A *manifold interface* for a high dimensional control space.

Insook Choi, Robin Bargar, and Camille Goudeseune
National Center for Supercomputing Applications
Beckman Institute, 405 S. Matthews, Urbana, IL 61801, USA
email: ichoi@ncsa.uiuc.edu

Abstract

The *manifold interface* was developed for intuitive navigation of an arbitrary multi-dimensional space where control variables for sound synthesis or composition algorithms are parameterized. Methods for sending control signals are implemented with an ability to access parameters simultaneously for real-time parameter variation. Graphical representation of control space helps visually identify parameter regions associated with acoustic results. Control signals are generated by a user's 3D gestures and can be stored and further structured and applied to synthesis engines in a real-time feedback environment. Working vocabularies such as *control path, interactive sequence, phase space* and *window space* are introduced. We present examples of control structures generated within the interface, and sounds created by real-time control of the CHANT synthesis engine.

1. Complexity, Dimension and Intuition

A complex system such as a musical instrument responds to the way a musician addresses it with her or his own comprehension of the system. The complexity of an acoustic signal corresponds to the complexity of a listener's auditory experience of that signal; however these corresponding complexities reside in different worlds. Signals from a musical system are compressed by a transmitting medium (sound) and expand in human cognition; when we make music our physical actions transmit signals to the musical instrument, and the signals come full circle when we hear the resulting sounds.

We describe this circle as a feedback process through which we learn to associate acoustic properties with the actions that produce them. Feedback allows us to measure the consequences of gestures and arrive at conclusions concerning the covariance of gestures and the corresponding sounds. The term *intuitive* often appears in the context of human-machine interaction, cited (inappropriately) as an explanation of relationships learned in a feedback process; the term really describes a person's experience and does not explain the mechanism involved. We will use *intuitive* to describe the experience of participating in feedback systems where the participant can draw compatible conclusions from a small number of trials and quickly learn to differentiate a large number of states in a complex system.

We can program a computer to represent signals in terms of dimension and state changes in time. However, numerical models in high-dimensional representations are counter-intuitive, that is, inefficient for specifying or learning states and changes in a signal. Information presented as a large number of data sets to cover the dimensions required is not an intuitive representation of a musical signal. "Dimension" as an indicator and a controller of complexity may be well-suited to synthesis engineering and signal processing but less well-suited to music composing and listening. We can understand auditory signals without calling upon the organizing principle of "dimension." At the same time, when we use parameterized representations of signals in the computer, we need to have unique access to each parameter to provide control specifications; still we prefer to direct a computer's processes from an intuitive feedback system.

2. Musical signal control and manifolds

Computer music technology has traditionally bound composers to a linear relationship between dimensional representation and control: in order to exercise greater control, interfaces become more complicated. Alternatively, simpler interfaces such as electronic keyboards sacrifice the ability to control many parameters in exchange for real-time manipulation of fewer parameters. Software derivatives of music-n languages distill from synthesis algorithms arrays of p-fields (parameter fields), but do not provide good tools for organizing arbitrary control of p-fields over time. The breakpoint function has become the composer's indispensable visual and conceptual partner. Often the breakpoint functions of a computer synthesis algorithm are evident in the resulting sound; too often these are taken as the primary dramaturgy of the composition, though the result we experience is predictable. One solution is to increase the number of functions involved in the sound synthesis algorithm. Then one has to find a way to direct traffic among a multitude of breakpoint functions. In the data-flow interface paradigm, for example, the number of components in the

graphical representation increases - often linearly - with the number of components in the algorithm. Each control function can require several geometric figures on the screen. The visual accumulation of geometric figures can be intimidating. To simplify we can hide the figures, but when they are hidden we cannot edit them. We can create a single function as a macroscopic controller for a bundle of other functions but this prohibits us from quickly altering relations among the functions within the bundle.

Nonlinearities between parameter variations of signals and the perceived sounds compound the need for another approach. In many synthesis algorithms, to preserve the auditory identity of a timbre class under a perceived-as-linear transform of pitch, loudness or duration, a composer is required to arrange nonlinear transfer functions among the synthesis parameters. There is usually no way to generalize these functions for a broad range of conditions. After many experiments if we can identify useful functions we still face a brute force approach to editing the functions when a new acoustic range is needed. Control problems have apparently stalled composers' timbre research somewhere in the neighborhood of *Klangfarbenmelodie*, resulting in a great deal of computer music that occupies the acoustic domain between wave distortion and sound sampling (oscillating between *control-sans-complexity* and *complexity-sans-control*).

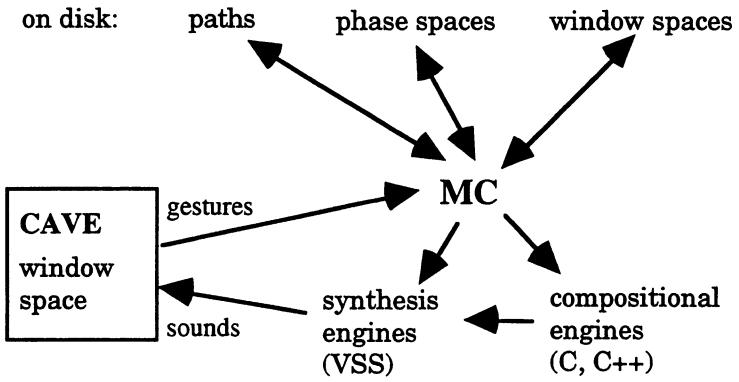
We treat musical control not as a complicated bundle of parameters but as a manifold involving high-dimensional structure. Then we apply a similarly-structured control signal by intuitive methods. Musical instruments are good paradigms for this model. They fit the human body and empower gestures with musical consequences. Musical signals are generated by a limited number of a performer's gestures in relation to immediate auditory feedback. The immediacy of the auditory feedback allows a performer to fine-tune gestures for inducing desired consequences. We apply this paradigm to our interface design so that the available acoustic variety is more easily controlled. This also encourages composers and performers to learn the ability to rapidly fine-tune relations among a small number of control variables, before deciding to proliferate the number of variables in a model.

3. A *manifold interface*

Describing the total musical signal control space as a high-dimensional manifold, we proceeded to develop an interface to support real-time auditory feedback, and to rapidly annotate and vary our position on a manifold in an ordered and time-critical manner. The manifold controller is an interactive graphical sound synthesis and composition interface; it's application is scalable from immersive virtual environments to desktop workstations. Its initial application for generating musical signals from a chaotic oscillator has been previously discussed in (Choi, I.). The manifold interface allows us to control a high-dimensional parametric space from a visual display having a continuous gesture input system with at least two degrees of freedom. Our current implementation includes 3D gesture input and 3D display. For workstations that support 2D controllers and 2D graphical display the references can be scaled down. A real-time sound synthesis and algorithmic composition environment is provided by the NCSA Sound Server (Bargar, R.).

The interface can be used concurrently for low-level control of synthesis engines and high-level control of composition parameters. Gesture inputs may be captured, reproduced and modified. Manifolds may also be stored, retrieved and modified. With the Sound Server the manifold interface may be applied to real-time and non-real-time synthesis.

Figure 1.
Data flow in the manifold controller



Architecture

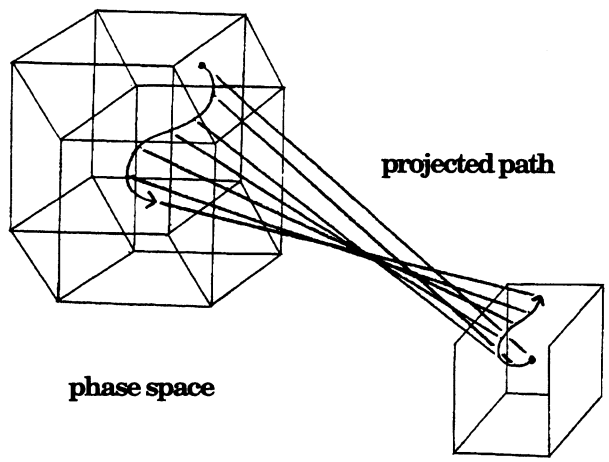
The manifold controller (MC) is a set of C++ classes implementing the following concepts: phase space, path and window space (Figure 1). By the *phase space* of a system we mean the traditional n-dimensional Eucli-

dean space where points - n-tuples of real numbers - correspond to states of a parameterized system. The phase space is all the permissible combinations of parameter values of an algorithm. Trajectories of control gestures are encoded in phase space. A *path* through a phase space is a mapping from some time interval [0, t_{Max}] to the phase space. This map need not be bijective or continuous; a path can cross itself, or make abrupt jumps. The *window space* defines how a three-dimensional visual representation is embedded in the high-dimensional phase space (Figure 2). We conceive of a three-dimensional visual display as a window onto the manifold. Like the performer of a musical instrument, a composer inputs changes to the system through the window space. In our implementation the window is displayed in the CAVE environment or a similar 3D view on a workstation (Bargar, R.). Position of a cursor in the window space determines the state of the system. The cursor responds to gesture-input devices such as the wand or mouse, and to voice and keyboard commands.

The window space

The window space is an important part of pre-compositional organization. A window space provides an efficient domain for generating and modifying classes of paths. Attributes of a window space are reflected in acoustic features common to the paths generated in that space. The design process begins with the choice of a small set of points in a phase space. These points represent combinations of parameter values associated with particular sounds. We want to be able to visit these "generating" points and move smoothly between them in phase space. Since the window space has to be displayed in three dimensions its embedding in the phase space may involve twists and bends; still we want the embedding to be continuous and "simple" while preserving a maximum amount of information.

Figure 2.
Embedding a window space

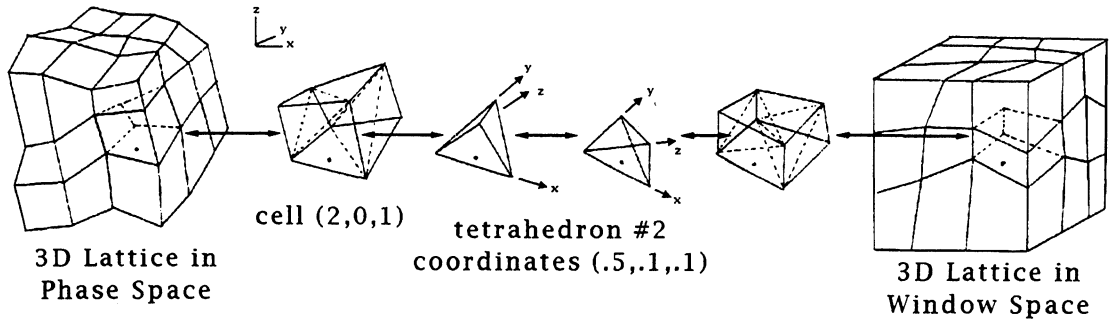


Defining the window space

Instead of the various statistical methods of data reduction, we use a genetic algorithm (GA) to find a near-optimal window space. (GA's find near-optimal solutions to a problem by starting with a random population of possible solutions and allowing only the better solutions to "multiply" and create offspring. This requires only a *fitness function* to compare solutions and a bit-vector representation of a solution. No further problem specification is needed.) The *generating points* in the phase space will map to certain points in the control space; the positions of these points in the control space are chosen to maximally preserve the structure of the phase space in the region of its generating points. We interpret "structure" as the matrix of Euclidean distances between points. So the states the GA explores are sets of points in the control space, represented as vectors of fixed-point numbers, and the fitness function measures the error between the original distance matrix and the matrix for a particular set of 3-dimensional points.

The image of the generating points in the control space is extended to a 3-dimensional lattice, lines through the generating points more or less parallel to the principal axes of the space. All points in the lattice are then used in a reversal of the previous GA to produce a corresponding lattice of similar geometry in the phase space. A point in the control space is mapped to one in the phase space by first finding which cube-like lattice cell it is in, and then finding its coordinates in the cell based on a tetrahedral decomposition of the cell (Figure 3). The corresponding cell and coordinates in the phase space define the resultant point in the phase space. The inverse map is computed similarly. As a point's cell-coordinates exist and are unique under certain conditions which the cells satisfy (convexity, noncoincidence of corner vertices), this map from one space to cell-coordinates and back to another space exists and is bijective. As the map is a patch of linear functions continuously connected, it is continuous as well.

Figure 3.
Bijective Map between the Phase Space and the Window Space



To smooth out the map's nondifferentiable "edges", we are investigating the use of high-dimensional splines, in particular, cubic B-spline volumes built on a perturbation of the 3-dimensional lattice in the product of the phase and window spaces. In a Euclidean space, given a sequence of control points $\{p_0,\ldots,p_n\}$ and an index parameter u,

$$P(u)=\sum_{k=0}^{n} p_k N_{k,t}(u)$$

defines the B-spline curve for these control points, where $N_{k,t}$ are the standard B-spline blending functions, polynomials of degree $t-1$ (Hearn, D.). We use cubic splines, hence $t=4$. Given a 3-dimensional lattice $\{p_{j,k,l}\}$ of control points, we define its associated B-spline volume by

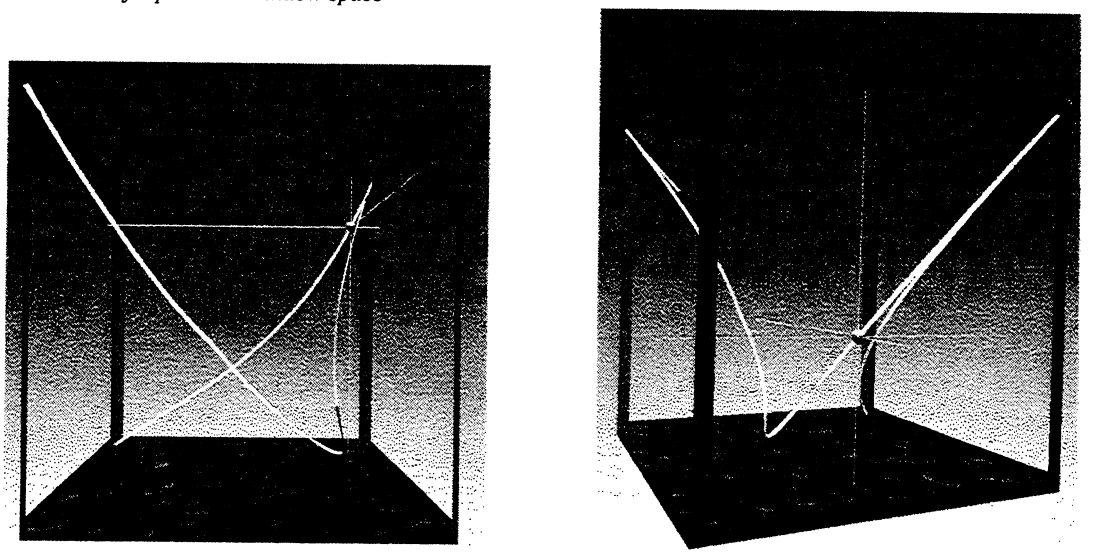
$$P(u,v,w)=\sum_{j=0}^{n_j}\sum_{k=0}^{n_k}\sum_{l=0}^{n_l} p_{j,k,l}N_{j,4}(u)N_{k,4}(v)N_{l,4}(w)$$

over the index parameters *u, v, w*. Since we want generating points to map onto their images in the window space, we perturb the original lattice in the product of the phase and window spaces with another GA to find a lattice whose use as a set of control points for a B-spline volume will yield this exact mapping. This search takes a long time to compute, because the GA's fitness function evaluates this spline equation for many values. The inverse computation is slower still, that of finding index parameters *u, v, w* which correspond to a given point in the product space (equivalently, in one of its two component spaces). Once we have these indices, though, they immediately provide the mapping between the component spaces without any linearizing steps such as the tetrahedral decomposition of a lattice cell. We are searching for ways to do this in real time.

4. Graphical surfaces and paths

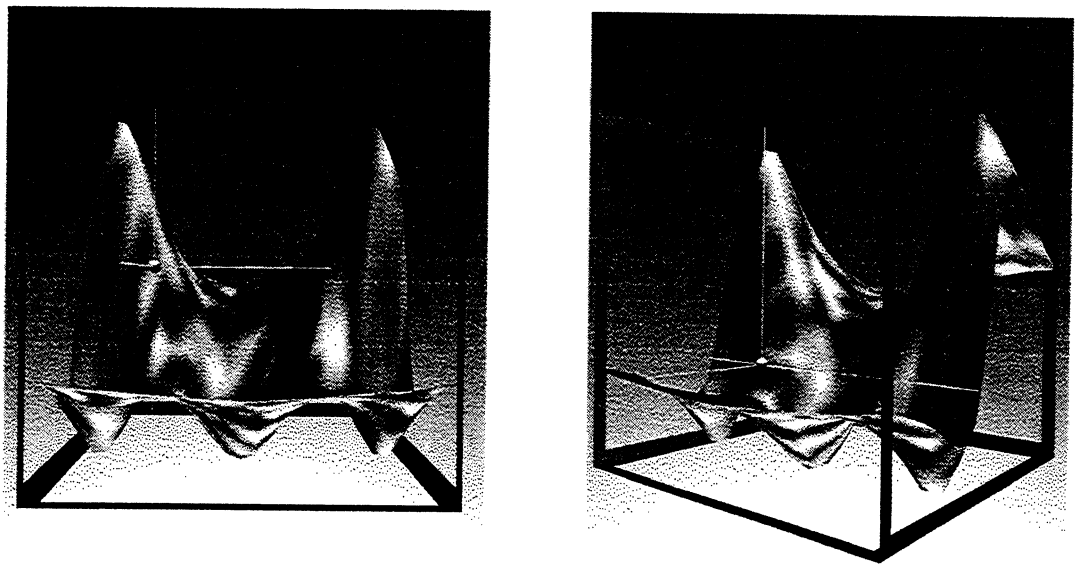
A controller with three degrees of freedom allows a composer to move arbitrarily through a window space; the cursor position in the window space then maps to a point in a phase space which controls the system directly (through a callback function). Having found an interesting region of a phase space, the composer can record a gesture therein as a path (Figure 4). The path is stored in the phase space, not in the window space, so it is defined with respect to the high-dimensional manifold and projected into the window to provide a visual representation with respect to the particular window space being used. On a workstation where a desktop mouse is inherently incompatible with three-dimensional control, we draw surfaces in the window space and constrain the cursor to the surface, thus compromising with the locally two-dimensional behavior of the mouse (Figure 5). Paths can then be recorded on the surface by gestures in two dimensions. To explore the whole control space a graphical surface can be defined *a priori* or modified (scaled, moved, rotated) extemporaneously while listening to its effect. The window space also can change extemporaneously as its generating points are modified; thus larger regions of the phase space can be explored. We can effectively control the window space by panning and zooming.

Figure 4
Two views of a path in a window space



Note the difference between this and conventional scientific investigation of a system, where one might hold all but one dimension constant and vary the last dimension as an "experiment." Here we work in the whole space at once in a way close to the working model of learning new instruments. We do not need to attend in detail to the dimensions being varied and how, since this information is encoded by the window space embedding prior to exploration. We can concentrate on grasping an intuitive orientation with respect to the space. Having found acoustically relevant gestures (paths) at this exploratory stage, the paths can then be subjected to rigorous experimentation. Control paths provide a form of gesture-based notation, allowing synthesis control signals (virtual sound events) to be exported from the manifold interface. Thus for composition and sound synthesis the graphics interface is not necessary past the exploratory stage (which is helpful if the composer doesn't have a CAVE at home). Using control paths as source materials of musical structures, additional computation tools can be applied to path files at later stages of experimentation and composition, producing new paths that are used to render newly-composed sounds.

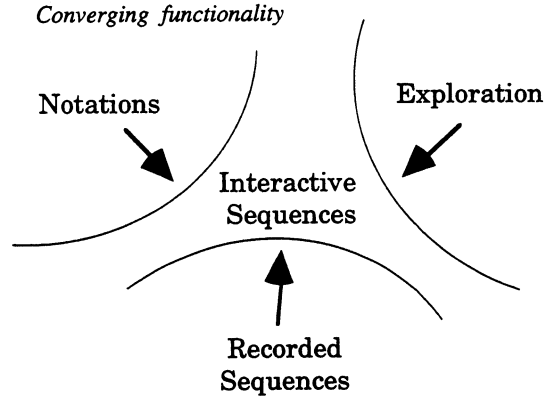
Figure 5.
Two views of a surface in a window space



Quantization
A path is initially recorded as a set of (n+1)-tuples, points in the Cartesian product of the n-dimensional phase space and one-dimensional time. This raw data is then smoothed we store it as a C++ path object, for data reduction and for easier retrieval for analysis and modification. The smoothing is done by approximating the original path through this (n+1)-space with a sequence of spline curves. These splines are also in time as well as in "spatial" dimensions and are computed in the high-dimensional space. This smoothing is also done with a GA, where the bit vector representation of a sequence of spline segments is a vector of fixed-point control points and the fitness function approximates a least-squares error measure integrated over the original path.

Sequences and free play
Paths and surfaces provide visual cues for subregions of a window space. The visual cues provided by a path or surface assist our memory of the sounds that reside in that region. In addition a path provides a control sequence of time-value pairs. Methods for a path include loading and saving to disk, graphical editing, adjusting control points, global modifications such as scaling and translating, and playback at arbitrary timesteps by converting a time to an exact n-tuple in the phase space. Compositional operations may be applied to manifold control paths, including the well-known augmentation, diminution, retrograde, inversion, transposition, and combinatorial operations. The acoustic results depend upon the system the path is controlling, which may include waveform synthesis and larger musical structures such as note patterns or phrases.

Figure 6.
Converging functionality



Free play indicates an exploration of a window space or subspace occurring in conjunction with the playback of a pre-recorded path. While a path playback is initiated as a background event, one can detach the cursor from the path and use it to send additional control messages to the same or to a separate set of parameters. In free play the cursor may move freely in the 3D window space, or the composer may provide boundary conditions around a pre-defined path or surface to explore slight variations of acoustic results. In this context surfaces and paths provide visual maps for *free play regions*. The predefined path and the free play region are intended to be used together to control the synthesis events.

Free play belongs to an *interactive sequence* paradigm, conceptually bounded by notation on one side, exploration on another side, and recorded sequences on a third (Figure 6). Interactive sequences are composed sequences of conditions unfolding in an interactive environment. Real-time variable input with feedback may be performed in parallel with fixed sequences of performance conditions. In addition to free play coupled with path playback, other interactive sequence configurations include path elasticity, or *bending*, where the cursor may be pulled away from a control path while it plays back. We are also in the process of implementing the capability to animate the deformation of surfaces or paths over time.

5. A CHANT synthesis example
In this section we discuss an application of the manifold controller to the CHANT synthesis engine (Rodet, X.). CHANT synthesizes sound from a description of spectral characteristics and a simulation of the output of an excitor-resonator system. CHANT waveforms require the specification of 7 parameters for each formant in the spectral envelope. For best results the spectral envelope should vary over time. We installed the CHANT libraries in the NCSA Sound Server, allowing the manifold interface to generate CHANT sounds in real time. To define a window space we associate specific sounds with specific locations - generating points - in the window space. Configuring a window space for rendering a CHANT waveform required 4 steps:

1. Identify sets of formant parameter values for specific vowel sounds.

2. For each vowel, associate its formant parameter set with a unique 3D position in a window space, creating a generating point.
3. Compute the embedding such that all points in the window space have acoustic properties consistent with those of the generating points (smooth transitions occur between generating points).
4. For the examples in Figure 6, create a path in the window space that visits each generating point.

The formant settings were identified in a published CHANT table. For these examples we rendered three formants, requiring 21 parameters. We decided to hold some parameters fixed while others varied along the control path. For each generating point we defined 8 parameters: the center frequency and bandwidth of the first formant, and the center frequency, bandwidth and amplitude of formants two and three. Four generating points were created; each was assigned a unique vowel sound (/u/, /i/, /e/, or /a:/) and each vowel point was positioned at a unique corner in the window space (Table 1). Amplitude is measured in dB and center frequency and bandwidth in Hz.

Table 1.
Generating points for a CHANT window space

Point	Window Coords (range 0.0 - 1.0) X Y Z	CHANT parameter values Formant 1 CF	BW	Formant 2 CF	BW	AMP	Formant 3 CF	BW	AMP	Vowel /IPA/
1	0 0 0	650	80	1080	90	-6	2650	120	-7	/a:/
2	1 0 1	400	70	1700	80	-14	2600	100	-12	/e/
3	1 1 0	290	40	1870	90	-15	2800	100	-18	/i/
4	0 1 1	350	40	600	06	-20	2700	100	-17	/u/

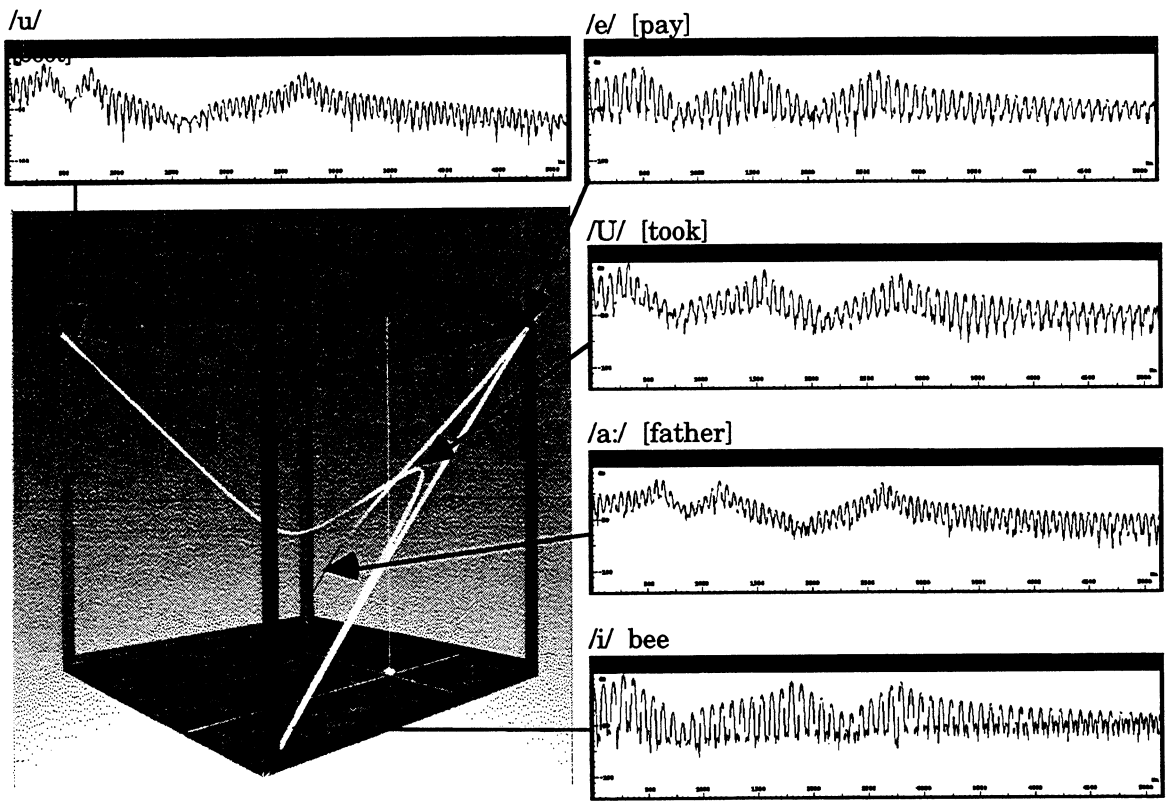
Using the same points as path control points, a path was created passing once through each of the vowels. Signals from five locations on this path are presented in figure 6. Intermediate positions on the path produce intermediate vowel sounds, such as the /U/ which occurs in a location toward the center of the window space. In figure 6 the cursor on the floor is positioned so that its vertical axis intersects the path at the point of the intermediate vowel, /U/.

6. Conclusions

Our experiences navigating in the manifold interface suggest the use of "traffic signals" for performers - text, lines and colors as notation on surfaces or freestanding symbols in window space. Additional orientation may be provided by time navigation tools such as tape-transport-style controls for path playback; by path and surface editing tools; and by displaying multiple surfaces or paths together in a window space. When multiple surfaces or paths are displayed together, each can be used to represent a different synthesis instrument and even a different window space. Multiple cursors can assist in orchestrating concurrent multi-path playback and free play events.

In this interface a continuous control path results from a continuous gesture input. This does not mean the resulting acoustic signal is also perceptually continuous. This also applies to traditional music performance, where not every performance gesture makes a sound. Musical signals include discontinuities, particularly silences. A question arises, how to depict a discontinuity in a control path paradigm? Perhaps a visual break in the path is required, however the discontinuity may not occur in every dimension of the phase space. The problem is magnified if one tries to generate a controlled discontinuity while drawing a path: how to draw a discontinuity with respect to some parameters but not others? One solution is to terminate one path and begin another such that the sound carries across both paths. Another solution is to notate discontinuous regions of a path or surface using a rough texture or broken line segments. Currently we do not provide visual cues for silences and other discontinuities. This approach allows control paths to remain generalized, eliminating the need for excessive one-to-one correspondences, for example between individual note events and individual paths.

Figure 6.
Vowel sounds created by a path controlling CHANT (symbols: /International Phonetic Alphabet/ [English])



A good definition of the window space is critical to all aspects of the Manifold Controller. There is an inevitable information loss as we reduce dimensions. The nature of the information loss affects the size and shape of the manifold region we can control from a given window space. As this problem is difficult and impossible to solve precisely (at least when the phase space has more than 3 dimensions), research must continue here. We feel that traditional statistical data reduction methods for reducing the dimensionality of a set of points in a high-dimensional sample space (e.g., a set of perceptual comparisons (Grey, J.)) are suboptimal for computing window spaces. These methods are in large part designed for finding trends and principal axes of variation in very large data sets, whereas the computation of the map defining a window space typically starts with less than a dozen points. Also, the map is often highly nonlinear because of our demand for continuity. Other methods do not address such nonlinearity. Still, there may well be ways to define maps which retain more information than our current choice of a GA and fitness function.

References

Bargar, R., I. Choi, S. Das, C. Goudeseune. 1994. "Model-based Interactive Sound for an Immersive Virtual Environment." *Proceedings of the International Computer Music Conference,* Aarhus, Denmark.

Choi, I. "Interactive Exploration of a Chaotic Oscillator for Generating Musical Signals in Real-Time Concert Performance." *Special Issue: Chaos and Nonlinear Dynamics, Journal of the Franklin Institute.* June, 1995.

Grey, J. M., and J. A. Moorer. 1977. "Perceptual evaluation of synthesized musical instrument tones." *Journal of the Acoustical Society of America* 62:454-462.

Hearn, D., and M. P. Baker. 1986. *Computer Graphics.* Englewood Cliffs: Prentice-Hall.

Rodet, X., Y. Potard, and J. Barriere. 1984. "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General." *Computer Music Journal* 8(3):15-31.