SEED: RESYNTHESIZING ENVIRONMENTAL SOUNDS FROM EXAMPLES

Gilberto Bernardes
INESC TEC
gba@inesctec.pt

Luis Aly
University of Porto, Faculty of Engineering
luis.aly@fe.up.pt

Matthew E. P. Davies
INESC TEC
mdavies@inesctec.pt

ABSTRACT

In this paper we present SEED, a generative system capable of arbitrarily extending recorded environmental sounds while preserving their inherent structure. The system architecture is grounded in concepts from concatenative sound synthesis and includes three top-level modules for segmentation, analysis, and generation. An input audio signal is first temporally segmented into a collection of audio segments, which are then reduced into a dictionary of audio classes by means of an agglomerative clustering algorithm. This representation, together with a concatenation cost between audio segment boundaries, is finally used to generate sequences of audio segments with arbitrarily long duration. The system output can be varied in the generation process by the simple and yet effective parametric control over the creation of the natural, temporally coherent, and varied audio renderings of environmental sounds.

1. INTRODUCTION

The need to extend a given environmental audio sample is a recurrent problem in sound design [1, pp. 38-39]. A typical example in sound post-production for television and film is when pre-recorded audio does not cover the entire duration of a scene. The most common solution is to manually find a smooth transition point in the sample and loop it [2, pp. 178, 204]. Since these tracks typically run in the background, this solution is acceptable despite being time-consuming and recognizably repetitive. A different scenario in which the need to extend a given environmental audio sample is when the file size of the audio content is an important consideration. This concerns the memory storage and/or the time needed to download or stream media assets in applications such as videogames, installations, and screensavers. Mostly, these applications use fixed audio samples, which means that greater variation implies more storage. Guaranteeing a small memory storage footprint is critical to the success of such applications.

A possible solution is to extend the duration of a given short environmental sound recording arbitrarily. While, to the best of our knowledge, no commercial applications exist to fulfill this purpose, academic research has recently looked at this problem [3-7]. For a comprehensive review of system for the expansion of environmental sounds refer to [8, 9]. Conducted research in this topic relies on the fact that, within a particular time and geographical span, environmental sounds are i) relatively monotonic, ii) simple in structure, and iii) have a high degree of redundancy [8]. Existing systems typically follow a threefold structure that handles segmentation, analysis and the resynthesis of a given environmental sound example.

Current technological solutions are mostly confined to the extension of simple sound textures and barely address environmental sounds with complex temporal structures. Additionally, most systems extend a given input file by generating new sequences based on the similarity among audio segments in order to create smooth transitions [4, 5]. While this approach provides good results for highly redundant audio content, it does not provide an optimal answer to the problem, especially when processing environmental sound scenes with a higher degree of complexity and temporal dependencies, such as moving vehicles, and storms.

Our work strives for a solution capable of generating long audio streams from a short input audio signal example of non-diegetic sounds excluding dialogue with variable complexity by capturing its intrinsic structure. Based on representations of an input environmental sound we propose a real-time system that generates new audio sequences of arbitrary duration with dynamic control over the amount of novelty introduced. We expand on previous research with novel methods for audio segmentation and learning audio structures based on a flexible dictionary-based representation of audio classes derived from clustering methods. Furthermore, we use a concatenation cost—a concept borrowed from concatenative sound synthesis and in particular from [10]—to measure the transition between audio segments towards generating smooth audio sequence transitions.

A prototype application, named SEED (Sound Environmental ExpanDer), implements the model detailed here in Pure Data.[1] SEED is composed of three top-level modules, detailed in the following three sections, responsible for i) segmenting an environmental input audio signal (Section 2), ii) creating a temporal model of the input signal (Section 3), and iii) generating new arbitrarily-long audio streams based on the audio input structure (Section 4). The last three sections present SEED's graphical user interface (Section 5), an evaluation of the system (Section 6), conclusions, and directions for future work (Section 7).

[1] https://puredata.info/, last access on January 2016.

2. AUDIO INPUT SEGMENTATION

In SEED, we adopt an audio segmentation strategy, which isolates events in time with clear spectral differences. Inspired by the work of Hoskinson and Pai [4], the boundaries of each segment correspond to stable moments (lowest spectral difference between audio analysis frames), aiming to favor smoother transitions between resynthesized audio segments during generation.

In greater detail, we first compute the spectral flux function $SF(m)$ of non-normalized audio frames m (window size ≈ 46.4 ms and window overlap of 50%) using the timbreID library [11]. Then, in order to minimize spurious detections, we smooth the spectral flux function $SF(m)$ by a bidirectional (or zero-phase shift) first-order infinite impulse response low-pass filter $\widehat{SF}(m)$ with cutoff frequency of 5 Hz.

The third step of the algorithmic chain is a valley-picking algorithm that defines segment boundaries. Valleys are computed by finding all local (or relative) minima on the filtered spectral flux function $\widehat{SF}(m)$ below a dynamic threshold value $t(m)$. The threshold $t(m)$ aims to regulate magnitude changes across the temporal dimension of the filtered spectral flux function $\widehat{SF}(m)$. It is computed as the difference between the local median $\mu(m)$ and the local standard deviation $\sigma(m)$ of a window of size 32 analysis frames around m in the filtered spectral flux function $\widehat{SF}(m)$ such that:

$$t(m) = \mu(m) - \alpha \cdot \sigma(m) \quad (1)$$

where α bias the relative weight of the standard deviation and is set to $\alpha = 0.5$.

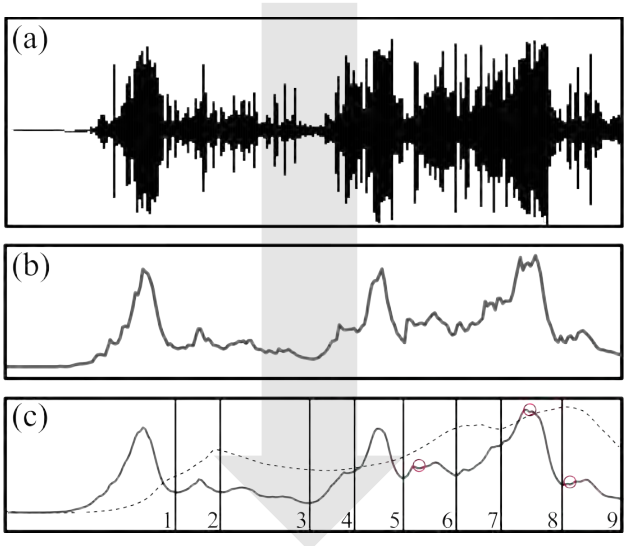

Figure 1. Illustration of the audio input segmentation in three stages: (a) waveform, (b) spectral flux function $SF(m)$, and (c) filtered spectral flux function $\widehat{SF}(m)$ on which we apply a valley picking algorithm which defines segment boundaries—circles indicate valleys discarded by $t(m)$, represented as a dashed line, and the local proximity constraint.

We additionally discard valleys that are fewer than four analysis frames apart to guarantee segments with a minimum duration of 185.8 ms wherein we apply a cross fade during generation. In Figure 1, from the three-circled valleys, the middle was excluded because it was greater than $t(m)$, and the two others were excluded because their position was fewer than four analysis frames apart from a previously detected valley.

Finally, in order to guarantee a minimum number of segments in case the input audio signal is purely composed of highly stationary sounds (or highly stable spectra), the system arbitrarily segments the input audio signal until a ratio r of the total duration of the signal (in seconds) to the total number of segments S reaches a number below unity.

3. LEARNING OPTIMAL TRANSITIONS

In view of our goal to generate environmental audio that preserves the inherent structure of an input signal, we create two representations of its structure. The first is detailed in Section 3.1 and consists of a table encoding transitions between *classes* representing the input audio segments. Prior to the table creation, segments are represented by a set of features extracted from the signal content with higher degree of variability (which due to our segmentation strategy is most likely to be in the middle region of a segment) and then clustered into classes. The second is detailed in Section 3.2 and consists of matrix encoding the concatenation cost among all possible segment transitions computed by comparing features from the troughs of the audio input segments.

3.1 Encoding the Audio Input Temporal Dynamics

Transition tables are a commonly applied strategy to encode finite and discrete high-level symbolic music patterns for style imitation [12]. They are easy to implement, computationally efficient and, when modeling structures with enough redundancy, have proven to be powerful in generating similar musical sequences to the structure they represent, while ensuring variation [12].

When derived from audio, transition tables are particularly difficult to compute, due to the low-level and noisy representation of the signal. The biggest challenge is to parameterize the audio using a finite and discrete space, while capturing its multidimensional attributes. To this end, we propose a two-step parameterization process, detailed in Sections 3.3.1 and 3.3.2, which groups clusters (pre-segmented) audio events into classes. The generated audio classes are then used to compute a transition table, which provides the basis for the generative process.

3.1.1 Parameterizing the Audio Signal

The first audio parameterization step represents each audio segment by the following collection of five audio features: spectral brightness, spectral flatness, zero crossing rate, spectral spread, and amplitude. These features are among the audio descriptors from Brent's timbreID library [11] chosen on the basis of their relevancy to describe the spectro-temporal dimension of environmental sound sources [13, 14].

For each segment, we first extract the abovementioned audio features on an overlapping window basis (window size ≈ 11.6 ms, and window overlap of 50%). Then, for each segment, we compute first and second order statistics (i.e., min, max, mean and variance). A feature vector of 20 (5 x 4) elements per audio segment is finally stored in a database.

To enhance the audio descriptions, we adopt an automatic strategy to weight each descriptor according to its variance across the entire input audio signal. This strategy, first proposed in [11] and explored in the realm of audio generative contexts in [15], assumes that features with higher variance across an audio example are more relevant, because their high variance provides a more distinctive characterization of the segments. Prior to the weight's assignment, we normalize each descriptor to the 0-1 range by their minimum and maximum range values.

3.1.2 A Dictionary of Audio Events

As discussed in Section 3.1, a major difficulty to represent audio segments in transition tables is to reduce them to a finite and discrete set of representations that capture their multidimensional audio content attributes. In Section 3.1.1, we already showed a strategy to represent an audio segment as a 20-element feature vector. Now, we perform additional data reduction by creating a dictionary of sound classes that represent the sound segments by a unique value.

Segment class creation is performed by an agglomerative hierarchical clustering algorithm [16], a method inspired by the work of Saint-Arnaud and Popat [17] on sound texture analysis and synthesis, which we extend by proposing several clustering solutions with variable numbers of elements (or audio classes) for a given audio input source. Our aim behind the choice of this particular algorithm was to allow a user to drive or adapt the dictionary construction intuitively by choosing the number of audio classes it includes. In doing so, we additionally avoid the formalization of many subjective and contextual factors inherent to the task (please refer to [15, pp. 74-79] for a broader discussion on this topic).

In agglomerative hierarchical clustering, each of the S input audio segments starts as its own cluster, and iteratively the algorithm pairs them until it reaches a configuration where all S segments belong to one cluster. In order to decide which audio segments are paired at each iteration, a similarity metric and a linkage criterion—which specifies how pairwise distances involving clusters with more than one segment are computed—is required. In our work, the Euclidean distance between audio segment feature vectors expresses similarity (small distances correspond to similar segments and large distance to dissimilar ones). Clustered segments are represented by the mean vector values of their constituent segment feature vectors. Consequently, inter-cluster distances are computed as the Euclidean distance between two such mean vectors—referred to as the centroid method [16].

Hierarchical clustering is commonly illustrated by a dendrogram, i.e. a tree structure that displays distances amongst input data elements and clusters (see Figure 2). Horizontal lines connect the most similar elements at a given iteration, thus forming a "new" element. The distance of a particular pair of segments or segment clusters is reflected in the height of the horizontal line.

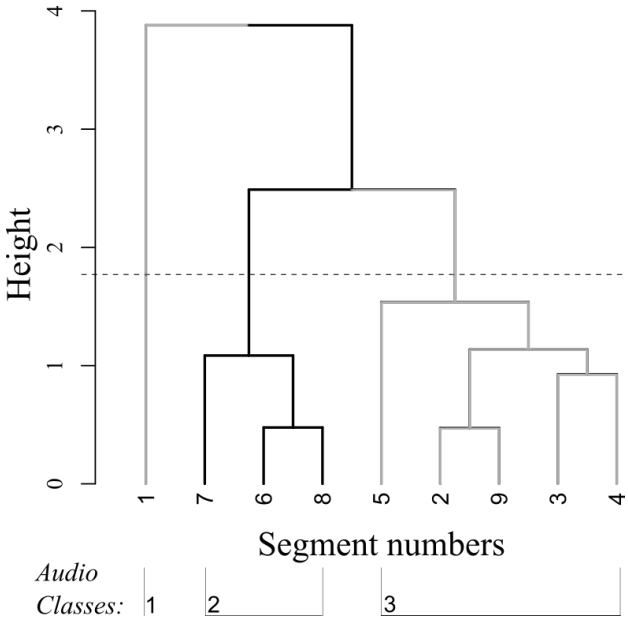

Figure 2. Dendrogram illustrating the hierarchical clustering of the audio segments shown in Figure 1. The dashed line shows the cutting point that instantiates three clusters expressed by alternating colors.

Table 1 shows all possible cluster configurations from the audio segments shown in Figure 1 as illustrated in the dendrogram in Figure 2. All possible numbers of clusters from nine clusters to a single cluster are illustrated. A hierarchy from the resulting set of clusters needs to be selected by user. This is equivalent to cutting the dendrogram at a particular height. The cutting point in Figure 2 corresponds to the configuration with three clusters. For a given solution, we then assign labels to each cluster using an increasing sequence of natural numbers to build a dictionary of audio classes (shown in Figure 2 and in the third column of Table 1 for all cluster hierarchies).

The higher the hierarchy of the agglomerative clustering algorithm, the greater redundancy in the creation of the audio classes dictionary.

Number of clusters	Clusters	Audio classes
9	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9
8	1, {2, 9}, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 6, 7, 8
7	1, {2, 9}, 3, 4, 5, {6, 8}, 7	1, 2, 3, 4, 5, 6, 7
6	1, {2, 9}, {3, 4}, 5, {6, 8}, 7	1, 2, 3, 4, 5, 6
5	1, {2, 9}, {3, 4}, 5, {6, 7, 8}	1, 2, 3, 4, 5
4	1, {2, 3, 4, 9}, 5, {6, 7, 8}	1, 2, 3, 4
3	**1, {2, 3, 4, 5, 9}, {6, 7, 8}**	**1, 2, 3**
2	1, {2, 3, 4, 5, 6, 7, 8, 9}	1, 2
1	{1, 2, 3, 4, 5, 6, 7, 8, 9}	1

Table 1. Cluster configurations from the audio segments shown in Figure 1 as illustrated in the dendrogram in Figure 2. Sets within brackets denote clustered audio segments.

3.1.3 Audio Classes Transition Table

To encode the structure of the input audio signal, we finally create a table representing its component audio classes transitions. We first create a string of numbers representing the temporal dimension of the input audio signal by substituting each segment number by its representative audio class (see Figure 3). A database establishing the correspondence between audio classes and their component audio segments is then created. Following the example shown in Figure 3, our database would include the following entries: {**1**: 1}; {**2**: 2 3 4 5 9}; and {**3**: 6 7 8}. The first index element is the audio class number and the following values (after the comma) correspond to the audio segments of that class.

Finally, we build a table encoding all possible transitions between audio classes. Based on the example shown in Figure 3, for a first-order transition table, we would have the sequences shown in Figure 4.

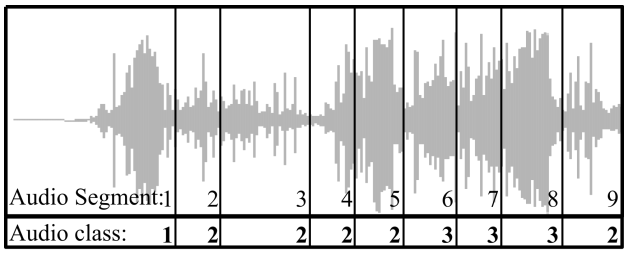

Figure 3. Correspondence between audio segments and audio classes derived from the hierarchical clustering tree shown in Figure 2.

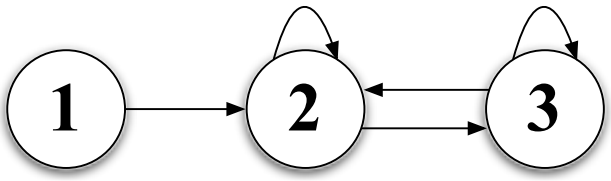

Figure 4. Visualization of all possible transitions among the audio classes from the example shown in Figure 3.

The order of the transition table is by default assigned to a third-order (chosen on the grounds of heuristic evaluations of various audio input signals with durations ranging from 30 seconds to a minute long) and can be manually changed by the user.

3.2 Audio Segments Concatenation Cost

An S x S matrix C expressing the concatenation cost between all audio segments is created to promote smooth transitions during generation. Given two segments s_1 and s_2: the cost of transitioning from frame s_2 to s_1, $C(s_1, s_2)$ is calculated as the Euclidean distance between two (24 coefficient) bark spectrum vectors l and f – where $s_{1,l}$ is the last (46.4 ms) frame of s_1 and $s_{2,f}$ is the first (46.4 ms) frame of s_2:

$$C(s_1, s_2) = \sum_{n=1}^{N=24} \left| s_{1,l}(n) - s_{2,f}(n) \right| \qquad (2)$$

Bark coefficients are used since they are shown to capture both pitch and timbral discontinuities [11].

4. GENERATION

We now detail the generation of arbitrarily long audio sequences in SEED, which rely on the transition table and concatenation cost matrix C detailed in Section 3. Generation is approached as a search problem, which aims at finding audio segment sequences which: i) exist as observable audio class series in the input audio signal, ii) have low concatenation cost, and iii) promote variation and a uniform exploration of the entire set of audio segments.

In order to satisfy the first condition we retrieve from the transition table a list of all possible audio classes that follow the last played audio classes and then unpack them into the audio segments they represent. From this collection of candidate audio segments, we then select the one with minimum concatenation cost C to the preceding segment, thus, favoring smoother transitions in the generated sequences.

Prior to the selection process we introduce a penalty in the concatenation cost of the most recently selected segments in order to prevent repetition (or promote variation). The concatenation costs C of the most recently played segment in the matrix is increased by a factor of $2 \leq \eta \leq 4$. Penalties are gradually reset to the original concatenation cost in eight segment selections by imposing a difference of $(\eta - 1)/8$ at each iteration. Selected segments are concatenated with a short cross-fade overlap of 46.4 ms.

Following the example shown in Figures 3 and 4, and assuming that we want to select an audio segment to follow the audio segment 1, we would first inspect all possible continuations for its audio class by retrieving transitions in the table illustrated in Figure 4, which would result in the single audio class **2**. Then, we would collect all audio segments linked to that audio class, which gives the set {2, 3, 4, 5, 9}. Lastly, we would select for playback the audio segment with the smallest concatenation cost from the previous audio segment 1.

5. USER INTERFACE

Figure 5 shows the graphical user interface of the SEED prototype implemented in Pure Data. On the upper part of the interface we find three elementary control settings for i) opening an input audio signal ('/'), ii) starting and pausing the generative process ('>'), and iii) controlling the overall volume of the generated output (top-right corner slider).

The lower part of the interface contains a function graph that plots the height function of the cluster hierarchies, i.e. the Euclidean distance between the linked pair of elements at each new iteration or the height of each hierarchy in the dendrogram representation. Below the graph, a slider with the same number of elements as the height function allows the user to specify the hierarchy of the clustering algorithm (or the cutting point in the dendrogram), which drives the dictionary construction by defining the number of audio classes to be adopted.

Figure 5. SEED graphical user interface.

The right-most option in the slider results in a dictionary composed of the same audio classes as the number of audio segments S, and will lead to the generation of sequences that correspond to the original input audio signal, thus no novelty is introduced. The left-most option in the slider results in a dictionary with a single audio class that encompasses all audio segments. In the latter case, the generated audio segment sequences result from uniformly-distributed random decisions. Within these extreme cases all clusters ranging from 1 to S can be selected, biasing the generation process towards a higher degree of novelty (i.e., less redundancy).

A height function expressing the similarity h between linked pairs of sound segments or cluster segments at each agglomerative clustering hierarchy q tends to assume an elbow-like shape due to the likelihood of environmental sounds to have a highly redundant content [8]. Thus, we consider the elbow point of this function a balanced solution between the novelty and redundancy in the signal content for driving the dictionary construction, and assign it as the default value once an audio input source is loaded. To compute this point, q^*, we find the point closest to $(1, h_1)$:

$$q^* = \text{argmin}_q \sqrt{(q-1)^2 + {h_q}^2} \qquad (3)$$

Figure 6 shows the height function from the data in Figure 2, from which we compute the elbow point by finding the minimum distance to $(1, h_1)$.

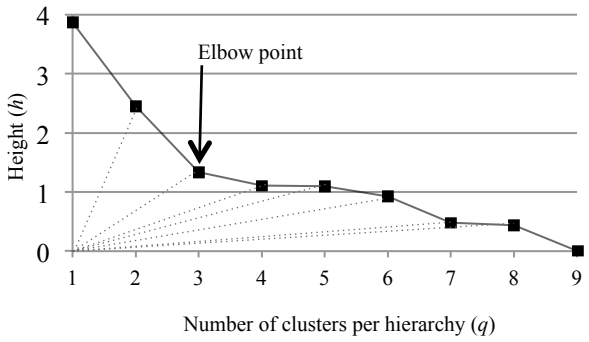

Figure 6. Height function for the different clusters hierarchies shown in Figure 2. Dashed segments indicate the distance to $(1, h_1)$ used to determine the elbow point.

The specification of a cluster hierarchy can be obtained in real-time since the subsequent transition table construction is very fast to compute. This allows users to easily explore different dictionaries of audio classes for a single input audio signal. Additionally, by adopting a manual strategy to fine-tune the audio similarity tolerance degree we avoid the formalization of contextual and subjective criteria inherent to the task.

6. EVALUATION

To evaluate our system SEED, we conducted an experiment to assess whether it can create natural sounding environmental sounds from examples. The experiment consisted of an online listening test, which aimed to i) compare SEED and related state-of-the-art systems in terms of the naturalness of their generated output and ii) assess the naturalness of SEED's output with complex input audio sources. Our hypothesis is that user ratings of SEED's generated examples will be higher than related systems and the difference between original and SEED complex examples are in line with the difference between the original and SEED simple examples, given the enhanced capacity of SEED to model the input signal structure, while enforcing smooth segment transitions.

6.1 Experiment Design

To compare SEED with related state-of-the-art systems in terms of the naturalness of their generated output, we conducted an online listening experiment based on the design and audio dataset used in the evaluation of Fröjd's and Horner's system for the resynthesis of environmental sounds [7]. This dataset consists of eight sonic environments with variable duration ranging from 3 to 23 seconds (see second column of Table 2), which cover a wide range of environmental sounds, whose source descriptions are listed in the first column of Table 2.

Besides the original files, the dataset includes generated output from three systems by: Dubnov et al. [5], Lu et al. [6], and Fröjd and Horner [7]. From the Fröjd and Horner system [7], one resynthesized audio example is provided for each source file. From those by Dubnov et al. [5] and Lu et al. [6] only a few renditions exist. Although a fair comparison to the two latter systems cannot be established, we include them in the listening test following a similar design principle as Fröjd and Horner [7].

To this dataset we added four more audio examples in order to assess the naturalness of SEED's output under more complex audio input signals, which oppose the simple sound textures of the first eight examples. Complexity in these four examples is expressed by the presence of highly dense sonic environments with overlapping events (e.g., sound sources *9. Battle* and *10. Square*), and in the presence of audio inputs with long-term temporal event dependencies (e.g. sound sources *11. Factory* and *12. Gallop*). Input signals with a high density of events can pose additional difficulties in segmentation and concatenation phases, and the long-temporal dependencies in modeling the temporal dimension of the input signal.

All generated audio examples have the same duration of their original audio source, which were also included in the listening test to allow a comparison between the effectiveness of SEED under more and less complex examples. Generated audio in SEED uses a clustering hierarchy computed automatically by the elbow method detailed in Section 5.

Participants were asked to rate on a 7-point Likert scale (1-7) *the naturalness of the sonic environments* (including source and generated audio), where 1 corresponds to highly unnatural and 7 to highly natural. To rate a given audio example participants were asked to listen to its entire duration using high quality headphones. To allow the participants to familiarize themselves with the experiment and, a short training phase was added to the listening test. To prevent response bias introduced by order effects, the musical examples were presented in a random order at each experiment trial. To submit their ratings and complete the listening test, the participants were obliged to rate all sound examples. Participants were not paid to take the experiment.

6.2 Results

In total, 20 subjects (14 female and 6 male) ranging in age from 19 to 41 years old ($mean = 28$ and $standard\ deviation = 7$) participated in the experiment. Four participants claimed to have *high expertise in sound design*, 12 claimed *some expertise in sound design*, and the remaining 4 *no expertise in sound design*. No participants declared hearing problems.

To examine the results of the listening test we show the average ratings per musical example for each system and original source (Table 2). The average and standard deviation (SD) ratings per system are shown in bold. Empty cells in Table 2 correspond to examples not included in the dataset. The average ratings per system largely concur with those of the experiment conducted by Fröjd and Horner [7] and the overall mean rating of SEED is higher than the compared systems. Yet, for the Fröjd and Horner and SEED systems, from which we could compare the entire set of available sound examples, a two-tailed t-test shows no statistically significant difference ($p = .2716$). Furthermore, all systems are rated lower than the original source files in terms of naturalness for most sonic environments (the difference between the original source files and Fröjd and Horner and between the original source files and SEED systems are statistically significant ($p < .0001$).

A deeper examination of the average subjective ratings of the first eight examples is shown in Figure 7. We compare the input source ratings to the systems for which we have the entire set of examples, i.e. Fröjd and Horner [7], and SEED. While the average ratings for the SEED system in the examples 2, 4, 5, and 7 are higher than those by Fröjd and Horner and are comparable with the average ratings of the naturalness of the audio input source. In the remaining four examples (1, 3 6, and 7) the average rating of the Fröjd and Horner are higher.

From the SEED examples with ratings lower than Fröjd and Horner, 1, 6, and 7 have almost identical ratings and audio example 3 has a noticeable difference. Listening back to excerpt 3, we can identify an obvious lack of variation in the example generated by SEED. In this (as in other examples) generated by Fröjd's and Horner's system, the clear overlap of audio chunks contributes to generate a larger degree of variation in simple textural environmental sounds. On the other hand, SEED seems to provide more compelling results for audio input sources that have clearly identifiable patterns (such as in examples 2 and 5).

Figure 8 shows the average naturalness ratings of the four audio input sources included in the listening test to assess the behavior of SEED under audio input signals with higher complexity. SEED ratings are uniformly lower than the input source signals, which is in line with the results from the less complex initial 8 examples. A significant decrease of 18% ($p < .0001$) of naturalness ratings between input sources and SEED output in the set of simples examples, follows a similar tendency in the set of complex examples, which have a significant decrease of 22% ($p < .0001$). Thus, we can conclude that SEED performs in a similar way under audio input sources with different degrees of complexity (as measured in terms of high density and long-term temporal dependencies of audio events).

		Average subjective ratings				
Scene description	Duration	Input Source	Lu et al.	Dubnov	Fröjd Horner	SEED
1. Aviary	9	4,80			2,85	2,80
2. Baby	14	5,75		3,20	3,25	4,40
3. Racing	16	5,15		1,90	4,40	2,90
4. Rain	3	4,40	3,65		2,90	4,25
5. Seagulls	15	5,05			4,15	5,20
6. Shore	19	4,60		1,85	4,70	4,25
7. Stream	5	3,70	3,65		3,05	3,45
8. Traffic	23	4,40		2,60	3,80	3,65
Average	**13**	**4,73**	**3,65**	**2,39**	**3,64**	**3,86**
SD	**6,87**	**1,729**	**2,05**	**1,84**	**1,83**	**1,82**
9. Battle	38	3,70				3,10
10. Square	45	6,10				4,30
11. Factory	59	5,40				4,60
12. Gallop	20	4,80				3,60
Average	**40**	**5,00**				**3,90**
SD	**16,22**	**1,47**				**1,67**

Table 2. Participant's ratings of the naturalness of 12 sonic environments extended in its original and resynthesized versions by four different systems.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented SEED, an application for generating variably long audio renderings of recorded environmental sounds while preserving their inherent structure. Our method introduces two main novelties in relation to state-of-the-art systems [3-7]. The first is to use concepts from concatenative sound synthesis to decouple structurally segmented audio into two moments (middle region and boundaries). Features extracted from segment boundaries are used to measure the concatenation cost between audio segments. Features extracted from the middle region of the segment are used to build a temporal model of audio signal input.

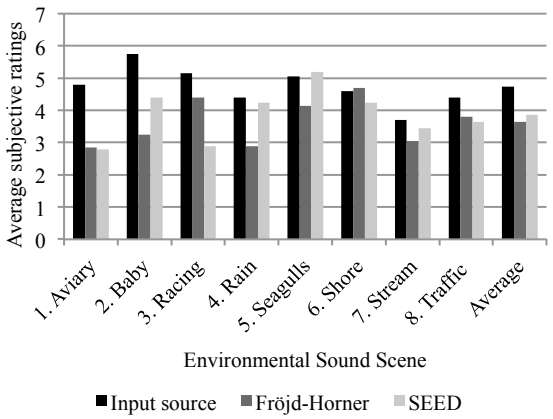

Figure 7. Average subjective ratings of the naturalness of eight (simple) environmental sound scenes, each including an original input audio source and two resynthesized versions by Fröjd and Horner and SEED.

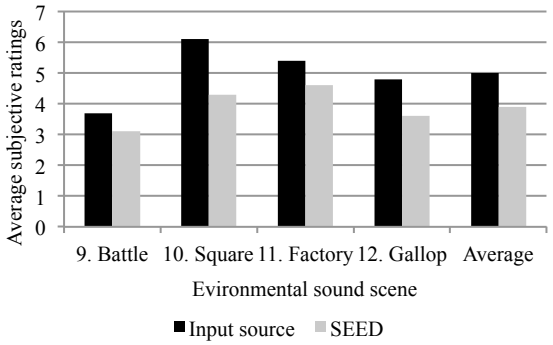

Figure 8. Average subjective ratings of the naturalness of four (complex) environmental sound scenes, each including an original input audio source and a resynthesized version by SEED.

The second contribution of our work is the use of a hierarchical clustering algorithm to reduce the dimensionality of segmented audio data into a discrete and finite dictionary of audio classes. This representation is then used to create a transition table, which encodes the audio input signal temporal structure. Based on the transition table we can then generate patterns that are similar to those in the input audio signal with variable degrees of variation.

Many potential applications exist for our work. Among these we can highlight: the modification and extension of recorded sounds to fit given scenes in sound design; the generation of ever-changing environmental sounds in games using limited memory storage; the restoration of audio signals (e.g. in the loss of audio or musical packets transferred over the Internet), and even in audio compression.

Through our preliminary evaluation, we have shown that SEED generates natural sounding environmental soundscapes to a degree that outperforms related state-of-the-art systems and that audio input signals with dense and complex temporal structures were similarly rated in relation to the original input audio signals. Furthermore, we hope SEED's interface encourages experimentation towards optimal results. Several examples generated by SEED, including those from the experiment are available online at: `http://bit.ly/29vnEhc`.

In future work, we will strive to assess the degree to which the parameter setting biases the generation as well as the best set of parameters in accordance to the type of input audio signals. In greater detail, we will further study the implication of the following parameters: duration of the input file, number of audio segments, entropy and redundancy in the audio input signal and its implications in the definition the number of classes in the dictionary, and the order of the transition table in relation to the all aforementioned settings. Furthermore, we plan to design and conduct an evaluation experiment in which generated audio examples exceed the duration of the input audio source. In this way we can then explicitly assess the extent to which a given file can be extended, and thus validate the ultimate goal of the application.

We will also study the use of audio descriptors that express information concerning the spatial-temporal content of the audio as a strategy to drive the resynthesis process aiming to further enhance temporal coherence of the meso structure of the generated output.

Acknowledgments

Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020" is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF). This research is also supported by the Portuguese Foundation for Science and Technology under the post-doctoral grant SFRH/BPD/109457/2015.

8. REFERENCES

[1] D. Sonnenschein, Sound Design. Michael Wiese Productions, 2001.

[2] R. Viers. Sound Effects Bible. Michael Wiese Productions, 2011.

[3] D. Keller and B. Truax, Ecologically based granular synthesis. Proceedings of the International Computer Music Conference, 1998, pp. 117-120.

[4] R. Hoskinson and D. Pai, "Manipulation and Resynthesis with Natural Grains," in Proceedings of the International Computer Music Conference, 2001, pp. 338-341.

[5] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Synthesizing Sound through Wavelet Tree Learning," in IEEE Computer Graphics and Applications, 22(4), 2002, pp. 38-48.

[6] L. Lu, L. Wenyin, and H.J. Zhang, "Audio Textures: Theory and Applications," in IEEE Trans. on Speech and Audio Processing, 12(2), 2004, pp. 156-167.

[7] M. Fröjd and A. Horner, "Fast Sound Texture Synthesis Using Overlap-add," in Proceedings of the

International Computer Music Conference, 2007, pp. 320-323.

[8] G. Strobl, G. Eckel, D. Rocchesso, and S. le Grazie, "Sound Texture Modeling: A Survey," in Proceedings of the Sound and Music Computing Conference, 2006, pp. 61-65.

[9] D. Schwarz, "State of the Art in Sound Texture Synthesis," in Proceedings of the Digital Audio Effects Conference, 2011, pp. 221-231.

[10] G. Bernardes, C. Guedes, and B. Pennycook, "Eargram: An Application for Interactive Exploration of Concatenative Sound Synthesis in Pure Data," in LNCS, 7900, 2013, pp. 110-129.

[11] W. Brent, "A Timbre Analysis and Classification Toolkit for Pure Data," in Proceedings of the International Computer Music Conference, 2009, pp. 224-229.

[12] J. Buys, Generative Models of Music for Style Imitation and Composer Recognition. Honours Project in Computer Science, University of Stellenbosch, 2011.

[13] D. Mitrovic, M. Zeppelzauer, and H. Eidenberger, "Analysis of the Data Quality of Audio Descriptions of Environmental Sounds," in Journal of Digital Information Management, 5(2), 2007, pp. 48-54.

[14] D. Keller and J. Berger, "Everyday sounds: Synthesis parameters and perceptual correlates," in Proceedings of the VIII Brazilian Symposium on Computer Music. Fortaleza, CE: SBC, 2001.

[15] G. Bernardes, Composing Music by Selection: Content-Based Algorithmic-Assisted Composition. Ph.D. thesis, University of Porto, 2014.

[16] B. S. Everitt, Cluster Analysis. Edward Arnold, 1993.

[17] N. Saint-Arnaud and K. Popat, "Analysis and Synthesis of Sound Texture," in D. F. Rosenthal, Horoshi G. Okuno (editors), Computational Auditory Scene Analysis. New Jersey, NJ: Lawrence Erlbaum Association, 1998.