Synthesizing Sound Textures through Wavelet Tree Learning

Shlomo Dubnov
Ben-Gurion University, Isreal

Ziv Bar-Joseph
Massachusetts Institute of Technology

Ran El-Yaniv
Technion–Israel Institute of Technology

Dani Lischinski and Michael Werman
Hebrew University of Jerusalem, Israel

Natural sounds are complex phenomena because they typically contain a mixture of events localized in time and frequency. Moreover, dependencies exist across different time scales and frequency bands, which are important for proper sound characterization. Historically, acoustical theorists have represented sound in numerous ways. Our research has focused on a granular method of sonic analysis, which views sound as a series of short, distinct bursts of energy. Using that theory, this article presents a statistical learning algorithm for synthesizing new random instances of natual sounds.

Applying wavelet analysis, our algorithm captures the joint statistics of the coefficients across time and scale. We develop a method for synthesizing new random instances of a *sound texture* given an example of such a texture as input. We can describe sound textures as a set of repeating structural elements (sound grains) subject to some randomness in their time appearance and relative ordering but preserving certain essential temporal coherence and across-scale localization. For a large class of natural and artificial sounds such as rain, a waterfall, traffic noises, people babble, machine noises, and so on, we can assume that the sound signals are approximately stationary at some scale.

More specifically, treating the input sound texture as a sample of a stochastic process, we construct a tree representing the signal's hierarchical wavelet transform. From this tree, we generate new random trees by learning and sampling the conditional probabilities of the paths in the original tree. Transforming these random trees back into signals results in new sound textures that closely resemble the original sound source's sonic impression without exactly repeating it. Note that our method generates the new tree down to the lowest detail level without resorting to higher level atomic segmentation, thus achieving a fine grained novel resynthesis. Moreover, our method works equally well for both stochastic and periodic sound textures.

Our method closely relates to several recent 2D texture synthesis techniques. We discuss this relation in more detail in the sidebar "Relation to Earlier Work."

Applications of our method are abundant and include, for example, automatic generation of sound effects, creative musical and sonic manipulations, and virtual reality sonification. We visually demonstrate the sound synthesis examples in this article and acoustically demonstrate them in an accompanying Web site (see http://www.cs.huji.ac.il/~danix/texsyn/sound) and in the *CG&A* CD-ROM supplement included with this issue.

We present a statistical learning algorithm for synthesizing random instances of sound textures from an existing natural sound example.

Granular synthesis and sound textures

More than 50 years ago, Gabor[1] formulated a theory that sound is perceived as a series of short, discrete bursts of energy. He suggested that within a short time window (10 to 20 milliseconds) the ear could register one event at a specific frequency. This approach has led to the development of granular methods of sonic analysis. Granular synthesis[2-4] is one of the most appealing models for sound texture synthesis. Creating a complex sound requires combining thousands of brief acoustical events (grains). 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 provides a local representation through waveforms or grains multiplied by coefficients independent of one another. Note that granular synthesis is a generative technique that lacks analysis methods. In this article, we suggest one such method that lets us construct, in a principled manner, a new desired sound according to statistics derived from an existing natural sound example.

When considering sound models, we must distinguish between quasiperiodic and stochastic signals. Existing methods for sound processing or synthesis usually differentiate between periodic and stochastic signals. Applying these analysis and synthesis methods requires some kind of preprocessing to separate the sound into periodic and stochastic parts[3] as well as determining natural transition points for segmenting a sound into atomic grains.[4]

0272-1716/02/$17.00 © 2002 IEEE

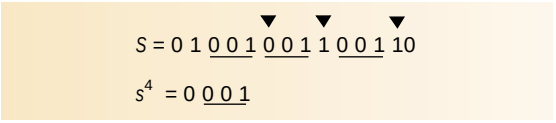
In our approach, we don't assume any specific model for the sound source. In this article, we focus on statistical analysis and random resynthesis of a subclass of sounds that we call textures. These types of sounds are amenable to local stationary modeling. Thus, our only assumption is that statistical characterization of the joint time-frequency or time-scale relations of the signal is possible.

Statistical learning

One of the main tasks in statistical learning[5] is estimating an unknown stochastic source given one or more training examples, which are *samples* from the source. For instance, a sample can be a sequence of movie frames, a texture image, a sound segment, and so on. Statistical learning aims to construct a statistical model of the source that fits the given sample(s) and generalizes it to generate previously unseen samples. Generalization is a key issue in learning. A model with good generalization can generate new random samples with a probability distribution that resembles that of the unknown source. Generating such new random samples is referred to as *sampling the model*. Thus, our basic assumption in this work is that the sound texture given to us as input is a random sample of an unknown stochastic source. Our goal is to faithfully learn a statistical model of this source.

Because we developed our method as an extension of a statistical learning technique for universal prediction (and synthesis) of discrete sequences (consisting of discrete symbols from a finite alphabet),[6,7] we begin by explaining the main idea behind the routine for predicting discrete sequences. Consider signal samples s_1, s_2, …, s_k where each of the signals s_i is assumed to emerge from a stochastic source S_i. Although the sources S_i are unknown, we assume that s_i is a typical example conveying the essential statistics of its source. Our task is to estimate the statistics of a hypothetical source $Z = Z(S_1, \ldots, S_k)$ called the *mutual source* of $S_1, \ldots, S_k$. Intuitively, the source Z is the closest (in a statistical sense) to all the S_i simultaneously. After learning the mutual source Z, we can sample it to synthesize mixed signals that are statistically similar to each of the sources S_i (and the examples s_i). In the special case where the samples $s_1, \ldots, s_k$ originate from the same source S (that is, $S=S_1=\cdots=S_k$), the mutual source Z is exactly S. When we sample from Z, we synthesize new random examples that resemble each of the given examples s_i.

The particular estimation algorithm for the sequences we use extends the algorithm by El-Yaniv,[7] which operates on sequences over a finite alphabet rather than on real numbers. Given a sample sequence s, this algorithm generates new random sequences that could have been generated from the source of s. In other words, based only on the evidence of s, each new random sequence is simultaneously statistically similar to s. The algorithm generates the new sequence without explicitly constructing a statistical model for the source of s as follows. Suppose we generated s^i, the first i symbols of the new sequence. To choose the next $(i+1)$ symbol, the algorithm searches for the longest suffix of s^i in s. Among all the occurrences of this suffix in s, the algorithm chooses one such occurrence x uniformly at random and chooses the next symbol of s^i to be the symbol appearing immediately after x in s. Figure 1 illustrates this process.



1 A step in a model-estimation algorithm for sequences over a two-letter alphabet.

Assume that the sequence s^4 has already been generated based on the input sample sequence s. To sample S for the next symbol (to be concatenated to s^4), we locate all the occurrences of the longest suffix of s^4 in s. In Figure 1, the longest suffix is 001 and its three occurrences in s are underlined. We now randomly and uniformly

Relation to Earlier Work

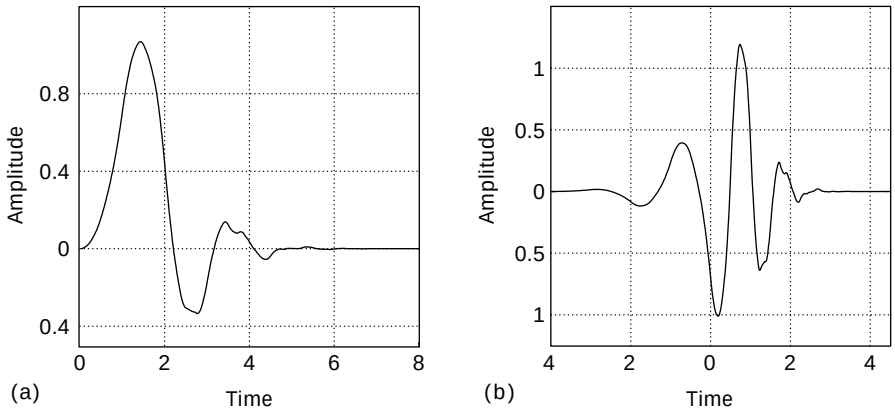
In previous work,[1] we demonstrated the application of our statistical learning algorithm to the task of synthesizing 2D textures and texture movies. However, a significant difference exists between synthesizing 2D texture images and sound textures. In our sound-synthesis algorithm, we demand that the candidates "to continue a node in the synthesized tree" come from nodes in the input tree that have a similar set of ancestors and a similar set of temporal predecessors. This additional constraint, which we used for the first time in this work, comes from the nature of the sound signal. There's a clear and natural ordering of the values of such a signal (along the temporal axis). Thus, we must consider the order of the values when generating the new MRA tree. This distinguishes our method from statistical 2D texture synthesis methods such as Bar-Joseph et al.[1] and De Bonet,[2] in which there's no linear ordering of values.

Another advantage of looking at the predecessors comes from our algorithm's tree nature. It's possible for temporally adjacent nodes in the tree to have paths with few common ancestors. Thus, if predecessor similarity weren't considered, then two such nodes would be synthesized almost independently of each other. By imposing predecessor similarity, we force our algorithm to consider each node's temporal neighborhood.

Our approach resembles the more recent independently developed 2D texture synthesis technique of Wei and Levoy.[3] Their method fills in pixels in a synthesized texture based on the similarity of the casual neighborhood of the pixel to be filled in to neighborhoods in the original texture, as well as based on the neighborhoods of the pixel's ancestors. To our knowledge, Wei and Levoy's method hasn't yet been applied to sound texture synthesis.

References

1. Z. Bar-Joseph et al., "Texture Mixing and Texture Movie Synthesis Using Statistical Learning," *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 2, Apr.–Jun. 2001, pp. 120-135.
2. J. S. De Bonet, "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," *Computer Graphics* (Proc. Siggraph 97), ACM Press, New York, 1997, pp. 361-368.
3. L-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Computer Graphics*, (Proc. Siggraph 2000), ACM Press, New York, 2000, pp. 479-488.



2 (a) The Daubechies scaling and (b) wavelet functions used in our implementation. (The time axis represents relative position for each scale.)

choose one of these subsequence occurrences and take the symbol immediately following the chosen occurrence. Two of these occurrences are followed by 1 and one occurrence is followed by 0 (denoted by the down arrows). Therefore, the probability of generating $s^5 = 00011$ is 2/3 and the probability of generating $s^5 = 00010$ is 1/3.

In our case, the sequences correspond to paths in a sound-texture wavelet tree. (We describe the construction of such trees in the next section.) Each tree node contains a real number and not a discrete symbol. Therefore, we must modify the sequence generation algorithm to work on sequences of real numbers instead of sequences over a finite alphabet. Accordingly, 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. We describe the modified algorithm in the section "Synthesis algorithm."

Wavelets

Wavelets have become the tool of choice in analyzing single and multidimensional signals, especially if the signal has information 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. Researchers have effectively used wavelets 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.

A wavelet representation is a multiscale decomposition of the signal. We can view it 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.

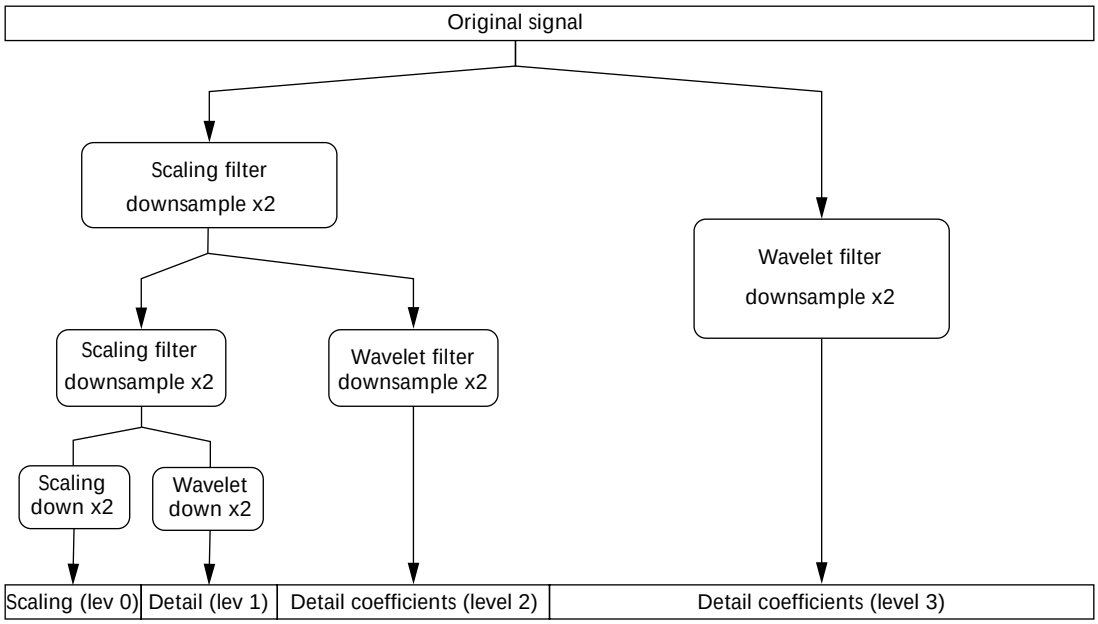
The wavelets we use in this work to analyze the input signal and generate the corresponding multiresolution analysis (MRA) tree are Daubechies wavelets[8] with five vanishing moments (see Figure 2). We compute the wavelet transform using a cascade filter bank as follows. Initially, we split the signal into low-pass or scaling coefficients by convolving the original signal with a low-pass or scaling filter and compute the wavelet and detail coefficients by convolving the signal using a Daubechies wavelet filter. We subsample both responses by a factor of 2 and apply the same filters again on the scaling coefficients, and so forth. Figure 3 illustrates this process. The time to compute this transform is linear in the size of the input signal.

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

Figure 4 shows the wavelet analysis of a crying baby sound, which contains sharp and continuous sound elements (coughing and crying). The detail coefficients at the four finest levels (as well as the two levels in the zoomed view) reveal a strong dependence between the parent coefficients in one level and their corresponding children coefficients in the next level. Furthermore, the signal's temporal structure at each level is far from random. We can visually observe the rather regular alternating sign (or a modulated sinusoidal-like) temporal behavior of the coefficients at scale 14. We can't preserve this behavior in the resynthesis step from scale considerations—it must be imposed over time. Achieving appropriate temporal evolution avoids clicks or other artifacts in the resynthesis.

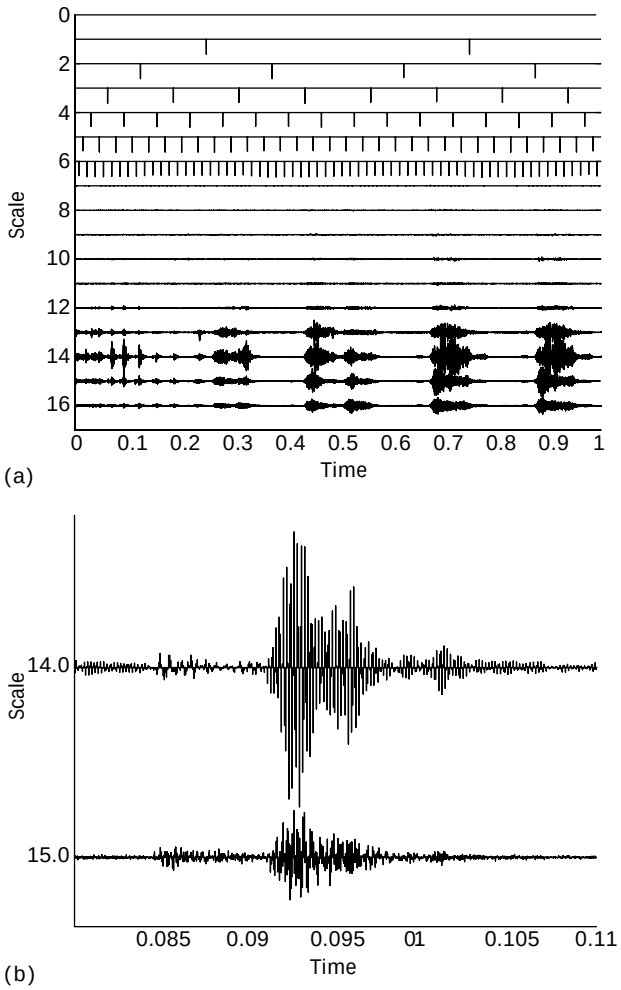
In summary, each coefficient depends on its scale ancestors (upper level) and temporal predecessors in the same level (those to its left). Thus, in our approach we model the statistics of every coefficient as dependent on both its ancestors from coarser levels and its predecessors in the same level.

A key issue when modeling, analyzing, and learning signals is choosing how to represent them. For example, we can represent a 1D signal with a sequence of values. Alternatively, we can represent it in the frequency domain via the Fourier transform. These two representations are common for modeling 1D (and 2D) signals, and various well-established approaches exist for estimating the underlying unknown source with respect to such representations (see, for example, Box et al.[9] and Merhav and Feder[10]). Although such representations have been successfully used in many application domains, they aren't adequate for representing and analyzing textural signals such as the ones we treat in this article. Such signals typically contain detail at different scales and locations.



3 Wavelet decomposition process.

Many recent research efforts suggest that better representations for such signals are multiresolution structures, such as wavelet-based representations (see, for example, Basseville et al.[11] and Wornell and Oppenheim[12]). Basseville et al. provide a theoretical formulation of statistical modeling of signals by multiresolution structures. These results consider a generative stochastic source that can randomly generate multiresolution structures. They show that stationary signals can be represented and generated in a hierarchical order, where first the coarse, low-resolution details are generated and then the finer details are generated with the probability that only depends on the already given lower resolution details. We've successfully applied hierarchical representation to modeling the statistics of 2D textures and 3D time-dependent textures.[13]



4 Crying baby sound. (a) The wavelet coefficients show the multiresolution structure of this signal with dependencies across scale and time. (b) This graph zooms in on a pair of components over a short period of time.

The assumption that the signal is emitted from a stationary source entails that its tree representation exhibits the following property: all the paths from the root to all nodes of the same level have the same distribution and any such path can be effectively modeled via a finite-order stochastic process. In the case of audio signals, we must formulate more sophisticated dependency structures to capture the connection across time and scale simultaneously. In this case, allowing arbitrary links might be too complicated and unnecessary. We simplify the situation by assuming that each wavelet sequence is stationary across time. This lets us model the joint distribution as

Input: The MRA tree T of the input signal, threshold ε
Output: A new tree T^{new} generated by
the estimated stochastic source of T

Initialization:
Root(T^{new}) := Root(T)
for $j = 1$ to 2
  Child$_j$(T^{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^{new}
    C := CandidateSet(T, i, v_i, ε)
    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
endfor

procedure CandidateSet(T, i, v_i, ε)
Let $v_1, v_2, \ldots, v_{i-1}$ be the ancestors of v_i
foreach node w_i on level i of T
  Let $w_1, w_2, \ldots, w_{i-1}$ be the ancestors of w_i
  $L[w_i] := 0$, sum := 0
  for $l = i$ to 1
    sum+= $|v_l - w_l|$
    if sum/$(i-l+1) < \varepsilon$ then $L[w_i]$++ else break
  endfor
endfor
$M1 := \max_{w_i} L[w_i]$
Let $p_{-1}, p_{-2}, \ldots, p_{-k}$ be the predecessors of v_i
foreach node w_i with $L[w_i] == M1$
  Let $q_{-1}, q_{-2}, \ldots, q_{-k}$ be the predecessors of w_i
  $S[w_i] := 0$
  for $l = -1$ to $-k$
    if $|p_l - q_l| < \varepsilon$ then $S[w_i]$++ else break
  endfor
endfor
$M2 := \max_{w_i} S[w_i]$
return the set of all nodes w_i
  such that $L[w_i] == M1$ and $S[w_i] == M2$

5 The tree synthesis algorithm.

an innovation process—that is, the probability of an error between its value and the expectation conditioned on its past. Again, we can do this by using the model-estimation algorithm.

We adopted this view and developed an algorithm that learns a source's conditional distributions. We transform the sample to its multiresolution, tree representation and learn the conditional probabilities along the tree's paths using an estimation method for linear sequences. Once a candidate tree is generated according to ancestor or child statistics, the algorithm performs a second learning phase across time. This second search is done along the predecessor nodes—neighboring nodes appearing previously in time. (Note that these nodes could be rather distant in terms of their tree graph topology.) This step results in a reduced set of candidate sequences that represent statistically significant dependencies among wavelet coefficients of the original signal across scale and time.

Synthesis algorithm

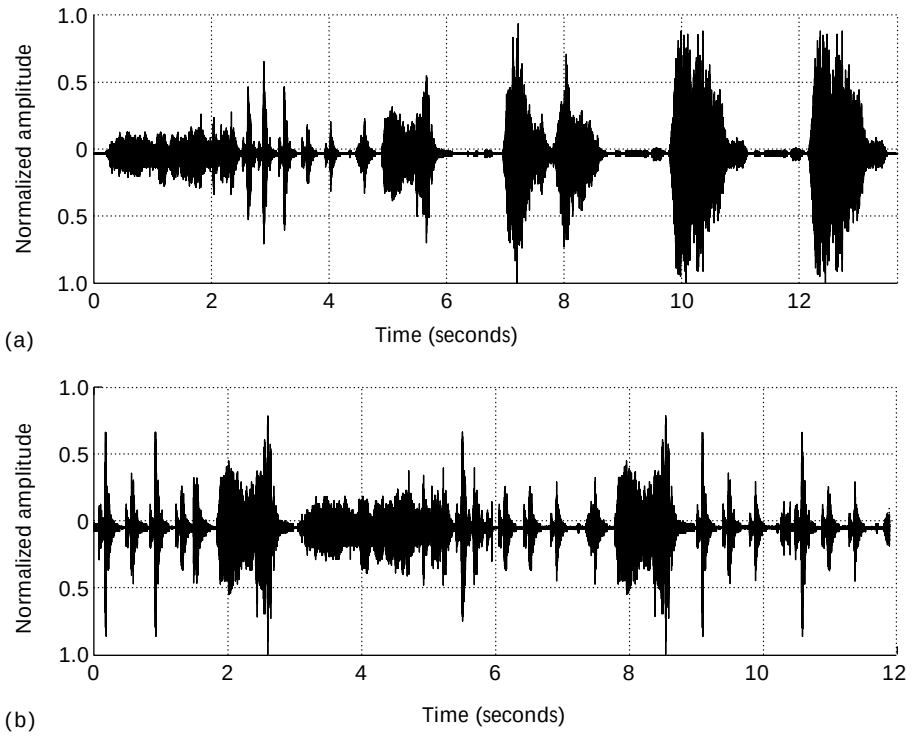
Here we describe our algorithm for synthesizing a new sound texture from an example. As we explained earlier, the input signal is wavelet-transformed to an MRA tree. From our synthesis algorithm's point of view, the input signal is now encoded as a collection of paths from the root of the tree toward the leaves. We assume each such path 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. We construct a new signal from the resulting random MRA tree by applying the inverse wavelet transform, as we described earlier.

Figure 5 shows the pseudocode of our tree synthesis algorithm. The generation of the new tree proceeds in breadth-first order (that is, level by level). First, the algorithm copies the value of the root of the new tree T^{new} from the root value of the input tree T. The algorithm copies the values of the nodes on level 1 as well. Now, let's assume that we've already generated the first i levels of T^{new}. 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 $v_{i-1}, v_{i-2}, \ldots, v_1$ as the values of that node's ancestors along the path toward the tree's root. Also, denote $p_{-1}, p_{-2}, \ldots, p_{-k}$ as 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 ith level of T for nodes w_i with the maximal-length ε-similar path suffixes $w_{i-1}, w_{i-2}, \ldots, w_j$, where ε is a user-specified threshold. This search is performed in the first part of the routine CandidateSet in Figure 5. For the nodes with the maximal-length suffixes, we look for those nodes whose k predecessors resemble those of v_i (where k is a user-defined value). The algorithm then randomly chooses one of these candidate nodes and copies the values of its children to the children of node v_i. This process forms a complete new tree.

Our algorithm constructs an output tree T^{new} of the same size as the input tree. In principle, it's easy to construct a larger (deeper) output tree. Let d be the input tree's depth. To construct an output tree of depth $d + 1$, we first invoke our algorithm twice to compute two new trees. We then feed the roots of these trees as a low-resolution version of the new signal 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, we generate sound textures of arbitrary length from a short input sample of the sound.

Reducing the number of candidates

A naive implementation of our tree synthesis algorithm requires examining all the nodes at level i in the original tree to find the maximal ε-similar paths for every node v_i on level i in the new tree. Given a 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 algo-



6 Crying baby sound. (a) The source recording and (b) resynthesis results.

rithm results in a quadratically growing number of checks. Because we search for a path of length $m-1$ and the k predecessors for each node, synthesizing long sound signals becomes prohibitively slow. However, we can avoid much of the search by inheriting the candidate sets from parents to their children in the tree. Thus, while searching for maximal ε-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. Of course, the actual number of candidates depends on the threshold, but in almost all cases, we found that the number is small (between 4 and 16).

Threshold selection

Our algorithm considers two paths similar when the differences between their values fall 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 might cause the result to be an exact copy of the input. Thus, the threshold's value has a large impact on the outcome.

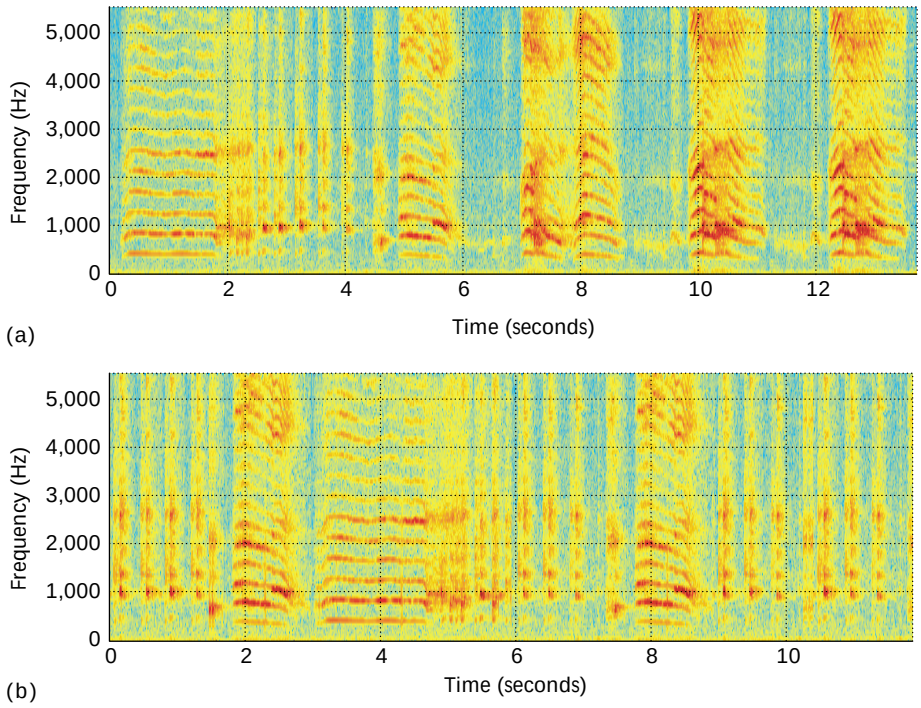
We can't expect users to select an appropriate threshold for the temporal dimension because it's difficult to assess the size of the temporal features in the sequence simply by observing it. Our technique for choosing a threshold for the temporal dimension is inspired by wavelet compression methods for images.[14] The idea behind wavelet compression is to zero out coefficients with an 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 let our algorithm switch between coefficients whose values are no more than $2a$ apart. Thus, we let users specify a percentage P. We then compute the interval $[-a, a]$ containing P percent of the signal coefficients and set the temporal threshold value to $2a$.

Results

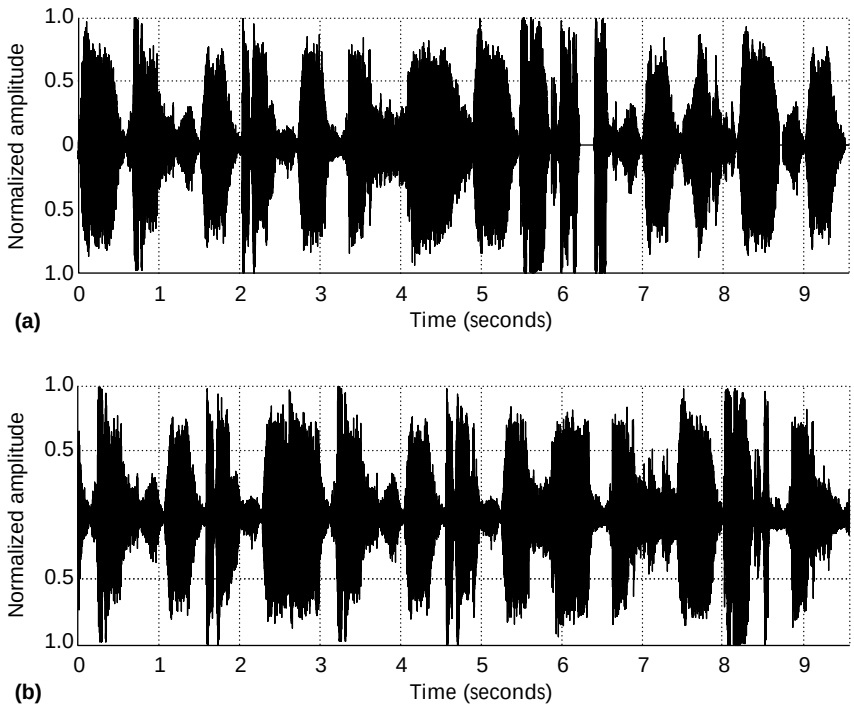
Our results demonstrate a high-quality resynthesized sound, with almost no artifacts due to the random granular recombination of different segments of the original input sound. We used the Daubechies wavelet with five vanishing moments to compute the MRA used by our synthesis algorithm. We computed the threshold for the learning algorithm with P usually set between 60 to 70 percent, and we checked five predecessors for each node (k).

The results show that there's a correct recreation of the short (transient or percussive), noisy, and burst-like parts as well as the more periodic or pitch parts. Applying our algorithm without learning the similarity of the predecessors—that is, only checking the ancestors' dependence—can still produce good results if the original input is completely segmented (for example, a sound that contains noise and transient or percussive components only). It's important to check the predecessors whenever a signal contains a pitch or quasiperiodic component. Without such checking, the resulting signal suffers from 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.

Figure 6 shows the application of our algorithm to the task of synthesizing a crying baby's sound. The input sound contains short coughing sounds and longer pitched crying sounds. To better observe the different sound elements' structure, we analyzed the same sound



7 Spectral images of the crying baby sound. (a) The source recording and (b) the resynthesis results.

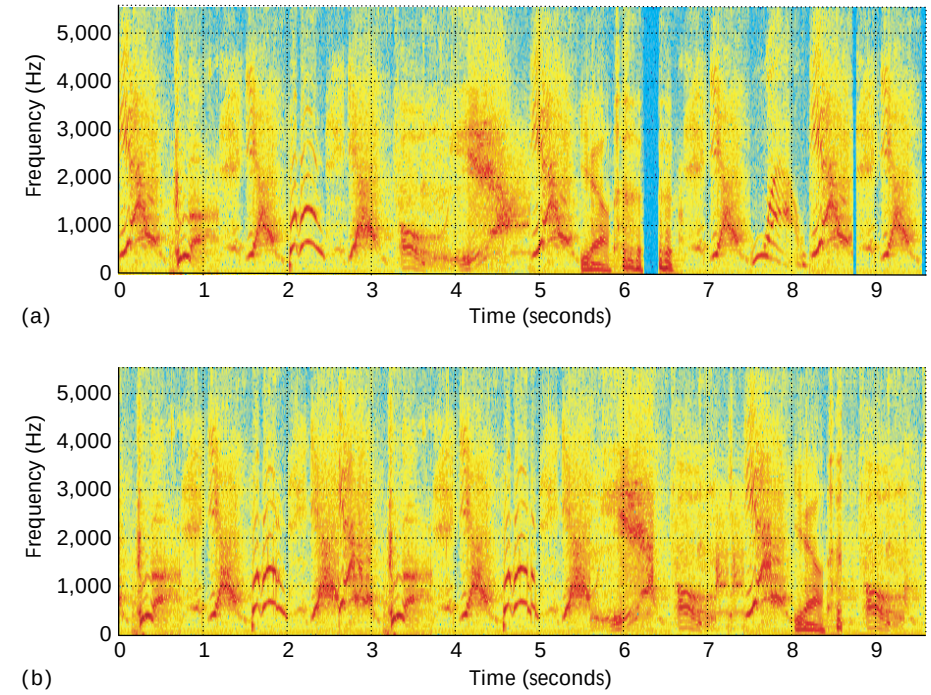


8 Medley sound. (a) The source recording and (b) resynthesis results.

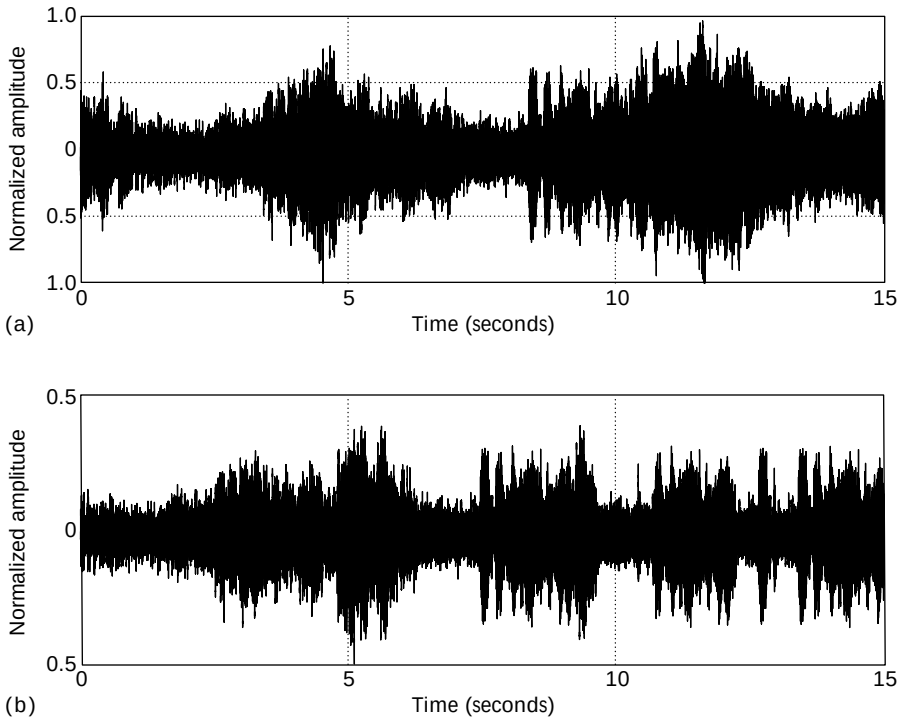
by short-time Fourier analysis (spectrogram). Figure 7 compares the resulting spectra.

Figure 8 shows another example that even better demonstrates the algorithm's capability to handle both short and continuous sounds of a stochastic and periodic nature. This example is a funny mixture of short vocal expression such as bursts of laughter, shouts, and exclamations. The mixing here is a reshuffling of the correct sound segments. The results show that the algorithm arrives at almost perfect segmentation of the different sounds and creates a smooth recombination. This is one of the best examples of the operation of our synthesis method. Figure 9 shows the spectral images of the same sounds.

Our third example is a traffic-jam sound recording (see Figure 10). This is an example of a sound texture that contains overlapping background noise of the street noise and shorter duration harmonic events of the car



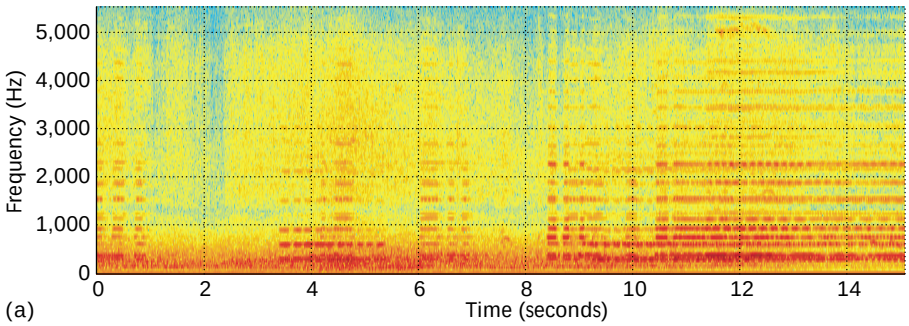
9 Spectral images of the medley sound. (a) The source recording and (b) resynthesis results.

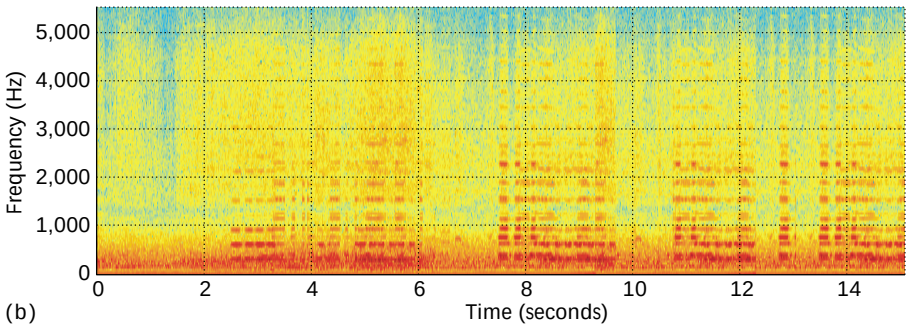


10 Traffic-jam sound. (a) The source recording and (b) resynthesis results. This example demonstrates how our algorithm handles overlapping background noise with shorter harmonic events.

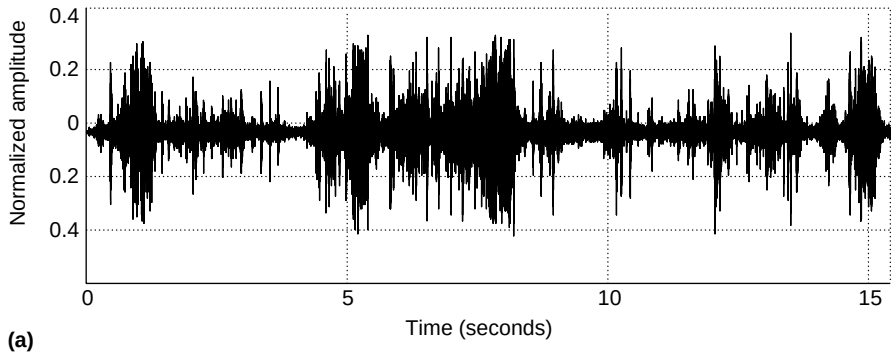
horns. The resulting sound (Figure 11, next page) combines all components, although in quite a different manner than in the original excerpt.

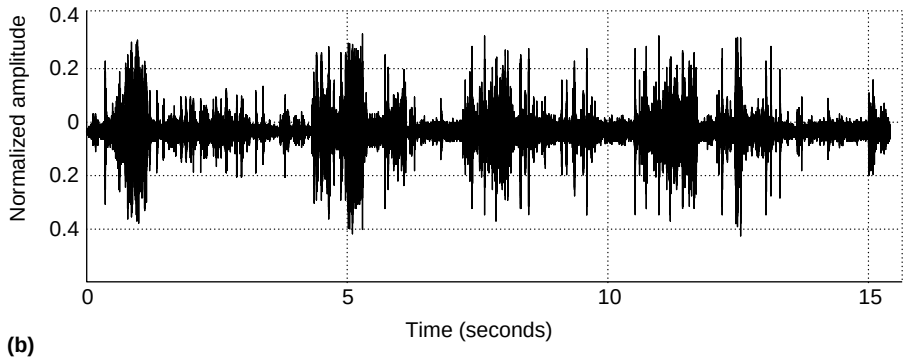
A sound that seems to be difficult for our algorithm is splashing water on a shore, shown in time and spectral views in Figures 12 and 13 (next page), respectively. This sound texture consists of a continuous background texture overlaid by transient splashes. The resulting synthesized sound reveals that the algorithm sometimes creates repetitions of short splashes that weren't apparent in the original sound. Because our ear is sensitive to rhythmical patterns, these repetitions create an impression of a more "nervous" splashing activity than the original sound. Although it's difficult to observe this artifact, you'll notice the fast repeating shapes in the sound waveform (see Figure 12b) at





11 Spectral images of the traffic-jam sound. You can see the overlapping background noise and shorter harmonic events in the short-time Fourier analysis of the (a) original and (b) resynthesized signal.

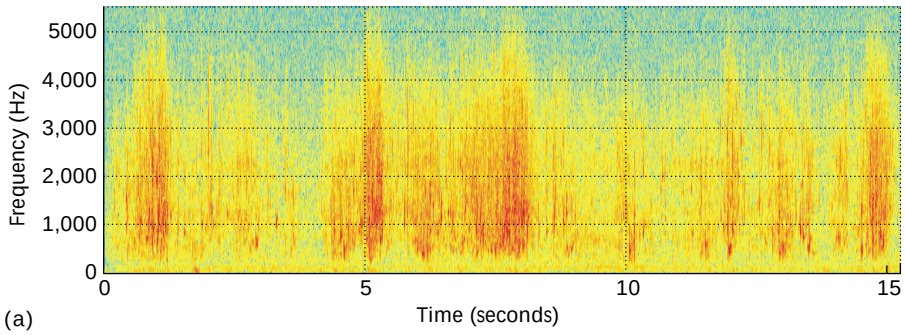


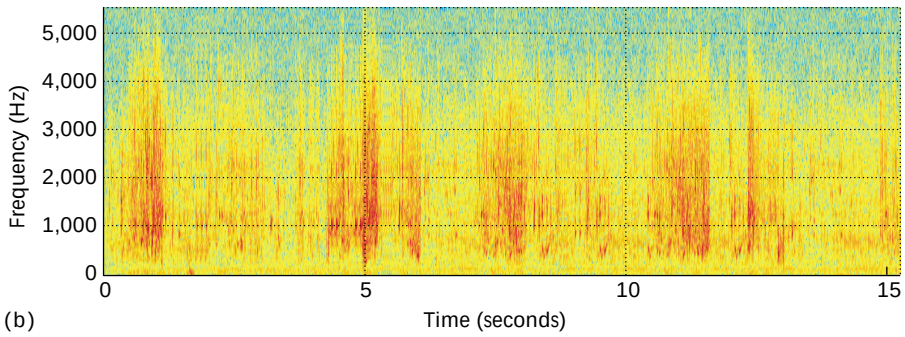


12 Washing shore sound. (a) The source recording and (b) resynthesis results. The resynthesized sound tends to create longer periodic splashes than what's apparent in the original recording.

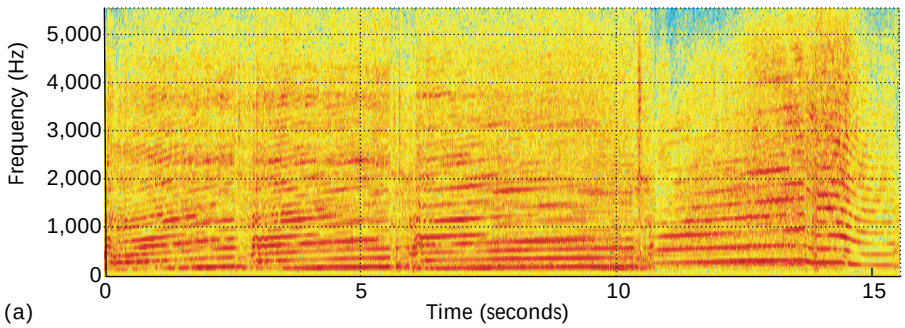
around 2 to 3 seconds of the synthesized sound.

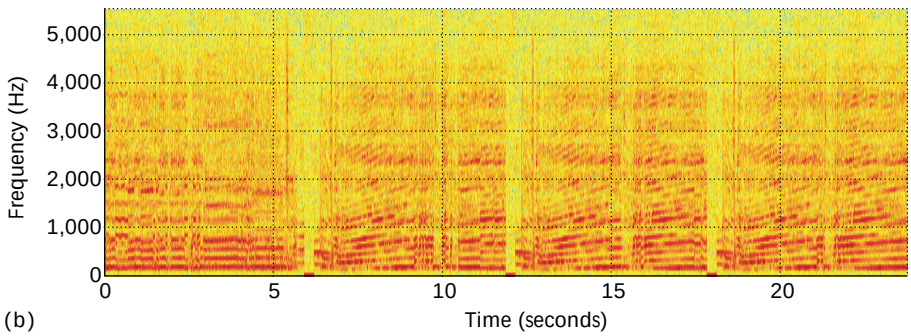
The last example is of a Formula 1 race (see Figure 14). Here the sound file contains the sounds of an accelerating engine, gear shifts, and another passing car. This example demonstrates that our algorithm still has limited capabilities in terms of capturing long sound phenomena, such as a sound of an accelerating car that takes several seconds. The synthesized output sound has much shorter segments of the same type (the accelerating car sound is nearly periodic sound with increasing pitch and shortening period in time), but the algorithm prefers to chop the segments by switching between gear shifts, the acceleration sound, and the sound of the second car rather than create a long accelerating segment of a realistic duration (see http://www.cs.huji.ac.il/~danix/texsyn/sound or the CD-ROM supplement with this issue). This is the nervous gear-shifting effect that seems to repeat from the previous washing shore example. We didn't show the waveform in this example because we can't see the long-





13 Spectral images of the washing shore sound. (a) The source recording and (b) resynthesis results.





14 Spectral images of the Formula 1 race sounds. There's a limit to how long a structure our algorithm can capture. (a) The original sound contains a long acceleration sound with three gear shifts. (b) The synthesis results make much shorter accelerating segments and sometimes a succession of fast gear shifts.

term increasing pitch structure from the signal, but it's evident in the spectral view.

Future directions

Our results suggest that a principled mathematical approach to sound texture analysis and resynthesis is possible and beneficial, even for mixed periodic and stochastic sounds. This is, to our knowledge, the first approach to sound-texture analysis and resynthesis that doesn't assume an implicit sound model. Moreover, it doesn't require a separate treatment of periodic and stochastic components. There are many open directions for future research related to synthesis of sound textures. For instance, we can use the algorithm described in the "Statistical learning" section to mix two different texture sounds. It would be interesting to explore this possibility.

Also, we can use an extension of our algorithm for additional texture and audio-processing tasks such as classifying sounds. To do this requires first learning the mutual source and then using some statistical similarity measure to estimate how close an unseen example is to a source in a given database of already learned sources.

An important aspect of texture generation in sound is the ability to control the generation process. For instance, it's interesting to explore a relation between possible perceptual features and texture statistics and then ask the machine to generate a sound according to user specifications such as, "More of this sound and less of that sound." For example, users might want to generate a quieter, relaxing, and smoother washing shore or try to generate a storm, ask the thunder to occur at a specific time, make the racing car accelerate for 3 seconds, calm down the baby or annoy it, and so on. These and possible additional extensions are subject to future work. ■

Acknowledgments

We presented a preliminary version of this article at the International Computer Music Conference (ICMC 99). 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

1. D. Gabor, "Acoustical Quanta and the Theory of Hearing," *Natur*e, vol. 159, no. 4044, 1947, pp. 591-594.
2. C. Roads, "Introduction to Granular Synthesis," *Computer Music J.*, vol. 12, no. 2, 1988, pp. 11-13.
3. G. De Poli and A. Piccialli, "Pitch-Synchhronous Granular Synthesis," *Representation of Musical Signals*, G. De Poli, A. Piccialli, and C. Roads, eds., MIT Press, Cambridge, Mass., 1991, pp. 187-219.
4. R. Hoskinson and D. Pai, "Manipulation and Resynthesis with Natural Grains," *Int'l Computer Music Conf.* (ICMC 01), Int'l Computer Music Assoc., San Francisco, 2001, pp. 338-341.
5. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.
6. S. Dubnov, R. El-Yaniv, and G. Assayag, "Universal Classification Applied to Musical Sequences," *Proc. Int'l Computer Music Conf.* (ICMC 98), Int'l Computer Music Assoc., San Francisco, 1998, pp. 248-254.
7. R. El-Yaniv, S. Fine, and N. Tishby, "Agnostic Classification of Markovian Sequences," *Advances in Neural Information Processing Systems*, vol. 10, M. Jordan, M. Kearns, and A. Solla, eds., MIT Press, Cambridge, Mass., 1998, pp. 465-471.
8. I. Daubechies, "Orhtonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, Oct. 1988, pp. 909-996.
9. G.E.P. Box et al., *Time Series Analysis: Forecasting and Control*, Prentice Hall, Upper Saddle River, N.J., 1994.
10. N. Merhav and M. Feder, "Universal Prediction," *IEEE Trans. Information Theory*, vol. 44, no. 6, Oct. 1998, pp. 2124-2147.
11. M. Basseville et al., "Modeling and Estimation of Multiresolution Stochastic Processes," *IEEE Trans. Information Theory*, vol. 38, no. 2, Mar. 1992, pp. 766-784.
12. G.W. Wornell and A.V. Oppenheim, Wavelet-Based Representations for a Class of Self-Similar Signals with Application to Fractal Modulation," *IEEE Trans. Information Theory*, vol. 38, no. 2, Mar. 1992, pp. 785-800.
13. Z. Bar-Joseph et al., "Texture Mixing and Texture Movie Synthesis Using Statistical Learning," *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 2, Apr.–Jun. 2001, pp. 120-135.
14. R.A. DeVore, B. Jawerth, and B. Lucier, "Image Compression through Wavelet Transform Coding," *IEEE Trans. Information Theory*, vol. 38, no. 2, part II, Mar. 1992, pp. 719-746.

Shlomo Dubnov *is a lecturer in the Communication Systems Engineering Department at Ben-Gurion University. He has a PhD in computer science from Hebrew University and a BMus from the Rubin Music Academy in Jerusalem. His research interests include multimedia signal processing, computer music, and machine learning.*

Ziv Bar-Joseph *is a PhD student at the Laboratory for Computer Science at the Massachussets Institute of Technology. He has a BSc and an MSc in computer science from the Hebrew University of Jerusalem. His research interests are in machine learning, computational biology, and distributed computing. The work described in this article was part of his master's thesis.*

Ran El-Yaniv *is a senior lecturer in the Computer Science Department at the Technion–Israel Institute of Technology. His research interests include statistical learning theory, data clustering and compression, and theoretical and empirical analysis of online algorithms. He has a BSc and an MSc in computer science from the Hebrew University of Jerusalem and a PhD in computer science from the University of Toronto.*

Dani Lischinski *is a lecturer in the School of Engineering and Computer Science at the Hebrew University of Jerusalem. He has been working on image-based modeling and rendering, texture synthesis, and use of wavelets to perform fast multiresolution operations on images. He has a BSc and an MSc in computer science from the Hebrew University of Jerusalem and a PhD in computer science from Cornell University.*

Michael Werman *is a professor in the School of Engineering and Computer Science at the Hebrew University of Jerusalem. He has a PhD in computer science from the Hebrew University of Jerusalem.*

Contact Shlomo Dubnov at the Communication Systems Engineering Dept., Ben-Gurion Univ., Beer-Sheva, 84105, Israel, email dubnov@bgumail.bgu.ac.il.