Soft Computing for Real-Time Control of Musical Processes

Michael A. Lee and David Wessel
Center for New Music and Audio Technologies
University of California
Berkeley, CA 94720
510 643 9990
{lee,wessel}@cnmat.berkeley.edu
http://www.cs.berkeley.edu/~leem and http://www.cnmat.berkeley.edu

ABSTRACT

In this paper, we present soft computing tools and techniques aimed at realizing musical instruments that learn. Specifically we explore applications of neural network and fuzzy logic techniques to the design of instruments that form highly personalized relationships with their users through self-adaptation. We demonstrate techniques for adapting sensor arrays and techniques for realizing highly expressive real-time sound synthesis algorithms.

1. INTRODUCTION

A musical instrument can be viewed as a tranducer that translates a musician's physical gestures into a sonic result. Effective use of this channel requires that the musician master the protocol of the instrument. Although the protocol can generally remain similar between instruments of the same family, the protocol for a particular instrument can differ in subtle, but distinct ways from that of another instrument. It is well known that a musicians will often spend a great deal of effort exploring and adapting to the capabilities of an instrument through practice and study. In fact, musicians often speak of a rather special and very personal relationship with the instrument. While it is possible for instrumentalists to physically adapt their instrument to the particuliarities of their playing style (i.e., by choosing the mouthpiece, bridge, or string guage), a radical modification can be prohibitively expensive or compromise the integrity of the instrument.

In this work, we leverage from recent advances in soft computing techniques that have been demonstrated to be effective for the identification and control of nonlinear systems to realize adaptive musical instruments [11]. Our goal is to realize techniques that foster efficient and expressive communication of musical ideas.

2. BACKGROUND

A General Instrument Architecture

Figure 1 presents a systems description of a com-

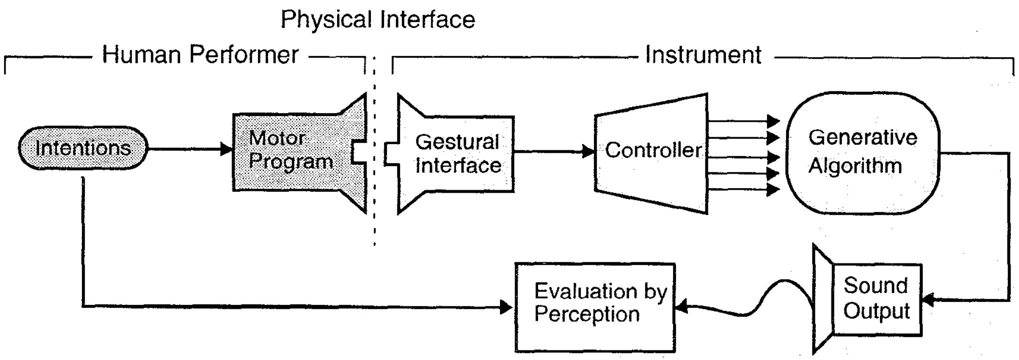

Figure 1. A generalized human performer/musical instrument system.

0-7803-2559-1/95 $4.00 © 1995 IEEE

puter-based musical instrument and its performer. The figure decomposes the instrument into three subsystems: the gestural interface, the controller, and the generative or synthesis process. The gestural interface provides the physical sensing mechanism that captures the performer's gestures. The controller is responsible for mapping from gesture space onto parameters of the underlying generative algorithm. The generative algorithm is the process that generates the sound or patterns of sound. It is worth noting here that with such computer based instruments, musicians can produce and control patterns of sound rather that just single notes as is most often the case with traditional acoustic musical instruments.

The musician evaluates the match between his intention, that is, the sound or musical result he would like to produce, and the actual result that was produced. Admittedly in traditional instruments, be they electronic or acoustic, most of the adaptation goes on in the human performer. Changes are made in the instrument but they are most often made by respecifying the controller mapping in an off-line manner.

One of our objectives in this work is to push more of the adaptive character this performer-instrument relationship into the instrument itself. We shall concentrate on the design of the controller.

Soft Computing

Neural networks, fuzzy systems, and neuro-fuzzy systems are natural candidates for acting as the adaptive element in an adaptive interface, because of their learning and generalization properties. Using them as adaptive elements in a user interface may provide techniques that give novices quick access to the underlying technology of a system and yet allow experts to develop virtuosity. The interface would initially be able to deal with a user's imprecision and yet have sufficient flexibility to accommodate the user as he develops both his ability to produce discriminatory input gestures and to articulate system demands. To facilitate our experiments, we have developed a real-time interactive fuzzy reasoning system and neural network simulator for the MAX real-time music programming language.

Features of our environment include real-time graphics display of membership functions and neural network activation levels and weights, interactive real-time tuning of membership functions, and objects for handling real-time input/output from external sensor devices such as data-gloves or infrared spatial locators. Although MAX was originally intended for musical applications, it is not limited to them. MAX's underlying real-time scheduler makes MAX an ideal environment for studying non-musical real-time control applications as well as musical applications.

3. SYSTEM ENVIRONMENT

This section introduces the MAX real-time music programming environment and introduces our neural and fuzzy systems tools. We give details on how the systems fit into the MAX environment and how to use them.

MAX

MAX is a widely available real-time music programming language that runs on Macintosh[9]. It is a graphical language in that programs are written by instantiating computational objects and then graphically connecting them. MAX was originally intended to address the needs of the computer music community by providing a real-time scheduling environment and an interface to the world of MIDI (musical instrument digital interface). MIDI is an international and industrial standard protocol developed and refined by a consortium of musical instrument manufacturers that allows communication of the instruments with computers and other instruments manufactured by different companies.

A major strength of the MAX environment is the ability to extend the language by adding external objects written in C. This facility gives the user the power to create objects that may be more efficiently computed in C or to create objects that talk to other external devices such as high-bandwidth gesture sensing devices or networking devices. The specification for external objects is well defined and new external objects transparently integrate into the environment.

It is through this external object interface that we have implemented a neural network simulator and a fuzzy reasoning system that function within the MAX environment. The network nature of these two computational paradigms fit well within the framework of the MAX environment. The next two subsections discuss each of the two systems in detail.

MAXNet

MAXNet is a multilayer feedforward network simulator that can also simulate limited recurrent neural network architectures and implements the classic back-propagation learning algorithm[2]. The user can specify the number of input, output, and hidden units and the functions, either linear or sigmoid, that each computes. When a MAXNet object is instantiated a dialog box appears that allows the user to specify these and back-propagation learning parameters. Embedding a neural network system into a MAX patch is as easy as drawing a line between two objects.

MAXNet also has the option to graphically display the state of the networks activation levels and weight values. The weights are color coded according to their magnitude and the activation level is represented by the size of a white dot in the middle of each neuron. Because MAX has a library of user interface objects, such as sliders and dials, investigating the network behavior given some input state can be done interactively and in real-time.

MAXFuz

MAXFuz is an object that can perform fuzzy reasoning within the MAX environment. Currently the membership functions are limited to trapezoidal shapes, however, extending the object to support other membership function types is straightforward. Our fuzzy system implementation in MAX consists of a collection of objects; fuzzy variables, fuzzy sets, and conventional or TSK fuzzy rules. As each fuzzy variable, fuzzy set, and fuzzy rule are added to the system, they are automatically wired into the fuzzy knowledge-base.

Gesture Sensing Devices

As previously mentioned, MIDI data from commercially available controllers, such as MIDI guitars, keyboards, and violins can be read directly into the MAX environment using built in MIDI objects. To experiment with alternate gesture sensing devices, we have also added objects that communicate with data acquisition cards and network communication devices. Once data is inside the MAX environment, it is treated the same regardless of whether it originated from a MIDI controller or another source.

4. APPLICATIONS

Prior to electronic and computer instrumentation, the control interface of a musical instrument was, with the exception of keyboard instruments, directly related to the acoustic properties of the instrument. Musicians bowed or plucked strings or excited air columns by blowing. To some degree, computer instrumentation separates us from the physical constraints of acoustic instruments by allowing us to decouple the physical gestures used to control an instrument from the sound production process. With this capability, we can realize instruments with custom interfaces suited to the specifications of a particular musician.

Although the notion of a customizable interface yields an extremely flexible system, accurately obtaining a musician's specification manually may be extremely tedious and inefficient. The manual customization approach becomes even more expensive as one considers building an interface for more than one musician. Consequently, a trainable user interface technique would be advantageous. From a functional standpoint, the interface serves as a mapping from user intentions to control actions for the underlying device. For this reason, we feel the automatic mapping capabilities of neural networks and fuzzy systems are well suited for our application.

A Glove Interface

One ongoing project is exploring non-traditional gesture input devices. One such device developed by Laetitia Sonami is a sensing glove, which can be used to control either a synthesis or note generating algorithm. This "Lady's Glove" privileges the hand's capability of producing a rich set of precise and repeatable gestures.

The glove is able to sense hand postures using a combination of resistive strip and Hall effect sensors. The resistive strip on each finger measures the degree of bend of the whole finger; individual joint angles are not measured. In addition, the distance between each finger and thumb is measured by hall

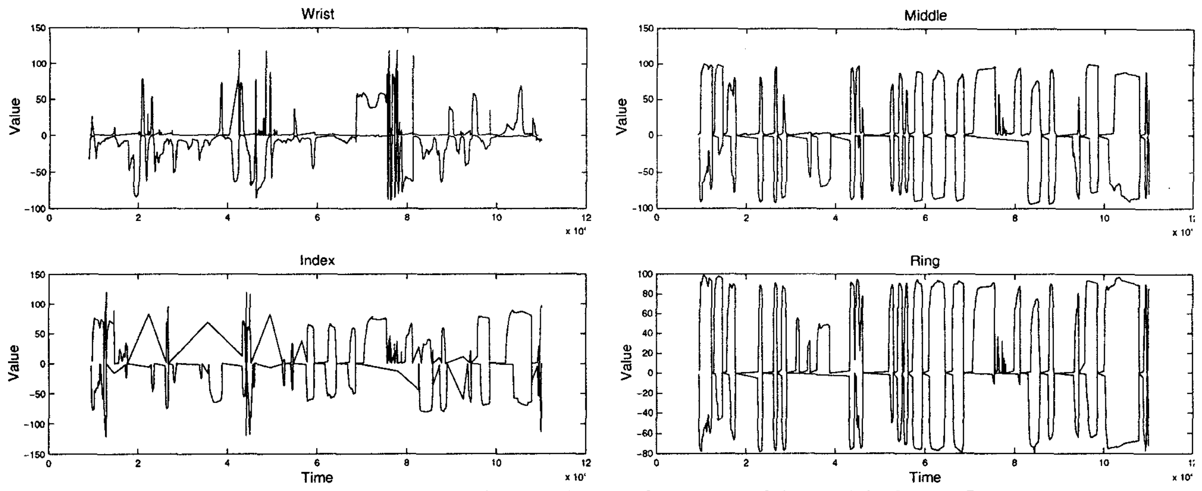

Figure 2. Wrist, and finger sensor outputs for a typical performance of the Lady's Glove. Each plot shows the outputs of the two sensors associated with each appendage with the upper and lower traces representing forward and backward wrist bend and tip and bottom finger joint bend.

effect sensors. A sonar range sensor is also used to determine the distance between the two hands and the distance to a foot. The sensor measures can be mapped to any number of the generative processes's parameters, such as internote timing in a note generator, reverb time in a signal processing algorithm, or parameters that control the course of a melodic process.

The sensors are wired to a custom device, called the Sensorlab, developed at STEIM in Amsterdam. This device digitizes the sensor values and translates them into a MIDI data stream. There unit itself contains a microprocessor, which can be programmed with the use of an external computer. The programs can specify the MIDI/sensor mapping and provide a limited means for specifying scaling and gain functions. Figure 2 shows sensor outputs sampled at 20msec during a typical performance. In this figure, the traces for one the two sensors associated with the wrist and each finger are inverted for readability.

As part of our initial study of the glove's movements, we performed a Principal Components Analysis on the input data. This analysis verified our hypothesis that the dimension of the glove is high. Using only the wrist and finger sensors in this analysis showed that the first two components only pick up 55% of the variance and six components are needed to obtain 90% of the variance. In addition, PCA on separate performances show good agreement, suggesting that the data collected is consistent (see Figure 3).

Controlling a musical processes with the glove requires that we build a map from the glove sensor space to the underlying parameters of the process, which can be an arduous process. Therefore, we propose using adaptive methods to customize the glove to a particular performer. The basic idea is to gather training data and then use machine learning techniques to automatically design the mapping function.

A straight forward approach is to use a connectionist network to learn to associate hand poses with the desired control outputs. In this scenario, the training

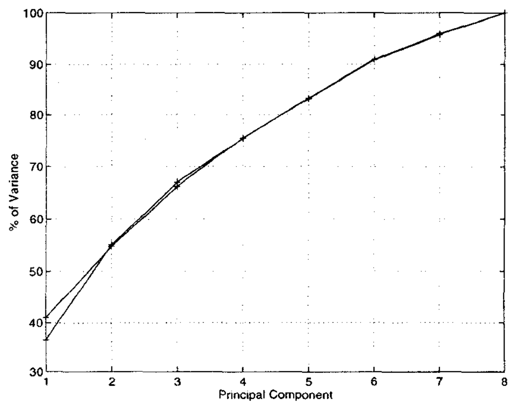

Figure 3. Results of PCA analysis on two different glove performances.

data is obtained by repeatedly having the user generate a pose for a particular output configuration. For example, in the case of static poses, the system indicates to the performer some output configuration, and then takes a snapshot of the sensor readings. Training algorithms, such as neural networks or genetic algorithms can then be used to design the controller. In [1], Fels and Hinton have successfully demonstrated neural network architectures for controlling a speech synthesizer. The system, dubbed Glove-Talk, uses a sensor-outfitted glove as the input to a multi-layer network that in turn controls the parameters of a speech synthesizer.

However, special care must be taken to construct efficient mappings; both temporal and topological components must be considered. For example, points that often occur adjacent in time should also be adjacent in sensor space. Therefore, an analysis of both spatial and temporal patterns would assist in constructing a coherent space.

In a PCA analysis of the Lady's Glove data that includes time lagged data (that is each sensor and its value at the previous sample are used as inputs), we observe that seven principal components are needed to capture 90% of the variance (see Figure 4). However, what is interesting to note is that only nine or ten components are needed to obtain up to 95%.

Another approach is to use other self-organizing techniques such as Kohonen networks [10]. The nodes in Kohonen networks are arranged in a grid-like neighborhoods and have a special learning rule that attempts to preserve topology. After an example is presented, the closest node (using some distance metric) and its active neighbors are rotated toward the example. As time progresses, the amount of rotation, or learning, and the active neighborhood size are decreased. The result is a low dimensional network, usually 2 or 3D, distributed in the space in a manner consistent with the frequency the inputs. Temporal adjacency information obtained from analyzing sensor traces can be used to dynamically modify the neighborhoods according to the temporal constraints.

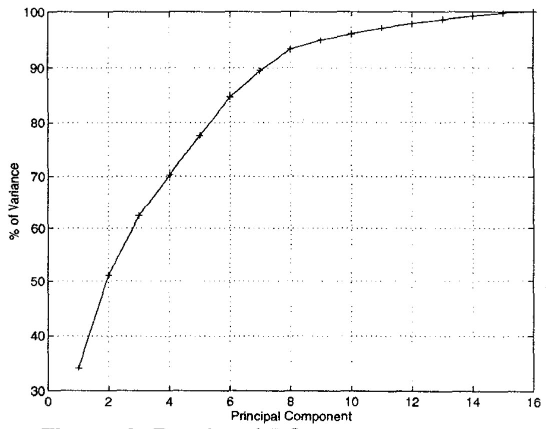

Figure 4. Results of PCA analysis on glove sensor data that includes lagged sensor values.

5. TRAINING STRATEGIES

Questions that immediately come to mind are those regarding training harnesses and training strategies; how do we obtain the data to train these structures. Specialized learning algorithms such as forward modeling[3] techniques or genetic algorithms[7] are available learning techniques, however they require a method for evaluating the error. Because the final product of a musical instrument are sonic events meant for human ears, the error should be perceptually weighted measure of the difference between the desired and obtained sounds.

Several difficult issues concerning adaptive user interfaces must be addressed, such as how does the overall system performance (human and machine) behave in an environment where both are learning. Having the machine learn too quickly can be detrimental when the human has not settled on a consistent mental model of what is expected.

A method for exploring the space of sonic capabilities of an instrument so that the musician can have some idea of the limitations of the instrument is needed. We proposed that the instrument designer initially assemble a set of trajectories which demonstrates the system's capabilities. The musician is then allowed to choose trajectories he would like to mimic. Generation of additional trajectories could be carried out randomly or by an interactive genetic algorithm. For example, the user selects interesting trajectories from a population of trajectories. The selected trajectories are then allowed to recombine and produce new trajectories. The advantage of using a genetic algorithm approach may be that interesting uninteresting trajectories may be avoided and interesting ones refined quickly.

6. CONCLUSIONS AND REMARKS

In this paper we have presented our tools for studying applications of soft computing to real-time control of musical processes. They consist of gesture sensing devices like the glove, instrumentation hardware and software that provides MIDI or other types of data streams from the sensors, the MAX programming environment with its real-time scheduling and musical orientation, a set of specialized MAX objects that implement various soft computing paradigms like neural networks and fuzzy control systems, and some sound synthesis technologies based on additive synthesis. The environment composed of this set of tools provides a effective laboratory for the study of soft computing in situations, like those provided by music, that demand hard real-time performance.

Experiments with glove and other non-traditional multidimensional controllers indicate that we can advantageously adapt a musical instrument to the personal requirements of a given performer.

ACKNOWLEDGMENTS

Special thanks go to Laetitia Sonami for her participation in this research. The "Lady's Glove" controller of her design was used in the experiments and in the preparation of the video tape of demonstrations. The authors would like to thank the CNMAT staff for their assistance and Adrian Freed for valuable input. The authors would also like to thank Silicon Graphics and Gibson Western Innovation Zone Laboratories.

REFERENCES

[1] Fels, S.S. and Hinton, G.E., "Glove-Talk: a neural network interface between a data-glove and a speech synthesizer," *IEEE Trans. on Neural Networks*, No. 1, Vol. 4, 1993, pp 2-8.

[2] Hertz, J. Krough, A., and Palmer, R.G., *Introduction to the Theory of Neural Computation*, Menlo Park, CA, Addison-Wesley, 1991.

[3] Jordan, M. and Rumelhart, D., "Forward Models: Supervised Learning with a Distal Teacher," *Cognitive Science*, 1992.

[4] Lee, M., Freed, A. and Wessel, D., "Neural Networks for Simulation Classification and Parameter Estimation in Musical Instrument Control," *Proc. of the SPIE Conf. on Adaptive and Learning Systems*, Orlando, FL, 1992.

[5] Lee, M. and Wessel, D., "Connectionist Models for Real-Time Control of Synthesis and Compositional Algorithms," *Proc. of the Int. Conf. on Computer Music*, San Jose, CA, 1992.

[6] Lee, M. A., Freed, A. and Wessel, D., "Real-Time Neural Network Processing of Gestural and Acoustic Signals," *Proc. of the Int. Conf. on Computer Music*, Montreal, Canada, 1991.

[7] Lee, M. A. and Takagi, H., "Integrating design stages of fuzzy systems using genetic algorithms," *Proc. IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE '93)*, San Francisco, CA, 1993, pp.612-617.

[8] Mathews, M. and Pierce, J., *Current Directions in Computer Music Research*, MIT Press, Cambridge,MA, 1989.

[9] Puckette, M. and Zicarelli, D., *MAX - An Interactive Graphical Programming Environment*, Opcode Systems, Palo Alto, CA, 1995.

[10] Ritter, H., Martinetz, T., Schulten, K., *Neural Computation and Self-Organizing Maps - An Introduction*, Addison-Wesley, Reading, MA, 1992.

[11] Roads, C. and Strawn, J., *Foundations of Computer Music*, MIT Press, Cambridge,MA, 1987.

[12] Wessel, D., "Instruments that Learn, Refined Controllers, and Source Model Loudspeakers," *Computer Music Journal*, Vol. 15, No. 4, Winter 1991, MIT Press.