Generating Soundwaves via Granular Synthesis and Reinforcement Learning*

Judy A Franklin

Computer Science Department
Smith College

Abstract. *We describe an audio granular synthesis generator with controllers that can be accessed by reinforcement learning agents. The movement of the controllers affects the sound, which is analyzed to produce a vallue called the reinforcement. The analysis is based on spectral goals and the reinforcement value is used to adjust the agents. Experiments are described using spectral features that are the spread and centroid, as well as the Mel-Frequency Cepstral Coefficients of the sound. We extend this work to include the complex task of generating a soundwave to match an instrumental recording. We have generated soundwaves that match criteria for reinforcement and gained insight in using MFCCs.*

Keywords: Granular Synthesis, Reinforcement Learning, MFCCs

1 Introduction

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 [16]. Sound may be synthesized and shaped from grains by varying the density of grains per second, the frequency range of the sine waves, the waveform oscillated to generate each grain, and 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. These elements aredescribed in this paper, along with experiments in controlling spectral features of streams of grains. 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.

We describe three sets of increasingly complex experiments and distinguish them by spectral goals. First, the RL agents must produce a grain stream with

* submitted to MCM2013

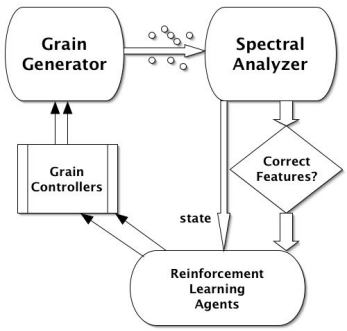

Fig. 1. The overall system showing how the synthesis engine, feature extractors, learning agents, and interface work at the top level.

a constant spread and centroid. Second, the RL agents must generate a grain stream that has a constant set of Mel Frequency Cepstral Coefficients (MFCCs). Third, the RL agents must produce a grain stream that produces a sequence of 75 sets of MFCCs.

2 The Granular Synthesizer

Working within a graphical music programming environment called Pure Data (PD) [15], we created a granular synthesizer program (patch), called GranWave, that uses wavetable oscillators to generate sound grains. As shown in the interface in Figure 2, to start grain generation, the user clicks grain.trigger, which generates a "generation rhythm" with a period of "speed" ms, and some randomness set by the percentage of desired deviation (%deviation). A decrease in the grain stream speed is a decrease in the amount of time between grains, as measured in milliseconds. So it is actually an increase in the number of generated grains.

grain.trigger triggers the patches (subprograms) that set the other grain parameters, through sent messages: mkDur, mkFreq, mkWave, and mkPan. mkDur triggers the actual grain duration, with some random variation. The default values of the range shown in Figure 2 are 5 msec minimum and 15 msec maximum duration. The chosen grain wavetable defaults to 1, the pure sinusoid. The mkWave and mkPan messages trigger the choice of wavetable (1,2, or 3 depending on desired harmonics), and stereo panning. Panning is set by default to have a spread of 45, with random variation enabled.

In the initial implementation (Figure 2), mkFreq triggers randomrange. Randomrange receives as input the values of the two horizontal sliders as the bounds on the range of frequencies within which grains will be generated. The top slider sets the minimum range value, and the bottom slider sets the maximum range value. These two sliders are the grain controllers shown in Figure 1. Referring again to Figure 2, the grains are produced in the nqpoly4 pd patch, bottom

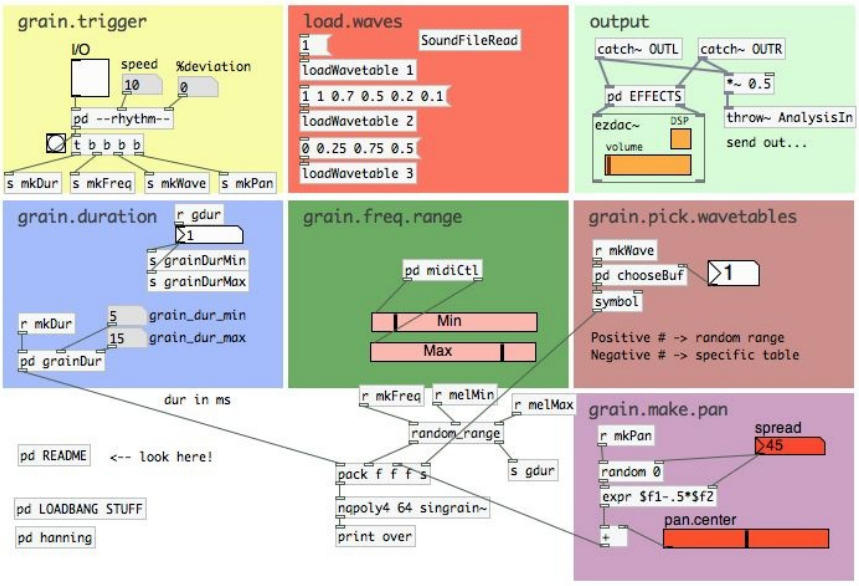

Fig. 2. The PD file that holds the granular synthesis engine and the mechanisms that send audio signals to the analyzer, and the slider controllers that are controlled by reinforcement learning.

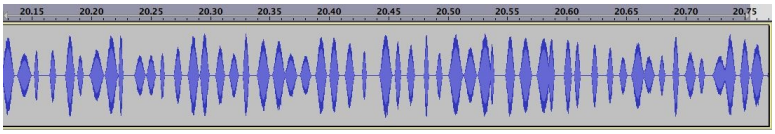

Fig. 3. A small portion (.6 sec) of a wave generated by the granular synthesis engine.

center. This patch, via 64 instantiations of the singrain~ patch, which calls the wavetable oscillator, throws the stereo OUTL~ and OUTR~ signals of the grain stream and these are sent to a digital to analog converter.

Figure 3 shows a short portion of sound generated by the granular synthesis engine. The grains in the figure vary in density (number of grains per second), duration (some are wider than others), and amplitudes.

3 Spectral Centroids and Spreads

In our early, first set of experiments, one of our evaluation criteria was a function of spectral centroids and spreads (bandwidths) of windows of grain streams. The Fast Fourier Transform (FFT) is taken of each window. The spectral centroid is an amplitude weighted average of the 256 frequency bins, that corresponds to the preceptual brightness of the sound [2].

The number of states is set to 200, corresponding to 100 divisions per slider, and setting the min and max of the frequency range as discussed in Section 2. The reinforcement learning algorithm that we used would work best if we created a 100x100 square state space since the position of one slider affects how the other slider alters the sound output. When we originally began this work, the 10,000-state size taxed existing computing systems, in real-time output. We decided to use a state space that was just the two slider spaces juxtaposed, whence the 200 states, 100 per horizontal slider. Each agent had the same finite set of actions 0,1,2,3,4, 5, 6 and 7, with actions 0 through 6 corresponding to sliding the controller -10,-5, -1, 0, +1, +5 and +10 spaces.

Action 7 changed a high level 2-valued state that switched which slider affected the grains' frequency value. The RL controller did not receive the high-level slider state as input. This configuration introduced perceptual aliasing [6] and made the task more difficult as the reward from action selection was not consistent. The effect of a single slider's action depended on the position of the other slider as well, and this was not available to the agent that was changing the slider position.

Each of the 200 states corresponded to a 86.133Hz wide subband of the frequency spectrum. Consequently if an agent action was +1, then it moved the horizontal slider (and so its own state) up to the next frequency bin (adding on 86.133Hz). If the action was -5, the horizontal slider moved -430.665Hz.

The Sarsa [17] reinforcement learning algorithm was implemented in a patch called GRControl . Its pseudocode is in the Appendix. Using this configuration, GRControl successfully learned to generate a grain stream that had both a fixed spread and centrum. For more detail see [14].

4 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 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 [5] 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 has yielded good results for music as well [9], and 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 in the granular synthesis generator. The goal MFCCs correspond to one window sample's worth of a single (standing) waveform with several harmonics as shown in Figure 4. We wrote the reinforcement learning (RL) algorithm in C, modifying code from earlier work [13, 14], using the C interface code provided for PD, and described by Zmölnig [19] and the MFCC code described by Sakari Tervo and Jukka Pätynen [18] The RL algorithm is the same Sarsa(λ) reinforcement learning algorithm. The number for MFCCs is 51 per analysis

window. With increased computing power, we were able to experiment with a

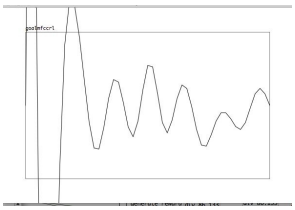

Fig. 4. Goal MFCCs, shown graphically, for the standing wave experiment.

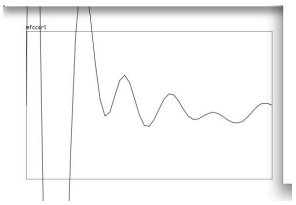

Fig. 5. Actual MFCCs, shown graphically, after reward criteria were met, to match goal MFCCs in Figure 4.

100 by 100 statespace, giving every dual slider position a unique state and a unique corresponding reinforcement learning agent.

4.1 A New 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 5 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 4 shows the graph of the goal MFCCs, which are static for the standing wave goal.

4.2 Discussion

We have determined that the RL agents can learn to control real-time granular sound generation. Controlling a fixed centroid and spread was a straightforward

reinforcement learning task, even with perceptual aliasing. 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.

5 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. [7] song (*Thunder* - see Figure 6). This section presents the process of understanding how to structure the system: the process of varying parameters in the granular synthesizer, the design of the three-dimensional state-space of the Reinforcement Learning controller, and the choice of wavetable that was oscillated by each grain.

5.1 The More Complex Controller

The structure of the new GRControl patch containing the RL agents was a three-dimensional state-space of Sarsa(lambda) reinforcement learning agents, with 17 actions each. The state-space consisted 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 one dimension of the state space to share two variables is not ideal. However, our PD indexing scheme prevented a straightforward four-dimensional implementation.

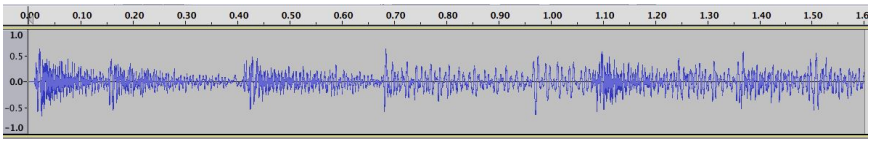

Fig. 6. Sound Excerpt from Stanley Clarke et al.'s [7] Thunder.

5.2 Using MFCCs in the Reinforcement Function

We chose the goal waveform because it is complex, and enjoyable to hear. Our goal in these experiments was to design an RL-based controller, reinforcement

function (of the MFCCs), and control interfaces that generated a 1.5 second waveform that is similar to the waveform shown in Figure 6. 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.

The reinforcement criterion was 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:

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

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

$$\hat{X_k} = \Sigma_0^{n_p - 1} x_i \tag{5}$$

and

$$\hat{Y_k} = \Sigma_0^{n_q - 1} y_i \tag{6}$$

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 [1].

5.3 First Results

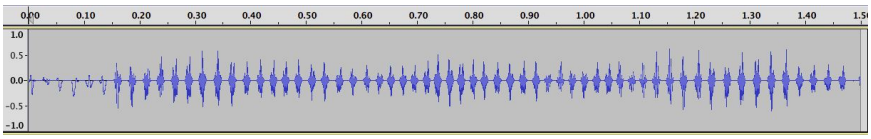

Fig. 7. A generated waveform that matched criteria $Y_{thresh} = 10$, $X_{thresh} = 2$ (up to 2 coefficient signs could be different), and 8 coefficients compared (no normalization). The speed was 6. One can see the distinct nature of the grains in this waveform, and its dissimilarity to the goal waveform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

Experiments involved setting RL-action output ranges as well as making changes in the grain engine when various discoveries were made. Previously our

duration region was between 1 and 8 or 1 and 16. The actions of the RL controller could change the duration by ±2, ±1, or 0. We tried passing the output through a linear function before being sent to the GS engine: duration = durationSlider*3. In another 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.

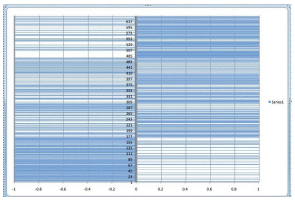

Fig. 8. Episodes for R = -1: 321 R = -1, for R = 1: 328. Note the shift from R = -1 on the left, to R=1, on the right, as the episode number increases along the Y axis.

5.4 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 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 [18], Brent, [3], and Loughran et al. [10]. This leaves 24 coefficients to use.

After more experimentation, it became clear that the threshold Y_{thresh} in Equation 6 depended on the amplitude of the soundwave rather than the relative values of the Mel-Frequency Cepstral Coefficients themselves. And the measure was 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:

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

And the resulting reinforcement function was 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 parameters for. the reinforcement criteria were $n_p = n_q = 8$, 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 = (durationSlider + 10)*3 and speed to be a constant 16.

5.5 Adding Domain Knowledge

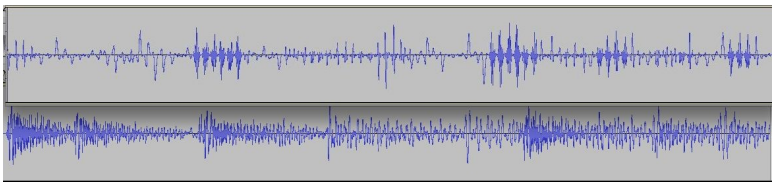

Fig. 9. Comparison of a matching wave, with the lower frequency range in place, fixed speed of 16, and the reinforcement criteria parameters 2,2, and 8.

During this experimentation, the maximum number of coefficients that matched per episode was recorded. The frequency slider had range 86.133 to 1076.65 Hz as before. The reinforcement learning controller frequency change output was in 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 added high frequencies as well according to the harmonics in the wave that composed each grain. It became obvious that some "domain knowlege" was necessary to improve the performance of this system (after we ran experiments with several wavetables). In this case, that meant 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

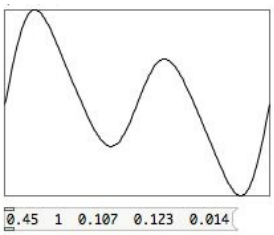

Fig. 10. The wavetable with several harmonics that is oscillated in each grain. The harmonics were determined by taking a Fourier Analysis of a small (.05 second) window of the goal waveform.

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.

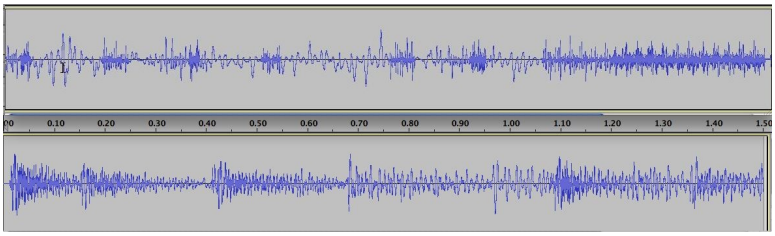

Fig. 11. Comparison of a matching wave; speed 16; parameters 2,2, and 8; lower frequency range; and using Bass harmonics Wavetable, shown with Thunder soundwave.

Figure 11 shows a grain engine-generated soundwave that received a value of R=1 along with the goal sound wave. At least 8 MFCCs sets must match per episode for R = 1. During this experiment, the maximum number of sets of coefficients matched was 22 (out of 75) in one episode. 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).

6 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 [8] that can predict and

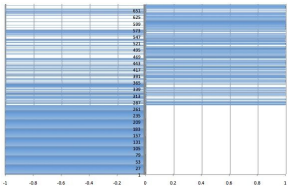

Fig. 12. R values over 670 episodes, frequency range halved, speed of 16. 298 R = -1 values, 372 R = 1 values, and shift toward R = 1 as episode number increases. Bass harmonics wavetable used, and lower frequency range used in sliders.

reiterate long sequences, for use in music re-generation and new generation [11, 12]. 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.

Two other areas to explore are the Bark scale, with its possible improvement over the Mel Scale [4], and to match a longer soundwave. Toward this goal, we plan to continue our experiments in topic models
citeLi06 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.).

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 $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
8 $e(s,a) \leftarrow e(s,a) + 1$
9 For all s,a:
10 $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
11 $e(s,a) \leftarrow \gamma\lambda e(s,a)$
12 $s \leftarrow s'; a \leftarrow a'$
13 until s is terminal

Fig. 13. Pseudo-code for Sarsa(λ) algorithm.

7 Using the Sarsa(λ) algorithm

Figure 13 shows the pseudo-code for the tabular Sarsa(λ) algorithm. The algorithm[17] 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 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 [17] for more detail and theory

8 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.

References

1. A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive element that can solve difficult learning control problems. In IEEE Transactions on Systems, Man, and Cybernetics *SMC-13:834–846.* IEEE Press, 1983.
2. J. Beauchamp. Synthesis by spectral amplitude and 'brightness' matching of analyzed musical instrument tones. In Proceedings of the Fourth International Pure Data Convention. 2011.
3. W. Brent. A timbre analysis and classification toolkit for pure data. In J. Audio Eng. Soc. *30(6):396–406.* 2010.
4. W. Brent. 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. 2011.
5. J. C. Brown. Calculation of a constant q spectral transform. In J. Acoust. Soc. Am. *89(1):425–434.* 1991.
6. L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of AAAI92.* AAAI Press, Menlo Park, CA, 1992.
7. SMV:Stanley Clarke, Marcus Miller, and Victor Wooten. Thunder. In *Thunder.* Heads Up, ASIN: B001DOQUHO, 2008.
8. F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. In Neural Computation *12(10): 2451-2471.* MIT Press, 2000.
9. B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the International Symposium on Music Information Retrieval.* 2000.
10. Risn Loughran, Jacqueline Walker, Michael ONeill, and Marion OFarrell. The use of mel-frequency cepstral coefficients in musical instrument identification. In *Ann Arbor, MI: MPublishing, University of Michigan Library.* 2008.
11. A. Non. Anonymous review. In *Withheld.* Withheld, Withheld, 2004.
12. A. Non. Anonymous reviews. In *Withheld.* Withheld, Withheld, 2004.
13. A. Non. *Withheld for review.* Withheld for review, 2004.
14. A. Non. Anonymous reviews. In *Withheld.* Withheld, 2006.

15. M. Puckette. Pure data (pd). In *Web URL*. http://www-crca.ucsd.edu/ msp/software.html, 2005.
16. C. Roads. *Microsound*. MIT Press, Cambridge MA, 2001.
17. R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge MA, 1998.
18. S. Tervo and J. Pätynen. Tutorial and examples on pure data externals: Real-time audio signal processing and analysis. http://www.tml.tkk.fi/~tervos/, Department of Media Technology, Aalto University School of Science and Technology, 2010.
19. J. M. Zmölnig. How to write an external for puredata. http://iem.at/pd/externals-HOWTO/pd-externals-HOWTO.pdf, institute of electronic music and acoustics, graz, 2011.