AUTOMATIC CONSTRUCTION OF SYNTHETIC MUSICAL INSTRUMENTS AND PERFORMERS

Ning Hu

November 2004

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

ninghu@cs.cmu.edu

Thesis Committee
Roger B. Dannenberg, *Chair*
Michael S. Lewicki
Richard M. Stern
David Wessel (UC Berkeley)

Thesis Proposal
for the Degree of Ph.D. in Computer Science

Table of Contents

Abstract

I propose an approach to create high-quality music synthesis by automatically constructing an instrument model and a performance model; the latter module generates control signals from score input, and drives the former module to produce synthetic instrumental sounds. By designing and applying appropriate machine learning techniques, the instrument model and the performance model can be trained with acoustic examples and their corresponding scores for a musical instrument. The automated model should be superior in performance to one manually modeled and tuned by hand.

In this proposal, I describe the framework of the automated synthesis and modeling system, explain the reasons for employing specific techniques, such as the necessity of modeling amplitude envelopes and a machine learning approach, discuss specific characteristics of individual modules, and propose several possible solutions for some parts of the system.

I state the contributions of the thesis topic, define criteria for project success, explore possible future work, and present a timetable for the research. I am confident that the thesis topic will yield interesting results and believe I can finish the proposed tasks within the scheduled time frame.

Introduction

The goal of this research is to create an automated system that can model musical instruments by learning from acoustic recordings. Ultimately it should be able to synthesize high-quality instrumental performances given score inputs.

The two most basic modules of the proposed framework are the instrument model and the performance model. The function of the instrument model is similar to that of ordinary music synthesis, that is, to generate synthetic sound samples given control signals as input; the performance model is in charge of converting the given digital score into control signals, which is essentially the driving force of the instrument model.

This research emphasizes designing and implementing a system framework that can automatically construct an instrument model and a performance model by intelligently learning from acoustic examples. Various machine learning techniques will be applied throughout the framework to automate the construction process.

I am particularly interested in modeling wind instruments, thus I will start the project by trying to model the trumpet. There are two main reasons for this.

First, most current commercialized synthesizers fail to synthesize realistic sounds of wind instruments. The problem lies in the conflict between the basic structure of synthesizers and the working mechanisms of wind instruments. On one hand, wind instruments are

controlled continuously by a source of energy exerted by the player. This continuous control drives the sound production. On the other hand, synthesizers (mostly sampling-based) are based on single, isolated notes, and do not offer a wide range of control over the spectrum, attacks, and envelopes. They can synthesize a single note very well, but when they are used for synthesizing a phrase or a passage, listeners will immediately notice problems.

Second, previous research by Dannenberg and Derenyi (1998) shows a similar scheme is capable of producing convincing trumpet sounds; it is natural to start from something known to be working.

For music style, I will mainly focus on synthesizing classical music first, as the playing style is "purer", more faithful to the score, and has fewer articulation effects that require additional tunings of the proposed system. In many non-classical music-playing techniques, inharmonic and transient sounds are significant. I expect those sounds can be modeled well with an appropriate residual model. However, modeling the noise (or the non-harmonic part of sounds) of musical instruments is not the primary purpose of this research. Once the synthetic performance for classical music is satisfactory, the system can be extended to synthesize other types of music.

In the following sections, I will first state the contributions of the thesis and the criteria for success; then I will describe the system structure and fundamental processes and further explain each important module in detail, including the alignment and segmentation, the instrument model, and the performance model. I will present a timetable for the work followed by the conclusion of this proposal.

Thesis Statement

The topic of this research is interesting to the world of music synthesis and also to machine learning applications. The goal is to demonstrate that, by properly designing and applying appropriate machine learning techniques, we can automatically create high-quality synthesis of musical instruments.

Several major branches in music synthesis all suffer from the problem of control, Sampling and additive synthesis (De Poli 1993) both have problems due to the fact that they are focusing on the note rather than either the production of sound or control mechanisms. Physical models (Roads 1996) promise to solve the problems of synthesis, but direct models of acoustic instruments are very difficult to build, and they are highly specific to individual instruments. The proposed method avoids the problem of note-oriented synthesis, bypasses the difficulties of building physical models, and simplifies the problem of control with spectral models and effective machine learning techniques. The thesis proposes to automatically model different musical instruments. Thus, I believe the thesis can make significant contributions to computer music and audio processing.

The minimum achievement for this research should be the completion of designing, implementing, and testing the basic system framework. The built system should be able to synthesize realistic trumpet performances for classical music, and the modeling process should be automated. Given corresponding acoustic training examples, it should also be able to automatically model some other musical instruments without much system tuning. At least it should be tested on one or two wind instruments, such as a trombone and clarinet.

Once the developed system has achieved the minimum requirement, I am planning to extend the system framework so that it can automatically model different musical instruments and music styles. I expect it will take a lot of effort to investigate the distinct characteristics of individual instruments, and the framework will become more complicated to fulfill different needs. The optimal outcome will be: the framework is general enough to be suitable for wider range of musical instruments as well as music styles, while its structure remains relatively simple and easy to control.

There are other interesting aspects of the topic that can be explored further, such as the real-time issue. Due to the time constraints, those tasks have lower priorities, and are deemed as future work, which means they are unlikely to be completed within the scheduled time frame of this thesis research.

System Structure

The whole system framework can be roughly broken down into four different processes, each targeting at a specific task. They are the synthesis process, training data pre-process, and the training processes for the performance model and the instrument model respectively.

1. Rendering an audio performance from a symbolic score

As shown in Figure 1, the synthesis system can be naturally divided into two sequential modules. The performance model gets the score input and produces a set of control signals as output. The instrument model accepts the control signals generated from the performance model as input, and synthesizes output audio. Here control signals are continuous and play a key role as an intermediate representation in the rendering process.

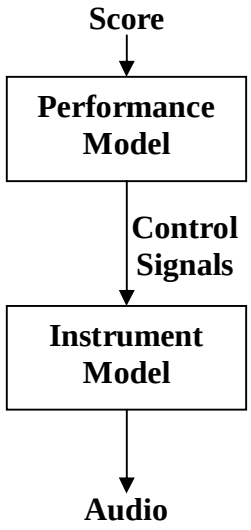

Figure 1. The synthesis process.

2. Pre-processing training data

For training purposes, the system needs acoustic examples of individual notes extracted from continuous musical phrases, along with their corresponding symbolic context. But the training inputs of the system are acoustic recordings and their corresponding scores, each representing a whole piece of music. We need an automated process that can extract audio and score for each note from the entire acoustic and symbolic input sequences. We first need to align them in time to find the precise correspondence between them. Then we will segment both scores and acoustic recordings into individual notes, and output segmented audio clips and score snippets that correspond to each other. As shown in Figure 2, pre-processing training data is an important step, and is required by both the training processes of the instrument model and the performance model.

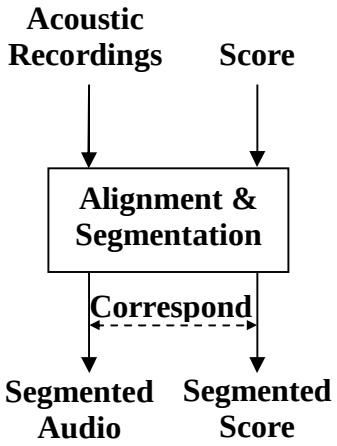

Figure 2. Data pre-processing before training.

As shown in Figure 3 and Figure 4, the segmented audio will be further analyzed by a parameter extraction module to obtain a root-mean-square (RMS) amplitude envelope and a fundamental frequency envelope. These envelopes are the training data to be learned.

3. Training the performance model

The core of the performance model is a machine learning system, which learns the mapping from score to control signals by training on segmented note examples. Figure 3 shows the automatic training process for the performance model. The control signals generated by the performance model will be compared with the ones extracted from actual acoustic examples. The comparison results will be fed back to the performance model to help the convergence of the model parameters. The ultimate goal is to minimize the difference between the synthesized control signals generated from the performance model and the actual control signals extracted from the acoustic performance examples.

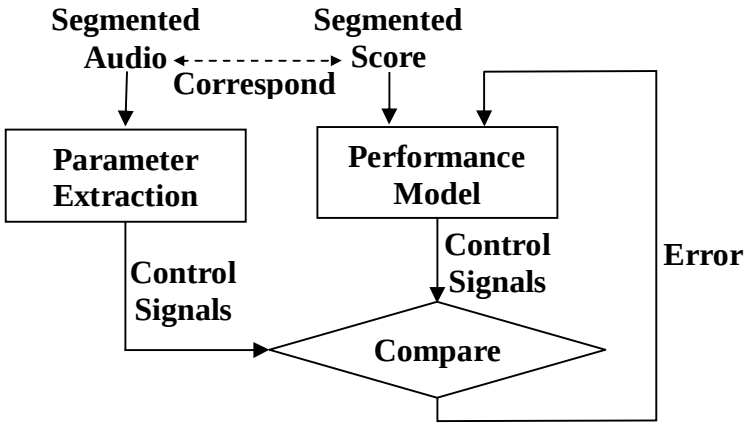

Figure 3. Automatic training process for the performance model.

4. Training the instrument model

In order to produce synthetic sound samples given control signals as input, the instrument model needs to first generate spectra from control signals. I will introduce several options for that step in later section discussing the instrument model in more detail. Some approaches are memory-based. They employ simple yet effective techniques, such as table lookup and interpolation, but they need to make certain assumptions about the system; others are based on machine learning models, and those appear to generalize quite well and very compact in size.

If a machine learning method is chosen for this specific task, we can train the model to learn the mapping scheme from control singles to spectra. Similar to the training process of the performance model, the automatic training process for the instrument model is shown in Figure 4.

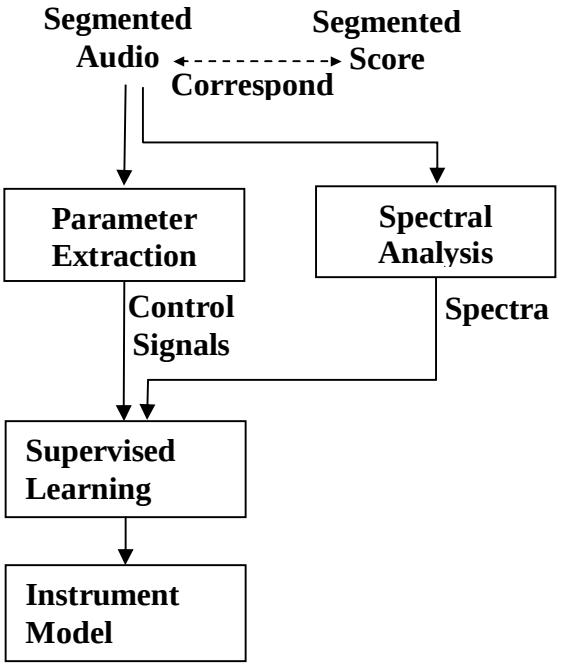

Figure 4. Automatic training process for the instrument model.

In the following sessions, I will further discuss each important module in more detail, i.e. the audio alignment and segmentation unit, the instrument model, and the performance model. I will explain the logic behind them, describe their inner structures, propose and compare one or multiple approaches for each of them.

Audio Alignment and Segmentation

As I just discussed, to train the instrument model and the performance model, we need audio clips and corresponding score snippets extracted from the training inputs, which are whole pieces of acoustic recordings and scores. The first step is to figure out how the acoustic recording is corresponding in time to the score. An optimal alignment between them will do the job of finding the correspondence.

I have done some research in the area of polyphonic audio alignment. I developed a method that aligns polyphonic audio recordings of music to symbolic score information in standard MIDI files without the difficult process of polyphonic transcription (Hu, Dannenberg, and Tzanetakis 2003; Dannenberg and Hu 2003). It extracts specific acoustic features, called chroma, from audio data, and directly maps MIDI data to the same type of acoustic features as well. Then it uses a dynamic programming (DP) algorithm or a Hidden Markov Model (HMM) to find the optimal alignment between the two feature sequences. The alignment represents the correspondence between the audio data and score. The experimental results are satisfactory, and show promise for this application.

After the optimal alignment between the acoustic recording and the score is found, the audio data needs to be further segmented into individual notes or phrases. Segmentation has a very high requirement for accuracy. Segmentation error allowance might be as low as several milliseconds.

Dannenberg and Matsunaga (1999) described some early work of trying to segment the notes with very accurate transition times. They defined some rules and thresholds to detect the appropriate start and end point of each note, using several features including power, number of peaks, and number of zero-crossings per fundamental period. That method shows promising results, but it is not reliable enough for a completely automated system, and accuracy is still below expectation.

In my view, alignment and segmentation are closely related. If the alignment is precise enough, we can directly truncate note segments from audio accordingly. The technique for polyphonic audio alignment can be easily applied for this task. But as higher accuracy is required, it obviously needs further modification by adding some functions that can deal with fluctuation details of waveforms. The shortest size of audio frames I have tried so far is 0.1 second. That is good enough for a reasonable alignment of whole music pieces, but it is too long for this particular task.

Kapanci and Pfeffer (2004) offered alternative approaches for accurate note segmentation. Basically they turned the segmentation problem into a classification problem and proposed a hierarchical machine-learning framework. The purpose is to overcome the difficulties of detecting soft onsets by comparing frames separated by increasingly longer distances. Though the reported results of the hierarchical approach were not significantly better than the non-hierarchical one, the idea is quite inspiring. I believe by appropriate design and implementation, it could be a successful solution for the task of audio segmentation.

Instrument Model

The instrument model accepts a set of control functions as input and produces a digital audio sound as output. The goal is to produce realistic instrument tones given the proper control signals. In order to insure the success of the whole synthesis model, the sound produced should be perceptually as close to the acoustic instrument being modeled as possible.

The sounds of many musical instruments can be described by the combination of harmonic and additive noise signals (Serra 1989). I intend to consider these two parts separately to ease the problem.

• Harmonic Model

Many successful attempts in music synthesis only focus on the harmonics and ignore the noise part. A classic example is the additive synthesis (Moorer 1977). It models sounds as

summations of deterministic sinusoidal components (the partials). I also impose a common constraint that the frequencies of the partials are all multiples of a fundamental frequency. The key to the success of those techniques is that the acoustic instrument being modeled is indeed nearly harmonic.

The block diagram of the harmonic model itself can also be partitioned into three sequential steps illustrated by the dotted lines shown in Figure 5.

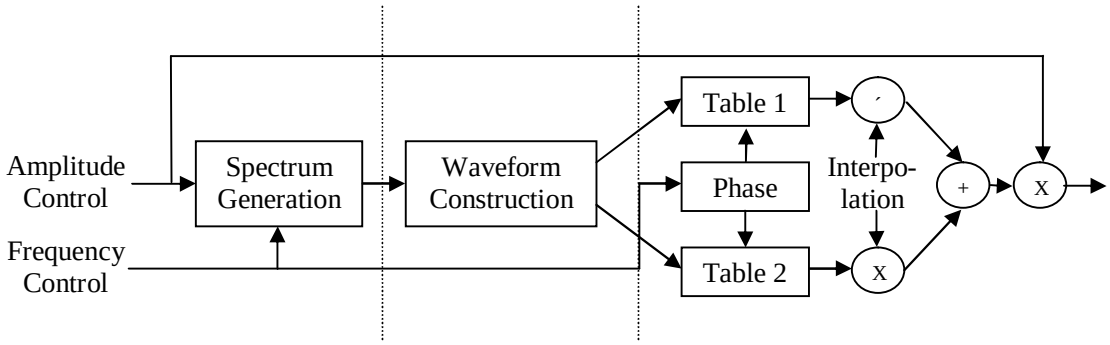

Figure 5. Block diagram for the harmonic model.

The first step takes the control signals as input, and outputs spectral characteristics, notably several harmonic partials and their amplitudes (the spectra); Given the spectrum information from the first step, the second step generates a wavetable (Moorer 1978), which represents one period of the periodic sound in the time domain; the third step produces every sound sample by interpolation between two adjacent wavetables.

1. From Control Signals to spectra

Dannenberg and Derenyi (1998) demonstrated the success of utilizing spectral interpolation techniques in their model for synthesizing trumpet tones. It will also be the first and one of the most important techniques I am going to use and explore. The technique itself is quite simple. I first need to compute some spectra according to the following steps:

1) Record a set of sounds, where each embodies a simple note in a distinct pitch and the amplitude level is either increasing or decreasing at a moderately fast speed, covering as wide a dynamic range as possible. The pitches are pre-defined.
2) Obtain a spectrogram for each sound, where each spectrum frame corresponds to a period.
3) Scan through the spectrogram and retain only spectra at which the amplitude function crosses predetermined thresholds.

Those spectra will be indexed into a two-dimensional lookup table by their corresponding pitches and RMS amplitudes. When accessing this two-dimensional

lookup table by the instantaneous amplitude along one dimension and the pitch along the other, the system will interpolate among four nearest spectrum samples to yield an output spectrum.

Wessel, Drame, and Wright (1998) also studied and compared two other synthesis techniques. One is the memory-based approach. It stores in the memory a set of the spectral information as data points in an n dimensional space; the dimension indexes are their corresponding frequency, RMS amplitude, and etc. The input is a vector indicating a specific point in that space. To generate the output corresponding to each input value, it chooses the **k** nearest neighboring data points, weights each of them as a function of the distance between input point and itself, and computes the output by averaging the weighted **k** selected data points. It is a very flexible model and relatively easy to modify for different requirements. I should point out that the spectral interpolation technique introduced before can be deemed as a special case of the memory-based approach, as here the dimension **n** is 2 (fundamental frequency and RMS amplitude), **k** is 4, and the weighting function of the distance between the input and the data points is linear interpolation.

The other method described utilizes a feed-forward neural network with multiple layers. The input units accept the frequency and amplitude functions, and the output units produce the frequencies and amplitudes of the sinusoidal components. The model can be trained with a back-propagation learning method. Unlike the memory-based approach, it does not need to make assumptions about either the distance function or the weighting function, which is its advantage. Thus, the neural network model is very compact and appears to generalize well.

The authors reported that both the neural-network and memory based models functioned well in a real-time context, and in general the neural network model provided a smoother sounding result than the memory-based model. That conclusion makes us particularly interested in the neural network model. If time permits, I would like to implement the neural network model and compare it to the spectral interpolation model.

2. From Spectra to Wavetables

There are several ways to compute wavetables from the spectra, such as IDFT, IFFT, etc. A simple approach is to perform a simple summation of sinusoids, each representing a harmonic partial and weighted by the corresponding relative amplitudes. How often the wavetables should be computed from the spectra is another issue to be considered. As the computing process from control signals to wavetables is relative expensive, it is technically applicable but inefficient to compute a wavetable for every period of the synthesized sound. A better way is to compute an adequate number of spectra per second, and produce every sample by interpolation between two tables. This leads to the third step.

Also in order to avoid any phase cancellation during the interpolation in the third step, wavetables must be created with matching phases.

3. From Wavetables to Sound Samples

In this step, I will interpolate between two adjacent wavetables to generate synthesized sound samples. This should effect a smooth, continuous spectral change. Then the result is scaled by the instantaneous RMS amplitude to produce the proper amplitude fluctuations in the sound.

After this step, the synthesized sound should have controlled fluctuations in pitch, amplitude and timbre.

• Residual Model

The residual component refers to the stochastic non-harmonic part of the sound. It is also an important factor that should not be simply overlooked, though that hugely depends on the characteristics of individual instruments. For example, I expect that for the trumpet and most woodwind instruments, the inharmonicity in general can be pretty much ignored except for the attack portions at the beginning of each note; on the other hand, modeling some "noisy" instruments, such as a flute may require careful consideration in designing and incorporating the residual model.

Derenyi and Dannenberg (1998) demonstrated that attacks are particularly important for synthesizing convincing trumpet tones. This indicates I can temporarily overlook the general inharmonicity throughout each note and focus more on the attack model.

A common way to give the impression of a natural attack is to use recorded attacks. In order to get a satisfactory attack sample, I first need to properly segment an attack portion from a recorded sound. This can be done automatically by observing the relations of the frequencies of the partials, as the attack represents the transitional part from silence or complete inharmonicity to harmonic partials.

Anther thing to note is the phases of wavetables should be adapted to the phrases of the attacks. Dannenberg and Derenyi (1998) introduced a good way to solve this problem: to measure the phases and amplitudes of the harmonics as well as the overall RMS amplitude of the sound at the end of the attack. These attributes determine the initial phase and amplitude for the first wavetable, and also the phases for all wavetables until silence or the next attack.

There are several questions to be answered: whether to attach an attack to every note onset, or just the beginning of every slur and legato phrase; how many and what kinds of sampled attacks are needed; and how to use and scale them.

Performance Model

The goal of the performance model is to automatically generate proper control signals for the instrument model based solely on symbolic information from a score.

Most sampling-based synthesizers only get some coarse note features of pitch, duration, and loudness as input. But that is not enough for our purpose of creating natural-sounding wind synthesis. The performance model has to focus on the fine details of control signals, including appropriate amplitude and frequency envelopes, slurs, vibrato, attacks, and other audible effects. The questions lies in whether I can obtain most of those fine details by analyzing information residing in music structure, the relations inside or between the phrases, and the interactions among notes.

Even though many musical instruments such as a trumpet are complicated dynamic systems, it appears that the system behavior is almost entirely characterized by the amplitude and fundamental frequency at any moment. There are dramatic and systematic changes to amplitude and frequency envelopes depending upon how the note is articulated, whether the note is part of an ascending or descending line, and other factors relating to the context of the note. A quarter note followed by a rest is played differently from one followed by another note. The experiments done by Dannenberg, Pellerin, and Derenyi (1998) also verified that, by properly modeling amplitude envelopes, very convincing trumpet tones can be synthesized. Thus, I am particularly interested in synthesizing amplitude and frequency envelopes.

Traditional computer music techniques, such as ADSR envelopes (Adams, 1986), are way too coarse for natural-sounding music synthesis, so a more sophisticated model with more parameters is required. Our successful modeling process will have to rely upon good control functions for pitch and amplitude.

Dannenberg and Derenyi (1998) successfully designed functions mapping from features in the score to amplitude envelopes. Though the mappings/functions tuning were done by hand, the way they approached the problem shows a promising future for using machine learning techniques to train on performance examples and learn the mappings automatically. This will be a major contribution of the thesis.

• Amplitude Envelope

As suggested by Clynes (1984), amplitude envelopes should be modified according to context. Dannenberg, Pellerin, and Derenyi (1998) demonstrated that on the trumpet sound. The comparison between Figure 6 and Figure 7 show how the amplitude envelope of a note can change drastically under different music context.

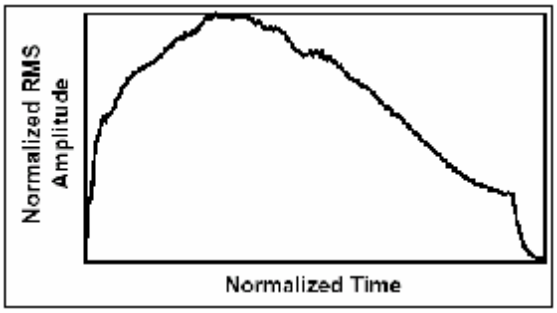

Figure 6. A typical trumpet amplitude envelope (a *mezzo forte* $A\flat_4$ from an ascending scale of tongued quarter notes). Figure borrowed from the work of Dannenberg, Pellerin, and Derenyi (1998).

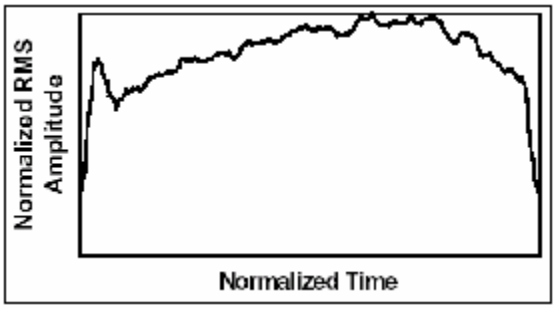

Figure 7. A typical trumpet slurred amplitude envelope (a *mezzo forte* C_4 from an ascending scale of slurred quarter notes). Figure borrowed from the work of Dannenberg, Pellerin, and Derenyi (1998).

An optimal shape of synthetic envelope should be as close to the actual acoustic envelope as possible, as illustrated in Figure 8. A typical metric for measuring the difference between a synthetic envelope and an actual one is to take RMS of the value differences between two envelopes at selected time points. Moreover, recent work (Horner, Beauchamp, and So 2004) has attempted to characterize how accurate envelopes and spectra should be to be perceptually similar. It is a useful reference of picking the effective error metrics for evaluating synthetic envelopes. As suggested, some good ones are relative-amplitude spectral error, and RMS relative-amplitude spectral error.

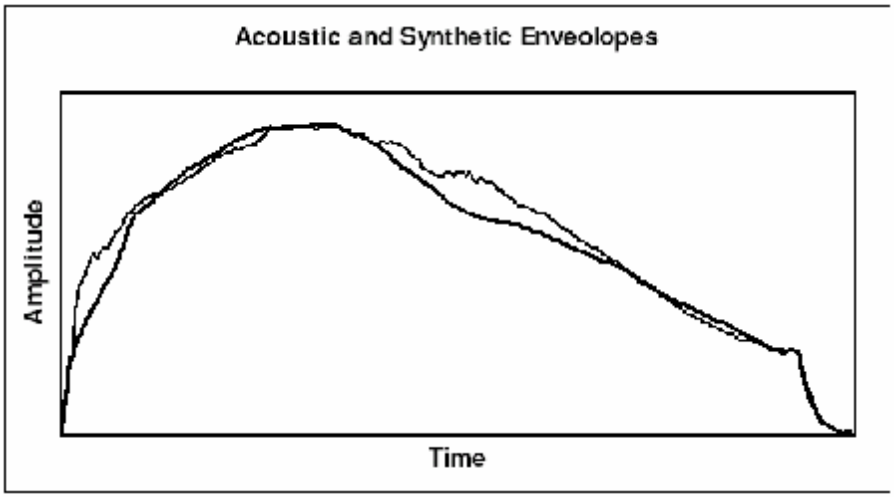

Figure 8. An actual amplitude envelope (thin line) and a synthetic envelope (heavy line). Figure borrowed from the work of Dannenberg and Derenyi (1998).

There are many possible approaches to applying machine-learning techniques to synthesize envelope shapes. Two most important and closely related issues are how to effectively represent envelope shapes, and what machine-learning model to be used for learning the mapping scheme from the information extracted from score (music context) to envelope shapes.

q Envelope Representation

As seen in Figure 6 and Figure 7, amplitude envelopes are univariate time series, and there are some specific properties of amplitude envelopes (for a note):

- Ø It has specific duration.
- Ø The envelope shapes are roughly arched, with two ends lower (not necessary zero) and middle higher in values.
- Ø It is well known that acoustic envelopes can be segmented into several continuous parts similar to ADSR model, e.g. attack, sustain, decay, release, etc. They are all very important and have distinct characteristics, for example, the duration of attack is always very short, while the duration of sustain is greatly determined by the note duration.

I can use various kinds of parameters to describe amplitude envelopes, and they can be roughly classified in following two categories:

v Collection of General Parameters

A lot of parameters are available for describing the curves, such as center of mass, global/local maximum/minimum, curve smoothness, and data points in absolute or relative scale with certain intervals. I can either pick some of them by hand, or let machine-learning model to figure out what parameters are more useful (contain more information) than the others. The later method is preferred.

v Wavelets

Wavelets are a mathematical tool for hierarchically decomposing functions. They allow any function to be described in terms of a coarse overall shape, plus details that range from broad to narrow. Wavelets have been a very popular and powerful technique for many areas, including computer graphics, computer vision, and signal processing.

q Mapping Scheme

To convert music context to envelope shapes, various machine-learning models are applicable. They can also be classified into two categories:

Ø Non-linear regression

Basically these kinds of models try to discover and describe the underlying relations between music context and actual acoustic envelopes. There are many non-linear regressors that are possibly good options for this task. Some examples are neural networks, Kalman filters, and functions approximation.

Ø Pattern clustering

Envelopes are classified into multivariate clusters based on their features from music context and acoustic shapes, each cluster with a representative envelope shape. For any arbitrary input with music context features, the model figures out what category it falls into, and uses the corresponding representative envelope as the base shape, then does appropriate local or global stretching, scaling and interpolation on the shape. This could be considered a form of case-based reasoning.

- **Frequency Envelope**

Frequency envelopes are also very important. A synthetic performance with steady, unwavering frequencies sounds artificial when compared to an acoustic recording. Frequency envelopes possess different characteristics from amplitude envelopes. But if I study them closely and carefully, I believe I will be able to synthesize frequency envelopes similar to how I model amplitude envelopes. Of course some variations will be needed, such as using a different set of parameters that can describe frequency envelopes better, or incorporating a vibrato model.

Schedule

October 2004 ~ November 2004	Propose thesis topic; Collect performance examples for initial experiments.
December 2004 ~ May 2005	Alignment and segmentation of acoustic recordings; Incorporate SNDAN package to prepare data necessary for instrument and performance models; Develop and compare instrument models; Design and implement performance model; System integration; Model other musical instruments; Synthesize other types of music.

June 2005 ~ October 2005	System and model evaluation; Fine tuning of the system; Thesis write-up and revision.
November 2005	Thesis defense.

Conclusion

Here I am proposing a scheme for automatically constructing high-quality musical instrument synthesis by listening to performance examples. Besides the traditional synthesis techniques, machine-learning methods will play a crucial role. They will be used pervasively throughout the whole system in order to automate the modeling process and intelligently control the synthesis.

I will be starting the project by trying to synthesize the trumpet. Once the system proves to be working correctly and shows satisfactory results, I will try it on other instruments, possibly some other wind instruments, such as alto saxophone and flute. String instruments may be considered as well. I expect that I may need to slightly tune and modify the system according to the characteristics of different instruments. My goal is to make the system general enough for synthesizing a handful of musical instruments without much specialization, while still outputting realistic synthetic sounds.

After first focusing on synthesizing classical music, I will try to synthesize other types of music as well, for example, jazz music. A successful system should be able to synthesize different types of music, reproducing their playing styles, effects, and other distinct characteristics.

I will also need to think about the possibility of making the system work in real-time. That is, the system will accept control signals triggered by electronic keyboards or other electronic musical instruments, and output corresponding synthetic audio immediately. The current design of the system is for off-line purposes, not because rendering audio takes some time, but mainly due to the fact that it requires phrase information. For example, to render a note, the system needs to know the properties (pitch, duration, etc.) of not only the note itself, but also the notes before and after that. But in real-time, one cannot get information about the future, not even the duration of the current note. However, I still believe I can make some progress in pursuing real-time synthesis. At least the system should be able to synthesize on-line with a very short time lag.

Overall I believe the topic is quite interesting, the experimental approach is clear, and I should be able to finish thesis project within the scheduled time frame.

Acknowledgments

This project is greatly inspired by Roger B. Dannenberg and Istvan Derenyi's (1998) outstanding work on designing and developing the combined spectral interpolation synthesis technique. As a trumpet musician, Roger B. Dannenberg also performed and helped record the acoustic examples for initial experiments. James Beauchamp generously provided the SNDAN package (Beauchamp 1993), which will be very useful for the modeling process.

References

Adams, R. 1986. *Electronic Music Composition for Beginners.* Dubuque, Iowa: Wm. C. Brown Publishers.

Beauchamp, J. 1993. "Unix Workstation Software for Analysis, Graphics, Modification, and Synthesis of Musical Sounds." *Audio Engineering Society Preprint,* No. 3479 (Berlin Convention, March).

Dannenberg, R.B., Pellerin, H., and Derenyi, I. 1998. "A Study of Trumpet Envelopes." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 57-61.

Dannenberg, R. B. and Derenyi, I. 1998. "Combining Instrument and Performance Models for High-Quality Music Synthesis." *Journal of New Music Research*, 27(3), (September), pp. 211-238.

Dannenberg, R. B. and Matsunaga, M. 1999. "Automatic Capture for Spectrum-Based Instrument Models." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 145-148.

Dannenberg, R.B. and Hu, N. 2003. "Polyphonic Audio Matching for Score Following and Intelligent Audio Editors." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 27-34.

De Poli, G. 1993. "Audio Signal Processing By Computer." In *Music Processing.* G. Haus, *ed*. Madison: A-R Editions, Inc., pp. 73-105.

Derenyi, I. and Dannenberg, R. B. 1998. "Synthesizing Trumpet Performances." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 490-496.

Horner, A., Beauchamp, J., and So, R. 2004 "A Search for Best Error Metrics to Predict Discrimination of Original and Spectrally Altered Musical Instrument Sounds." In

Proceedings of the International Computer Music Conference. San Francisco: International Computer Music Association, pp. 9-16.

Hu, N., Dannenberg, R. B., and Tzanetakis, G. 2003. "Polyphonic Audio Matching and Alignment for Music Retrieval." In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York: IEEE, pp. 185-188.

Kapanci, E. and Pfeffer, A. 2004. "A Hierarchical Approach to Onset Detection." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 438-441.

Moorer, J. A. 1977. "Signal Processing Aspects of Computer Music – A Survey." In *Proceedings of the IEEE.* (July).

Moorer, J. A. 1978. "How Does a Computer Make Music?" *Computer Music Journal.* 2(1) (July), pp. 32-37.

Roads, C. 1996. "Physical Modeling and Formant Synthesis." In *The Computer Music Tutorial.* Cambridge: MIT Press, pp. 263-316.

Serra, X. 1989. *A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition*. PhD thesis, Stanford University.

Wessel, D., Drame, C., and Wright, M. 1998. "Removing the Time Axis from Spectral Model Analysis-based Additive Synthesis: Neural Networks versus Memory-Based Machine Learning." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 62-65.