UNIVERSITY OF CALIFORNIA, SANTA BARBARA

Individualized Intuitive Control of Spectral Model Synthesis

Phillip Popp

Submitted in partial fulfillment of the requirements for a Master of Science degree
Media Arts and Technology Program
University of California, Santa Barbara

CONTENTS

ABSTRACT

The mapping between performer input control and auditory output plays a fundamental role in the design of any musical instrument. Studies have shown that the choice of mapping strategy has a qualitative and quantitative effect on a user's ability to explore sonic spaces and perform complex musical gestures. This project aims to derive an individualized and intuitive mapping between input gestures and synthesis control parameters by capturing the user's input gestures as he/she pantomimes to synthesized audio. This allows the user to define their gesture language as well as the sonic space they wish to explore. The proposed mapping technique builds upon previous works that mapped input devices to gestures and psycho-acoustic information to synthesizer controls. The software developed for this project facilitates this process by recording and pairing input gestures extracted from a Wacom Tablet to Spectral Model Synthesis (SMS) frames extracted from an audio file and analyzing those paired values to derive a mapping. First, gestural-related features such as the current state of the Wacom Pen, derivatives of the pen's state and a wavelet transform of the pen's most recent states are extracted from the input gesture data. Second, a modified form of principal component analysis (PCA) is used to choose and emphasize combinations of features which display high variance, and hence high expressivity. Finally, either an artificial neural network (ANN) or a k nearest neighbors (KNN) algorithm is built to translate new combinations of gestures into SMS frames. Analysis of the proposed design shows that the act of pantomiming audio offers useful training data for slowly changing audio features, but produces prohibitively noisier data when quickly changing audio features are present within consecutive SMS frames. A modified version of PCA is shown to outperform traditional PCA in this context by emphasizing features that show high expressive potential. Comparatively, ANN offers a greater ability to explore new gesture and sonic space but fails to capture many nuances of the original sound due the averaging nature of ANN regression and noisiness of the training data. KNN reproduces recognizable sounds from the original input audio but suffers from the inability to generalize beyond the original audio.

INTRODUCTION

The advent of digital audio synthesis has opened many doors to and unleashed many constraints for the instrument designer and the musician. Particularly, we have seen an expansion in both the variety of input devices and the number of parameters required to control synthesis engines. Many instrument designs have attempted to map between input devices and synthesis engine parameters by exploiting expert knowledge of the mapping domain and/or lower dimensional metaphors to explore the high dimensional space often present in a synthesis engine. However, these designs fail to explicitly address the varied ways in which users express sonic gestures [7][8]. This work attempts to derive a more intuitive and individualized mapping between input device parameters and synthesis engine parameters by capturing a user's interaction with a Wacom Tablet as he or she pantomimes to audio synthesized from a Spectral Model Synthesis (SMS) file. A means to project tablet parameters into a gesture language is derived from the captured input parameters by treating high varying features extracted from the input device as carrying high expressive potential. After the captured input device parameters are projected into the gesture

language space, a learning algorithm derives a translation between the gesture and individual SMS frames. Once the gesture language has been defined and the translation of gestures to synthesis engine parameters determined, parameters from the input device are mapped to synthesis engine parameters in real time to provide intuitive and individualized control of a synthesis engine.

Previous attempts to create an intuitive mapping between an input device and a synthesis engine have explored a variety of strategies. Some have exploited expert knowledge of the input device and synthesis engine to derive an explicit mapping between the two [15][27][32][35]. Others express changes in a high-dimensional synthesis engine control space through lower dimensional metaphors [2][20]. Many turn to machine learning techniques to recognize gestures or map from low dimension device parameters to a high dimension synthesis engine parameters [2][6][17][19][33]. The work presented here differs from these previous works by recognizing that users intuitively control sound in varied manners. Rather than defining an a priori gesture language, one is extracted as a user pantomimes to synthesized audio. Then, after a mapping is learned between gestures and synthesis engine controls, the user is presented with an intuitive and individualized control of the synthesis engine.

This thesis is organized by first providing a brief history of mapping techniques and reviewing previous works that address the mapping of input device parameters to synthesis control parameters. Second, design goals are specified in order to create an individualized and intuitive mapping between input device parameters and synthesis control parameters. Third, a method to create a desired mapping as well as use it within a real-time synthesizer is defined. Last, analysis of the proposed methodology presented and future work projected.

A MAPPING HISTORY

Instrument design methodology has undergone several major changes resulting from the freedom provided by electronic and digital synthesis. Mapping performed in classic musical instruments required the designer to integrate interface, mapping and sound production into a single physical object. The advent of electronic audio synthesis abstracted the interface from the sound producing mechanism affording the designer greater freedom. Despite this, designers predominantly kept to the piano keyboard to promote a familiar interface for novel audio synthesizers. Most recently, the rapid growth in digital control interfaces and synthesis engines has encouraged the study and experimentation of varying mapping techniques. Here, a brief history of musical instrument control mapping is presented in order to provide a context for the proposed design.

In order to understand the scope of the discussion at hand, it is important to assign some formal definitions to the word "mapping" as it is used throughout this text and many of the texts referenced. We are primarily concerned with the act of interpreting real-time performance data from an input device and translating those interpretations into the control parameters of a synthesis engine as shown in Figure 1. This act will be called "mapping" throughout this paper.

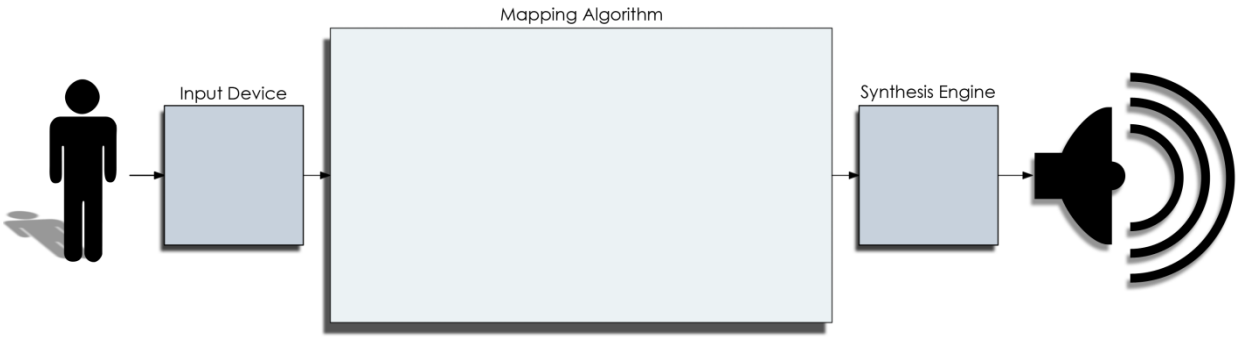

Figure 1 Modular Description of Instrument Using an Input Device, Mapping Algorithm and Synthesis Engine

MAPPING IN TRADITIONAL INSTRUMENTS

Mapping executed in a traditional musical instrument generally combines the input device, control mapping and synthesis engine into a single physical object. Their design requires a careful balance between the physical structure of the instrument, its sound producing mechanism, and control of the sound producing mechanism. This limits the sonic space it can explore by placing definite boundaries originating from physical limitations. The flute, for example, roughly divides the control domain into pitch, loudness, and timbre. A mix of finger placement and airflow produce distinct pitches arranged along a chromatic scale. Loudness, on the other hand, is continuously controlled by the strength of the performer's breath, allowing the loudness to swell and diminish smoothly. Lastly, timbre is given only slight control by altering the embouchure. If we consider a flute in the context of an input device, mapping, and synthesis engine, we find that they are all combined into a single object. Hence, it is impossible to independently alter the input interface, mapping or synthesis engine. There are, however, several exceptions to this rule. The piano provides a good example of a classical instrument whose user interface has little to do with producing sound other than that it transfers energy from the musician's fingers unto hammers which strike the strings. The church organ expands even further in modularity by introducing a separate energy source to generate sound.

MAPPING IN ELECTRONIC INSTRUMENTS

The design and machination of electronic musical instruments has rich history, albeit a short one compared to the development of many traditional musical instruments. Reviewing their evolution helps us understand their effect on the current state of electronic instrument design.

Electronic instruments primarily began as additions and/or simulations of preexisting instruments. The Electro-Mechanical Piano, built in 1867 Switzerland by Msr Hipps is one of the first electronic instruments built where the user interface is abstracted from the sound generating mechanism. This and most of the first generation of electronic musical instruments prescribed to a keyboard interface which acted as a set of switches to a large circuit board. The keyboard interface was easy to integrate into an electrical circuit, as it can act as a simple switch controlling the power delivered to a sub-circuit. It also provided a familiar interface that could be played by anyone who had previously learned to play piano. Several unique outliers amongst this first generation of electronic instruments are the Theremin and the Hellertion, both of which allowed for continuous amplitude and pitch control [3]. While the Theremin and the Hellertion both utilize unique

interfaces for their time, most all instruments of this time relied on a similar circuitry to provide a synthesis engine. Each interface controlled either a bank of oscillators, or the pitch and volume of a single oscillator.

The proliferation of the semi-conductor in the 1970s spurred variations in both control interfaces and synthesis techniques. Interfaces diverged into smaller and more portable devices, such as the compact PAiA OZ, while concurrently expanding into more elaborate and grandiose contraptions such as the Buchla Electric Music Box with its web of cables and patches. Synthesis techniques also diversified and expanded by allowing multiple synthesis presets to be saved, additional controls to be employed and even computer screens to give textual control and feedback.

Following the introduction of the personal computer was a cavalcade of audio synthesis and compositional software. Initially the software emerged in the form of scripting languages such as CSound and the Music-N series, which focused on orchestrating music rather than doing live compositions. Later other products emerged such as ProTools and Audacity which allowed for editing, recording, and arranging of audio as one would at an audio mixing board. As well, packages such as Cakewalk utilized MIDI synthesizers to create sounds from instruments connected to the computer. This allowed researchers and musicians to experiment and prototype new techniques in audio synthesis and synthesis control. In turn, this gave birth to strong foundation of audio synthesis research concerned with input devices, audio synthesis techniques, and the mapping between input device parameters to synthesis engine parameters.

THE STUDY OF MAPPING

Mapping has evolved significantly since it first became an active study in electronic instrument design. Research into our relationships with instruments and the implications mapping techniques have on that relationship have spurred various organizations and abstractions of the mapping process as well as many concrete examples of effective mapping techniques.

IMPLICATIONS OF MAPPING

Several works investigate the effect of mapping on real-time musical performance and how gestures relate to musical sounds. They argue quantitatively or qualitatively the importance of proper mapping strategies, their relation to musical performance and their relation to human gesturing.

Hunt et al. argue that more complex mappings aid, rather than abate, the ability to perform a musical task [14]. They defended their conclusion by performing experiments where participants were asked to perform musical tasks. They found that those with more complex control mappings outperformed their peers with simpler control mappings. Pain reiterates this argument and also claims that the combination of input device and mapping must physically communicate the tasks they are performing if they are to be engaging during real-time performance [22]. Wessel rather focuses on the experience of the musician and the need for sensory-motor engagement in the musical experience [33]. Combined, these works claim the need to model complex mappings that express their intentions both to the musician and the audience through physical motions.

In addition to the effects of mapping on musical performance, others have taken detailed looks into how humans express music gesturally and to which parts of music these gestures relate. Levitin et al. analyzed the expressive qualities of a musical tone, separating it into several stages so that different control techniques may be applied appropriately [18]. Rasamimanana et al. performed an in-depth view of bowing techniques and how they relate to the produced sound [23]. Godoy et al. inspect the similarities and dissimilarities in which people pantomime audio when there is no instrument present [7] [8]. Their studies imply that the relation between gesture and sound is a complicated one which varies from sound to sound and person to person.

Organizing Mapping Process

Several authors formally set out to define frameworks for the creation and analysis of mapping strategies. These range from introducing abstract spaces within the mapping algorithm, creating a framework to analyze an instrument's qualities, and a refutation of the basic mapping problem formulation. Hunt et al., in addition to arguing for complex mappings, define three spaces, the input device space, an abstract space, and a synthesis engine space. By defining an abstract intermediary space, the input device and synthesis engine are allowed to vary independently as long as a mapping can be made to and from the center abstract space. They argue that introducing an abstract layer allows for more complex mappings which in turn result in better performance [14]. Arfib et al. go one step further than the previous authors by describing four separate spaces that must be mapped [1]. Naturally, the first space is an input device parameter space. It is followed by a gestural-perceptual space, then by a sonic-perceptual space, and finally a synthesis model space. The authors argue that they have improved the previous design using only a single space by introducing more modularity to their design and defining what roles each space should play. Rather than defining a design space Overholt specifically targets the analysis of digital instruments, providing a framework to analyze the mapping strategy [21]. Much of the previous three works is challenged by Van Nort. He attempts to refute the framework of gesture to mapping to control as the a priori model for synthesis design. As an example he suggests that for certain designs, it is more applicable to consider human notions of sonic gesture and the perception of human intentionality in sound. This work dives much more deeply into human considerations and largely utilizes a phenomenological model of instrument design.

Explicit Mapping

Explicit mapping requires the intervention of a designer to explicitly choose how to map each input device parameter to each synthesis engine controls. Several authors have built prototypes and provided tools to facilitate this process. Steiner provides users of the audio synthesis platform PD with a wide range of linear, non-linear, many-to-one and one-to-many mappings so that they may build their mappings from scratch [27]. Van Nort et al. use a Wacom Tablet to map to higher dimensional control structures by warping the control space as well as allowing parallel and meta control of the mapping layer [29]. Momeni et al. attempt to provide an intuitive mapping from a low dimensional controller to a high dimensional synthesis engine by using geometric models. They explore the use of Gaussian kernels to aid users in understanding the interpolation of points in a high dimensional synthesis control space [20].

GENERATIVE MAPPING

Generative techniques have been applied to several different stages of the mapping process either as a means to intuitively navigate a high dimensional control space, translate input device parameters into control gestures, or to map between high level audio descriptors and synthesis engine controls. Each uses some form of training data and machine learning algorithm to determine a mapping. A large number of authors have presented works built upon machine learning algorithms. They utilize them to perform a wide variety of functions from building controllers to extracting manifolds. Notably, Modler uses artificial neural networks to translate data-glove controls to sound control gestures [19]. Wessel et al. train artificial neural networks and k nearest neighbors algorithms with high level sonic descriptors, such as loudness and pitch, to recreate additive synthesis controls for several wind instrument synthesizers and a voice synthesizer [31]. Wessel uses artificial neural networks again to derive a forward model of control mechanisms. His design utilizes perceptual distance metrics to calculate errors between the desired output and the perceived output. These errors update a neural network forward model and controller [33]. A more novel machine learning algorithm is utilized by Choi et. al. to create paths within high dimensional spaces. They use a generative algorithm to create manifolds so that paths between two sets of high dimensional control data can be intuitively traveled and explored [2].

DESIGN GOALS

Studies exploring the varying gestures used by individuals to express abstract sonic gestures upon a non-musical interface support the need for individualized and intuitive control of synthesis engines [7][8]. As well, the control of synthesis engines with high dimensional control spaces make it difficult to provide intuitive control using explicit mapping techniques. In an attempt to solve this problem we aim to extract an intuitive and individualized mapping between a Wacom Tablet and SMS parameters by capturing and analyzing pantomimed controls to audio synthesized from a script of synthesis control parameters.

DESIGN AND IMPLEMENTATION

The design and implementation of a synthesizer offering individualized and intuitive control between a Wacom Tablet and SMS synthesizer requires several discrete components. These are presented here in a chronological manner reflecting the steps a user would take to use the proposed system and the order in which Wacom Tablet parameters are translated into SMS frames as shown in Figure 2. First, we briefly introduce the Wacom Tablet, Spectral Model Synthesis, and the software developed to implement the proposed solution. Second, we present a method for capturing and pairing Wacom Tablet parameters and SMS frames by allowing the user to pantomime to synthesized audio. Third, we define a gesture language and a means to extract gestures from a series of Wacom Tablet parameters by assigning greater significance to features from the Wacom Tablet parameters that displayed high expressive potential. Fourth, two methods are proposed, k nearest neighbors (KNN) and artificial neural networks (ANN), to learn a mapping between extracted gestures and SMS frames. Finally, we present a completed instrument including

the aforementioned gesture extraction and gesture to SMS frame mapping algorithm, as well as a method for smoothing the information in consecutive SMS frames.

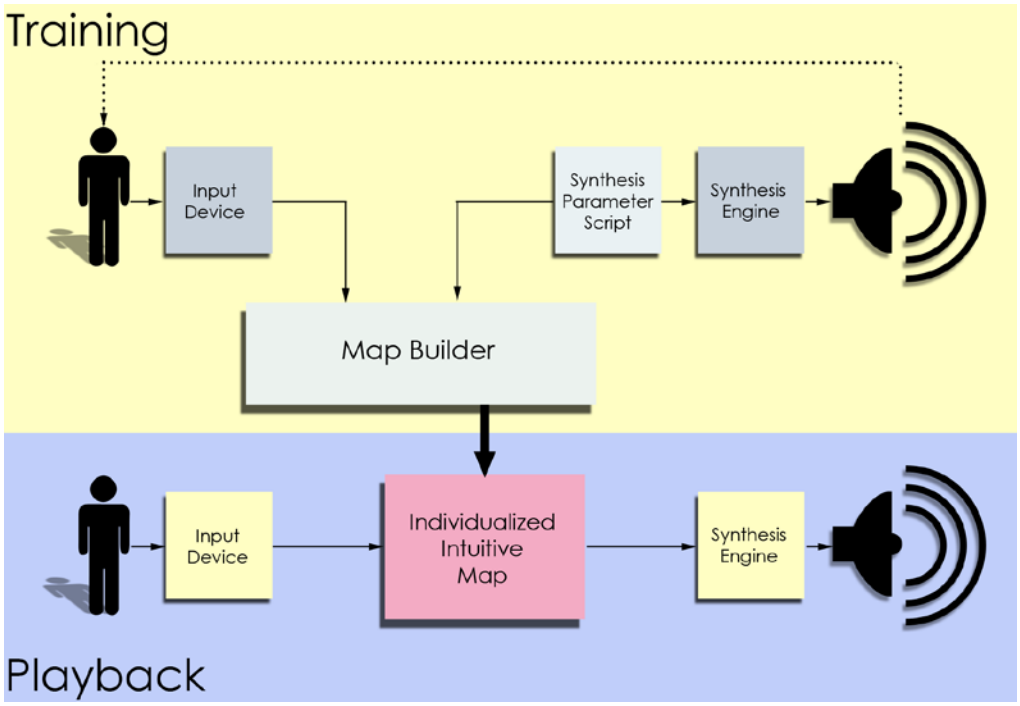

Figure 2 Entire Process of Capturing Pantomimes and Building an Individualized Intuitive Map

WACOM TABLET

The Wacom Tablet is a practical input device. While obviously intended for drawing, it was chosen because it has proven itself an instrument capable of musical expression, yet not tied down to any particular synthesis engine or mapping [35]. Additionally, since it lacks a resemblance to any analog instruments, it implores varied modes of musical interaction from user to user. Six parameters are captured from the Wacom Tablet, the pen's X position, Y position, Z position, tip pressure, tilt in the X direction and tilt in the Y direction as shown in Figure 3.

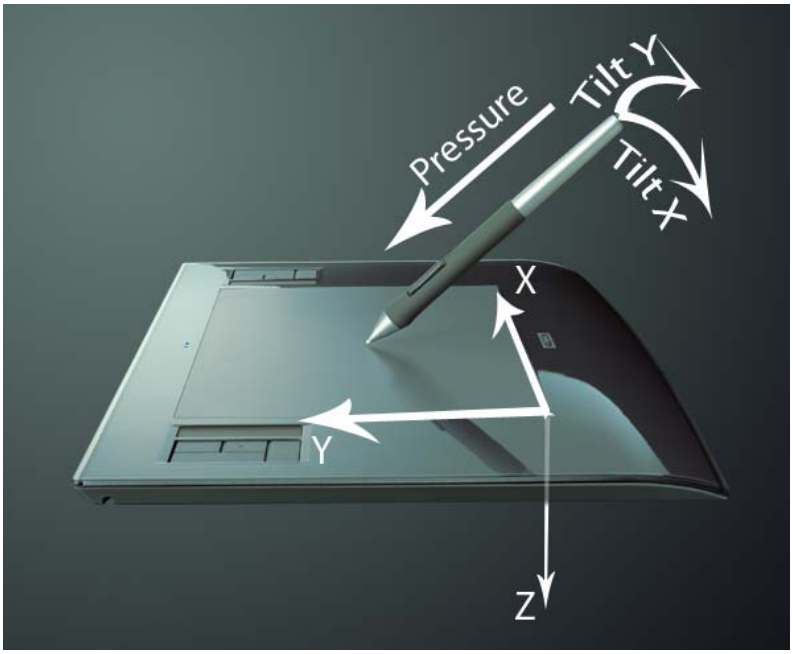

Figure 3 Parameters Captured From the Wacom Tablet

SPECTRAL MODEL SYNTHESIS

Spectral Model Synthesis (SMS) offers an equally flexible synthesis engine. Its control parameters take the form of sinusoidal amplitudes, sinusoidal frequencies and noise coefficients [28]. At any moment in time, these parameters have minimal correlation to the perceived qualities of the sound they produce. Rather, the gestalt of the synthesized audio emerges from the collective qualities in a series of control parameters presented to the synthesizer over time. Each set of sinusoidal and noise coefficients is referred to as a SMS frame as shown in Figure 4. SMS has an advantage over many other synthesis engines because it is companioned with an analysis engine that generates SMS parameters from audio wave files. The SMS file resulting from the audio analysis can be used as a script of synthesis parameters. Thus, a script of synthesis control parameters can be derived from any prerecorded sound. Throughout this text the terms SMS file and SMS script will be used interchangeably to refer to the same collection of consecutive SMS frames generated from the SMS analysis engine.

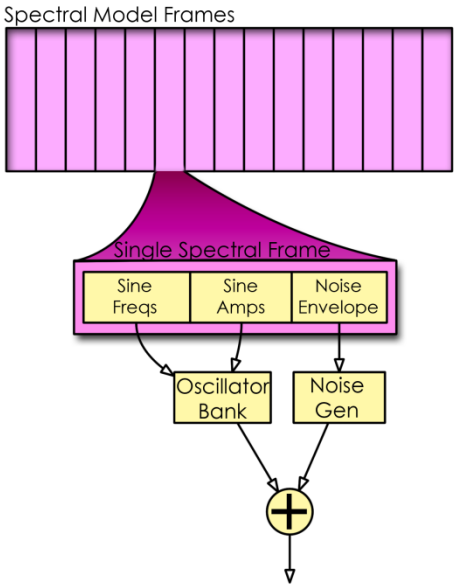

Figure 4 Spectral Model Synthesis Frame

SOFTWARE

The *Intui-Synth* software, shown in Figure 5, provides a simple environment where users can load SMS files, practice pantomiming to a synthesized SMS file and capture their pantomimed motion as the file is played back. Additionally, it projects the instantaneous parameters of the Wacom Tablet onto a canvas so that users may receive visual feedback on the controls they input. The software also implements the algorithm described in the following sections of this paper and allows for playback of synthesized audio.

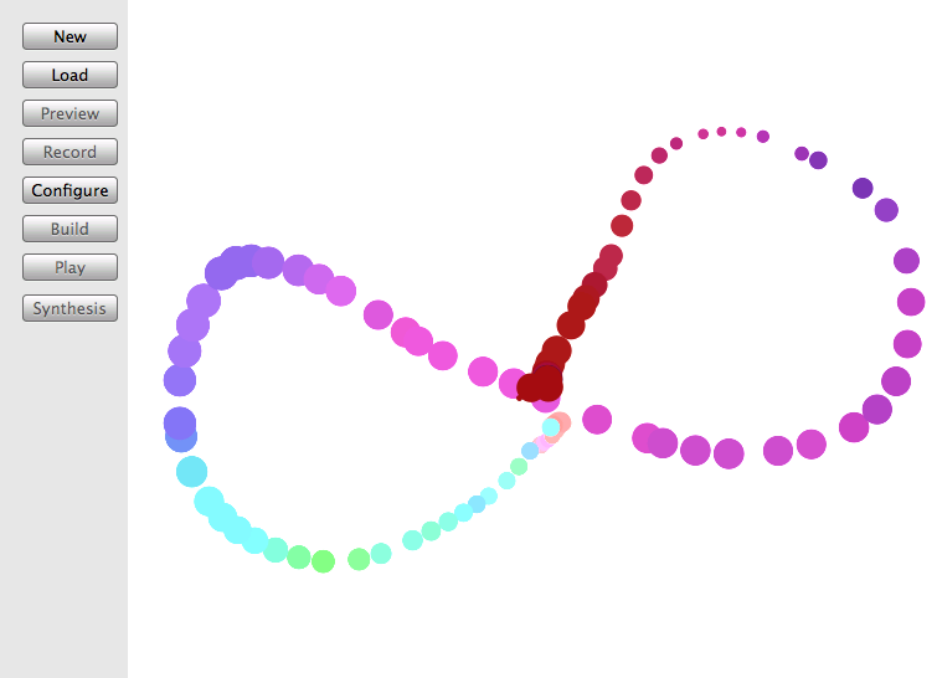

Figure 5 Intui-Synth Software

PANTOMIME, CAPTURE AND PAIR

The later components of this mapping algorithm rely on accurate pairing of input device parameters and synthesis engine parameters to deduce an effective mapping between the two. Achieving this pairing is by no means a trivial task. Inaccuracies in pairing primarily result from temporal inaccuracy and a user's unfamiliarity with the synthesis script. A stop-light starting signal, visual feedback of input device control parameters, and a means to preview the synthesized SMS frames abate these errors by preparing the user before their pantomime is captured.

Gathering pairs of input device parameters and synthesis engine parameters requires the user to have advanced knowledge of the sounds they will pantomime. Therefore, the software allows the user to preview the audio derived from the SMS file and receive visual feedback on the input controls produced while they pantomime. The visual feedback consists of a trail of slowly fading dots upon a white canvas on the computer screen. These dots encapsulate all of the parameters captured from the Wacom Tablet. The X and Y parameters correspond to the position of the dot, the pen-tip pressure and Z parameters are related through hue, and the X-tilt and Y-tilt parameters control the size of the dot as shown in Figure 6. The most saturated dot represents the most recent set of parameters while dots created from past sets of parameters fade closer to white with the onset of each new set of parameters.

Figure 6 Visual Feedback of Pen Parameters

The first software prototype began capturing Wacom Tablet parameters and synthesizing the SMS file immediately after the user clicked the record button. Initial tests of this method uncovered a need for a feedback method to prepare the user for the beginning of playback. Introducing a stop-light metaphor relieved this issue. When the user clicks the record button a large circle displays on screen and changes from red, to yellow to green in one second intervals. After the green circle displays for one second, it disappears from the screen and audio playback begins.

Finally, to create a set of tablet parameters paired to SMS frames, sets of parameters from the tablet are captured and time-stamped as the user pantomimes to the synthesized audio. The tablet parameters are then paired to the SMS frame by comparing the time stamp of the tablet parameters to the time within the script an SMS frame occurred. Those with the least amount of time difference are matched together. From here on, the act of pressing record and pantomiming once to a single SMS script will be described as a pantomime performance. The collection of pantomiming performances done on a single SMS script will be referred to as a pantomime session.

Gesture Language Extraction and Transformation

Given the opportunity to pantomime to a script of SMS frames, each user will assuredly perform different gestures to the same synthesized audio. Additionally, one would expect widely differing gestures for a single user pantomiming to different SMS scripts [7][8]. These gestures reflect the way in which a user would intuitively control the sound if they were producing it, rather than merely pantomiming to it. In order to create individualized and intuitive control of SMS parameters, features likely to express musical control are extracted from the captured Wacom Tablet parameters. We assume that features contain a high variance amongst the set of captured input device parameters encapsulate a high expressive potential. To this end, we define our gesture language as several linear combinations of features with high expressive potential and derive the gesture language using Weighted Principal Component Analysis (WPCA). Thus, transforming the features calculated from the Wacom Tablet into the gesture language emphasizes features with high expressive potential and deemphasizes those with lower expressive potential. It also serves a second purpose of reducing the dimensionality of the information used by the learning algorithms present in later stages of this mapping algorithm.

Feature Extraction

Extracting features from the input device parameters extends the ability of our mapping algorithm to encapsulate and translate gestures. Extracted features can perform non-linear calculations that reflect the way in which a performer may interact with their instrument. The goal

is not to extract the exact features a performer would use to express a gesture, but rather to collect a set of features likely to contain high expressive potential. Attempting to extract a concise, relevant, immediate and descriptive set of features, the extracted features include the current state, velocity (first derivative), acceleration (the second derivative) and a brief history (wavelet coefficients) of the Wacom Tablet parameters.

For the sake of mathematical clarity, throughout this paper scalar values are denoted by a lower case italicized variable such as *x*. A lower case bolded variable such as **y** refers to either a row or column vector and an uppercase bolded variable such as **Z** defines a matrix. Subscripts generally describe the position of an element within a vector or matrix, though at times they may denote a modified version of a vector or matrix. Arguments within square brackets, such as j in **f**[j], should be read as, "the element at position j in vector **f**."

We define the parameters capture from the Wacom Tablet in the following manner,

$$x_1(n) = x\ position\ of\ pen\ at\ sample\ n,$$
$$x_2(n) = y\ position\ of\ pen\ at\ sample\ n,$$
$$x_3(n) = z\ position\ of\ pen\ at\ sample\ n,$$
$$x_4(n) = pressure\ of\ pen\ tip\ at\ sample\ n,$$
$$x_5(n) = x\ tilt\ of\ pen\ at\ sample\ n,$$
$$x_6(n) = y\ tilt\ of\ pen\ at\ sample\ n,$$
$$t(n) = \ relative\ time\ from\ begining\ of\ audio\ playback\ at\ sample\ n.$$

Here, *n* is used to denote the order in which Wacom Tablet parameters are captured. Therefore, *n = 0* denotes the first set of parameters received, *n = 1* the second set, *n = 3* the third, and so on.

Before extracting features from the Wacom Tablet parameters, they are arranged in the column vector

$$\mathbf{x}(\mathrm{n}) = \ [x_1(\mathrm{n}), x_2(\mathrm{n}), \dots\ x_6(\mathrm{n})]'.$$

The first and second derivatives of each individual parameter are approximated by first- and second-order differences as follows

$$x_i'(\mathrm{n}) = \left(\frac{x_i(\mathrm{n}) - \ x_i(\mathrm{n}-1)}{\mathrm{t(n)} - \mathrm{t(n}-1)}\right)$$

$$x_i''(\mathrm{n}) = \left(\frac{x_i'(\mathrm{n}) - \ x_i'(\mathrm{n}-1)}{\mathrm{t(n)} - \ \mathrm{t(n}-1)}\right).$$

To calculate the wavelet transform for each sample *n* and each Wacom Tablet parameter x_i, the past N values of a particular parameter x_i is place in a vector $\mathbf{a}_i(n)$.

$$\mathbf{a}_\mathrm{i}(\mathrm{n}) = [x_i(\mathrm{n}), x_i(\mathrm{n}-1), \dots x_i(\mathrm{n}-\mathrm{N}+1)]' \quad \text{for N} = 2^\mathrm{j}, \mathrm{j} = [1,2,3\dots)$$

Here, N is variable and chosen by the user, but must be a power of 2. If N=0, then no wavelet transform is computed. The output of the wavelet transform is captured in the row vector

$$\mathbf{d}_{\mathrm{i}}(\mathrm{n}) = [d_{i1}, d_{i2}, \ldots d_{iN}] \quad \text{for N} = 2^{\mathrm{j}}, \mathrm{j} = [1,2,3 \ldots),$$

and calculated using the discrete wavelet transform

$$\mathbf{d}_{\mathrm{i}}(\mathrm{n}) = \mathrm{DWT}\{\mathbf{a}_{\mathrm{i}}(\mathrm{n})\}.$$

The intention of the wavelet transform is to capture characteristics of a parameter over time. Once might assume that this could be achieved simply by $\mathbf{a}_{\mathrm{i}}(\mathrm{n})$ since both contain the same type of information. Unfortunately, $\mathbf{a}_{\mathrm{i}}(\mathrm{n})$ conflicts with latter gesture language extraction stage. As a feature, wavelet coefficients were chosen over a simple delay line because they offer a more sparse representation of the time dependent data. As well, by transforming a delay line of a single parameter using the discrete wavelet transform, the information within the delay line is decorrelated from the current value of the parameter. This decorrelation is important for the gesture language extraction which views highly correlated values as expressing the same gesture. While an in-depth discussion of the discrete wavelet transform is beyond the scope of this text, it should be noted that the Harr wavelet was used to perform the discrete wavelet transform.

The features for a single Wacom Tablet parameter are placed within the row vector

$$\mathbf{f}_{\mathrm{i}}(\mathrm{n}) = [x_i(\mathrm{n}), x_i^{'}(\mathrm{n}), x_i^{''}(\mathrm{n}), \mathbf{d}_{\mathrm{i}}(\mathrm{n})],$$

and the features from all the Wacom Tablet parameters are collected into the feature column vector

$$\mathbf{f}(\mathrm{n}) = [\mathbf{f}_1(\mathrm{n}), \mathbf{f}_2(\mathrm{n}), \ldots \mathbf{f}_6(\mathrm{n})]'. \tag{1}$$

Supposing that M samples of the Wacom Tablet parameters were collected for a single pantomime performance, the entire feature matrix can be described as

$$\mathbf{F}_k = [\mathbf{f}(\mathrm{n}), \mathbf{f}(\mathrm{n}-1), \ldots \mathbf{f}(\mathrm{n}-\mathrm{M}+1)].$$

Here, the subscript *k* describes the features collected for a particular pantomiming performance. That is, the user may choose to record several pantomiming performances using the same SMS script in order to provide additional data for the gesture language and gesture to SMS frame mapping stages of the algorithm. Before deriving the gesture language, feature matrices from all pantomime performances within the pantomime session are collated end to end in the session feature matrix

$$\boldsymbol{\Gamma} = [\mathbf{F}_1, \mathbf{F}_2, \ldots \mathbf{F}_{\mathrm{L}}], \tag{2}$$

for L pantomime performances.

GESTURE LANGUAGE

We aim to determine a gesture language that encapsulates the features that show high potential expressivity for a particular user, as well as a means to transform the features extracted from the Wacom Tablet into the gesture language in a real-time fashion. We make the assumption that features that show high variance during the pantomiming session posses a high potential for expressivity. This assumption is backed by previous work showing that users pantomiming to

music correlate energetic motion with musical control [8][33]. As well, we assume that features with a high correlation have similar gestural meaning. Given these inferences and requirements, Principal Component Analysis (PCA) provides a means to determine a transformation to a gesture language of N continuous values where the first value contains the most expressivity, and the second value contains the second most expressivity, the third containing the third most expressivity and so on. Additionally, we emphasize/deemphasize the values in the gesture language by weighting each of the N values by the respective variance associated with it. We refer to the combination of PCA with the proposed weighting as Weighted Principal Component Analysis (WPCA).

As described in this paper, WPCA is primarily viewed as a means to extract a mapping to a gesture language from the session feature matrix. It should be noted that PCA has a long history of use within the machine learning domain. Many machine learning algorithms suffer from the curse of dimensionality [10]. If we imagine that a machine learning algorithm is searching an N dimensional space in order to learn generalizations from the given data, we can see that the volume it needs to search increases exponentially with each additional dimension. PCA can be used as a dimensionality reduction technique, as it takes in a set of features of high dimension and projects them down to a lower dimension in a way that preserves the most energy of the original signal in a mean squared error sense. Dimensionality reduction serves two significant purposes in our case. First, it reduces the volume of the search space a machine learning algorithm must look for a solution. Second, it generally improves speed and memory performance in both the training and transduction stages of a learning algorithm.

While PCA is a well known and well studied method, it is presented it here to illuminate how it derives the transformation from a high dimensional space to a lower dimensional space, as well as the modifications made to traditional PCA methods to help it reflect the attributes desired in a gesture language transformation.

Using PCA we intend to derive a transformation from a feature vector to a gesture vector of the form

$$\mathbf{g}(n) = \mathbf{P}\mathbf{f}(n),$$

$$\mathbf{g}(\mathrm{n}) = [g_1(\mathrm{n}), g_2(\mathrm{n}), \ldots\ g_N(\mathrm{n})]$$

Here, $\mathbf{f}(n)$ is the feature vector as defined in equation (1) of M dimensions, $\mathbf{g}(n)$ denotes our gesture vector of N dimensions, and $\mathbf{P}$ is a N by M matrix transforming $\mathbf{f}(n)$ into $\mathbf{g}(n)$ and encapsulates our gesture langauge. To determine $\mathbf{P}$ we utilize the set of Wacom Tablet features held in session feature matrix Γ as described in equation (2),

$$\mathbf{\Gamma} = [\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_\mathrm{K}],$$

where K is the total number of feature vectors extracted during the pantomiming session. First, a column vector containing the mean of each row in the session feature matrix is calculated by

$$\bar{\mathbf{y}} = \frac{1}{\mathrm{K}}\sum_{\mathrm{i}=1}^{\mathrm{K}} \mathbf{y}_{\mathrm{i}}$$

The mean is extracted from each column in the session feature matrix, resulting in the centered session feature matrix

$$\mathbf{\Gamma}_{\mathrm{c}} = \mathbf{\Gamma} - \bar{\mathbf{y}}$$

Using Singular Value Decomposition (SVD) we can decompose the matrix into

$$\mathbf{U\Sigma V}^{*} = \mathbf{\Gamma}_{\mathrm{c}} \tag{3}$$

where

$$\mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \boldsymbol{u}_M],$$

$$\boldsymbol{u}_i = [u_{i1}, u_{i2}, \ldots\ u_{iM}]',$$

The columns of **U** represent orthogonal, unit length, principal components of the session feature matrix **Γ** and are ordered from most principal component to least principal component. That is, if one projected **Γ** onto $\boldsymbol{u}_1$' the most information would be preserved, $\boldsymbol{u}_2$' the second most information, $\boldsymbol{u}_3$' the third most information, and so on.

Generally PCA stops here and a transformation matrix is concocted out of the first N columns of **U**. While this could provide a mapping between the feature vector **f**(n) and the gesture vector **g**(n) it would not describe the expressivity of the features other than to say the features projected onto **P** and encapsulated in $g_1(n)$ contain the most expressivity, $g_2(n)$ the next most, and so on. Returning to equation (3), the output of SVD includes

$$\mathbf{\Sigma} = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_M \end{bmatrix}$$

where λ_i relates to the variance of the features projected onto $\mathbf{u}_i$. Since we have defined features with a higher variance to contain a higher expressivity, we multiply the columns of **U** by their respective λ_i. This effectively emphasizes the features containing high expressivity, and deemphasizes those containing low expressivity. Finally, the transformation matrix takes the form

$$\mathbf{P} = [\lambda_1\boldsymbol{u}_1, \lambda_2\boldsymbol{u}_2, \ldots \lambda_N\boldsymbol{u}_N]^T \; for \; N \leq M,$$

and is used to translate feature vectors into gesture vectors via

$$\mathbf{g}(n) = \mathbf{P}\mathbf{f}(n). \tag{4}$$

We defined this modified version of PCA Weighted Principal Component Analysis (WPCA).

Mapping Gestures to Spectral Model Synthesis Frames

The non-trivial relation between gesture vectors and SMS frames requires a flexible framework able to deduce a mapping between the two. Previous works have used k nearest neighbors (KNN) and artificial neural networks (ANN) to derive a mapping between high level sonic descriptors, such as root frequency and loudness, and additive synthesis parameters [31]. While gestures and high level sonic descriptors are not equivalent, the demonstrated ability of KNN and ANN to perform such complex mappings in real-time make them prime candidates for mapping gestures to SMS frames. KNN was chosen because it can be built quickly in comparison to other machine learning techniques, make few assumptions about the underlying data and provides control over its variance and bias. ANN was chosen for its ability to map complex relations, low computational complexity in the transduction stage and ability to generalize beyond the training data. Additionally, KNN and ANN provide a useful contrast as they make significantly different assumptions about the mapping being learned.

K Nearest Neighbors

K-Nearest Neighbors (KNN) techniques are an important cornerstone in machine learning as they rely on very few assumptions about the underlying data [10]. They work on the assumption that data can be arranged into a metric space, and that a new, unclassified piece of data can best be described by inspecting the K nearest classified data within a training set. This property is extremely attractive in that it allows us to use our gesture vectors as direct predictors of the output SMS frame, ignoring the complex relationship between the gesture vector and the particular format of the SMS frame. The relatively small amount of time it takes to build a KNN map from training allows for fast experimentation with varying feature extraction settings and pantomiming performances. Also, in this implementation of KNN, the number of neighbors inspected, K, is easily varied. Thus, the bias and variance of the KNN algorithm can be altered so that high variance/low bias and vice a versa situations can be inspected.

When mapping between gestures and SMS frames, the gesture vectors correspond to the locations of SMS frames within a Euclidean space. Building the KNN algorithm requires placing the SMS frames within that Euclidean space based off of the gesture vectors paired to them. When a new gesture vector is presented, KNN retrieves the K nearest SMS frames based upon a Euclidean distance metric between the presented gesture vector and the gesture vectors already present within the KNN space. When K is greater than 1, the output SMS frame is derive from a weighted average of the K retrieved input frames. For a further discussion of calculating the weighted average of several SMS frames, see the Appendix.

Artificial Neural Networks

An Artificial Neural Network (ANN) is a machine learning model motivated by biological neural networks. It consists of interconnected layers of artificial neurons. It is utilized here because ANNs offer an encouraging mathematical model for inferring relationships between complex sets of data such as a gesture vector and a SMS frame. While ANNs can spend a large amount of time in the learning phase, they are fairly quick and efficient to implement the transduction stage. This bodes well for our audio application where the audio graph must be traversed in less than 10 milliseconds. Depending upon the configuration of the ANN, they generally provide continuous transitions with continuous first derivatives. This allows smooth

transitions from one SMS frame to another when the input gesture vector also contains smooth transitions. This avoids unwanted abrupt transitions which would produce the false sense of an onset. Unfortunately, ANN's non-linearity results in an error surface with many local minima, making an optimum configuration difficult to converge upon.

A deep discussion of ANN is beyond the scope of this text, yet a brief introduction is presented here familiarize the reader with this model. The main building block of an ANN is the neuron shown in Figure 7. It consists of several inputs, weights for each of those inputs, a bias and a transfer function.

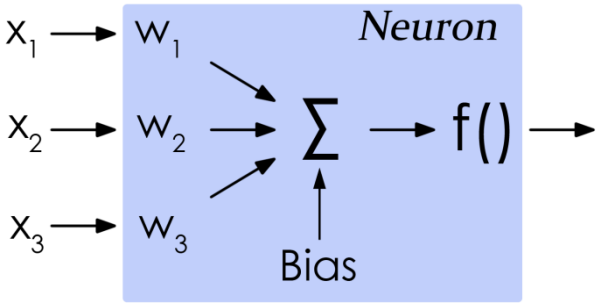

Figure 7 A Single Neuron

During the transduction stage, the inputs to the neuron (x_1, x_2, x_3) are multiplied by their respective weights (w_1, w_2, w_3). Note, the number of inputs to a neuron need only be one or more, not necessarily three. These values are summed, along with the neurons bias, and given to the transfer function. A variety of transfer appropriate functions exist, yet the most widely used is the sigmoid function shown in Figure 8

$$f(x) = \frac{1}{1 + e^{-x}}$$

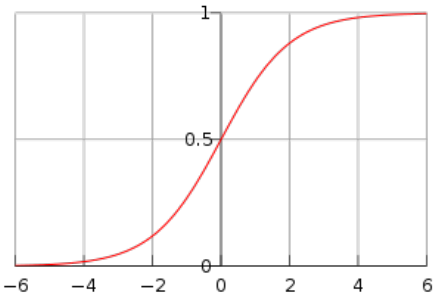

Figure 8 The Sigmoid Function

and the linear function

$$f(x) = ax$$

The nature of the ANN arises out of the connection of several layers of neurons. There are an infinite number of ways to arrange these neurons, but by far the most common formulation is of a three layer feed forward network displayed in Figure 9.

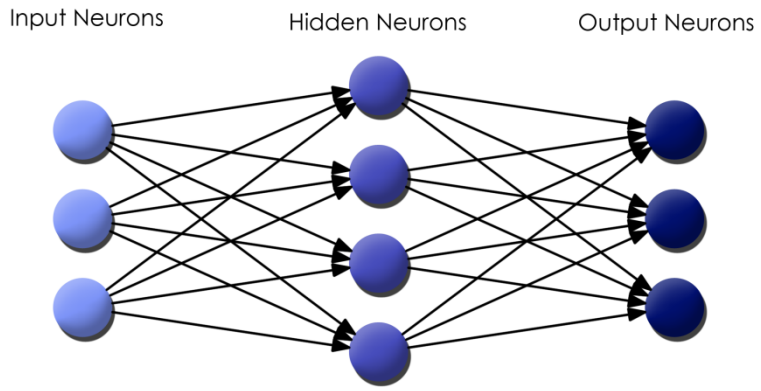

Figure 9 Three Level Feed Forward Artificial Neural Network

The number of input, hidden and output neurons may be altered based upon a user's need. To train an ANN, pairs of inputs and their desired outputs are presented to the ANN. The predicted output and the desired output are compared, and the error between the two is propagated from the output neurons to the input neurons, updating the weights and biases of each neuron in the net. This process is more commonly known as back-propagation. Pairs of inputs and desired outputs are repeatedly used to update the ANN through back-propagation until it has satisfactorily converged to a useful collection of neuron weights and biases.

In this work, three feed forward ANNs attempt to map between gestures and the separate components of an SMS frame. The first ANN maps between gestures and sinusoidal frequencies, the second between gestures and sinusoidal amplitudes, and the third between gestures and stochastic coefficients. Three separate ANNs were used to help the ANN cope with the complex format of the SMS frame. For instance, the sinusoidal frequencies contain an extremely sparse representation with few varying parameters and many zeros, while the noise coefficients are not sparse and contain many varying parameters. If only one ANN were used, the ANN would be tasked with not only learning a mapping from a gesture vector to an SMS frame, but also the structure of the SMS frame.

The values of the individual components within the SMS frame are perceptually weighted before being presented to the output of the ANN in order to more closely approximate a linear scale of human sensitivity. Each sinusoidal frequency is transformed

$$f_p = \begin{cases} \log_2 f, & 20 \geq f \leq 20000 \\ 4, & otherwise \end{cases}$$

where f denotes the frequency value within the SMS frame and f_p is the value presented to the output of the ANN. The sinusoidal amplitudes and stochastic coefficients weighted logarithmically by

$$a_p = \begin{cases} \log_{10} a, & 0.00001 \geq a \\ -5, & otherwise \end{cases}$$

where a denotes either the sinusoidal amplitude or the stochastic coefficient within the SMS frame and a_p is the value presented to the output of the ANN. For both perceptual weightings, an equivalent inverse is used to interpret the output of the ANNs and convert them into SMS frames.

The ANNs are trained until a minimum mean squared error is achieved for a particular ANN, or the maximum number of training iterations has completed. Both of these values can be controlled by the user.

A Complete Individualized Intuitive Mapping

Up to this point, the process of creating an intuitive, individualized mapping between the Wacom Tablet and Spectral Model Synthesis frames includes the following steps:

1. Capture the Wacom Tablet parameters as the user pantomimes to a script of SMS frames.
2. Derive a mapping to a gesture language by extracting possibly relevant features from the captured tablet parameters, performing Weighted Principal Component Analysis upon those features and creating a linear map from the features to gestures using the N most principal components weighted by the variance of the features along that component.
3. Using pairs of input gestures and target SMS frames, train either an Artificial Neural Network or K Nearest Neighbor learning algorithm.

The complete, real-time mapping between Wacom Tablet parameters to SMS frames emerges by logically arranging these components. First, features from each new set of parameters from the Wacom Tablet are extracted just as described in the previous *Feature Extraction* section of this text. As each new set of features is calculated, they are projected into the gesture language using equation (4). Then, either KNN or ANN translates the resulting gesture vector into an SMS frame.

Initial experiments with this design produced nearly adequate results. For the most part, the mapping algorithm produced SMS frames with a satisfactory correlation to the input gestures, yet the coherence of the audio was negatively affected by spurious SMS frames creating unintended transients and onsets. To combat this issue, an SMS frame smoothing stage receives frames from the output of the learning algorithm and precedes the synthesis engine. In software, the update rate of the learning algorithm is tied to the refresh rate of the Wacom Tablet. It determines a new SMS frame every 60th of a second. Each of these new frames are combined with the current frame within the synthesis engine by taking a weighted average of the two (see Appendix) where the weights are based off of a smoothing factor. The weighted average of these frames produces the new output SMS frame to be synthesized. We denote the most recent SMS frame output from the learning algorithm as $\mathbf{x}(n)$, the previous SMS frame within the synthesis engine as $\mathbf{y}(n\text{-}1)$ and intend to compute the current state $\mathbf{y}(n)$ using a smoothing factor α.

$$0 \leq \alpha \leq 1$$

$$\mathbf{y}(n) = \alpha\mathbf{y}(n-1) + (1-\alpha)\mathbf{x}(n)$$

Hence, when α equals zero, there is no smoothing, and when α equals one the SMS frame used by the synthesis engine does not change.

RESULTS

Analysis of the proposed design lends itself much more to qualitative analysis rather than quantitative analysis because it seeks to derive an individualized mapping from pantomimes to synthesized audio. The empirical equivalence between a pantomime and a sound is virtually impossible to extract due to the inherent subjectivity of both pantomimes and sound perception. Therefore, analysis is primarily focused upon the entirety of the design. Spotlight is given to the performance of the design for varying types of source sounds, the performance of WPCA compared to traditional PCA, and a comparison between the KNN and ANN learning algorithms.

As a whole, the design succeeds in meeting the design goals presented earlier. Still, exploring the algorithm under varying circumstances spreads light upon possible future work. To this end, the algorithm is tested using a variety of SMS scripts to explore how it performs using SMS scripts derived from slowly changing sounds, fast percussive sounds and sounds with rapidly varying qualities such as a tremolo. The effect of the weighted PCA compared to the traditional PCA is investigated to validate its use within the algorithm. Additionally, a comparison between KNN and ANN illuminates the performance differences between the two.

VARYING SONIC SOURCES

A mapping was derived using SMS scripts that originated from three separate sound sources. The first script was derived from a recording of an opera singer singing four notes in succession. The second script came from a recording of a growling dog. The third script originated from simple kick, snare and high-hat drum loop. Each script was pantomimed to four times and a mapping derived for each set of pantomime session. The most convincing mapping came out of the pantomimes to the opera singer. The mapping preserved the pitches and timbre of her voice when attempting to recreate the original sound. The least convincing mapping originated from the drum loop. The sounds reproduced only resembled the original sound in that a loud noise emitted from the speakers whenever the Wacom Tablet could sense the pen within its vicinity. The mapping extracted from the dog growl pantomimes produced a mapping that allowed the user to recreate the timbre of a dog growl, but failed to recreate the nuanced trembling created in the dog's throat.

These results bring two main points to the forefront. First, the act of pantomiming provides grand gestures for control, but cannot capture fine control of quickly changing sonic features. This is evident particularly in the dog growl example. It is nearly impossible to recreate the exact undulations of the dog's throat gurgling with the pen simply because the gurgles alter too quickly in time. Second, the Wacom Tablet provides an interface much more suited for slower harmonic gestures rather than percussive gestures. Every time the pen is lifted more than two inches above the tablet drawing surface, the tablet loses communication with the pen. This makes it difficult to track the pen as if it was being used as a drum stick. As well, the sampling rate of the Wacom Tablet (approximately 60 Hz) fails to capture the nuances of a pen striking its surface.

WPCA vs. Traditional PCA

A simple test was performed where the user pantomimed to a sinusoidal frequency slowly ascending and descending. The pantomime was performed to specifically suggest that the position of the pen along the diagonal of the tablet surface should control the frequency of the sinusoid. Mappings were derived using both weighted PCA and traditional PCA methods in order assess their ability to emphasize the diagonal position of the pen tip as a control of frequency and deemphasize other gestures' control of frequency. Anecdotally, WPCA far outperformed traditional PCA. While WPCA was robust to motions along the perpendicular diagonal and tilts in the pen, traditional PCA produced wild alterations in frequency as the pen was tilted in the x or y directions and as the pressure of the pen tip altered.

KNN vs. ANN

KNN and ANN algorithms vary greatly in the manner they learn a mapping, the computational costs they impose and the assumptions they make about underlying models. None-the-less, focus is given here to their qualitative outputs rather than the manner in which they are implemented. The resulting differences between KNN and ANN greatly reflect earlier results deduced in an experiment attempting to learn a mapping between high level sound descriptors and additive synthesis parameters [31]. KNN results in sounds much more closely resembling the original audio. This is to be expected since KNN retains all the SMS frames used to build it. While new SMS frames are extracted by a weighted average of K SMS frames, generally these neighboring frames are very similar to each other. As a result the frames generated by KNN do not vary thematically from the original frames used to build the KNN. ANN, on the other hand, produces SMS frames that can wildly fluctuation from the training SMS frames. This has both the upside of allowing additional exploration of timbres and pitches, and downside of creating sounds the do not resemble the original sound. Rather they resemble the underlying representation of an SMS frame where the collective emergence of a sonic gesture is replaced by a collection of varying sinusoids and noise envelopes.

Future Work

The demonstrated ability to generate an individualized and intuitive mapping between pantomimes and sound encourages further research into this domain. Several areas stand out. The use of different input devices may purport additional aspects of pantomimes and allow for a richer gesture language. Extracting additional features from the input device could have similar effects upon the gesture language. Lastly, mapping between gestures and high level sonic descriptors, rather than low level synthesis engine controls, may more accurately describe the sonic attributes that motivate the pantomimes resulting in a more accurate mapping.

Conclusion

A method to extract an intuitive and individualized mapping for real-time synthesis between Wacom Tablet parameters and Spectral Model Synthesis frames has been described and implemented. The method works by capturing tablet parameters as the user pantomimes to

synthesized Spectral Model Synthesis frames. A gesture language is obtained from pairs of SMS frames and tablet parameters by first extracting features from the tablet parameters and then assigning a high expressive potential to those features displaying high variance. Once the gesture language is calculated, an artificial neural network or a k nearest neighbor algorithm learns a map between gestures and Spectral Model Synthesis frames. Analysis of the design shows that it works best for slowly varying harmonic sounds and worse for quick percussive sounds due to the inability of a pantomime to express quickly varying changes in the Spectral Model Synthesis frame, as well as the limited interface of the Wacom Tablet. Weighted Principal Component Analysis was show to outperform traditional Principal Component Analysis by deemphasizing features that did not relate to any gestures. Lastly, a comparison between k nearest neighbors and artificial neural networks concluded that k nearest neighbors offers sounds similar to the original sound used to create the Spectral Model Synthesis frames, while artificial neural networks create more varied frames that allow a larger exploration of sonic space but do not exhibit much resemblance to the original sound.

APPENDIX

LIST OF ACRONYMS USED

ANN – Artificial Neural Networks

KNN – K Nearest Neighbors

PCA – Principal Component Analysis

SMS – Spectral Model Synthesis

SVD – Singular Value Decomposition

WPCA – Weighted Principal Component Analysis

WEIGHTED AVERAGE OF SMS FRAMES

The non-trivial structure of SMS frames presents a unique challenge when one attempts to derive a weighted sum of several SMS frames. As well, we should attempt to combine SMS frames keeping in mind human's approximately logarithmic perception of pitch and amplitude differences.

SMS frames contain a constant number of sinusoidal frequencies and amplitudes, but not all frequencies are used when synthesizing. For instance, a frame may contain 50 sinusoidal tracks where the first 10 contain valid frequency values, and the remaining 40 equal zero. If two SMS frames are averaged where one has 10 valid frequencies and the other has 12 we accommodate the varying number of valid frequencies by ignoring all frequency values equal to zero and their corresponding amplitudes. Aided by the fact that all non-zero frequency values are placed in ascending order, we can be assured that general trends in frequency ascension are preserved. A single sinusoidal trajectory can be described by

$$a_i \sin(2 * \pi * f_i / f_s)$$

where a_i is the amplitude of the sinusoidal track, f_i its frequency and f_s a constant sampling rate. An SMS frame consists of two vectors

$$\mathbf{a}_\mathrm{j} = [a_1, a_2, \ldots\ a_N]$$

$$\mathbf{f}_\mathrm{j} = [f_1, f_2, \ldots\ f_N]$$

where the subscript j denotes the SMS frame the vectors originate from and N is the number of sinusoidal tracks within the frame. Given M SMS frames, with weights w_1 to w_m, we determine the weighted combination of their frequencies by

$$\mathrm{fcount}(f) = \begin{cases} 1, & f > 0 \\ 0, & otherwise \end{cases}$$

$$\mathrm{fweight}(f) = \begin{cases} \log_2 f\,, & f > 20 \\ 0, & otherwise \end{cases}$$

$$z_i = \frac{\sum_{\mathrm{j}=0}^{\mathrm{M}} \left(w_i * \mathrm{fweight}\left(\mathbf{f}_\mathrm{j}[i]\right)\right)}{\sum_{\mathrm{j}=0}^{\mathrm{M}} \left(w_i * \mathrm{fcount}\left(\mathbf{f}_\mathrm{j}[i]\right)\right)}$$

$$\mathbf{f}_\mathrm{o} = [2^{z_1}, 2^{z_2}, \ldots\ 2^{z_N}]$$

and the amplitudes by

$$\mathrm{aweight}(a) = \begin{cases} \log_{10} a\,, & a > 0.00001 \\ -5, & otherwise \end{cases}$$

$$p_i = \sum_{\mathrm{j}=0}^{\mathrm{M}} \left(w_i * \mathrm{aweight}\left(\mathbf{a}_\mathrm{j}[i]\right)\right)$$

$$\mathbf{a}_\mathrm{o} = [10^{p_1}, 10^{p_2}, \ldots\ 10^{p_N}]$$

where $\mathbf{a}_\mathrm{o}$ and $\mathbf{f}_\mathrm{o}$ are the frequencies and amplitudes of the output SMS frame. Note that for the special case when $\mathbf{f}_\mathrm{j}[i] = 0$ for all j, the output frequency $\mathbf{f}_\mathrm{o}[i]$ and amplitude $\mathbf{a}_\mathrm{o}[i]$ are set to zero. As well, the weighted sum of stochastic coefficients is performed identically to the weighted sum of sinusoidal amplitudes.

REFERENCES

[1] Arfib, D., J. Couturier, L. Kessous, and V. Verfaille. Strategies of Mapping Between Gesture Parameters and Synthesis Model Parameters Using Perceptual Spaces. *Organised Sound*, 7(2), 2002.

[2] Choi, I., R. Bargar, and C. Goudeseune. A Manifold Interface for a High Dimensional Control Space. *Proc. 1995 Int. Computer Music Conf.*, pp. 385–392.

[3] Crab, S. *120 Years Of Electronic Music*. Web. 10 Dec. 2009. http://120years.net.

[4] Dannenberg, R., B. Thom, and D. Watson. A Machine Learning Approach to Musical Style Recognition. In *Proceedings of the ICMC* (Thessaloniki, Greece, 1997).

[5] Dubnov, S. and X. Rodet. Statistical Modeling of Sound Aperiodicities, *Proc. ICMC, Tessaloniki* :43-50, 1997.

[6] Fels, S. and G. Hinton. Glove-talki : A Neural Network Interface Which Maps Gestures to Parallel Formant Speech Synthesizer Controls. *IEEE Transactions on Neural Networks*, 9(1) :205–212, 1998.

[7] Godoy, R.I., E. Haga, and A. Refsum-Jensenius. Exploring Music-Related Gestures by Sound-Tracing – A Preliminary Study. *2nd ConGAS International Symposium on Gesture Interfaces for Multimedia Systems*, Leeds, UK.

[8] Godoy, R. I., E. Haga, and A. Refsum-Jensenius. Playing 'Air Instruments': Mimicry of Sound-Producing Gestures by Novices and Experts. *Gesture in Human- Computer Interaction and Simulation: 6th International Gesture Workshop* (GW 2005). Berlin, Heidelberg: Springer-Verlag, 256–67.

[9] Goto, S., and T. Suzuki. The Case Study of Application of Advanced Gesture Interface and Mapping Interface, Virtual Musical Instrument " le superpolm" and Gesture Controller "bodysuit", *In Proceedings of the 2004 Conference on New Interfaces for Musical Expression* (NIME-04), Hamamatsu, Japan, June 3-5 2004.

[10] Hastie, T., R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2009. Print.

[11] Hsu, W. Design Issues in Interaction Modeling for Free Improvisation. *Proceedings of the 2007 Conference on New Interfaces for Musical Expression*, New York, NY, USA.

[12] Hunt, A. and M. Marcelo Wanderley. Mapping Performance Parameters to Synthesis Engine. *Organised Sound*, 7(2) :103–114, 2002.

[13] Hunt, A., M. Marcelo Wanderley, and Matt Paradis. The Importance of Parameter Mapping in Electronic Instrument Design. *Journal of New Music Research*, 32(4) :429–440, 2003..

[14] Hunt, A., M. Marcelo Wanderley, and R. Kirk. Towards a Model for Instrumental mapping in Expert Musical Interaction. *International Computer Music Conference*. 2000.

[15] Jones, R.E. Intimate Control for Physical Modeling Synthesis. M.Sc. Thesis, *University of Victoria*. 2008.

[16] Juang, B. H., and L. R. Rabiner. Hidden Markov Models for Speech Recognition, *Technometrics*, Vol. 33, No. 3 (Aug., 1991), pp. 251-272

[17] Lee, M., and D. Wessel. 1992. Connectionist models for real-time control of synthesis and compositional algorithms. *Proc. of the 1991 Int. Computer Music Conf.*, pp. 277-80, Montreal, QC, Canada. San Francisco: International Computer Music Association.

[18] Levitin, D. J., S. McAdams, and R. L. Adams. Control parameters for musical instruments: a foundation for new mappings of gesture to sound, *Organised Sound*, v.7 n.2, p.171-189, August 2002.

[19] Modler, P. Neural Networks for Mapping Hand Gestures to Sound Synthesis Parameters. *Trends in Gestural Control of Music*, CD-ROM, publication IRCAM, 2000.

[20] Momeni, A. and D. Wessel. Characterizing and Controlling Musical Material Intuitively with Geometric Models, *Proceedings of the New Interfaces for Musical Expression Conference*, :54-62, 2003.

[21] Overholt, D. The Musical Interface Technology Design Space, *Organised Sound*, 14:217-226, 2009.

[22] Paine, G. Towards Unified Design Guidelines for New Interfaces for Musical Expression. *Organised Sound* (2009), 14:142-155 Cambridge University Press

[23] Rasamimanana, N., F. Kaiser and F. Bevilacqua. Perspectives on Gesture–Sound Relationships Informed from Acoustic Instrument Studies. *Organised Sound*, 14:208-21, 2009.

[24] Reblitz, A. The Golden Age of Automatic Musical Instruments. *Mechanical Music Press*, 2001.

[25] Roads, C., J. Strawn , C. Abbott , J. Gordon , P. Greenspun. The computer music tutorial, *MIT Press*, Cambridge, MA, 1996

[26] Salomon, R., and J. Weissmann. Evolutionary Tuning of Neural Networks for Gesture Recognition, *The 2000 Congress on Evolutionary Computation,* 1528-1534, 2000.

[27] Steiner, H. Towards a Catalog and Software Library of Mapping Methods, *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*, Paris, France

[28] Serra, X., and J. O. Smith. A Sound Decomposition System Based on a Deterministic plus Residual Model. *J. Acoustic Soc*. Am, Supp. 1, 89(1) :425–34, 1990.

[29] Van Nort, D. and M. Wanderley. Control Strategies for Navigation of Complex Sonic Spaces. *Proceedings of the 2007 Conference on New Interfaces for Musical Expression*, New York, NY, USA

[30] Van Nort, D. Instrumental Listening: Sonic Gesture as Design Principle, *Organised Sound*, 14:177-187, 2009.

[31] Wessel, D., C. Drame, and M. Wright. Removing the Time Axis from Spectral Model Analysis-Based Additive Synthesis: Neural Networks versus Memory-Based Machine Learning, *International Computer Music Conference*, Ann Arbor, Michigan, 1998.

[32] Wessel, D. Timbre Space as a Musical Control Structure. *Computer Music Journal* **3**(2): 32-45, 1979.

[33] Wessel, D. An Enactive Approach to Computer Music Performance. *Le Feedback dans la Creation Musical*, pp. 93–98, 2006.

[34] Winkler, T. 1995. Making motion musical: Gesture mapping strategies for interactive computer music. Paper presented at the *Int. Computer Music Conf.*, Banff, Canada

[35] Zbyszynski, M., M. Wright, A. Momeni, and D. Cullen. Ten Years of Tablet Musical Interfaces at CNMAT, *Proceedings of the International Conference on New Interfaces for Musical Expression*, 100-105, 2007.