Reforming Sounds with Granular Synthesis

submitted to AIIDE13

Judy A Franklin
Computer Science Department
Smith College
Northampton, MA 01063

Abstract

We describe an audio granular synthesis generator with controllers that can be accessed by reinforcement learning agents. The generated sound is analyzed, based on spectral goals, to produce a value called the reinforcement. The reinforcement value is used to adjust the agents' future behavior. Our current focus is in generating a soundwave that has similarity to an instrumental music recording. There are aural feature generation and interactive real-time applications in game sound.

Granular Synthesis Control Overview

We research combining machine learning algorithms with granular synthesis, the building of sound objects using brief microacoustic events called grains of sound. A grain of sound may be a pure sine wave passed through an amplitude (volume) envelope, lasting barely as long as the threshold of human auditory perception (Roads 2001). Sound may be synthesized and shaped from grains by varying the density of grains per second, varying the frequency range of the sine waves, oscillating a non-sine waveform to generate each grain, and varying the durations of individual grains, etc. Parameters may be time-varying and can be means in a gaussian distribution or some other kind of random distribution.

We have implemented a granular synthesis engine, spectral analyzers, feature extractors, and reinforcement learning (RL) agents that change parameters in the granular synthesis engine. The overall system diagram is shown in Figure 1. As a result of the changes in the granular synthesis (grain) controllers, the spectral features of the grains change. The features are used to determine the state of the learning system (which agent should be active next), and the reinforcement value.

Our goal is to generate sound waves that are recognizable from the original sound wave, but clearly generated by many grains of sound. In our first experiments, RL agents learned to control a grain stream with a constant spectral spread and centroid (citation in final paper). The more complex Mel Frequency Cepstral Coefficients (MFCCs) are used in this paper's research. First RL agents learn to control a grain stream that produces a constant set of MFCCs. Then, in order to produce a soundscape or soundwave, the RL agents must control a grain stream that produces a sequence

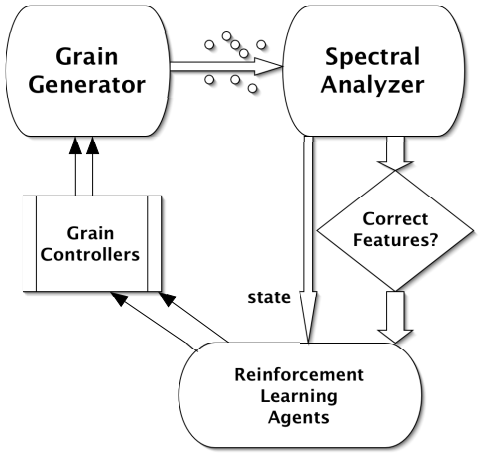

Figure 1: The overall system showing how the synthesis engine, feature extractors, learning agents, and interface work at the top level.

of MFCCs. These are compared to the MFCCs of the original sound wave to produce a reinforcement value.

The Granular Synthesizer

Working within a graphical music programming environment called Pure Data (pd) (Puckette 2005), we created a granular synthesizer program, called GranWave, that uses wavetable oscillators to generate sound grains. Following Road's description, the grain density can be varied, as can other parameters such as duration and frequency of oscillation of the wavetable. Gaussian random variation can be added, for example between 5 and 15 msec durations. There is a choice of wavetable, depending on desired harmonics, and stereo panning. Figure 2 shows a stream that is a variety of grains, with different frequencies, amplitudes, and durations. A common practice is to make duration proportional to frequency as can be seen here. Higher frequency grains are darker and last longer.

Generating a Standing Wave with MFCCs

In a previous publication, we note that the spread and centroid measurements are functions of the linear FFT bins, and

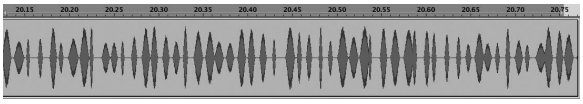

Figure 2: A small portion (.6 sec) of a wave generated by the granular synthesis engine.

do not correspond well to what we hear as the subjective pitch spread. At the time, we planned to pursue using a "constant Q" transform (Brown 1991) that makes sense with respect to pitch rather than frequency.

Since then, we discovered that the mel-frequency scale provides frequency values for pitch as perceived by the human auditory system for speech. The transform based on this scale, that produces the mel-frequency cepstral coefficients (MFCCs) has yielded good results for music as well (Logan 2000) is more efficient to compute than the Q-transform. The first experiments using MFCCs are in using reinforcement learning to match one set of goal MFCCs by changing the max and min frequency values of the grains generated by the granular synthesis generator. These frequency values are not tied to duration, which is a fixed value.

The goal MFCCs correspond to one window sample's worth of a single (standing) waveform with several harmonics as shown in Figure 3. We wrote the reinforcement learning (RL) algorithm in C, modifying code from earlier work using the C interface code provided for pd, and described by Zmölnig (Zmölnig 2011) and the MFCC code described by Sakari Tervo and Jukka Pätynen (Tervo and Pätynen 2010) The RL algorithm is the Sarsa(λ) reinforcement learning algorithm (Sutton and Barto 1998); the pd object is called GRControl. Its pseudocode is in the Appendix. Using this configuration, GRControl successfully learned to generate a grain stream that has the goal MFCC spectral characteristics.

By definition, there are many parameters that can be varied by the actions of GRControl. In these experiments, some parameters are set empirically (we eventually fix speed at 16msec). Frequency and duration changed via actions taken by GRControl. These actions move horizontal sliders that are pd objects that provide discrete values in a given range. The slider values are sent to the grain generator.

The number for MFCCs is 51 per analysis window. Initially, the statespace is a 100 by 100 statespace, giving every dual slider position a unique state and a unique corresponding reinforcement learning agent.

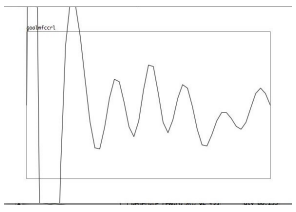

Figure 3: Goal MFCCs, shown graphically, for the standing wave experiment.

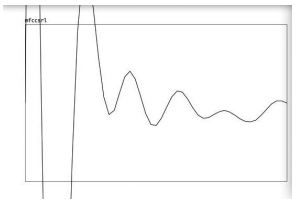

Figure 4: Actual MFCCs, shown graphically, after reward criteria were met.

Reinforcement Criterion

For each window of sound samples, let n = number of mel-frequency cepstrum coefficients.
Let A be the set of actual coefficients at each time step, and G be the set of goal coefficients.
Let $x_i = sign(G_i - A_i)$ for $i = 0, 1, \cdots, n-1$.
Let $y_i = |G_i - A_i|$
and finally, to see qualitative behavior, let

$$X = \Sigma_0^{n-1} x_i \tag{1}$$

and

$$Y = \Sigma_0^{n-1} y_i \tag{2}$$

Then we compute R as:

$$R = \begin{cases} 1, & \text{if } Y <= Y_{thresh} \text{ and } X <= X_{thresh} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

For Y_{thresh} = 20 and $X_{thresh} = 10$ we show in Figure 4 a graph of a snapshot of the 51 actual MFCCs. They are shown as a graph, of the 51 distinct values; i.e. this is not a sound wave. Figure 3 shows the graph of the goal MFCCs, which are static for the standing wave goal.

Discussion

We have determined that the RL agents can learn to control real-time granular sound generation for static sound waves. Controlling a fixed centroid and spread was a straightforward reinforcement learning task. Making the reinforcement criterion a function of the MFCCs of a fixed soundwave made it a more difficult machine learning problem. The conversion of the aliased statespace to a non-ambiguaous one enabled the achievement of matching MFCCs of a goal and generating standing wave of sound grains, although not perfectly. The next milestone is in learning to produce sequences of MFCCs through generation of soundscapes. We report on experiments in this direction next, that include further interpretation of the MFCCs.

Generating Dynamically changing MFCCs

We report on experiments in using RL to produce a complex wave-form, by matching MFCCs. We chose a waveform of 1.5 seconds from a Stanley Clarke et al. (Clarke, Miller, and Wooten 2008) song (*Thunder* - see Figure 5). We chose the goal waveform because it is a real-world example, it is complex, and is enjoyable to hear. Our goal in these experiments is to design an RL-based controller that can run in pure data, a reinforcement function (of the MFCCs), and control interfaces that generate a 1.5 second waveform that is similar to the waveform shown in the figure. The similarity is determined by some function of the difference between the set of MFCCs that represent this waveform, called the goal MFCCs, and the MFCCs that are generated during each 1.5 second episode of waveform generation via granular synthesis. 1.5 seconds is 75 windows of 1024 samples each, at a sampling rate of 44.1k samples per second. Each set of MFCCs is the output of analysis of one of these windows.

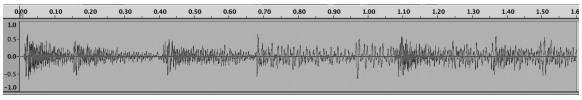

Figure 5: Sound Excerpt from Stanley Clarke et al.'s (Clarke, Miller, and Wooten 2008) Thunder.

The More Complex Controller

The structure of the new GRControl patch containing the RL agents is a three-dimensional state-space of Sarsa(λ) reinforcement learning agents, with 17 actions each, as shown in Figure 6. The state-space consists of the current time-step's frequency in one dimension, and the previous time-step's frequency in the second dimension, and both current grain amplitude and current grain duration in the third dimension. The actions make changes to frequency, grain duration, and grain amplitude volume. We note that the use of one dimension of the state space to share two variables is not ideal. However, our PD indexing scheme prevented a straightforward four-dimensional implementation.

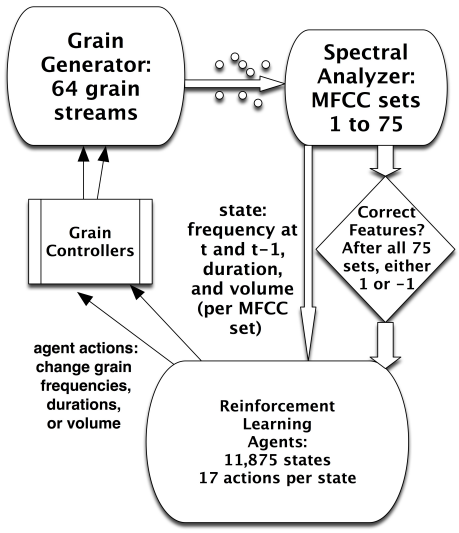

Figure 6: A detailed diagram showing the dynamic sound wave generation with 75 sets of MFCCs.

Using MFCCs in the Reinforcement Function

The reinforcement criterion is modified to contain the structure of an episode. One episode is one 1.5 seconds' worth of sound generation, equivalent to one "pass" through the goal wave. Reinforcement Learning, with its delayed reward capability, can be used in a setting where the reinforcement value is given at the end of an episode. Each agent's eligibility trace, a time-decaying measure of its activity, enables it to receive part of the reward if it is due.

The modified reinforcement function was defined as:

$$R = \begin{cases} 0, & \text{if not at end of episode} \\ 1, & \text{if } \Sigma\hat{Y_k} <= Y_{thresh} \text{ and } \Sigma\hat{X_k} <= X_{thresh} \\ -1, & \text{otherwise.} \end{cases} \quad (4)$$

where $k = 1, 2, \cdots, 75$ over the 75 episodes, and similarly to equations (1) and (2)

$$\hat{X_k} = \Sigma_0^{n_p - 1} x_i \quad (5)$$

and

$$\hat{Y_k} = \Sigma_0^{n_q - 1} y_i \quad (6)$$

where $n_p <= 51$ and $n_q <= 51$ can be set as parameters to the reinforcement analysis function. The use of episodes in this way resembles some of the original pole-balancing research in the early days of reinforcement learning (Barto, Sutton, and Anderson 1983).

First Results

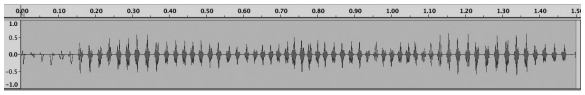

Figure 7: A generated waveform that matched criteria $Y_{thresh} = 10$, $X_{thresh} = 2$ (up to 2 coefficient signs could be different), and 8 coefficients compared (no normalization). The speed was 6. One can see the distinct nature of the grains in this waveform, and its dissimilarity to the goal waveform of Figure 5.

Experiments involves setting RL-action output ranges as well as making changes in the grain engine due to various discoveries. The output of GRControl changes frequency and duration. The duration ranges between 3 and 48msec with GRControl output changes of ± 6, ± 3, or 0. In one experiment, the Speed was set to 6 with duration varying between 20 and 28, with resulting waveforms such as the one in Figure 7 that are quite dissimilar to the goal. This was however one of our early successes in that the RL agents were able to learn to control the grain stream and match a set of criteria. The graph of Reinforcement (R) values is shown in Figure 8. The total number of negative R vs. positive R values is almost equal. Furthermore, as the number of episodes increases, shown on the vertical axis, the number of R=1 values increases. We include these interim results in part to give a glimpse of the difficulty of this task, but also to provide information for the next step, understanding more about MFCCs.

Using Timbre and Normalized MFCCs

Early in this set of experiments, we used the above criterion, keeping X_{thresh} and Y_{thresh} the same, and varying how many of the 51 coefficients to use. Of the 51 MFCCs coefficients total, the first one is always 0. The second coefficient contains a measurement of the power (amplitude level) of the window of 1024 samples. We dropped that value

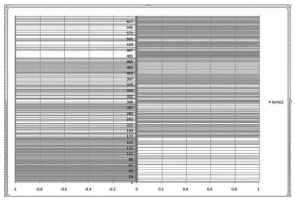

Figure 8: Episodes for R = -1: 321 R = -1, for R = 1: 328. Note the shift from R = -1 on the left, to R=1, on the right, as the episode number increases along the Y axis.

since the original signal could have a power level that the grain generator could not produce easily and it is the relative amplitude levels that are important. Noting that the first half of a set of MFCCs represent timbre information, and the second half, pitch, we decided to employ the first half only. At the 44.1k sampling rate, 1024 samples is 23ms, and matching the timbre measurements should be sufficient. These decisions are based on various readings such as Tervo and Pätynen (Tervo and Pätynen 2010), Brent, (Brent 2010), and Loughran et al. (Loughran et al. 2008). This leaves 24 coefficients to use.

It is also clear that the threshold Y_{thresh} in Equation 6 depends on the amplitude of the soundwave rather than the relative values of the Mel-Frequency Cepstral Coefficients themselves. And the measure is too coarse to be of use as a reinforcement calculation. We ran some experiments in normalizing using the power coefficient (the first non-zero MFCC) to normalize. But found the best success in using the coefficient values themselves, both goal and actual:

$$y_i^n = \frac{y_i}{\Sigma_0^{n_q-1} y_i} \tag{7}$$

The resulting reinforcement function is defined as:

$$R = \begin{cases} 0, & \text{if not at end of episode} \\ 1, & \text{if } \hat{Y_k^n} <= Y_{thresh} \text{ and } \hat{X_k} <= X_{thresh} \\ & \text{for at least T episodes} \\ -1, & \text{otherwise.} \end{cases} \tag{8}$$

where $k = 1, 2, \cdots, 75$ over the 75 episodes, and for the sum of normalized coefficient magnitudes,

$$\hat{Y_k^n} = \Sigma_0^{n_q-1} y_i^n \tag{9}$$

One of the most successful set of reinforcement criteria values are $n_p = n_q = 8$, and $Y_{thresh} = 2, X_{thresh} = 2$ and T=8. After more experimentation with duration and speed, we determined the best configuration to be duration ranges between 10 and 54msec with GRControl output changes of ± 6, ± 3, or 0. and speed to be a constant 16 msec between grains.

Adding Domain Knowledge

During this experimentation, the maximum number of coefficients that matched per episode is recorded. The frequency slider has range 86.133 to 1076.65 Hz as before. The reinforcement learning controller frequency change output is

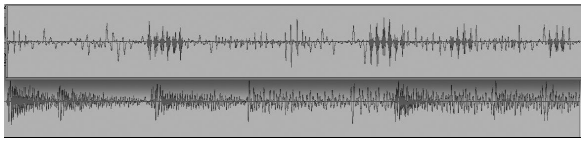

Figure 9: Comparison of a matching wave, with the lower frequency range in place, fixed speed of 16, and the reinforcement criteria parameters 2,2, and 8.

integer increments of $\pm 10, \pm 5, \pm 2, \pm 1$, and 0. A single +1 move adds 43.066 to the freq. But it adds high frequencies as well according to the harmonics in the wave that composes each grain. It became obvious that some domain knowlege is

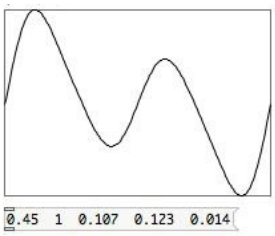

Figure 10: The wavetable with several harmonics that is oscillated in each grain. The harmonics were determined by a Fourier Analysis of a small (.05 second) window of the goal waveform.

necessary to improve the performance of this system (after we ran experiments with several wavetables). In this case, that means knowing that a bass instrument generated the original soundwave. A simple adjustment was made by halving the range of the frequency slider; the slider now ranges from 43.066 to 538.325. Figure 9 shows a grainstream-generated waveform compared to the goal.

Finally, a wavetable was constructed that is an approximation of the first several harmonics of the bass guitar, as determined by running a Fourier Analysis on a very small window of the goal soundwave. It is important to make a distinction here between using this harmonic structure and using an actual piece of the sound wave itself. The resulting wavetable with the harmonic weights is shown in Figure 10.

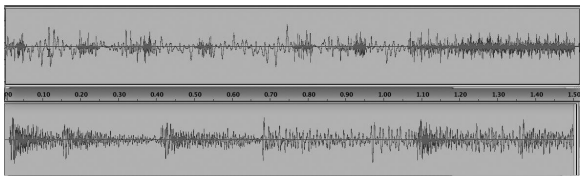

Figure 11: Comparison of a matching wave; speed 16; parameters 2,2, and 8; lower frequency range; and using Bass harmonics Wavetable, shown with Thunder soundwave.

Figure 11 is a grain engine-generated soundwave that received a value of R=1, shown with the goal sound wave. At least 8 MFCCs sets must match per episode for R = 1.

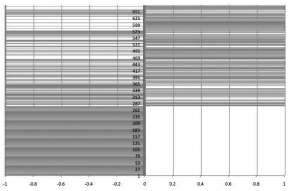

Figure 12: R values over 670 episodes, frequency range halved, speed of 16. 298 R = -1 values, 372 R = 1 values, and shift toward R = 1 as episode number increases. Bass harmonics wavetable used, and lower frequency range used in sliders.

During this experiment, the maximum number of sets of coefficients matched was 22 (out of 75) in one episode. Over the whole experiment, the total number of R = -1 is 298, and the total R =1 is 372. Figure 12 shows the dramatic shift in R values from R = -1 to R =1 over the course of the 50,000 iterations or 670 episodes. Recall that an episode is 75 iterations and at its end, either R = -1 or R = 1 is assigned as reward.

Discussion

The system as is stands can reliably match up to one-third of the MFCCs of a soundwave in a single episode. This is an achievement that can lead to even more interesting results. It is note-worthy that the fairly simple Sarsa(λ) algorithm can learn to generate an action path through time. In previous work the author has used a machine learning algorithm called LSTM (Gers, Schmidhuber, and Cummins 2000) that can predict and reiterate long sequences, for use in music re-generation and new generation. The LSTM algorithm may be combined with the RL agents to produce a machine that can better produce grain streams that have better matching sequences of MFCCs. The GRControl object can be instantiated any number of times and longer sequences may be produced just by multiple copies over time. It can also provide the starting point for implementations of hierarchical RL structures such as MOSAIC (Samejima, Doya, and Kawato 2003).

Two other areas to explore are the Bark scale, with its possible improvement over the Mel Scale (Brent 2011), and to match a longer soundwave. Toward this goal, we plan to continue our experiments in topic models (Li, Wang, and McCallum 2006) research applied to spectral analysis to provide parameters to the granular synthesis engine. Topics could be used to guide what limits on grain sliders whould be, what parameters should be chosen for MFCCs, or what the controllers change (grain duration, frequencies, etc.).

Why is Reforming Useful for Games?

Regenerating game music as granular synthesized waves can provide variations to match implied moods for games, and hints of memories of previous states/encounters. It can be done in real-time by recording a player's voice and regenerating it for fun at a later time in the game; or by regenerating some of a player's own music for the player's purpose.

Acknowledgments. This material is based on work supported by the National Science Foundation under grants IIS-0222541 and IIS-0914988 and by Smith College. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. The author acknowledges the excellence of the John Payne Music Center in Brookline, MA.

Appendix: Using the Sarsa(λ) algorithm

Figure 13 shows the pseudo-code for the tabular Sarsa(λ) algorithm. The algorithm(Sutton and Barto 1998) associates with each agent some number of actions. An agent contains two sets of numerical values called Q-values and eligibility traces, respectively, one of each per possible action. Each agent uses its set of Q-values to choose an action when the system is in its state. The eligibility trace $e_t(s, a)$ is used to

```
1 Initialize Q(s, a) arbitrarily and e(s, a)=0, for all s, a
2 Repeat (for each episode):
3     Initialize s, a
4     Repeat (for each step of episode):
5         Take action a, observe r, s'
6         Choose a' from s' using Q-policy (ε-greedy)
7         δ ← r + γQ(s', a') − Q(s, a)
8         e(s, a) ← e(s, a) + 1
9         For all s, a:
10            Q(s, a) ← Q(s, a) + αδe(s, a)
11            e(s, a) ← γλe(s, a)
12        s ← s'; a ← a'
13    until s is terminal
```

Figure 13: Pseudo-code for Sarsa(λ) algorithm.

update the Q-value according to how instrumental the associated action was in acquiring the most recent reinforcement. We refer the reader to the Sutton and Barto text (Sutton and Barto 1998) for more detail and theory.

References

Barto, A. G.; Sutton, R. S.; and Anderson, C. W. 1983. Neuronlike adaptive element that can solve difficult learning control problems. In IEEE Transactions on Systems, Man, and Cybernetics *SMC-13:834–846.* IEEE Press.

Brent, W. 2010. A timbre analysis and classification toolkit for pure data. In J. Audio Eng. Soc. *30(6):396–406.*

Brent, W. 2011. Perceptually based pitch scales in cepstral techniques for percussive timbre identification. In Department of Music and Center for Research in Computing and the Arts, UCSD.

Brown, J. C. 1991. Calculation of a constant q spectral transform. In J. Acoust. Soc. Am. *89(1):425–434*.

Clarke, S.; Miller, M.; and Wooten, V. 2008. Thunder. In *Thunder*. ASIN: B001DOQUHO: Heads Up.

Gers, F. A.; Schmidhuber, J.; and Cummins, F. 2000. Learning to forget: Continual prediction with lstm. In Neural Computation *12(10): 2451-2471*. MIT Press.

Li, W.; Wang, X.; and McCallum, A. 2006. A continuous-time model of topic co-occurrence trends. In *Proceedings of AAAI Workshop on Event Detection*. AAAI.

Logan, B. 2000. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the International Symposium on Music Information Retrieval*.

Loughran, R.; Walker, J.; ONeill, M.; and OFarrell, M. 2008. The use of mel-frequency cepstral coefficients in musical instrument identification. In *Ann Arbor, MI: MPublishing, University of Michigan Library*.

Puckette, M. 2005. Pure data (pd). In *Web URL*. http://www-crca.ucsd.edu/ msp/software.html.

Roads, C. 2001. *Microsound*. Cambridge MA: MIT Press.

Samejima, K.; Doya, K.; and Kawato, M. 2003. Inter-module credit assignment in modular reinforcement learning. In *Neural Networks*. Elsevier. 985–994.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning*. Cambridge MA: MIT Press.

Tervo, S., and Pätynen, J. 2010. Tutorial and examples on pure data externals: Real-time audio signal processing and analysis. Department of Media Technology, Aalto University School of Science and Technology: http://www.tml.tkk.fi/tervos/.

Zmölnig, J. M. 2011. How to write an external for puredata. institute of electronic music and acoustics, graz: http://iem.at/pd/externals-HOWTO/pd-externals-HOWTO.pdf.