward* procedure.

To describe the algorithm in a formal manner, we use the following notations: Query Audio is represented as $Q = \{Q_1, Q_2, \ldots, Q_N\}$ where each each Q_i is the (user-specified) feature description that corresponds to the i^{th} time-domain analysis window. Similarly, the search target audio is represented by $Y = \{Y_1, Y_2, \ldots, Y_M\}$ and also by its corresponding Audio Oracle structure represented by $\{Tran(i)| \quad i \leq M\}$ described earlier. Using these notations, algorithm 1 shows the steps taken during the forward pass of *Guidage*.

Algorithm 1 returns the structure I, which refers to found indexes on Audio Oracle structure during the forward pass and can be regarded as an analogy to the score

Algorithm 1 Forward pass of `Guidage`

```
Require: Query Audio as $Q = \{Q_1, Q_2, \ldots, Q_N\}$,
    Search Target Audio as $Y = \{Y_1, Y_2, \ldots, Y_M\}$, and
    search target Audio Oracle described by $Tran(i)$.
 1: for $i = 1$ to $N$ do
 2:   Obtain $I_i^i$ where
          $I_i^i = \left\{ j \,|\, 0 < j \leq M,\, dist(Q_i, Y_j) \leq \theta \right\}$
 3:   Assign $k = i$
 4:   while $I_i^k \neq \emptyset$ do
 5:     Increment $k$ by 1
 6:     Assign $\Gamma = \left\{ j | Tran(j) = \mu \,, \mu \in I_i^{k+1} \right\}$
 7:     Obtain $I_i^k$ where
            $I_i^k = \left\{ j \,|\, j \in \Gamma, dist(Q_k, Y_j) \leq \theta \right\}$
 8:   end while
 9: end for
10: return $I$
```

matrix in a dynamic programming paradigm. Within I, subsets $I_i = \{I_i^1, \ldots, I_i^k\}$ contain several paths associated with target Audio Oracle states and refer to the possible *reconstruction indexes* in the search target audio that can reassemble the sub-clips or factors in the query audio between indexed frames $1 \rightarrow i$, if $k = i$; or the factor between frames $k \rightarrow i$, if $k \neq i$.

Having I from algorithm 1, finding the *best paths* amounts to a backtracking procedure over subsets I_i, resulting into a *reconstruction matrix* $R : N \times \ell$. Here, N refers to the total number of analysis frames for the audio query and ℓ is the *search depth* or the longest sequence that was achieved during the search operation. Similar to I, each row k in the reconstruction matrix R corresponds to a subclip in the audio query and the contents of that row are index references to the Audio Oracle of the target audio. These indexes correspond to audio frames in the target audio that can *reassemble* the "longest possible factor" of the audio query up to frame k within a similarity margin θ provided by the user. Obviously, in the case of self-similarity for an audio query of total N analysis frames, the described algorithm results to an $N \times N$ symmetric reconstruction matrix R with each M^{th} row ($M \leq N$) in the form $\{M, M-1, \ldots, 2, 1, 0, \ldots, 0\}$.

6. RESYNTHESIS

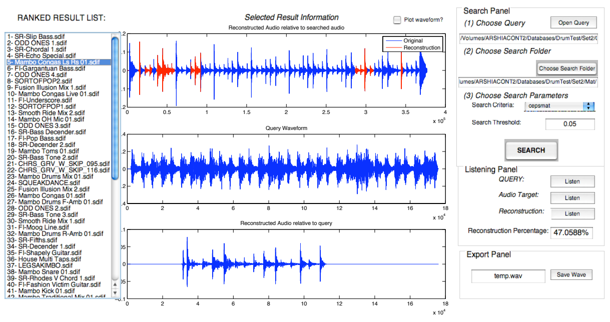
Once the reconstruction matrix R is obtained by *Guidage*, resynthesis of the search results is straightforward. Each row of R contains indexes to an audio frame window and resynthesis algorithm is dependent on the choice of representation of inputs to Audio Oracle, therefore modular. For our goal, each index refers to time-domain samples corresponding to the frame index. Thus, synthesis of the results amounts to a concatenation of the indexed time-domain samples with a windowed overlap-add algorithm to assure phase continuity. Resynthesis in the general case of Audio Oracle navigation is described in [1].

7. APPLICATIONS AND RESULTS

The *Guidage* algorithm can be easily integrated into any application that calls for the framework described earlier. For this presentation, we describe two immediate applications. In one application, an audio query is given by the user as well as a pointer to a search target folder, and some search parameters. The application then runs *Guidage* over the whole database and a Graphical User Interface (GUI) demonstrates ranked search results and visualizes different parameters. In the second application, an audio query is searched within *one* given target audio, where the aim is to reassemble various occurrences of the query in the target file and access the micro-structure level reassemblage. Hereafter, we discuss each application in depth and demonstrate results.

7.1. Query over an Audio Database

For the first application described earlier, we have implemented a GUI that both controls input file, folder and parameters by the user and also visualizes the results and exports them as desired by the user. The front interface of the GUI is demonstrated in figure 2.



1- SR-Slip Bass.sdif
2- ODD ONES 1.sdif
3- SR-Chordal 1.sdif
4- SR-Echo Special.sdif
5- Mambo Congas Ls Rs 01.sdif
6- FI-Gargantuan Bass.sdif
7- ODD ONES 4.sdif
8- SORTOFPOP2.sdif
9- Fusion Illusion Mix 1.sdif
10- Mambo Congas Live 01.sdif
11- FI-Underscore.sdif
12- SORTOFPOP1.sdif
13- Smooth Ride Mix 2.sdif
14- Mambo OH Mic 01.sdif
15- ODD ONES 3.sdif
16- SR-Bass Decender.sdif
17- FI-Pop Bass.sdif
18- SR-Decender 2.sdif
19- Mambo Toms 01.sdif
20- SR-Bass Tone 2.sdif
21- CHRS_GRV_W_SKIP_095.sdif
22- CHRS_GRV_W_SKIP_116.sdif
23- Mambo Drums Mix 01.sdif
24- SQUEAKDANCE.sdif
25- Fusion Illusion Mix 2.sdif
26- Mambo Congas 01.sdif
27- Mambo Drums F-Amb 01.sdif
28- ODD ONES 2.sdif
29- SR-Bass Tone 3.sdif
30- Smooth Ride Mix 1.sdif
31- FI-Moog Line.sdif
32- Mambo Drums R-Amb 01.sdif
33- SR-Fifths.sdif
34- SR-Decender 1.sdif
35- FI-Shapely Guitar.sdif
36- House Multi Taps.sdif
37- LEGSAKIMBO.sdif
38- Mambo Snare 01.sdif
39- SR-Rhodes V Chord 1.sdif
40- FI-Fashion Victim Guitar.sdif
41- Mambo Kick 01.sdif

(1) Choose Query
/Volumes/ARSHIACONT2/Databases/DrumTest/Set2/
(2) Choose Search Folder
lumes/ARSHIACONT2/Databases/DrumTest/Set2/Mat/
(3) Choose Search Parameters
Search Criteria: cepsmat
Search Threshold: 0.05
QUERY:
Audio Target:
Reconstruction:
Reconstruction Percentage: 47.0588%
temp.wav

Figure 2. GUI for Audio Query over an Audio Database

This GUI is designed as follow: At the right of figure 2, there are three panels that separate different functionalities of the application. The first and most important is the *search panel* where the user can freely choose the audio query file and a folder as the search target. In the next step, the user can choose the search criteria that she intends to perform the algorithm on from a pop-up menu. The list of search criteria is either (1) loaded automatically from the meta-data in case the chosen query is in SDIF format, or (2) is the list of available features to the system as described in section 4, also open to further expansions. Another parameter that can be controlled by the user is the distance threshold or θ used in algorithm 1. Pressing the *Search* button then performs the algorithm over the whole database assigned by the user. Once the results are obtained, the result-box at the left of the GUI demonstrates a ranked list of sound files in the database according to the *reconstruction percentage* with regards to the audio query. The best reconstruction for each file is thus the *longest* sequence that can be reassembled through resynthesis to achieve similar factors to the audio query. Following the definition of the reconstruction matrix in section 5, this path corresponds to one row of the matrix R and the one with non-zero elements. Choosing each file in the result-box reproduces three figures in the middle of the GUI for visualization and better browsing of the semantical contents that has been found during the procedure. The top figure shows the chosen target audio waveform where the found factors within the audio are highlighted with red. The middle figure shows the query audio waveform and the bottom one shows the concatenative synthesis of the highlighted factors in the first figure relative to the audio query. A *Listening Panel* respectively allows listening to the three described waveforms and an additional *Export Panel* allows exporting the resynthesis, if desired by the user, to an external audio file.

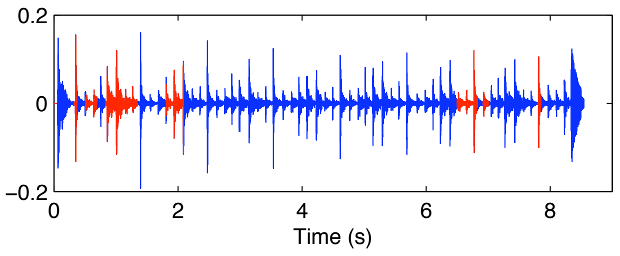
7.1.1. *Sample Results*

Here we demonstrate results on two sets of sounds and different queries. All the sounds reproduced here and more sample results on other data sets are available for audition on the internet[2]. The first set corresponds to a database of musical loops and rhythms out of realistic recording sessions taken from the *Kontakt Sound Library*[3]. The collection of loops chosen for the search session amounts to approximately $200Mb$ of disk space, 140 audio files with mean duration of 7 seconds. The query used for this demonstration is an *African drum sequence* that lasts approximately 4 seconds and non-existant in the search database. Figure 3 shows the 3^{rd} ranked results, corresponding to 47.06% reconstruction and the produced waveforms out of the GUI described earlier. The search criteria used for this query session are the MFCC feature vectors with an Euclidean distance threshold of 0.05.

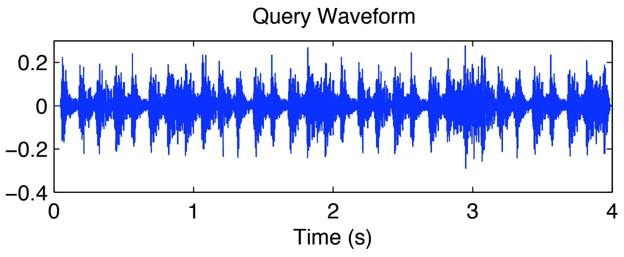
Figure 3(c) shows the resynthesized result audio as the best reassembly of the factors in the target audio (figure 3(a)) to achieve similar effects to the query waveform (figure 3(b)). As mentioned earlier, the silences in figure 3(c) correspond to factors where no match has been found. Highlighted (red) waveforms in figure 3(a) correspond to factors of the target which are used for concatenative synthesis and indexed results of *Guidage* to achieve the waveform in figure 3(c). Comparing figure 3(c) and 3(b), we can see that the algorithm has used factors corresponding to sound objects in figure 3(a) to achieve a similar rhythmic pattern as in figure 3(b). Listening to the synthesized audio (on the provided URL) also reveals the timbral similarity between the query and result - another reason why this sample has appeared among the top 10 results in a search session over 140 sound files.

[2] `http://www.cosmal.ucsd.edu/arshia/index.php?n=Main.Guidage`

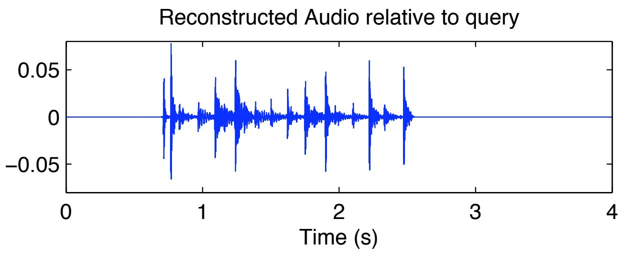
[3] `http://www.native-instruments.com/`



(a) Target Audio Waveform and Found Factors (highlighted)



(b) Query Audio Waveform

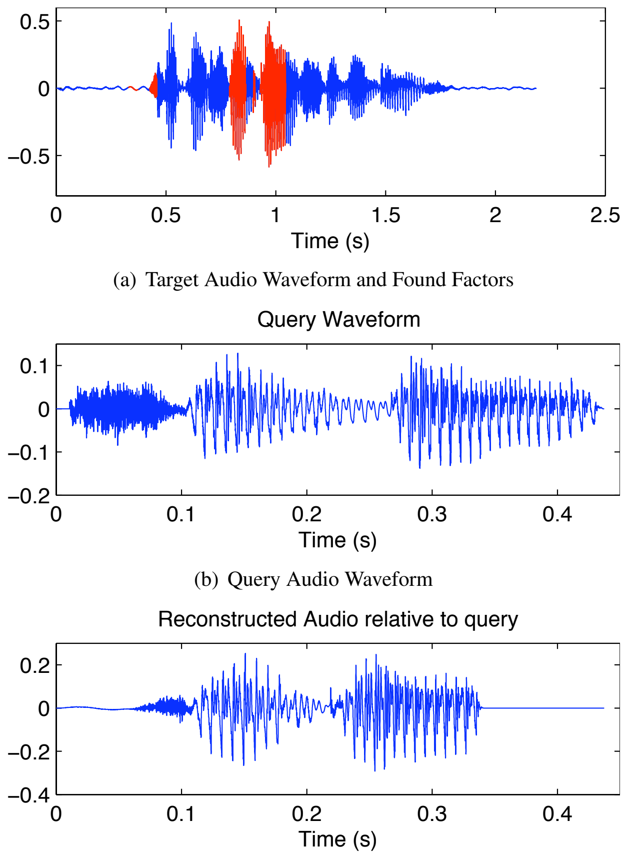


(c) Synthesized Best result relative to query

Figure 3. Music Result Sample of the Audio Query by Content

The second experiment-set corresponds to speech audio files. This experimental database corresponds to 10 recordings of theatre actors saying the same phrase in French with different durations and intonations. The repeated phrase in French is: `C'est un soldat à cheveux gris`. The query is taken out of one of the sound files and corresponds to the pronunciation of the word `Soldat` in French. The aim of the search algorithm here is therefore to find the occurrences of the word `Soldat` in different phrases despite their variances *or* to reconstruct the same word by concatenating different phonemes where linear reconstruction is not possible. The search criteria used for this query session is a mixture of MFCC feature vectors and their first derivative (ΔMFCC). This choice of audio feature is common among speech recognition systems. For this sample result, we demonstrate the 9^{th} ranked result among 10, thus towards the worst, in figure 4. The interest in showing “worse” results is to demonstrate the behavior of the re-assemblage.

Similar to figure 3, the three subfigures correspond to the query waveform of the word query `Soldat` (figure 4(b)), the target phrase recording and the used factors during resynthesis (figure 4(a)) and the resynthesized result in figure 4(c). The remarkable result here is that since the intonation and duration of the target audio is quite different from the original source, the algorithm has reinforced



(a) Target Audio Waveform and Found Factors

(b) Query Audio Waveform

(c) Synthesized Best result relative to query

Figure 4. Speech Result Sample of the Audio Query by Content

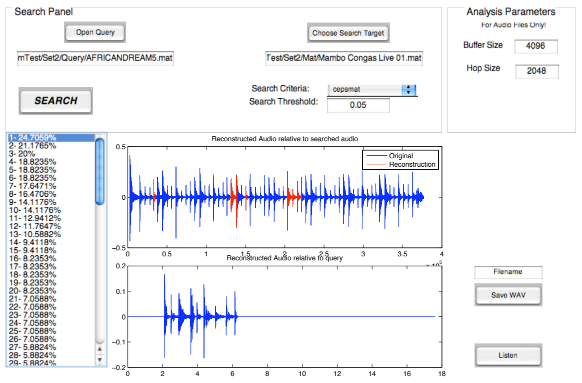
the reconstruction of the query using different phoneme sequences than the original. More precisely, factors used during reconstruction (highlighted waveforms in figure 4(a)) correspond to the `/s/` sound in the pronunciation of the word `C'est` (in French) in the beginning of the phrase (and not the `/s/` sound in `Soldat`) and partial factors of the spoken word `Soldat` as seen in the sub-figure to achieve reconstruction.

7.2. Query within Micro Audio Structure

The application described in the previous section uses only one result path among all that is found by the algorithm described in section 5. Recall that the reconstruction matrix R has N rows where each can contain a candidate path. In some applications, and especially musical ones, users might be interested to focus on the microscopic search results rather than the longest path result that is represented in section 7.1. In this section, we demonstrate this aspect of the *Guidage* algorithm where we focus on one audio query and *one* (eventually long) search target, and visualize *all* the results obtained by the *Guidage* algorithm. Once again, the user has the ability to control search and audio analysis parameters to a fine-scale control, allowing access to query guided grains in the target audio.

Figure 5 and 6 show two snap shots of the application run on the same query/target pair, but showing the first and second results respectively. Here again, as in section 7.1, a *Search panel* allows the user to choose the audio query and target files as well as search parameters and analysis parameters (in case an audio file is chosen instead of pre-formated SDIF meta-data). Clicking on the *Search* button lists all the results in the *result-box* on the left and choosing each result (represented by the reconstruction percentage), visualizes the results in two corresponding figures in the GUI. Here, the top figure corresponds to the target audio waveform, again, where highlighted waveforms correspond to the found factors for the chosen result set. The lower figure visualizes the resynthesized waveform relative to the query waveform (not shown here). The query used for this demonstration is the same *African drum* sequence used in section 7.1.1 and the search target audio is a *Live Mambo Congas* audio sequence.

Figure 5 shows the *first* result that corresponds to the highest reconstruction percentage (24.7% here). This is the typical "best" result used in the audio query over database application. As before, the resynthesized audio has the same timeline as the query; showing that in figure 5, the reassembled sequence corresponds to the original query audio that appears approximately between samples $40K - 60K$ or times $0.9 - 1.4$ seconds.



mTest/Set2/Query/AFRICANDREAM5.mat

Test/Set2/Mat/Mambo Congas Live 01.mat

Search Criteria: cepsmat

Search Threshold: 0.05

Buffer Size 4096

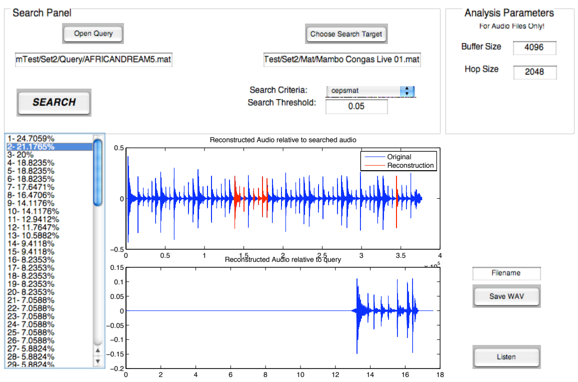
Hop Size 2048

1- 24.7059%
2- 21.1765%
3- 20%
4- 18.8235%
5- 18.8235%
6- 18.8235%
7- 17.6471%
8- 16.4706%
9- 14.1176%
10- 14.1176%
11- 12.9412%
12- 11.7647%
13- 10.5882%
14- 9.4118%
15- 9.4118%
16- 8.2353%
17- 8.2353%
18- 8.2353%
19- 8.2353%
20- 8.2353%
21- 7.0588%
22- 7.0588%
23- 7.0588%
24- 7.0588%
25- 7.0588%
26- 7.0588%
27- 5.8824%
28- 5.8824%
29- 5.8824%

Figure 5. GUI for Micro Audio Query

Figure 6 shows the 2nd result in the list chosen by the user. This result corresponds to 21.1% reconstruction. What is remarkable here is that comparing both visualized subfigures in figures 5 and 6, it is clear that (1) the factors used during re-assemblage and results of *Guidage* are different. And (2) the resynthesized sequences correspond to two different timelines of the original audio query (in figure 6 between $3 - 4$ seconds.

In contrary to the query over database application presented earlier, a simple access to all results as demonstrated in this section, gives more control over microscopic content structure of target audio guided by an audio query. As before, implemented GUI facilitates export of audio material for further mixing and musical use.


Search Panel

mTest/Set2/Query/AFRICANDREAM5.mat

Test/Set2/Mat/Mambo Congas Live 01.mat

Search Criteria: cepsmat

Search Threshold: 0.05

Analysis Parameters

For Audio Files Only!

Buffer Size 4096

Hop Size 2048

1- 24.7059%
2- 21.1765%
3- 20%
4- 18.8235%
5- 18.8235%
6- 18.8235%
7- 17.6471%
8- 16.4706%
9- 14.1176%
10- 14.1176%
11- 12.9412%
12- 11.7647%
13- 10.5882%
14- 9.4118%
15- 9.4118%
16- 8.2353%
17- 8.2353%
18- 8.2353%
19- 8.2353%
20- 8.2353%
21- 7.0588%
22- 7.0588%
23- 7.0588%
24- 7.0588%
25- 7.0588%
26- 7.0588%
27- 5.8824%
28- 5.8824%
29- 5.8824%

Open Query
Choose Search Target
SEARCH
Reconstructed Audio relative to searched audio
Original
Reconstruction
Reconstructed Audio relative to query
Filename
Save WAV
Listen

Figure 6. GUI for Micro Audio Query

8. DISCUSSIONS

We introduced the concept of *Guidage* as query guided audio assemblage and provided an algorithm for fast discovery of audio structure, fast retrieval audio-content, and resynthesis of retrieved factors as a re-assembly of the target audio. We presented two immediate applications of such concept in section 7. The results are suggestive of a semantical-level control of sound objects for musical applications. For example, demonstrated results in section 7.1.1 suggest that the factors that are found by *Guidage* correspond to semantically independent sound objects with clear boundaries which are then re-assembled to replicate the audio query. Unlike similar systems, we do not perform any kind of explicit segmentation or transcription of audio sources to achieve this level of semantic control. We believe that this is due to an explicit consideration of temporal morphologies of audio structures during structure discovery and guided search operation.

With this introduction, the presented concept should be further explored for interactive and creative computer music environments. The simplicity of both Audio Oracle and Guidage algorithms and their fast performance also make them strong candidates for high-level control modules in interactive computer music environments. Speed performance of *Guidage* depends on the degree of semantic similarity between the query and search targets. For the experiment described in section 7.1.1 on a corpus of 140 audio files with total size of $200Mb$, *Guidage* performs in approximately 80 seconds using MATLAB and a $2.3Ghz$ Mac-Intel machine. In another audio query over database experiment on natural bird sounds with 244 audio files and over $600Mb$ of data, *Guidage* performs in less than 80 seconds. This suggests some degree of *scalability* of the proposed algorithm when dealing with large corpuses of audio and makes it even more attractive for high-level interactive control of computer music processes. We are currently pursuing further research and developments along these lines.

All the examples and sounds presented in this paper, and further experiments on other data sets, are available for audition and further examination on the internet [4].

Guidage is currently under development and the progress and eventual release of *Guidage* can be tracked from the same URL.

9. REFERENCES

[1] Shlomo Dubnov, Gerard Assayag, and Arshia Cont. Audio oracle: A new algorithm for fast learning of audio structures. In *Proceedings of International Computer Music Conference (ICMC)*. Copenhagen, September 2007.

[2] Aymeric Zils and François Pachet. Musical Mosaicing. In *Proceedings of Digital Audio Effects (DAFx)*, Limerick, Ireland, Decembre 2001.

[3] Ari Lazier and Perry Cook. MOSIEVIUS: Feature driven interactive audio mosaicing. In *Proceeisng of DAFx*, pages 312–317, London, UK, September 2003.

[4] Diemo Schwarz. Corpus-based concatenative synthesis. *IEEE Signal Processing Magazine*, 24(1):92–104, March 2007.

[5] Bob L. Sturm. MATConcat: An Application for Exploring Concatenative Sound Synthesis Using MATLAB. In *Proceedings of Digital Audio Effects (DAFx)*, Naples, Italy, October 2004.

[6] Bee Suan Ong. *Structural Analysis and Segmentation of Music Signals*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, February 2007.

[7] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *IEEE International Conference on Multimedia and Expo (I)*, pages 452–, 2000.

[8] Geoffroy Peeters. Deriving musical structures from signal analysis for audio summary generation: “sequence” and “state” approach. In *CMMR*, volume 2771, Montpellier, France, 2004.

[9] A. Rauber, E. Pampalk, and D. Merkl. Using psychoacoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarities. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, 2002.

[10] Wei Chai. *Automated Analysis of Musical Structure*. PhD thesis, Massachusetts Institute of Technology, MA, USA, September 2005.

[11] Pedro Cano. *Content-Based Audio Search from Fingerprinting to Semantic Audio Retrieval*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, April 2007.

[12] Michael Casey. Acoustic lexemes for organizing internet audio. *Contemporary Music Review*, 24:489–508(20), Number 6/December 2005.

[13] Cyril Allauzen, Maxime Crochemore, and Mathieu Raffinot. Factor oracle: A new structure for pattern matching. In *Conference on Current Trends in Theory and Practice of Informatics*, pages 295–310, 1999.

[14] Diemo Schwarz and Matthew Wright. Extensions and Applications of the SDIF Sound Description Interchange Format. In *Proceedings of the International Computer Music Conference*, Berlin, August 2000.

[4] `http://www.cosmal.ucsd.edu/arshia/index.php?n=Main.Guidage`