An Audio-Driven Perceptually Meaningful Timbre Synthesizer

Tristan Jehan, Bernd Schoner*
MIT Media Laboratory
email: tristan@media.mit.edu, schoner@media.mit.edu

Abstract

A real-time synthesis engine that models and predicts the timbre of acoustic instruments based on perceptual features is presented. The timbre characteristics and the mapping between control and timbre parameters are inferred from recorded musical data. In the synthesis step, timbre data is predicted based on new control data enabling applications such as synthesis and cross-synthesis of acoustic instruments and timbre morphing between instrument families. The system is fully implemented in the Max/MSP environment.

1 Introduction

Timbre is generally defined as the quality of a sound that distinguishes it from other sounds of the same pitch and loudness. Based on the realization that timbre distinguishes one instrument from the other, we present a novel approach to control and synthesize the timbre and sound of different instruments. Perceptual parameters, namely pitch, loudness, and brightness, are extracted from the audio stream of an acoustic monophonic instrument and are used as input features for the prediction of spectral data, i.e. the timbre characteristics of a particular instrument.

In our approach to timbre modeling, we start by extracting perceptual features and a sinusoidal representation from the audio data. We then infer a mapping between perceptual features and the spectral representation. During synthesis this mapping is used to predict a harmonic structure of the timbre from new control parameters (e.g. from a muted instrument). The inference and prediction system was implemented using Cluster-Weighted Modeling (CWM), a probabilistic toolkit for nonlinear function approximation (Schoner et al., 1998; Gershenfeld et al., 1999).

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. Hence we are able to model instruments based on recordings of the instrument without redesigning the model architecture. Furthermore, we can exchange control and audio signals of different instruments since the perceptual representation and data structure is preserved across different instruments and instrument families. This novel technique approach enables several applications, including the cross-synthesis and morphing of musical 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 (see *Max/MSP Implementation*). In particular we present a new *Max/MSP* external that executes the full analysis functionality in real time (`analyzer~`).

2 Previous Work

There have been a variety of relevant attempts to use connectionist models to synthesize and control the sound from acoustic instruments.

Métois (1996) introduces the synthesis technique *Psymbesis* (Pitch Synchronous Embedding Synthesis). He defines a vector of perceptual control parameters including pitch, loudness, and brightness and clusters normalized sound periods in a low-dimensional space. For synthesis Métois re-samples the periods at the desired pitch, and adjusts them at the desired loudness.

Wessel et al. (1998) analyzed and parameterized a database of sounds with respect to pitch, loudness, and brightness and decomposed the sample sounds into frames of spectral data. The perceptual parameters serve as inputs to a feed-forward network, whereas the spectral parameters serve as outputs. The network is trained to represent and predict a specific instrument using an artificial neural network and a memory-based network.

Schoner et al. (1998) used Cluster-Weighted Modeling to predict a spectral sound representation given physical input to the instrument. While the target data was similar to the data used in (Wessel et al., 1998), the feature vector consisted of actual physical gestures of the violin player. Special recording hardware was needed to create the set of training data and to replay the model.

This paper combines the efficiency of Cluster-Weighted Modeling with spectral synthesis and the idea of perceptual controls.

*Current address: ThingMagic LLC, Cambridge, MA 02142

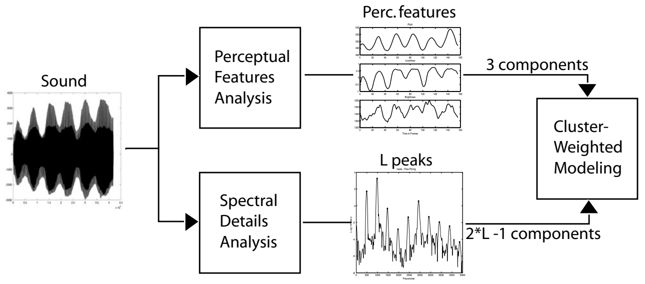
Sound
Perceptual Features Analysis
Perc. features
3 components
Cluster-Weighted Modeling
Spectral Details Analysis
L peaks
2*L -1 components

Figure 1: Timbre analysis and modeling using Cluster-Weighted Modeling.

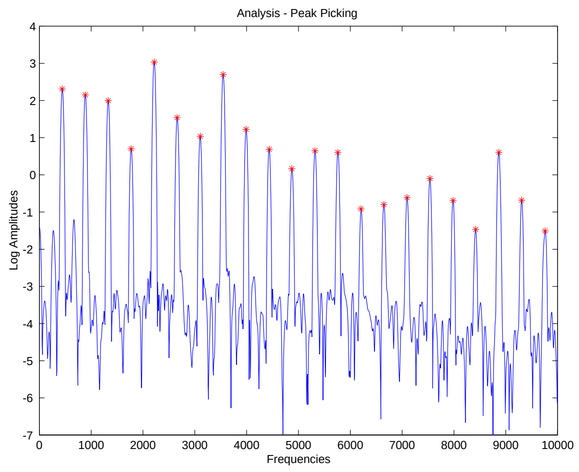

Figure 2: Spectrum of a violin (24ms frame of data). The stars indicate the harmonic peaks of the spectrum as found by the peak tracking algorithm.

3 Analysis, Modeling and Synthesis

Our approach to timbre modeling is based on the following 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.

We conclude that a unique spectral representation of a sound can be inferred given perceptual sound data and a timbre model. We estimate both perceptual and spectral representations from recorded data and then predict the latter given the former.

We analyze the sound recordings frame by frame using a short-term Fourier transform (STFT) with overlapping frames of 24ms at intervals of 12ms. Long windows (e.g. 2048-4096 points at 44.1KHz) with large zero-padded FFTs can be used since latency is not an issue here.

A spectral peak-picking algorithm combined with instantaneous frequency estimation (see next paragraph) finds the partial peaks from one analysis frame to the next, resulting in L $(= 10$ 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, 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 (Noll, 1967) and then operate on the harmonic peaks of the STFT. Because 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 use the instantaneous frequencies[1] of the selected bins to obtain a much higher resolution with little extra computation (Métois, 1996).

Given the spectral decomposition, we extract pitch as the frequency of the fundamental component. In order to extract the instantaneous loudness of the signal, the power-spectrum bins are weighted by coefficients from the Fletcher-Munson curves which simulate the frequency response of the ear. The spectral centroid of the signal is used as an estimator for the brightness of the sound (Wessel, 1979). In a second pass through the data, estimation errors are detected and eliminated. Frames that are considered "bad" are dropped. The peaks of the spectrum are used as an harmonic representation of the audio signal and as target data for our predictive model.

In the synthesis step we start with 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. The output vector is used for an additive synthesis engine that modulates sinusoidal components and superimposes them in the time domain, resulting in the deterministic component of the signal. In the next section, we show how a stochastic noise process will be added to create a more accurate timbre representation.

We observe that the timbre of any particular instrument or instrument family is contained in the predictor model (see

[1] Given the non-windowed discrete Fourier transform $X(k) = \sum_{n=0}^{n-1} s(n)e^{-jwnk}$ with $w = \frac{2\pi}{N}$ the estimate for bin k's instantaneous frequency is:

$$\begin{aligned} F_{\text{inst}}(k) &= F_s\left(\frac{k}{N} + \frac{1}{2\pi}\text{Arg}\left[\frac{A}{B}\right]\right) \qquad (1) \\ A &= X(k) - \frac{1}{2}\left[X(k-1) + X(k+1)\right] \\ B &= X(k) - \frac{1}{2}\left[e^{jw}X(k-1) + e^{-jw}X(k+1)\right] \end{aligned}$$

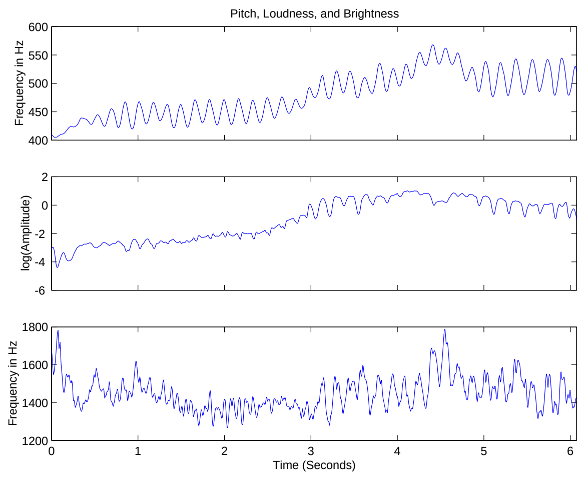

Figure 3: Typical perceptual-feature curves for a female singing voice.

section *Cluster-Weighted Modeling*), whereas the musical intent is contained in the parameterization of the perceptual control data.

4 Noise Analysis/Synthesis

We seek to improve the sound quality of the additive synthesis approach by modeling the residual nondeterministic components of the sound in addition to the deterministic harmonic components (Serra, 1997; Rodet, 1997). While the harmonic structure is usually described as a sum of sinusoidal functions, there are several approaches to model the residue or noise spectrum (Goodwin, 1996). Here we present a novel approach to noise modeling by means of a polynomial expansion.

In general, the power spectrum of the non-harmonic components of the signal reflects its noise characteristics. Subtracting the power spectrum of the deterministic harmonic components from the power spectrum of the mixed signal we extract the residual spectrum for each time frame. We then approximate the shape of the power spectrum of the residue by means of a superposition of polynomial basis functions:

$$f(\mathbf{x}) = \sum_{k=0}^{K} a_k f_k(\mathbf{x}) \tag{2}$$

where $f_k(\mathbf{x})$ are all polynomial function of $\mathbf{x}$ up to a given polynomial order. The input vector $\mathbf{x}$ consists of the usual perceptual parameters as well as a noise/signal ratio (noisiness) estimator. In addition, $\mathbf{x}$ contains the frequency f of a particular point in the spectrum.

We use up to 30 basis functions and coefficients. The coefficients are determined by means of a simple matrix inversion generating the best solution in the least square sense:

$$\begin{aligned} \mathbf{a} &= \mathbf{B}^{-1} \cdot \mathbf{c} \\ [\mathbf{B}]_{ij} &= \langle f_i(\mathbf{x}) \cdot f_j(\mathbf{x}) \rangle \\ [\mathbf{c}]_j &= \langle y \cdot f_j(\mathbf{x}) \rangle \end{aligned} \tag{3}$$

where $\langle \cdot \rangle$ is the inner product.

In the synthesis step, white noise is multiplied with the reconstructed noise function in the spectral domain. After scrambling the perceptually irrelevant phase information the colored noise is transformed back into the time domain using an inverse FFT. The method is particularly successful with breath noise that appears in the residue of instruments like the flute. The accuracy of the noise model scales with the number of basis functions used and can be adjusted during synthesis. Computational speed is generally not an issue when executed on a state of the art PC or Macintosh computer.

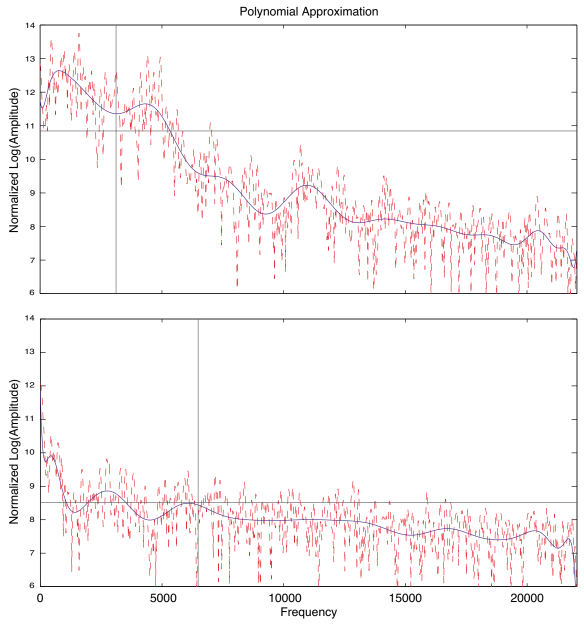

Figure 4: Typical noise spectrum of the violin (24ms FFT) approximated with a polynomial function (25 basis functions) at the onset of a new note (*top*) and at decay (*bottom*). The vertical line shows brightness and the horizontal line shows loudness.

Our parameterization of sound is comparable to Serra's *Spectral Modeling Synthesis* implementation (Serra, 1997). However, our system removes the temporal axis to dynamically control musical features by generating new envelope functions in real time.

5 Cluster-Weighted Modeling

We approximate the nonlinear mapping from the feature vector to the harmonic target vector using the general inference framework, Cluster-Weighted Modeling. 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 and can be interpreted as a flexible and transparent technique to approximate an arbitrary function. 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.

The model architecture has been described in detail in different places (Schoner et al., 1998; Gershenfeld et al., 1999). Here we only indicate the predictor function $\mathbf{y}(\mathbf{x})$:

$$\mathbf{y}(\mathbf{x}) = \frac{\sum_{k=1}^{K} \mathbf{f}(\mathbf{x}, \mathbf{a}_k)\; p(\mathbf{x}|c_k)\; p(c_k)}{\sum_{k=1}^{K} p(\mathbf{x}|c_k)\; p(c_k)} \tag{4}$$

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

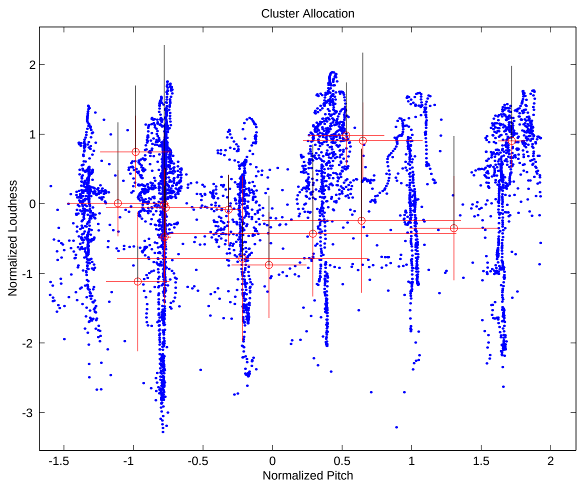

Figure 5: Example with 4622 Stradivarius data points (6 notes) and allocated clusters in a two-dimensional input space. The vertical and horizontal lines represent the weight and variances of each cluster.

The model parameters are found in an iterative search based on the Expectation-Maximization algorithm (EM). A detailed description of the search updates is available in (Schoner et al., 1998) and (Gershenfeld et al., 1999).

6 Max/MSP Implementation

The analysis, prediction, and synthesis system has been completely implemented in the Max/MSP environment (Puckette, 1988; Zicarelli, 1998) and runs in real time. 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 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 only one argument: *myModelName*.

3. `analyzer~` estimates the following series of perceptual features: pitch, loudness, brightness, noisiness, onsets, and Bark scale decomposition. The user chooses the type of window (Rectangular, Bartlett, Welch, Hanning, Hamming, or Blackman), the window size N (default: 1024 points), the percentage of window overlap (default: 50%), and the FFT size (default: 1024 points). Pitch and onset estimations are based on the MSP extension `fiddle~` (Puckette and Apel, 1998). Loudness is estimated by weighting the frequency bins k of the power spectrum by the coefficients W_k obtained from the interpolation of the Fletcher-Munson curves:

$$\text{loudness} = \sum_{k=2}^{\frac{N}{2}+1} \left(W_k \cdot |a_k|^2\right) \tag{5}$$

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

The spectral centroid of a frame (Wessel, 1979) measures brightness:

$$\text{centroid} = \frac{\sum_{k=2}^{\frac{N}{2}+1} f_k \cdot a_k}{\sum_{k=2}^{\frac{N}{2}+1} a_k} \tag{6}$$

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

The Spectral Flatness Measure (SFM) determines if the actual frame is more noise-like or tone-like. It is defined as the ratio of the geometric to the arithmetic mean of the energy per critical band E_b, expressed in dB:

$$\mathrm{SFM_{dB}} = 10 \log 10 \left\{ \frac{\left(\prod_{b=1}^{b_t} E_b\right)^{\frac{1}{b_t}}}{\frac{1}{b_t}\sum_{b=1}^{b_t} E_b} \right\} \qquad (7)$$

where b_t is the total number of critical bands on the signal. In `analyzer~`, we first decompose the spectrum into a Bark scale, which gives 25 bands at 44.1 KHz (see below).

The SFM value is used to calculate the noisiness or "tonality factor" (Johnston, 1988) as follows:

$$\alpha = \min\left(\frac{\mathrm{SFM_{dB}}}{\mathrm{SFM_{dBmax}}}, 1\right) \qquad (8)$$

with $\mathrm{SFM_{dBmax}} = -60\mathrm{dB}$. The closer α is to zero, the noisier the frame is.

The Bark scale is an auditory filter bank (Smith and Abel, 1999) with the number of bands depending on the sampling rate: 25 bands at 44.1 KHz. It is estimated from the FFT using the approximation function (Sporer and Brandenburg, 1995):

$$b = 13\tan^{-1}\left(\frac{0.76 * f}{1000}\right) + 3.5\tan^{-1}\left(\left(\frac{f}{7500}\right)^2\right)$$

where f is the frequency in Hertz and b is the mapped frequency in Barks.

`analyzer~` uses a specifically optimized real-FFT. As phase is irrelevant in our application, we can perform the FFT twice as fast by considering only the real components 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.

An optional phase argument delays the initial FFT. Several objects may run together without having to compute all parallel FFTs simultaneously since their occurrences are unsynchronized. The object was measured to use less than 5% of CPU load on a 500 MHz Macintosh G4 with a 4096-point FFT overlapping by 3584 points at 44.1 KHz.

4. Externals which extract each of the described perceptual parameters individually are also available: `pitch~`, `loudness~`, `brightness~`, `noisiness~`, and `bark~`.

5. The MSP extension `sinusoids~` is used for real-time additive synthesis (Freed and Jehan, 1999).

The full implementation requires modest amounts of computing resources. A 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.

7 Applications

7.1 Timbre synthesis

Several full timbre models were created including models of a male singing voice, two soprano female singing voices, 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 singers to sing long glissandi, various volumes, sharp and soft attacks, vibratos, etc. We also used the McGill University Master Sample library for woodwind instruments.

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 does not suffer from undesired artifacts due to sample interpolation, sample looping, and pitch shift. The sound quality scales nicely with the number of sinusoidal and polynomial basis functions. The number of harmonics used ranges from a few to up to 40 in different experiments. Before synthesis the user can dynamically change the control data in order to modify the pitch (see *pitch shifting*), loudness or brightness of the output sound.

Figure (6 - *left*) shows the reproduction of the first seven harmonics predicted by a full female singing voice model based on a new female singing voice input. The predicted signal matches closely the original although the input data was not part of the training data.

7.2 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, a singer controls the model of a Stradivarius model or alternatively, the audio signal generated on an electric violin controls the

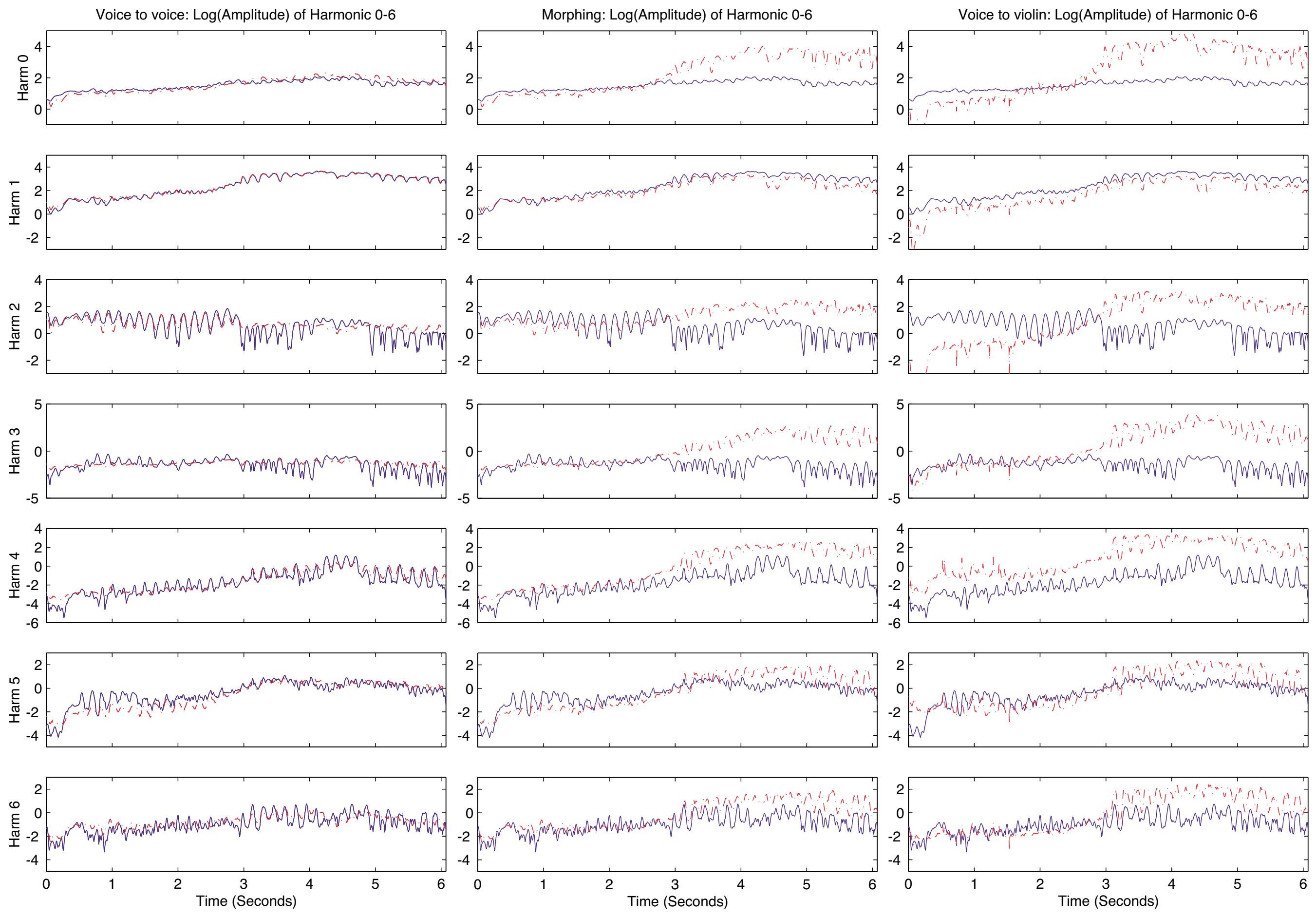

Figure 6: Three results of prediction using for each one the three perceptual inputs from the female singing voice of figure 3. Each figure represents the first seven harmonics extracted from the recorded signal (plain line) and predicted signal using the model (dashed line) - fundamental at the top, sixth harmonic at the bottom. *left:* The input drives a full female singing voice model. *right:* The input drives a full Stradivarius violin model. *middle:* The input drives a linear morphing between the two models (from voice to violin).

model of a female singing voice. The resulting sound output imitates the timbre of the violin in the first case and the singing voice in the second case, while it follows the musical intentions and control encoded in the perceptual control signal.

We rescale the loudness and brightness functions of the source to fall into the loudness and brightness range of the target model. However, in order to really sound like the original, the controller instrument needs to imitate the articulation and vibrato of the original target. The pitch range accessible to the source instrument is essentially limited to the pitch range of the recorded target instrument.

Figure (6 - *right*) shows an example of cross-synthesis between a female singing voice controller and a violin model. Comparing this figure to figure (6 - *left*), we observe that the predicted harmonics differ significantly from the measured harmonics of the voice. This indicates that violin and singing-voice timbres are highly distinguishable although the two sounds have the same relative values of pitch, loudness, and brightness.

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-faded voices.

7.3 Morphing

Because the parameterization structure is kept equal across different instruments, 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 ($0 < \alpha < 1$):

$$C_i(n) = C_{1,i}(n) \cdot \alpha + C_{2,i}(n) \cdot (1 - \alpha) \tag{9}$$

where $C_{1,i}$ and $C_{2,i}$ 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. 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.

Figure (6 - *middle*) shows an example of linear morphing between a female singing voice and a Stradivarius violin, using a voice input. The signal at first matches the predicted voice signal of example (6 - *left*) and in the end matches the predicted violin predicted signal of example (6 - *right*).

7.4 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. about 1Kb/s, the control parameters can be easily sent over the internet in real time. Remote real-time synthesis through an Ethernet network was performed successfully by transmitting the control data with OpenSound Control (OSC) (Wright and Freed, 1997).

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.

Figure (7 - *left*) shows the analysis server Max patch that was used to analyze a streaming electric violin signal and to send the resulting control data over the network. Figure (7 - *right*) shows the synthesis client Max patch that was used to receive the control data and to synthesize a female singing voice.

We are also experimenting with a five-string polyphonic electric violin that is used to control different sound models on each string. Such a system allows the musician to play double-stops with a different sound for each note. Such complex set-ups can either be handled on one machine or shared by several machines.

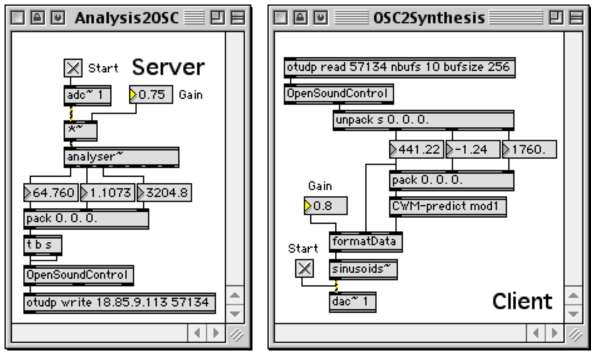

Figure 7: *left:* Max analysis server patch, *right:* Max synthesis client patch.

7.5 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 attack at the onset of new notes than instrument B. This limitation is not necessarily a problem because musicians usually agree that the expressivity of controls is more important than the reproduction of the specific waveforms. In other words, the hybrid (cross-synthesized) instrument may not behave exactly like the original target sound but it provides the musician with an expressive tool that expands his or her artistic space. Designing new computer instruments and controllers, we should not underestimate the familiarity and closeness of the skilled musician with his or her instrument, although he or she can adapt to a new controller or feedback mechanism.

8 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 controlled by the perceptual features pitch, loudness, and brightness which are extracted from an arbitrary monophonic input audio stream. The predictor model 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 efficiently in Max/MSP on a Macintosh platform.

A noise model that is based on a polynomial expansion of the noise spectrum 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. We will also extend the application with models of very different instruments such as trombone and flute.

9 Acknowledgements

The authors would like to thank Tod Machover, Neil Gershenfeld, Joshua Bell, Hila Plittman, Tara Rosenberger, Youngmoo Kim, Nyssim Lefford, Michael Broxton, Cyril Drame, Ricardo Garcia, Catherine Vaucelle, Mary Farbood, and CNMAT. This work was made possible by the Media Lab's Things That Think consortium and by Sega Corporation.

References

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

Gershenfeld, N. A., Schoner, B., and Métois, E. (1999). Cluster-weighted modeling for time series analysis. *Nature*, 379:329–332.

Goodwin, M. (1996). Residual modeling in music analysis/synthesis. In *Proceedings of ICASSP*, volume 2, pages 1005–1008.

Johnston, J. D. (1988). Transform coding of audio signals using perceptual noise criteria. *IEEE on Selected Areas in Communications*, 6:314–323.

Métois, E. (1996). *Musical Sound Information. Musical Gestures and Embedding Synthesis*. PhD thesis, MIT Media Lab.

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

Puckette, M. (1988). The patcher. In *Proceedings International Computer Music Conference*, pages 420–429, Köln, Germany.

Puckette, M. and Apel, T. (1998). Real-time audio analysis tools for Pd and MSP. In *Proceedings International Computer Music Conference*, pages 109–112, Ann Arbor, Michigan.

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

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

Serra, X. (1997). Musical sound modeling with sinusoids plus noise. In *Musical Signal Processing*. Swets & Zeitlinger.

Smith, J. and Abel, J. (1999). Bark and ERB bilinear transforms. *IEEE Transactions on Speech and Audio Processing*, 7(6):697–708.

Sporer, T. and Brandenburg, K. (1995). Constraints of filter banks used for perceptual measurement. *Journal of the Audio Engineering Society*, 43:107–115.

Wessel, D., Drame, C., and Wright, M. (1998). 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.

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

Wright, M. and Freed, A. (1997). OpenSound control: A new protocol for communicating with sound synthesizers. In *Proceedings International Computer Music Conference*, pages 101–104, Thessaloniki, Greece.

Zicarelli, D. (1998). An extensible real-time signal processing environment for Max. In *Proceedings International Computer Music Conference*, pages 463–466, Ann Arbor, Michigan.