Audio Engineering Society
Convention Paper
Presented at the 110th Convention
2001 May 12–15 Amsterdam, The Netherlands

An Audio-Driven, Spectral Analysis-Based, Perceptual Synthesis Engine

Tristan Jehan and Bernd Schoner*

MIT Media Laboratory

20 Ames Street, Cambridge, MA 02139, USA

ABSTRACT

A real-time synthesis engine is presented which models and predicts the *timbre* of acoustic instruments based on perceptual features. The paper describes the modeling sequence including the analysis of natural sounds, the inference step that finds the mapping between control and output parameters, the timbre prediction step, and the sound synthesis. Demonstrations include the timbre synthesis of stringed instruments and the singing voice, as well as the cross-synthesis and timbre morphing between these instruments.

INTRODUCTION

Timbre is defined as the quality of a sound that distinguishes it from other sounds of the same pitch and loudness. This paper introduces a new technique for modeling and controlling pitched timbre. We present an expressive sound synthesis engine that is driven by continuously changing perceptual parameters, namely pitch, loudness, and brightness. These perceptual parameters are extracted from the audio stream of an acoustic or electric monophonic instrument and are used as controlling features for the prediction of spectral data.

In the offline analysis step, we extract perceptual features and a sinusoidal representation from the audio data. In the modeling step, we infer a mapping between perceptual features and spectral representation. In the synthesis step, the model is used to predict a harmonic structure of different timbre from new control parameters. The new control input is either extracted offline from a sound database or generated in real time on a muted instrument. The inference and prediction system was implemented using Cluster-Weighted Modeling (CWM), a probabilistic toolkit for the analysis and prediction of nonlinear time series [13, 3].

Unlike many other synthesis and modeling techniques, e.g. physical modeling, our approach extracts essential perceptual characteristics directly from the audio signal of a real acoustic instrument, for example a Stradivarius violin. Therefore this approach allows us to model arbitrary instruments based on recordings of the instrument without redesigning our model. Furthermore, we can easily exchange control and audio signals

*Current address: ThingMagic LLC, Cambridge, MA 02141.

of different instruments since our perceptual representation is preserved across instruments and even instrument families. Pitch, loudness, and brightness are considered perceptually meaningful and are used as feature vectors for the predictor. The underlying assumption of our approach is that any two frames of sound have similar spectral properties if pitch, loudness, and brightness are similar.

Additive synthesis is commonly recognized as a powerful technique for sound description and synthesis. Unfortunately it is rather difficult to control, and only very few commercial synthesizers exploit the potential of this technique. While keyboards enable many synthesis applications, non-discretely pitched musical controllers such as saxophone, trombone, violin, guitar, or voice, are typically not used for controlling synthesis algorithms. This is mostly due to the fact that musical gestures like finger position, blown air, or bow pressure are difficult to measure and to interpret musically. Our approach uses additive synthesis for sound production and audio based perceptual features as controls. It therefore overcomes the mentioned limitations and is applicable to any acoustic pitched instruments.

Our modeling approach enables several applications, including the cross-synthesis of musical instruments. In this particular application, the perceptual control channel of an instrument is used to generate the sound of a different instrument. A second powerful application morphs between sounds of different instruments. The software environment has been entirely implemented in Max/MSP. The library of novel Max objects includes objects that extract perceptual parameters as well as inference and prediction objects using CWM.

PRIOR WORK

Ever since the invention of neural networks, there have been research efforts to model the complexity of musical signals and of human musical action by means of artificial neural networks (ANNs). Connectionist tools have been applied to musical problems such as harmonizing a melody line and recognizing and classifying instrument families from sound. However, connectionist approaches to musical synthesis are not very common.

Métois introduces the synthesis technique *Psymbesis*, for Pitch Synchronous Embedding Synthesis [6]. He defines a vector of perceptual control parameters including pitch, loudness, and brightness. He clusters this data in a control space and assigns periods of sound to each cluster. Each cluster period is resampled with respect to a reference pitch and is characterized by the statistical mean and variance of each sample. For synthesis, the chosen period is represented in a low-dimensional lag-space rotating around a closed curve. Depending on the sample variance of the output, samples are slowly pulled back to the mean values ensuring that the transition between different sampled periods happens smoothly. The periods are re-sampled at the desired pitch and adjusted for the desired loudness.

Wessel et al. presented a synthesis model which inspired our approach [15]. A database of recorded sounds is analyzed and parameterized with respect to pitch, loudness, and brightness and is decomposed into spectral frames consisting of frequencies and amplitudes. The perceptual parameters serve as inputs to the feed-forward network, whereas the spectral parameters serve as outputs. A network is trained to represent and predict a specific instrument. The framework is tested with an ANN using one hidden layer and independently with a memory-based network. It was found that the ANN model provides smoother output, while the memory-based models are more flexible – easier to modify and easier to use in a creative context [15].

Schoner et al. used Cluster-Weighted Modeling to predict a spectral sound representation given physical input to the instrument [13]. While the target data was similar to the data used in [15], the feature vector consisted of actual physical movements of the violin player. Special recording hardware was needed to create the set of training data and to replay the model. The model was successfully applied in the case of violin family instruments. Special violin/cello bows and fingerboards were built to track the player motion, and these input devices were used to synthesize sound from player action.

This paper combines the efficiency of Cluster-Weighted Modeling with spectral synthesis and the idea of a perceptual control as feature vector.

TIMBRE ANALYSIS AND MODELING

Underlying our approach to timbre modeling are two fundamental assumptions:

1. We assume that the timbre of a musical signal is characterized by the instantaneous power spectrum of its sound output.
2. We assume that any given monophonic sound is fully described by the perceptual parameters pitch, loudness, and brightness and by the timbre of the instrument.

Based on these assumptions we conclude that a unique spectral representation of a sound can be inferred given perceptual sound data and a timbre model. In our approach we estimate both perceptual and spectral representations from recorded data and then predict the latter given the former.

A monophonic musical signal is represented in the spectral domain. The sound recording is analyzed frame by frame using a short-term Fourier transform (STFT) with overlapping frames of typically 20ms at intervals of 10ms. Longer windows (e.g. 2048-4096 points at 44.1KHz) and large zero-padded FFTs may be used as latency is not an issue here.

A spectral peak-picking algorithm combined with instantaneous frequency estimation (see next paragraph) tracks the partial peaks from one analysis frame to the next, resulting in L (= 30 to 40) sinusoidal functions. The number of stored harmonics L usually determines the sound quality and model complexity. Since pitch is considered an input to the system, not an output, the spectral vector contains $2L-1$ components ordered as $[A_0, M_1, A_1, M_2, A_2, \ldots, M_L, A_L]$ where A_i is the logarithmic magnitude of the i-th harmonic and M_i is a multiplier of the fundamental frequency F_0, i.e. pitch. F_0 relates to the frequency F_i of the i-th harmonic ($M_i = F_i/F_0$).

For pitch tracking we first perform a rough estimation using the Cepstrum transformation [7] or an autocorrelation method [11] and then operate on the harmonic peaks of the STFT. An N-point FFT discretizes the spectrum into $N/2$ useful bins of resolution F_s/N Hz, where F_s is the Nyquist frequency. We identify the peaks of the spectrum and identify the bins they fall into. The ambiguity associated with the extraction of a bin versus a peak frequency may be much bigger than a semitone, especially in the lower range of the spectrum. We therefore use the instantaneous frequencies of

the bins of highest energy to obtain a much higher resolution with little extra computation [6].

Given $X(k)$ the non-windowed discrete Fourier transform of the signal $s(n)$ for bin k is

$$X(k) = \sum_{n=0}^{n-1} s(n) e^{-jwnk} \quad (1)$$

with

$$w = \frac{2\pi}{N}$$

The estimate for bin k's instantaneous frequency is:

$$F_{\text{inst}}(k) = F_s \left(\frac{k}{N} + \frac{1}{2\pi} \text{Arg} \left[\frac{A}{B} \right] \right) \quad (2)$$

where

$$A = X(k) - \frac{1}{2} \left[X(k-1) + X(k+1) \right]$$

$$B = X(k) - \frac{1}{2} \left[e^{jw} X(k-1) + e^{-jw} X(k+1) \right]$$

Given the spectral decomposition we easily extract pitch as the frequency of the fundamental component. We furthermore extract instantaneous loudness from the total spectral energy. The power-spectrum bins are previously weighted by coefficients based on the Fletcher-Munson curves in order to simulate the ear frequency response. The output is in dB. The spectral centroid of the signal is used as an estimator for the brightness of the sound. In a second pass through the data, estimation errors are detected and eliminated. Frames are considered *bad* if no pitch could be detected or if it is outside a reasonable range, in which case the frame data is simply dropped. The peaks of the spectrum are used as an harmonic representation of the audio signal and as target data for our predictive model.

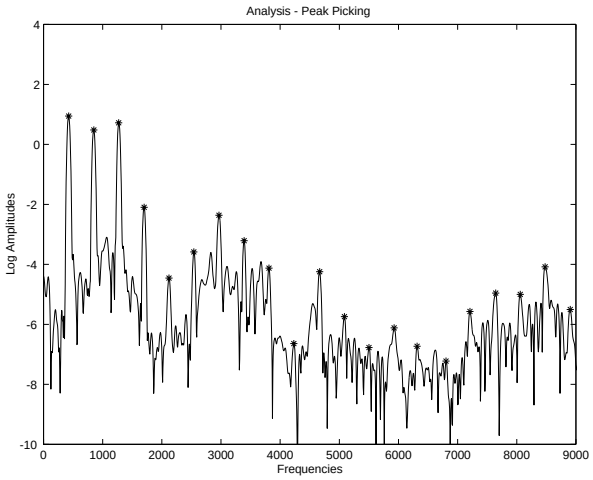

Fig. 2: Spectrum of a singing voice (23.2 ms frame of data). The stars indicate the harmonic peaks of the spectrum as found by the peak tracking algorithm.

We summarize: the data analysis step provides us with unordered vector-valued data points. Each data point consists of a three-dimensional input vector describing pitch, loudness, and brightness, and a 40 to 60-dimensional output vector containing frequency and amplitude values of 20 to 30 harmonic partials. We use this data to train a feed-forward input-output network to predict frequencies and amplitudes (see section *Cluster-Weighted Modeling*).

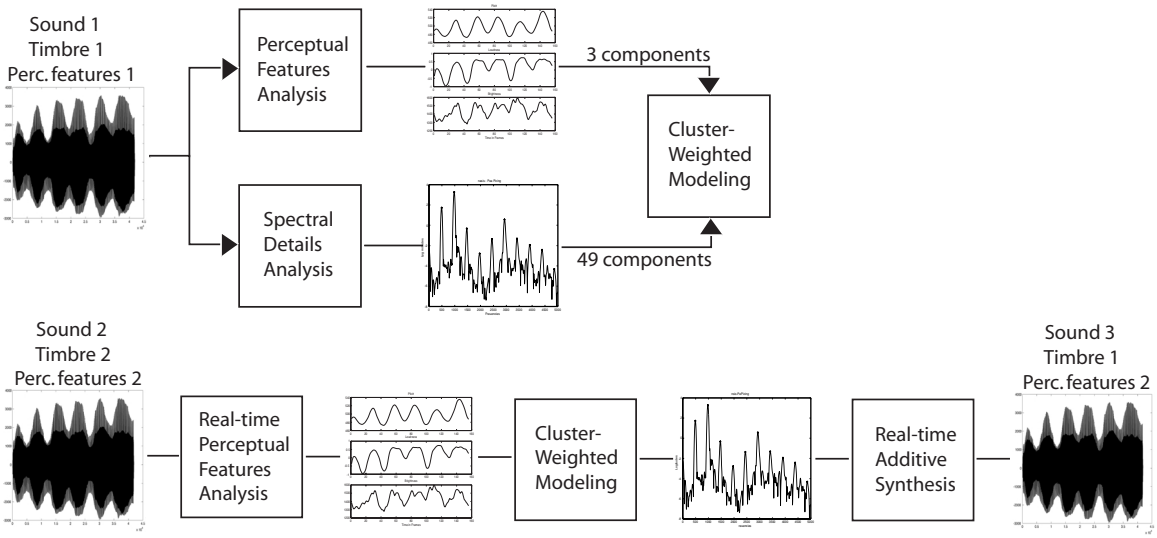

Fig. 1: Analysis + Modeling step (top) and Analysis + Prediction + Synthesis step (bottom).

TIMBRE PREDICTION AND SYNTHESIS

Timbre prediction and audio-driven synthesis are based on a new stream of audio input data. This time, the perceptual control features are extracted in real time from the audio stream. They are used as input to the nonlinear predictor function which outputs a vector of spectral data in real time (20 to 30 sinusoids depending on what level of sound quality is desired).

The specific model consists of three input parameters (pitch, loudness, and brightness), and L (= 40 to 60) output parameters. In the case of cross-synthesis, the perceptual control features are extracted and carefully rescaled to fall into a window of dynamic range, which is kept consistent across different instruments. This procedure does not apply to pitch but is important for the loudness and brightness parameters. The input vector is used with the predictor function on a frame by frame basis, generating an output vector at intervals of about 10ms. If our model is based on L sinusoidal parameters, the predictor generates $2L-1$ output values consisting of $[A_0, M_1, A_1, M_2, A_2, \ldots, M_L, A_L]$ where A_i is the logarithmic magnitude of the i-th harmonic and M_i is a multiplier of the fundamental frequency F_0.

The output vector is used with an additive synthesis engine that modulates sinusoidal components and superimposes them in the time domain, resulting in the deterministic component of the signal:

$$d(n) = \sum_{l=1}^{L} A_l \cos(\omega_l n + \Phi_l) \tag{3}$$

where n is a discrete time index and A_l and Φ_l are amplitude and phase of the partials l. This additive approach is computationally less efficient than an inverse FFT, but much simpler to implement.

In the next section, we show how a stochastic process will be combined with the deterministic component $d(n)$ of expression (3) to create a more accurate timbre (see *Noise Anlaysis/Synthesis*). The full signal model $s(n)$ then becomes:

$$s(n) = d(n) + r(n) \tag{4}$$

where $r(n)$ represents the residual noise component of the signal.

We observe that the timbre of any particular instrument or instrument family is contained in the predictor model (see Cluster-Weighted Modeling), whereas the musical intent is contained in the parameterization of the perceptual control data. By mixing control data from one instrument with the timbre model of a different instrument, the system allows a skilled player of a non-discretely pitched instrument (e.g. violin, trombone, or voice) to play the (previously modeled) timbre of any other pitched instrument without having to learn this new instrument or controller (see section *Applications*).

NOISE ANALYSIS/SYNTHESIS

The sound quality of additive synthesis can be improved significantly by synthesizing the residual nondeterministic components of the sound in addition to the deterministic harmonic components [14, 12]. The noise components are particularly important at the onsets of notes, which often is the most characteristic element of the timbre of a musical instrument. While the harmonic structure is usually described as a sum of sinusoidal functions, the residue (commonly called noise) can be modeled in several different ways [4]. Here we present a novel approach to noise modeling by means of a polynomial expansion.

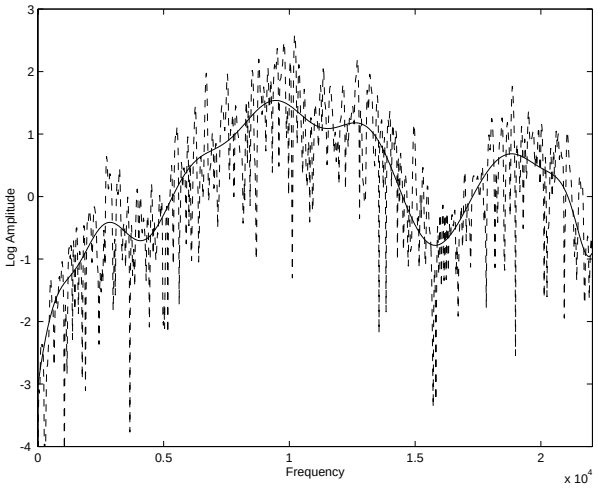

Fig. 3: Typical noise spectrum of the singing voice (23.2 ms FFT), approximated with a polynomial function (20 basis functions).

In general, the noise characteristics of a signal are captured in the shape of the power-spectrum of its non-harmonic components. We extract this residual spectrum for each time frame and approximate the spectral function (in a logarithmic scale) using polynomial basis functions. The residue is obtained by substracting the power-spectrum of the deterministic signal $d(n)$ from the power spectrum of the original signal $s(n)$ as described in expression (4). The approximation is of the form

$$f(x) = \sum_{k=0}^{K} a_k x^k \tag{5}$$

Since the spectrum is a one-dimensional function the number of basis terms and coefficients equals the order of the polynomial K plus one additional term for the constant component a_0. We use up to 30 basis functions and coefficients. The coefficients a_i form the output vector of a predictor model, which, in synthesis, interpolates between the coefficients of different noise spectra. The input vector of this second predictor consists of perceptual parameters and, in addition, a noise/signal ratio (noisiness) estimator and/or an indicator for note onsets.

During synthesis, the predictor model generates polynomial coefficients, which are used to reconstruct the noise spectrum for every frame. White noise is modulated with the reconstructed function in the spectral domain. The colored noise spectrum is retransformed into the time domain after scrambling the perceptually irrelevant phase information using an inverse FFT. The method accurately reproduces the noise properties of natural sound and it is particularly successful with breath noise that appears in the residue of instruments like the flute, saxophone, and trumpet. The noise predictor is currently being combined with the additive synthesis engine.

The accuracy of the noise model depends on the number of basis functions used and is easily scalable at synthesis.

This parameterization of sound is comparable to Serra's *Sinusoids Plus Noise* implementation [14]. However, our system removes the temporal axis to dynamically control musical features by generating new envelope functions in real time.

CLUSTER-WEIGHTED MODELING

We approximate the nonlinear mapping from the feature vector onto the harmonic target vector using the general inference framework, Cluster-Weighted Modeling. CWM is a probabilistic modeling algorithm that is based on density estimation around Gaussian kernels. Unlike Artificial Neural Networks, it is flexible and easy to understand and has a transparent network architecture.

Model Architecture

The descriptive power and algorithmic beauty of graphical probabilistic networks is widely appreciated in the machine-learning community. Unfortunately, the generality and flexibility of these networks are just about matched by their difficulty of use. Unless the architectures are constrained appropriately and are tailored for particular applications, they are of little use in a practical modeling situation. Gaussian mixture models, a subclass of graphical models, resolve some of these deficiencies. In this paper we use CWM, a Gaussian mixture architecture that combines model flexibility with fast model design and ease of use.

CWM is a framework for supervised learning based on probability density estimation of a joint set of input feature and output target data. It is similar to mixture-of-experts type architectures [5] and can be interpreted as a flexible and transparent technique to approximate an arbitrary function. However, its usage goes beyond the function fitting aspect, because the framework is designed to include local models that allow for the integrations of arbitrary modeling techniques within the global architecture. CWM describes local data features with simple polynomial models, but uses a fully nonlinear weighting mechanism to build overall powerful nonlinear models. Hence CWM combines the efficient estimation algorithm of generalized linear models with the expressive power of fully nonlinear network architecture.

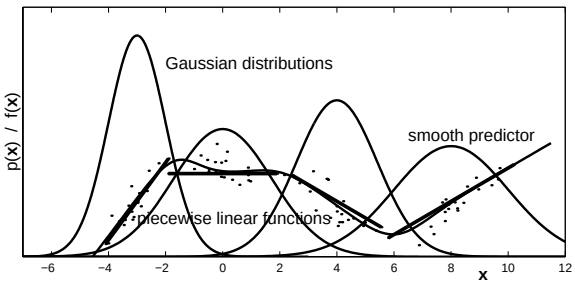

Fig. 4: One dimensional function approximation with locally linear models weighted by Gaussian kernels.

We typically start with a set of discrete or real-valued input features $\mathbf{x}$ and corresponding discrete or real-valued target vectors $\mathbf{y}$. $\mathbf{x}$ consists of measured sensor data, discrete classifiers, or processed features (this application). It is composed of independent observations or of time-delayed values of an embedded time series. $\mathbf{y}$ may be the scalar-valued sample of a time series, a classifying label, or independent target vector (this application). We consider the joint input-output set $\{\mathbf{y}_n, \mathbf{x}_n\}_{n=1}^N$, and it is our goal to infer the joint density $p(\mathbf{y}, \mathbf{x})$, which is the most general, compact, and statistically sufficient description of the data set.

$p(\mathbf{x}, \mathbf{y})$ is expanded in a sum over clusters c_k. Each cluster contains an input distribution, a local model, and an output distribution.

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}) &= \sum_{k=1}^{K} p(\mathbf{y}, \mathbf{x}, c_k) \\ &= \sum_{k=1}^{K} p(\mathbf{y}|\mathbf{x}, c_k)\ p(\mathbf{x}|c_k)\ p(c_k) \end{aligned} \tag{6}$$

The input distribution is parameterized as an unconditioned Gaussian and defines the domain of influence of a cluster.

$$p(\mathbf{x}|c_k) = \frac{|\mathbf{P}_k^{-1}|^{1/2}}{(2\pi)^{D/2}} e^{-(\mathbf{x}-\mathbf{m}_k)^T \cdot \mathbf{P}_k^{-1} \cdot (\mathbf{x}-\mathbf{m}_k)/2} \tag{7}$$

where $\mathbf{P}_k$ is the cluster-weighted covariance matrix in the feature space.

Given a continuous valued output vector $\mathbf{y}$, the output distribution is taken to be

$$p(\mathbf{y}|\mathbf{x}, c_k) = \frac{|\mathbf{P}_{k,y}^{-1}|^{1/2}}{(2\pi)^{D_y/2}} e^{-(\mathbf{y}-\mathbf{f}(\mathbf{x},\mathbf{a}_k))^T \cdot \mathbf{P}_{k,y}^{-1} \cdot (\mathbf{y}-\mathbf{f}(\mathbf{x},\mathbf{a}_k))/2} \tag{8}$$

where the mean value of the Gaussian distribution is replaced by the function $\mathbf{f}(\mathbf{x}, \mathbf{a}_k)$ with unknown parameters $\mathbf{a}_k$. In both (7) and (8) the off-diagonal terms of the covariance matrices can be dropped if needed.

Expression (8) is easily understood considering the conditional forecast of $\mathbf{y}$ given $\mathbf{x}$,

$$\begin{aligned} \langle \mathbf{y}|\mathbf{x} \rangle &= \int \mathbf{y}\ p(\mathbf{y}|\mathbf{x})\ d\mathbf{y} \\ &= \frac{\sum_{k=1}^{K} \mathbf{f}(\mathbf{x}, \mathbf{a}_k)\ p(\mathbf{x}|c_k)\ p(c_k)}{\sum_{k=1}^{K} p(\mathbf{x}|c_k)\ p(c_k)} \end{aligned} \tag{9}$$

Expression (9) is used as our predictor function. We observe that the predicted $\mathbf{y}$ is a superposition of the local functions, where the weight of each contribution depends on the posterior probability that an input point was generated by a particular cluster. The denominator assures that the sum over the weights of all contributions equals unity.

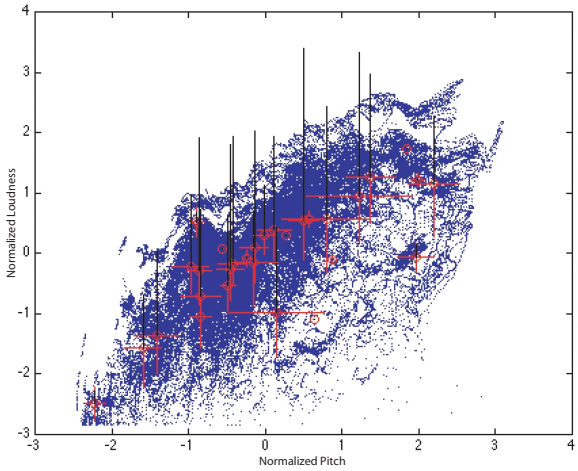

Fig. 5: Example of the female singing voice data. 89950 data points and allocated clusters in a two dimensional input space. The vertical and horizontal lines represent the weight and variances of each cluster.

Model Estimation

The model parameters are found in an iterative search which uses two estimators combined in a joint update: we use the expectation-maximization algorithm (EM) to find the parameters of the Gaussian kernels and fit the local model parameters by an inversion of the local covariance matrix.

The EM algorithm has been widely used as an efficient training algorithm for probabilistic networks [1]. Given some experimental data, EM assumes that there is a set of known states (the observed data) and a set of hidden states, characterizing the model. If the hidden states were known, model estimation would be easy, because we would only need to maximize a parameterized distribution. Yet, since we don't know the hidden states we need an iterative search for a solution that satisfies the constraints on the hidden states and maximizes the likelihood of the known states. The EM algorithm converges to a maximum of the likelihood of the observed data, reachable from the initial conditions. Unlike conventional kernel-based techniques, CWM requires only one hyper-parameter to be fixed beforehand, the number of Gaussian kernels K. Other parameters of the model are results of the estimation process rather than an inputs to the training algorithm. K is determined by cross-validation on left-out data or in a boot-strapping approach.

A detailed description of the search updates is available in [13] and [3].

REAL-TIME IMPLEMENTATION IN MAX/MSP

The analysis, prediction, and synthesis system has been completely implemented in the Max/MSP [9, 18] environment. The new library of Max objects includes the following utility functions:

1. `CWM-model` infers a CWM model from training data. The function reads in multi-dimensional feature and target data from two independent data files. It then optimizes the coefficients of the probabilistic network to best fit the nonlinear function that maps the input vector to the output vector. After convergence, the object creates a third text file that contains the model data including a description of the specific architecture, i.e. the dimensionality of the problem and the coefficients of the network. The object takes the arguments *myModelName*, *numberOfClusters*, *NumberOfIterations*, and *polynomialOrder*. The object is generic and can be used to model other nonlinear systems.

2. `CWM-predict` reads in the text file containing the model data at start-up. Given a stream of input data, that is a list containing the elements of the feature vector, the object continuously predicts output lists, which in our application contain a spectral parameterization of the predicted sound. The object takes one argument: *myModelName*.

3. `loudness~` calculates energy and estimates loudness of a signal in the time or the spectral domain giving the user a choice of window (Rectangle, Hanning, Hamming, or Blackman), window size N (default: 1024 points), and a percentage of window overlap (default: 50%). For loudness estimation, the object uses the Fletcher-Munson curves to best approximate the spectral response of the ear and outputs a loudness value in a linear, logarithmic or dB scale. The frequency bins k of the power-spectrum are weighted by coefficients W_k obtained from the interpolation of the Fletcher-Munson curves:

$$\text{loudness} = \sum_{k=2}^{\frac{N}{2}+1} \left(W_k \cdot |a_k|^2\right) \tag{10}$$

where a_k is the linear amplitude of frequency bin k up to $N/2+1$ FFT bins. $N/2+1$ corresponds to the frequency $F_s/2$. Note that the lowest bin is discarded to avoid unwanted bias from DC component.

4. `brightness~` estimates brightness by calculating the spectral centroid of a frame [16]. Like `loudness~`, the object offers different choices of FFT windows, window sizes and overlaps.

$$\text{centroid} = \frac{\sum_{k=2}^{\frac{N}{2}+1} f_k \cdot a_k}{\sum_{k=2}^{\frac{N}{2}+1} a_k} \tag{11}$$

where f_k is the frequency in Hz of frequency bin k.

Both `loudness~` and `brightness~` use specifically optimized FFTs. As phase is irrelevant in our application, we can perform the FFT twice as fast by considering only the real component of the FFT. We exploit the symmetry of the transform and split the audio data set in half. One data set takes the even-indexed numbers and the other the odd-indexed numbers, thereby forming two real arrays of half the size. The second real array is treated as a complex array [8].

5. The MSP extension `fiddle~` is used for real-time pitch extraction [10].

6. The MSP extension `sinusoid~` is used for real-time additive synthesis [2].

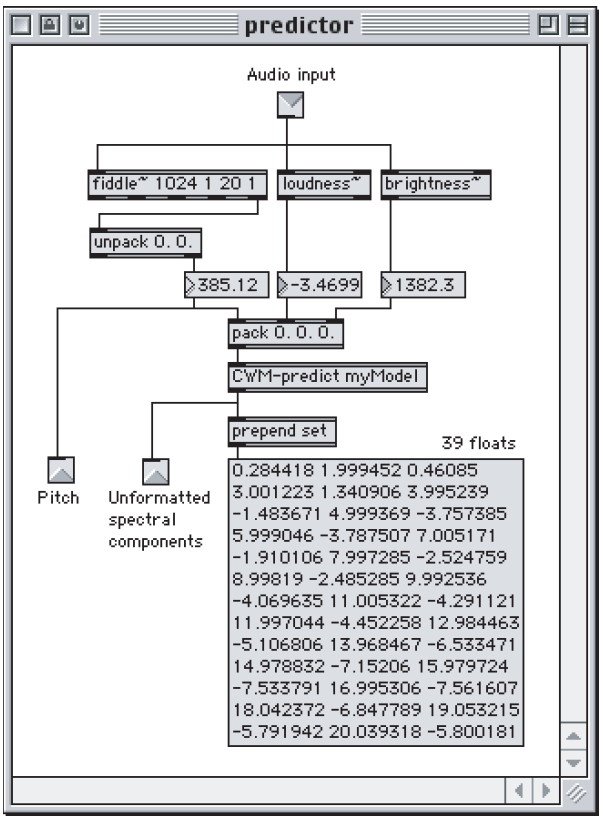

Fig. 6: Max predictor subpatch screen shot.

The implementation requires modest amounts of computing resources. A full timbre-prediction model needs as little as a few tens of kilobytes of text in storage. For combined real-time timbre prediction and synthesis using three perceptual input features and thirty sinusoidal output components, less than 15% of CPU time on a 500MHz Macintosh G4 is required.

Whereas the real-time synthesis is fast, the offline modeling step is computationally intensive. Depending on the complexity of the model, a few hours of computation at 100% CPU load are needed for optimization of the model parameters.

APPLICATIONS

Timbre synthesis

Several timbre models were created including models of a male and a female singing voice, a Stradivarius violin, and woodwind instruments. Up to 20 minutes of sound data covering a range of possibilities of each instrument were recorded, i.e. various pitches, dynamics, and playing styles. For instance, we instructed the musicians to play long glissandi, various volumes, sharp and soft attacks, vibratos, etc. Since the room acoustics affect the timbre of the instruments considerably, the recording room was kept as dry as possible and the microphone was placed carefully. In the case of the violin, we used a directional Neumann microphone located about three feet above the violinist's head. We used up to 100,000 data points (time frames) for each timbre model.

We are able to control timbre synthesis dynamically. The technique allows for continuous changes in articulation and musical phrasing, and for highly responsive sound output. The output sound doesn't suffer from undesired artifacts due to sample interpolation (i.e. smooth transition from one sample to another), sample looping (in order to maintain sustain), and pitch shift (e.g. when simulating a slide). The sound quality scales nicely with the number of sinusoidal and polynomial basis functions. The number of harmonics used ranged from a few to up to 30 in different experiments.

Figure (7 - *left*) shows the reproduction of the first seven harmonics predicted by a violin model based on violin input. The plain line represents the spectrum extracted from recorded data and the dashed line represents the predicted data. The predicted signal is close to indistinguishable from the original.

Cross-synthesis

Instead of driving an instrument model with control data generated on the same instrument, we mix controls and timbre models from different instruments. For example, the audio signal generated on an electric violin controls the model of a female singing voice. The resulting sound output imitates the timbre of the singing voice, while it follows the musical intentions and control encoded in the perceptual control signal.

As was pointed out earlier, we rescale the violin loudness and brightness functions to fall into the loudness and brightness range of the singing voice. However, in order to really sound like the original voice, the violinist needs to imitate the articulation and vibrato of the original singer. The pitch range accessible to the violinist is essentially limited to the pitch range of the recorded singer.

Figure (7 - *right*) shows an example of cross-synthesis between a violin controller and a female singing voice model. Comparing this figure to figure (7 - *left*), we observe that the predicted harmonics significantly differ from the measured harmonics of the violin. This indicates that violin and singing voice timbres are highly distinguishable.

In order to extend the output pitch range, we interpolate between different voice models, i.e. the model of a female and a male voice. The interpolation (see section *Morphing*) is strictly done in the frequency domain, which assures that the resulting sound is artifact-free and does not sound like two cross-fading voices.

Morphing

The timbre modeling approach enables morphing between different timbres. Because the structure parameterization is kept equal across different sounds, we can interpolate between parameterizations and models. In the applications discussed above, the electric violin controls either the sound of a modeled Stradivarius violin or the sound of a female singing voice. We can choose to synthesize any timbre "in between" by running two predictions simultaneously and creating two spectral frames, one representing a violin spectrum and the other one representing a voice spectrum. We introduce a morphing parameter α to weight the two spectra ($0 < \alpha < 1$):

$$C_i(n) = C1_i(n) \cdot \alpha + C2_i(n) \cdot (1 - \alpha) \qquad (12)$$

where $C1_i$ and $C2_i$ are the output components i of model 1 and 2, and C_i are the resulting components i of the morphed spectrum for time frame n.

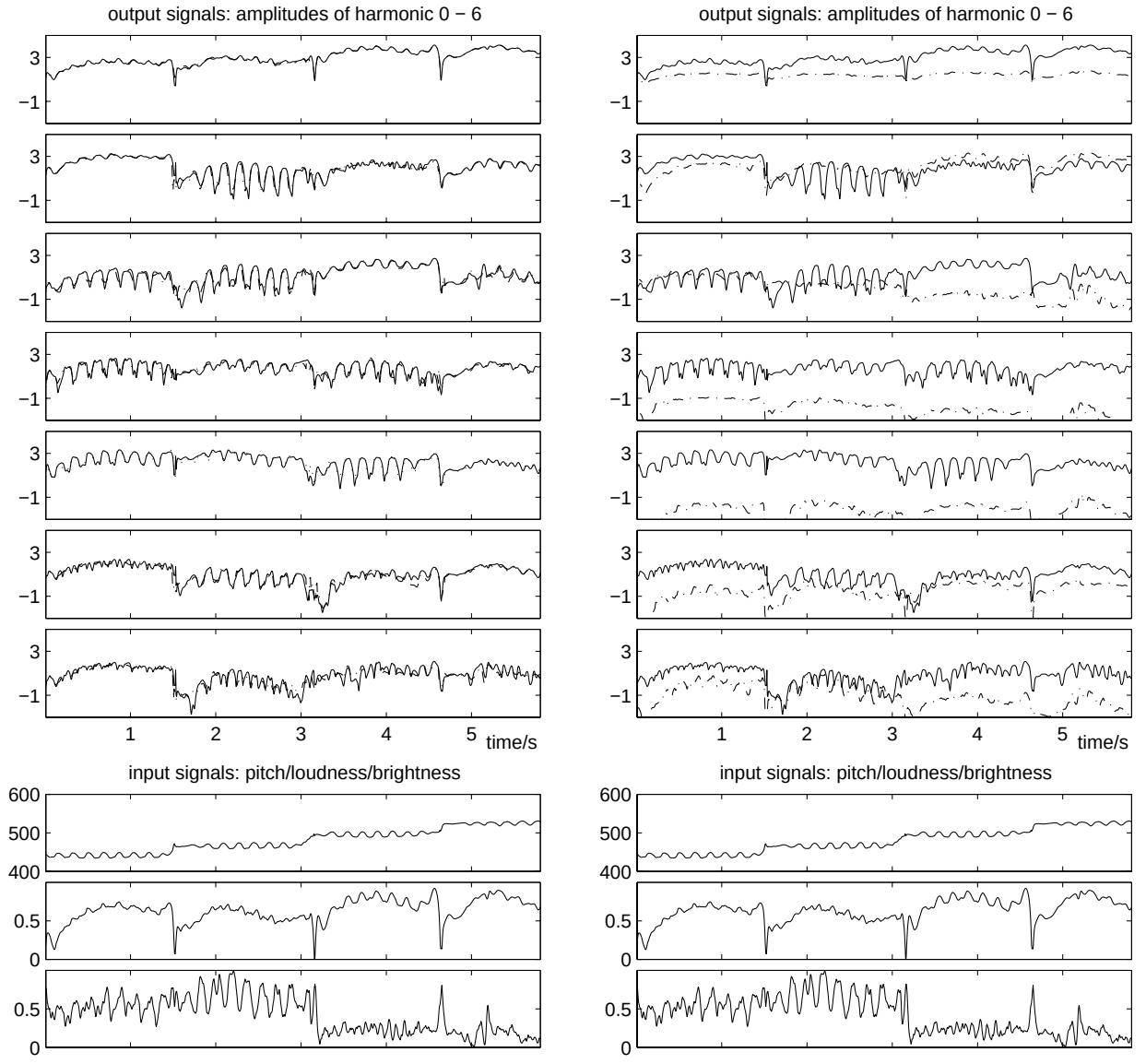

Fig. 7: *left:* Violin-control input driving a violin model. The bottom figure represents the three perceptual inputs and the top figure represents the first seven harmonics extracted from the recorded signal (plain line) and predicted using the model (dashed line) - fundamental at the top, sixth harmonic at the bottom. *right:* Violin-control input driving a female singing voice model. The bottom figure represents the three perceptual inputs and the top figure represents the first seven harmonics extracted from the recorded signal (plain line) and predicted using the voice model (dashed line).

α is specified offline or is changed dynamically. For example we can use a MIDI controller such as a volume pedal or a bow sensor, to modify the percentage of each timbre in real time.

Compression

The proposed methodology of timbre modeling and synthesis can be used for efficient audio compression and transmission.

Since the amount of input data for the sound model is very small, i.e. three floats at 86Hz, the control parameters can be easily sent over the internet in real time. Remote real-time synthesis through an ethernet network was performed suc-

cessfully by transmitting the control data with OpenSound Control (OSC) [17]. The system handles *missing data* robustly because the client synthesizes the audio signal from scratch. Missing frames can easily be replaced by previously received information. The client continuously generates the time domain signal based on the data that was last received properly. Hence there are no audible artifacts.

Discussion

Our approach preserves the perceptual parameters of the audio signal and only transforms the spectral content of the musical message. The response sounds surprisingly close to the target instrument while preserving the musical intents of the player. From the musician's perspective, the playability of the instrument is preserved and the instrument behaves intuitively and predictably.

In the case of cross-synthesis, i.e. the control features of the played instrument are used as inputs for the model of a different instrument, the resulting timbre may not always sound exactly as expected. The perceptual control of one instrument family may look very different from that of another family. In particular the attack characteristics of an instrument vary strongly across different instruments and the loudness curve of instrument A may have a much sharper attacks at the onset of new notes than instrument B. For instance, a modeled guitar sound generated from the stroke of a violin does not have the very characteristic sharp attack and long logarithmic release as we could expect it but rather a slow attack, flat sustain, and shorter release, more characteristic of a violin envelope. This limitation is not necessarily a problem because musicians usually agree that the expressivity of controls is more important than the reproduction of specific wave forms. In other words, the violin-guitar controller may not behave exactly like a guitar but it provides the violinist with an expressive tool that expands his/her artistic space. Yet, the design of a new computer instrument and controller should not underestimate the familiarity and closeness between the musician and his/her specific instrument controller. It should also not underestimate the ability of the musician to adapt to a new controller or feedback mechanism driven by the will to achieve an artistic goal.

CONCLUSIONS AND FUTURE WORK

We have presented a perceptually meaningful acoustic timbre synthesizer for non-discretely pitched acoustic instruments such as the violin. The timbre is modeled based on the spectral analysis of natural sound recordings using the probabilistic inference framework Cluster-Weighted Modeling. The timbre and sound synthesizer is driven by the perceptual features pitch, loudness, and brightness which are extracted from an arbitrary monophonic input audio stream. The perceptual features can be thought of as controllers for the timbre model predictor. The predictor model CWM outputs the most likely set of spectral parameters which are then used in an additive synthesis approach to generate an audio stream. The real-time system is implemented in Max/MSP on a Macintosh platform.

A noise model that is based on a polynomial expansion of the noise spectrum is currently being integrated into the system. This extension enables a more accurate model and representation of genuinely noisy instruments such as the flute or the shakuhachi.

Future work will include algorithms that extract more perceptual features. In particular a better indicator for noisiness is needed. This can be achieved by considering high-level insights into the music making, for example, considering note onsets or bow changes in the case of a violin signal. Regarding the computing infrastructure we will design an `analyzer~` MSP extension that combines the extraction of pitch, loudness, and brightness in a single object. We will also extend the application with models of very different instruments such as trombone and flute.

ACKNOWLEDGEMENTS

The authors would like to thank Tod Machover and Neil Gershenfeld for their support, Joshua Bell for playing his beautiful Stradivarius violin for the purpose of data collection, Tara Rosenberger and Youngmoo Kim for lending their voices, Nyssim Lefford and Michael Broxton for help with the recordings, Cyril Drame whose research inspired this work, Ricardo Garcia for technical help, Catherine Vaucelle for conceptual and personal support, and Mary Farbood for valuable assistance regarding the English language. Part of the MSP analysis objects were developed at Berkeley's Center for New Music and Audio Technologies: thanks to the CNMAT team. This work was made possible by the Media Lab's Things That Think consortium and by Sega Corporation.

REFERENCES

[1] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood From Incomplete Data via the EM Algorithm. *J. R. Statist. Soc. B*, 39:1–38, 1977.

[2] A. Freed and T. Jehan. CNMAT Max/MSP externals available at *http://www.cnmat.berkeley.edu/max/*, 1999.

[3] Neil A. Gershenfeld, Bernd Schoner, and Eric Metois. Cluster-weighted modeling for time series analysis. *Nature*, 379:329–332, 1999.

[4] Michael Goodwin. Residual modeling in music analysis/synthesis. In *Proceedings of ICASSP*, volume 2, pages 1005–1008, 5 1996.

[5] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[6] Eric Métois. *Musical Sound Information. Musical Gestures and Embedding Synthesis.* PhD thesis, MIT Media Lab, 1996.

[7] A. Michael Noll. Cepstrum pitch determination. *Journal of Acoustic Society of America*, 41(2):293–309, 1967.

[8] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, New York, 2nd edition, 1992.

[9] M. Puckette. The patcher. In *Proceedings International Computer Music Conference*, Koln, 1988.

[10] M. Puckette and T. Apel. Real-time audio analysis tools for pd and msp. In *Proceedings International Computer Music Conference*, 1998.

[11] L.R. Rabiner. On the use of autocorrelation analysis for pitch detection. In *Proceedings of IEEE*, volume 58, pages 707–712, 1970.

[12] Xavier Rodet. Musical sound signal analysis/synthesis: Sinusoidal + residual and elementary waveform models. In *IEEE Time-Frequency and Time-Scale Workshop*, Coventry, Great Britain, 1997.

[13] B. Schoner, C. Cooper, C. Douglas, and N. Gershenfeld. Data-driven modeling and synthesis of acoustical instruments. In *Proceedings International Computer Music Conference*, pages 66–73, Ann Arbor, Michigan, 1998.

[14] Xavier Serra. Musical sound modeling with sinusoids plus noise. In *Musical Signal Processing*, 1997.

[15] D. Wessel, C. Drame, and M. Wright. Removing the time axis from spectral model analysis-based additive synthesis: Neural networks versus memory-based machine learning. In *Proceedings International Computer Music Conference*, pages 62–65, Ann Arbor, Michigan, 1998.

[16] David L. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, 3(2):45–52, 1979. republished in *Foundations of Computer Music*, Curtis Roads (Ed., MIT Press).

[17] Mathew Wright and Adrian Freed. OpenSound control: A new protocol for communicating with sound synthesizers. In *Proceedings International Computer Music Conference*, 1997.

[18] D. Zicarelli. An extensible real-time signal processing environment for Max. In *Proceedings International Computer Music Conference*, Ann Arbor, Michigan, 1998.