Granular Synthesis of Sound Textures using Statistical Learning

Ziv Bar-Joseph  Dani Lischinski  Michael Werman
Institute of Computer Science
The Hebrew University of Jerusalem, Israel

Shlomo Dubnov  Ran El-Yaniv
Communication Systems Engineering
Ben-Gurion University, Beer-Sheva, Israel

Department of Computer Science
Technion – Israel Institute of Technology

Abstract

We present a statistical learning algorithm for synthesizing random sound textures resembling an input sound texture segment. Our approach begins by constructing a hierarchical multi-resolution representation of the input signal. The resulting tree data structure is then statistically sampled to generate a new tree from which the output sound texture is reconstructed. This method works for both periodic and stochastic sounds and for mixtures of both, without assuming any explicit model for the data. Our results indicate that the proposed technique is effective and robust.

1 Introduction

A large class of natural and artificial sounds such as rain, waterfall, traffic noises, people babble, machine noises, etc., can be regarded as sound textures — sound signals that are approximately stationary at some scale. This paper presents a new statistical learning algorithm for the granular synthesis of such sounds.

Granular synthesis [7] is one of the most appealing models for sound texture synthesis. To create a complex sound, thousands of brief acoustical events (grains) are combined. These methods, although rooted in time domain manipulations, are closely linked to time-frequency sound representation schemes such as the wavelet or short time Fourier transform, which provide a local representation by means of waveforms or grains multiplied by coefficients that are independent of one another.

An important distinction must be made between quasi-periodic signals and stochastic signals. Existing methods [3] for sound processing or synthesis assume specific sound models that differentiate between periodic and stochastic signals. Applying these methods requires some kind of preprocessing to separate the sound into periodic and stochastic parts. In our approach we do not assume any specific model for the sound source. Moreover, our method works equally well for both stochastic and periodic sound textures.

In this paper we present a statistical learning algorithm for synthesizing new random instances of a sound texture, given an example of such a texture as input. Treating the input sound texture as a sample of a stochastic process, we construct a tree representing a hierarchical wavelet transform of the signal. From this tree, new random trees are generated by learning and sampling the conditional probabilities of the paths in the original tree. Transformation of these random trees back into signals results in new sound textures.

Applications of this method are abundant and include, for example, automatic generation of sound effects, creative musical and sonic manipulations, and virtual reality sonification. Sound synthesis examples will be demonstrated during the talk.

2 Statistical Learning

In this section we briefly describe the statistical ideas that underly our synthesis algorithm. Since our method was developed as an extension of a statistical learning technique for "universal" prediction (and synthesis) of discrete sequences (consisting of discrete symbols from a finite alphabet) [6, 5], we begin by explaining the main idea behind the routine for prediction of discrete sequences. Given a discrete sequence \( x \), assumed to be sampled from some unknown statistical source \( S_x \), we would like to generate another sequence \( y \) that exhibits the same statistics. Without any assumption on the nature of the source there is little one can do in a principled manner. But when we assume that the unknown source \( S_x \) is stationary and ergodic it is possible to compute an estimated model \( \hat{S}_x \) for the source of \( x \), so that the maximum deviation between the probabilities assigned by these two models for any other string \( z \) is bounded by \( o(|z|) \) (which means that it approaches zero when \( z \) increase). Usually the model \( \hat{S}_x \) is in the form of conditional probabilities (of a symbol given a prefix string). Thus, the model can be used to generate new random sequences that exhibit the same statistics as \( x \).
Instead constructing an explicit model and then sampling new random sequences from it, one can generate a random sequence directly from $x$ as follows. Initially choose a random symbol of $x$ and set it to be $y_1$. Then, suppose that we have generated the first $i$ symbols $y^{(i)} = y_1, y_2, \ldots, y_i$ of the new random string $y$. Now, find the longest suffix $z$ of $y^{(i)}$ in $x$ and suppose $z$ occurred $k$ times in $z$. Uniformly choose at random one such occurrence and set $y_{i+1}$ to be the symbol immediately after this occurrence. Under the assumption that $x$ originates from a stationary Markov source of finite order, this method will generate a random sequence $y$ that will be statistically similar to $x$. In fact, a version of this method was first used by Shannon to generate random English (gibberish) sentences.

A simple extension of this method to sequences of real numbers (instead of "nominal" alphabet symbols) could be applied directly on sound waveforms, but it would fail to capture any large scale features of the signal, and probably not even pitch. Therefore, in this work we apply an extended method to paths in the tree representing the hierarchical multi-scale transform of the sound signal, as described in Section 4. Since we are dealing with sequences of real numbers we replace the exact suffix match in the original method by an approximate match using the sum of the absolute differences along the relevant paths.

3 Wavelets

Wavelets have become the tool of choice in analyzing single and multi-dimensional signals, especially if the signal has information both at different scales and times. The fundamental idea behind wavelets is to analyze the signal's local frequency at all scales and times, using a fast invertible hierarchical transform. Wavelets have been effectively utilized in many fields, including music and voice coding, image and video compression and processing, stochastic resonance, computer graphics and geometric modeling, fast numerical algorithms, and financial data analysis.

The wavelet is a multi-scale decomposition of the signal and can be viewed as a complete tree, where each level stores the projections of the signal onto the wavelet basis functions of a certain resolution. All basis functions at a particular level are translated copies of a single function. Thus, the wavelet transform is a series of coefficients describing the behavior of the signal at dyadic scales and locations.

The wavelets we use in this work to analyze the input signal and generate the corresponding multi-resolution analysis (MRA) tree are Daubechies wavelets [2]. The wavelet transform is computed as follows: initially, the signal is split into lowpass/scaling coefficients by convolving the original signal with a lowpass/scaling filter and the wavelet/detail coefficients are computed by convolving the signal using a Daubechies wavelet filter. Both responses are subsampled by a factor of 2, and the same filters are applied again on the scaling coefficients, and so forth. The time to compute this transform is linear in the size of the input signal.

The wavelet coefficients also can be transformed back into the original signal in linear time. The computation proceeds from the root of the tree down to the leaves, using filters that are complementary to those used in the wavelet transform.

4 Synthesis Algorithm

In this section we describe our algorithm for synthesizing a new sound texture from an example. As explained in the previous section, the input signal is wavelet-transformed to an MRA tree. From the point of view of our synthesis algorithm, the input signal is now encoded as a collection of paths from the root of the tree towards the leaves. Each such path is assumed to be a realization of the same unknown stochastic source. Our goal is to generate a new tree, whose paths are typical sequences generated by the same source. A new signal is constructed from the resulting random MRA tree by applying the inverse wavelet transform, as described earlier.

The pseudocode of our tree synthesis algorithm is shown in Figure 1. The generation of the new tree proceeds in breadth-first order (i.e., level by level). First, the value of the root of the new tree $T^*$ is copied from the root value of the input tree $T$. The values of the nodes on level 1 are copied as well. Now, let us assume that we have already generated the first $i$ levels of $T^*$. In order to generate the next level we must add two children nodes to each node in level $i$ (the MRA of a 1D signal is a binary tree). Let $v_i$ be the value of such a node, and denote by $v_{i-1}, v_{i-2}, \ldots, v_{i}$ the values of that node's ancestors along the path towards the root of the tree. Also denote by $p_{-1}, p_{-2}, \ldots, p_{-k}$ the nodes that appear on the same level in the MRA tree immediately preceding $v_i$ in time — in other words, the $k$ neighbors of $v_i$ from the left. The algorithm searches among all the nodes in the $i$-th level of $T$ for nodes $v_i$ with the maximal-length $\epsilon$-similar path suffixes $w_{i-1}, w_{i-2}, \ldots, w_j$, where $\epsilon$ is a user-specified threshold. This search is performed in the first part of the routine CandidateSet in Figure 1. For the nodes with the maximal-length suffixes, we look for those nodes whose $k$ predecessors are similar to those of $v_i$ (where $k$ is a user defined value). One of these candidate nodes is then randomly chosen and the values of its children are copied to the children of node $v_i$. In this way a complete new tree is formed.

The algorithm described above constructs an output tree $T^*$ of the same size as the input tree. In principle, it is easy to construct a larger (deeper) output tree. Let $d$ be the depth of the input tree. In order to construct an output tree of depth $d + 1$ we first invoke our algorithm twice to compute two new trees. The roots of these trees are then fed as a low resolution version of the new sig-
nal to the MRA construction routine, which uses those values to construct the new root and the first level of the new MRA hierarchy. In this manner it is possible to generate sound textures of arbitrary length from a short input sample of the sound.

| Input: | The MRA tree $T$ of the input signal, threshold $\varepsilon$ |
| Output: | A new tree $T^{\text{new}}$ generated by the estimated stochastic source of $T$ |

**Initialization:**

Root($T^{\text{new}}$) := Root($T$)
for $j = 1$ to 2
Child$_j(T^{\text{new}})$ := Child$_j(T)$
endfor

**Breadth-First Construction:**

for $i = 1$ to $d - 1$ (where $d$ is the depth of $T$):
foreach node $v_i$ on level $i$ of $T$
    $C$ := CandidateSet($T$, $i$, $v_i$, $\varepsilon$)
    Randomly choose a node $w_i$ in the candidate set $C$
    Copy the values of the children of $w_i$ to those of $v_i$
endfor

**procedure CandidateSet($T$, $i$, $v_i$, $\varepsilon$)**

Let $v_1$, $v_2$, ..., $v_{i-1}$ be the ancestors of $v_i$
foreach node $w_i$ on level $i$ of $T$
    Let $w_1$, $w_2$, ..., $w_{i-1}$ be the ancestors of $w_i$
    $L[w_i] := 0$, $sum := 0$
    for $\ell = i$ to 1
        $sum_+ := |w_\ell - w_\ell|$
        if $\frac{sum_+}{sum_{+\ell}} < \varepsilon$ then
            $L[w_i]++$ else break
    endfor

end procedure

$M_1 := \max_{w_i} L[w_i]$ Let $p_{-1}, p_{-2}, \ldots, p_{-k}$ be the predecessors of $v_i$
foreach node $w_i$ with $L[w_i] = M_1$
    Let $q_1, q_2, \ldots, q_k$ be the predecessors of $w_i$
    $S[w_i] := 0$
    for $\ell = -1$ to $-k$
        if $|p_\ell - q_\ell| < \varepsilon$ then
            $S[w_i]++$ else break
    endfor

end procedure

$M_2 := \max_{w_i} S[w_i]$ return the set of all nodes $w_i$
such that $L[w_i] = M_1$ and $S[w_i] = M_2$

Figure 1: The tree synthesis algorithm

4.1 Reducing the number of candidates

A naive implementation of the tree synthesis algorithm described above requires the examination of all the nodes at level $i$ in the original tree in order to find the maximal $\varepsilon$-similar paths for every node $v_i$ on level $i$ in the new tree. Given an $2^m$ input signal, in the bottom level our algorithm has to check $2^{m-1}$ nodes in the new tree, so applying the naive algorithm results in a number of checks that is quadratic in this number. Since for each node we check a path of length $m - 1$, and the $k$ predecessors, this exhaustive search makes the synthesis of long sound signals very slow. However, much of the search can be avoided by selecting the candidate sets from parents to their children in the tree. Thus, while searching for maximal $\varepsilon$-similar paths of node $v_i$ the algorithm must only examine the children of the nodes in the candidate sets that were found for $v_{i-1}$ while constructing the previous level. The result is a drastic reduction in the number of candidates. The actual number of candidates depends of course on the threshold, but in almost all cases we found that the number is very small (between 4 and 16).

4.2 Threshold selection

Two paths are considered similar by our algorithm when the differences between their values are below a certain threshold. The threshold controls the amount of randomness in the resulting signal, and its similarity to the input. In the extreme, too small a threshold may cause the result to be an exact copy of the input. Thus, the value of the threshold has a large impact on the outcome.

We cannot expect the user to select an appropriate threshold for the temporal dimension, because it is difficult to assess the size of the temporal features in the sequence simply by observing it. Our technique for choosing a threshold for the temporal dimension is inspired by wavelet compression methods for images [4]. The idea behind wavelet compression is to zero out coefficients with $L_1$ norm less than some small number $a$. This decimation of the coefficients results in little perceptual effect on subjective image quality. By the same token, we assume that switching is permitted between coefficients whose values are no more than $2a$ apart. Thus, we let the user specify a percentage $P$. We then compute the interval $[-a, a]$ containing $P$ percent of the signal coefficients. The temporal threshold value is then set to $2a$.

4.3 Relation to earlier work

In previous work [1], we demonstrated the use of our statistical learning algorithm to the task of synthesizing 2D textures and texture movies. However, there is a significant difference between the synthesis of those textures and the synthesis of sound textures. In our sound synthesis algorithm, we demand that the candidates to continue a node in the synthesized tree come from nodes in the input tree that have not only a similar set of ancestors but also a similar set of predecessors. This additional constraint, which is used for the first time in this work, comes from the nature of the sound signal. There is a clear and natural ordering of the values of such a signal (according to the temporal dimension). Thus, the order of the values must be taken into account when generating the new MRA tree. This distinguishes our method from the 2D texture synthesis methods, in which there is no reference to ordering.
Another advantage of looking at the predecessors comes from the tree nature of our algorithm. It is possible for temporally adjacent nodes in the tree to have paths with very few common ancestors. Thus, if predecessor similarity is not taken into account, such two nodes would be synthesized almost independently of each other. By imposing predecessor similarity, we force the temporal neighborhood of each node to be taken into account by our algorithm.

5 Results and Summary

We extensively tested our algorithm on a variety of sound examples, ranging from "noisy" textures such as waterfall sounds, to musical excerpts such as a solo clarinet or a jazz ensemble. The MRA we used for the synthesis algorithm was obtained using the Daubechies wavelet of length 10 as explained in Section 3. The threshold for the learning algorithm was computed as described in Section 4.2 where $P$ was usually between 60 to 70 percent. The number of predecessors we checked for each node $(k)$ was 5.

Our results demonstrate a high quality resynthesized sound, with almost no artifacts due to the random granular recombination of different segments of the original input sound. Moreover, there is a correct recreation of the percussive and "noisy" parts as well as the pitched parts. Testing the same idea without checking similarity of the predecessors (only checking the ancestors), can still produce good results if the original input is completely "textural", such as a sound that contains noise and percussion components only. Whenever a pitch or quasi-periodic component is present in the signal, the checking of the predecessors is crucial. Without such checking, the resulting signal has significant distortion. Grossly speaking, imposing predecessor similarity ensures a smooth interpolation between adjacent components in time, which is necessary when periodicity exists in the input signal.

In Figure 2 we demonstrate the application of our algorithm to the task of synthesizing a jazz music segment. The input track contains rapid percussion (a drum set including hi-hat) and electric guitar playing relatively short notes. The output signal also contains drums and guitar segments, but the sound is noticeably different from the original input.

Our results suggest that a principled mathematical approach to granular analysis and resynthesis is possible, even for the case of mixed periodic and stochastic sounds. This is, to our knowledge, the first approach to sound texture analysis/resynthesis that does not assume an implicit sound model. Moreover, it does not require a separate treatment of periodic and stochastic components.

Acknowledgements

This research was supported in part by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities. Ran El-Yaniv is a Marcella S. Geltman Memorial Academic Lecturer.

References


