Data-driven Modeling and Synthesis of Acoustical Instruments

Bernd Schoner, Chuck Cooper, Chris Douglas, Neil Gershenfeld
Physics and Media Group
MIT Media Laboratory
20 Ames Street
Cambridge, MA 02139
schoner@media.mit.edu

Abstract

We present a framework for the analysis and synthesis of acoustical instruments based on data-driven probabilistic inference modeling. Audio time series and boundary conditions of a played instrument are recorded and the non-linear mapping from the control data into the audio space is inferred using the general inference framework of Cluster-Weighted Modeling. The resulting model is used for real-time synthesis of audio sequences from new input data.

1 Introduction

Most of today's musical synthesis is based on either sampling acoustical instruments [Massie, 1998] or detailed first-principles physical modeling [Smith, 1992]. The sampling approach typically results in high sound quality, but has no notion of the instrument as a dynamic system with variable control. The physical modeling approach retains this dynamic control but results in intractably large models when all the physical degrees of freedom are considered. The search for the right combination of model parameters is difficult and there is no systematic way to ascertain and incorporate subtle differences between instruments of the same class, such as two master violins.

We present a new synthesis method, inferring the physical behavior of the instrument from observation of its global performance, that is conceptually intermediate between the traditional techniques. Dynamical systems theory shows that we can reconstruct a state space of a physical system that is homeomorphic to the actual state space using the input and output observables of the system along with their time lags (embedding space) [Takens, 1981, Casdagli, 1992]. The reconstructed space captures the dynamics of the entire system and its dimensionality may be chosen in such a way that it corresponds to the number of effective rather than actual degrees of freedom of the system. In the case of dissipative systems, such as musical instruments, this effective dimensionality can be considerably lower than the physical one. We combine this result with adequate signal processing and sensing technology to build a modeling and synthesis framework that introduces new capabilities and control flexibility into accepted and mature synthesis techniques.

Using the violin as our test instrument, we developed unobtrusive sensors that track the position of the bow relative to the violin, the pressure of the forefinger on the bow, and the position of the finger on the fingerboard. In a training session, we record control input data from these sensors along with the violin's audio output. These signals serve as training data for the inference engine, Cluster-Weighted Modeling, which learns the non-linear relationship between the control inputs and the target audio output. We have developed Cluster-Weighted Modeling (CWM) as a general probabilistic inference engine that

naturally extends beyond linear inference and signal processing practice into a powerful non-linear framework that handles non-Gaussianity, non-stationarity, and discontinuity in the data. CWM retains the functionality of conventional neural networks, while addressing many of their limitations. Once it has converged the model can predict audio data based on new control data. A violinist plays the interface device (which could be a silent violin), and the sensors now drive the computer model to produce the sound of the original violin.

2 Data Collection and Hardware

Several sensors capture the gestural input of the violin player. We measure the violin bow position with a capacitive coupling technique that detects displacement current in a resistive antenna [Paradiso and Gershenfeld, 1997] and infer a bow velocity estimate by differentiation and Kalman filtering. We determine the distance between bow and bridge of the violin by the same technique. Bow pressure is inferred from the force exerted by the player's finger on a force-sensitive resistor mounted on the bow. A microphone placed near the violin detects the acoustic output.

The finger-position sensor consists of a thin strip of stainless-steel ribbon attached to the finger board. An alternating current at 5 kHz passes through the violin string and is divided according to the distance between the contact point and the two ends of the ribbon. Synchronous detection measures the difference between the two currents, from which we infer the position of the player's finger. We use the sum of the currents to determine whether the string is in contact with the finger board.

During data recording sessions, we simultaneously record sensor and audio data along with an initial synchronization impulse to ensure proper time alignment of the signals (see Fig.1). For performance, the violinist plays the same instrument with the strings covered by a shield that prevents bow-string contact. The sensor signals are again fed into the computer, and now a real-time program predicts sound parameters and synthesizes audio output corresponding to the player's performance data.

3 Modeling Sequence

The model building and prediction sequence is broken into the following steps: (1) transform the output data representation from time series audio samples to spectral data, (2) prepare the effective state space from the input and output data series and reduce the dimensionality of this space using a principal component analysis, (3) build an input-output prediction model using the cluster-weighted algorithm, (4) use this model to predict output spectral data based on new input data, and (5) synthesize the audio stream from the predicted spectral sequence.

3.1 Data Analysis and Representation

Our first attempts at using embedding synthesis to model driven input/output systems were entirely time domain-based. While this approach is close to the techniques suggested by dynamic systems theory [Casdagli, 1992], it suffered from instability due to the difference in characteristic time scales between control inputs and signal outputs. The time-domain approach models a particular realization of a process, not the process itself, and it retains perceptually irrelevant features such as phase information or unpredictable information such as noise. The spectral representation we now use overcomes these problems [McAulay and Quatieri, 1985, Serra and Smith, 1990].

We decompose the measured audio signal into spectral frames in such a way that each audio frame corresponds to a measured set of input variables $\mathbf{x}'$. At each frame time step we compute the discrete

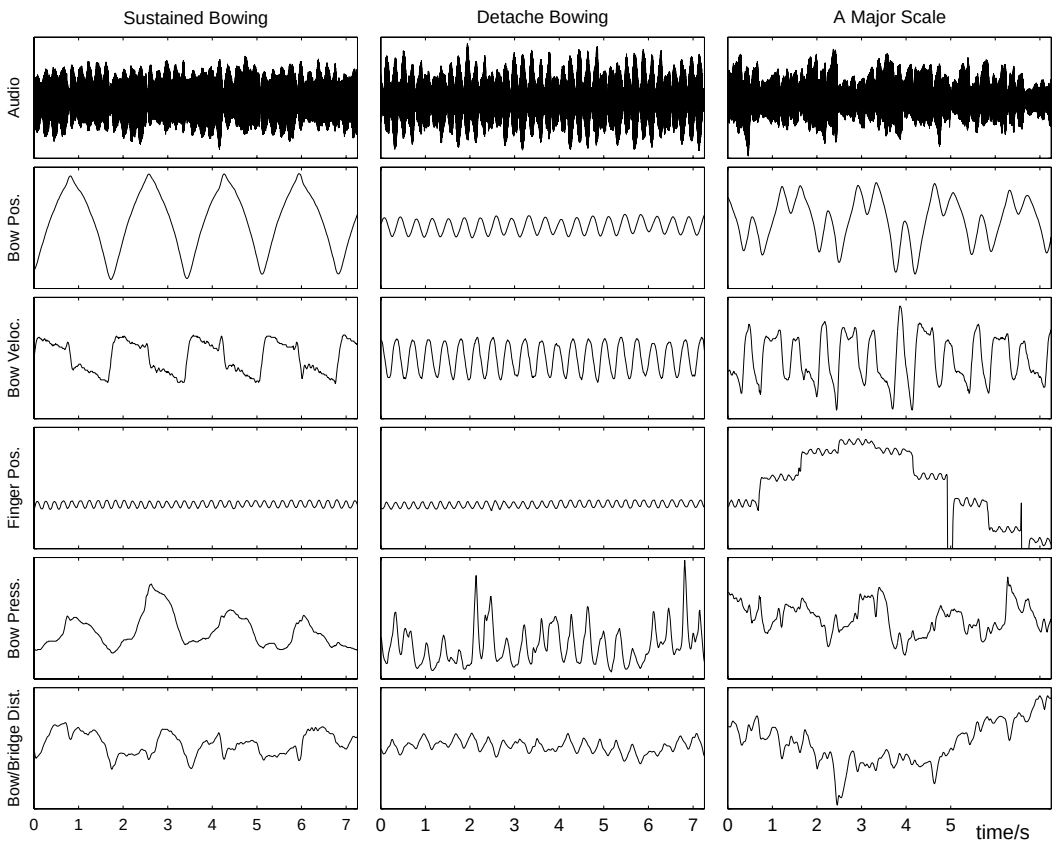

Figure 1: Audio and input sensor data for various bowings/notes. Left column: E-natural, sustained bowing with strong vibrato. Middle column: E-natural, détaché bowing. Right column: A-major scale.

Fourier transform of a short interval of audio samples weighted by a Hamming window. From those transforms we retain only the information corresponding to actual partials of the sound. While the amplitude of each partial is taken to be proportional to the magnitude of the DFT, we obtain a precise estimate of the partial frequency from the phase difference of DFTs applied to two windows shifted by a single sample [McAulay and Quatieri, 1985, McAulay and Quatieri, 1986]. After analysis, the training data is reduced to the set $\{\mathbf{y}_n, \mathbf{x'}_n\}_{n=1}^{N}$, where the index n refers to consecutive frames, $\mathbf{x'}$ refers to the vector of inputs consisting of bow velocity, pressure, finger position, and bow bridge position, and $\mathbf{y}$ refers to the vector of partials with each partial represented by a pair describing frequency and amplitude.

The modeling task starts with evaluating the length of the instrument's "memory" and finding the input signals which best represent the influence of the past on the current output. To this end we add time lagged components of $\mathbf{x'}$ to the input vector. In particular, we attempt to augment the input data set such that the input/output mapping becomes a single-valued function. We examined two approaches to this feature selection that resulted in comparable performance. In the first we carefully select among the available inputs and their time lags and evaluate different combinations using out-of-sample synthesis. In the second we choose a large number of inputs which we reduce using a principal component analysis (PCA)[Gershenfeld, 1998]. By keeping only the strongest PCA components we construct a reasonable size feature vector $\mathbf{x}$ that can be used to predict $\mathbf{y}$.

The pair $\{\mathbf{x}, \mathbf{y}\}$ is used to train the inference algorithm as described in the next section. The converged

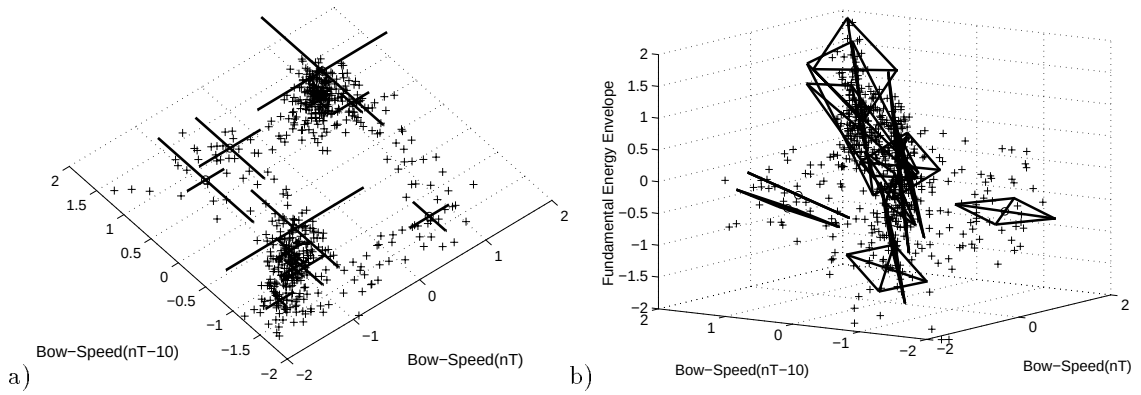

Figure 2: Data and clusters in the joint input/output space. *a:)* Vertical view of the input space. Clusters are represented by their centers and their domains of influence (variances). *b:)* Clusters in 3D with the rectangles representing the plane of the linear functions.

model serves as a predictor that generates an estimated output $\hat{\mathbf{y}}$ given a new input vector $\mathbf{x}$. From the estimated spectral vector $\hat{\mathbf{y}}$ we reconstruct a time domain waveform by linearly interpolating frequencies and amplitudes of the independent partials between frames. The sinusoidal components corresponding to the partials are summed into a single audio signal.

3.2 Cluster-Weighted Modeling

Cluster-Weighted Modeling (CWM) is an input/output inference framework based on probability density estimation of a joint set of feature (control) and target (spectral/audio) data. It is derived from hierarchical mixture-of-experts type architectures [Jordan and Jacobs, 1994] and can be interpreted as a flexible and transparent technique for approximating an arbitrary function. Clusters automatically "go to where the data is" and approximate subsets of the data space according to a smooth domain of influence (Fig.2). Globally, the influence of the different clusters is weighted by Gaussian basis terms, while locally, each cluster represents a simple model such as a linear regression function. Thus, previous results from linear systems theory, linear time series analysis and traditional musical synthesis are applied within the broader context of a globally non-linear model.

After preprocessing the experimental measurements we obtain a set of training data $\{\mathbf{y}_n, \mathbf{x}_n\}_{n=1}^N$, where $\mathbf{x}$ refers to the feature input vector at a time n and $\mathbf{y}$ refers to the target vector containing spectral information at time n. We infer the joint probability density of feature and target vector $p(\mathbf{y}, \mathbf{x})$, which lets us derive conditional quantities such as the expected value of $\mathbf{y}$ given $\mathbf{x}$, $\langle \mathbf{y}|\mathbf{x} \rangle$, and the expected covariance matrix of $\mathbf{y}$ given $\mathbf{x}$, $\langle \mathbf{C}_{yy}|\mathbf{x} \rangle$. The value $\langle \mathbf{y}|\mathbf{x} \rangle$ serves as prediction of the target value $\mathbf{y}$ and $\langle \mathbf{C}_{yy}|\mathbf{x} \rangle$ serves as its error estimate [Gershenfeld et al., 1997].

The joint density $p(\mathbf{x}, \mathbf{y})$ is expanded in clusters labeled c_m, each of which contains an input domain of influence, a local model, and an output distribution:

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}) &= \sum_{m=1}^{M} p(\mathbf{y}, \mathbf{x}, c_m) \\ &= \sum_{m=1}^{M} p(\mathbf{y}|\mathbf{x}, c_m)\ p(\mathbf{x}|c_m)\ p(c_m) \quad . \end{aligned} \tag{1}$$

The probability functions $p(\mathbf{y}|\mathbf{x}, c_m)$ and $p(\mathbf{x}|c_m)$ are taken to be of Gaussian form so that $p(\mathbf{x}|c_m) = \mathcal{N}(\mu_m, \mathbf{C}_m)$ and $p(\mathbf{y}|\mathbf{x}, c_m) = \mathcal{N}(\mathbf{f}(\mathbf{x}, \beta_m), \mathbf{C}_{y,m})$, where $\mathcal{N}(\mu, \mathbf{C})$ stands for the multi-dimensional

Gaussian distribution with mean vector μ and covariance matrix $\mathbf{C}$. The function $\mathbf{f}(\mathbf{x}, \beta_m)$ with unknown parameters β_m in general needs to be a linear coefficient model [Gershenfeld, 1998], for example a polynomial model.

The complexity of the local model is traded off against the complexity of the global architecture. In the case of polynomial expansion, there are two extreme forms that illustrate this trade-off. We may use locally constant models in connection with a large number of clusters, in which case the predictive power comes from the number of Gaussian kernels only. Alternatively we may decide to use a high-order polynomial model and a single kernel, in which case the model reduces to a global polynomial model. In this particular application we choose the linear model

$$f(\mathbf{x}, \beta_m) = \beta_{0,m} + \sum_{d=1}^{D} \beta_{d,m} x_d \quad . \tag{2}$$

where d refers to an input dimension and D to the total number of input dimensions.

Given the density estimate, we infer a conditional forecast

$$\langle \mathbf{y}|\mathbf{x} \rangle = \frac{\sum_{m=1}^{M} \mathbf{f}(\mathbf{x}, \beta_m)\ p(\mathbf{x}|c_m)\ p(c_m)}{\sum_{m=1}^{M} p(\mathbf{x}|c_m)\ p(c_m)} \quad . \tag{3}$$

as well as a conditional error forecast,

$$\langle \mathbf{C}_{yy}|\mathbf{x} \rangle = \frac{\sum_{m=1}^{M} [\mathbf{C}_{y,m} + \mathbf{f}(\mathbf{x}, \beta_m) \cdot \mathbf{f}(\mathbf{x}, \beta_m)^T]\ p(\mathbf{x}|c_m)\ p(c_m)}{\sum_{m=1}^{M} p(\mathbf{x}|c_m)\ p(c_m)} - \langle \mathbf{y}|\mathbf{x} \rangle^2 \quad . \tag{4}$$

The choice of the number of clusters M controls under- versus over-fitting. The model should have enough clusters to model the predictable data, but should not become so complex that it tries to predict the noise and non-generalizable features. The optimal M can be determined by cross validation with respect to an abstract error measure such as the square error or with respect to perceptual performance.

We find the model parameters using a variant of the Expectation-Maximization (EM) algorithm : Conventional EM updates are used to estimate the unconditioned cluster probabilities $p(c_m)$, cluster locations μ_m, and covariances $\mathbf{C}_m$. We then use pseudo-inverses of the cluster weighted covariance matrices to update the local model parameters β_m. The EM algorithm finds the most likely cluster parameters by iterating between an expectation step and a maximization step [Dempster et al., 1977, Jordan and Jacobs, 1994].

E-step: Given a starting set of parameters, we find the probability of a cluster given the data:

$$\begin{aligned} p(c_m|\mathbf{y}, \mathbf{x}) &= \frac{p(\mathbf{y}, \mathbf{x}|c_m)\ p(c_m)}{p(y, \mathbf{x})} \\ &= \frac{p(\mathbf{y}, \mathbf{x}|c_m)\ p(c_m)}{\sum_{l=1}^{M} p(\mathbf{y}, \mathbf{x}|c_l)\ p(c_l)} \quad , \end{aligned} \tag{5}$$

where the sum over clusters in the denominator lets clusters interact and specialize in data they best explain.

M-step: Now we assume the current data distribution correct and maximize the likelihood function by recomputing the cluster parameters. The new estimate for the unconditioned cluster probabilities becomes:

$$\begin{aligned} p(c_m) &= \int p(c_m|\mathbf{y}, \mathbf{x})\ p(\mathbf{y}, \mathbf{x})\ d\mathbf{y}\ d\mathbf{x} \\ &\approx \frac{1}{N} \sum_{n=1}^{N} p(c_m|\mathbf{y}_n, \mathbf{x}_n) \quad . \end{aligned} \tag{6}$$

The cluster-weighted expectation of any function $\theta(\mathbf{x})$ is defined as

$$\begin{aligned} \langle \theta(\mathbf{x}) \rangle_m &\equiv \int \theta(\mathbf{x})\ p(\mathbf{x}|c_m)\ d\mathbf{x} \\ &\approx \frac{\sum_{n=1}^{N} \theta(\mathbf{x}_n)\ p(c_m|\mathbf{y}_n, \mathbf{x}_n)}{\sum_{n=1}^{N} p(c_m|\mathbf{y}_n, \mathbf{x}_n)} \quad . \end{aligned} \tag{7}$$

This lets us update the cluster means and the cluster weighted covariance matrices :

$$\begin{aligned} \mu_m &= \langle \mathbf{x} \rangle_m \\ [\mathbf{C}_m]_{ij} &= \langle (x_i - \mu_i)(x_j - \mu_j) \rangle_m \quad . \end{aligned} \tag{8}$$

The derivation of the maximum likelihood solution for the model parameters yields

$$\beta_m = \mathbf{B}_m^{-1} \cdot \mathbf{A}_m \quad , \tag{9}$$

with $[\mathbf{B}_m]_{ij} = \langle f_i(\mathbf{x}, \beta_m) \cdot f_j(\mathbf{x}, \beta_m) \rangle_m$ and $[\mathbf{A}_m]_{ij} = \langle y_i \cdot f_j(\mathbf{x}, \beta_m) \rangle_m$.
Finally the output covariance matrices associated with each model are estimated,

$$\mathbf{C}_{y,m} = \langle [\mathbf{y} - \mathbf{f}(\mathbf{x}, \beta_m)] \cdot [\mathbf{y} - \mathbf{f}(\mathbf{x}, \beta_m)]^T \rangle_m \quad . \tag{10}$$

We iterate between the E- and the M-step until the overall likelihood of the data, as defined by the product of all data likelihoods (equ.1) does not increase further.

4 Experimental results

We collected approximately 30 minutes of input/output violin data, both single notes and scales with various bow strokes (Fig.1). We then built models based on subsets of this data, and used them for off-line and on-line synthesis.

The model performs very well on small subsets of the overall training data. It is particularly robust at representing pitch and amplitude fluctuations caused by vibrato. Figure 3 illustrates that CWM can reproduce the spectral characteristics of a segment of violin sound. The sound examples demonstrate the results of audio resynthesis both in- and out-of-sample (For training only the first half of each original sound file was used). We were able to build a real-time system running on Windows NT using a Pentium II 300 MHz. The system responds well to control changes, but considerable latency is caused by operating system, data acquisition board and sound card.

Large scale models that cover the entire musical space are not yet satisfying. Note transitions are particularly difficult to model, because there is very little data representing them. While the inference approach interpolates well between observed control data, it is not designed to extrapolate into unseen control domains. Therefore control input outside the range covered by the training data results in unpredictable output during performance.

5 Conclusions and Future Work

We have shown how the general inference framework CWM can be used in a sensing and signal-processing system for musical synthesis. The approach is particularly appropriate for musical instruments that rely on continuous human control, such as the violin, since CWM naturally relates input and output time series.

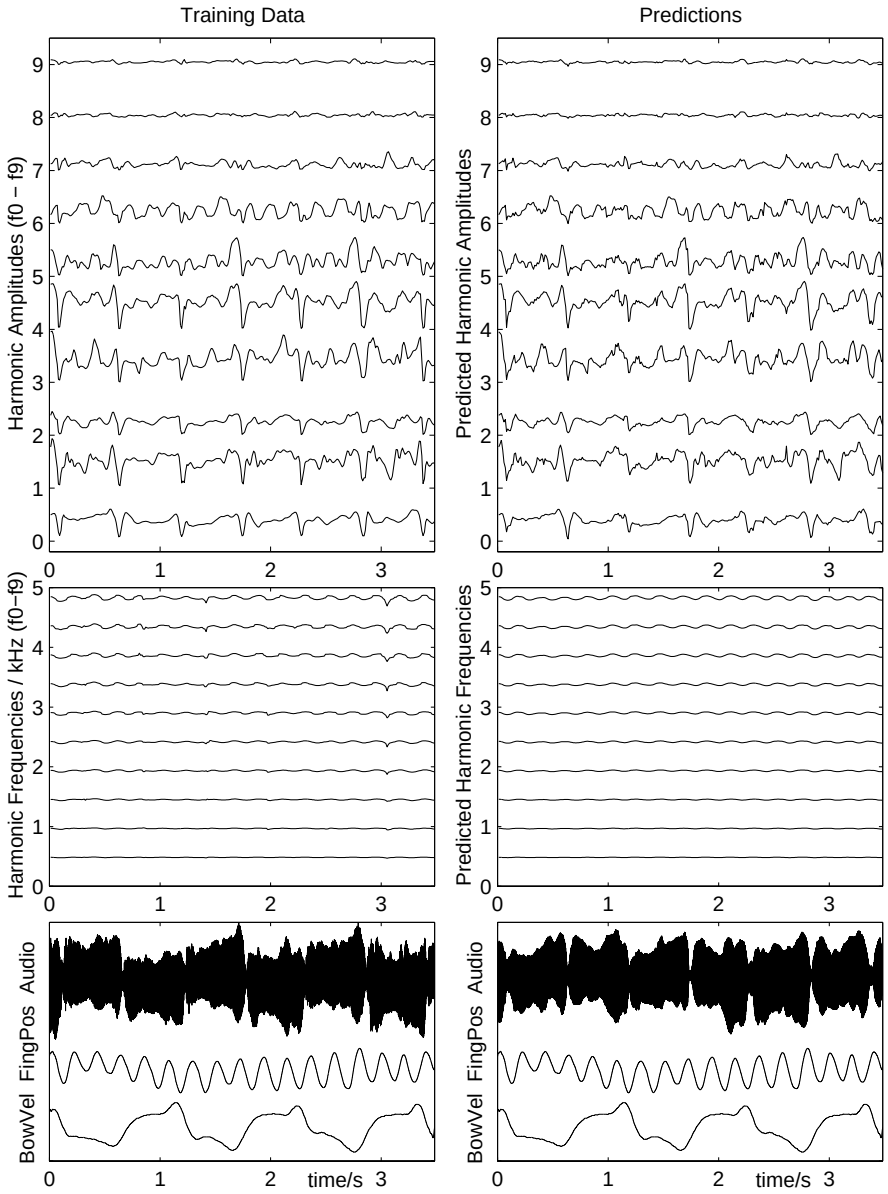

Figure 3: *Left:* measurements on a violin, showing the bow position, player's finger position, resulting audio time series and harmonic structure. *Right:* The audio re-synthesized from the sensor data by a model trained in the joint input-output space.

CWM overcomes many of the limitations of conventional inference techniques, yet it can be only as good as the audio representation. The current spectral representation does not provide model flexibility, makes strong assumptions about the nature of the physical device, and misses certain elements of natural sound. Future work will therefore focus on improving the combination of representation and inference by embedding local filter and sample architectures and explicit time constraints into the general CWM framework.

Historically the literature of computer synthesis has been split between physical modeling and sound sampling. In this work we have shown how an intermediate approach combines features of the traditional

synthesis techniques to generate a model that provides both control flexibility and high fidelity to the original. More work will be needed to progress to a high quality digital instrument but these preliminary results indicate the promise of "physics sampling".

6 Acknowledgments

The authors would like to thank Romy Shioda, Sandy Choi and Teresa Marrin for helping in the data collection process, Edward Boyden for writing parts of the code used in this work, and Joe Paradiso for the design of the violin bow. This work was made possible by the Media Lab's Things That Think consortium.

References

[Casdagli, 1992] Casdagli, M. (1992). A dynamical systems approach to modeling input-output systems. In Casdagli, M. and Eubank, S., editors, *Nonlinear Modeling and Forecasting*, Santa Fe Institute Studies in the Sciences of Complexity, pages 265–281, Redwood City. Addison-Wesley.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *J. R. Statist. Soc. B*, 39:1–38.

[Gershenfeld, 1998] Gershenfeld, N. (1998). *The Nature of Mathematical Modeling*. Cambridge University Press, New York.

[Gershenfeld et al., 1997] Gershenfeld, N., Schoner, B., and Metois, E. (1997). Cluster-weighted modeling for time series prediction and characterization. submitted.

[Jordan and Jacobs, 1994] Jordan, M. and Jacobs, R. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214.

[Massie, 1998] Massie, D. C. (1998). Wavetable sampling synthesis. In Kahrs, M. and Brandenburg, K., editors, *Applications of Digital Signal Processing to Audio and Acoustics*, pages 311–341. Kluwer Academic Publishers.

[McAulay and Quatieri, 1985] McAulay, R. and Quatieri, T. (1985). Speech analysis/synthesis based on a sinusoidal representation. Technical Report 693, Massachusetts Institute of Technology / Lincoln Laboratory, Cambridge, MA.

[McAulay and Quatieri, 1986] McAulay, R. and Quatieri, T. (1986). Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34 No.4:744–754.

[Paradiso and Gershenfeld, 1997] Paradiso, J. A. and Gershenfeld, N. (1997). Musical applications of electric field sensing. *Computer Music Journal*, 21(2):69–89.

[Serra and Smith, 1990] Serra, X. and Smith, J. O. (1990). Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24.

[Smith, 1992] Smith, J. O. (1992). Physical modeling using digital waveguides. *Computer Music Journal*, 6(4).

[Takens, 1981] Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, D. and Young, L., editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381, New York. Springer-Verlag.