Real-time Adaptive Control of Modal Synthesis

Reynald Hoskinson
Department of Computer Science
University of British Columbia
Vancouver, Canada
reynald@cs.ubc.ca

Kees van den Doel
Department of Computer Science
University of British Columbia
Vancouver, Canada
kvdoel@cs.ubc.ca

Sidney Fels
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
ssfels@cece.ubc.ca

ABSTRACT

We describe the design and implementation of an adaptive system to map control parameters to modal audio synthesis parameters in real-time. The modal parameters describe the linear response of a virtual vibrating solid, which is played as a musical instrument by a separate interface. The system uses a three layer feedforward backpropagation neural network which is trained by a discrete set of input-output examples. After training, the network extends the training set, which functions as the specification *by example* of the controller, to a continuous mapping allowing the real-time morphing of synthetic sound models.

We have implemented a prototype application using a controller which collects data from a hand-drawn digital picture. The virtual instrument consists of a bank of modal resonators whose frequencies, dampings, and gains are the parameters we control. We train the system by providing pictorial representations of physical objects such as a bell or a lamp, and associate high quality modal models obtained from measurements on real objects with these inputs. After training, the user can draw pictures interactively and "play" modal models which provide interesting (though unrealistic) interpolations of the models from the training set in real-time.

Categories and Subject Descriptors

H.5.5 [**Sound and Music Computing**]: Systems, Signal analysis, synthesis, and processing; J.5 [**Arts and Humanities**]: Performing arts (e.g., dance, music)

1. INTRODUCTION

Musical instruments are usually selected before a performance and then played in real-time. Occasionally a versatile performer may play several instruments during a piece, sometimes even simultaneously. However, switching instruments is usually not considered to be part of the performance skills of the artist but taken more or less for granted.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NIME '03 Montreal, Canada
Copyright 2002 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

This metaphor has been propagated to digital instruments which have elaborate real-time controllers (keyboard, MIDI wind-controller, drum pad, etc.) for playing the instrument, but simple switches to select the instruments or "presets".

Physical musical instruments allow a limited amount of real-time modification of the instrument's behavior, and in the 20th century music composers have moved some of these controls into the performance area. For example requiring a cello player to retune a string while playing, can extend the scope of the instrument.

Synthetic digital instruments using real-time audio synthesis [26] offer the possibility to make the virtual instrument completely flexible and, by changing the synthesis parameters in real-time, allow the morphing of different instruments into each other. This gives the performer the ability to control the nature of the instrument itself in real-time but poses the challenge of finding intuitive and natural interfaces to control these "design parameters".

In this paper we describe a software system which attempts to provide a generic framework to construct real-time controllers for digital synthesis algorithms. Our system uses a backpropagation neural network to map the control variables, which the performer directly controls, to the synthesis variables in a configurable and adaptive way. This is done by training the network on a set of input-output pairs which describe some of the desired properties of the mapping. This can be thought of as defining a collection of instrument presets which are specified by input variables of the performers choice. Once the network is trained, a real-time control map is generated which generalizes the training set to a continuous map allowing continuous control. Because of the neural network's ability to detect features, we believe this mapping is able to generalize the performer's intent in some sense, rather than just provide some arbitrary interpolation.

1.1 Related Work

There have been several attempts to create adaptive mappings between gesture and sound. Most notably, [13] used neural networks to map hand gestures to speech formant amplitudes and frequencies, which were excited by a different controller. The neural networks allowed the system to learn the relationship between the speaker's mental model space and the actual sound space. The speaker thus needed only to work in the relatively easy articulatory space instead of formant space.

A combination of neural networks and fuzzy logic software intended for real-time musical instruments control written in the MAX real-time programming environment was de-

scribed in [15]. An adaptive conductor follower based on neural networks was described in [16].

Of course, many hand-crafted systems to help facilitate learning the mapping between gesture and music have been attempted. For example, refer to [25, 12] for a description of a number of these devices. These mappings strategies all depend upon the intuition of the designer. Several common strategies have been developed to make the mapping easy to learn and understand. One typical strategy is to instrument a pre-existing acoustic instrument such as a cello [17] or saxophone [1]. This approach has the advantage of constraining the player's gesture space to a predefined, already learned space. Unfortunately, the output space may not have any obvious relationship to the gestures. Another technique uses objects that already have clear affordances [21] for control but are not necessarily based on acoustical instruments [2]. Objects such as a coffee mug can be instrumented and interactions with them mapped to sounds. While the mapping may not be clear at the outset, the fun of the interface form encourages a player to begin making sounds and exploring the interface. Other strategies include the use of metaphors [12].

In all the situations above, an adaptive system may be helpful in improving the transparency of the mapping. By carefully choosing the objective space and letting an adaptive algorithm match this to the player's mental model of the gesture-to-sound mapping, improvements should be possible. The role that the mapping plays in determining whether a musical interface is expressive is very complex [23]. The adaptive interface is one technique to help make new interfaces for musical expression.

1.2 Overview

Our prototype system has been applied to generate a control strategy for modal synthesis using hand-drawn greyscale pictures. Several pictures are associated with physical models of the objects they are intended to depict, which are linear modal models whose parameters were obtained by fitting them to sound recordings of real objects. Modal models of "everyday" objects such as lamps, kettles, coffee cups, etc. require anywhere from 4 to 100 modes for high quality sounds, which results in $12 - 300$ synthesis parameters to control, which is a very large space. This space contains the linear sound behavior of every imaginable rigid body, from wooden tables to the liberty bell, to the sound of an oddly shaped piece of scrap metal lying on some junkyard! Because of the large size of the sound space it is not possible to manually design the coupling of every synthesis parameter to some physical controller, and the need for a more automated approach to control such as that proposed in this paper becomes apparent.

Because there are so many synthesis parameters, we need a control space which is large enough to reach a substantial portion of the possible sound models. The greyscale level of the pixels of an image provide this large control space. After training the network on the examples, we deploy the trained network in a real-time application where the user can interactively draw a picture and have the modal parameters change in real-time. This simple interface requires no special hardware and is easy to work with, even for non-musicians, and therefore allows us to use it as a good testbed application for our controller design. We believe it also results in an very entertaining sonified drawing application.

The modal model can be excited by any means (or could be embedded in a more complicated synthesis patch) and for testing purposes we use impulses, noise excitations and a live data stream from a contact mike [4] which allows a more direct interaction with the virtual object.

The remainder of this paper is organized as follows. In Section 2 we describe and justify our control model and establish some notation. In Section 3 we describe our instrument model and design and summarize modal synthesis. In Section 4 we describe our prototype application and results obtained, and conclusions and directions for future work are presented in Section 5.

2. THE CONTROL MAP

To articulate the problem we find it useful to describe the mapping in a somewhat abstract manner. Let us denote the continuous synthesis parameters describing a virtual instruments by an N-dimensional vector $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$, which we can visualize as a point in "instrument space" Θ. This space consists of all possible virtual instruments that can be modeled by changing parameters of a synthesis algorithm. A "preset" of an algorithm corresponds to a single point in Θ. We can visualize a conventional synthesizer with preset buttons as consisting of a cloud of points in Θ which we can navigate with buttons (or some other discrete interface).

A continuous interface to instrument selection allows the performer to navigate smoothly between the presets and for example morph a woodblock into a gong while playing. However, its is not clear how to move from one preset to the other in the most natural way. Naively one could interpolate linearly in parameter space but this is arbitrary and does not "sound linear".

For example, let us morph the sound of a bell into the sound of a desk lamp by a linear trajectory in modal space (consisting of the frequencies, dampings, and gains), and control this with a single parameter λ which runs from 0 (a metal desk lamp) to 1 (a church bell). An interactive application which runs in most web browsers demonstrating this can be found on the web [6]. If we start at 1 and decrease λ, we first hear the bell going out of tune. Somewhere around $\lambda = 0.9$ the bell character is lost and from 0.9 to around 0.1 it sounds like "some metal object", but the character of the sound remains fairly constant until we come close to the lamp, around $\lambda = 0.1$ when the sound appears to rapidly "come into focus" and morph into the sound of a desk lamp. This somewhat subjective description illustrates the fact that though the trajectory is linear in parameter space and we move uniformly from one point to the other, what we hear does not sound linear and uniform at all.

Another challenge in designing interfaces is to provide gestural metaphors which are natural to the performer. Controlling motion in Θ *adaptively* allows the performers to customize the mapping according to their own peculiarities and wishes within the same system. A control interface is a continuous mapping

$$\kappa : \mathcal{C} \longrightarrow \Theta$$

from a control space $\mathcal{C}$ to the instrument model space Θ. The K-dimensional space $\mathcal{C}$ consists of all possible settings of the control variables $\boldsymbol{c} = \{c_1, \ldots, c_K\}$. These control variables are obtained from sensors such as Cybergloves, position trackers, etc. and are controlled by the performer in real-time.

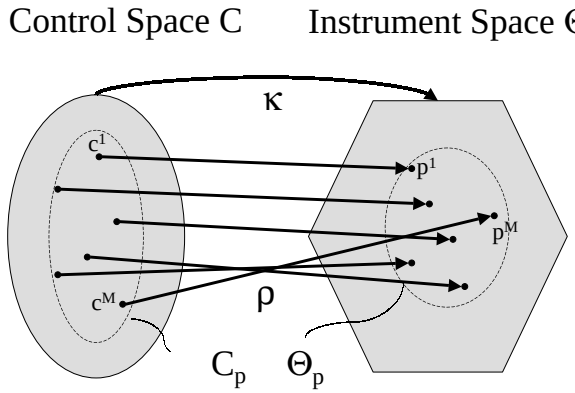

Figure 1: Control space C and instrument space Θ. The discrete preset mapping ρ is generalized to the continuous mapping κ by training a 3 layer backpropagation neural network on ρ.

Presets are input configurations (points in $\mathcal{C}$) which are mapped to fixed instruments. The preset configuration ρ is defined by specifying M pairs $\rho = \{\{\boldsymbol{c^1}, \ ^1\}, \ldots, \{\boldsymbol{c^M}, \ ^M\}\}$, where $\boldsymbol{c^i} \in \mathcal{C}$ and $^i \in \Theta$. It is a discrete mapping ρ from $\mathcal{C}$ to Θ. We shall notate the preset control set by $\mathcal{C}_p = \{\boldsymbol{c^1}, \ldots, \boldsymbol{c^M}\}$, and the preset instrument set by $\Theta_p = \{^1, \ldots, \ ^M\}$. See Figure 1 for the notation.

A natural framework for constructing the continuous mapping κ as a generalization of the discrete mapping ρ is a 3 layer backpropagation feedforward neural network [19] with K inputs and N outputs which, appropriately scaled, provides the mapping κ. The preset configuration ρ provides a set of M training examples, and training the network on this set results in the desired mapping κ. An important feature of neural networks is their ability to detect and generalize features [19]. This is very relevant as the preset map ρ captures the performer's metaphor for control. The continuous interpolation of the preset configuration can incorporate features which are detected during the training phase by the neural net and generalize them. The preset configuration can also be seen as the specification by example of the desired behavior of the controller.

3. MODAL INSTRUMENT SPACE

A good physically motivated synthesis model for vibrating rigid bodies is modal synthesis [28, 14, 20, 3, 8, 9, 7, 22]. Modal synthesis models a vibrating object by a bank of damped harmonic oscillators which are excited by an external stimulus. See Fig. 2 for an illustration. The frequencies and dampings of the oscillators are determined by the geometry and material properties (such as elasticity) of the object and the coupling gains are determined by the location of the force applied to the object. The impulse response $p(t)$ of the modal model with L modes is given by

$$p(t) = \sum_{n=1}^{L} a_n \exp(-d_n t) \sin(2\pi f_n t), \qquad (1)$$

for $t \geq 0$ and is zero for $t < 0$, where $p(t)$ denotes the audio signal as a function of time. The modal parameters are the frequencies f_n, the dampings d_n, and the gains a_n. The frequencies and dampings are pure object properties whereas the gains also depend on the location of the interaction point on the surface of the object. The model ignores phase effects.

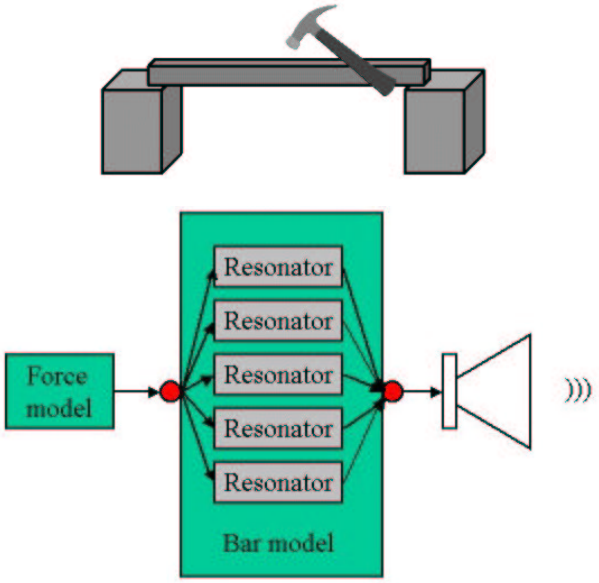

Figure 2: Modal synthesis of the sound made by hitting a bar with a hammer. The hammer force is modeled by a contact force model, and send to a bank of resonators, which is the modal model of the bar. Each resonator has a characteristic frequency, damping, and gain and the outputs of the resonators are summed and rendered.

We create sound models with the FoleyAutomatic [7] system, which allows the creation of realistic sound models based on modal synthesis as well as various contact force models which include striking, scraping, sliding, and rolling. The FoleyAutomatic system is freely available from the web as part of the JASS system [10, 5], a Java based real-time audio synthesis toolkit. The modal models can be acquired by parameter fitting to recorded sounds using the techniques described in [24]. Preliminary user studies [11] have shown that impact sounds constructed with this technique are indistinguishable from the real sound.

4. INTERACTIVE DRAWING

We have applied our adaptive controller framework to an interactive drawing application which allows the user to draw pictures on a square window. The picture is downsampled to 16×16 greyscale pixels with values in the range $0 - 1$. The pixels are taken as inputs to a neural net with 256 input units, 32 or 128 hidden units, and 60 output units, allowing for modal models of 20 modes.

The neural network was designed using the Java Object-Oriented Neural Engine (JOONE), an open-source neural net package implemented in Java [18]. JOONE provides a graphical environment to design multilayer neural networks, train them, and export trained networks into real-time applications. All of the neurons are implemented as sigmoid functions $y = 1/(1 + e^{-x})$. The learning rate is set to 0.8, and the momentum factor 0.3.

The 60 outputs of the net are numbers in the range $0 - 1$.

They mapped to the 60 modal synthesis parameters defined in Equation 1, for $L = 20$ modes. For optimum training of the neural net, the range $0 - 1$ should be mapped as uniformly as possible to perceptually relevant parameters. For instance, frequencies are perceived on a roughly logarithmic scale, so we would like a linear change in outputs to produce a logarithmic change in frequency. The three types of modal parameters are handled separately in order to best take into account the perceptual characteristics of the sounds.

For frequencies, we convert to the Bark scale [27], designed to uniformly cover the human auditory range. It can be expressed as $z = [26.81/(1 + 1960/f)] - 0.53$, with f the frequency in Hz. The result z is then scaled to between 0 and 1. For damping, the conversion is given by $(\log_e(d + 1.0))/5.0$. It covers dampings of up to roughly $150/s$, the most heavily damped modes that occur in the specific physical models we have used. Gains are converted to decibels, and we allow a range of $160dB$, enough for most (non-lethal) applications. The conversion is given by $1 + dB(a)/160$, with $dB(a) = 20 \log_{10}(a)$ the decibel level in the range $-160dB$ to $0dB$.

The preset configuration consists of four hand-drawn pictures depicted in Figure 3. The outputs corresponding to

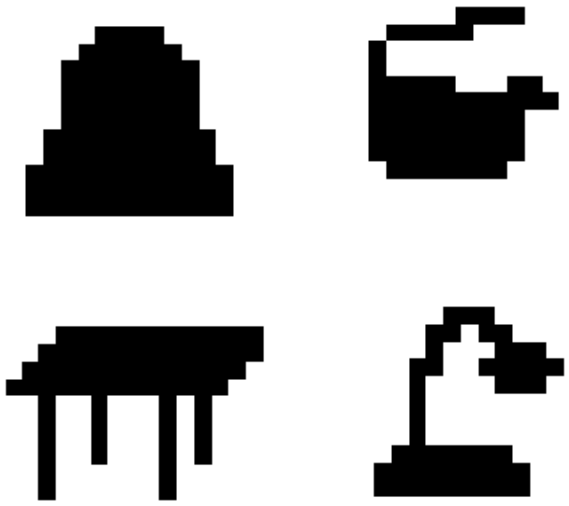

Figure 3: The four input images to the neural net, depicting a bell, a kettle, a wooden table, and a desk lamp.

the images are modal models obtained from parameter fitting to recorded sounds of the objects depicted, using the 20 most important modes selected by a perceptual criterion as described in [11], which result in very realistic sounds.

Two neural networks were created, one with 32 hidden units and one with 128 hidden units. Both were trained until the error in frequencies was below 10 cents (one tenth of a semitone). Errors in the dampings and gains are perceptually much less noticeable, which is why we use the frequencies as a convergence criterion.

Convergence required about 200 iterations, less than one minute on a desktop computer with 733 MHz dual Pentium III processors. In Figure 4 we show the average error of the output as a function of the number of training epochs.

Qualitatively, we listened to the sounds at various stages in the training, obtained by using a picture from the training set as input. After 100 training epochs the results were recognizable as the target sounds but quite distorted, whereas the sound was indistinguishable from the target at 200 training epochs.

After training, we tested our real-time drawing application with fully converged nets containing 32 and 128 hidden nodes, using various excitations. We did not notice any qualitative differences in the behavior of the nets, though there were clear differences between them in sound for pictures we drew which did not resemble any in the training set. The interface allows us to load any of the pictures in the training set and then interactively draw over them. Though the preset configuration with just four presets is very minimal, we were surprised by the richness of the interface. For example, if we start with the bell, when its lower or upper portions are erased, the sound changes dramatically and rapidly loses its bell-like character. But if we erase parts of the picture starting from the middle, the pitch of the bell seems to change, and it is almost possible to etch out a shape inside the bell such that the modes remain in tune and the bell character of the sound is preserved.

If the picture is completely erased or completely black, we do not get a silent model, but rather something which we can only describe as "non-descript". When we draw random shapes, they sound just like that, like random sounds. It is only when features of the input images appear in the drawing that the sounds become "interesting".

We find it very hard to describe the experience with the interface, and intend to convert the application into a Java applet and make it available on the web to interact with through a standard web-browser.

5. CONCLUSIONS

This paper has described the design of a general framework to control audio synthesis algorithms with many continuous parameters. The controller maps an input space, which is the space in which the performer manipulates input devices, into the parameter space of a synthesis algorithm using a neural network.

The behavior of the controller is specified by example by specifying a discrete set of input-output pairs, which we have called the "preset configuration". These examples capture the performers intent and a neural network can possibly extract enough features from the examples to generalize it to a "natural" continuous mapping.

Our implementation consists of an interactive drawing application, with the drawing functioning as the controller. Through a neural network the drawing application is controlling parameters of a modal synthesis algorithm. The neural network is trained on a set of images with associated sound models. A real-time synthesis kernel then allows the user to "play" this modal synthesis algorithm by various means. When one of the training examples is drawn, the exact sound model is reproduced, but when a picture outside the training set is drawn the result is not a-priory known, but determined by the neural network's interpolation. Of course, if we draw a realistic image of a real object not in the training set, the resulting sound model will not be re-

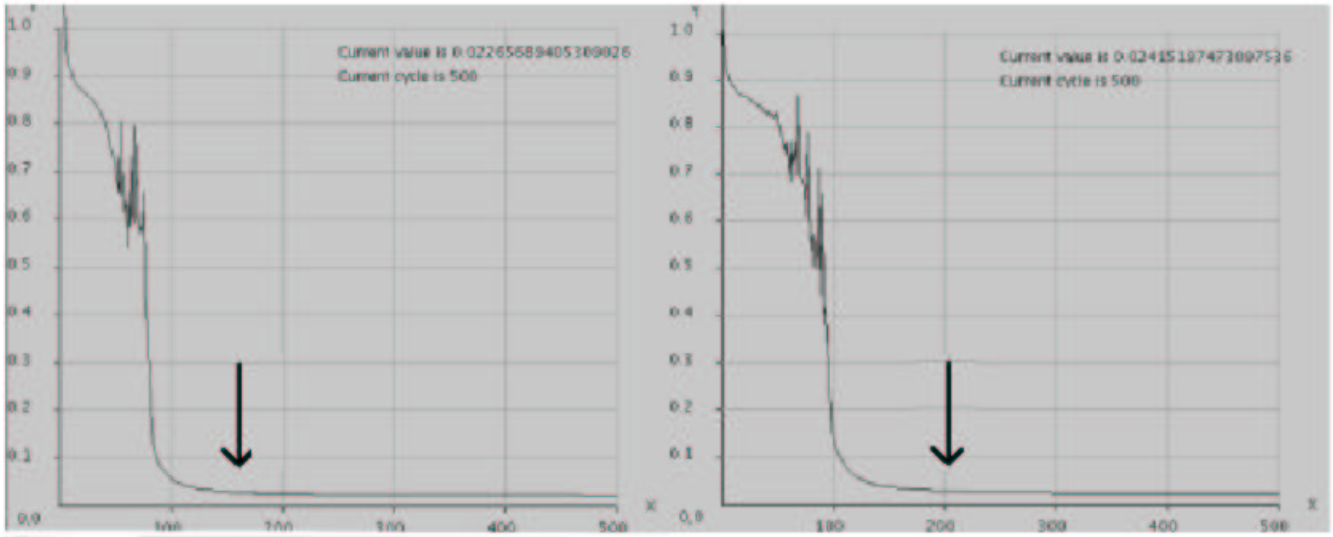

Figure 4: Convergence graphs of two neural nets we tested. Each has 256 inputs. The first, with 128 hidden shows convergence at under 200 iterations. The second, with 32 hidden nodes, shows convergence a little later, but is still acceptable at the 200-iteration mark.

alistic, as the modes will depend on the internal structure and other material properties not contained in an image. However, the interpolated models are musically rich and interesting, drawing on features of the objects in the training set.

Our implementation is in an early stage of development and there are several issues which we will address in the near future. First we will extend the training set to include more images to allow the neural net to extract meaningful features. Many similar drawings of the same object should be included in the training set, which can probably achieved by adding noise to the input set. It would be interesting to verify if translation and rotation invariance can easily be learned by including translated and rotated examples in the training set. Next we will incorporate a webcam into the current implementation as an input device, which will provide a very interesting live controller. We are also very interested in applying the controller to live performance, or as a base of an interactive acoustic installation.

6. REFERENCES

[1] M. Burtner. Noisegate 67 for Metasaxophone: Composition and Performance Consideration of a New Computer Music Controller. In *Second International Conference on New Interfaces for Musical Expression(NIME02)*, pages 71–76, Dublin, Ireland, 2002.

[2] P. Cook. Principles for Designing Computer Music Controllers. In *First Workshop on New Interfaces for Musical Expression (NIME01), ACM Special Interest Group on Computer-Human Interfaces*, Seattle, USA, 2001.

[3] P. R. Cook. Physically informed sonic modeling (PhISM): Percussive synthesis. In *Proceedings of the International Computer Music Conference*, pages 228–231, Hong Kong, 1996.

[4] K. v. d. Doel. *Sound Synthesis for Virtual Reality and Computer Games*. PhD thesis, University of British Columbia, 1998.

[5] K. v. d. Doel. JASS Website, http://www.cs.ubc.ca/~kvdoel/jass, 2003.

[6] K. v. d. Doel. JASS Website, Morph Example, http://www.cs.ubc.ca/~kvdoel/jass/morph2/morph2.html, 2003.

[7] K. v. d. Doel, P. G. Kry, and D. K. Pai. FoleyAutomatic: Physically-based Sound Effects for Interactive Simulation and Animation. In *Computer Graphics (ACM SIGGRAPH 01 Conference Proceedings)*, Los Angeles, 2001.

[8] K. v. d. Doel and D. K. Pai. Synthesis of Shape Dependent Sounds with Physical Modeling. In *Proceedings of the International Conference on Auditory Display 1996*, Palo Alto, 1996.

[9] K. v. d. Doel and D. K. Pai. The Sounds of Physical Shapes. *Presence*, 7(4):382–395, 1998.

[10] K. v. d. Doel and D. K. Pai. JASS: A Java Audio Synthesis System for Programmers. In *Proceedings of the International Conference on Auditory Display 2001*, Helsinki, Finland, 2001.

[11] K. v. d. Doel, D. K. Pai, T. Adam, L. Kortchmar, and K. Pichora-Fuller. Measurements of Perceptual Quality of Contact Sound Models. In *Proceedings of the International Conference on Auditory Display 2002*, Kyoto, Japan, 2002.

[12] S. Fels, A. Gadd, and A. Mulder. Mapping transparency through metaphor: towards more expressive musical instruments. In *Organized Sound*, page to appear. Cambridge Press, 2003.

[13] S. S. Fels and G. E. Hinton. Glove-TalkII: A neural network interface which maps gestures to parallel formant speech synthesizer controls. *IEEE Transactions on Neural Networks*, 9(1):205 – 212, 1998.

[14] W. W. Gaver. Synthesizing auditory icons. In

Proceedings of the ACM INTERCHI 1993, pages 228–235, 1993.

[15] M. Lee, G. Garnett, and D. Wessel. An Adaptive Conductor Follower. In *Proceedings of the International Computer Music Conference*, pages 172–175, San Jose, CA, 1993.

[16] M. Lee and D. Wessel. Neuro-Fuzzy Systems for Adaptive Control of Musical Processes. In *Proceedings of the International Computer Music Conference*, Tokyo, Japan, 1993.

[17] T. Machover. Hyperinstruments: A Composer's Approach to the Evolution of Intelligent Musical Instruments. In *Organized Sound*, pages 67–76, San Francisco, 1991. Cyberarts.

[18] P. Marrone. JOONE Website, http://joone.sourceforge.net, 2003.

[19] J. L. McClelland, D. E. Rumelhart, and the PDP Research Group. *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1*, volume 1. MIT Press, Cambridge, 1986.

[20] J. D. Morrison and J.-M. Adrien. Mosaic: A framework for modal synthesis. *Computer Music Journal*, 17(1), 1993.

[21] D. Norman. *The Design of Everyday Things*. Currency/Doubleday, 1990.

[22] J. F. O'Brien, C. Chen, and C. M. Gatchalian. Synthesizing Sounds from Rigid-Body Simulations. In *SIGGRAPH 02*, 2002.

[23] N. Orio, N. Schnell, and M. Wanderley. Input Devices for Musical Expression: Borrowing Tools from HCI. In *First Workshop on New Interfaces for Musical Expression (NIME01), ACM Special Interest Group on Computer-Human Interfaces*, Seattle, USA, 2001.

[24] D. K. Pai, K. v. d. Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau. Scanning physical interaction behavior of 3D objects. In *Computer Graphics (ACM SIGGRAPH 01 Conference Proceedings)*, Los Angeles, 2001.

[25] J. Paradiso. Electronic music interfaces: new ways to play. *IEEE Spectrum Magazine*, 34(12):18–30, 1997.

[26] J. O. Smith. Physical modeling synthesis update. *Computer Music Journal*, 20(2):44–56, 1996.

[27] H. Traunmuller. Analytical expressions for the tonotopic sensory scale. *J. Acoust. Soc. Am.*, 88:97–100, 1990.

[28] J. Wawrzynek. VLSI models for real-time music synthesis. In M. Mathews and J. Pierce, editors, *Current Directions in Computer Music Research*. MIT Press, 1989.