Real-Time Feature-Based Synthesis for Live Musical Performance

Matt Hoffman
Dept. of Computer Science, Princeton University
35 Olden St.
Princeton, NJ, USA 08540
mdhoffma at cs.princeton.edu

Perry R. Cook
Depts. of Computer Science, Music,
Princeton University
35 Olden St.
Princeton, NJ, USA 08540
prc at cs.princeton.edu

ABSTRACT
A crucial set of decisions in digital musical instrument design deals with choosing mappings between parameters controlled by the performer and the synthesis algorithms that actually generate sound. Feature-based synthesis offers a way to parameterize audio synthesis in terms of the quantifiable perceptual characteristics, or features, the performer wishes the sound to take on. Techniques for accomplishing such mappings and enabling feature-based synthesis to be performed in real time are discussed. An example is given of how a real-time performance system might be designed to take advantage of feature-based synthesis's ability to provide perceptually meaningful control over a large number of synthesis parameters.

Keywords
Feature, Synthesis, Analysis, Mapping, Real-time.

1. INTRODUCTION
How best to map from performer intention to sonic output is a fundamental question in the design of musical instruments intended to be played in real time [4]. In those cases where an electronic interface is used to control a software synthesizer, the problem often reduces to how best to map from some set of continuous or discrete control signals to a set of synthesis parameters that provide control over the signal processing algorithms that generate the synthesizer's audio. In some cases, these synthesis parameters have a clear, one-dimensional, easily learned effect on the perceptual content of the sound. An example might be the frequency of an oscillator, which (often) maps clearly to the percept of pitch. Sophisticated synthesis algorithms, however, may have parameters whose effects may be individually quite subtle, unpredictable, or dependent on the settings of other parameters. Furthermore, there may be a large number of these parameters - more than can be explicitly controlled in real-time by a human being [7].

As a result, real-time control signals are frequently mapped only to a small number of parameters, or a small number of controller parameters are mapped onto a larger number of synthesis parameters. In the former case, the full expressive range of the controller-synthesizer pairing may not be exploited. In the latter case, a satisfactory one-to-many mapping must be determined. This can be done by hand by the instrument designer, using what is likely to be a time-consuming (although potentially insight-provoking) process of trial and error. Alternatively, the mapping can be determined automatically by the computer, according to some set of criteria. When designing a system that performs an automatic mapping of this kind, the crucial question becomes how to define the criteria that the computer uses to design the mapping.

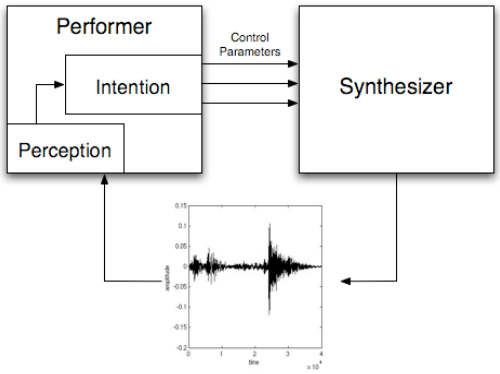

Figure 1. A performer can control a few synthesizer parameters directly, continuously making adjustments to make perception match intention.

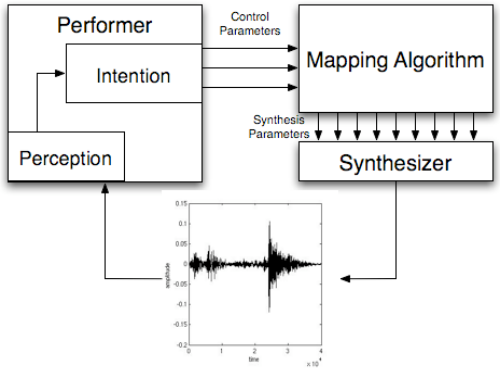

Figure 2. Alternatively, a performer can control many parameters using only a few control signals by means of an intervening mapping layer.

Nime'07, Jun 7-9, 2007, New York, NY, USA.

Our approach looks to research in psychoacoustics and music information retrieval to answer this question. Many music information retrieval systems begin by extracting feature vectors of perceptually motivated features describing successive frames of audio. The individual features are simply numbers describing the audio frame in terms of its characteristics along some set of perceptual dimensions such as loudness, timbre, brightness, pitch, and harmonicity. The goal of frame-level feature-based synthesis is to, given values for an arbitrary set of these features and the ability to compute those features on new audio, produce a frame of audio that manifests those feature values. In other words, feature-based synthesis allows us to more directly parameterize audio synthesis in terms of its quantifiable perceptual content.

2. PREVIOUS WORK

There have been numerous attempts to automatically find mappings from a few control parameters to many synthesizer parameters over the years. Lee and Wessel [7] and Johnson and Gounaropoulos [6] (among many others) have used neural networks to try to learn user-defined timbre spaces. More recently, Bencina [1] implemented a sort of mapping-by-example approach based on nearest neighbor interpolation.

The importance of intelligent parameter mapping to gestural control has been emphasized by (among others) Hunt, Wanderley, and Verfaille [4, 12, 13]. Verfaille and Arfib [11] specifically proposed including a feature-aware mapping layer to add an adaptive component to digital audio effects control.

Similar ideas to those used in feature-based synthesis have been used in numerous papers by Andrew Horner et al [3, 14], although to different purposes and in a less general way. Much recent work on concatenative synthesis and audio mosaicing by Diemo Schwarz and others pursue similar goals using methods similar to feature-based synthesis, although applied to samples of previously recorded material [9].

3. FEATURE-BASED SYNTHESIS

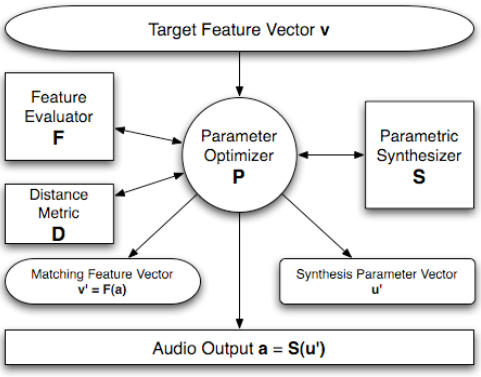

Figure 3. Overall system architecture.

We have implemented a general framework for synthesizing audio frame by frame to match any set of feature values as they change over time [2]. Our architecture divides the tasks to be performed between four main modular components: *parametric synthesizers, feature evaluators, distance metrics*, and parameter optimizers. *Feature evaluators* take a frame of audio as input and output an n-dimensional vector of real-valued features. *Parametric synthesizers* take an m-dimensional vector of real-valued inputs and output a frame of audio. *Distance metrics* define some arbitrarily complex function that determines how "similar" two feature or parameter vectors are. Finally, *parameter optimizers* determine how best to map from an arbitrary vector of feature values **v** to a vector of synthesis parameters **u'**.

An instrument designer can connect this framework to a hardware controller that sends various control parameters to the computer in real-time by choosing a set of features and a synthesis algorithm, mapping the control signal values directly onto feature values, and choosing an appropriate parameter optimizer and distance metric (more on this step below). So long as the parameter optimizer does not take too long to find a set of synthesis parameters that produces audio with the desired feature values, the system can be used to produce sound in real time. Alternatively, feature vectors can be extracted from sounds generated acoustically by the performer (vocal sounds, for example) and used to control the synthesis algorithm. The resulting audio will match whatever perceptual content in the performer-generated audio is captured by the features used, but will likely sound quite different in ways characteristic of the synthesizer used. See Janer [5] and Loscos and Aussenac [8] for feature/synthesizer pairing-specific examples of this sort of approach.

3.1 Distance Metrics

Distance metrics judge the similarity of two vectors of features or parameters. They may be defined in any arbitrary way, but a natural choice is the LN norm of the difference of the two vectors. The LN norm is defined as the Nth root of the sum of the Nth powers of each element, so for example the Euclidean norm is identical to L2, and the Manhattan norm (simply the sum of absolute values on each dimension) is L1. We provide a standard metric that implements these norms and allows higher or lower weights to be assigned to more or less important features or parameters.

For certain synthesizers and small to moderately sized feature sets there may be numerous quite dissimilar parameter settings that produce audio with similar feature characteristics. Rapidly switching between these parameter settings in pursuit of small improvements in feature distance can lead to undesirable artifacts, and so our framework allows for the calculation of distances in parameter space as well as feature space. This parameter distance can be combined with the feature distance as a secondary criterion in parameter selection in the interests of smoother-sounding audio. For example, when looking for a good parameter set one might give parameter distance 1/4 the weight of feature distance.

Although these metrics can be created by hand with a minimum of effort, our system also allows each feature evaluator to define its own default distance metric that attempts to more accurately capture the overall perceptual distance between two feature vectors extracted using that feature evaluator. Synthesizers are also able to define default metrics, for example by weighting parameters with more dramatic effects more heavily than subtler parameters.

3.2 Parameter Optimizers

The parameter optimizers used in our framework are not necessarily expected to have any a priori knowledge about possible relationships between synthesis parameters and feature values. Typically, they apply an iterative search algorithm such as simulated annealing to minimize the distance between the target feature values and the feature values obtained by analyzing the synthesizer's output. Such algorithms run in a loop such as the

one shown in figure 4, generating a frame of audio, analyzing that audio, comparing the resulting features with the target features (and optionally the tested synthesis parameters with the previous frame's synthesis parameters), and using the resulting distance to inform the choice of parameters for the next iteration. This continues until suitably similar feature and parameter values are obtained.

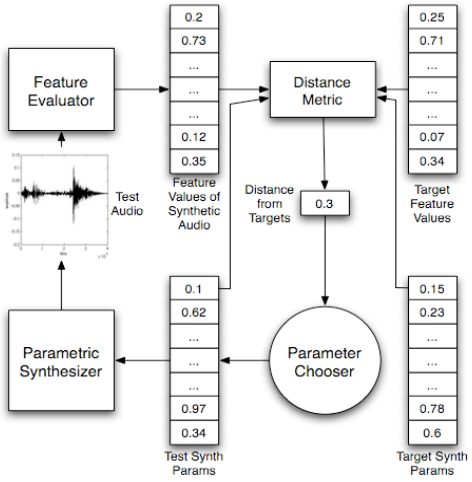

Figure 4. Optimizer loop.

Although this approach permits a great deal of flexibility and robustness, it may take a large number of iterations to find an acceptably close match. This presents a problem for real-time applications, since a frame of audio must be synthesized and analyzed at significant computational cost each iteration.

4. REAL-TIME PERFORMANCE ISSUES

In order to speed up the process of finding synthesis parameters that produce the desired feature values, we need to incorporate some information about the relationships between the parameters of the chosen synthesizer and the features we are attempting to match. Unfortunately, we are not in general guaranteed that any simple, predictable relationship between feature values and synthesis parameters exists. The actual relationships may be highly nonlinear, rendering them difficult or impossible to describe in closed form.

If, however, we can safely assume that a given set of synthesizer parameters will always result in nearly identical feature values, then we can perform many expensive synthesis/feature evaluation steps offline and store the results in a database for future reuse. As long as retrieving these parameter-feature mappings can be done efficiently, when we need to obtain good synthesis parameters for a given set of feature values quickly (i.e. during live performance) we can find the best match cached in the database. These parameters can be used to produce audio immediately, or they can be used to give our parameter optimizer a head start on the process of finding still better parameters.

4.1 Filling the database

The simplest approach to getting useful information into the database is to simply generate and test random parameter vectors, saving the results. This will maximize the variety of parameter values represented in our database, but ultimately we are more likely to want to maximize the variety of feature values stored. Since it may be that large segments of the parameter space map to more or less identical feature values while a very small space of parameter sets might map to a large portion of the feature space, another approach may yield faster results.

One option is to use an iterative parameter optimizer to search for random feature values instead of using random parameter values. The optimizer can be set up to cache each parameter-feature mapping it tests while searching for the best match for the chosen random features. This often has the added bonus of storing an entire path through feature space from one feature vector to another, and hopefully yields a somewhat more uniform distribution of information about feature values in the database.

4.2 Accessing the database

Since the number of possible sets of feature values grows exponentially in the number of features, it may be that for some feature sets a very large number of mappings must be stored to provide the maximum speedup. This is especially true if no efforts are made to avoid redundancy in the database. Performing a linear search through all of the entries in the database may be prohibitively expensive, and so some kind of indexing strategy must be employed to improve lookup speed.

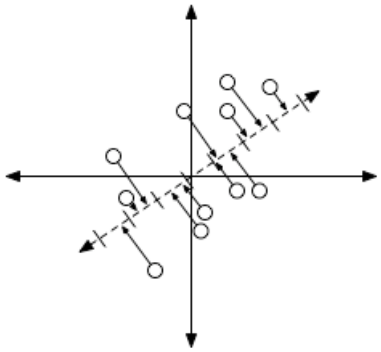

Figure 5. Points in a 2-D space being projected onto an arbitrary axis (dashed line). The tick marks represent the boundaries of hash bins into which the points fall. Note that points near each other are likely to fall into the same bin.

We use a very basic version of locality-sensitive hashing to perform fast approximate lookups. This technique involves projecting each feature vector in the database onto an arbitrary unit vector by computing the inner product of thtwo vectors. The resulting number is quantized and used as a hash function. As each point is added, a reference to it is maintained in a hash table indexed by this hash function. To find the nearest neighbor to a new point, we determine what hash bin it would map to and check the points associated with that bin. In general, points that map to the same bin will be closer to each other than other points. Although this method does not guarantee that we will find the single closest point to the query point in the database, in practice it works fairly well, is simple to implement, eliminates the need to check large portions of the database, and is quite fast. Its robustness can also be improved by checking multiple hashes utilizing different projection functions.

4.3 Direct parameter control

In some cases the mapping from a synthesizer parameter to perception is so clear that no intervening layer is necessary to translate from intention to perception. An example might be the frequency of an oscillator's relationship with pitch. To avoid the

hassle and expense of controlling such factors indirectly using feature-based synthesis, we provide a method for directly controlling parameters by exempting them from the optimization process.

5. EXAMPLE PERFORMANCE SYSTEM

It may be instructive to go over a complex example of a live performance system that can be built using feature-based synthesis. We first describe the synthesizer used by the system, then the features used to control that synthesizer, and finally the way in which the user interacts with the system.

5.1 Synthesizer

The synthesizer generates sound by summing the weighted outputs of a set of four sine oscillators and a white noise generator filtered by a resonant bandpass filter. In total, the number of parameters needed to control the synthesizer is 11: four for the frequencies of the sine oscillators, two for the resonance and center frequency of the filter, and five for the relative gains of the four oscillators and the filtered noise. The rms power of the system's summed output is kept constant.

5.2 Features

The system uses the spectral centroid (a measure of the brightness of the sound), the harmonicity (a measure of how strongly pitched the sound is), and the first five mel-frequency cepstral coefficients (a.k.a. MFCCs, a measure of the coarse shape of the spectrum commonly used in speech recognition) to control the synthesizer. The implementations come from the MARSYAS framework [10].

5.3 Control

The first two features, centroid and harmonicity, can be controlled simply enough by using a pair of sliders, sensors, or even a mouse's x and y coordinates. The MFCCs can be controlled by extracting the first five MFCC values from the performer's voice in real time and passing the results as control values to the system. The result is that the amplitudes and frequencies of the sine tones and filtered noise change to match the spectral shape and brightness specified, and the relative amplitudes of oscillator and noise adjust to produce the desired harmonicity. Although the performer cannot control every aspect of the sound, the result is that he or she is able to exert control over a large number of parameters in a perceptually meaningful way very quickly.

6. FUTURE WORK

Much can still be done to improve the efficiency and usefulness of our framework. More feature evaluators and synthesis algorithms need to be incorporated and implemented. Further improvements to the efficiency of our parameter optimization and database algorithms are also in the works.

One important architectural extension that would make our system more flexible would permit meta-features encapsulating how features change over time. Another would be to incorporate a more sophisticated notion of state into the parametric synthesizer architecture, permitting more interesting synthesizers such as physical models. We are in the process of deciding on a way to do so without violating the assumption that a one-to-many mapping between synthesizer parameters is impossible (and therefore complicating offline computation of such mappings).

7. REFERENCES

[1] Bencina, R. The Metasurface – Applying Natural Neighbor Interpolation to Two-to-Many Mapping. In *Proceedings of the Conference on New Interfaces for Musical Expression (NIME05)* (Vancouver, BC, Canada).

[2] Hoffman, M., Cook, P. R.. Feature-Based Synthesis: Mapping Acoustic and Perceptual Features onto Synthesis Parameters. In *Proceedings of the International Computer Music Conference (ICMC '06)* (New Orleans, LA, USA, 2006).

[3] Horner, A., Beauchamp, J., and Haken, L. Machine Tongues XVI: Genetic Algorithms and Their Application to FM Matching Synthesis. *Computer Music Journal* 17(3) (1993), 17-29.

[4] Hunt, A. and Kirk, R. Mapping Strategies for Musical Performance – Trends in Gestural Control of Music. In *Trends in Gestural Control of Music*, M. Wanderley and M. Battier, eds. Paris, France Institut de Recherche et Coordination Acoustique Musique—Centre Pompidou, 2000, pp. 231–258.

[5] Janer, J. Voice-Controlled Plucked Bass Guitar Through Two Synthesis Techniques. In *Proceedings of the Conference on New Interfaces for Musical Expression (NIME05)* (Vancouver, BC, Canada).

[6] Johnson, C. and Gounaropoulos, A. Timbre Interfaces Using Adjectives and Adverbs. In *Proceedings of the Conference on New Interfaces for Musical Expression (NIME06)* (Paris, France, 2006).

[7] Lee, M., and Wessel, D. Connectionist Models for Real-Time Control of Synthesis and Compositional Algorithms. In *Proceedings of the International Computer Music Conference (ICMC '92)* (San Jose, CA, USA, 1992).

[8] Loscos, A. and Aussenac, T. The Wahwactor: a Voice Controlled Wah-Wah Pedal. In *Proceedings of the Conference on New Interfaces for Musical Expression (NIME05)* (Vancouver, BC, Canada).

[9] Schwarz, D. Current Research in Concatenative Sound Synthesis. In *Proceedings of the International Computer Music Conference (ICMC '05)* (Barcelona, Spain, 2005).

[10] Tzanetakis, G. and Cook, P. MARSYAS: a Framework for Audio Analysis. *Organized Sound* (1999), 4: 169-175.

[11] Verfaille, V. and Arfib, D. A-DAFX: Adaptive Digital Audio Effects. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFx-01)* (Limerick, Ireland, December 6-8, 2001).

[12] Verfaille, V., Boissinot, J., Depalle, Ph., and Wanderley, M.M. *SSynth*: a Real Time Additive Synthesizer with Flexible Control. In *Proceedings of the International Computer Music Conference (ICMC '06)* (New Orleans, LA, USA, 2006).

[13] Wanderley, M., Schnell, N., and Rovan, J.B. Escher – modeling and performing composed instruments in real-time. In *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics (SMC'98),* (San Diego, USA, 1998) pp. 1080-4.

[14] Wun, S., Horner, A., and Ayers, L. A Comparison Between Local Search and Genetic Algorithm Methods for Wavetable Matching. In *Proceedings of the International Computer Music Conference (ICMC '04)* (Miami, FL, USA, 2004