Perceptual Synthesis Engine: An Audio-Driven Timbre Generator

Tristan Jehan

Master of Science in Media Arts and Sciences
at the
Massachusetts Institute of Technology

September 2001

Perceptual Synthesis Engine: An Audio-Driven Timbre Generator

Tristan Jehan

Diplôme d'Ingénieur en Informatique et Télécommunications
IFSIC — Université de Rennes 1 — France (1997)

Submited to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

Massachusetts Institute of Technology

September 2001

Author .....
Program in Media Arts and Sciences
September 2001

Certified by .....
Tod Machover
Professor of Music and Media
Thesis Supervisor

Accepted by .....
Dr. Andrew B. Lippman
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Perceptual Synthesis Engine: An Audio-Driven Timbre Generator

Tristan Jehan

Submited to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

Massachusetts Institute of Technology

September 2001

Abstract

A real-time synthesis engine which models and predicts the *timbre* of acoustic instruments based on perceptual features extracted from an audio stream is presented. The thesis 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. The system enables applications such as cross-synthesis, pitch shifting or compression of acoustic instruments, and timbre morphing between instrument families. It is fully implemented in the Max/MSP environment. The Perceptual Synthesis Engine was developed for the Hyperviolin as a novel, generic and perceptually meaningful synthesis technique for non-discretely pitched instruments.

Advisor: Tod Machover
Title: Professor of Music and Media

Perceptual Synthesis Engine: An Audio-Driven Timbre Generator

Thesis Committee

Thesis Supervisor

Tod Machover
Professor of Music and Media
MIT Program in Media Arts and Sciences

Thesis Reader

Joe Paradiso
Principal Research Scientist
MIT Media Laboratory

Thesis Reader

Miller Puckette
Professor of Music
University of California, San Diego

Thesis Reader

Barry Vercoe
Professor of Media Arts and Sciences
MIT Program in Media Arts and Sciences

To my Cati...

Preface

As a concert violinist with the luxury of owning a Stradivarius violin made in 1732, I have always been skeptical of attempts to "electrify" a string instrument. I have tried various electric violins over the years but none have compelled me to bring them to the concert hall. The traditional methods of extracting sound from a violin and "enhancing" it electronically usually result in an unappealing and artificial sound.

Recently, though, I have been intrigued by the work being done at the Media Lab by Tristan Jehan. I have had the privilege of working with him in the development of a new instrument dubbed the "hyperviolin." This new instrument uses raw data extracted from the audio of the violin and then fed into the computer. Using Tristan's "sound models," this raw data provided by me and the hyperviolin can be turned into such sounds as the human voice or the panpipes. When I first heard the sound of a singing voice coming from Tristan's computer, I thought it was simply a recording. But when I found out that it was not anyone singing at all, but merely a "print" of someone's voice applied to random data (pitch, loudness, etc.), I got excited by the possibilities.

When these sound models are used in conjunction with the hyperviolin, I am able to sound like a soprano or a trumpet (or something in between!) all while playing the violin in a normal fashion. The fact that this is all processed on the fly with little delay between bow-stroke and sound is testament to the efficiency of Tristan's software.

Tristan Jehan's work is certainly groundbreaking and is sure to inspire the minds of many musicians. In the coming months I plan to apply these new techniques to music both new and old. The possibilities are endless.

Joshua Bell

Aknowledgements

I would like to gratefully thank

my advisor Tod Machover for providing me with a space in his group, for supporting this research, and for pushing me along these two years. His ambition and optimism were always refreshing to me.

the other members of my comittee, Joe Paradiso, Miller Puckette, and Barry Vercoe, for spending the time with this work, and for their valuable insights.

Bernd Schoner for providing his CWM code and for helping me with it. He definitely knows what it means to write a paper, and I am glad he was there for the two that we have written together. Bernd is my friend.

my love Cati Vaucelle for her great support, her conceptual insight, and simply for being there. She has changed my life since I have started this project and it would certainly have not ended up being the same without her. My deepest love goes to her, and I dedicate this thesis to her.

Joshua Bell for playing his Stradivarius violin beautifully for the purpose of data collection, for his musical ideas, for spending his precious time with us, and for being positive even when things were not running as expected.

Youngmoo Kim, Hila Plittman and Tara Rosenberger for lending their voices for the purpose of data collection. Their voice models are very precious material to this work.

Nyssim Lefford and Michael Broxton for help with the recordings and sound editing.

Cyril Drame whose research and clever ideas originally inspired this work and for his friendship.

Ricardo Garcia for his valuable insight, refreshing excitement, and for his friendship.

Mary Farbood for her help correcting my English and for her support. Mary is my friend.

Laird Nolan and Hannes Högni Vilhjálmsson for useful assistance regarding the English language.

the members of the Hyperinstruments group who helped in one way or another, and for providing me with a nice work environment.

the Media Lab's Things That Think consortium, and Sega Corporation for making this work possible.

my friends and family for their love and support.

Thank you all.

Contents

List of Figures

Introduction

From the beginning, with the organ, through the piano and finally to the synthesizer, the evolution of the technology of musical instruments has both reflected and driven the transformation of music. Where it once was only an expression in sound — something heard — in our century music has also become information, data — something to be processed.

Digital audio as it is implemented at present, is not at all *structured*: controllable, scalable, and compact [Casey, 1998]. In the context of musical instruments, this is a major limitation since we would like to control every aspect of the sound in a musically meaningful manner. There are needs for higher level descriptions of sound.

Digital instruments as they are implemented today, systematically combine the notion of *gesture control* and the notion of *sound synthesis*. Typically, an arbitrary gesture is used to control at least one synthesis parameter, e.g., a key equals a fundamental frequency, velocity maps with sound amplitude, etc. This basic principle led to the MIDI[1] system almost 20 years ago. The format is in fact very well suited for the keyboard interface and its low-dimensional control space, i.e., note on/off, key number, and velocity. The sound synthesizer behind it generates a more-or-less complex waveform that can be more-or-less transformed using additional controllers such as a volume pedal or a pitch-bend joystick.

However, MIDI does not describe very well the high-dimensional instrument controllers such as the violin. While keyboards enable many synthesis

[1]Musical Instrument Digital Interface

applications, other instruments[2] 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.

Music is built from sound [Bregman, 1990] and from the interaction between the musician and the sound generated on his instrument. Music was born from listening rather than performing a gesture. The gesture is a haptic feedback mechanism in order to reach a musical goal [O'Modhrain, 2000] but the sound is the auditory feedback that has rooted the music. In that matter, I believe that perception of sound should play a key role in the sound synthesis process and the musical creation.

The purpose of this thesis is to develop a timbre model that can be used as a creative tool by professional musicians playing an arbitrary controller instrument. The model is controlled by the perceptual features pitch, loudness and brightness, extracted from the audio stream of the controller instrument, rather than the musical gestures. Ideally, the system aims to be a "universal" synthesizer or can be seen as an *interface* between a musical sound controller and a musical sound output of arbitrary timbre. Chapter 2 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.

This novel structured technique enables several applications, including the cross-synthesis and morphing of musical instruments.

The work described in this thesis was partly published in the two articles below:

[Jehan and Schoner] Jehan, T. and Schoner, B. (2001) An Audio-Driven, Spectral Analysis-Based, Perceptual Synthesis Engine. *Audio Engineering Society, Proceedings of the 110th Convention.* Amsterdam, May 2001.

[Jehan and Schoner] Jehan, T. and Schoner, B. (2001) An Audio-Driven Perceptually Meaningful Timbre Synthesizer. In *Proceedings International Computer Music Conference*, La Habana, Cuba.

[2] Violin, cello, trumpet, oboe, trombone, saxophone, or flute, to name a few

Chapter 1

Background and Concept

The appearance of new musical instruments comes together with the artistic creation and the development of new composition styles. For instance, there has been a constant evolution among keyboard instruments begining with the organ (Middle Ages), and followed by the harpsichord (14th century), piano forte (18th century), electric piano (1950's), electronic synthesizer (1960's), and digital synthesizer (1980's). Each evolution offers a particular and original new dimension in the sound output and control, although the interface is already familiar to the player.

Some musicians have changed their playing style when shifting from one instrument to another. For example Herbie Hancock — very popular jazz pianist since the 60's (The Miles Davis quintet) — played a key role in the development of the jazz-rock movement of the late 60's and 70's when playing a Fender Rhodes electric piano in his band "Headhunters" [Hancock, 1973]. New artistic values are associated with new musical instruments. These new instruments may feature original interfaces (see section *Hyperinstruments*) or they can be based on already existing interfaces, e.g., a keyboard, which has the advantage of being instantly exploitable by the already skilled musician who can find new areas to express his mature art.

Our digital age permits very ambitious developments of instruments. The information technology and signal processing algorithms now serve music composition [Farbood, 2001] and sound analysis/synthesis worlds

[Mathews, 1969]. Computing power has become cheap and available for most demanding real-time applications. The amazing success of keyboard instruments such as the Yamaha DX7 (180,000 units sold) has demonstrated the interest for new and creative digital instruments: a greater variety of sounds have become accessible to the keyboard player. Unfortunately, there is little or no digital synthesis technology available to the non-keyboard player.

What do musicians control while playing a musical intrument? They are different possible answers to that question. A non-musican would probably say things like "finger position, bow speed and pressure, amount of blown air." The expert would rather say "pitch contour, articulation, or timbre:" he does abstraction of the gesture that leads to the music and concentrates on the artistic value that he wants to address. Fine musicians are very sensitive to the sound response of a particular instrument at which they are proficient. With electronic instruments, they usually agree on the expressivity of controls as more important than the reproduction of waveforms.

Unlike with acoustic instruments, digital controllers are disconnected from the sound generating mechanisms that they are virtually attached to, allowing totally new forms of instruments. However, one of the main challenges when designing these instruments is to reattach these two modules in an intuitive and meaningful manner. It is a hard research topic that encourages much exploration. In the case of an expert instrument such as the violin, the controlling mechanism — the action of bowing — is intuitively correlated to the sound that is generated — the vibration of the string is amplified by the body of the instrument, which produces the sound. The design of sound controllers for skilled musicians should not underestimate that traditional tight relationship between the musician and his/her instrument.

Specially designed commercial controllers with embedded sensors already exist, e.g., Yamaha WX5 wind MIDI controller. Some devices have been developed that pick up the sound of an electric instrument and convert it to MIDI, e.g., Roland GR-1 pitch-to-MIDI converter. Roland has also produced a guitar synthesizer module (GR-33) that first tracks pitch and loudness. It then controls an internal synth but also adds an "intelligent" harmony feature that can generate complex tones from the guitar signal. All current systems present weaknesses either on the quality of sounds they can generate or on the controls they offer over the synthesis. They are also instrument specific.

1.1 What is Timbre?

Timbre is defined as the particular quality of a sound that distinguishes it from other sounds of the same pitch and loudness. This definition addresses the hard problem of characterizing the notion of timbre. We certainly lack the vocabulary for describing it. It may be rough, sharp, thin, bright, etc. We find better cues in the observation of the acoustic signal.

One important timbral factor is certainly the harmonic structure — the *(in)harmonicity* [Handel, 1989] — how equally spaced the partials are (see figure 2.1 in section 2.1). Into that category, and closely related, falls the notion of *periodicity*. We consider pitched musical instruments periodic as pitch is rooted in the notion of periodicity (20–20KHz) in some form. Another factor is the average spectral shape or how rapidly does the energy fall off as you go into the higher partials. We approximate it by using the spectral centroid (see equation 2.12), a sort of center of gravity for spectrum. A third but important one is the formant structure: the "bumpiness" of the spectrum. This for example allows to differentiate voice sounds such as "aaaaa" and "eeeee." And finally, an important timbral aspect is the spectrum variations in time, especially at the attack and decay [Risset, 1969, Grey, 1978, Wessel, 1979, Risset and Mathews, 1981]. A lot of timbral information is, for instance, contained in the onset of a note when the periodic partials were born and before they settle. Timbre is difficult to fully describe with few numbers of controls, either for compression [Verma, 1999], analysis, or musical synthesis applications [Masri, 1996, Jensen, 1999].

Different techniques are used to describe those timbral parameters. For example *Linear Predictive Coding* (LPC) [Makhoul, 1975] is a method that efficiently describes a formant structure and is widely used for speech synthesis. It is implemented as a filter and is excited by white noise (to simulate unvoiced phonemes) or a pulsed source whose repetition rate is the desired pitch (to simulate voiced phonemes).

At IRCAM, Rodet et al. have implemented a singing voice model entitled CHANT [Rodet et al., 1984] based on a modified synthesis method termed FOF (Forme d'Onde Formantique[1]). Each formant filter is implemented

[1]Formant Wave Functions.

separately and phase-aligned to avoid interference. Each pitch period impulse is individually filtered and responses are then time-aligned and summed to generate the full sound.

Some other techniques also allow one to modify some aspects of timbre, and for example take some audio parameters of one source to influence another. Appeared a long time after the original analog vocoder, the *phase vocoder* [Portnoff, 1976, Dolson, 1986, Roads, 1995] is a good example of spectrum-domain manipulation of sound. The vocoder is an electronic signal processor consisting of a bank of filters spaced across the frequency band of interest. A voice signal is analyzed by the filter bank in real time, and the output applied to a voltage-controlled filter bank or an oscillator bank to produce a distorted reproduction of the original. In any case, the phase vocoder inevitably involves modification of the analysis before resynthesis, resulting in a musical transformation that maintains a sense of the identity of the source. Two analyzed signals can be multiplied in the spectrum domain, i.e., each point in spectrum A are multiplied by each corresponding point in spectrum B. The result, named *cross-synthesis* sounds like a source sound (e.g. a voice) controlling another sound (e.g. a synthesizer sound). The effect can be heard in many popular music tracks.

1.2 Synthesis techniques

A sound synthesis technique maps time-varying musical control information into sound. Each different synthesis method can be evaluated not only in terms of the class of sounds it is able to produce, but also in terms of the musical control it affords the musician. However, certain fundamental ideas for sound synthesis are shared by multiple techniques. The next few paragraphs recall the different main classes of digital synthesis techniques since Mathews' first instrument[2].

[2]In 1970, Mathews pioneered the GROOVE system (Generated Real-time Output Operations on Voltage-controlled Equipment), the first fully developed hybrid system for music synthesis, utilizing a Honeywell DDP-224 computer with a simple cathode ray tube display, disk and tape storage devices. The synthesizer generated sounds via an interface for analog devices and two 12-bit D/A converters. Input devices consisted of a "qwerty" keyboard, a 24-note keyboard, four rotary knobs, and a three dimensional rotary joystick.

1.2.1 Physical modeling

Physical models reconstruct the acoustical behavior of the instruments by simulating their mechanical properties. They retain the natural expressiveness of the acoustic instrument and may sound very good, but they are usually CPU intensive and are very limited in the range of sounds they can generate with one model. Each one requires a lot of knowledge on the actual acoustics and physics of the instrument [Smith, 1992, Rodet and Vergez, 1996]. In the end, the mathematical approximations are such that it becomes difficult to distinguish for instance a beginner violin from a Stradivarius. The Yamaha VL1 is a good example of commercial physical modeling synthesizer.

1.2.2 Sampling

Sampling (or wavetable synthesis) in some ways contrasts with physical modeling. The basic principle is to record and store large databases of waveforms [Massie, 1998]. It is able to provide high sound accuracy, but offers very little flexibility and expressive freedom. It has been predominant in modern commercial synthesizers (e.g. Korg M1). There are a few reasons for its popularity: sampling requires not much more than the acoustic instrument, a player, and a recording device. As digital archiving has become very cheap, many libraries of sounds are easily available. Finally, the technique is very well suited to keyboards that have very few controls, i.e., note on/off, pitch, and velocity.

1.2.3 Abstract modeling

Abstract modeling attempts to provide musically useful parameters in an abstract formula. This large group of synthesis techniques (e.g. FM [Chowning, 1973], granular [Roads, 1995], waveshaping [Risset, 1969, Arfib, 1979, LeBrun, 1979], scanned [Verplank et al., 2000]) is not derived from any physical laws but arbitrarily aims to reconstruct complex dynamic spectra. Sometimes computationally cheap, these are in any case good at creating new sounds. A good example of successful commercial synthesizer

that implements FM synthesis is the Yamaha DX7.

1.2.4 Spectral modeling

Widely accepted as a very powerful sound synthesis technique, *Spectral modeling* (or additive synthesis) attempts to describe the sound as it is perceived by the ear. Like sampling, it only relies on the original sound recording. Unlike physical modeling, it does not depend on the physical properties of the instrument but yet remains flexible and sounds natural [Makhoul, 1975, Lansky and Steiglitz, 1981, Serra, 1989, Serra and Smith, 1990, Depalle et al., 1994].

In most pitched instruments (e.g., violin, trumpet, or piano) the sound signal is almost entirely described with a finite number of sinusoidal functions (i.e. harmonic partials) [Chaudhary, 2001]. However, there is also a noisy component left (e.g., loud in flute, saxophone, or pipe organ) that is usually better described stochastically with colored noise [Goodwin, 1996]. Moreover, the sound quality is scalable and depends on the number of oscillators being used. Unlike most methods, it allows spectrally-based effects such as sound morphing.

Conceptually appealing, the main difficulty remains in musically manipulating its high dimentionality of control parameters. This thesis presents a solution to dynamically and expressively control additive synthesis. The method is also not computationally expensive and appears to be an efficient tool for compressing an audio stream (see section 3.5).

1.3 Hyperinstruments

This thesis was first motivated by the need to develop a novel synthesis technique for the new generation of *hyperviolin*, an augmented instrument from the Hyperinstruments group at the Media Lab.

We define *hyperinstrument* [Machover, 1991, Machover, 1992] as an ex-

tended more-or-less traditional instrument. It takes musical performance data (audio and gestures) in some form, processes and interprets it through analysis software, and generates a musical result. The whole chain of events preferably happens in real-time so it can be used during a performance. It is considered "virtual" since its meaning and functionality is entirely reconfigurable in software at any time. It can either feature a totally new interface that is accessible to the novice, such as the "Sensor Chair," [Paradiso and Gershenfeld, 1997] the "Singing Tree" [Oliver, 1997], or the "Melody Easel" from the *Brain Opera* [Paradiso, 1999] or it can make use of already existing musical interfaces such as the series of *hyperstrings*.

Conceptually, a major difficulty with digitally enhanced instruments comes from the choice of mappings between inputs and outputs [Sparacino, 2001]. Indeed, there is no "true" mapping between a gesture and a synthesized result: with most traditional instruments, the sound output is generated from a non-linear interconnection of complex gesture inputs [Schoner, 2000]. However, some intuitive mappings are sometimes fairly good approximations, e.g., bow pressure as volume.

Schoner in [Schoner et al., 1998] models the sound output of a violin from the gesture data captured on a muted instrument. In this digital version of the violin, a network was previously trained to learn the mapping from physical gesture input to audio parameters. During synthesis, the network generates appropriate audio, given new input. The gesture input (bow position, bow pressure, finger position etc.) is measured with a complex sensing hardware setup.

My approach differs from Schoner's in many ways: the input is an acoustic audio stream instead of measured gestures. The system allows for modeling of any timbre, only from recordings, and does not require any additional hardware. It also allows arbitrary timbre control and sound morphing from a single sound source. Thus, I believe there is a strong artistic value to this technique.

Obviously, in the case of the violin, the interface is such that it applies more to sound models of string instruments, but also works well with voices, brass, or other non-discretely pitched instruments. There would not be anything wrong with synthesizing a piano sound from a violin input, but the

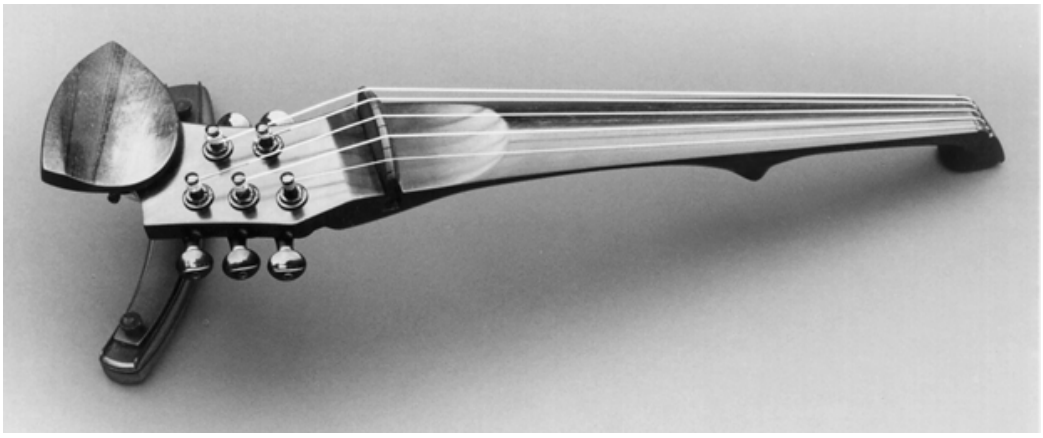

Figure 1.1: Our controller: a five string Jensen electric violin.

result would not sound anything like a piano. In fact, we can see it as a hybrid sound (see section 3.2) in between a violin — the controller — and a piano — the sound model.

The development of expanded instruments was started by Tod Machover at the Media Lab in 1986 to "convey complex musical experiences in a simple and direct way." They were designed to allow the performer's normal playing technique and interpretive skills to shape and control computer extensions of the instrument, thus combining the warmth and "personality" of human performance with the precision and clarity of digital technology.

Previous examples of these instruments include the hyperkeyboard and hyperpercussion that were used for the opera VALIS[3], the hypercello, hyperviola, and hyperviolin, of the *Hyperstring Trilogy*[4], and have been used by some of the world's foremost musicians such as Yo-Yo Ma. A combination of gesture measurements via sensors (e.g., wrist angle, bow position), sound measurements (e.g., pitch tracking, timbre analysis [Hong, 1992]), and score follower were used to monitor and eventually "understand" nuances of the musical performance, so that the musician's interpretation and feeling

[3]By composer Tod Machover (1986-87, revised 1988), Bridge Records: BCD 9007

[4]Begin Again Again..., Song of Penance, and Forever and Ever, by composer Tod Machover (1991-93)

could lead to an enhanced and expanded performance — usually by generating extra layers of MIDI orchestration, controlling sound effects, or shaping triggered sound samples.

The new *hyperviolin* is an attempt to extend the violin possibilities in a more subtle, yet musical manner. It is an expert performance instrument that drives multi-channel audio analysis software and embedded wireless hardware technology. It aims to give extra power and finesse to a virtuosic violinist. It allows for quick, intuitive, and creative artistic results. This thesis describes the analysis/synthesis technique that was specifically developed and applied to the hyperviolin instrument. Although its "interface" is identical to a standard violin (see figure 1.1[5]), the sound output is different, and creatively controllable. The new hyperviolin is designed to respond more closely and intuitively to the player's music and to be fully autonomous, allowing for improvisation.

1.4 A Transparent Controller

Figure 1.2 shows a traditional synthesis system where the musical gesture is captured from a MIDI interface, analyzed and interpreted before synthesis [Sapir, 2000]. The haptic feedback is different from that of a real instrument and the auditory feedback may not necessarily correlate intuitively with the haptic feedback. As appropriate gesture sensing and interpretation is in the case of most instruments very difficult [Paradiso, 1997], few digital versions of acoustic instruments are available today that come close to matching the virtuosic capabilities of the originals.

Since the valuable musical information is contained in the sound that the audience — and player — perceives, our system aims to control sound synthesis from the music produced rather than the input gesture on the physical instrument. We hope to overcome the hard problems of gesture interpretation and of simulating the physics of a complex vibrating acoustic system (see *Physical Modeling*).

[5]Photography reproduction coordially authorized by Eric Jensen.

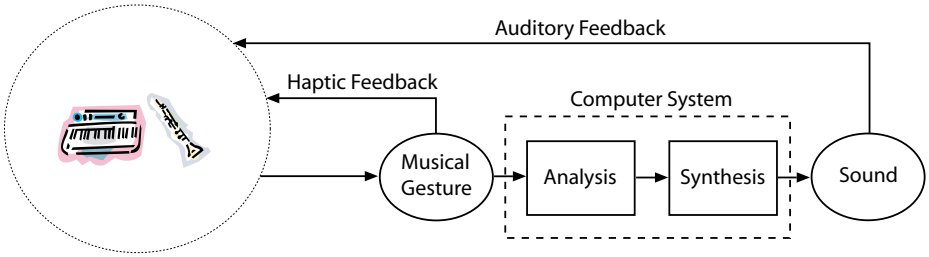

Figure 1.2: A traditional digital synthesis system. Controller instruments are specially designed MIDI devices. The computer system converts musical gestures into synthesized sounds.

Figure 1.3 shows our synthesis system. It applies to arbitrary acoustic instruments and there is no gesture sensing and interpretation. The haptic feedback feels natural to the musician. Sound 2 features the same perceptual characteristics as sound 1, thus the auditory feedback is meaningful and correlates well with the haptic feedback.

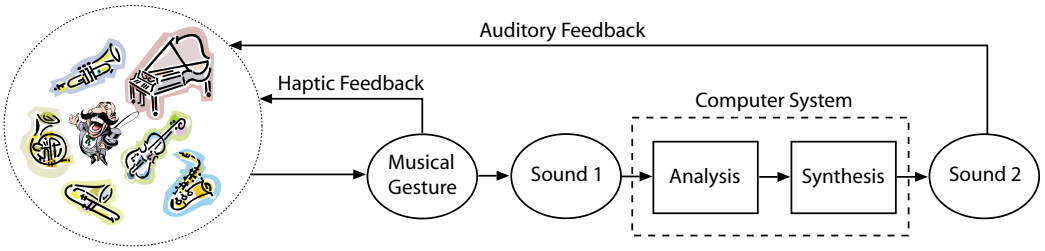

Figure 1.3: Our synthesis system. Controllers are arbitrary acoustic or electric instruments. The computer system converts the sound from the controller instrument into a synthesized sound with identical perceptual qualities.

Both systems can either run in real time or be processed offline for post-production. In the traditional system, the musician needs to adapt to a new haptic and auditory feedback mechanism at recording. At post-production, any change in the computer system (e.g. a new sound selection) may not reflect the musician's exact musical intent anymore. In our system, the musician does not need to adapt to a new feedback mechanism, and whatever the modifications in the computer system, the musical intent is preserved.

We can see our system as a *transparent* analysis/synthesis layer in between the instrument sound output and the musician's ear. That layer is implemented on a computer system that takes in the audio stream coming from an acoustic — possibly muted — instrument, and puts out a second audio stream with identical musical content but with a different timbre. This computer system is the "hyper" of the professional category of hyperinstruments that we are interested in, such as the hyperviolin (see section 1.3).

From the original audio stream, we pull out perceptually relevant features that the player controls. These are for instance continuous *pitch*, *loudness*, and *brightness*[6].

"Sound" considered as either a physical or a perceptual phenomenon are not the same concept. Auditory perceptions and physically measurable properties of the sound wave need to be correlated significantly. Hence, physical attributes such as frequency and amplitude are kept distinct from perceptual correlates such as pitch and loudness [Sethares, 1998].

- *Pitch* is the perceptual correlate of the *frequency* of a periodic waveform.
- *Loudness* is the perceptual correlate of the *amplitude*.
- *Brightness* is the perceptual correlate of the *spectral centroid*.

We choose to model what is in a musical sound and that is *not* the perceptual features mentioned above: we call it *timbre model*.

Almost no work has been done on perceptually-controlled sound synthesis. The field of sound and music perception is fairly new and is still not very well understood [Cook, 2001]. Works from Max Mathews, Jean-claude Risset, Barry Vercoe, David Wessel, or more recently Eric Scheirer [Scheirer, 2000], show that there is a need for smart algorithms capable of emulating, predicting and characterizing the real sound world into digital machines.

Simulating with algorithms that describe real-world non-linear dynamic systems is a difficult task of great interest to the Artificial Intelligence com-

[6] violinists increase brightness of their sound by bowing closer to the bridge.

munity. Such algorithms are needed for the system we present here. Although the required computing power is important, it is finally manageable on today's desktop computers.

1.5 Previous Work

While interactive and electronic music has become more accesible and popular in the last decade [Rowe, 1992, Rowe, 2001, Winkler, 1998, Boulanger, 2000, Dodge and Jerse, 1997, Miranda, 1998, Roads, 1995], there is still little research on augmented acoustic instruments (see section 1.3 *Hyperinstruments*), and even less on specifically designed synthesis techniques for non-discretely pitched instruments.

Camille Goudeseune [Goudeseune, 1999, Goudeseune et al., 2001] uses an electric violin as a gesture-input device. He measures the violin position and orientation using a SpacePad motion tracker and the relative position of bow and violin with magnetic sensors. These are used for spatialization of the sound output. He also measures pitch and loudness of the instrument to control various synthesis models that include FM synthesis, the physical model of a clarinet, a high-dimensional interpolation of four different instruments, simulating an orchestra, a "Hammond organ" additive synthesis model and a singing voice using the vocal model CHANT from IRCAM (see section *What is Timbre?*).

Dan Trueman [Trueman, 1999] has also explored various ways of expanding the violin possibilities. He mixes sound spatialization techniques, using spherical speakers (*SenSAs*), sensor-speaker arrays (*BoSSA*) [Trueman and Cook, 1999], and various synthesis techniques [Trueman et al., 2000]. He especially developed *PeRColate*, a collection of synthesis, signal processing and image processing externals for Max/MSP based on the Synthesis Toolkit (STK) by Perry Cook (Princeton) and Gary Scavone (Stanford CCRMA) [Cook and Scavone, 2001].

Similar interesting work by cellist Chris Chafe, keyboard player Richard Teitelbaum, jazz trumpetist Dexter Morrill, reeds and piano player Anthony Braxton or jazz trombone player George Lewis should also be mentioned.

In particular, George Lewis' approach is to augment the music in an improvisatory manner. For example, he uses a pitch-to-MIDI converter that feeds a probabilistic software algorithm designed to improvise with him. His system is driven from the audio and does not use pre-composed sequences.

Significant work was done on Analysis/Transformation/Synthesis of sound using a sinusoidal decomposition. It was started with the LPC approach (see section 1.1) of Makhoul [Makhoul, 1975] and Lansky [Lansky and Steiglitz, 1981], then was refined by Serra who separated periodic from non-periodic signals. Serra has developed a set of techniques and software implementations for the analysis, transformation and synthesis of musical sounds entitled *Spectral Modeling Synthesis* [Serra and Smith, 1990]. SMS aims to get general and musically meaningful sound representations based on analysis, from which musical parameters might be manipulated while maintaining high quality sound. The techniques are used for synthesis, processing and coding applications and other music related problems such as sound source separation, musical acoustics, music perception, or performance analysis.

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 uncommon.

Métois introduces the synthesis technique *Psymbesis*, for Pitch Synchronous Embedding Synthesis [Métois, 1996]. He defines a vector of perceptual control parameters including pitch, loudness, noisiness and brightness. He clusters this data in a control space and assigns periods of sound to each cluster. Each cluster period (cycle) 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 cycles happens smoothly. The periods are re-sampled at the desired pitch and adjusted for the desired loudness. In the end, the synthesis engine is a sort of generalized wavetable where the

"index" of the table is dynamically adjusted in a lag space instead of being forced by an external counter. Our system also uses perceptual controls as input and a statistical approach for modeling the data, but differs in the characterization of the sound and the synthesis technique. We characterize the sound in the spectrum domain rather than the time domain and synthesize the sound using additive synthesis. Métois has experimented with cello and voice models. Only 10 seconds of sound recordings were used to train a model (typically a sequence of a few notes) and the system was not implemented in real time.

Wessel et al. presented a synthesis model which inspired our approach [Wessel et al., 1998]. 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 a feed-forward network, whereas the spectral parameters serve as outputs. The network is trained to represent and predict a specific instrument (examples with wind instruments and the singing voice were shown). At synthesis, a new set of inputs are given to the network that outputs the corresponding spectral parameters. The sound result is generated using additive synthesis. 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 is more compact and provides smoother output, while the memory-based models are more flexible — easier to modify and easier to use in a creative context [Wessel et al., 1998]. Limited sound data was used for training (typically a 10-second musical phrase or a few glissandi). In the case of cross-synthesis between two instruments for instance, the same phrase was played with both instruments. Given a recorded sequence of perceptual inputs, the system could synthesize in real time but was not implemented to be flexible and used with a new real-time input. Our system uses a different modeling technique, comparable to Métois's and is implemented to be flexible and easy to use in a real musical context (see *Max/MSP Implementation* and *Applications*).

Schoner et al. used Cluster-Weighted Modeling (see section *Cluster-Weighted Modeling*) to predict a spectral sound representation given physical input to the instrument [Schoner et al., 1998]. While the target data was similar to the data used in [Wessel et al., 1998], 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 thesis combines the efficiency of Cluster-Weighted Modeling with spectral synthesis and the idea of a perceptual control as feature vector. The following chapter introduces this new technique for modeling and controlling timbre. It describes an expressive sound synthesis engine driven only by continuously changing perceptual parameters, i.e., pitch, loudness, and brightness, extracted in the audio signal of an acoustic instrument.

Chapter 2

Perceptual Synthesis Engine

This chapter is about the functionality of the Perceptual Synthesis Engine. First, the analysis, modeling, prediction, and synthesis steps are described, then a novel approach for noise synthesis. The Cluster-Weighted Modeling algorithm that was developed by Bernd Schoner and Neil Gershenfeld at the Media Lab is reviewed. Finally, the full system, real-time implementation in the Max/MSP environment is presented.

2.1 Timbre Analysis and Modeling

Underlying this approach to timbre modeling are two fundamental assumptions:

1. It is assumed that the timbre of a musical signal is characterized by the instantaneous power spectrum of its sound output.
2. It is assumed 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 can conclude that a unique spectral representation of a sound can be inferred given perceptual sound data and a timbre model. In this approach, both perceptual and spectral representations are estimated from recorded data. Then, the latter given the former is predicted.

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 24 ms at intervals of 12 ms. 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 (= 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, not an output, the spectral vector contains $2L - 1$ components ordered as $[A_0, M_1, A_1, M_2, A_2, \ldots, M_{L-1}, A_{L-1}]$ 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 I first perform a rough estimation using the Cepstrum transformation [Noll, 1967] or an autocorrelation method [Rabiner, 1970] 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. The peaks of the spectrum and the bins they fall into are identified. 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. Therefore, the instantaneous frequency estimation of the bins of highest energy is used to obtain a much higher resolution with little extra computation [Métois, 1996].

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

$$X(k) = \sum_{n=0}^{N-1} s(n)e^{-jwnk} \tag{2.1}$$

with

$$\begin{aligned} w &= \frac{2\pi}{N} \\ k &= 0, 1, \ldots, N-1 \end{aligned}$$

The estimate for bin k's instantaneous frequency is:

$$F_{\text{inst}}(k) = F_s \left(\frac{k}{N} + \frac{1}{2\pi} \text{Arg} \left[\frac{A}{B} \right] \right) \tag{2.2}$$

where

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

The full derivation for this expression can be find in the *Appendix A*, page 62.

Given the spectral decomposition we can easily extract pitch as the frequency of the fundamental component. The author is aware that this is an approximation that may not necessarily be accurate for all instruments but it meets the requirements of our study and application. Furthermore, instantaneous loudness is extracted 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 [Wessel, 1979]. 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.

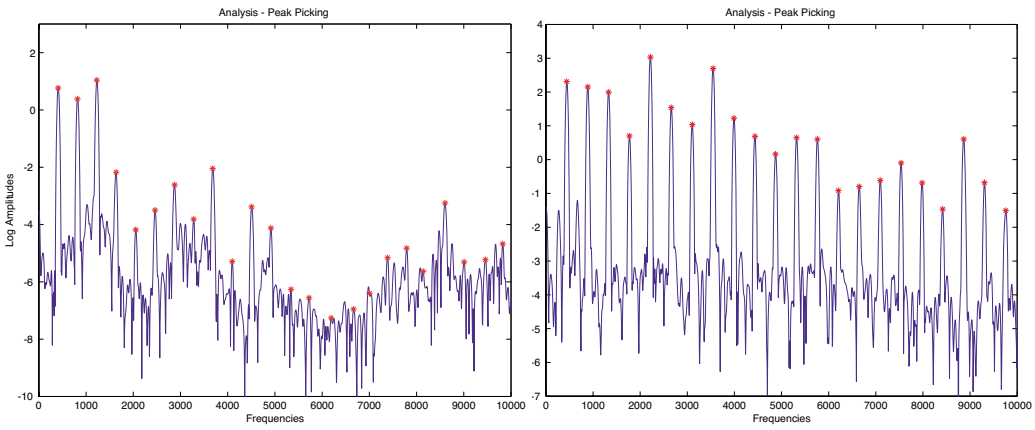

Figure 2.1: Spectrum of a singing voice (*left*) and the Stradivarius violin (*right*) — 24 ms frame of data. The stars indicate the harmonic peaks of the spectrum as found by the peak tracking algorithm.

To summarize, in this section we have seen a technique to parameterize and model an arbitrary acoustic instrument from the analysis of its recording. 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 20 to 80-dimensional output vector containing frequency and amplitude values of 10 to 40 harmonic partials. This data is used to train a feed-forward input-output network to predict frequencies and amplitudes (see figure 2.3 - *top* and section *Cluster-Weighted Modeling*). We have, in some ways, reduced a complex timbre description to a black box: the timbre model. It has generated itself from training[1] without making any particular assumption on the structure of the sound or the instrument to begin with.

[1]There is no simple and general mathematical description of an arbitrary timbre for an acoustic instrument, so a training-based approach seems reasonable to the author.

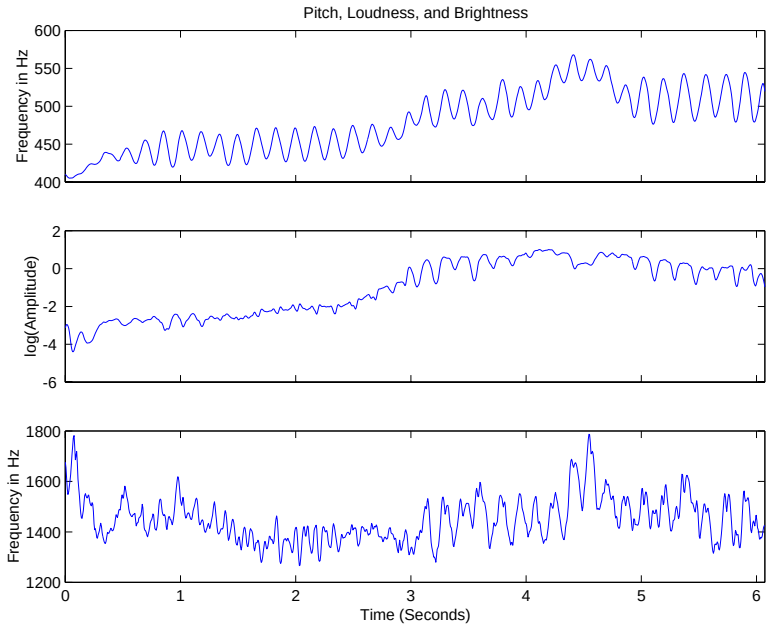

Figure 2.2: Typical perceptual-feature curves for a female singing voice.

2.2 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 — 10 to 40 sinusoids depending on what level of sound quality is desired (see figure 2.3 - *bottom*).

The specific model consists of three input parameters (pitch, loudness, and brightness), and $2L$ (= 20 to 80) 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 12 ms. If the model is based on L sinu-

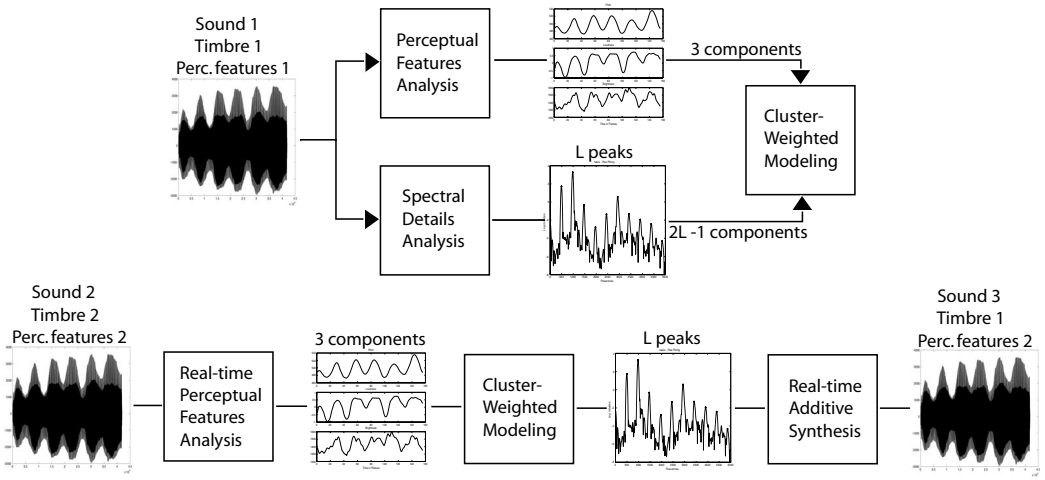

Figure 2.3: *top:* Timbre analysis and modeling using cluster-weighted modeling. *bottom:* New analysis, prediction and synthesis of a new sound with modeled timbre. Ripples in pitch represent vibrato and ripples in loudness represent tremolo.

soidal parameters, the predictor generates $2L-1$ output values consisting of $[A_0, M_1, A_1, M_2, A_2, \ldots, M_{L-1}, A_{L-1}]$ 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:

$$d(n) = \sum_{l=1}^{L} A_l \cos(\omega_l n + \Phi_l) \tag{2.3}$$

with

$$\omega_l = 2\pi M_l F_0$$

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, a stochastic process will be combined with the deterministic component $d(n)$ of expression (2.3) to create a more accurate timbre

(see *Noise Analysis/Synthesis*). The full signal model $s(n)$ then becomes:

$$s(n) = d(n) + r(n) \tag{2.4}$$

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 *Applications*).

2.3 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 [Serra, 1997, Rodet, 1997]. The noise components are particularly important at the onsets of notes (see figure 2.4), 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 [Goodwin, 1996]. Here I present a novel approach to noise modeling by means of a polynomial expansion, more generally known as *linear least squares*.

In general, the noise characteristics of a signal are captured in the shape of the power-spectrum of its non-harmonic components. Figure 2.5 shows examples of noise spectra for the singing voice and the clarinet. The residue is obtained by subtracting the power-spectrum of the deterministic signal $d(n)$ from the power spectrum of the original signal $s(n)$ as described in expression (2.4). I extract this residual spectrum for each time frame (e.g., frames of 24 ms at a rate of about 12 ms) and approximate the spectral function (in a logarithmic scale) by using polynomial basis functions f_k.

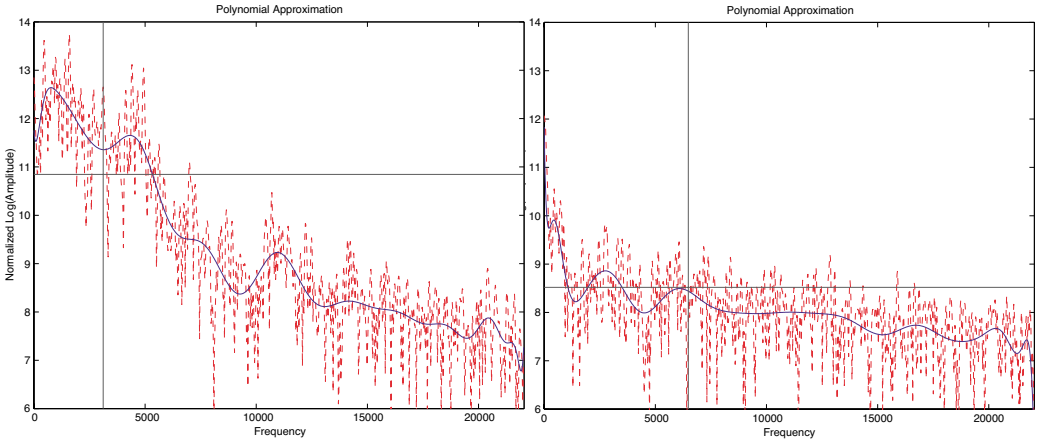

Figure 2.4: Typical noise spectrum of the violin (24 ms FFT) approximated with a polynomial function (25 basis functions) at the onset of a new note (*top*) and at decay (*bottom*). Loudness and brightness are also displayed, respectively with a horizontal and vertical line.

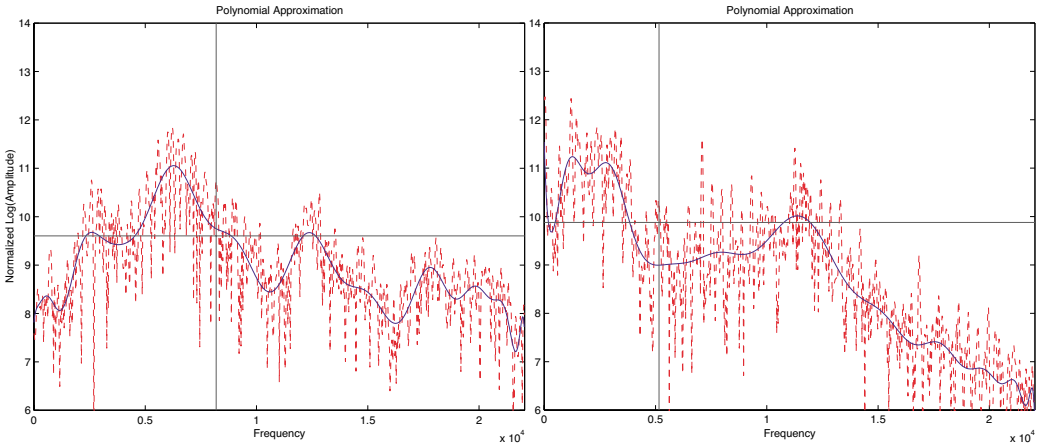

Figure 2.5: Typical noise spectrum of the singing voice (*left*) and the clarinet (*right*) with a 24 ms FFT, approximated with a polynomial function (25 basis functions). Loudness and brightness are also displayed, respectively with a horizontal and vertical line.

The approximation is of the form:

$$y(\omega) = \sum_{k=0}^{K} a_k f_k(\omega) \tag{2.5}$$

where $f_k(\omega)$ are the polynomial functions of ω up to a given polynomial order K. ω is a vector that contains the frequencies of the spectrum.

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. Up to 30 basis functions and coefficients are used. The coefficients a_i form the output vector of a predictor model, which, in synthesis, interpolates between the coefficients of different noise spectra. They are determined by means of a simple matrix inversion generating the best solution in the least squares sense:

$$\mathbf{a} = \mathbf{B}^{-1} \cdot \mathbf{c} \tag{2.6}$$

with

$$\begin{aligned} [\mathbf{B}]_{ij} &= \langle f_i(\omega) \cdot f_j(\omega) \rangle \\ [\mathbf{c}]_j &= \langle y(\omega) \cdot f_j(\omega) \rangle \end{aligned}$$

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

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. The output vector contains the polynomial coefficients a_i.

During synthesis, the predictor model generates polynomial coefficients, which are used to reconstruct the noise spectrum for every frame (i.e., at a rate of 12 ms). 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. Computational speed is

generally not an issue when executed on a state of the art PC or Macintosh computer.

This parameterization of sound is comparable to Serra's *Deterministic plus Stochastic Decomposition* [Serra, 1989] implemented in *Spectral Modeling Synthesis* (SMS) [Serra and Smith, 1990]. However, our system removes the temporal axis to dynamically control musical features by inputing new pitch, loudness, brightness, and noisiness envelope functions in real time.

2.4 Cluster-Weighted Modeling

The nonlinear mapping from the feature vector onto the harmonic target vector is approximated using the general inference framework, Cluster-Weighted Modeling. CWM was first introduced by Neil Gershenfeld in [Gershenfeld, 1999] and was fully developped by Bernd Schoner in [Schoner, 2000]. This section reviews the functionalities of CWM that fulfill our needs.

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.

2.4.1 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 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. For the Perceptual Synthesis Engine application I 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 [Jordan and Jacobs, 1994] 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 (see figure 2.6).

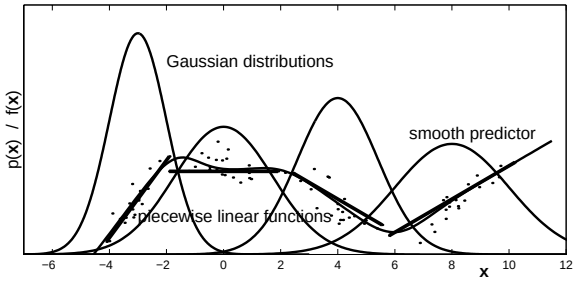

Figure 2.6: One dimensional function approximation with locally linear models weighted by Gaussian kernels.

This technique typically starts 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). Consider the joint input-output set $\{\mathbf{y}_n, \mathbf{x}_n\}_{n=1}^{N}$, and the goal is 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.

$$\begin{aligned} p(\mathbf{y},\mathbf{x}) &= \sum_{k=1}^{K} p(\mathbf{y},\mathbf{x},c_k) \\ &= \sum_{k=1}^{K} p(\mathbf{y}|\mathbf{x},c_k)\ p(\mathbf{x}|c_k)\ p(c_k) \end{aligned} \tag{2.7}$$

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

$$p(\mathbf{x}|c_k) = \frac{|\mathbf{P}_k^{-1}|^{1/2}}{(2\pi)^{D/2}} e^{-(\mathbf{x}-\mathbf{m}_k)^T\cdot\mathbf{P}_k^{-1}\cdot(\mathbf{x}-\mathbf{m}_k)/2} \tag{2.8}$$

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:

$$p(\mathbf{y}|\mathbf{x},c_k) = \frac{|\mathbf{P}_{k,y}^{-1}|^{1/2}}{(2\pi)^{D_y/2}} e^{-(\mathbf{y}-\mathbf{f}(\mathbf{x},\mathbf{a}_k))^T\cdot\mathbf{P}_{k,y}^{-1}\cdot(\mathbf{y}-\mathbf{f}(\mathbf{x},\mathbf{a}_k))/2} \tag{2.9}$$

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 (2.8) and (2.9) the off-diagonal terms of the covariance matrices can be dropped if needed.

Expression (2.9) is easily understood considering the conditional forecast of $\mathbf{y}$ given $\mathbf{x}$:

$$\begin{aligned} \langle \mathbf{y}|\mathbf{x}\rangle &= \int \mathbf{y}\ p(\mathbf{y}|\mathbf{x})\ d\mathbf{y} \\ &= \frac{\sum_{k=1}^{K} \mathbf{f}(\mathbf{x},\mathbf{a}_k)\ p(\mathbf{x}|c_k)\ p(c_k)}{\sum_{k=1}^{K} p(\mathbf{x}|c_k)\ p(c_k)} \end{aligned} \tag{2.10}$$

Expression (2.10) 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.

Figures 2.7 and 2.8 depict two types of data analysis in a two-dimensional space (pitch and loudness) for a short segment of audio (figure 2.7) and a full

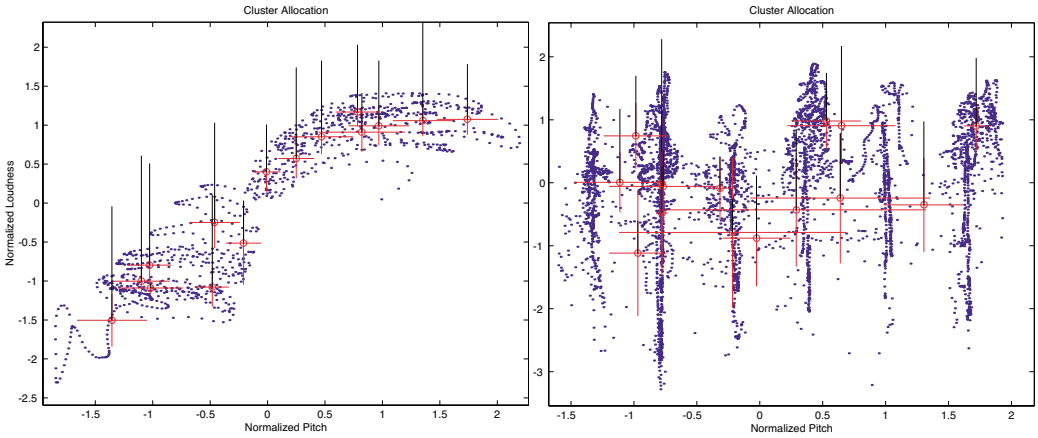

Figure 2.7: Selected data and cluster allocation. The vertical and horizontal lines represent the weight and variances of each cluster. *Left:* Example with the female singing voice data: 1047 points (as sung in figure 2.2) and 15 clusters in a two-dimensional input space. *Right:* Example with 4622 Stradivarius data points and 15 clusters (6 notes).

model (figure 2.8). The female singing voice and the Stradivarius violin cases are compared. While the first case shows non-structured clouds of points, the second case shows clear patterns — vertical narrow clouds displaying notes played by the violinist. While the singer mostly sung free melodies, the violinist recording was more closely directed. Note the singer tendency to sing quieter at lower pitch and louder at higher pitch.

2.4.2 Model Estimation

The model parameters are found in an iterative search which uses two estimators combined in a joint update: CWM uses 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 [Dempster et al., 1977]. 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

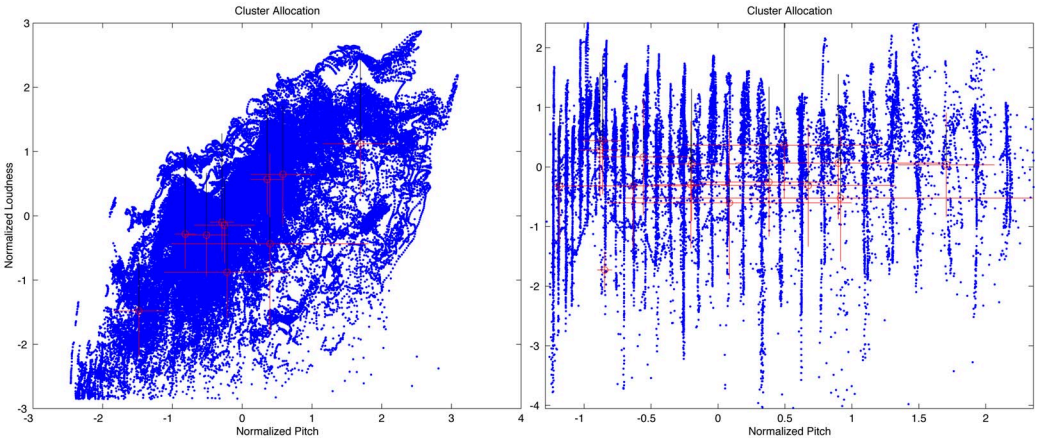

Figure 2.8: Full model data and cluster allocation. The vertical and horizontal lines represent the weight and variances of each cluster. *Left:* Example with the female singing voice data: 89950 points and 10 clusters in a two-dimensional input space. *Right:* Example with 19823 Stradivarius data points and 15 clusters.

were known, model estimation would be easy, because we would only need to maximize a parameterized distribution. Yet, since we do not 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 [Schoner et al., 1998] and [Gershenfeld et al., 1999].

In this application, the only 3 parameters we deal with are:

1. Number of clusters.
2. Number of iterations of the EM algorithm (see section 2.4.2).
3. Polynomial order of the local model used by the cluster.

2.5 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[2] 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* of the EM algorithm, and *polynomialOrder* of the local model used by the cluster. 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*. The two CWM objects are based on Bernd Schoner's original C implementation.

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

[2] All Max/MSP objects described here are available for download from the author's web site: `http://www.media.mit.edu/~tristan`

the coefficients $W_k(a_k)$ obtained from the interpolation of the Fletcher-Munson curves:

$$\text{loudness} = \sum_{k=2}^{\frac{N}{2}+1} \left(W_k(a_k) \cdot |a_k|^2 \right) \tag{2.11}$$

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:

$$\text{centroid} = \frac{\sum_{k=2}^{\frac{N}{2}+1} f_k \cdot a_k}{\sum_{k=2}^{\frac{N}{2}+1} a_k} \tag{2.12}$$

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:

$$\text{SFM}_{\text{dB}} = 10 \log 10 \left\{ \frac{\left(\prod_{b=1}^{b_t} E_b \right)^{\frac{1}{b_t}}}{\frac{1}{b_t} \sum_{b=1}^{b_t} E_b} \right\} \tag{2.13}$$

where b_t is the total number of critical bands on the signal.

In `analyzer~`, the spectrum is first decomposed 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:

$$\alpha = \min \left(\frac{\text{SFM}_{\text{dB}}}{\text{SFM}_{\text{dBmax}}}, 1 \right) \tag{2.14}$$

with $\text{SFM}_{\text{dBmax}} = -60\text{dB}$. 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]:

$$b = 13 \tan^{-1}\left(\frac{0.76 * f}{1000}\right) + 3.5 \tan^{-1}\left(\left(\frac{f}{7500}\right)^2\right) \tag{2.15}$$

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

`analyzer~` and other FFT-based objects enumerated below use a specifically audio-optimized real-FFT[3]. 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 [Press et al., 1992].

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 only 2% of CPU load on a 500 MHz Macintosh G4 with a 4096-point FFT overlapping by 85% (update rate is 12 ms). The `analyzer~` help file is displayed in figure A.1, page 65.

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~` written by Freed is used for real-time additive synthesis in the time domain [Freed and Jehan, 1999]. This object takes a list of frequencies and amplitudes and outputs the time-domain sum of their associated sinusoidal functions as described in equation 2.3.

The full implementation requires modest amounts of computing resources. A timbre-prediction model (i.e., the file loaded by `CWM-predict` and that

[3]Originally written at CNMAT by Adrian Freed, this efficient FFT was first ported to MSP by the author as a new external called `ffft~` standing for Fast-FFT.

contains the full model data) 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, up to a few hours of computation at 100% CPU load are needed for optimization of the model parameters. For example, the singing voice data showed in figure 2.8 - *left* represents the full pitch range of the singer (i.e., a little more than two octaves), and her full dynamic range, from pianissimo to fortissimo. 25 sinusoidal functions were used to synthesize the sound, and results were convincing. 89950 frames were analyzed. With 10 clusters, 15 iterations, and a linear local model, the training took about three hours of computation.

Chapter 3

Applications

There are many possible applications for the Perceptual Synthesis Engine. This chapter presents some general ones — i.e., timbre synthesis, cross-synthesis, morphing, pitch shifting and compression — as well as the ones that first motivated this research — Toy Symphony and the Bach Chaconne.

3.1 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 for each instrument were recorded, i.e., various pitches, dynamics, and playing styles. For instance, I 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, I used a directional Neumann microphone located about three feet above the violinist's head. I also used the McGill University Master Sample library for woodwind instruments.

Up to 100,000 data points (time frames) were used for each timbre model.

Typically, I have chosen from a few to up to twenty clusters to describe a more-or-less complex model: ten seemed to give reasonable results in most cases (e.g. voice model). Twenty iterations seem to give sufficient convergence. It is usually dependent on the number of clusters used: the more clusters there are, the more iterations are needed. Finally, it was discovered that a polynomial order of one (linear model) is best for local descriptions.

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 (i.e., smooth transition from one sample to another), sample looping (in order to maintain sustain), and pitch shift (see section *Pitch shifting*). 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 40 in different experiments. In [Chaudhary, 2001], Chaudhary has experimented with scalability. He demonstrates that harmonic sounds were often perceived as almost identical to the original when synthesized with hundreds or sometimes as little as 10 partials. For instance he shows that the suling — an indian flute — can be synthesized with only 10 partials and still sounds close to the original. He based the scheduler in his programming environment on that principle in order to save unnecessary computation.

Figure (3.1 - *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.

Figure (3.2 - *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 the original closely even though the input data was not part of the training data.

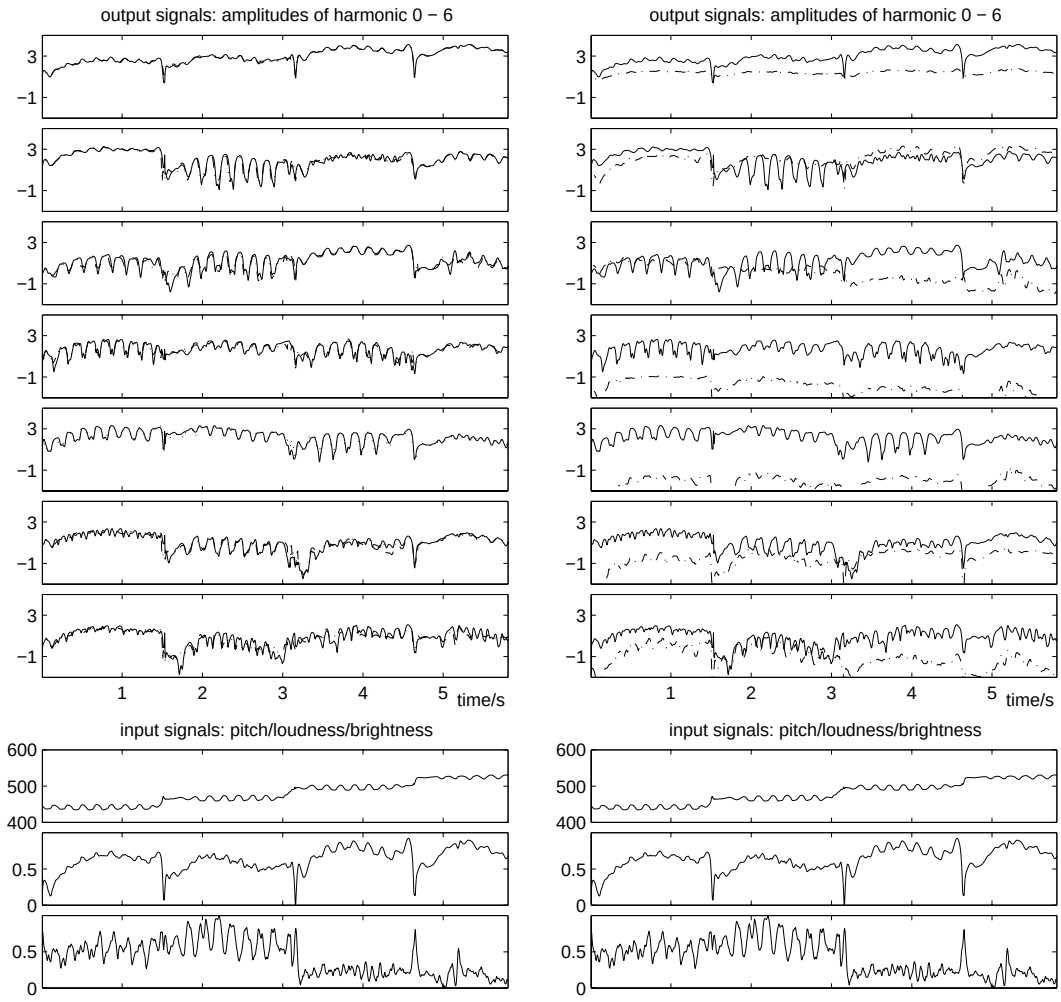

Figure 3.1: *left:* Violin-control input driving a violin model. The bottom figure represents the three perceptual inputs and the top figure represents the first seven harmonics extracted from the recorded signal (plain line) and predicted using the model (dashed line) - fundamental at the top, sixth 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). Ripples are due to vibrato and tremelo as showed by the pitch and loudness curves.

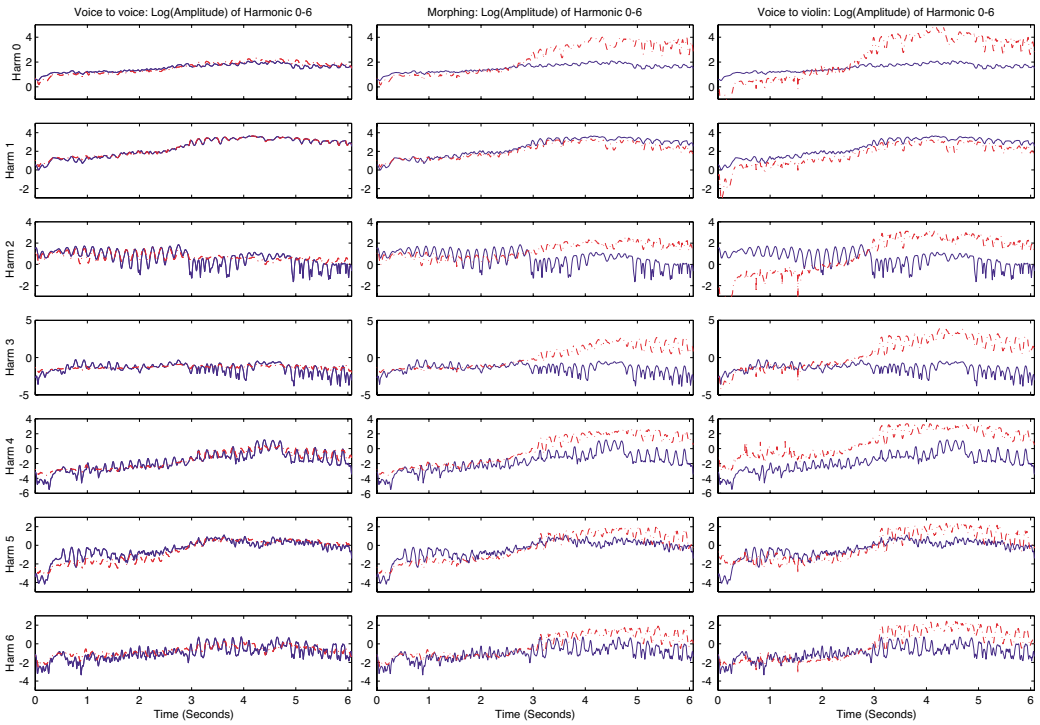

Figure 3.2: Three prediction results, each using the three perceptual inputs from the female singing voice of figure 2.2. Each figure represents the first seven harmonics extracted from the recorded signal (plain line) and predicted signal using the model (dashed line) - fundamental at the top, sixth 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).

3.2 Cross-synthesis

Instead of driving an instrument model with control data generated on the same instrument, controls and timbre models from different instruments can be mixed. For example, a singer controls the model of a Stradivarius violin or alternatively, 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 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.

As was pointed out earlier, loudness and brightness functions of the source are rescaled 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 (3.2 - *right*) shows an example of cross-synthesis between a female singing voice controller and a violin model. Comparing this figure to figure (3.2 - *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. Figure (3.1 - *right*) shows the opposite example of cross-synthesis between a violin controller and a female singing voice model. This time the violin input is turned into a singing voice.

In order to extend the output pitch range, we can interpolate between different voice models, for instance the model of a female and a male voice. The interpolation (see next section) 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. A good example of spectral morphing application can be find in [Depalle et al., 1994].

3.3 Morphing

The timbre modeling approach enables morphing between different core timbres. Because the structure parameterization is kept equal across different sounds, we can interpolate between parameterizations and models. In the application 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. A morphing

parameter α that weights the two spectra ($0 < \alpha < 1$) is introduced:

$$C_i(n) = C1_i(n) \cdot \alpha + C2_i(n) \cdot (1 - \alpha) \tag{3.1}$$

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.

α is specified offline or is changed dynamically. It can be controlled in several different ways:

1. The timbre[1] is chosen by the set of perceptual inputs. In that case the resulting sound is a complex structured varying timbre that evolves with the feature vector pitch, loudness, and brightness. Each core timbre can be represented by a zone in a 3D space defined by pitch, loudness and brightness. The zones can possibly overlap (morphing zones).

2. The timbre is chosen by additional physical controllers or sensors. One can define an hyperspace with each axis being represented by a physical gesture on a particular controller, e.g., pressure, displacement, rotation, etc. For example we can use a MIDI controller such as a volume pedal or a pressure-sensitive touch pad (e.g. MTC Express[2]) to modify the contribution of each timbre in real time. The timbre is then being controlled independently from the music played — unless sensors are embedded on the bow itself.

3. The timbre is music dependent. One may want to analyse the score, or some high level parameters in the music being played, e.g., density of notes, length of notes, type of attack, beat, rhythm, etc., and decide on the timbre from these.

Figure (3.2 - *middle*) shows an example of linear morphing between a female singing voice and a Stradivarius violin, using a voice input. The

[1] When there are multiple models running simultaneously, one can define the current timbre as the morphing between the core models. That is, there is an infinite number of combinations: e.g., 30% violin, 20% female voice and 50% trumpet is the description of a new timbre that would not sound exactly like any other core timbre.

[2] A mouse-pad sized multi-touch pressure-sensitive controller from Tactex Controls inc.

signal at first matches the predicted voice signal of example (3.2 - *left*) and in the end matches the predicted violin predicted signal of example (3.2 - *right*).

A Max patch displaying a simple morphing between two models is displayed in the *Appendix*, figure A.3 page 67.

3.4 Pitch shifting

Efficient and novel pitch-shifting applications are possible. Since the acoustic instrument is synthesized with pitch as a real-time control input, it is possible to change that control value and synthesize the new sound at a different pitch. For example, given an instrument and its model (e.g. a trombone), the instrument being played could be re-synthesized in real time at a new pitch — in which case, changing the pitch value only requires a floating multiplication: $newPitch = inputPitch * floatValue$. Thus, the advantage being the small amount of extra computation this manipulation demands: we do not need to process the sound output with a real-time pitch-shifting algorithm such as the one described in [Lent, 1989]. We are mostly limited by the pitch range of the original instrument. For example, if the target pitch value goes beyond the model limits, the system can extrapolate but may result in some unpredictable and undesirable sound output, i.e., some partials may become much louder than others, loosing the timbral quality of the original instrument. Kept in the limits, there is no additional artifact, even with a strong pitch shift like you can expect with traditional audio pitch shifters. Being synthesized from scratch, the pitch shifted instrument sounds more like the real instrument playing at a different pitch rather than a pitch shifted sound. A Max patch allowing for pitch shifting is displayed in the *Appendix*, figure A.2 page 66.

3.5 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 32-bit floating-point[3] numbers at 86Hz (about 1 Kb/sec), the control parameters can easily be sent over the internet in real time. Remote real-time synthesis over an ethernet network was performed successfully by transmitting the control data with OpenSound Control[4] (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 (3.3 - *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 (3.3 - *right*) shows the synthesis client Max patch that was used to receive the control data and to synthesize a female singing voice.

We have also experimented with having a five-string polyphonic electric violin 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 a complex set-up can either be handled on one machine or shared by several machines. Figure 3.4 shows how we could easily share the analysis process and the synthesis between two different 500MHz Macintosh G4 computers. Experiments show that there was no loss of data after a few minutes of transmission and no audible delay when computers shared the same local network.

[3]Another implementation of the system could possibly use integers instead.

[4]Earlier experiments of OSC and additive synthesis with a different system (The Matrix) were done between the MediaLabEurope in Dublin and MIT. They showed surprising low delays (about 0.5 sec) and little loss [Overholt, 2001].

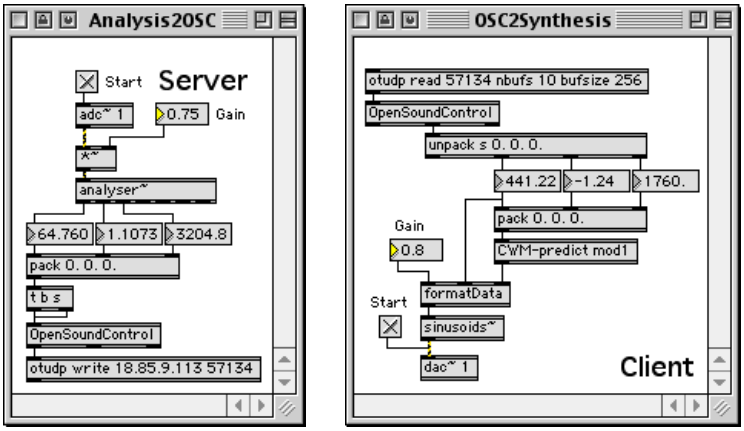

Figure 3.3: *left:* Max analysis server patch. *Right:* Max synthesis client patch.

3.6 Toy Symphony and the Bach Chaconne

The Perceptual Synthesis Engine described in this thesis, has been especially developed for the *hyperviolin*, a soloist instrument to be performed by violinist Joshua Bell in the Toy Symphony[5] project. More details on the project can be found on the following web site: `http://www.toysymphony.net`

3.6.1 Classical piece

In Toy Symphony, Joshua Bell will first perform a 15 minute solo-violin piece from the classical repertoire: the Bach *Chaconne*. This piece, that was often transcribed for other instruments, features several sections and "moods." For its "hyperized" version, Josh will play the original score, but will benefit from several synthesis techniques, including the one described in this thesis. He will be able to choose between several timbre models, and morphing transitions between them. The Jensen violin (see figure 1.1) is multichannel,

[5]Toy Symphony is a three-year project (1999-2002) that brings together children, virtuosic soloists, composers, and symphony orchestras around the world to radically alter how children are introduced to music, as well as to redefine the relationship between professional musicians and young people.

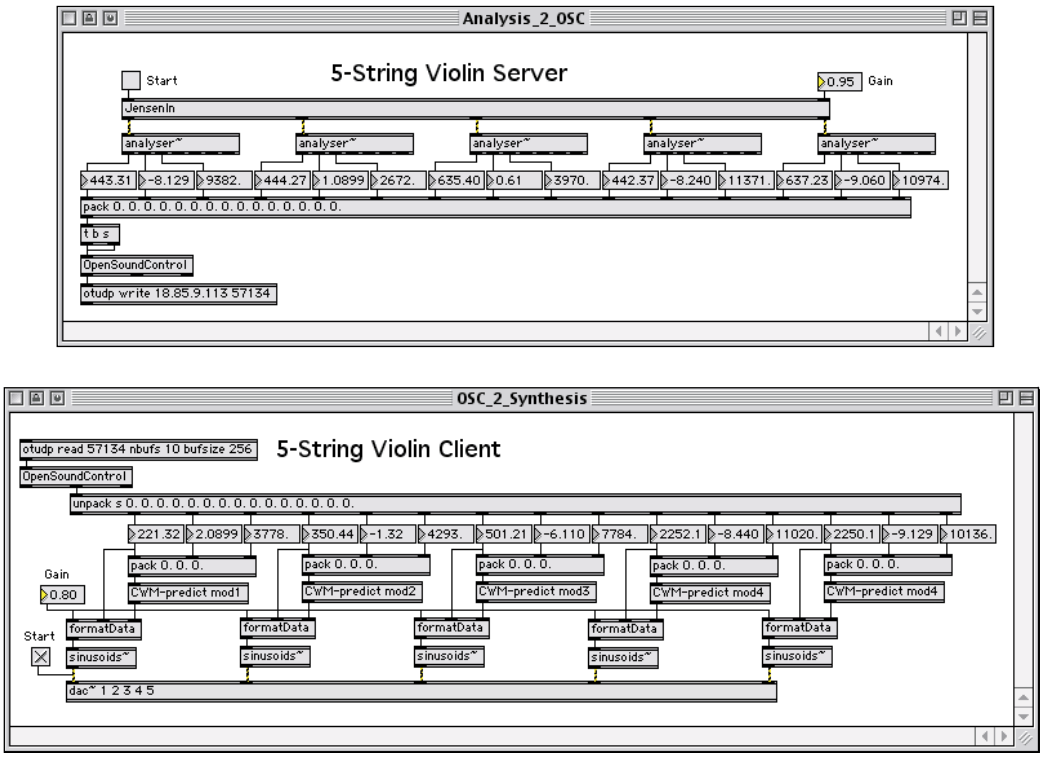

Figure 3.4: OSC implementation for the 5-string violin. *Top:* Max analysis server patch. *Bottom:* Max synthesis client patch.

so one or more models will be assigned on each string.

In that piece, we will use a combination of the three controlling options enumerated in the *Morphing* section (see section 3.3). For instance, by wearing wireless expressive footwear[6] [Paradiso et al., 2000b, Paradiso et al., 2000a] enhanced with many embedded sensors (e.g., continuous/dynamic pressure, 3D direction, 3D acceleration, height, 2D location, etc.), the player will be able to continuously manipulate the timbre that he is playing. Several experiments have already shown that the shoes have great potential as a high-level musical-parameter controller. For example, given a set of core models, the direction the foot is pointed out may determine the current chosen model or morph between the two adjacent models.

[6]more commonly called "sensor shoes" "musical shoes" or "dancing shoes"

The piece will be divided in musical sections that will correspond to independent complex presets of models available, additional sound effects (e.g., reverb, delay, etc.) and control rules. Josh will walk along a path, which, visually, will represent the piece timeline. A new step on stage is more likely to be detected via pressure sensors along that path. At any time in the piece, a combination of sensor values coming from the shoes will result in a sound configuration pulled out from the virtual *timbre space* available in the current preset. The possible sound results might go from a choir of voices in a cathedral, to a more intimate flute sound in a small room. As every models can morph between each other, we therefore avoid any unwanted abrupt transitions between sounds and preset changes. While allowing for much creativity in its interpretation, the piece preserves its whole entity and does not sound like a juxtaposition of unrelated and disconnected sections.

3.6.2 Original piece

The hyperviolin will finally be a part of an original piece from composer Tod Machover, titled *Toy Symphony*[7] that involves a full symphony orchestra and children playing specially designed "smart" musical toys [Weinberg, 1999, Weinberg et al., 2000].

For that piece, the basic principle will be similar to the one described in the previous section, but will feature an other dimension: a layer of interaction with the other players, and more particularly with the children. Josh will be free to walk on stage and musically interact with a chosen child or group of children playing "shapers[8]." The child will be able to subtly modify the timbral characteristics of Josh's music without altering his musical intention.

[7]Première is in February 2002 in Berlin with the "Deutsches Symphonie-Orchester Berlin" conducted by Kent Nagano.

[8]Simple musical interface that allows to shape a timbre or manipulate a musical piece.

3.7 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. 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, 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. When designing new computer instruments and controllers, we should respect the familiarity and closeness of the skilled musician with his or her instrument, even though he or she can adapt to a new controller or feedback mechanism.

The correlation between analysis and prediction and synthesis that I use in this thesis and have described in the section *Timbre Analysis and Modeling*, is a one-to-one instantaneous mapping, i.e., one $\mathbf{x}$-dimension-point input gives a one $\mathbf{y}$-dimension-point output. However music, or audio in general, is time-dependent. I believe it is important to take into account the analysis *history* in order to better predict the current synthesis. In other words, we want to refer to the *path* that analysis took rather than the current state of the analysis. We want to model the dynamic changes of the analysis — Is the pitch going up or down? Is the loudness static or changing? Is it a

vibrato or a more dramatic change? Some experiments were done using past analysis data to better describe the current state of the spectrum. Typically, multiple points were added at the modeling step, i.e., the input vector became $[P_0, L_0, B_0, P_{-1}, L_{-1}, B_{-1}, \ldots, P_{-N}, L_{-N}, B_{-N}]$ where P_j, L_j, and B_j represent pitch, loudness, and brightness of sample j and where N is the number of past analysis samples that we consider. The same description for the output vector was used: $[A_0, M_1, A_1, M_2, A_2, \ldots, M_{L-1}, A_{L-1}]$ where A_i is the logarithmic magnitude of the i-th harmonic and M_i is a multiplier of the fundamental frequency F_0. Number of samples from 2 to 15 have been experimented with. No significant improvement have been found yet but further experiments need to be conducted.

The author would like to mention that in the case of instrument models such as the violin, where there is more than one possible way of playing a given note, no distinctions were done between these different cases. For example, a G on a particular string, may not sound exactly the same as that identical G on a different string. A choice could have been systematically made between the two in order to be consistent. We rather end up with an approximated G somewhere in between these two.

Conclusions and Future Work

This thesis described 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 that are used in an additive synthesis approach to generate a new 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 more algorithms that extract better and new perceptual features. A greater variety of instruments such as trombone and flute will be modeled and the noise model will be completely integrated into the application, using noisiness as a new perceptual input descriptor. Finally, a fully working system (see section *Toy Symphony and the Bach Chaconne*) will be presented as part of a major artistic performance with virtuoso violinist Joshua Bell. The system will be fully autonomous and controlled by the musician like a novel instrument.

Extensive experiments will be done with the idea of observing the dynamic path rather than the instantaneous state of the analysis (see *Discussion*). I will for instance explore the Hidden Markov Model technique (HMM)

as well as others to model the dynamics of the analysis.

The author is particularly interested in the analysis of sound and music, and in exploring the many possible musical applications of a learning approach. Observing, characterizing and measuring what we perceive in a waveform, should give us all the useful information about our musical experience.

Appendix A

Instantaneous Frequency Approximation

Let's consider the discrete Fourier transform (DFT) of the sampled signal $s(n)$ for bin k and for a rectangular window of samples starting at time m:

$$X_m(k) = \sum_{n=0}^{N-1} s(n+m)e^{-jwnk} \tag{A.1}$$

with

$$\begin{aligned} w &= \frac{2\pi}{N} \\ k &= 0, 1, \ldots, N-1 \end{aligned}$$

The equivalent DFT of the sampled signal $s(n)$ weighted by a Hanning window is:

$$Y_m(k) = \sum_{n=0}^{N-1} s(n+m)h(n)e^{-jwnk} \tag{A.2}$$

with

$$h(n) = 1 - cos(wn)$$

Given its polar form:

$$Y_m(k) = \alpha_m(k)e^{j\beta_m(k)} \tag{A.3}$$

the instantaneous frequency associated with the k-th bin is expressed as:

$$F_{\text{inst}}(k) = \frac{F_s}{2\pi}\left[\beta_m(k) - \beta_{m-1}(k)\right] \tag{A.4}$$

$$= \frac{F_s}{2\pi}\text{Arg}\left[\frac{Y_m(k)}{Y_{m-1}(k)}\right]$$

This expression implies the computation of two FFTs, one at time m and the other shifted by one sample, at time $m-1$. However, a simple approximation enables us to estimate the instantaneous frequency from a single FFT.

A first observation gives us:

$$\begin{aligned} X_m(k) - \frac{1}{2}\left[X_m(k-1) + X_m(k+1)\right] & \qquad \text{(A.5)} \\ &= \sum_{n=0}^{N-1} s(n+m)\left(1 - \frac{e^{jwn}+e^{-jwn}}{2}\right)e^{-jwnk} \\ &= \sum_{n=0}^{N-1} s(n+m)\left(1-\cos(wn)\right)e^{-jwnk} \\ &= Y_m(k) \end{aligned}$$

Another one leads to an approximation relating $X_m(k)$ to $X_m(k-1)$. This approximation holds because of the absence of any special window applied to $s(n)$ prior to the FFT:

$$\begin{aligned} X_m(k-1) &= \sum_{n=0}^{N-1} s(n+m-1)e^{-jwnk} \qquad \text{(A.6)} \\ &= e^{-jwk}\left(\sum_{n=0}^{N-1} s(n+m)e^{-jwnk}\right) \\ &\simeq e^{-jwk}X_m(k) \end{aligned}$$

Combining these two observations allows us to express both $Y_m(k)$ and $Y_{m-1}(k)$ — implying two FFTs — in terms of $X_m(k)$, $X_m(k-1)$, and $X_m(k+1)$ — only one non-windowed FFT — as follows:

$$\begin{aligned} Y_m(k) &= X_m(k) - \frac{1}{2}\left[X_m(k-1) + X_m(k+1)\right] \qquad \text{and} \qquad \text{(A.7)} \\ Y_{m-1}(k) &= e^{-jwk}\left(X_m(k) - \frac{1}{2}\left[e^{jw}X_m(k-1) + e^{-jw}X_m(k+1)\right]\right) \end{aligned}$$

Substituting these into expression (A.3) finally leads us to the following estimate of bin k's instantaneous frequency:

$$F_{\text{inst}}(k) = F_s \left(\frac{k}{N} + \frac{1}{2\pi} \text{Arg} \left[\frac{A}{B} \right] \right) \tag{A.8}$$

where

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

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

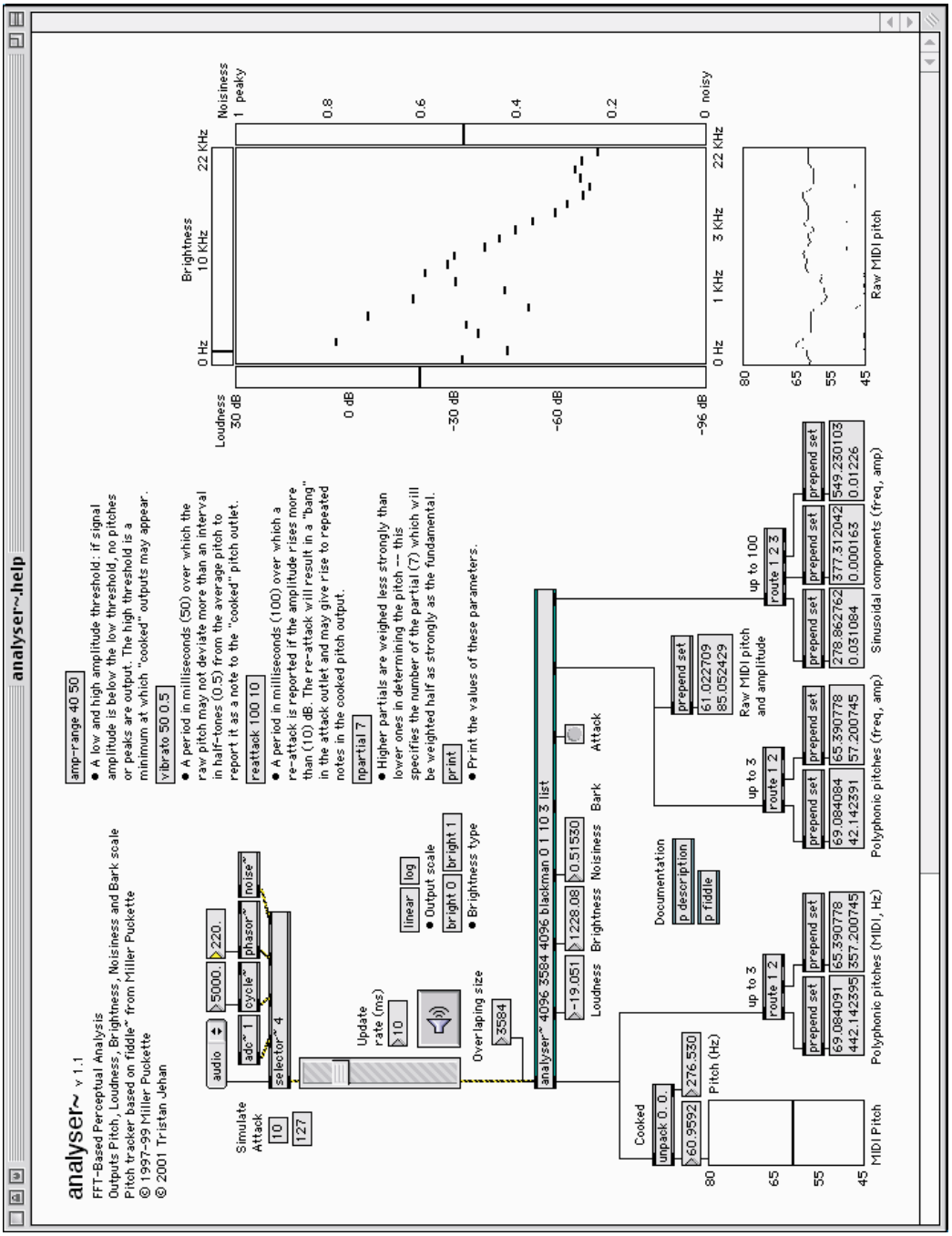

Figure A.1: `analyzer~` help file.

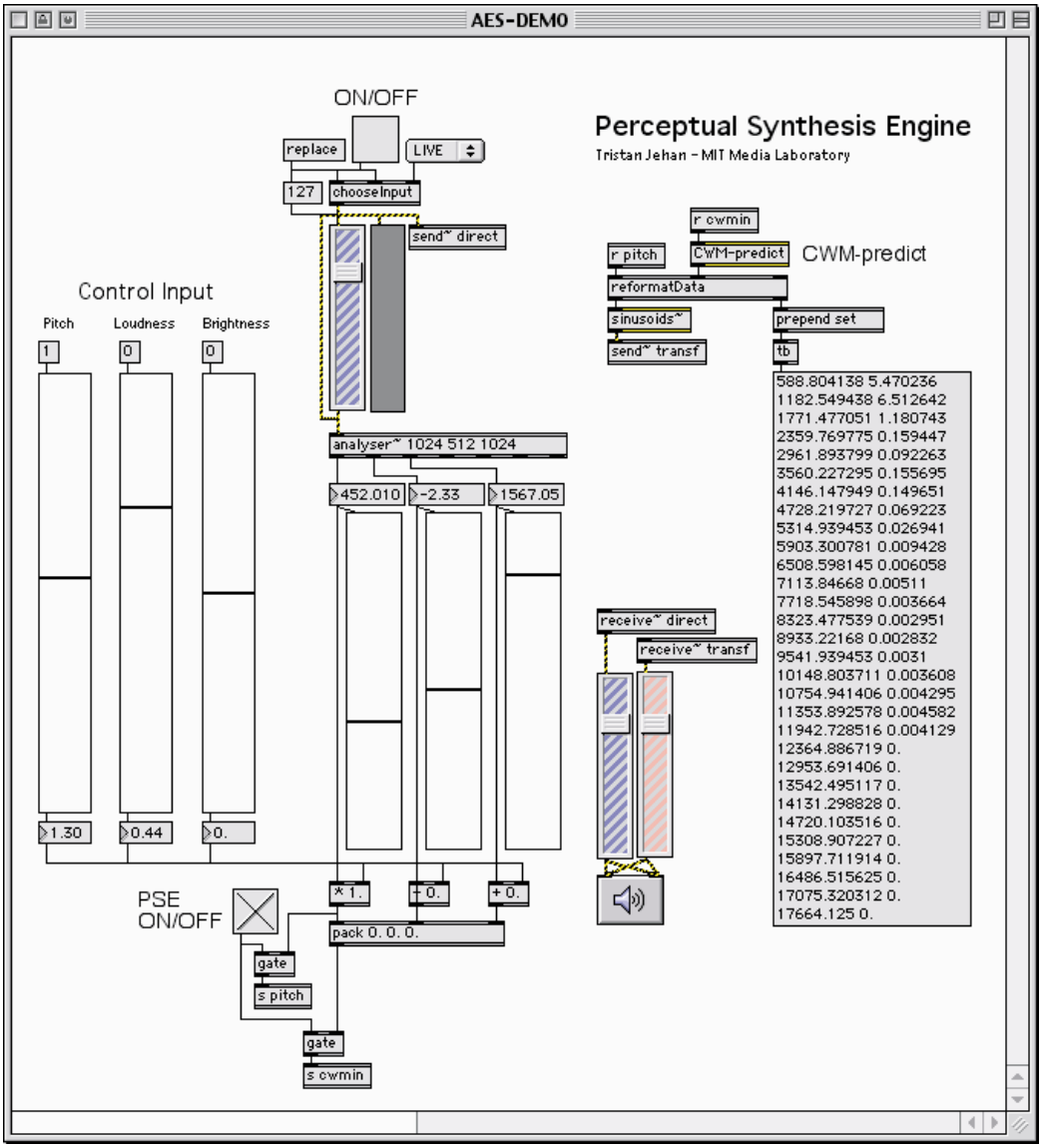

Figure A.2: Simple Perceptual Synthesis Engine Max patch with arbitrary input. The three sliders on the left allow one to modify the analyzed parameters before synthesis, for pitch shifting for instance. The list of floats on the right displays the frequencies and amplitudes of each harmonic. This model outputed 20 sinusoidal functions.

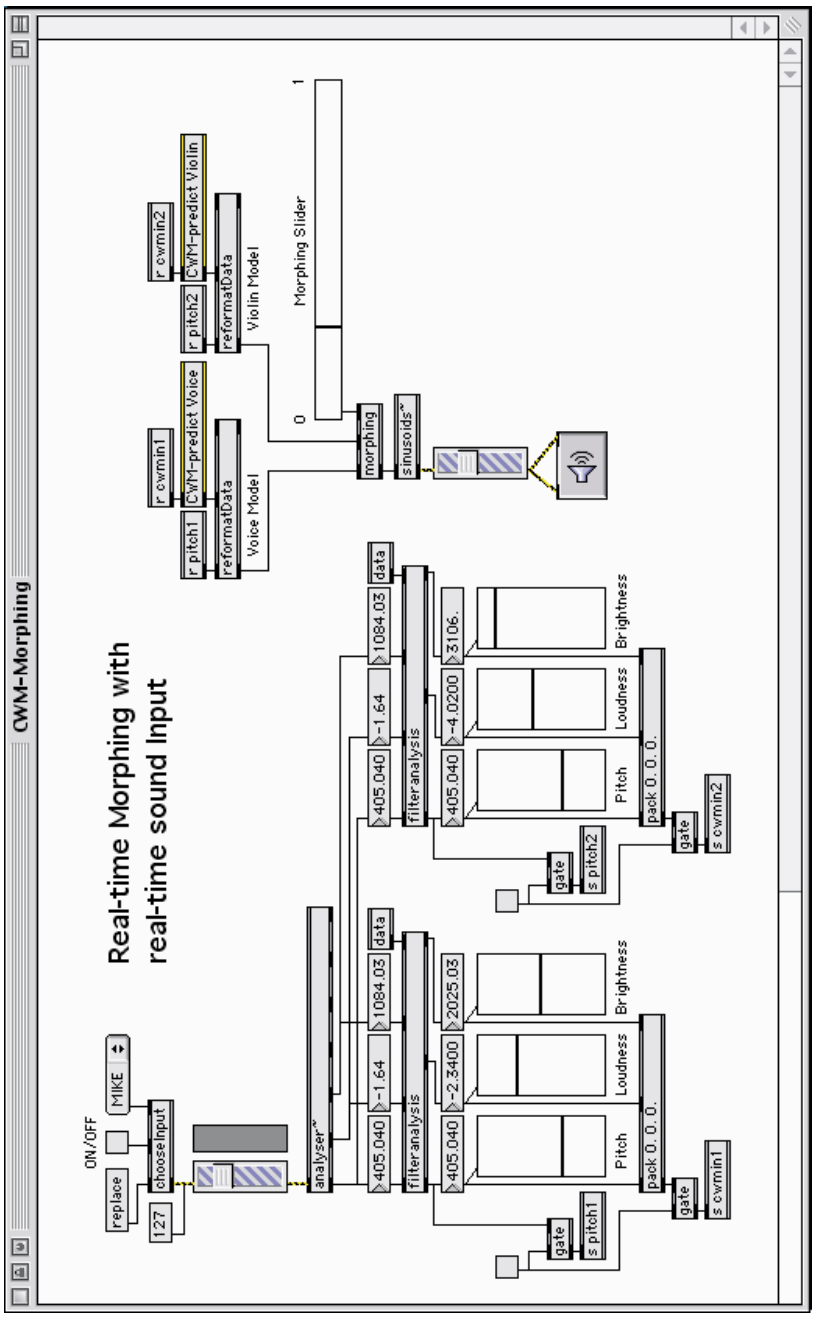

Figure A.3: Simple Morphing Max patch between two models. The input is an arbitrary sound. The values of loudness and brightness are rescaled to fall into the target model range (in "filteranalysis"). The slider on the right morphs between model 1 and model 2. The interpolation between the two lists of synthesis parameters is done in the patch "morphing."

Bibliography

[Arfib, 1979] Arfib, D. (1979). Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves. *Journal of the Audio Engineering Society*, pages 757–779.

[Boulanger, 2000] Boulanger, R. C. (2000). *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*. MIT Press.

[Bregman, 1990] Bregman, A. (1990). *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press.

[Casey, 1998] Casey, M. A. (1998). *Auditory Group Theory with Applications to Statistical Basis Methods for Structured Audio*. PhD thesis, MIT Media Laboratory.

[Chaudhary, 2001] Chaudhary, A. S. (2001). *Perceptual Scheduling in Real-time Music and Audio Applications*. PhD thesis, Department of Computer Science, University of California at Berkeley.

[Chowning, 1973] Chowning, J. (1973). The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21(7):526–534.

[Cook, 2001] Cook, P. R. (2001). *Music, Cognition, and Computerized Sound: An Introduction To Psychoacoustics*. MIT Press.

[Cook and Scavone, 2001] Cook, P. R. and Scavone, G. P. (2001). The synthesis toolkit (stk). `http://ccrma-www.stanford.edu/software/stk/`.

[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.

[Depalle et al., 1994] Depalle, P., Garcia, G., and Rodet, X. (1994). A virtual castrato (!?). In *Proceedings International Computer Music Conference*, Aarhus, Denmark.

[Dodge and Jerse, 1997] Dodge, C. and Jerse, T. A. (1997). *Computer Music: Synthesis, Composition, and Performance*. Schirmer Books.

[Dolson, 1986] Dolson, M. (1986). The phase vocoder: a tutorial. *Computer Music Journal*, 10(4):14–27.

[Farbood, 2001] Farbood, M. (2001). Hyperscore: A new approach to interactive, computer-generated music. Master's thesis, MIT Media Laboratory.

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

[Gershenfeld, 1999] Gershenfeld, N. (1999). *The Nature of Mathematical Modeling*. Cambridge University Press, New York.

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

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

[Goudeseune, 1999] Goudeseune, C. (1999). A violin controller for real-time audio synthesis. Technical report, Integrated Systems Laboratory, University of Illinois at Urbana-Champaign. `http://zx81.ncsa.uiuc.edu/camilleg/eviolin.html`.

[Goudeseune et al., 2001] Goudeseune, C., Garnett, G., and TimothyJohnson (2001). Resonant processing of instrumental sound controlled by spatial position. In *Proceedings of CHI*, Seatle. AMC Press.

[Grey, 1978] Grey, J. (1978). Timbre discrimination in musical patterns. *Journal of the Acoustical Society of America*, 64:467–472.

[Hancock, 1973] Hancock, H. (1973). Headhunters. Columbia/Legacy Records.

[Handel, 1989] Handel, S. (1989). *LISTENING: An Introduction to the Perception of Auditory Events*. MIT Press, Cambridge, Massachusetts.

[Hong, 1992] Hong, A. C. (1992). Non-linear analysis of cello pitch and timbre. Master's thesis, MIT Media Laboratory.

[Jensen, 1999] Jensen, K. (1999). *Timbre Models of Musical Sounds*. PhD thesis, Department of Computer Science, University of Copenhagen.

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

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

[Lansky and Steiglitz, 1981] Lansky, P. and Steiglitz, K. (1981). Synthesis of timbral families by warped linear prediction. In *the IEEE ICASSP, DAFX-6*, Atlanta, Georgia.

[LeBrun, 1979] LeBrun, M. (1979). Digital waveshaping synthesis. *Journal of the Audio Engineering Society*, pages 250–266.

[Lent, 1989] Lent, K. (1989). An efficient method for pitch shifting digitally sampled sounds. *Computer Music Journal*, 13:65–71.

[Machover, 1991] Machover, T. (1991). Hyperinstruments: A composer's approach to the evolution of intelligent musical instruments. *Cyberarts, William Freeman*.

[Machover, 1992] Machover, T. (1992). Hyperinstruments. a progress report 1987-1991. Technical report, MIT Media Laboratory.

[Makhoul, 1975] Makhoul, J. (1975). Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4).

[Masri, 1996] Masri, P. (1996). *Computer Modelling of Sound for Transformation and Synthesis of Musical Signals*. PhD thesis, Department of Electrical Engineering, University of Bristol.

[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.

[Mathews, 1969] Mathews, M. V. (1969). *The Technology of Computer Music*. MIT Press, Cambridge, Massachusetts.

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

[Miranda, 1998] Miranda, E. R. (1998). *Computer Sound Synthesis For The Electronic Musician (Music Technology Series)*. Focal Press.

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

[Oliver, 1997] Oliver, W. D. (1997). The singing tree : A novel interactive musical experience. Master's thesis, MIT Media Laboratory.

[O'Modhrain, 2000] O'Modhrain, M. S. (2000). *Playing by Feel: Incorporating Haptic Feedback into Computer-Based Musical Instruments*. PhD thesis, Stanford University.

[Overholt, 2001] Overholt, D. (2001). The MATRIX: A novel controller for musical expression. In *Proceedings of CHI*, Seatle. AMC Press.

[Paradiso et al., 2000a] Paradiso, J., Hsiao, K., Benbasat, A. Y., and Teegarden, Z. (2000a). Design and implementation of expressive footwear. *IBM Systems Journal*, 39(3 & 4):511–529.

[Paradiso et al., 2000b] Paradiso, J., Hsiao, K., and Benbassat, A. (2000b). Interfacing the foot: Apparatus and applications. In *Proceedings of the ACM CHI Conference, Extended Abstracts*, pages 175–176.

[Paradiso, 1997] Paradiso, J. A. (1997). Electronic music interfaces: New ways to play. *IEEE Spectrum Magazine*, 34(12):18–30.

[Paradiso, 1999] Paradiso, J. A. (1999). The brain opera technology: New instruments and gestural sensors for musical interaction and performance. *Journal of New Music Research*, 28(2):130–149.

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

[Portnoff, 1976] Portnoff, M. (1976). Implementation of the digital phase vocoder using the fast fourier transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(3):243–248.

[Press et al., 1992] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, 2nd edition.

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

[Puckette and Apel, 1998] 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.

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

[Risset, 1969] Risset, J.-C. (1969). *Catalog of computer-synthesized sound*. Bell Telephone Laboratories, Murray Hill.

[Risset and Mathews, 1981] Risset, J.-C. and Mathews, M. (1981). Analysis of musical instrument tones. *Physics Today*, 22(2):23–40.

[Roads, 1995] Roads, C. (1995). *The computer music tutorial*. MIT Press.

[Rodet, 1997] 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.

[Rodet et al., 1984] Rodet, X., Potard, Y., and Barrière, J.-B. (1984). The CHANT project: From the synthesis of the singing voice to synthesis in general. *Computer Music Journal*, 8(3):15–31.

[Rodet and Vergez, 1996] Rodet, X. and Vergez, C. (1996). Physical models of trumpet-like instruments. detailed behavior and model improvements. In *Proceedings International Computer Music Conference*, Hong Kong, China.

[Rowe, 1992] Rowe, R. (1992). *Interactive Music Systems*. MIT Press.

[Rowe, 2001] Rowe, R. (2001). *Machine Musicianship*. MIT Press.

[Sapir, 2000] Sapir, S. (2000). Interactive digital audio environments: Gesture as a musical parameter. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy.

[Scheirer, 2000] Scheirer, E. (2000). *Music Listening Systems*. PhD thesis, MIT Media Laboratory.

[Schoner, 2000] Schoner, B. (2000). *Probabilistic Characterization and Synthesis of Complex Driven Systems*. PhD thesis, MIT Media Laboratory.

[Schoner et al., 1998] 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, 1989] Serra, X. (1989). *A system for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition*. PhD thesis, CCRMA, Department of Music, Stanford University.

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

[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.

[Sethares, 1998] Sethares, W. A. (1998). *Tuning, Timbre, Spectrum, Scale*. Springer-Verlag, Berlin, Heidelberg, New York.

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

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

[Sparacino, 2001] Sparacino, F. (2001). *Sto(ry)chastics: a bayesian network architecture for combined user modeling, sensor fusion, and computational storytelling for interactive spaces*. PhD thesis, MIT Media Laboratory.

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

[Trueman, 1999] Trueman, D. (1999). Reinventing the violin. Technical report, Princeton University. `http://silvertone.princeton.edu/~dan/rtv/`.

[Trueman et al., 2000] Trueman, D., Bahn, C., and Cook, P. R. (2000). Alternative voices for electronic sound. In *Proceedings International Computer Music Conference*, Berlin.

[Trueman and Cook, 1999] Trueman, D. and Cook, P. R. (1999). BoSSA: The Deconstructed Violin Reconstructed. In *Proceedings International Computer Music Conference*, Beijing.

[Verma, 1999] Verma, T. S. (1999). *A Perceptually Based Audio Signal Model with Application to Scalable Audio Compression*. PhD thesis, Department of Electrical Engineering, Stanford University.

[Verplank et al., 2000] Verplank, B., Mathews, M., and Shaw, R. (2000). Scanned synthesis. In *Proceedings International Computer Music Conference*, Berlin, Germany.

[Weinberg, 1999] Weinberg, G. (1999). Expressive digital musical instruments for children. Master's thesis, MIT Media Laboratory.

[Weinberg et al., 2000] Weinberg, G., Orth, M., and Russo, P. (2000). The embroidered musical ball: A squeezable instrument for expressive performance. In *Proceedings of CHI*, The Hague. AMC Press.

[Wessel et al., 1998] 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 *Proceed-*

ings International Computer Music Conference, pages 62–65, Ann Arbor, Michigan.

[Wessel, 1979] 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).

[Winkler, 1998] Winkler, T. (1998). *Composing Interactive Music: Techniques and Ideas Using Max*. MIT press.

[Wright and Freed, 1997] 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, 1998] Zicarelli, D. (1998). An extensible real-time signal processing environment for Max. In *Proceedings International Computer Music Conference*, pages 463–466, Ann Arbor, Michigan.