MUSICALLY EXPRESSIVE SOUND TEXTURES FROM GENERALIZED AUDIO

Ben Recht and Brian Whitman

MIT Media Lab
20 Ames Street
Cambridge, MA 02139
{brecht, bwhitman}@media.mit.edu

ABSTRACT

We present a method of musically expressive synthesis-by-analysis that takes advantage of recent advancements in auditory scene analysis and sound separation algorithms. Our model represents incoming audio as a sub-conceptual model using statistical decorrelation techniques that abstract away individual auditory events, leaving only the gross parameters of the sound– the "eigensound" or generalized spectral template. Using these approaches we present various optimization guidelines and musical enhancements, specifically with regards to the beat and temporal nature of the sounds, with an eye towards real-time effects and synthesis. Our model results in completely novel and pleasing sound textures that can be varied with parameter tuning of the "unmixing" weight matrix.

1. INTRODUCTION AND BACKGROUND

There are many ways to generate perceptually lifelike sounds [1] but our main interest is making these textures *musical* – that is, finding analytical approaches that generate musical textures for parameterized synthesis. We discuss below a method to generate interesting sound textures from a large amount of 'source' audio material and ways to control the synthesis process in real-time.

Much of this work was influenced by pure audio analysis techniques; specifically the case of general audio understanding tools or classification, but also the auditory scene analysis and source separation literature. Representing audio as a weighted set of statistical basis functions as in [2] is used in the MPEG-7 standard for coding audio representation and classification.

Other synthesis measures for creating sound textures have used linear prediction [3] to capture both time and frequency information, and also functional iteration synthesis [4] to generate realistic rain and thunderstorm sounds. Other texture generation techniques have used the principal components analysis (PCA) transform [5].

In the image domain, we find many applications of principal components analysis for recognition and synthesis [6]. The analogy of sound textures is prevalent in image and movie coding, specifically to generate realistic non-linear natural image sequences [7].

Here we explore using the residuals of these statistical auditory analysis techniques to generate musically-interesting and controllable textures.

1.1. Statistical Decorrelation Techniques

In general, removing statistical dependence of observations is used in practice to dimensionally reduce the size of datasets while retaining important perceptual features. Many audio analysis systems use these techniques as a pre-processing step prior to classification for understanding, as in [8].

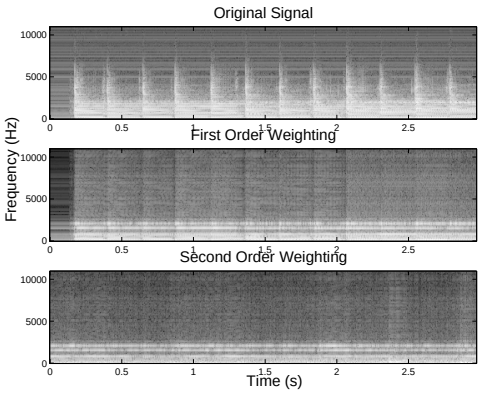

Figure 1: Spectrograms of the stages towards generalized audio. The original signal (drums and guitar for 3 seconds) is first rank-reduced in frequency (the beats are still visible) and then rank-reduced in time (where the final texture is shown.)

1.1.1. PCA

Principal components analysis (PCA) aims to reduce the dimensionality of a data set by only keeping the components of the sample vectors with large variance. By projecting onto these highly varying subspaces, the relevant statistics can be approximated by a smaller dimensional system. This provides efficiency in storage, regression, and estimation as algorithms can take advantage of the statistical compression.

1.1.2. Singular Value Decomposition

The main tool of Principal components analysis is called the *Singular Value Decomposition* (SVD) [9]. The SVD provides both a framework and a convenient algorithm for diagonalizing the sample covariance matrix $\mathbf{C}_x$. Given n data points of length m, we can construct a matrix $\mathbf{A} = [(x_1 - \bar{x})/\sqrt{n} \ldots (x_n - \bar{x})/\sqrt{n}]$. There exist unitary matrices $\mathbf{U}$ of size $m \times m$ and $\mathbf{V}$ of size $n \times n$ such that

$$\mathbf{U}^T \mathbf{A} \mathbf{V} = \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n) \tag{1}$$

with $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$. Using this, we see that

$$\mathbf{C}_x = \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma\mathbf{U}^T = \mathbf{U}\Sigma^2\mathbf{U}^T \tag{2}$$

Figure 2: General overview of our synthesis by analysis system. Time domain audio is converted to the frequency domain and then rank-reduced to a low number of frequency dimensions. Multiple short time chunks of these frequency fingerprints are then lined up and given to another PCA algorithm, which reduces dimensionality along the time axis. The end result is a spectral fingerprint representing "generalized audio," or the observational mean of the audio perception. This resultant data is invertible back to the time domain via inverting both the time and frequency PCA weight matrices. Enhancements to this process (described below) include adaptive dimensionality reduction (for a real-time effect implementation) and beat and tempo locking for synthesis of a generalized texture loop.

and thus computing the SVD of $\mathbf{A}$ is equivalent to diagonalizing the covariance matrix $\mathbf{C}_x$. Furthermore, by computing the SVD, we find the appropriate scheme for dimensionally reducing the samples x_n. Writing $\mathbf{U} = [u_1 \dots u_m]$, we can take the first K singular values and construct the matrix

$$\mathbf{W} = \mathrm{diag}(1/\sigma_1, \dots 1/\sigma_K)[u_1, \dots, u_K]^T \tag{3}$$

Then the data $\mathbf{W}x_k$ will be vectors of length K and each component of the vectors will be uncorrelated with variance 1.

2. APPROACH

Our general approach is illustrated in Figure 2. We start with a source signal $y(t)$ which is usually a few measures long. We first create a spectrogram and perform an initial complex-valued PCA (which creates the first weight matrix $\mathbf{W_1}$) on the spectra along the frequency domain, reducing the b bins of frequency resolution to a statistically-uncorrelated set of rank-reduced bins. This reduced spectra $\mathbf{A}$ is then cut up into uniform-length pieces, each of which are unrolled to a single vector and treated as a column of the new observation matrix $\mathbf{M}$.

We find that best-sounding results on beat-oriented data are achieved when the observations are *registered*. Registration aligns the tempo and start point of each observation. Various techniques can be used to achieve registration, such as beat detection [10] or hand-alignment. We then compute another PCA on $\mathbf{M}$, reducing the time resolution. (This creates our second-order weight matrix $\mathbf{W_2}$.) The $\mathbf{W_1}$ and $\mathbf{W_2}$ saved are then used to resynthesize the reduced $\mathbf{M}$ back to a $y_r(t)$. The result of this process is shown in Figure 1. We note that the $\mathbf{W_1}$ step is not critically necessary in an offline process– it often colors the timbre, and if computational power and memory is available to store the entire rank of frequency rather than the reduced representation, the PCA on the frequency domain can be omitted.

3. MUSICAL IMPLEMENTATION

We next describe the steps we take to make the generalized audio musical and useful as both an effect and a controllable synthesis technique.

3.1. Musical Parameters

If we view this process as a source separation technique, the generalized audio created in the default case is better represented as the optimal mixing (given the constraints and parameters) of the decorrelated spectral activity. To make an evolving musical texture expressive, we can modulate the mixing parameters of the generalized audio using a multi-controller mixer paradigm. Each 'optimal channel' controller affects the relative strength of one independent audio feature over the time period of the texture. There can be as many channel controls as there is rank in the W_1 and W_2 weight matrix. In a loop-based resynthesis situation, there can be envelopes placed on the weighting matrices of both frequency and time.

3.2. Time and Beat Matching for Tempo-Locked Effects

In an offline "model-building" situation we have found success with aligning each example sound to a common beat template. For example, given n source sounds (or n subdivisions of a single source sound) with heavy beat content, we choose a source beat clock sound $n*$ through some heuristic manner (beat strength, periodicity, or complexity.) We choose a period length of this beat clock and all other sounds are then warped to this source using a dynamic time warping algorithm (DTW) [12] which uses the the magnitude of the PCA-down-weighted frequency as its examples and a gaussian cost function

$$V(x_1, y_2) = e^{-\frac{(|x_1 - y_2|)^2}{\sigma^2}}, \sigma = 0.5 \tag{4}$$

to find the best fitting path of an example sound n in the tempo space of $n*$.

Using the above cost function, dynamic time warping finds an optimal warping path $w(k)$ through FFT space by solving the dynamic program

$$\begin{aligned} g_1(x_1, y_1) &= V(x_1, y_1)w(1) \\ g_k(x_k, y_k) &= \min_{x_k, y_k} g_{k-1}(x_{k-1}, y_{k-1}) + V(x_k, y_k)w(k) \end{aligned} \tag{5}$$

Here x_k is a particular FFT frame of the example sound and y_k is a frame of the tempo master. The resulting series x_k is a time series for the playback of the frames of n so that they are tempo locked to $n*$.

The result is a set of same-length examples of audio, all with roughly the same beat structure (as much as would be reflected in the spectral magnitudes.) We then iterate through each period length of $n*$ in this manner for a user-specified amount of time, creating an evolving structure that retains the tempo information of $n*$ but whose frequency and time envelopes are generalized from the other sound examples. This process is implemented in Eigenradio [11], an on-line art project that synthesizes a constant stream of generalized audio from multiple real-time radio sources. See Figure 3 for a diagram.

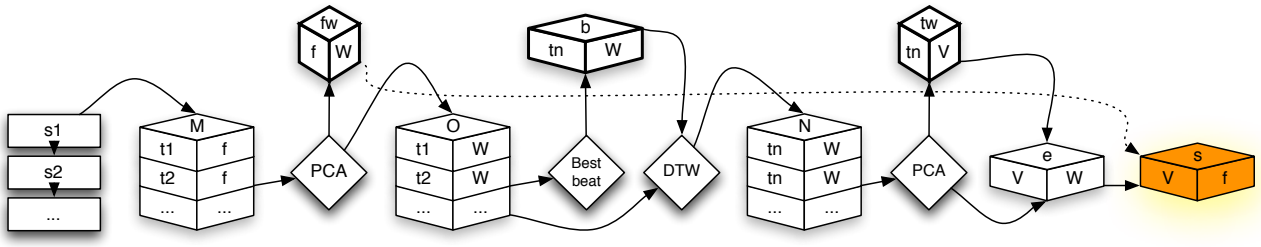

Figure 3: Offline generalized audio synthesis process used in the Eigenradio[11] system to create a constant stream of sound texture from multiple real-time sources. Radio data from multiple stations is segmented into songs. The generalized audio process then creates the W_1 (here the f by f_w matrix) and time stretches equally-sized periods of example songs to a common time base (obtained from a beat detection component) using a dynamic time warping algorithm (here labeled DTW) before creating the W_2 (the t_n by t_w matrix) and resynthesizing a single song from the many examples.

3.3. Considerations for a Real-Time Implementation

We also consider work in two real-time scenarios– effect and synthesis. In the synthesis case, a model is trained offline of a long sample, broken into the necessary subdivisions. The regenerated audio from the model is looped as a single generalized subdivision (possibly beat-locked to the host's tempo.) In this case an offline DTW beat registration step is helpful to align time information in the loop.

The effect scenario uses adaptive methods of dimensionality reduction to synthesize generalized audio from real-time sources. We initialize randomly our W_1 and W_2 and buffer incoming audio into the chunk size necessary for the time envelope generalization. As the required amount of data comes into to the buffers, we update our weight models iteratively, which requires far less processing time than solving the system of equations at once.

There are several iterative update rules for computing an approximate PCA factorization. An attractive method is an adaptation of Seung and Lee's Non-negative Matrix Factorization algorithm [13] for general PCA. This algorithm finds a local minimizer of component-wise norm

$$\|V - WT\| \tag{6}$$

where V is a given $n \times m$ matrix and W and T are matrices of size $n \times r$ and $r \times m$ respectively. Such a factorization can be found using the update rules

$$\begin{aligned} T_{kj}^{new} &= T_{kj} \frac{(W^\top V)_{kj}}{(W^\top WT)_{kj}} \\ W_{ik}^{new} &= W_{ik} \frac{(VW^\top)_{ik}}{(WTT^\top)_{ik}} \end{aligned} \tag{7}$$

If the matrix V is positive, the iterative algorithm returns only positive W and T matrices. In our case, V is a general complex-valued spectrogram and we cannot restrict the W and T to be positive. Instead, the iterations returns a proxy for a rank reduced model of the frequency space W and time space T by forcing W and T to have the same component-wise norm. In real time, the matrix V will be constantly changing with the input stream and hence W and T will never settle to a final answer, but will be pulled to a general center of frequency/time down-weightings.

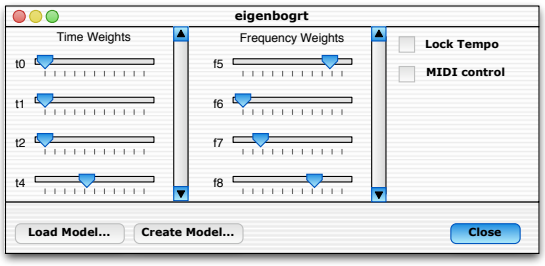

Figure 4: Mockup of real-time synthesis interface from a trained model. Users can adjust the weighting of the various time and frequency components from the source audio as the loop plays. MIDI control can LFO or apply envelopes to the parameters.

In both real-time effects models an interface allows the user to tweak the mixing parameters of the frequency and time resynthesis. In future implementations these parameters can be controlled by ADSR envelopes or LFOs for musical control locked to master tempo. See Figure 4 for a potential user interface to the real-time synthesis system.

4. CONCLUSIONS

We presented a model of musical synthesis-by-analysis based on auditory scene and source separation research that allows for expressive generation of sound textures. We are currently extending our model to the real-time implementation and are investigating user interface solutions for controlling the mixing parameters of the resynthesized textures.

5. ACKNOWLEDGMENTS

Thanks to Youngmoo Kim for his helpful contributions and our anonymous reviewers for their insightful comments.

6. REFERENCES

[1] Nicolas Saint-Arnaud and Kris Popat, "Analysis and synthesis of sound textures," in *AJCAI workshop on Computational Auditory Scene Analysis*, 1995.

[2] Michael Casey, "General sound recognition and similarity tools," in *MPEG-7 Audio Workshop W-6 at the AES 110th Convention*, May 2001.

[3] Marios Athineos and Daniel P.W. Ellis, "Sound texture modelling with linear prediction in both time and frequency domains," in *In Proceedings of ICASSP-2003*, 2003.

[4] Agostino Di Scipio, "Synthesis of environmental sound textures by iterated nonlinear functions," in *Proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects (DAFx99)*, 1999.

[5] J. Stapleton and S. C. Bass, "Synthesis of musical tones based on the karhunen-loeve transform," *IEEE Trans. Acoust., Speech, Signal Processing*, pp. 305–319, 1988.

[6] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3(1), pp. 71–86, 1991.

[7] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Texture mixing and texture movie synthesis using statistical learning," *IEEE Trans. Visualization and Comp. Graphics*, vol. 7(2), pp. 120–135, 2001.

[8] Brian Whitman and Paris Smaragdis, "Combining musical and cultural features for intelligent style detection," in *Proc. Int. Symposium on Music Inform. Retriev. (ISMIR)*, October 2002, pp. 47–52.

[9] C.F. Van Loan G.H. Golub, *Matrix Computations*, Johns Hopkins University Press, 1993.

[10] Eric Scheirer, "Tempo and beat analysis of acoustic musical signals," *Journal of the Acoustical Society of America*, , no. 50, pp. 588–601, 1998.

[11] Brian Whitman, "Eigenradio – the top 20 singular values all day every day – http://eigenradio.media.mit.edu/," Web-based artwork, 2003.

[12] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, , no. ASSP-26(1), pp. 43–49, February 1978.

[13] Daniel D. Lee and H. Sebastian Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2000, pp. 556–562.