Characterizing and Controlling Musical Material Intuitively with Geometric Models

Ali Momeni & David Wessel

Center for New Music and Audio Technologies (CNMAT)
Department of Music
University of California Berkeley
1-510-643-9990
{ali, wessel}@cnmat.berkeley.edu

ABSTRACT

In this paper, we examine the use of spatial layouts of musical material for live performance control. Emphasis is given to software tools that provide for the simple and intuitive geometric organization of sound material, sound processing parameters, and higher-level musical structures.

Keywords

Perceptual Spaces, Graphical Models, Real-time Instruments, Dimensionality Reduction, Multidimensional Scaling, Live Performance, Gestural Controllers, Live Interaction, High-level Control

1. INTRODUCTION

Spatial arrangement is a natural way to organize things. Similar objects or objects that belong together are placed proximate to each other and dissimilar or disassociated objects are kept at a distance. Maps are useful for exploration and without them we are quite likely to miss the most interesting paths from here to there. In this paper we describe the development of some general tools that bring such cognitively compelling spatial metaphors to the problem of characterizing and controlling musical material in an intuitive way.

Abstract geometric descriptions of musical material have been around for some time. Shepard [14], Krumhansl [5], and Lerdahl [7] provide numerous examples of organizing pitch in a spatial manner. Cognitive psychologists have made extensive use of techniques like multidimensional scaling to map perceptual and cognitive structures. The study of music perception and cognition has benefited greatly from such techniques. Timbre, rhythm, harmony, and texture spaces have enriched our understanding of the behavior of musical material [19].

Our goal goes beyond the search for insight into the perceptual and cognitive structure of musical material. We are interested in musically expressive control in a real-time live-performance context as well. Geometric models of low dimensionality, say one, two, or three dimensions, provide natural hooks for a large class of controllers such as joysticks, tablets, gloves, etc. The problem is that most interesting musical objects such as timbres, rhythms, and processing algorithms have considerably higher dimensionality. As a result, musically sensible dimensionality reduction is a central focus of our research.

In this paper we present a number of different musical material spaces. We describe tools that make their design easy and their use in performance potentially musically expressive. What is new about this work is that it brings spatial metaphor modeling and dimensional reduction techniques to the actual practice of composing and performing music.

1. HISTORY

The application of multidimensional scaling and related geometric models to audio has a long and rich history and we can touch on but a few of the highlights here. Most of the early studies in the 1960's [9, 10] were carried out by experimental psychologists interested in understanding the mechanisms of auditory perception. Many of the these early studies were purely psychoacoustic in character involving steady state tones and had little direct application to music or even the study of music perception. Other studies, like that of Wedin and Goode [17] used tones from traditional acoustic instruments but were not really carried out with an application to composition in mind. The first published reference we are aware of to the compositional application of multidimensional scaling was made by Milton Babbitt [1].

In the early 1970's one of the authors [20] proposed that the spatial layouts of timbres from multidimensional scaling could function as a palette of musical material whose organization could provide intuitive navigational advice to the composer. John Grey followed suit with his landmark PhD thesis and subsequent experiments [3]. After a number of experimental replications confirmed the basic structure of a timbre space for harmonic acoustic instrument sounds playing the same pitch and loudness [21] the challenge was to show that the spatial layout could actually make predictions about perception of musically viable sequences. Wessel demonstrated that auditory stream formation and rhythmic organization of *klangfarben* sequences could be predicted from a timbre space. It was also demonstrated that a timbre space could be used to specify perceivable timbral transpositions [8, 21].

In 1978, Jean-Claude Risset created a timbre space using multidimensional scaling and used it to compose passages of his work *Mirages* for the Ensemble Intercontemporain. This we believe to be the first application of the technique to a large-scale composition. This piece was performed in the concerts celebrating the official opening of IRCAM that fall.

Other compositional applications followed but admittedly the practice of using perceptual spaces for composition has not fallen into general use. The technique, as it was, is far too tedious.

1. MOTIVATION AND JUSTIFICATION

Our goal here is the make the use of geometric models for the characterization and eventual control of musical material more approachable. The first thing that had to go was the tedium of making pair-wise dissimilarity judgments. Consider that if we wish to work with a 100 different percussion sounds by the classical methods we would be required to make 4950 dissimilarity judgments. The second necessity is a user interface that combines the process of producing the geometric model with the use of the space in real-time performance.

In the mid 1980's experiments in laying out two-dimensional timbre spaces by arranging the sound objects on a screen proved successful.

The spaces laid out in this simple intuitive manner were shown to be very much like those generated by the multidimensional scaling of pair-wise dissimilarity judgments [6, 18]. Further justification for this spatial layout technique is provided by Goldstone [2].

1. SOFTWARE IMPLEMENTATION AND APPLICATIONS

A space designer and explorer was implemented using Cycling 74's Max/MSP and Jitter software. The main patch, named *space-master*, allows one to design a 2-d perceptual space made up of a number of *objects*. Each object can be a recorded sample, a single number, or a list of numbers. Each data point is also the center of a Gaussian kernel, whose value at any given point in the space indicates the weight of its associated data point in the interpolated mixture. The result is a 2-d space that allows weighted interpolation among all data points based on the values of the Gaussian kernels at each point in the space. This implementation aimed to meet several design goals: 1) one unified environment for both design and exploration of a space, 2) a space-designing environment that allows auditioning and real-time adjustments in locations and weights of each data point, 3) the ability to manage large sets of data, 4) the possibility of compelling and mutable graphic representations, 5) the ability to automate movements in the space, 6) the ability to easily save, recall and adjust any parameter in the spaces and to switch among spaces with ease, and 7) the ability communicate with other applications for building, modifying or analyzing spaces. The introduction of Jitter, a set of externals for Max/MSP that allow storage, manipulation and visualization of matrices, made these design goals possible.

1.1 Designing And Using a Space

This section describes the process of designing a space, entering data into the space and using the space. We go into some detail about the data storage methods used in this implementation, as well as some programming details specific to the Max/MSP/Jitter environment.

1.1.1 What Is a Space?

A space is made up of a set of Gaussian kernels whose centers, amplitudes and standard deviations are specified by the user. Each Gaussian kernel is associated with a list of floating point numbers—coordinates in a high-dimensional space—that are also specified by the user. The space is visualized in two dimensions by an image that is a bird's-eye projection of all the Gaussian kernels onto a plane. The space is also visualized by a 3-d surface plot of the kernels.

1.1.1 Creating a Space

The process of designing a space begins by opening the patch *space-master* (Figure 1). The user first defines the "list-length" parameter (dimensionality) and places a desired number of points onto the space using the patch's graphical interface.

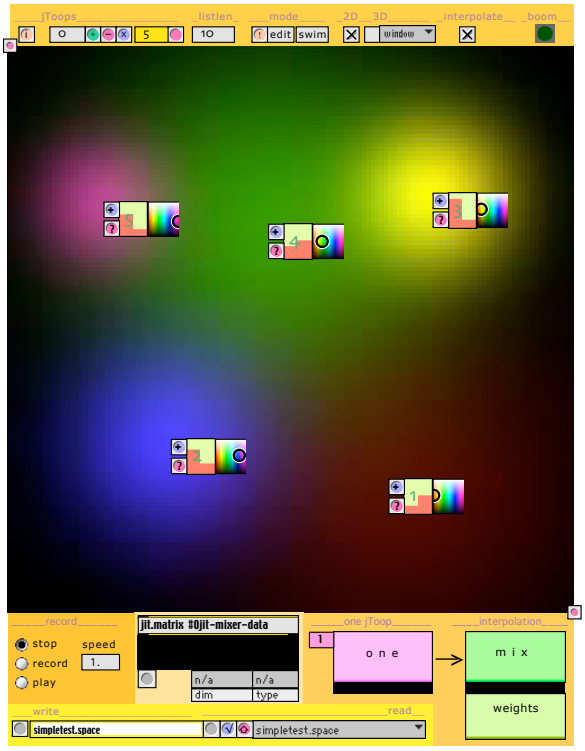


Figure 1. The figure shows the main interface for the *space-master* patch. The large black square is the space being designed; the five rectangular objects in the space are five *one-point* abstractions; the two sliders behind each *one-point's* ID number (1 through 5) are the Gaussian kernel's amplitude and standard deviation. The figure shows the 2-d representation of these kernels as colored regions whose center is beneath the top left corner of each *one-point* abstraction.

The patch dynamically creates—or destroys—an abstraction called *one-point* for each data point in the space. At its creation time, each *one-point* instance is given an ID number and is linked to the main patch. Each *one-point* abstraction has its own user interface which includes sliders for the amplitude and standard deviation of the Gaussian kernel as well as a color swatch for the graphical representation of the kernel. With the main patch in "edit" mode, the user then moves each *one-point* instance to a desired location in the space, selects an amplitude and standard deviation for the kernel, selects a color and clicks the "+" button in the top left of the abstraction. This creates a Gaussian kernel centered at that point (the current mouse location) and links that kernel to the data point whose "+" button was clicked. The background

image of the space is updated to show a 2-d representation of all the kernels in the space by mapping the height of the kernels to a brightness scale applied to the selected color of that kernel. For more accurate visualization of the Gaussian kernels, *space –master* also renders the kernels in 3-d (Figure 2). Gaussian or similar kernels provide not only a mechanism for interpolation but also for extrapolation beyond the perimeter of the points in the space.

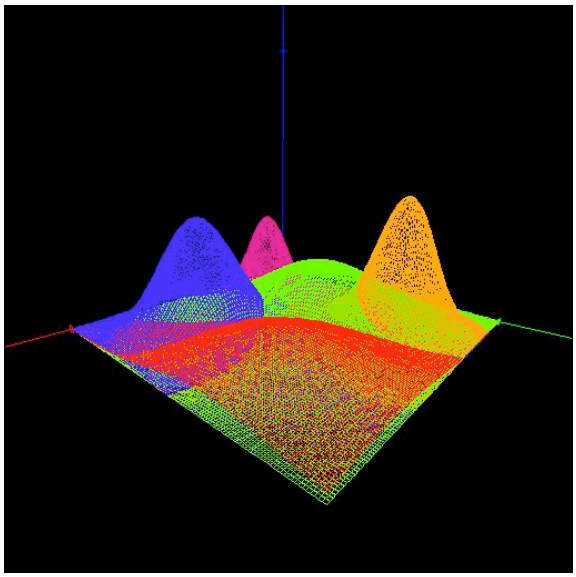


Figure 2. This figure shows a 3-d representation of the same space as that in Figure 1. The color of each Gaussian kernel corresponds to its color in the 2-d representation.

Auditioning while designing a space is a necessary capability. In our implementation, each *one-point* has a button labeled "?" which, when clicked, sends that *one-point's* ID to a global receiver named *space-master-doer*. The destination receive object can then reside in any other Max/MSP patch and perform any desired playback, synthesis or calculation that is appropriate to the space at hand.

Once the space is created, the user puts the patch in "swim" mode; this converts the image representing the space in 2-d to a JPEG format image file. This image is assigned as the background image for a 'pictslider' graphical interface object, a 2 dimensional slider that allows interaction using the mouse or incoming control values.

The space is now ready to be saved; after providing a filename (we will use the name *simpletest.space* for this example) and pressing the "W" button, 4 files are created on the hard disk: three .jxf files (Jitter's binary file format for matrices) that contain the *space, data,* and *points* matrices, and a JPEG file with an image of the 2-d representation of the space. A naming convention is used for consistency: a space by the name of *simpletest.space* would be comprised of four files:

- *simpletest.space.points.jxf*
- *simpletest.space.data.jxf*
- *simpletest.space.space.jxf*
- *simpletest.space.space.jpeg*

1.1.2 Entering Data into the Space

In certain applications of spatial layouts, the weight of each data point at any given coordinate in the space is all that is necessary for performing the interpolation (e.g. see section 4.2.1 below). For these applications, the space is now complete and ready to be used. However, this system also allows each point to represent a list of floating point numbers; it then uses the weight of each data point at a space coordinate as a weight for its associated list in the overall weighted-interpolated mixture of all the lists. For these applications, the user must also provide the lists of floating point numbers among which to interpolate.

Now that the points have been placed in the space and kernels have been created, we are ready to enter data into the data-matrix for the space. This is done within the Max/MSP programming environment by making an instance of the *space-master* patch with an argument indicating the name of the space, e.g. *simpletest.space*. The user communicates with the *space-master* patch using a set of OpenSoundControl messages [22]. For entering data into the space the message '/store' is used: the '/store' messages must be followed by an integer and a list of floating point numbers. The list is then linked to the data point whose ID is the provided integer. For example, '/store 3 0.2 1.4 .8' would store the list '0.2 1.4 .8' as the data for point number 3 in the space.

Once the user has entered lists for all of the points in the space, the space is ready to be used to interpolate among the lists.

1.1.3 Interpolating Among Stored Lists

The '/lookup' message is used for triggering calculations by *space-master*. The message '/lookup' must be followed by two integers. These integers represent the x and y coordinates in the space at which we wish to make an interpolation.

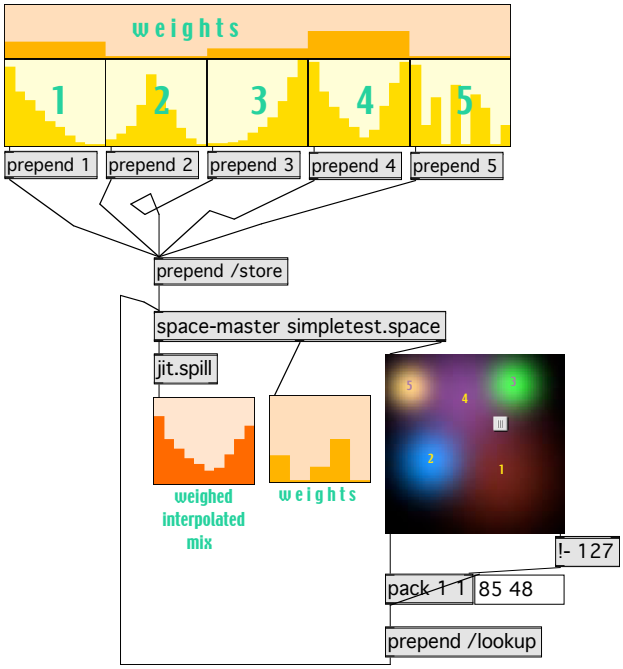


Figure 3. A Max/MSP patch that stores 5 lists into *space-master* and then uses a two dimensional slider to interpolate between the lists.

The '/lookup x y' message will output a list of numbers between 0 and 1 that represent the normalized values of each kernel at the point (x, y) in the space. This list comes out of the *space-master's* second outlet. In a space with 5 data points this list would have 5 items. The '/lookup x y' message will also trigger the patch to calculate the resulting weighted interpolation between all stored lists. That is, the normalized value of each point's kernel at (x, y) is used as the weight for that point's associated list in the weighted-interpolated list. The calculated list comes out of the first outlet of *space-master* (Figure 3).

1.1.4 Data Storage

The main patch uses three Jitter matrices to store all of the data associated with a space:

- *data-matrix*: A 2-d, 1-plane matrix whose rows contain the lists of floating point numbers among which we want to interpolate; in a space with 5 points where each point represents a list of 10 numbers, this would be a 10x5 matrix.
- *space-matrix*: A 3-d, 1-plane matrix that contains the height of the 2-d Gaussian surface for each data point in the space. By default, the designed interpolation spaces are 128x128 points in size; therefore, in our example space with 5 lists of 10 items, this would be 128x128x5 matrix.
- points-matrix: A 1-d, 9-plane matrix whose cells contains all the information associated with each data point in the space:

  1) Whether or not the point is active

  2) Size of interpolation space (default 128)

  3-4) x and y location of the point in the space

  5-6) Amplitude and standard deviation of the Gaussian kernel

  7-9) RGB values for the point's graphic representation

  This matrix it is used to reconstruct a space for making changes.

The three .jxf files created when the user saves a space correspond to the above three matrices.

It is worthy noting that the chosen data storage mechanism not only makes calculations efficient within Max/MSP, it also provides a hook for creating spaces in other applications besides Max/MSP. Although the described method for creating spaces is extremely useful for designing with one's intuition (i.e. spaces based on subjective similarity measures), it can be impractical for very large sets of data. There are, however, numerous computational techniques in the large body of research that addresses the fundamental problem of dimensionality, that allow one to derive locations in a 2-d space from large sets of high-dimensional data. These algorithms are often implemented in applications like MatLab. We have developed a number of auxiliary Max/MSP patches which translate matrices exported from another application as delimited text files, into Jitter matrices ready to be used by *space-master*.

1.1.5 More Detailed Notes on Max/MSP/Jitter Implementation

When designing a space, the *one-point* abstractions created for each point in the space are instantiated using Max/MSP scripting capabilities. Each instance is connected to the main patch using 'send' and 'receive' objects. 'Send' and 'receive' objects are used as opposed to Max's patch-chords in order to have a less cluttered space-designing environment. Furthermore, each *one-point* abstraction is instantiated as a 'bpatcher'; this allows the user to see the abstraction's own interface and adjust parameters for that point.

The entire functionality of *space-master* is accessible through a set of OpenSoundControl messages [22]. The patch *space-master* understands a large number of OSC messages including read/write commands, clearing commands, turning interpolation on/off, turning 3-d rendering on/off, adding/removing/modifying data points and their kernels. The technique of using OSC as a communications scheme allows efficient management of multiple spaces within one application [23].

All data storage for *space-master* is done using Jitter matrices. This allows us to work with dimensionalities and numbers of points in the space that are larger than 256, Max's inherent limit on the number of items in a list. In addition to using Max's lists for entering data into the data-matrix, data can also be entered using the name of a Jitter matrix after the '/store n' message. Similarly, the weighted interpolated list produced by *space-master* is output as a 1-d, 1-plane Jitter matrix containing 32-bit floating points, again to avoid the 256-item limit of lists in Max. The matrix can be easily converted to a list using the 'jit.spill' object.

Jitter's OpenGL functionality brings the benefits of hardware-accelerated 3-d graphics and its amenities like the ability to freely rotate the 3-d objects or zoom in and out of the surface plots. To take advantage of this feature, the 3-d representation of the space rendered by *space-master* is produced using OpenGL.

1.2 APPLICATIONS

Numerous applications of the interpolation technique were developed using Max/MSP and Jitter. These applications, which are generally in the form of real-time performance instruments, aim to provide the composer/performer with high-level control of a process that functions in a high-dimensional space, that is, a process that has a large number of control parameters. The described system allowed us to use our musical intuition to define the spaces. This in turn made performing with these spaces intuitive and rewarding.

1.2.1 Drum Space: Timbre Space of Percussion Samples

A simple application of this interpolation technique is a 2-d timbre space designed to interpolate among percussive sounds. The strong similarity in the amplitude envelopes for percussive hits contributes to very effective fusing between the sounds and thereby makes them ideal candidates for weighted-amplitude mixing. These samples, 38 low sounding membranophones ranging from tympanis and concert bass drums to a south Indian mirdangam and a classic Roland 808 kick drum, were laid out subjectively on the 2-d space to achieve the musical goal of a continuous space of low drum sounds that contains interesting mixtures (Figure 4).

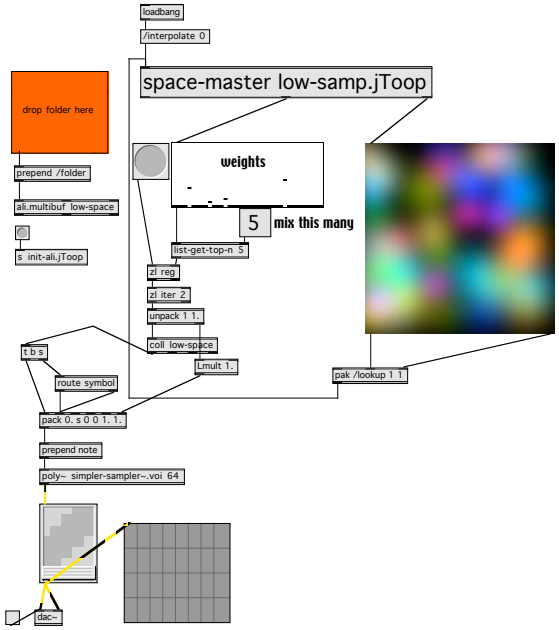


Figure 4. Each colored region in the space represents an audio sample of a low drum. As the user navigates around the space, the list of normalized weights at that coordinate in the space is output from *space-master's* second output. Each time playback is triggered, the values in this list are used to weigh the amplitudes of the corresponding samples in the mixture.

In this example, the list of normalized weights at a given coordinate in the space is the only required data for producing a weighted mix of the samples; in other words, the data matrix in this space is empty. List interpolation was therefore turned off to preserve computation power.

An additional parameter in the patch designates how many samples to mix at a time. Experimentation proved that it was computationally wasteful to mix any more than 5-7 samples at a time since the relative amplitudes for samples beyond the strongest 5-7 contributors were low enough to render them unnoticeable in the mix.

1.2.2 Res Space: Timbre Space Based on Transformations of Resonance Models

Models of resonance provide an efficient and highly mutable approach to sound synthesis. CNMAT's *resonators~* object for Max/MSP [4] supply an implementation of parallel banks of 2-pole resonators and a set of possible transformations for existing models. When provided with a frequency, an amplitude and a decay-rate for each frequency component in the resonance model, *resonators~* efficiently models a bank of resonating filters with the given parameters that can then be excited by an impulse, enveloped noise or other audio signals. The *res-transform* [4] object allows one to apply global transformations to a resonance model by way of a large number of scaling, adding, or component producing functions. Together they allow musicians to produce an extremely wide range of sounds from one set of data, usually derived from analysis of a recorded sound.

One obstacle to finding musically satisfying results with *resonators~* and *res-transform* is the complexity of the interdependencies between various transformation parameters. For instance, the 'spectral-slope' and 'spectral-corner' parameters to *res-transform* work together to redistribute energy to different parts of the frequency spectrum. Similarly 'cluster-size', 'frequency-around', 'attenuation-spread' and 'rate-spread' all contribute to the interesting timbres and intricate beating patterns that result when additional frequency components are created from ones already extant in the model. Furthermore, parameters like 'rate-scale' (which changes the decay rates of the frequency components) can significantly affect the perceptual loudness of the model.

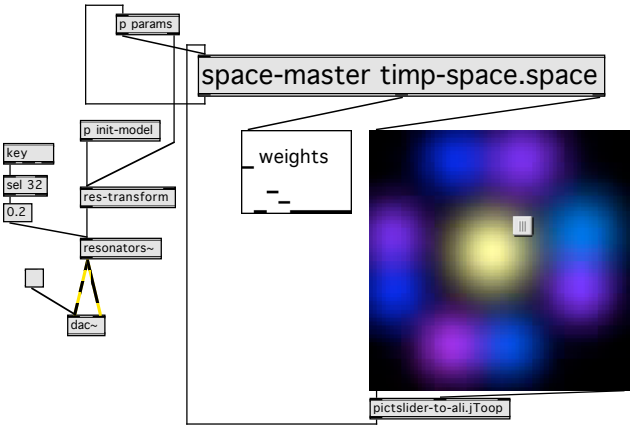


Figure 5. The space in the figure above shows the resulting models as nine colored regions where the center represents the unaffected model; the models are arranged in pairs to convey the local similarities within the larger space. The two regions to the right, for instance, both scale the frequency of the model upwards, lower the decay-rates and adjust the gains to achieve uniform loudness but each slightly differently, whereas the regions in the bottom are two low sounding models made up 2-times clusters with differing beating patterns.

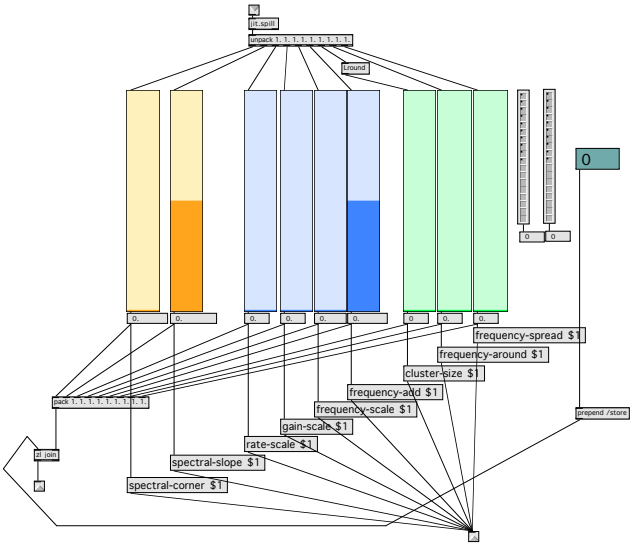


Figure 6. The list of floating point number representing a weighted interpolation among sets of *res-transform* parameters is unpacked, formatted and sent to *res-transform* out of the right outlet.

A timbre space was produced around the resonance model of a tympani and eight transformations of that model (Figure 5). In addition to the unaffected model consisting of 50 frequency, amplitude, decay-rate triplets, eight additional models were produced by carefully exploring what *res-transform* can do to the model; specifically, additional models were found by experimenting with the following eight parameters of *res-transform:* 'spectral-slope', 'spectral-corner', 'rate-scale', 'gain-scale', 'frequency-scale', 'frequency-add', 'cluster-size', 'frequency-around', 'frequency-spread'. Desirable sets of these *res-transform* parameters were then stored and interpolated among using *space-master* (Figure 6).

Two features of this approach to managing related models of resonance are noteworthy: first, interesting musical results are quite often found when exploring the space in between the known regions. In regard to live performance control, this provides not only a continuously mutable model for synthesis of percussive sounds, it also opens the way for countless new sonic possibilities derived from those designed by the composer. Second, the use of Gaussian kernels—as opposed to Euclidian distance for instance—is crucial in this application. Since many of the transformations apply frequency-scaling to the model, using weight function that are not adjustable by way of a parameter like standard deviation would result in a timbre space that is filled with glissandi. In our experience, continuously changing pitch adds a transparent synthetic quality to the sound which detracts from the efficacy of the instrument a drum-like entity. By using adjustable Gaussian kernels, however, these glissandi can be effectively localized to one region of the space.

It is worth noting that a timbre space of resonances could also be constructed by interpolating among the much larger lists of frequency/amplitude/decay-rate for each model (as opposed to interpolating among lists of *res-transform* parameters). The advantage of this approach is that one could use completely unrelated models as the points in the space. The disadvantages are: first, since the models are unrelated effectively localizing the aforementioned glissando problem and still getting a satisfying level of blending between the models becomes much more difficult; second, since a timbrally rich model often requires at least 30 frequency components (i.e. yielding 90 floating point numbers to represent each model), the interpolation among them becomes much more computationally intensive.

1.2.3 Reverb Space

We step away now from the realm of synthesis and into that of processing. In the recent years it has become feasible to provide very high quality reverberation entirely in software. This is evident in numerous high quality reverberators now available in the form of VST plugins in use in many professional studios. The parameter space for these units, however, is often very high in dimensionality and therefore difficult to maneuver beyond simply switching from one preset to another. In another application of our dimensionality reduction technique, a space of reverb settings was constructed to control TrueVerb, room modeler developed by Waves (Figure 8).

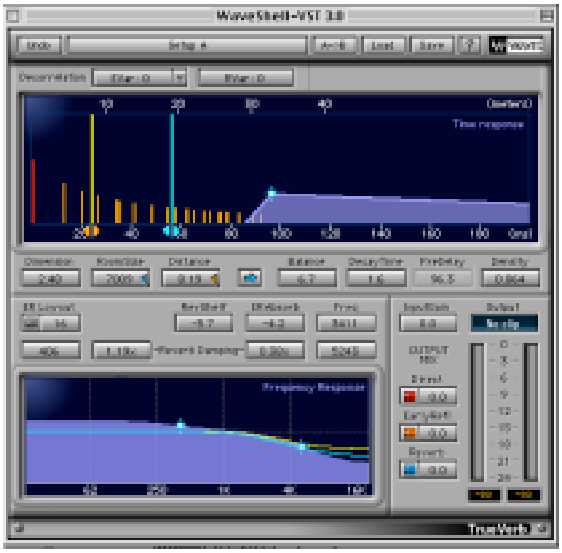


Figure 8. The user interface for the VST plugin Waves TrueVerb Room Modeler. This plugin, like many other high quality processing units has a large number of finely adjustable parameters—too many to control independently in real-time.

TrueVerb, which "combines two separate modules - an Early Reflections simulator, and a Reverb - to produce a high quality, natural-sounding room effect" [16] has a total of 45 parameters. As composers and performers interested in having high-level control over reverberation without having to manage 45 knobs, we developed a patch that contains a reverb space with six regions corresponding to six distinct settings of the TrueVerb module (Figure 9). Moving about in this 2-d reverb space shifts from the sound of a small hall, through that of a dampened practice room, and into a lush stadium. Changes occur gradually and smoothly, in a way that would be impossible to reproduce if the parameters were to be controlled individually in real-time.

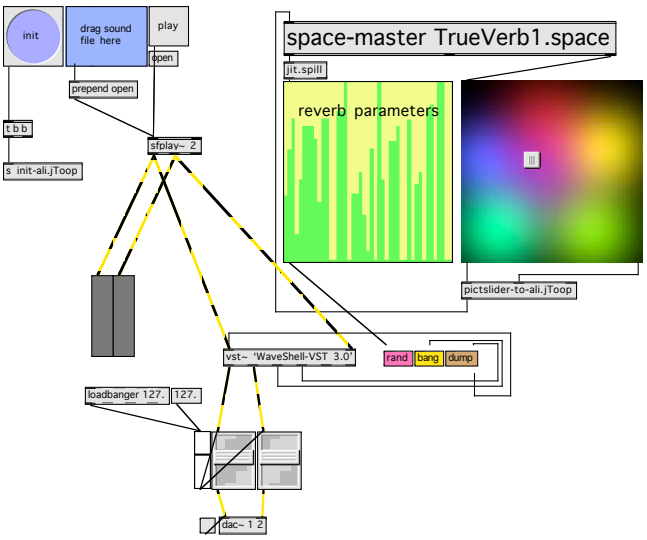


Figure 9. Patch developed to interpolate among various reverb settings of the VST plugin TrueVerb by Waves.

1.2.4 Grain Space: Amplitude Envelopes, Waveforms, Durations and Harmonic Content of Granular Clouds

A real-time performance instrument was developed around the concept of granular clouds (Figure 10) [12]. The instrument is a generalized granular synthesizer that polyphonically produces grains of sound, that is enveloped waveforms with a given frequency, amplitude and durations. It uses probability tables to select the duration and pitch of each grain while the amplitude envelope and the waveform are results of *space-master* interpolations. Four separate *space-master* modules with the following contents were used:

1. Five amplitude envelope types
2. Nine 512-sample wave tables for the waveform
3. Five different probability tables for duration
4. Fourteen different probability tables for the pitch

Each time the instrument is triggered, it probabilistically chooses a duration between 10ms-3000ms, a frequency between 40HZ-5000HZ, and it uses the current interpolated waveform and amplitude envelope to play a grain. Each instance of this instrument uses a 64-voice polyphonic player to allow extremely dense clouds of grains whose control parameters are controlled in real-time using the spatial arrangement.

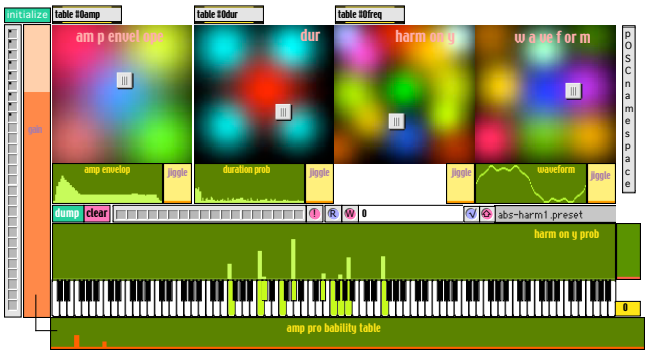
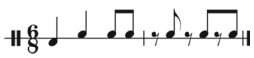


Figure 10. The graphical interface for the Grain Space instrument. The four colored spaces in the top area of the patch correspond to amplitude envelope, probabilistic grain duration, probabilistic grain pitch, and the waveform in the grain. The smaller interface objects below each space (with the exception of the "harmony" space) show the current interpolated mix. The interface comprised of a keyboard and set of sliders visualizes the current interpolated probability distribution for the grain's pitch. Note that an additional 'jiggle' parameter is provided for each space to allow for slight—or drastic—deviations in the parameters of the grain by jiggling the location in each interpolation space by a given percentage.

In performance, it is possible to use multiple instances of this instrument too produce contrasting granular clouds with entirely different sets of parameters.

1.2.5 Beat Space: High-level Control of Rhythmic Material Using Spatial Layouts

A real-time instrument was developed to allow performance of probabilistic variations derived from a simple accent pattern. For an example of this instrument's usage we choose the omnipresent west African bell pattern:



We represent this rhythm as the duration vector

[2 2 1 2 2 2 1]

where the eighth note is the lowest duration-value, represented by the number 1 in the above vector. The patch then assigns probabilities to each beat in the pattern. Beats that contained hits in the original pattern are assigned high probabilities, while the beats in between are assigned low ones. Working with probability ranges between 0 and 1, the above pattern would be deterministically represented the probability vector

[1 0 1 0 1 1 0 1 0 1 0 1]

Note that there are a total of 12 probabilities, each corresponding to a beat in the pattern; the 1's represent eighth note slots where there was a hit and the 0's represent rests.

The patch then creates a perceptual space comprised of 5 regions (Figure 11). The region in the center represents the original pattern by way of the above deterministic beat-wise probability vector. The region to the right represents the opposite, a probability vector emphasizing all of the beats missing in the original patter, that is,

which is represented by the beat-wise probability vector:

[0 1 0 1 0 0 1 0 1 0 1 0]

The region at the top of the space is assigned the densest possible pattern made of the minimal pulse value (every eighth notes's probability equal to 1) and the region to the bottom the opposite (every eighth note's probability equal to 0). The region to the left is initially assigned the same probability values as the original pattern, but is reserved for user-defined probability. The user can draw a new probability vector and click the button to the left of the probability display to store the probability vector in the region in the left of the space. By navigating within this space, the user is able to produce rhythmic patterns that range in level of syncopation in relation to the original pattern (from center to right of space) and overall density (the vertical axis of the space).

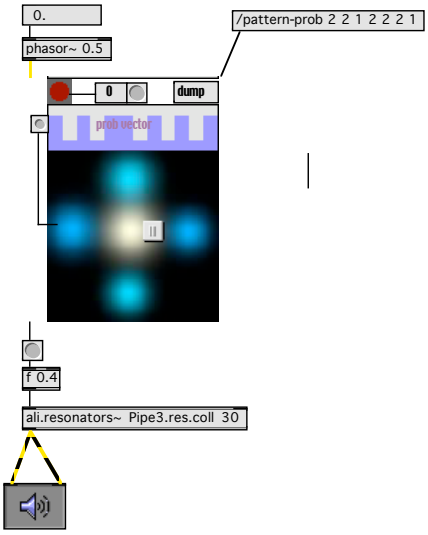


Figure 11. The interface for the Beat Space instrument. The perceptual space is made up of 5 regions whose associated data is calculated by the patch each time a new rhythmic pattern is entered into the patch.

1.2.6 Boids Space: Perceptual Space for the Parameters of a Bird Flocking Algorithm

A real-time performance instrument was developed that utilized Eric Singer's Max/MSP implementation [15] of Craig Reynolds's famous bird flocking algorithm [11]. This Max/MSP externals, named Boids, models the flight path of a designated number of birds with regard to a set of 17 parameters (e.g. centering tendency, maximum speed, inertia, repelling tendency, etc.). Different settings of these parameters give very particular results in the overall shape and movement of the flock (Figure 12).

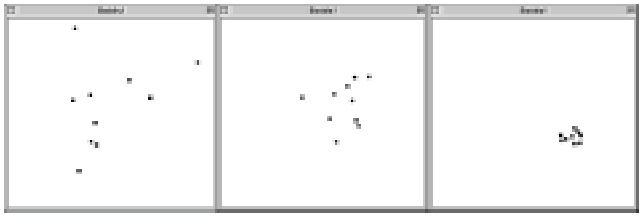


Figure 12. This figure shows three different arrangements of the parameters to the Boids bird flocking algorithm. Each black dot represents the location of a "boid" in the flock's flight area. As parameters for the algorithm are changed, the flight tendencies of individual birds change, thereby changing the overall shape and behavior of the entire flock.

In the Boids Space instrument, designed for live interaction with a MIDI piano, each bird was represented in sound by one voice of a resonators~-based synthesizer. The vertical position of the bird was mapped to quantized tempo, and the horizontal position to pitch register. Pitch material was extracted in real-time from the MIDI piano, and applied to the current pitch register of each bird. The performer using Boids Space controls the attraction point for the flock of birds, as well as the overall flocking tendencies by way of a perceptual space made up of seven predefined sets of Boids' parameters (Figure 13). He also controls the overall amplitude and resonance for the synthesizers, The overall effect was a tremendous amount high-level of control over number of independent voices. By navigating the space for the Boids algorithm's parameters, it was possible to find interesting regions of varying rhythmic and registral correspondence.

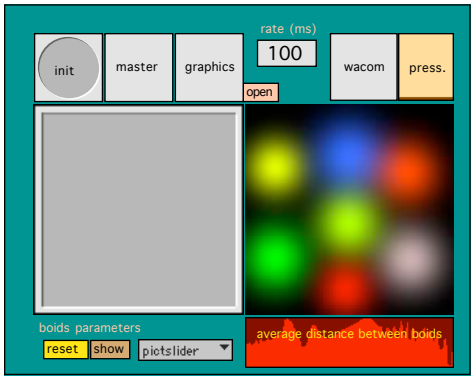


Figure 13. The interface for the Boids Space Instrument. The 2 dimensional slider on the left controls the attraction point for the flock of birds, the colored space on the right allows the user to interpolate among seven different sets of parameters to the Boids algorithm. Average distance between the Boids was calculated and mapped to reverberation wetness.

1.2.7 SPACE Space

Finally, we introduce the notion of creating spaces of spaces. As conveyed by a number of the previous examples, sophisticated applications of geometric arrangements in performance environments can involve multiple perceptual spaces in one instrument. The same method of dimensionality reduction that is applied to each individual perceptual space can also be applied to the entire system in order to organize specific arrangements of the individual components. Specifically, a space of spaces can be created by using a mother-space to interpolate between sets of coordinates in the daughter spaces (Figure 13). For example, in order to control four daughter-spaces with one mother-space, one would store 8-membered lists that contains desirable x-y coordinates for each daughter-space in the mother-space and interpolate between these lists. The result is an even higher level of control over a system with a very high number of parameters, by way of navigation in a space that has only two dimensions.

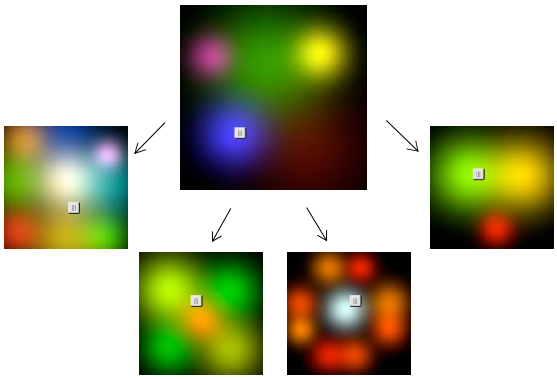


Figure 13. This figure depicts the notion of creating spaces of spaces. The larger space in the center allows one to interpolate among sets of coordinates in the 4 smaller spaces below. Each of the smaller spaces subsequently controls the parameters to a different musical process.

2. CONTROLLERS

Use of gestural controllers for real-time performance is an ongoing focus of research at CNMAT as well as in the interactive computer music community in general. At CNMAT we have developed a great deal of software for controllers like the Buchla Thunder, Saitek Cyborg 3D Joysticks and Wacom drawing tablets. The reduction of high-dimensional parameter spaces down to 2 dimensions has proved to be an extremely effective technique for controlling real-time computer instruments at a high level. The paradigm is simple: gestural controllers translate a performer's physical gestures in space into streams of data for controlling musical processes. Although some controllers can output more than 3 unique streams of data as a result of a single physical gesture, they most often result in 1 to 3 dimensions of control data. In order to allow intimate control over a sophisticated musical process, it is often necessary to control many more parameters than a simple one-to-one mapping of a controller allows. Let us take the task of controlling reverb by way of a Wacom drawing tablet as an example. The Wacom interface serves as a good example because it in fact outputs five continuous streams of data from a single gesture with the pen (horizontal and vertical position, 2 dimensions of tilt, and pressure). However, even with such rich output from the controller, a one to one mapping of its five dimensions to a high quality reverberator

like Waves TrueVerb would give the performer very little control, for the reverb unit has not five but 45 parameters. It is also worth mentioning that with the Wacom tablet, as is the case with many other controllers that output more than 3 continuous streams from the same gesture, it can be exceedingly difficult to reliably reproduce a gesture in performance. On the other hand, the horizontal and vertical positions alone can be mapped to a perceptual reverb space like the one described earlier, thereby giving the performer full control over this process with maximal accuracy and reliability in the gesture-to-control mapping. Furthermore, visual feedback of the process under control by the performer becomes a much simpler problem when the controllers physical existence in 3 dimensions is directly correlated with an intuitively laid out perceptual space that functions in 2 dimensions.

3. FUTURE WORK

Besides building more real-time instruments that make use of graphical layouts of high-dimension spaces, we intend to experiment with performing transformations on the spaces in real-time. That is, by moving points around in the space or by transforming their kernels, one can find entirely different sets of interpolated results from the same data set. These transformations would not only expand the palette of musical capabilities that an instrument has, they could also elucidate structural similarities and dissimilarities in the data that may not have been evident in the original spatial layout.

We also plan to work more with purely numerical techniques for creating spaces. The technique of Locally Linear Embedding as demonstrated by Sam T. Roweis and Lawrence K. Saul's work [13] seems extremely promising in its applications to our work with real-time instruments and our desire to organize and perform with very large sets of data.

4. ACKNOWLEDGMENTS

Our thanks to Joshua Kit Clayton, Adrian Freed, Eric Singer, Matt Wright, and the Max/MSP/Jitter community for the help with the development. We also thank David Bithell, Ben Jacobs and Roberto Morales, for their most insightful musical interactions with our instruments.

5. REFERENCES

[1] Babbitt, M. (1965) The use of computers in musicological reserach. *Perpectives of New Music*, *3* (2).

[2] Goldstone, R.L. (1994) An efficient method for obtaining similarity data. *Behavior Research Methods, Instruments, & Computers*, *26*, 381-386.

[3] Grey, J.M. (1977) Multidimensional perceptual scaling of musical timbres. *Journal of Acoustical Society of America*, *61*, 1270-1277.

[4] Jehan, T., Freed, A. and Dudas, R., Musical Applications of New Filter Extensions to Max/MSP. in *International Computer Music Conference*, (Beijing, China, 1999), ICMA, 504-507.

[5] Krumhansl, C.L. *Cognitive Foundations of Musical Pitch*. Oxford University Press, Oxford, 1990.

[6] Krumhansl, C.L. Why is musical timbre so hard to understand? in Nielzén, S. and Olsson, O. eds. *Structure and perception of electroacoustic sound and music*, Elsevier, Amsterdam, 1989, 43-53.

[7] Lerdahl, F. *Tonal Pitch Space*. Oxford University Press, 2001.

[8] McAdams, S. and Cunibile, J.C. (1992) Perception of timbral analogies. *Philosophical Transactions of the Royal Society, London* (Series B 336), 383-389.

[9] Plomp, R. Timbre as multidimensional attribute of complex tones. in Plomp, R. and Smoorenburg, R.G. eds. *Frequency analysis and preiodicity detection in hearing*, Suithoff, Leiden, 1966.

[10] Plomp, R. and Steeneken, J.M. (1969) Effect of phase on the timbre of complex tones. *Jounal of the Accoustical Society of America*, *46*, 377-380.

[11] Reynolds, C.W., Flocks, Herds and Schools: A Distributed Behavioral Model, in Computer Graphics. in *SIGGRAPH*, (1987), 25-34.

[12] Roads, C. *Microsound*. MIT Press, Cambridge, Ma, 2001.

[13] Roweis, S.T.S., Lawrence K., *Locally Linear Embedding*. 2003, (http://www.cs.toronto.edu/~roweis/lle/).

[14] Shepard, R.N. Structural Representations of Musical Pitch. in Deutsch, D. ed. *Psychology of Music*, Academic Press, New York, 1982, 343-390.

[15] Singer, E., *Boids*. 1998, (http://www.ericsinger.com).

[16] Waves, *TrueVerb Room Emulator*. 2003, (http://www.waves.com/htmls/prods/indi/tv.html).

[17] Wedin, L. and Goude, G. (1972) Dimension analysis and the perception of instrumental timbre. *Scandinavian Journal of Psychology*, *13* (3), 228-240.

[18] Wessel, D., Bristow, D. and Settel, Z., Control of Phrasing and Articulation in Synthesis. in *International Computer Music Conference*, (Champaign, Urbana, USA, 1987), International Computer Music Association, 108-116.

[19] Wessel, D. and Wright, M. (2002) Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal*, *26* (3), 11-22.

[20] Wessel, D.L. (1973) Psychoacoustics and music: A report from Michigan State University. *Bulletin of the Computer Arts Society,*, *1* (30).

[21] Wessel, D.L. (1979) Timbre space as a musical control structure. *Computer Music Journal*, *3* (2), 45-52.

[22] Wright, M. and Freed, A., Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. in *International Computer Music Conference*, (Thessaloniki, Hellas, 1997), International Computer Music Association, 101-104.

[23] Wright, M., Freed, A., Lee, A., Madden, T. and Momeni, A., Managing Complexity with Explicit Mapping of Gestures to Sound Control with OSC. in *International Computer Music Conference*, (Habana, Cuba, 2001), 314-317.