SINGING-DRIVEN INTERFACES FOR SOUND SYNTHESIZERS

A DISSERTATION SUBMITTED TO THE DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGIES OF THE UNIVERSITAT POMPEU FABRA OF BARCELONA FOR THE PROGRAM IN COMPUTER SCIENCE AND DIGITAL COMMUNICATION IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR PER LA UNIVERSITAT POMPEU FABRA

Jordi Janer

2008

Thesis Director

Dr. Xavier Serra
Department of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona

This research was carried out at the Music Technology Group of the Universitat Pompeu Fabra of Barcelona. Primary support was provided by the EU projects FP6-IST- 507913 Semantic Hi-Fi and FP6-027122 SALERO.

Abstract

Together with the sound synthesis engine, the user interface, or controller, is a basic component of any digital music synthesizer and the primary focus of this dissertation. Under the title of singing-driven interfaces, we study the design of systems, that based on the singing voice as input, can control the synthesis of musical sounds.

From a number of preliminary of experiments and studies, we identify the principal issues involved in voice-driven synthesis. We propose one approach for controlling a singing voice synthesizer and another one for controlling the synthesis of other musical instruments. In the former, input and output signals are of the same nature, and control to signal mappings can be direct. In the latter, mappings become more complex, depending on the phonetics of the input voice and the characteristics of the synthesized instrument sound. For this latter case, we present a study on vocal imitation of instruments showing that these voice signals consist of syllables with musical meaning. Also, we suggest linking the characteristics of voice signals to instrumental gestures, describing these signals as vocal gestures.

Within the wide scope of the voice-driven synthesis topic, this dissertation studies the relationship between the human voice and the sound of musical instruments by addressing the automatic description of the voice and the mapping strategies for a meaningful control of the synthesized sounds. The contributions of the thesis include several voice analysis methods for using the voice as a control input: a) a phonetic alignment algorithm based on dynamic programming; b) a segmentation algorithm to isolate vocal gestures; c) a formant tracking algorithm; and d) a breathiness characterization algorithm. We also propose a general framework for defining the mappings from vocal gestures to the synthesizer parameters, which are configured according to the instrumental sound being synthesized.

As a way to demonstrate the results obtained, two real-time prototypes are implemented. The first prototype controls the synthesis of a singing voice and the second one is a generic controler for other instrumental sounds.

Resum

Juntament amb el motor de síntesi sonora, la interfície d'usuari, o controlador, és un component bàsic dels instruments musicals digitals i principal tema d'estudi d'aquesta tesi. L'objectiu és el disseny d'interfícies d'usuari que permetin el control de la síntesi de sons instrumentals amb el cant.

Partint d'una sèrie d'experiments i estudis preliminars, identifiquem les principals qüestions a tenir en compte en la síntesi sonora controlada per veu. Per abordar-ho proposem dues direccions, una pel control d'un sintetitzador de veu cantada, i una altra pel control de la síntesi de sons instrumentals. En el primer cas, el mapeig entrada-sortida és directe. En el segon, el mapeig és més complexe i dependrà tant de la fonètica de la veu d'entrada com de les característiques del so instrumental que sintetitzem. Per aquest últim cas, presentem un estudi sobre la imitació vocal d'instruments musicals, que ens mostra com la veu està formada per síl.labes, les quals tenen un significat musical. A la vegada, suggerim relacionar les característiques de la veu amb els gests instrumentals d'un intèrpret, describint aquests senyals de veu com a gests vocals.

El control vocal de la síntesi sonora es pot enfocar de diverses maneres. Aquesta tesi busca la relació de la veu i els sons del instruments musicals, abordant tan la descripció automàtica de la veu, com les estratègies de mapeig adients pel control del so sintetitzat. Les contribucions inclouen d'una banda, diversos mètodes d'anàlisis de veu: a) un algoritme d'alineament fonètic basat en programació dinàmica; b) un algoritme de segmentació per aillar gests vocals; c) un algoritme de seguiment de formants; i d) un algoritme de caracterització de la fonació. D'altra banda, proposem un marc general per definir el mapeig des dels gests vocals als paràmetres del sintetitzador, el qual es configurarà segons el so instrumental a sintetitzar. Per tal de demostrar els resultats obtinguts, hem implementat dos prototips a temps real. El primer controla la síntesi de veu cantada i el segon és genèric per la síntesi d'altres sons instrumentals.

Resumen

Junto con el motor de síntesis, el interfaz de usuario, o controlador, es un componente básico de los instrumentos musicales digitales y principal tema de estudio de esta tesis. El objectivo es el diseño de un interfaz de usuario que permita el control de la sintesis de sonidos instrumentales con el canto.

A partir de un número de experimentos y estudios preliminares, identificamos las principales cuestiones a tener en cuenta en la síntesis de sonido controlada por voz. Para tratarlo, se proponen dos direcciones, una para el control de un sintetizador de voz cantada, y otra para el control de la síntesis de sonidos instrumentales. En el primer caso, el mapeo entrada-salida es directo. En el segundo, el mapeo es más complejo y dependerá tanto de la fonética de la voz de entrada como de las características del sonido instrumental que sintetizamos. Para este último caso, presentamos un estudio sobre la imitación vocal de instrumentos musicales, que nos indica que la voz está formada por sílabas que conllevan un significado musical. A la vez, sugerimos relacionar las características de la voz con los gestos instrumentales de un intérprete, y representar la seal de voz como gestos vocales.

El control vocal de la síntesis sonora puede tener muchos enfoques. Esta tesis busca la relación de la voz y los sonidos de los instrumentos musicales, abordando la descripción automática de la voz, así como las estrategias de mapeo adecuadas para el control del sonido sintetizado. Las contribuciones incluyen por un lado, varios métodos de análisis de voz: a) un algoritmo de alineamiento fonético basado en programación dinámica; b) un algoritmo de segmentación para aislar gestos vocales; c) un algoritmo de seguimiento de formantes; y d) un algoritmo de caracterización de la fonación. De otro lado, proponemos un marco general para definir el mapeo de los gestos vocales a los parámetros del sintetizador, el cual se configurará según el sonido instrumental a sintetizar. Para demostrar los resultados obtenidos, hemos implementado dos prototipos a tiempo real. El primero controla la síntesis de voz cantada y el segundo es genérico para la síntesis de otros sonidos instrumentales.

Acknowledgments

First of all, I would like to express my gratitude to my supervisor Dr. Xavier Serra for inviting me to join the Music Technology Group in Barcelona.

Of course, this work could not be possible without the help of many people at MTG during the last five years. I wish to thank specially my colleagues Jordi Bonada, Esteban Maestre, Merlijn Blaauw, Alex Loscos, Sergi Jordà, Alfonso Perez, Lars Fabig, Maarten de Boer, Oscar Mayor and Perfecto Herrera. Thanks to Graham Coleman, Greg Kellum and Paul Brossier for reviewing this document. Thanks to Jose Lozano, Lucas Vallejo, Amaury Hazan, Ines Salselas, Ricard Marxer, Jose Pedro García-Mahedero for the recording sessions. I also would like to thank the friendship and support of Fabien Gouyon, Guenter Geiger, Martin Kaltenbrunner, Pedro Cano, Alvaro Barbosa, Markus Koppenberger, Emilia Gómez, Enric Guaus, Oscar Celma, Ramon Loureiro, Marcos Alonso, Salvador Gurrera, Gunnar Holmberg, Pau Arumí, Joana Clotet and Cristina Garrido.

Outside MTG, I wish to thank Vincent Verfaille, Philippe Depalle, Alicia Peñalba, Alexander Jensenius, Joseph Malloch, Stefania Serafin, Marcelo Wanderley and Gary Scavone for the collaboration. Thanks to Pierre Dutilleux and Paulo Ferreira-Lopes for introducing me to the music technology research; and Klaus Piehl, Michael Ruf, Martin Honisch, Michael Zehnpfennig, Matthias Klag, Holger Drenkelfort and Michael Hirsch for my fruitful experience in the audio industry.

Finally, I would like to thank all my friends and family, and very specially to Gemma for her big and persistent smile.

Contents

Chapter 1

Introduction

This dissertation is framed in the field of *Sound and Music Computing*, which is committed to bring new musical tools to society. Sound and Music Computing is a contemporary designation encompassing fields historically known as Audio Processing, Music Technology and Computer Music, among others.

This chapter aims at putting into context our research, before going into the specific issues of the following chapters. First, we detail the motivations of the author toward voice-driven synthesis and the research context at different levels. Next, we pose the main objectives of this dissertation and summarize the relevant contributions.

1.1 Motivations

The topic of this dissertation is motivated from the confluence of several factors. This section goes from an overview of the scientific and technological contexts to a brief mention to the author's personal trajectory.

1.1.1 Scientific and technological contexts

Research in Sound and Music Computing (SMC) field involves several disciplines, including: Musicology, Physics, Engineering (Signal Processing, Computer Science and Electronics), Psychology and Music Composition (Serra et al., 2007). In most cases, this research is applied to different areas such as Digital Music Instruments, Music Information Retrieval, Interactive Multimedia Systems, etc. Concerning the impact of the SMC research on the society, figure 1.1 suggests a historical tendency. From early research in audio processing and synthesis in the second half of the past century, we are moving to a position where areas such as Music Information Retrieval or Interactive Multimedia systems are more relevant than ever.

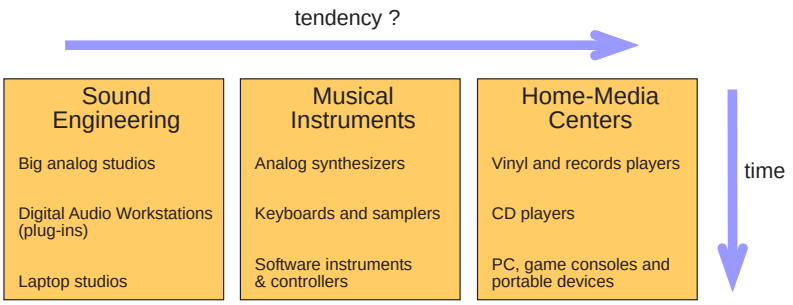

Figure 1.1: Evolution of the technology transfer of Sound and Music Computing research.

The topic of this dissertation could be approached from different perspectives within the SMC field, such as Engineering, Musicology or, outside SMC, also Phonetics. Our research, however, takes an Engineering perspective, focusing on Signal Processing, while having weaker links to Human-Computer Interaction and to a much lesser extent to Musicological aspects. The outcomes of this research should be applied to digital musical instruments. During the last few decades, technology has revolutionized the way we understand musical instruments, both in terms of interaction and the generated sounds. One of the interaction possibilities that arises is to use the voice for controlling digital musical instruments. Our particular interest is the design of systems that take the singing voice as input, in what we have called *Singing-driven Interfaces*. Traditionally, there has been a separation between gesture controllers and sound synthesis engines, which has led to a reduction of the "feel" of controlling the sound. However, nowadays electronic music is mostly performed with laptops and their accompanying interfaces (mouse, keyboard, tablet), which were not designed for music performance. Also, musical controllers have been very often designed without a knowledge of the sound generation process, which actually limits the control possibilities. This is the major criticism to new digital instruments, and one of the trends in designing future digital musical instruments is the integration of control with sound generation. These new instruments should not be composed of separate and disconnected units (gesture controller/sound generator) but should be tackled in an holistic manner. Their design should consider sound control, mappings, interface design, ergonomics, etc. This dissertation's research moves in this direction.

Concerning the electronic manipulation of the singing voice, several tools have been developed since the first studio recordings. First analog hardware devices were compressors and reverberation units that gave more presence to the voice recording when mixed with other instruments. In the digital realm, among other effects, specific tools for the singing voice commonly used are auto-tune and harmonizers. The former has become the perfect companion for every singer in a recording studio, allowing to correct a posteriori accidental pitch deviations. Another area of application of singing voice technology is entertainment, basically karaoke systems[1]. Although having been

[1]It includes morphing systems such as *Elvis*(Cano et al., 2000) or video games such as Sony's Singstar or Karaoke

investigated for several decades, singing voice synthesizers have only been released commercially very recently.

Besides singing, research on the human voice has been primarily carried out within the field of Speech Processing. Actually, the title of this dissertation is inspired by speech-driven interfaces (SDIs). These speech interfaces are increasingly being integrated in a variety of applications, and rely on automatic speech recognizers to control a specific task using a vocal input. Apart from biometric applications such as speaker identification, speech-driven interfaces can be employed in everyday activities (e.g. light-switching). In terms of research initiatives, a number of projects with public funding in the past years (e.g. SpeeCon[2]) confirms the interest in speech-driven interfaces.

Other related applications in the field of Speech Processing include Automatic Speech Recognition (ASR), Lip-Syncing and Text-to-Speech synthesis (TTS). In particular, we believe that part of the results of our research can be useful in non-music contexts such as speech interfaces. For example, the proposed phonetic-alignment module for controlling a singing voice synthesizer (see chapter 4) could be integrated in lip-syncing applications.

1.1.2 Research at the Music Technology Group (UPF)

This work has been carried out at the Music Technology Group (MTG) [3], Universitat Pompeu Fabra (UPF) in Barcelona. The MTG is a research group founded in 1994 by Dr. Xavier Serra, having now more than forty researchers working on computer music and audio technologies. From the initial work on spectral modeling, the research activities cover currently different areas such as sound synthesis, audio identification, audio content analysis, description and transformation, and interactive systems. It is important to emphasize that such an environment, surrounded with so many people working on music technology is an excellent and unique situation to carry out this dissertation. Among the various areas of expertise within the MTG, these topics are specially related to our work:

- Voice Processing
- Interactive Systems
- Musical Content Description

Regarding voice processing, the MTG has carried out several research projects, some of them being commercialized. Likewise, a rapidly emerging area in the Sound and Music Computing community is *Music Information Retrieval*. Within this area, the MTG is also especially active participating in European projects such as *Cuidado* or *Simac* [4]. An example of the integration of audio description algorithms is the SoundPalette (Celma et al., 2004), developed within the Cuidado Project.

Revolution (`http://www.singstargame.com`, `http://www.konami.com`).

[2] SpeeCon (Speech Driven Interfaces for Consumer Applications) project IST-1999-10003 `http://www.speechdat.org/speecon/index.html`

[3] `http://www.mtg.upf.edu`

[4] `http://www.semanticaudio.org/`

1.1.3 Personal trajectory

The work presented in this dissertation spans over 5 years. Apart from the research here presented, I have been involved in other activities such as the participation in various European research projects and technology transfer actions. Here I list the main research projects relevant for this work. I also include a few lines on my personal background.

- **Cuidado Project**: The objectives of the Cuidado project was to integrate content-based audio description in music applications, including the production side and the listening side. The author was involved in the integration of audio description algorithms in the *SoundPalette Offline* application, and building a ready-to-use installation package. This application was shown at the 25th International AES Conference (Celma et al., 2004).

- **Espresso Project**: The analysis of the singing voice in a karaoke context was the principal objective of this project funded by Yamaha Corp. Actually, the research on voice description has been a useful starting point for this dissertation. Several methods were investigated, providing an extensive list of extracted voice features.

- **SemanticHiFi Project**: This thesis has been partially carried out under the umbrella of the EU-Project *Semantic HiFi* [5], FP6-IST-507913, which started in 2004 with a duration of three years. *Semantic HiFi* 's goal was to develop a hard-disk based HiFi system of the future. This novel system should deal with meta-data, as well as with sound transformations, bringing to the home user, not the traditional passive but an interactive experience when listening to music.

  In addition, the project contemplated the development of several software applications targeted to the DJ community. Within the Semantic HiFi Project, the MTG was responsible of a work-package devoted to performance applications dealing with: Audio Transformation, Voice Control and User Interaction devices. Some of the developments were included as final prototypes of the Semantic HiFi project. One example is the *Instrumentizer*, a VST-plugin for voice-driven synthesis that was integrated in the *Authoring Tool* prototype, developed by Native Instruments.

- **BE-2005 grant**: Besides the previous research projects, I received a grant by the Catalan government (BE-2005 grant, AGAUR) for a 3-months research stage at the McGill University in Montréal (Quebec, Canada) under the supervision of Dr. Philippe Depalle (Music Technology Area, Faculty of Music). The results of the research carried out are described in section 3.2.3.

Finally, a few words about my personal background. An early attraction for music and sciences and later for acoustics and audio technologies led me to the enrollment in a degree on Electronic

[5] `http://shf.ircam.fr/`

Engineering, specialized in Sound and Image. A growing interest on Digital Audio Processing encouraged me to complete my undergraduate studies with an internship at the Institute for Music and Acoustics at ZKM—Center for Art and Media in Karlsruhe, Germany[6]. During this internship I wrote my Master's Thesis (Janer, 2000), and collaborated on the development of a multimedia installation[7]. After my engineering studies I worked as a Digital Signal Processing Engineer at the audio company Creamware Datentechnik, GmbH[8] in Siegburg, Germany. There, I was involved in the developments of the following commercial products: NOAH (hardware synthesizer), B-2003 (virtual organ synthesis plugin), Optimaster (mastering plugin) and Vinco (compressor plugin).

In December 2003 I became a researcher and PhD student at the Music Technology Group of the Universitat Pompeu Fabra (UPF) in Barcelona. In 2004, I became Assistant Lecturer in Signal Processing in the Department of Information and Communication Technologies of the UPF.

Apart from my professional and academic trajectory in the audio field, my passion for music (an indispensable reason for working in this field) has other facets, such as being a very limited pianist, who enjoys trying to produce sound from any kind of musical (or not musical) instrument. Other music related activities are co-direction of a weekly musical program in a local radio station, and the production of soundtracks for several audiovisual works.

1.2 Dissertation aims and main contributions

As already mentioned, a current bottleneck in the design on new digital instruments is on the control side. We strongly believe that human voice as a control is an interesting path to explore. We should be able to build tools that capture effectively the musical expression[9] of the voice, and use it meaningfully to control sound synthesizers.

Summing up, the general aims of this dissertation are the following:

1. to highlight the possibilities of the voice as a means for controlling sound synthesizers.

2. to study how people imitate musical instruments with their voices, as it may give us cues for the synthesis control.

3. to design signal processing methods for extracting voice features of relevance for controlling tasks.

4. to design appropriate mapping strategies between voice features and synthesizer parameters.

[6] `http://www.zkm.de`

[7] The installation *Architektur und Musik Labor* was conceived by Pierre Dutilleux, I was involved in porting it to Max/MSP. The updated version was presented at ZKM and in an exhibition at the Architecture Museum in Augsburg, Germany, `http://www.architekturmuseum.de/augsburg/ausstellungen/detail.php?which=26`

[8] `http://www.creamware.de`

[9] In this dissertation, we use the term of musical expression for describing low level attributes such as pitch and dynamics.

5. to develop prototypes to demonstrate the use of the human voice as a controller for sound synthesis devices.

From those general aims, our main contributions are:

- Characterization of the vocal imitation of musical instruments: a web survey collecting more than 850 transcriptions of a set of musical phrases performed on sax, violin and bass guitar.
- Low-delay Phonetic Alignment: it aligns the singing voice input and the lyrics, allowing a real-time interaction.
- Formant Tracking algorithm: it is based on the Discrete-Cepstrum and it is specifically designed for vocal control in real-time by looking at *anchor* vowel sounds.
- Syllabling Segmentation algorithm: the segmentation algorithm relies on heuristic rules that primarily look at timbre variations and the musical role of the phonetics.
- Mapping strategies for voice to instrumental sound synthesis: it a generic framework composed of three-layers, which converts a voice signal into the control parameters of a sample-concatenation synthesizer.
- Development of real-time prototypes:
  1. *PDriven* Prototype featuring real-time control of a singing voice synthesizer.
  2. *VDriven* Prototype featuring real-time control of an instrumental sound synthesizer.

1.3 Dissertation outline

This dissertation is about controlling digital musical instruments. In our case, it is about the use of the singing voice as a musical controller. Although, we use this concept throughout this document (e.g. in the title of chapter 3), considering the voice as a controller might be controversial. A more precise and consistent definition of our topic is *voice-controlled* or *voice-driven sound synthesis*. Finally, we opted to describe our system as a musical counterpart of speech-driven interfaces[10], resulting in the definitive title of *Singing-driven Interfaces for Sound Synthesizers*.

Concerning the structure of this document, it is divided in seven chapters. In chapter 1, we have introduced the context of the research and describe the aims of the dissertation. Chapter 2 is devoted to scientific background, providing also a survey of existing sound-driven sound systems. Chapter 3 presents a first approximation to the various issues involved when using the voice as a control signal. It discusses the limitations of existing pitch-to-MIDI systems, and presents a number of experiments

[10]This is the term used the Human-Computer Interaction community for those devices that react to spoken commands.

and studies. The outcomes of chapter 3 establishes the structure of the remaining chapters. First, chapter 4 addresses the control of a singing voice synthesizer based on sample concatenation. It consists of a phonetic alignment and a performance analysis module. In this first case, input and output signals are of the same nature, and the mappings to the synthesizer's parameters can be direct. Next, we dedicate chapter 5 to study the automatic description of the voice signals that will be used to control the synthesis of other instruments. The contributed analysis methods address voice signal segmentation and timbre description. At the same time, the chapter proposes to represent these signals as vocal gestures, searching for the relation to instrumental gestures. Chapter 6 addresses the control issues for other instrument synthesizers. From the results of chapter 5, we define a set of mappings that depend on the synthesis technique, imitated instrument and time mode (real-time and off-line). Chapter 7 identifies the relevant contributions of this dissertation, and proposes further research directions. Figure 1.2 illustrates the overall organization of this dissertation, indicating how chapters are interrelated.

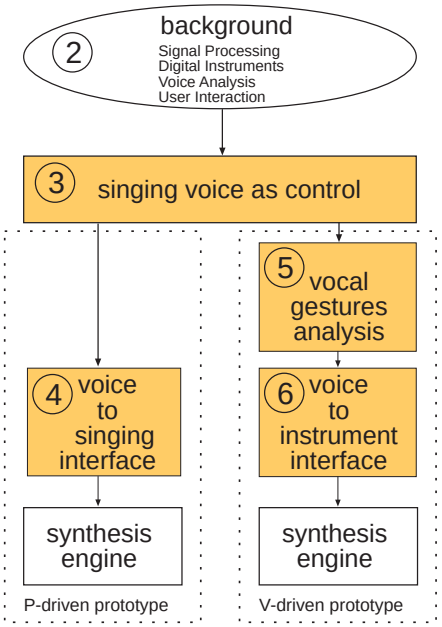

Figure 1.2: Graphical representation of the structure of this dissertation. The numbers indicate in which chapter each module is described.

Accompanying this dissertation we generated a number of sound examples that might be useful to better comprehend some specific topics. All audio examples are described in the appendix B, and available online in `http://www.mtg.upf.edu/~jjaner/phd`.

Chapter 2

Scientific Background

This chapter aims at putting together previous work related to the topic of voice-controlled synthesis in the field of Sound and Music Computing. In a first section, we point out the main characteristics of Digital Musical Instruments, focusing on control and interaction. Next, we survey sonic systems that use sound as control source. Finally, we address computational methods for understanding and processing voice signals.

2.1 Control issues in Digital Musical Instruments

2.1.1 General concepts

The origins of electronic music can be traced back to the end of the 19th century with the first electromagnetic systems. During the 20th century the next significant step was made in the development of electronic organs such as the Hammond [1] organs in the mid-1950's, which used tone-wheels[2] to generate multiple harmonics. These organs were originally conceived as a light and cheap replacement for church organs, and there was no intention to create an innovative and distinct sound, but this was the eventual outcome. Later, the invention of the transistor popularized the analog synthesizers, as Robert Moog's[3] (1925-2005) ones. During the same period, Buchla (2005) also developed innovative analog synthesizers although his inventions never became as popular as Moog's synthesizers. Buchla's developments did not have a keyboard interface, which presumably limited their reach.

Around a decade before the popularization of analog synthesizers, initial experiments on digital sound synthesis were achieved on primitive computers. Pioneering developments were carried out

[1] `http://www.hammondorganco.com`

[2] Tone-wheels are several spinning wheels at different frequency that couple to a magnetic coil generate a quasi-sinusoidal electric signal.

[3] `http://www.moogmusic.com`

by Max Mathews and his colleagues at Bell Labs (Mathews and Guttman, 1959). As computers became more affordable, researchers started investigating digital sound synthesis, thereby bringing about the advent of Computer Music. Digital sound synthesis evolved rapidly, the creation of pioneer real-time systems (e.g. GROOVE by Mathews and Moore (1970)) led to the development of MIDI and a variety of sound synthesizers, which have now gained universal acceptance both musically and socially.

Even in earliest examples of electronic instruments, a remarkable novelty was the fact that the synthesis layer became dissociated from the control layer. This fact meant a radical new perspective for the building of musical instruments. In acoustic instruments, the designer is confined by several limitations, such as the physical dimensions of the sound generation mechanism, and the manner in which the performer controlled the instrument. Moreover, the control interface of the instrument should allow the development of virtuosity in order to facilitate the creation of musically interesting experiences. On the contrary, *Digital Lutherie*[4] - the term employed by Jordà (2005) for referring to the design of digital instruments - has explored multiple alternatives as input devices. Far from being a closed issue, new control interfaces are continuously evolving. Nowadays, the design of new interfaces combines research in Music Technology and Human Computer Interaction, and has dedicated conferences such as the International Conference on New Interfaces for Musical Expression (NIME)[5]. In this section, we aim to give an overview of some key ideas, topics and main achievements within this field.

2.1.1.1 Control in acoustic instruments

In acoustic instruments, the sound generation mechanism is intimately coupled to the playing interface. The sound of a guitar string, is produced by its vibration after being excited (i.e. plucked). A common characteristic among acoustic instruments is that in all cases, the musician's energy is transformed into the instrument's sound energy (Hunt and Wanderley, 2002). Even in a piano in which the interface (keyboard) is separated from the producing mechanism (strings), it is the musician's energy that produces the sound. Also, the performer controls the sound by interacting with the sound production mechanism. Changes of pitch in a string instrument, is done by sliding the fingers up and down on the string. The same concept serves for wind instruments, where a change of pitch is controlled by the position of the fingers on the bore holes.

Acousticians often classify instruments according to the way that the energy is applied. One can distinguish between those requiring a continuous-excitation and those requiring a discrete or impulsive excitation. In the first group, the energy has to be applied continuously to produce sound, as in a flute. In the second group, energy is applied in a discrete manner such as in plucked strings.

The most notable exception in traditional instruments is the organ, which in many ways resembles

[4] *Lutherie* is the French term for the craft of instrument-making, usually for string instruments.
[5] `http://www.nime.org`

electronic instruments. Firstly, the musician's energy does not create the produced sound. Instead the player just controls a keyboard and pedals which act as switches for several valves. The energy of the actual sound produced comes from the air which is pushed by bellows through the pipes. In the case of the organ, the interface is isolated from the sound producing system. Thus, organs could actually have been built with a completely different user interface, perhaps one more innovative than the omnipresent keyboard.

2.1.1.2 Control in digital instruments

In opposition to acoustic instruments, the main attribute of digital instruments is the decoupling of the control from the sound generation. Figure 2.1 shows three examples of musical instruments from a control perspective. Performer's gestures on the digital instrument are coded in a particular digital data stream, which is then used in the sound generation algorithm. What gestures and how they are coded, and more important, how this information is converted to synthesizer parameters (mappings) is the essence of a digital musical instruments.

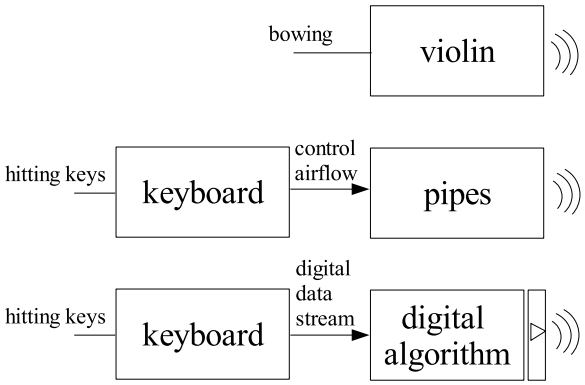

Figure 2.1: Three examples of musical instruments: *top*) violin; *middle*) organ, and *bottom*) digital keyboard synthesizer.

In the literature we find several attempts to build a general model of the Digital Music Instrument. Jordà (2005) talks of a digital instrument chain, consisting of a *gestural controller* and a *sound generator*. In their work, Miranda and Wanderley (2006) cover also the interaction with digital musical instruments, focusing on gestural controllers that go beyond the omnipresent keyboard paradigm. One advantage of the separation of the input device and the sound generator is the possibility to combine different controllers and sound generators. A clear example is MIDI (Musical Instrument Digital Interface), which has become the standard "crossbar switch" between gesture and timbre (Rubine and McAvinney, 1990). MIDI facilitated a number of innovations in electronic music, inspiring digital instrument designers to build new and experimental controllers that could be coupled to any sound generator.

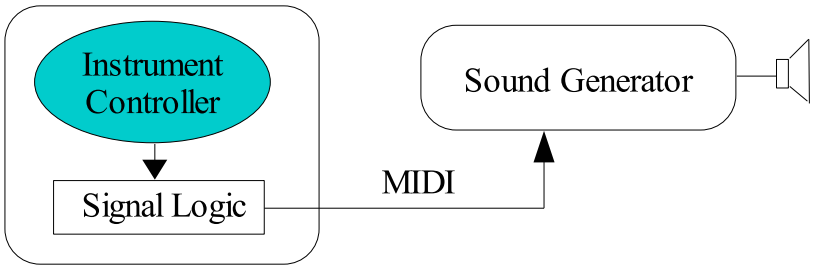

Figure 2.2: Standard MIDI controller/synthesizer system.

MIDI is not a musical language in that it does not directly describe musical sounds. Rather, it is a communication protocol that permits the exchange of musical information by means of control signals. It was motivated by an agreement amongst manufacturers of music equipment, computers and software. The full *MIDI 1.0 Detailed Specification* was first released in 1983 (The International MIDI Association, 1983) and already discussed few years later(Moore, 1988). MIDI messages can be discrete (e.g note on, note off), and continuous controllers (e.g. pitch bend or expression). As we have mentioned, MIDI data does not provide any information about the actual sound (instrument, timbre, etc.). Thus, the timbre depends entirely on the sound generator. In 1990, a new standard was added to the MIDI specification under the name of *General MIDI*. This provided a set of 128 fixed timbres, which added a degree of uniformity among the different commercial instruments.

MIDI has a number of significant limitations. As Roads (1996) points out, these limitations can be grouped in three categories: bandwidth, network routing, and music representation. We are concerned with the first and third group. Firstly, the bandwidth of MIDI can be overwhelmed by a single performer making heavy use of continuous controllers such as foot pedals or a breath controller. With regard to musical representation, MIDI was primarily conceived for equal tempered popular songs played on a musical keyboard. However, for the objectives of this dissertation, a representation more suited to the singing voice would be convenient, as we cover in detail in chapter 5.

In order to overcome some limitations of MIDI, researchers at CNMAT, UC Berkeley, developed the Open Sound Control (OSC) protocol[6]. It has made a significant headway towards becoming the next standard protocol among sound devices (including computers, controllers, sound synthesizers and others). The first specification dates from 2002, and an increasing number of musical software applications already support OSC.

[6] `http://www.opensoundcontrol.org` OSC has been integrated in software applications such as Native Instrument's *Reaktor*, Cycling'74 *Max/MSP* and *SuperCollider*.

2.1.2 Controlling expression

Music's principal objective is to transmit emotion. In digital musical instruments, this *emotion* can be viewed as a human quality that is conveyed by, a sometimes unnatural, synthesized sound. Achieving expressive results with digital musical instruments can be accomplished mainly from distinct perspectives. In the first, a performer employs interfaces to attain an expressive sonic result[7]. In the second, a score generates expressive control data by means of artificial intelligence techniques [8]. Yet another alternative is to control the sound synthesis with the acoustic signal of a performance, which already conveys expression.

2.1.2.1 Control with input devices

Musical input devices, also referred as *musical interfaces*, *musical controllers* or *gesture controllers*, are mainly employed in performance situations and work in real time. These interfaces capture performer actions or *gestures*. Which gestures and how they are parameterized will determine the synthesized sound.

Pressing (1992) presents also a thorough study on the use of musical controllers in sound synthesis, putting special emphasis in instrument imitation. He explores which parameters of a given controller (keyboard, windcontroller, etc.) are important for controlling different types of instrument sounds. He describe two performer models, one for sustained monophonic sounds (bowed strings, brass, etc.) and one for decaying polyphonic sounds (guitars, piano, etc.). Nowadays, a wide variety of musical controllers are available, ranging from traditional keyboards to sophisticated artificial vision systems. Roads (1996) and Paradiso (1997) explore the myriad of controllers and studies how they affect music playing. Looking at the most typical example, the MIDI keyboard provides users with only two instantaneous parameters to control the sound synthesis, velocity and pitch. With the exception of aftertouch, other controls present such as pitch-bend wheels, pressure sensitive ribbons or joysticks imply making separate actions from the traditional keyboard playing gestures. Therefore, controlling non-percussive sounds with a keyboard controller faces inherent limitations. For an up to date list of new developments, the reader can refer to the proceedings of recent NIME conferences[9].

Specifically related to voice, in commercial singing voice synthesizers such as Yamaha's Vocaloid system[10], one can use a MIDI interface for recording a score. However, there is no known *musical controller* appropriate for singing voice synthesis, as there are for other types of synthesis such as MIDI wind controllers for wind instrument synthesizers. This fact restricts the generation of

[7] This is the most common use of digital musical instruments, using them as we would do with acoustic instruments.

[8] Adding expression to a musical score is a more reduced use of digital musical instruments. By training one or more models of expression (e.g different moods, or different performer), one can control the expression of the synthesized result.

[9] NIME (International Conference for New Interfaces for Musical Expression) is the major scientific conference dedicated exclusively to the design on musical controllers. `http://www.nime.org`

[10] `http://www.vocaloid.com`

expressive MIDI scores which are suitable to the voice. Alternatively, the control parameters of the synthesizer can be manually edited using a GUI, drawing pitch and loudness contours, or adding high-level expressive gestures such as vibrato. Nevertheless, manually editing parameters does not solve the problem completely either since it is a slow and tedious process, albeit very flexible.

2.1.2.2 Control with automatic expression models

Part of the research on musical expression deals with characterizing and modeling performances in order to, given a neutral performance or a written musical score, bring emotion (e.g. "sad", "happy", "dark", etc.) or even to reproduce the playing style of a given performer (Widmer and Goebl, 2004). Initial approaches worked solely at an abstract level, such as MIDI scores. Using machine learning techniques the system learns different types of manually annotated performances for a given score. Then it modifies the neutral performance to another synthesized one that conveys a particular expression. Other approaches address expressivity-aware tempo transformations also on a symbolic level (Grachten, 2006). Recent studies tackle the modeling and control of expressive performances at a signal level. In these systems, given a neutral performance the user navigates in a two-dimensional expression space, controlling the mood of the transformed sound (Canazza et al., 2004). The type of control offered by automatic expression models is of a very high-level. User actions are therefore limited to selecting among predefined expressive performances or at most to controlling the amount of morphing between two of them.

2.1.2.3 Performance-driven control

An alternative way of applying expressive control to a synthesizer is to use an audio-stream as a control. The term "Performance-driven" is used in the literature (Rowe, 1993) as opposite to "score-driven" systems such as score-followers. More precisely, herein we relate this term to systems where an audio input controls the sound generator in an instantaneous manner (within a time frame in the order of milliseconds). This concept of *performance or audio-driven* synthesis is also referred in the literature as "Indirect Acquisition". Several systems addresses this issue, either using a general audio-signal as control (Métois, 1996; Jehan and Schoner, 2001b), or specifically using the singing voice as control for other sounds (Janer, 2005b). In an off-line manner, Meron (1999) uses singing performances as input for an automatically trained system that produces high-quality singing voice synthesis. This dissertation focuses on this type of control where, as we take the singing voice as input signal, we will also name it *voice-driven synthesis*.

2.1.3 Classification of musical input devices

As we have learned, digital musical instruments incorporate an interface (input device) that captures musician gestures and converts them to control parameters for the synthesizer. Designing the interface is hence of high importance, since it defines a *hyperplane* within the *gesture space* for interacting with the instrument. Cook (2001) introduces a number of principles for designing musical controllers taking into account design, human and artistic factors. Musical interfaces, commonly referred as *Musical Controllers*, can be classified from different perspectives. Here, we adopt a classification that is common in the literature (Wanderley, 2001; Miranda and Wanderley, 2006). We categorize interfaces as follows: augmented instruments, instrument-like controllers, and alternate controllers. Concerning voice-driven synthesis, we might consider the voice as an instrument-like controller, where singers benefit from vocal control in a similar manner of a clarinetist using wind-controllers.

2.1.3.1 Augmented instruments

These controllers can be also referred as *extended controllers*, which are equipped with add-on sensors that extend the music control possibilities of a traditional instrument. In this category, we should include Tod Machover's *Hyperinstruments* (Machover, 1992), with the goal of building expanded instruments to give extra power to virtuoso performers. Other developments include percussion, keyboard and string instruments.

2.1.3.2 Instrument-like controllers

There are several reasons for building digital musical instruments that resemble traditional ones. Most importantly, they allow a musician to start playing immediately, since he is already familiar with the instrument. In some cases, these electronic instruments attempt to imitate the original sound (e.g. electronic pianos). Yet another reason, albeit a non-scientific one, is the potential of such instruments for greater commercial success. However, instrument-like controllers can restrict the potential control of synthesized sounds.

Without a doubt, most digital instruments have a keyboard. This fact was accentuated by the popularization of MIDI, which limited the control possibilities of other instrumentalists. In addition to the sensors placed along the keyboard (on/off switches), some keyboards are equipped with other sensors in order to extend the number of transmitted performer's actions. It includes among others: thumb-wheels for pitch bend, touch-sensitive keys for changing the timbre, or vibrato; or more recently, some devices include infrared light-beams. The introduction of breath controllers allowed wind instruments players to use synthesizers more expressively. Yamaha introduced the WX-7 in the late 80's, the first breath controller to output MIDI, with a fingering layout close to a saxophone. Breath controllers capture the state of the valves and the reed, replaced by switches, and breath pressure and mouthpiece pressure sensors, respectively.

2.1.3.3 Alternate controllers

In the category of *alternate controllers*, we consider those designs not included in any of the preceding categories. Here, we describe briefly three different technological approaches toward interface design of new digital musical instruments. An advantage of these innovative input devices is that they can more fully exploit the control possibilities of sound synthesis, since they are not bound up with any preconceived performance notions.

Body-Sensors Sensing technology has become widely available through a number of analog to MIDI converters, that are capable of transforming any physical gesture into a MIDI message. These interfaces have often been used in conjunction with software such as Max/MSP or Pd. Commonly, these devices are found in real-time performances or interactive installations, some popular devices are: Steim's Sensorlab (Steim, 2002), Infusion System's ICube (Infusion Systems Ltd., 2002), Ircam's AtomicPro (Fléty, 2002), and finally, La-Kitchen's Toaster and Kroonde (La Kitchen, 2002). A recurrent goal in gestural controllers is to provide ways of converting dancing events into music. For this purpose several *wearable* interfaces have been developed. A body suit incorporates several position and motion sensors that track the movement of the performer. Another application of sensor interfaces is data-acquisition of musical expression. In the MIT's *Conductor Jacket* (Marrin, 2000), they extract parameters in order to describe the emotional state of an orchestra conductor.

Computer Vision More recently, Computer Vision has also been employed as input device for musical performances, capturing body movements. Such systems allow to play with the whole body, like dancers, freeing the performer of controllers, cables and sensors, thus, favoring expression. A popular computer vision system for real-time musical performances has been the *Eyes Web* [11], developed by Antonio Camurri's group (Camurri et al., 2000). With this application the user can program patches that connect input devices (video, audio or data) with processing algorithms. The outputs can then combine video stream, sound or data stream such as MIDI or OSC 2.1.1.2.

Tangible Musical Interfaces Finally, tangible interfaces have been used in the design of table-top digital instruments such as the AudioPad[12], developed by James Patten and others at MIT (Patten et al., 2002) and the *reacTable**[13] developed at the UPF (Jordà et al., 2005). The latter aims at creating a multi-user instrument that is satisfactory for both kind of potential users: novice and trained musicians. Then, these actions are captured by the computer vision system that maps this data to the sound synthesis engine.

[11] `http://www.eyesweb.org`
[12] `http://www.jamespatten.com/audiopad`
[13] `http://www.mtg.upf.edu/reactable`

2.1.3.4 Evaluation of musical input devices

Looking at the wide spectrum of possible musical input devices, one aims to find ways of comparing them methodologically. Often, new musical controllers are rarely used more than in a demonstration context. Before addressing the evaluation of musical input devices, one should have a look at the field of Human-Computer Interaction (HCI). Baecker and S (1987) defines a number of tasks that can be measured quantitatively from a HCI perspective. Trying to bring these tasks into the musical domain is not straightforward since music encompasses aesthetic considerations. Wanderley and Orio (2002) endeavor to make objective evaluations of musical controllers following the standard methodologies used in Human Computer Interaction. In this article, authors propose a number of musical tasks that can be related to pitch control. These tasks can then be extended to timbre, loudness and timing control. They argue that there exist parallelisms between some musical tasks and tasks from research in HCI. For instance, target acquisition can be similar to achieving a particular single tone (with pitch, loudness and timbre).

In our opinion, the difficulties in evaluating musical controllers quantitatively is one of the major problems in considering new musical interfaces as a research entity. For instance, one cannot trace a clear progression line by looking at the NIME[14]. It explains that some researchers (Cook, 2001) refer to musical interface construction more as an art than a science.

2.1.4 Mapping to sound synthesis engines

We have previously seen that in acoustic instruments the performer interacts directly with the sound generation mechanism (e.g bowing a violin string). After the introduction of the analog electronic instruments first, and later digital instruments, this tight relationship between *control* and *sound production* became no longer necessary.

Concerning digital sound synthesis, since the beginning of computer music a large number of techniques have been developed. However, only few have succeed in reaching a large number of musicians. Most commercial synthesizers, still in 2007, use sample-based techniques. On the one hand, the latest improvements in model-based techniques such as physical models, achieve good sound quality and permit a better parameter configuration. On the other hand, though, memory chips and hard-drives have become much smaller and cheaper, so that sample-based synthesizers can rapidly access huge databases covering a wide range of sounds and articulations. Smith (1991) presents a general overview of digital synthesis including a taxonomy for digital synthesis techniques (see figure 2.3).

In this section, we survey different synthesis techniques and their interface layer. More precisely, we aim to identify which parameters of the algorithms are controllable by the musician and their corresponding mappings.

[14]NIME (International Conference for New Interfaces for Musical Expression) is the major scientific conference dedicated exclusively to the design on musical controllers. `http://www.nime.org`

Processed Recording	Spectral Model	Physical Model	Abstract Algorithm
Concrète	Wavetable F	Ruiz Strings	VCO,VCA,VCF
Wavetable T	Additive	Karplus-Strong Ext.	Some Music V
Sampling	Phase Vocoder	Waveguide	Original FM
Vector	PARSHL	Modal	Feedback FM
Granular	Sines+Noise (Serra)	Cordis-Anima	Waveshaping
Prin. Comp. T	Prin. Comp. F	Mosaic	Phase Distortion
Wavelet T	Chant		Karplus-Strong
	VOSIM		
	Risset FM Brass		
	Chowning FM Voice		
	Subtractive		
	LPC		
	Inverse FFT		
	Xenakis Line Clusters		

Figure 2.3: J. Smith's taxonomy of digital synthesis techniques. This figure is extracted from (Smith, 1991) and has the corresponding copyright.

2.1.4.1 Synthesis techniques control layer

Abstract techniques Under *Abstract techniques*, one understands those methods that are not derived from any physical law but arbitrarily aim to reconstruct complex dynamic spectra. As has been demonstrated in many commercial synthesizers, these techniques are computationally cheap and very flexible in terms of producing a large variety of sounds.

Probably the most popular example of abstract techniques is frequency modulation (FM) [15]. Functionally speaking, we can control three parameters of a basic FM structure: amplitude and frequency of the modulator x_m, and amplitude of the carrier x_{out}, as we observe in the equation 2.2. The most interesting feature of this equation is that the instant frequency $f_i(t)$ of x_{out}, will be also a sinusoidal function. A proper choice of modulating frequencies will produce a fundamental frequency and several overtones, creating a complex timbre.

$$x_{out}(t) = A(t)cos(\psi(t)) \tag{2.1}$$

$$\psi(t) = 2\pi f_c + I(t)cos(2\pi f_m t + \phi_m) + \phi_c \tag{2.2}$$

Finally, another abstract technique is Waveshaping synthesis, first carried out by Jean-Claude Risset in 1969. Like FM, this technique is musically interesting because of the wide ranges of possible timbres it can produce in an efficient manner. Actually, Waveshaping is a non-linear distortion, where an input value x in a range $[-1, +1]$ is mapped by a transfer function w to an output value $w(x)$ in the same range. In its digital form, the transfer function is a table of N values. Depending on

[15] Frequency Modulation synthesis, developed by Chowning in 1973 (Chowning, 1971), enjoying success in the commercial market when it was implemented in the massively sold Yamaha's DX7 Synthesizer

the functions stored in the waveshape, the input's spectrum can be modified in such a way that resembles an acoustic instrument.

Regarding the control, it has been demonstrated by Arfib (1979), that is possible to predict the output spectrum of a waveshaper by restricting the input signal to be a cosine, and using *Chebychev functions* as transfer function. Moreover, this will produce rich harmonic sounds that can be controlled by changing either the amplitude of the input cosine, or the Chebychev polynomial function.

Physical Models Physical Modeling synthesis (PhM) strives to emulate acoustical methods for sounds production, and in particular, to model the physical acoustics of musical instruments. This technique aims at finding the mathematical equations that describe a mechanical-acoustic process. In other words, instead of trying to reproduce a waveform directly (as in sampling synthesis), Physical Modeling simulates first the production mechanism, and this model will "produce" the desired sound output. Because of the mathematical nature of these methods and, on the other hand, their high computational burden, the PhM have not been widely used in commercial synthesizers. Exceptions are Yamaha's VL1 and Creamware's Six-String included PhM algorithms, whose developments are based on research from CCRMA-Stanford University and University of Erlangen, respectively. Apart from recreating faithfully existing instruments, PhM opens new ways for "unreal" sounds; for example, once we have a cymbal model, we can generate a sound of a cymbal with a diameter of 25 meters excited by any impulse signal (e.g. voice) (Bilbao, 2005).

In general, there are two types of Physical Models used in music sound synthesis: lumped and distributed. The former method consists of masses, springs, dampers, and non-linear elements. They can be used to model the players' lips in a brass instrument. Considering this approximation, when a mass and a spring are connected, an elementary second-order resonator is formed, which in terms of Digital Signal Processing corresponds typically to a second-order digital filter.

The *distributed* methods implement delay lines in combination with digital filters and non-linear elements. These delay lines, also referred as *Digital Waveguides* (Smith, 1992), model wave propagation in distributed media such as strings, bores, horns, plates, and acoustic spaces. In order to model losses and dispersion, the distributed methods can be combined with lumped elements, so that, a better simulation of the acoustic instrument is achieved. A basic waveguide building block is a pair of digital delay lines. This pair represents a wavefront traveling up and down the waveguide, due to the reflections. Basically, we have two waves traveling in opposite direction, which causes resonances and interferences. Figure 2.4 depicts a general model for a waveguide instrument model that can simulate string and woodwind instruments.

In terms of control, a drawback of physical modeling synthesis is that each instrument is modeled in a very specific way, following the inherent acoustic properties of the mechanical system. This fact leads to a poor generalization in terms of control parameters. Furthermore, a physical model is controlled by dozens of parameters. If we refer to a computer implementation, controlling the

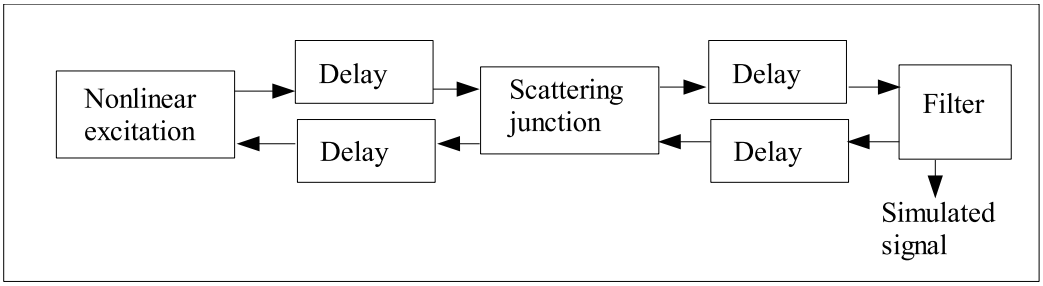

Figure 2.4: Generic waveguide instrument model. (Roads, 1996)

PhM instrument with a mouse and an alphanumeric keyboard would be really difficult. Hence, in parallel to the implementation of new PhM algorithms, researchers have often developed input devices consisting of several sensors. The devices should aid the player in transmitting his or her musical gestures to the algorithm.

Spectral Models First techniques employed for generating sound electronically were *additive* and *subtractive* synthesis. In additive synthesis, a particular timbre is achieved by summing multiple sinusoidal partials of different amplitudes and particular frequencies, following an harmonic series. It shares the same principle used in organs, where a fixed set of oscillator are capable of producing a variety of timbre (registers). In terms of cost, it needs one oscillator per sinusoidal, which makes it rather expensive compared to other techniques. Subtractive synthesis allows also broad range of timbres, where a single highly periodic oscillator is used. In this category, rather than a pure sinusoidal, the oscillators generate a more complex waveform such as triangular, square or saw-tooth. The spectrum presents harmonic components that extend over the whole frequency range. Given one oscillator, the signal passes through filters that *subtract* energy from the original signal, modifying the spectral content. Recent approaches (Verfaille et al., 2006a) combine both additive and subtractive techniques for modeling also noisy components. Amplitude and frequency of the oscillators are taken from pre-analyzed templates of actual note recordings.

Spectral Models attempt to characterize the sound in the frequency domain, which is more related to how our hearing system works. Roughly speaking, what Physical Models are to the sound generation mechanism, Spectral Models are to the sound perception process. This approach decomposes the sound in two parts: deterministic and stochastic (Serra and Smith, 1990). The deterministic part consists of multiple time-varying sinusoids, and the stochastic part is modeled as noise with a particular spectral shape. This synthesis technique is known as *Sinusoidal plus Residual*. The main advantage of this technique is the existence of analysis procedures that extract the synthesis parameters out of real sounds, thus being able to reproduce and modify (transform) actual sounds. The different components are added to a spectral frame. Then, a single IFFT for each frame is computed; and, finally the output is generated after windowing and the add-overlap

process. As we observe in the figure 2.5, the spectral synthesis needs various input parameters: sine frequencies, sine magnitudes, sine phases, and residual spectral data.

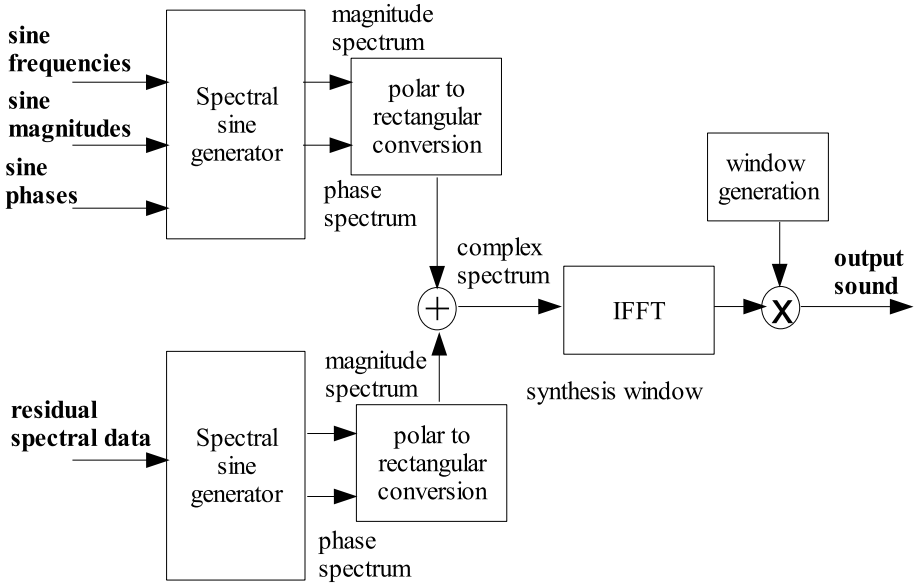

Figure 2.5: Block diagram of the spectral synthesis (Zölzer, 2002).

A remarkable advantage of Spectral Models over other synthesis techniques such as Physical Models is the fact that a number of musically meaningful parameters are more likely to have a perceptual or musical meaning (e.g. residual component amplitude ⇒ breathy amount). An example of a practical implementation of Spectral Models for saxophone synthesis is *SALTO* (Haas, 2001). A database (Timbre Template Database) holds pre-analyzed spectral sound segments, which are then used as anchor-points for generating a continuous timbre-space. This data consists of a complete frequency representation of the sound according to the Sinusoidal plus Residual Model. Concerning the interface, this system uses a Yamaha WX7 Breath Controller. MIDI controls are mapped to the synthesis control parameters, as shown in the table 2.1.

Sample-based During the 70's and 80's most commercial synthesizers implemented abstract techniques (FM, Phase-distortion, etc.). Sampling synthesis appeared later as an attempt to get a better sound quality for synthesizing acoustic instruments. The principle is to record with accuracy a large number of samples of an acoustic instrument. By means of a controller, usually a keyboard, the user triggers the playback of a sample. Although, in the 1970's there had appeared a pre-digital sampling synthesizer, the *Mellotron*, it was many years later that this technique became popular. Primarily, it was due to the larger memory capacity and the faster processors available.

A major drawback of the sampling synthesis technique, for imitating existing instruments, is its lack of the so-called "prosodic rules" for musical phrasing. Individual notes may sound like realistic

Initial MIDI Controller	*Synthesis Parameter*
Pitch (fingering)	Pitch
Attack Velocity	Attack Character Synthesis Volume
Note On	Note On
Continuous Controller	*Synthesis Parameter*
Lip Pressure	Pitch Modulation
Breath Speed	Volume Timbre Interpolation (Transition Recognition)
Note On	Note On

Table 2.1: MIDI Controllers (Breath Controller) from (Haas, 2001).

reproductions of traditional instrument tones. But when these tones are played in phrases, the nuances of the note-to-note transitions are missing. Partially, this problem is due to the inflexibility of the controllers. The degree of transformation will depend on the size of the samples database. Typically, note-to-note transitions use cross-fading; for transposition, the most common technique is resampling, which does not preserve timbre.

A different and conceptually new approach is granular synthesis. The sonic grains have a short duration (in the order of milliseconds) and they are shaped by an envelope, which can take different forms. The content of the grain, i.e. the waveform inside the grain, can be of two types: synthetic or sampled. In terms of control, granular synthesis requires a large number of parameters, since for each grain we need to specify starting time, amplitude, etc. Therefore, a higher-level control layer is necessary. Granular synthesis can be *synchronous*, *quasi-synchronous* or *asynchronous*, depending on the repetition pattern of the grains. A practical application of granular synthesis is the flexible generation of sound textures such as explosions, rain, breaking glass, and the crushing of rocks, to name a few.

Finally, a further step in sample-based techniques is *sample concatenation*, which might acquire more and more relevance in future synthesizers. In early sample-based techniques, transitions of two samples was done by cross-fading. This processing sounded unrealistic when imitating the phrasing of a musician. In sample-concatenation, the transition of two samples uses advanced signal processing techniques to assure a smooth and realistic sound. These techniques, such as the phase-locked vocoder (Laroche and Dolson, 1999), perform transposition and spectral envelope modifications with phase continuation. In terms of control, sample-concatenation systems use typically an internal score containing sample sequence and the transformations parameters. In a real-time situation, the internal score is created on-the-fly.

2.1.4.2 Mapping strategies

The performance interface of acoustic instruments is inherently bound up with the sound source (e.g. a guitar string). The connection between these two are complex and determined by physical laws. The situation is radically different for electronic instruments, where this relationship known as *mapping* has to be defined. The mapping layer is of high importance (Hunt et al., 2002) and is in fact non-trivial, and thus it must not be overlooked by the designers of new musical instruments.

Mapping has acquired some relevance in the past years, partially due to the growth of the computational power of real-time systems, which has widened the design of new musical instruments. Several research works (Métois, 1996; Wanderley, 2001) address the question of mapping from different perspectives. Basically, all these works try to explore the *art* of mapping in order to find a general model for digital instruments. One approach is to model a mapping consisting of multiple layers (Arfib et al., 2002). In the figure 2.6, we show an example of a two layer mapping. This give more flexibility, since for the same set of intermediate parameters and synthesis variables, the second layer is independent of the choice of controller being used. The same would be true in the other sense: for the same controller and set of parameters, multiple synthesis engines can be used just by adapting the second mapping layer, the first being constant. Arfib et al. (2002) proposes a three-layer mapping (i.e. a *gesture to gesture-perception* first layer, a one-to-one *gesture-perception to sound-perception* second layer, and a last *sound perception to synthesis parameters* layer).

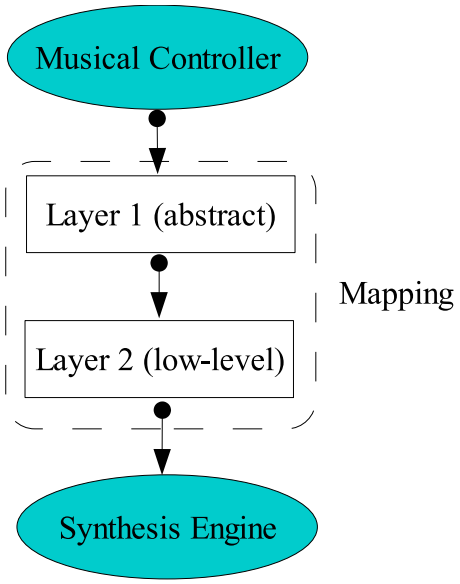

Figure 2.6: Block diagram of a two layer mapping, similar to (Arfib et al., 2002).

As Miranda and Wanderley (2006) point out, there are two main trends in the literature on mapping: *self-trained* (automatic) mapping, and *explicit* mapping. In the first case, the mapping

strategies derive from a model trained with machine learning techniques and a set of examples. Particularly interesting is their application in user-adapted systems, where from a few examples the system adapts to user behavior.

Explicit mapping strategies define a fixed relation of input parameters to synthesizer parameters. A more detailed study of explicit mapping devises four categories:

- one-to-one
- one-to-many
- many-to-one
- many-to-many

Concerning the design of mappings, Jordà (2005) argues the drawbacks of "splitting the chain" (decoupling between controller and synthesis engine) in creating new digital instruments. Gesture controllers are often designed without a clear idea of the sound output, which limit the degree of intimacy between the performer and the instrument. He proposes to take a more *holistic approach* that should consider the design of digital instruments as a whole.

Finally, another clear problem in defining mapping strategies is the difficulty of evaluation. In other words, to identify correct and incorrect mappings. In many cases, these mapping strategies are arbitrary, sometimes designed by a composer for a specific composition. Hence, in spite of the research carried out, an evolutionary path is difficult to discern in this topic. In recent years, some efforts attempted to find ways to standardize and formalize mappings in sound systems (Van Nort et al., 2004; Bevilacqua et al., 2005; Steiner, 2005). Further readings on mapping strategies are some publications by Hunt et al. (2002), Wanderley and Depalle (2004) or Verfaille and Wanderley (2005).

2.2 A survey of sound-driven sound systems

After a brief introduction to aspects of digital musical instruments, this section reviews various existing systems that are conceptually similar to the work presented in this dissertation: controlling sound synthesis with sound.

Historically, we can consider that the first attempt in using the voice for generating new sounds was the *Vocoder*. The *Vocoder* consists of a bank of filters spaced across the frequency band of interest. In real-time , the voice (modulator) is analyzed by the filter bank, and the output is applied to a voltage-controlled filter bank or an oscillator bank to produce a distorted reproduction of the original signal (carrier). A further extension is the *Phase-Vocoder* (Dolson, 1986), which is based on the Short-Time Fourier Transform (STFT) and allows many manipulations in the spectral domain, and was first used for Time-Scale and Pitch-Shift applications. A particular application of the Phase-Vocoder is called *cross-synthesis*. The result is a source sound (e.g voice) controlling

another sound (e.g. a cat's meow sound). It is based on the fact that we can multiply, point by point, the magnitude spectrum of two analyzed signals.

Among the surveyed systems, there is a disparity of musical objectives, implementations and contexts. It is worth mentioning that this classification does not intend to be an exhaustive list of all existing sound-controlled sound systems available. Instead, we selected some examples that, in our opinion, illustrate different perspectives of sound-driven synthesis.

2.2.1 Audio-to-MIDI converters

2.2.1.1 Principal characteristics

Converting an audio signal to MIDI has been traditionally an interesting goal for electronic musicians. The advantages of converting an audio stream to MIDI are multiple. Basically, transformations at a symbolic level are computationally cheap and will not affect the posterior audio quality. On the other hand, currently there are dozens of synthesizers available, both hardware and software, that are controllable with MIDI. We can differentiate two types of audio-to-midi conversion, real-time and off-line (or *transcription*). First, we find those systems that work converting on-the-fly an audio stream into a symbolic representation (e.g. MIDI messages), functioning as a controller. Second, *transcription* systems try to extract the musical information from an audio signal and put it in a symbolic form, often a musical score. The latest developments work with polyphonic audio signals and include multi-pitch estimation (Klapuri, 2006) or tonal description (Gómez, 2006). Actually, this is a research topic in its own right and goes beyond the scope of this work. Other potential applications of audio-to-MIDI are score-following (Puckette, 1995) or Query-by-Humming (Lesaffre et al., 2003).

2.2.1.2 Examples

Pioneer systems Historically, the first Audio-to-Midi hardware device that acquired notable popularity was the *PitchRider 4000*, which was introduced by IVL [16] in May 1986. In 1995, the American company *Opcode* released Studio Vision Pro 3.0, a professional MIDI sequencing and digital audio recording software, which included a novel Audio-to-MIDI converter.

State-of-the-art systems Some commercial systems are: *Epinoisis' Digital Ear*[17], *e-Xpressor's MIDI Voicer 2.0* [18] or WIDI [19]. Figure 2.7 shows the graphical user interface of Digital Ear. In the context of Query-by-Humming, recent studies (Clarisse et al., 2002) compared different commercially available Audio-to-MIDI systems with non-commercial libraries for melody extraction. Results

[16] IVL Audio Inc. `http://www.ivl.com`
[17] `http://www.digital-ear.com`
[18] `http://www.e-xpressor.com`
[19] `http://www.widisoft.com`

showed that commercial systems such as DigitalEar perform much worse than other implementations such as Pollastri (Haus G., 2001). In the same article, Clarisse et al. (2002) propose a new transcriber of singing sequences that was later integrated in the MAMI system (Lesaffre et al., 2003).

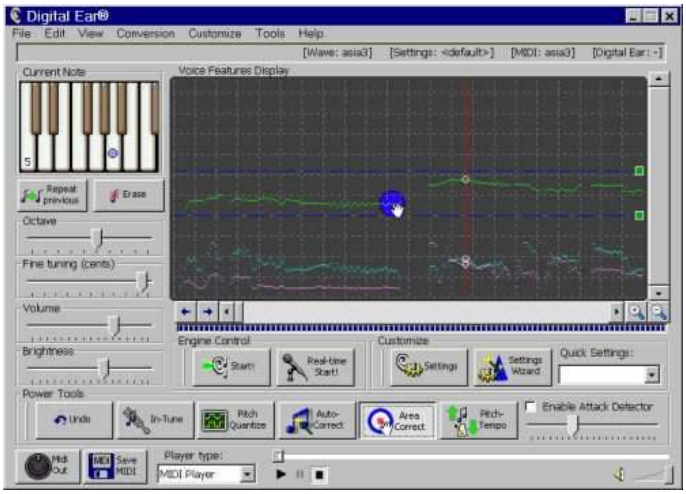

Figure 2.7: Graphical User Interface of Epinoisis' Digital Ear. This figure has the copyright of Epinoisis Software.

Other applications, like *Melodyne*[20], use the same concept of identifying notes from an audio stream, but in this case, the purpose is to transform the original audio using the extracted information. Commonly, after the analysis the user modifies the time-scale or the pitch of one or several notes.

2.2.2 Audio-driven synthesizers

2.2.2.1 Principal characteristics

With the increase of computational capacity, analysis/synthesis in real-time are possible. A common aspect of these systems is that they rely on perceptual parameters, namely: loudness, pitch and brightness. First attempts we found are from Wessel et al. (1998), which are based on timbre studies by Grey and Moorer (1977). Wessel utilizes Neural Networks to find appropriate mappings to the set of spectral frames. We present several systems that aim to control sound synthesis in real-time. In addition, we include in this list two interactive systems: a public installation and a network-collaborative system that are related to the concept of audio-driven synthesis.

[20] `http://www.celemony.com`

Analysis	*Synthesis*
Input Dynamics	Freq. Filter1
Input Timbre	Q Filter1
Input Pitch	Pitch Osc1

Table 2.2: Example of *Kantos mapping between analysis and synthesis parameters.*

2.2.2.2 Examples

MIT's *Singing Tree* Back in 1997, the Hyperinstruments Group at the Massachusetts Institute of Technology, developed and composed an interactive opera, calling it "The Brain Opera"[21]. This project was directed by Tod Machover, and included the development of six interfaces that combined sound and visuals in real-time. One of this interfaces is *The Singing Tree* (Oliver et al., 1997), which is related, to some extent, to the paradigm of this work: using the nuances of the voice for controlling synthesis. The quality of the voice is analyzed and a video sequence and a musical sequence are perturbed accordingly.

A relevant issue for us is the mapping strategy. Here, as long as the user maintains a steady pitch, the video and musical sequences proceed toward the goal; alternatively, if the user stops, the sequences reverse to an inanimate state. Nevertheless, other changes in the user's voice (loudness, pronounced vowel, etc.) alter also the sonic result. Focusing on the voice's analysis, various vocal parameters are extracted by using Eric Metois' DSP Toolkit (Métois, 1996).

Antares kantos 1.0 It is a commercial virtual instrument plug-in that can be integrated in most Digital Audio Workstations. Basically, *Kantos* uses relatively familiar synthesizer modules (oscillators, filters, LFO's, etc.), but instead of being controlled via keyboard or via MIDI, it is driven by an audio signal. Principally, this audio signal needs to be monophonic for getting acceptable results. The system consists of three functional blocks: an analysis stage, a modulation matrix, and a final synthesis stage. Each of these blocks provides the user with a set of controllable parameters (see table 2.2). Currently, *Kantos* development and commercialization has been discontinued.

MIT's Perceptual Timbre Synthesizer The first example is a system developed by the *Hyperinstruments Group* at the Massachusetts Institute of Technology, under the direction of Tod Machover and the system was developed by Jehan and Schoner (2001a), employing previous work done by Schoner (2000) and Métois (1996).

Basically, the goal of this system is to use perceptual information of an arbitrary input audio stream for controlling a synthesis engine. The musical gestures generated by bowing a violin should be consistently mapped to the new synthesized sound, and thus transmitting the violinist's expression. The synthesizer models and predicts in real-time the *timbre* of an instrument. It is based on the spectral analysis of natural sound recordings using the probabilistic inference framework

[21] The Brain Opera is currently installed at the House of Music in Vienna, Austria. `http://brainop.media.mit.edu`

Cluster-Weighted Modeling (Schoner, 2000). The timbre and sound synthesizer is controlled by the perceptual features pitch, loudness and brightness, which are extracted from the input audio. A crucial step in this system is the mapping. The predictor model outputs the most likely set of spectral parameters that are used in an additive synthesis approach to generate the new sound.

In order to improve the sound quality, a noise component is added, following the SMS method (Serra and Smith, 1990). For the noise component a second predictor is implemented, where the input vector consists of the perceptual parameters and, a nosiness estimator. The mapping process is, with Cluster-Weighted Modeling (CWM), probably the most novel part of the system. CWM was first introduced by Gershenfeld et al. (1999), and fully developed by Schoner (2000). It is a probabilistic modeling algorithm based on density estimation around Gaussian kernels.

Akademie Schloss Solitude's *Auracle* A more recent approach that uses the singing voice in an interactive environment is the *Auracle* system (Ramakrishnan et al., 2004), developed at the Akademie Schloss in Stuttgart, Germany[22]. In this case, instead of an interactive installation, this is a network-based instrument, which considers the participations of multiple users simultaneously. *Auracle* is a group instrument controlled by the voice that works over the Internet and has its origin in the interest of Max Neuhaus in participatory musical events.

For controlling the synthesis component, features of the incoming voice must be extracted. The considered features are: voiceness/unvoiceness, fundamental frequency, root-mean-square (RMS) and the first two formant frequencies. Auracle makes also use of the term *gesture* for segmenting voice signal; gestures are here defined as the utterance between silences longer than $200ms$. In a further step, envelopes of low-level features over a gesture are classified with Principal Component Analysis (PCA), achieving data reduction.

Yamaha's Easy Trumpet Yet another audio-driven synthesis system considered is Yamaha's Easy Trumpet (EZ-TP), released in 2006. In this digital musical instrument, the audio processing algorithms are licensed from Alto Research (Jameson, 2002). The EZ-TP device (see figure 2.8) allows to synthesize sound either by singing into the trumpet or by blowing as in a traditional trumpet jointly with trumpet valves for pitch changes. In both cases, it might be used a MIDI controller for external sound synthesis modules.

Initial user reviews praised two aspects of this product. First it provides trumpet players with a specific MIDI trumpet-controller, and second it provides the ability to control sound synthesis with singing. Unfortunately, we have not had the opportunity to test this system at the time of finishing the writing of this dissertation.

[22] `http://www.akademie-solitude.de`

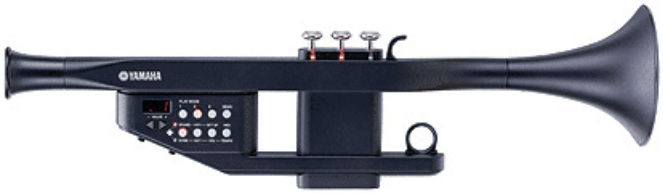

Figure 2.8: Yamaha EZ-TP. Users can sing in to produce trumpet sounds. This figure has the copyright of Yamaha Corp.

2.2.3 Morphing systems

2.2.3.1 Principal characteristics

In the context of spectral audio processing, *Morphing* (Cook, 1998) has appeared as a technique for generating new and never heard sounds. Actually, the concept of Morphing comes from the field of video processing, where an image is transformed sequentially into another. In audio processing, attempts for synthesizing sounds, whose acoustic characteristics lie between two real instruments' sounds, have been carried out. In the literature, this technique is also referred as Cross-Synthesis. Technically, morphing is based on the Phase Vocoder(Dolson, 1986). Starting from two audio samples, we can achieve a third sound whose perceptual characteristics are taken from a mix of the two original sounds. Among all existing audio morphing systems, we have selected here a number of relevant systems employing audio morphing.

2.2.3.2 Examples

Karaoke impersonation Conceived as a karaoke application, the goal of a system developed at the Universitat Pompeu Fabra was to morph the user voice with the original singer voice (Cano et al., 2000). For instance, one could sing an Elvis song with the voice of Elvis. Instead of doing voice transformation, this approach combined morphing with score-following (Loscos et al., 1999), since both the score and original singer's recoding were available. Despite the impressive results, the system was neither commercialized nor continued due to the cost of adding new karaoke titles to the system.

Prosoniq's Morph Among the commercial software that achieve audio morphing, PROSONIQ morph[23] was , in 2003, one of the first releases. Is is an extension of previous software for the Atari ST in 1983 and for Macintosh and the SGI in 1993.

[23] `http://www.prosoniq.com`

Larynxophone Also at the UPF's MTG lab, Loscos and Celma (2005); Loscos (2007) investigated morphing the voice with instrument sounds in the context of voice transformation. The *Larynxophone* permitted transforming the timbre of the user's voice with a trumpet, a violin and a guitar. From the voice signal, pitch and loudness were kept, while the spectral envelope was a combination of the voice and the instrument sample. The amount of voice and instrument timbre was controllable, thus, setting the timbre parameter 100% to instrument, led to a voice-driven instrument morphing effect.

It is worth to stress that voice-to-instrument morphing was the starting point of the research presented in this dissertation. Upon this, we approach from a broader perspective the use of the human voice as a control input in digital musical instruments.

2.2.4 Music Information Retrieval systems

2.2.4.1 Principal characteristics

There are a number of *Sound-driven sound systems* that also relate to the area of Music Information Retrieval (MIR). MIR addresses the automatic understanding of musical signals, and has multiple applications such as digital libraries searches, music transcription and music recommendation. In our context, the following systems borrow computational methods from MIR to, first, analyze the incoming sound, and then retrieve sounds from a database according to the previous results.

2.2.4.2 Examples

Meron's Singing Voice Synthesis In his thesis, Meron (1999) describes a high quality singing synthesis system based on sample selection. The goal of this system was to achieve natural sounding singing along with flexible control capabilities. Also the system should be able to reproduce different target singers with automatic training. Compared to, at its publication date, state-of-the-art systems, they argued that their system surpassed other singing voice synthesizers. Actually, the system is trained with the voice of the singer to be synthesized. It is able to produce more variations in voice quality since it takes advantage of a large training database of singing, containing examples of different variations. In terms of voice quality, Meron employs only the amplitude of the singer's formant, although they suggest that other voice parameters could be also implemented as described by Ding et al. (1996).

Query-by-humming One of the MIR flagship applications has been Query-by-Humming. This application has been technologically very challenging, and despite the large expectations for it, it has not yet proven to be realizable. Such systems have two distinct parts: the query and the target part. Users query the system by humming, whistling or singing (with or without lyrics); the system searches a database and retrieves a ranked list of closest audio files. As addressed in (Clarisse et al.,

2002), a main difficulty is to code consistently user's query, since it may contain errors of both rhythm and melody. Currently, one can find web portals[24] offering a Query-By-Humming feature to find new music, which uses technology from Fraunhofer IDMT[25].

BeatBoxing systems Kapur et al. (2004) make use of BeatBoxing to retrieve drum loops in a system targeted to DJs in live performances. Instead of humming a melody, users produce vocal sounds to imitate a rhythm or drum-loop. Based on the query, the system retrieves a list with the closest drum loops found in the database. In (Gillet and Richard, 2005), we can find a more detailed study on vocal queries using Support Vector Machines on a set of cepstral and spectral features. Compared to query-by-humming systems, in these cases queries are not based on melody information but on sound timbre properties. Also, commonly these systems distinguish between solely three types of sounds for transcribing sequences: bass drum, snare and hi-hat.

2.2.5 Audio-driven transformations

2.2.5.1 Principal characteristics

In the last category of *Sound-driven sound systems*, we look at those systems that transform a given sound using characteristics of a second audio stream as control source. This is similar to approaches such as Content-based Transformations (Amatriain et al., 2003) or Adaptive-Digital Audio Effects (Verfaille, 2003); transformation parameters derive from audio features. However, we do not consider one unique signal stream with adaptive transformations, but two streams: control audio and transformed audio. In the following subsection, we present two system that apply filter-based and tempo-based transformations respectively.

2.2.5.2 Examples

Filter-based: *Wah-wactor* The *Talkbox* was an invention of Bob Heil in 1971. In this device, sound was directed into the performer's mouth through a tube, and a microphone captured the sound modified by mouth's shape. Peter Frampton popularized the Talkbox in his albums *Frampton comes Alive!*. Aiming to bring the Talkbox in the digital domain, Loscos and Aussenac (2005) developed a voice-controlled wah-wah effect. An analysis algorithm based on the Fast Fourier Transform extracts one control parameter that is related to the formats structure. The system was primarily targeted to guitarists, who could control a wah-wah effect directly without the aid of any pedal. In addition, this effect can be used to any polyphonic sound source achieving nice results.

Tempo-based: *Groovator* A particular use case of voice control in another context than synthesis control was applied to tempo transformation. In the original implementation of *Groovator* (Janer

[24] Musicline, `http://www.musicline.de/de/melodiesuche`
[25] `http://www.idmt.fraunhofer.de`

et al., 2006b), the user can drive the tempo by means of a MIDI foot controller. A potential use case is a solo instrumentalist controlling his accompaniment with a foot pedal, thus freeing the musician from a fixed tempo. Other control possibilities of the system, as depicted in figure 2.9, were through a Graphical User Interface, or using an input audio stream. This system was extended to allow input audio streams to control tempo. A typical use case is live percussion, where the percussionist can change the tempo of an accompaniment audio loop dynamically with his performance. This use case is interesting because it is not intrusive, the percussionist can play his traditional instrument. A particular use case and the one that relates to this thesis is voice control where the rhythm could be extracted from a beat-boxing performance.

Preliminary user experiments showed limitations in using the *Groovator* in the performance-driven mode mentioned above. Mainly, the problems arises from the length of the tempo analysis window, set initially to 8 seconds in order to get a reliable estimation. It forces tempo either to be constant or to change slowly.

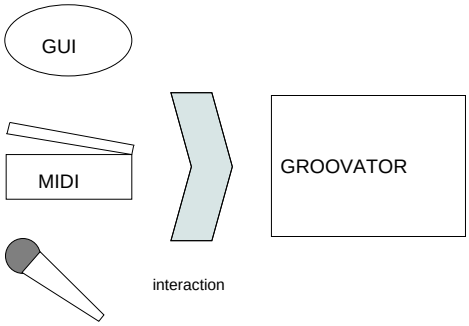

Figure 2.9: User interaction with the *Groovator* for the tempo transformation. *Top:* GUI controls; *middle:* foot-tapping with a MIDI controller; *bottom:* audio stream input (e.g beatboxing).

2.3 Voice Signal Analysis

In the previous sections, we have introduced general concepts of digital musical instruments, and surveyed systems that share the sound-driven sound synthesis paradigm. This section is devoted to the human voice signal, introducing the principal methods of voice analysis, and particularly singing voice analysis.

2.3.1 Physiology of voice production

We can define *voice* as the sound generated by our voice organ by means of an air-stream from the lungs, modified first by the vibrating vocal folds in the glottis, and then by the rest of the larynx, the pharynx, the mouth and some time by the nasal cavities. The voice is typically explained as a

source/filter system, where an excitation is filtered by the vocal tract resonances. The physiology of the voice is how we use the voice organ to produce sound. We will observe why the vocal folds start to vibrate, why they generate sounds and how the timbre characteristics of a voiced sounds depends on the shape of the tongue, the lips, and the jaw opening. A special attention will be put on the articulation, identifying the control possibilities of the singer.

2.3.1.1 Source and articulation

Three different mechanisms constitute the voice organ: the breathing apparatus, the vocal folds, and the vocal tract. The air is compressed in the lungs and then, an air-stream is generated passing through the glottis and the vocal tract. Looking at these systems from a functional point of view, we can describe them as *compression, phonation and articulation*. The compressed air produces an air-stream, and when it passes the vocal folds, they may start vibrating, opening and closing alternatively the passage for the air-stream. This way, the air pressure is raised above the glottis at regular intervals, generating an acoustic signal composed of variations in the air pressure. The duration of these intervals determine the vibration frequency of the vocal folds, which corresponds to the frequency of a certain perceived tone $freq = 1/T_{pulse}$.

In the vibration process of the vocal folds, the so-called Bernoulli force plays an important role. This force is activated when the air-stream passes through the glottis, generating an underpressure, which strives to close the glottis. But when the glottis is closed, the air pressure is higher below the glottis than above it. Since in case of phonation the vocal folds don't resist this air pressure difference, they open and allow a new air pulse to escape. Then the Bernoulli force comes again and shuts the glottis.

There are other factors that contribute to the vibration of the vocal folds, such as our muscles for adduction and abduction, or our nerve signals, but they are much too slow to produce vibrations of hundreds of cycles per second. Nevertheless, adduction and abduction muscles are significant in the control of the voice source. Adduction is required for phonations to take place, and abduction is necessary for both taking a breath and producing voiceless sounds. When we generate a voiced sound of a certain frequency, it was mentioned that the vocal folds are opening and closing at this precise rate. Moreover, we are able to change this frequency consciously, which is for us essential when dealing with the singing voice. Two factors affect the change of frequency: the overpressure of the air in the lungs *(subglottic pressure)*, and the langyreal musculature, which determines the length, tension, and vibration mass of the vocal folds. We will see later in more detail all aspects related to the control of the voice.

Thanks to photographic techniques, as is explained in (Sundberg, 1987), some aspects of the movement of the vocal folds were revealed. For lower frequencies, the glottal closure is initiated far down in the glottis. The closure rolls upward toward the upper part of the vocal folds, producing the so-called *mucosal wave*. On the other hand, when the phonation frequency is high, the vocal

folds are thin and tense, and no clear wave can be seen.

The range of the phonation frequency can go from $35Hz$ to $1500Hz$, although a person has a range of usually no more than 2 octaves. By observing the signal of the *Voice Source* in the spectral domain, we apprehend a more complex spectra than a single delta placed at the vibration frequency of the vocal folds. The vibration produces a series of harmonic partials which are multiples of the fundamental frequency or frequency of vibration f_0.

Measurements of the voice signal show that the level of these partials decrease by 12 dB/octave, therefore the spectrum of the voice source presents a negative slope in a logarithmic scale. However, this slope suffers from important deviations in the voice source spectrum of real subjects. Some experiments presented by Sundberg (1987), show that these deviations affect the difference of timbre between male and female voices. Also, variations of phonation frequency and loudness result also in deviations of the aforementioned slope.

We have seen how the vocal folds produce a pulsed excitation in form of variations of the air pressure. This glottal pulse passes through the vocal tract, which is in fact a resonant filter. The acoustic tube of the oropharynx and the various chambers of the naso-pharynx form a resonant system, which filters the glottal pulses, shaping the spectrum of the final output voice sound. This shaping is finally controlled by the so-called *articulation*. In a resonator, the sounds decays slowly, at the same time emphasizing some frequencies *(resonance frequencies)* and attenuating other (anti-resonances). In our case, the resonator is the vocal tract, and these resonances are called *formants*. The four or the five first formants are the most relevant in the vocal tract. The partials generated in the glottal oscillator will be affected in various ways by the formants, which raise or attenuate certain frequencies. Furthermore, the vocal tract is unique for every individual, thus the human voice is really representative of our personality.

The formant frequencies depend on the length and the shape of the vocal tract, defined the former as the distance between the glottis and the lips; the latter is determined by the positioning of the articulators: the lip and jaw openings, the tongue shape, the velum and the larynx. For instance, the length of the vocal tract is varied continuously when we speak or sing. By extending the length of the vocal tract, we can easily see that the formant frequencies will be lowered. For a given configuration of articulators, the transfer function of the formants is defined, and hence so is the spectrum envelope. As we will observe in the next section, during the action of speaking or singing, humans focus on the produced sound rather than on the position of the articulators.

2.3.1.2 Control abilities

We have already mentioned that the goal of this work is to understand the voice in terms of a valid musical controller. In this section, we review in more detail the parts of our phonation system that are controllable, and how they are involved in sound production. Later, it will help us in identifying potential control parameters to map to a synthesis engine.

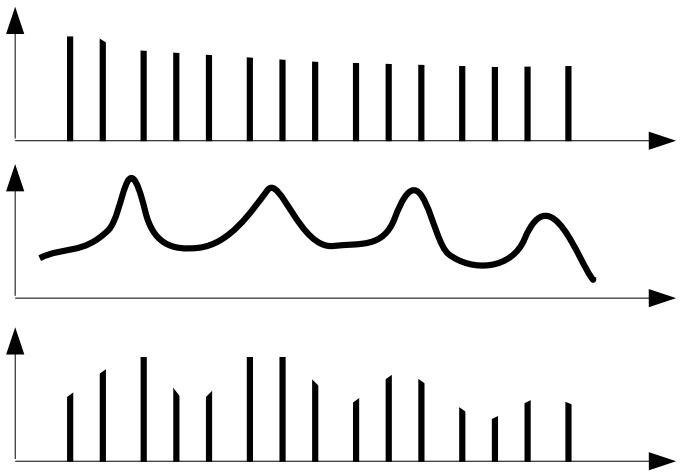

Figure 2.10: Approximative spectral representation of: glottal source (top), formants (middle) and output spectrum (bottom). Frequency *x-axis* and spectrum magnitude on the *y-axis*.

Previously, we presented the phonation organ as a system consisting of two parts: *Source* and *Articulation*. When we produce sound with our vocal apparatus, we are controlling elements of both parts simultaneously. Moreover, the necessary actions for this control are mostly performed unconsciously. For instance, for producing a vowel of a certain pitch, we must tense the vocal folds muscles with a precise amount (source control), and at the same time place the articulators in a particular position (articulation control), so that formants appear on specific frequencies. It is obvious that when we speak or sing, we do not think of the jaw's opening angle or muscle tension, instead we focus on the produced sound and adapt our phonation system unconsciously, partially thanks to the auditory feedback.

At one end of the vocal apparatus there is the breathing system, which plays also an important role in terms of control. The vocal folds need an overpressure from the respiratory mechanism in order to start vibrating. This overpressure, called *subglottic pressure* (Sundberg, 1987) is significant for the amplitude and also, to some degree for the phonation frequency. The importance of the breathing system is obvious, since these are two parameters (amplitude and pitch) over which a singer needs excellent control. Additionally, it explains the attention that is given to the respiratory system in both therapy and training of the singing voice.

Regarding control, basically we can identify three organs: sub-glottic pressure, the larynx and the vocal tract. The sub-glottic pressure, air-pressure in the lungs, is raised when the abdominal muscles contract, and is related to the phonation loudness. The sub-glottic pressure is a parameter that can vary quickly, and it can depend on factors such as the type of singing, tempo, etc. We will consider the breathing system, and more precisely the sub-glottic pressure the main parameter for controlling the phonation and loudness.

In the larynx, we find the functional part responsible for the oscillation: the vocal folds. The interaction between a given air-pressure and the cords that form the vocal folds, generates a modulation of the air-pressure that results in phonation. The phonation takes place when the laryngeal muscles are tensed (*adduction*), the vocal folds vibrate at a given frequency which corresponds with the frequency of the modulated air-pressure, and therefore with the frequency of the generated sound spectrum of the glottal source, formants and output spectrum. Basically, when changing the fundamental frequency (*f0*), the laryngeal musculature determines the length, tension and vibrating mass of the vocal folds. For raising or lowering *f0*, the musculature determines the properties of the vocal folds. They can be stretched to different degrees, becoming thinner and more tensed. In this case, the vocal folds will vibrate with higher frequency, and thus produce a tone with higher frequency. On the other hand, we should also consider that if the sub-glottic pressure is raised, the phonation frequency will also increase, although not excessively.

Another important issue related to voice control that takes place in the larynx is the generation of vocal disorders. When we talk of vocal disorders, we consider both unintentionally and intentionally produced disorders. The former is widely studied in medicine, and is treated as a misfunction of the voice organ. We are more interested in the latter, and in which way we can control voice disorders. A clear example in the singing voice is the *growl* effect. In jazz or popular music, we find often singers that alter his or her voice in order to increase the expression by producing a rough sound. Another example, is the *breathy* effect. In this case, we can modify the tension of the vocal folds, lowering the amount of vibration, while maintaining air-flow and pitch.

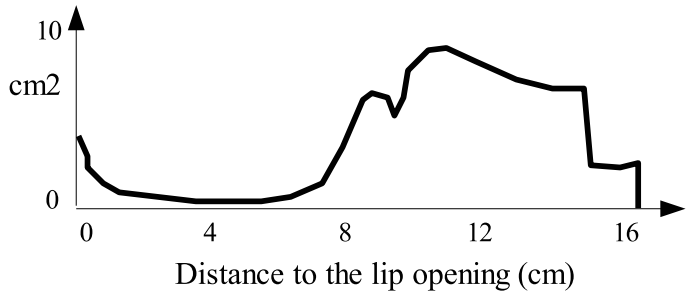

Figure 2.11: Approximative representation of the cross sectional area function of the vocal tract for the vowel */i/*, as shown in (Sundberg, 1987).

Finally, we find the control of the voice color. As we have mentioned, the formant frequencies depend on the length and the shape of the vocal tract. The *vocal tract length* is defined as the distance between the glottis and the lips. The vocal tract can be considered as a tube with variable cross-sectional area. Therefore, the shape of the vocal tract can be determined by an area function along the longitudinal axis, as we can observe in figure 2.11.

In an average adult male, the vocal tract length is between 17 cm. and 20 cm, but it depends on the individual morphology. This is the first parameter that we can control. The larynx can be

raised and lowered, extending the vocal tract length. Additionally, we can modify the mouth corners by protruding the lips. These variations in the vocal tract length affect the position of the formant frequencies. The longer the vocal tract length, the lower the formant frequencies. Regarding the shape of the vocal tract, the area function is determined by the position of the *articulators*: jaw opening, lips, tongue, velum and the larynx. In general, any movement of the articulators affects the frequencies of the formants. The first formant frequency is sensitive to the jaw opening. The shape of the tongue is mostly responsible for the second formant. The third formants is particularly sensitive to the cavity directly behind the incisors. If this cavity is large, the third formant frequency decreases. However, when we speak we are obviously not aware of the position of the articulators. Instead, we have patterns associated to different phonemes, and intuitively we apply one of these patterns. These patterns are not identical in all languages.

Focusing on voiced phonemes, each articulation corresponds to the formant frequencies characteristic of that phoneme. Looking only at vowels, the two lowest formant frequencies are sufficient for determining a vowel. Furthermore, vowels can be observed in a two-dimensional plane with the first and seconds formants frequencies as axes. The figure 2.12 shows the vowels scattered along a triangular contour. The vowels */a/*, */i/* and */u/* are the corners of the triangle. As we have mentioned, the articulators are responsible for: first, to characterize our personal voice's timbre; and second, to determine the produced timbre (e.g. vowel). Also, we have seen that the voice production is controlled in different ways in three different parts of the phonation system: breathing, vocal folds and vocal tract.

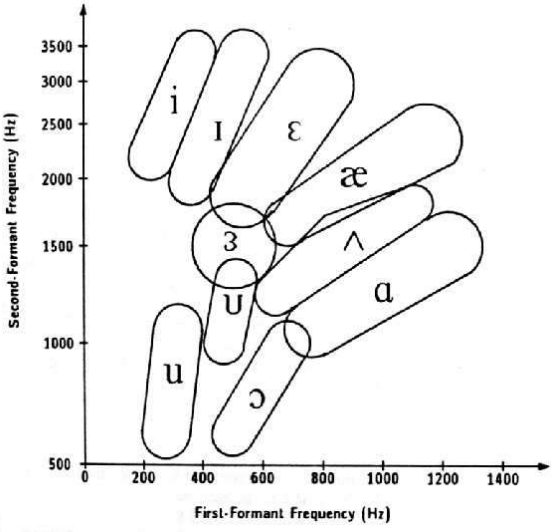

Figure 2.12: Vowels as a function of first and second formant. This figure is extracted from (O'Shaughnessy, 1987) and has the corresponding copyright.

2.3.2 Time-Frequency techniques

The computational analysis of music makes use of Digital Signal Processing techniques to extract features that describe audio signals. An audio signal can be equally represented in the time domain as a sequence of samples (*waveform*) or in the frequency domain as a sequence of spectra (*spectrogram*). In this section, we review some common techniques used by researchers to extract additional information from audio signal in both domains. Some features such as fundamental frequency, can be estimated in both domains, while other features only are calculated in one domain. Usually, both are complementary.

In practice, an audio signal is divided in blocks of samples (of around $30ms$) that can overlap. A number of features is computed for each block, commonly referred to as a *frame*. There has been in the past attempts toward a standard format for describing audio signals such as SDIF (Wright et al., 2000)[26] and MPEG7 (Salembier et al., 2002). In this latter format, features were classified as low-, or high-level features depending on whether they refer to physical properties of the signal, or to more abstract and musical concepts (Herrera et al., 1999). A very complete compendium of descriptors in both domains in the context of MPEG7 can be also found in Peeters (2003). Although in both cases, after its appearance several applications supported those formats (Celma et al., 2004), none of them have succeed as a standard format for audio signals descriptions.

2.3.2.1 Time domain techniques

Working in the time domain, the audio signal consists of a sequence of samples at a given sample rate. Usual sample rates for music signals are $22.050, 44.1, 48, 96$ or $192kHz$. Extracting features in the time-domain involves a processing directly the waveform, although a single descriptor value can represent a block of samples (or *frame*), as already explained. Below, we describe some of the most common techniques in time-domain.

Root-Mean Square It describes basically the power envelope of the audio signal. It is calculated by squaring all values within a block of samples, and taking the root of the mean. Taking into account auditory perception, this could be extended to measurements of a model of "loudness" (e.g. perceived volume) using filter curves at different frequencies.

Autocorrelation *Autocorrelation* is used to calculate periodicity information, (e.g. pitch). For a given time lag j, the autocorrelation gives the amount of self-similarity of a time-shifted version of itself. Thus, when j coincides with the samples per period of a quasi-periodic signal, the autocorrelation value is likely to be high.

[26] SDIF stands for Sound Description Interchange Format.

Zero-crossing rate It counts the times that the waveform crosses the 0 level. It gives a measurement of *noisiness* or *brightness*. In speech, for instance, it is an easy way to discriminate fricatives such as "s" or "f" from voiced sounds.

Comb-Filter bank A bank of comb-filters can be used to extract the periodicity of a monophonic signal. In this case, each filter attenuates frequencies that are not multiples of the fundamental. Therefore, having a bank of several comb-filters, we can estimate the fundamental frequency of the incoming signal. Likewise, comb filters can also be used to extract the amount of *breathines* in a signal. In this case, knowing *a priori* the fundamental frequency and tuning the comb filter, we suppress the energy due to the harmonics and calculate the remaining energy to estimate *breathiness*.

Pulse-Onset detection This technique is specific for voice signals and is useful for applying transformations such as transposition and timbre mapping using PSOLA (Moulines and Charpentier, 1990). Here, the unique goal is to set time marks at the glottal opening times (Pulse Onset). Peeters (2001) describes and compares several methods for pulse onset extraction based on the group phase delay.

2.3.2.2 Spectral domain techniques

Since the early 90's Spectral Processing has become more and more common in the field of Sound and Music Computing. An explanation for this success is two fold. First, its relation with psychoacoustics, since our auditory system works internally similar to a spectrum analyzer; and second, the speed of current processors that allow this type of processing in real-time.

Short-Term Fourier Transform There are several methods for analyzing the spectrum of a signal. Here, we look at methods that are based on the Fourier Transform, which models the input sound as a sum of harmonically related sinusoids. To adapt Fourier analysis to the practical world, dealing with sampled time-varying signals, the *Short-Time Fourier Transform* (STFT) is used. A complete process of the STFT includes windowing the input signal and calculating the DFT (Discrete Fourier Transform), which yields the magnitude and phases of the spectrum. For efficiency, the FFT (Fast Fourier Transform) technique is commonly employed for computing the STFT. The parameters of this process will determine the results. Configuration parameters include: window size, window type, window overlap and zero-padding[27].

Spectral features extraction Once having transformed our audio signal into a set of spectral coefficients, we can use a number of functions to extract *spectral features*. Those can give us hints about properties of the signal that can not be achieved in the time domain. Peeters (2003) reported a

[27] A detailed explanation of the influence of these parameters can be found in general Digital Signal Processing books such as (Proakis and Manolakis, 1983)

Spectral Features
Spectral Shape
Spectral centroid
Spectral spread
Spectral skewness
Spectral kurtosis
Spectral slope
Spectral decrease
Spectral rolloff
Spectral variation

Table 2.3: List of Spectral Features as suggested by Peeters (2003).

Harmonic Features
Fundamental frequency
Fundamental fr. Modulation (frequency, amplitude)
Noisiness
Inharmonicity
Harmonic Spectral Deviation
Odd to Even Harmonic Ratio

Table 2.4: List of Harmonic Features as suggested by Peeters (2003).

complete list of audio features in the context of the CUIDADO IST Project [28]. In his report, Peeters enumerate the following *spectral features*(see table 2.3) that can be calculated from the STFT.

Fundamental frequency estimation Several methods for fundamental frequency estimation have been proposed in both frequency and time domain. In the frequency domain, Maher and Beauchamp (1994) proposed the Two-Way Mismatch technique. In this procedure, the estimated F_0 is chosen as to minimize differences between the measured partials (detected peaks), and the series of harmonic frequencies generated by F_0. Cano (1998) introduced the following modifications: pitch dependent analysis window, optimization of F_0 candidates, and optimization of the Peak Selection. In addition to the fundamental frequency, in (Peeters, 2003) we find a number of harmonic features that describe characteristics of harmonic signals (see table 2.4).

2.3.3 Singing voice analysis

So far, we have learned on one hand, the voice production mechanism and, on the other hand, common techniques for processing audio signals in both time and frequency domains. These techniques will be of great importance for analyzing voice signals. We focus now on the analysis of the singing voice. First, we survey existing techniques in the field of speech processing that can

[28]CUIDADO (IST-1999-20194) stands for Content-based Unified Interfaces and Descriptors for Audio/music Databases

be useful for singing voice analysis. Next, and going specifically into the singing voice, we look at models for analysis/synthesis. Finally, we have a look at some work related to expressive modeling of singing performances. This latter research can be used either for singing voice synthesis or automatic performance description.

2.3.3.1 Speech analysis techniques

At this point, we can not forget that most of the voice research carried out is in the field of *Speech Processing*. Obviously, the impact of telephony and communications in general motivated the rapid emergence of this field. While during the initial years, speech and singing voice processing shared the same techniques and resources, nowadays they are considered as two completely different research fields. Observing and comparing speech and singing, we identify some salient differences: average pitch; pitch and loudness variations; and duration of vowels and consonants. However, we find it worthwhile to survey in this section some Speech Processing techniques that will be of interest in our study of the singing voice for controlling sound synthesis. Lots of resources are available on the ISCA's website [29].

Mel-Frequency Cepstrum Coefficients (MFCC) Speech recognition systems combine acoustic and language models to convert voice utterances to a sequence of words. In most systems, the former model is a Hidden Markov Model that estimates a phoneme from a vector of voice features. The voice signal is divided in blocks of samples, and a vector of voice features is computed. Obviously, these features must convey timbre information of the phoneme. For this purpose, the Mel-Frequency Cepstrum Coefficients (MFCC) are widely employed (Davis and Mermelstein, 1980). The main advantage of MFCC is that they are based on auditory perception, by calculating spectral coefficients on a Mel-scale[30]. After calculating the FFT, amplitudes are mapped to a filter-bank on the Mel-scale. Finally, the MFCCs are taken from the Discrete Cosine Transform of these amplitudes. Usually, the number of coefficients are 12 or 13.

As we will see in section 4, for controlling a singing voice synthesizer we need *Phonetic Alignment*. This task is accomplished by calculating the MFCC's and using *Hidden Markov Models* in a *Finite States Network*.

Formant tracking Two techniques commonly used in formant estimation are Linear Prediction and Cepstral Analysis. Formant analysis can be traced back to the 1950's, and there are various existing techniques. Basically, these techniques achieve different types of "peak-picking" in the spectral domain. Some try to identify formants frequencies either directly from the magnitude of the Discrete Fourier Transform, from a smoothed spectrum such as Cepstral Analysis or taking the frequency of the poles in Linear Prediction analysis

[29] `http://www.isca-speech.org`
[30] The Mel-scale divides the spectrum in frequency bands based on our auditory perception.

Additionally, it is important to mention that the formant analysis of the singing voice has implicitly some added difficulties. The major difference with speech is that singing voice has a considerable higher pitch. Formant analysis needs a low pitch in order to have the formant structure with enough resolution. For instance, if the pitch is about $500Hz$, only one harmonic component will be found within the first formant frequency region (typically, from 300 to $1000Hz$ for women). Thus, it will be not feasible to determine the formant with just one peak value.

Linear Prediction Linear Prediction (LP) is a popular technique for identifying linear systems. Regarding speech processing, its applications are analysis, compression and synthesis. Linear prediction models the human vocal tract as an infinite impulse response (IIR) system that produces the speech signal. For vowel sounds and other voiced regions of speech, which have a resonant structure and high degree of similarity over time shifts that are multiples of their pitch period, this modeling produces an efficient representation of the sound (see figure 2.13).

The linear prediction problem can be stated as finding the coefficients a_k which result in the best prediction (which minimizes mean-squared prediction error) of the speech sample $s[n]$ in terms of the past samples $s[n-k], k = \{1..P\}$. The predicted sample $\hat{s}[n]$ is then given by Rabiner and Juang. (1993) as,

$$\hat{s}[n] = \sum_{k=1}^{P} a_k s[n-k] \tag{2.3}$$

where P is the number of past samples of $s[n]$ which we wish to examine. Next, we derive the frequency response of the system in terms of the prediction coefficients a_k. The systems needs to be excited with a flat spectrum signal $e[n]$, either noise or short impulses with $E(z) = 1$. In equation 2.3, when the predicted sample equals the actual signal (i.e., $\hat{s}[n]] = s[n]$), we have

$$S[z] = \sum_{k=1}^{P} a_k S[z] z^{-k} \tag{2.4}$$

$$S[z] = \frac{1}{1 - \sum_{k=1}^{P} a_k z^{-k}} \tag{2.5}$$

Cepstral Analysis The technique was developed by Rabiner and Schafer based on *cepstrum analysis* and the chirp-z transform (Schafer and Rabiner, 1970). This model includes the Glottal Source $G(z)$ and Radiation Load Spectra $R(z)$, which jointly with the Vocal Tract transfer function $V(z)$, will determine the frequency response of voiced sounds. As Schafer and Rabiner (1970) describe, the entire model can be expressed as,

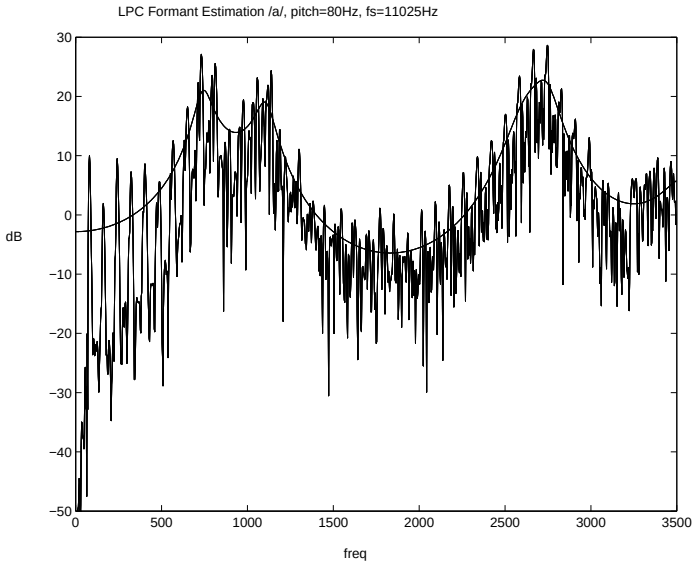

Figure 2.13: Spectrum of the vowel /a/ with $pitch = 80Hz$. Superimposed, the LPC approximation with 14 coefficients of the spectral envelope.

$$H_{system}(z) = G(z)V(z)R(z) = (G(z)R(z))\prod_{i=1}^{4} H_i(z) \tag{2.6}$$

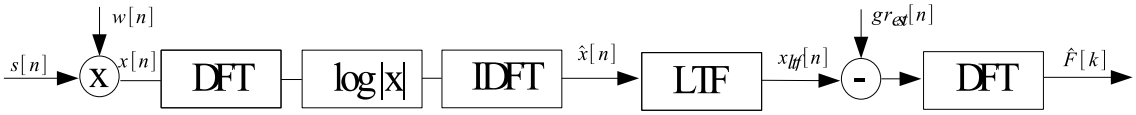

Figure 2.14: Cepstral Analysis. Block diagram of the entire algorithm, with the Discrete Fourier Transform (DFT), logarithmic operator, Inverse DFT and the Low-Time Filtering (LTF).

The Discrete Fourier Transform of the input signal is computed using the FFT algorithm. Next, the log of the FFT's magnitude is calculated, and finally, the inverse DFT gives the *real cepstrum* $\hat{x}[n]$. Figure 2.15 depicts the resulting spectral envelope.

$$\hat{x}[n] = IDFT\{\log|X(z)|\} = \hat{p}_\omega[n] + \hat{gr}[n] + \sum_{k=1}^{4} \hat{h}_i[n] \tag{2.7}$$

Dynamic Programming Various applications make use of Dynamic Programming (DP) in the field of Speech Processing. Its main feature is to give robustness to the estimation of time-varying processes. In formant tracking (Talkin, 1987; Xia and Espy-Wilson, 2000), given a set of formant

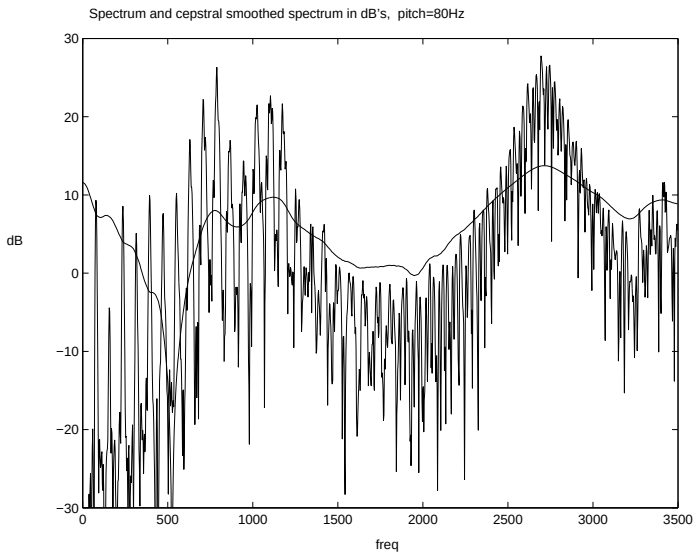

Figure 2.15: Spectrum and cepstrally smoothed spectrum of the vowel /a/ with $pitch = 80Hz$

candidates at each frame of the speech signal, DP utilizes the evaluation of transition costs between two consecutive frames. For phonetic alignment tasks, DP is used for determining the phonetic boundaries given a Finite State Network (FSN). As we explain in chapter 4, this technique is employed in the performance-driven singing voice synthesis.

Prosody contour Prosody modeling is principally used in Text-To-Speech (TTS) applications. It is of great importance in synthesizing natural voices, since prosody conveys also semantic information. Hence, researchers in speech have addressed the analysis of prosody extensively. Moreover, one can find international conferences explicitly dedicated on prosody such as *Speech Prosody* held by ISCA [31] Current trends in prosody include modeling with parametric curves (Cardenoso-Payo and Escudero-Mancebo, 2002) and studying the role of prosody in emotional speech (Roach et al., 1998). Concerning the singing voice, prosody does not convey semantic information but rather melodic information. Hence, in the singing voice, the term prosody is rarely used.

2.3.3.2 Models for singing voice analysis/synthesis

In this section, we review different models for the singing voice. Although our study focuses on the voice's analysis, these models were mostly conceived for Analysis/Synthesis systems, the goal of many scientists and engineers being to synthesize the singing voice. Other relevant models particularly oriented to the synthesis of the singing voice are not discussed here. In this latter category we find the FOF (Formant Wave Functions) (Rodet, 1984), which was used in the IRCAM's singing voice

[31] International Speech Communication Association, `http://www.isca-speech.org`

synthesizer called Chant. We look at three approaches that tackle the analysis of the singing voice from different viewpoints:

- Formant Filter model
- Acoustic Tube model
- Sinusoidal model

Formant Filter model Traditionally, the voice has been viewed as a source/filter system. An early but still prominent parametric model is the one proposed by Fant (1960). By means of different sources and filters, we are able to generate a wide range of sounds with our voice organ. The *Formant Filter* model aims at identifying each filters' parameters and sources' parameters. As we have already mentioned, the filter is composed of a number of resonances existing in the vocal tract. Each one of these resonances is one of the so-called formants. The main issue in the *Formant Filter* model is to parameterize correctly all resonances. In this model, three parameters characterize a formant: formant frequency, bandwidth and amplitude. Usually, a filter consisting of three formants is sufficient for recognizing a vowel. An attractive feature of the Formant Filter model is that Fourier or LPC (see 2.3.3.1) analysis can be employed to automatically extract the parameters. However, the source can have two different natures: voiced or unvoiced. Simplifying, the former is represented by its pitch and its amplitude, while the latter is characterized by its amplitude. This model has been used in voice synthesis since the early 70's.

Acoustic tube model This model can be considered as a *Physical Model*, since it simulates acoustically the vocal tract. In section 2.3.1, we have seen that an air-flow (modulated in the glottis), passes through the vocal tract producing sound. We have seen also, that the sound's timbre is determined by the shape of the vocal tract. In this model, the vocal tract is considered as an acoustic tube that modifies a sound wave with a particular transfer function. The transfer function is simulated by solving the one dimensional wave equation inside a tube. The justification for approximating the wave equation to one dimension is that the vocal tract's length is larger than the width in any position. Therefore, only the longitudinal modes are relevant for frequencies up to $4kHz$ (Cook, 1998).

Previous work with the acoustic tube model in the Speech Synthesis field was carried out at Bell Labs. Kelly and Lochbaum (1962) developed a model for a smoothly varying tube, which consisted of approximating the tube by several cylindrical sections of different cross-sectional area. This model can be simulated by a digital WaveGuide Filter (WGF) (Cook, 1991; Smith, 1992). The tube is divided into sections of the same length. The acoustic impedance of each tube section is a function of the cross-sectional area, which results from physical tract measurements. The filter has a ladder filter structure, with scattering junctions connecting the different tube sections. The

scattering junctions is known as the Kelly-Lochbaum junction (Kelly and Lochbaum, 1962). The figure 2.16 shows the acoustic tube in its different forms: smooth varying, sampled version and the digital filter simulation. This model has been used in the implementation of the singing synthesizer SPASM (Singing Physical Articulatory Synthesis Model) (Cook, 1992). SPASM provided a graphical interactive environment, in which the user could change the physical parameters of the vocal tract.

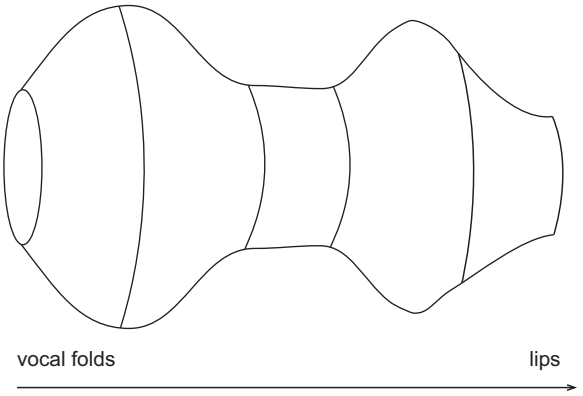

Figure 2.16: Acoustic tube model.

Sinusoidal model The third approach works in the spectral domain, where the voice signal is modeled as a sum of sinusoidal components, as shown in equation 2.8.

$$s(t) = \sum_{r=1}^{R} A_r(t)\ cos(\phi_r(t)) \tag{2.8}$$

In order to get the representation of the sound, one needs to estimate the amplitude, frequency and phase of each sinusoidal. First proposed for speech signals by McAulay and Quatieri (1984), this estimation is based on the STFT (Short-Time Fourier Transform) of the sound, detecting then a number of prominent spectral peaks. An improvement to this model is the *Sinusoidal plus Residual Model* (Serra and Smith, 1990) which models noise components of the sound. In addition to the sinusoidal components, the analysis extracts a residual signal $e(t)$, which in fact is assumed to be a stochastic signal. In order to model the signal $e(t)$, we can use white noise filtered by a time-varying filter. For identifying the characteristics of the filter, we can approximate the magnitude spectrum with connected linear segments. Another method is to perform an LPC analysis of the $e(t)$ signal, and extract the filter's coefficients. Due to its flexibility, this model is useful for applying transformations such as pitch transposition, morphing, time scaling, etc. It achieves remarkable sound quality particularity for pseudo-harmonic sounds. A more detailed explanation of the Sinusoidal plus Residual Model is found in (Amatriain et al., 2002), including some practical applications. The most relevant processes are peak detection, pitch estimation and peak tracking.

2.3.3.3 Performance description

Besides the singing voice analysis at a signal level, other approaches look at the expressive qualities of the singing voice from a musical perspective. We can devise two main direction in describing singing voice expression, those based on artificial intelligence and those based on heuristic rules. The former learns automatically from a set of annotated examples, while in the latter human experts define a number of rules based on their knowledge. Singing voice description has been applied to the problems of performer identification, performance rating, mood transformation or expressive singing voice synthesis.

Widmer and Goebl (2004) gives an extensive look at computational models for expressive performance based on machine learning. From an heuristic perspective, interesting work has been carried out at KTH-Royal Institute of Technology [32] in Stockolm. Since early research in the eighties by Sundberg et al. (1983), they have been developing a set of rules that are able to give expression to a written score as if a human was playing. In one of the latests developments (Friberg, 2006), performances can be "conducted" in real-time, controlling its expression by means of high-level concepts. Ongoing research on rules-based performance systems is detailed in (Friberg et al., 2006).

Specifically related to the singing voice, at UPF-MTG they have developed a system for an objective analysis and evaluation of a singing performance (Mayor et al., 2006). Such system is focused on the area of edutainment, and it has been used to develop a karaoke game and a training tool for singing students. More oriented to the description and modeling of performances, Maestre's approach (Maestre, 2006) aims at defining a grammar of performance gestures. Then, from a score and a performance recording, the system is able to characterize the performance using Hidden Markov Models and a previously trained model.

2.4 Summary

This chapter was devoted to review previous research related to voice-driven synthesis. We addressed the design of digital instruments, focusing on control aspects. Understanding the process involved in creating new digital instruments is a necessary step. Concerning the mappings, two concepts seem to be appropriate in our approach. On the one hand, to use multi-layer mappings (Arfib et al., 2002) in order to permit a variety of instrument synthesis and/or synthesis techniques without having to redesign the complete system. On the other hand, to take an holistic approach when designing the mappings, i.e. to build a closed system from input to synthesis, being aware of the produced sound at all stages of the mappings design.

We looked at several systems that share the same paradigm of a sound-driven sound system. Although, some of the presented systems are not strictly sound-controlled synthesizers, we found it appropriate to give a general overview of the state of the art in related musical systems.

[32] `http://www.speech.kth.se`

Since, the primary goal of our research is to exploit the expressive possibilities of the voice in a sound synthesis context, it cannot be accomplished without a prior knowledge of the characteristics of the human voice, and in particular of the science of the singing voice. In terms of signal processing, we looked at different techniques for feature extraction both in time and frequency domains, as well as different models for signing voice analysis/synthesis. The next chapter proceeds towards the use of the voice as a controller, in what we have called singing-driven interfaces. We present some preliminary case studies and experiments about the characteristics of the voice when imitating instruments; the vocal control of synthesizers, as well as the limitations of existing pitch-to-MIDI approaches for this task.

Chapter 3

The Singing Voice as a Controller

After reviewing previous work related to our topic, this chapter poses several questions about the vocal control of synthesizers, and the different points of view from which it can be addressed. A first step in this process is to identify the limitations of existing audio-to-MIDI (also referred as pitch-to-MIDI) systems. Following steps include three preliminary experiments, as well as two studies on vocal imitation of musical instruments. The conclusions of this chapter will determine the approaches taken in the remaining chapters.

3.1 Introduction

Singing voice qualities have been exploited in music since ancient times. It possesses extraordinary expressive possibilities, and moreover one can say the the singing voice is the first and most universal musical instrument. It has been present along music history, spanning different historical periods and cultural contexts (from popular music to refined artistic styles). It is not the aim of this research to study qualities of the singing voice as a musical instrument. Rather, we benefit of the high degree of expression and the nuances of the singing voice in order to exploit it as a musical controller for sound synthesizers.

As revealed in the recent roadmap on Sound and Music Computing Research (Serra et al., 2007), the weakness of new digital instruments are mainly in the control side. Our research can be observed as an attempt to overcome the mentioned limitations, proposing the voice as a valid musical controller[1]. Digital musical instruments permits a wide range of sounds while enabling a high degree of control over the sound generation process. It overcomes physical limitations of acoustic instruments in terms of control, and overcomes the limitations of analog synthesizers in terms of the

[1]Likewise, other developments use the voice as input device in other contexts. For example in video games, a microphone can replaces the traditional joystick as a controller. In (Hämäläinen et al., 2004), example games are presented where the pitch controls the player position.

sonic possibilities of the synthesized sound. The flexibility of control has led to the emergence of numerous musical controllers.

Also, in the past decades many controllers appeared with the aim to provide new forms of musical expression beyond the traditional instruments, and in particular linked to computer music. One of these systems are gestural controllers, which rely on the idea that body gestures are an appropriate form of controlling the generation of music. On the one hand, gestural controllers permit to combine music with other artistic disciplines such as dance. On the other hand, we believe that there is still much work to be done in intimately joining body gestures and sound synthesis. Recent works in the field of Musicology are in the direction of music-related body gestures, studying the couplings of body actions and sound (Jensenius, 2007), and considering emotive aspects as in the embodied music cognition approach (Leman, 2007). The latter fosters the relationship between performer's intention and the body movements. We can partially incorporate ideas of embodied music cognition in our research. For instance, when we talk of *Vocal Gestures* in chapter 5, we consider actually the movements in our phonation system that have a musical function, i.e. the musical properties conveyed by the voice signal.

Also in the area of Neuropsychology, recent works have direct implications in our research topic. Dalla Bella et al. (2007) carried out experimental studies demonstrating that most humans can sing(!); that is, to carry a tune without problems. Only 10% of general participants performed very poorly when singing. It seems clear that on a violin, only trained musicians are able to carry a tune. We can relate it to the concept of learning curve and instrument *musical efficiency*, as pointed out by Jordà (2005)[2]. On the one hand, the results presented by Dalla Bella et al. (2007), should encourage us to design tools that allow people to play any other instrument by singing. On the other hand, our voice-driven synthesis approach can benefit from the complexity of the singing voice, where skilled performers/users will have a more precise control of the synthesized instrument sound (e.g. achieving vibrato, portamento, etc.). Moreover, in terms of musical expression, voice can be very useful in the context of sound synthesis design, as Pressing (1992) indicates:

> One important resource in designing such expressivity is to use one's own voice to sing the expression in the part. Event if the sound quality is beyond the powers of your (or perhaps anyone's) voice, its time shaping may be imitable. As a general rule, if expressive control can be sung, it will soon be found to be playable.

As we cover later in this chapter, this "sung imitation" conveys valuable information that can be used to control a synthesis engine.

[2] Jordà (2005) shows a figure where the learning curve and the complexity of various instruments (kazoo, kalimba, violin and piano) are compared. If we had to add the singing voice in this graph, one could say that the singing voice has a low entry fee, but it is an insturment that can reach a high degree of complexity.

3.1.1 Voice transformation or vocal control?

In the context of digital musical instruments, as seen in section 2.1.1, our objective is to use the voice signal as a "musical input" that controls a sound generator. At this point, a significant question arises: are we *controlling* sound synthesis or are we doing sound *transformation ?*

There is not an evident answer, but addressing this question will put us into context, and additionally, it will help us to better define the characteristics of the system. This discussion explores the evolution of some electronic musical devices, such as synthesizers or sound processors. We try to point out, concepts and techniques in order to clarify this question.

In fact, we can approach our work from different perspectives and using at the same time different naming conventions. We identify at least two ways of considering voice-driven synthesis: as a *voice transformation* (which could be also related to *hyperinstrument*[3]) or as a vocal control (in the manner that pitch-2-MIDI devices work). Let us describe their characteristics, which will allow us to clarify ideas and hopefully to tackle in a more formal way the concept of voice-driven synthesis.

When we talk about hyperinstruments, we consider the acoustic instrument sound to be modified by a series of electronic processes controlled by the performer gestures. Performer gestures are captured by means of sensors embodied in the acoustic instrument (e.g. sensors in the bow) or in the performer (e.g. sensors in the wrist). Seeking an analogy of hyperinstruments for the singing voice, the *hypersinging* would consist of an interface that captures vocal gestures, which are used then to apply some electronic process to modify the singing voice acoustic signal. In fact, in our opinion this is very related to the concept of adaptive audio effects (Verfaille et al., 2006b). Yet another example is the voice morphing for karaoke applications by Cano et al. (2000). Hence, we consider *hypersinging* equivalent of voice transformations (voice effects).

In contrast, we see vocal control as conceptually different. Principally, the difference resides in that in voice transformation, there is only one sound stream that is manipulated (top diagram in figure 3.1). In vocal control, two distinct sound streams are involved, voice input and synthesis output (bottom diagram of figure 3.1).

3.1.2 Factors involved in singing-driven interfaces

As a preliminary step in the process of of designing, what we call singing-driven interfaces, we investigate the variables or factors that are involved. Figure 3.2 shows the main dependant factors of the proposed singing-driven interface. We detail next the implications of these four factors.

[3]This concept was coined in the 90's at MIT by Tod Machover and collaborators, and has been widely used to refer those musical instruments that incorporate electronic extensions to acoustic instruments (e.g. the *hyperviolin* developed at MIT).

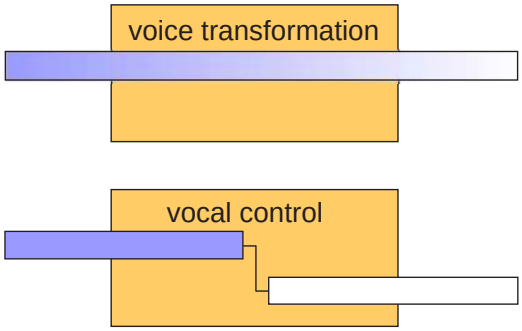

Figure 3.1: On top, a single sound stream is modified in a voice transformation approach. In a vocal control approach, features of an input voice stream are used to generate the output synthesis stream.

3.1.2.1 Performer factor

When building a musical controller, it should be ergonomic and universal, in the sense that it should be playable by a majority of users. Although the human voice fulfills these qualities, it has also some inherent limitations, for instance, the vocal range. Each person has a particular vocal range, usually two octaves. In classical singers, the *tessitura*, is the part of the vocal range which the singer is most comfortable producing: for tenors, C3 to C5; for baritones, A2 to A4; for basses, E2 to E4; for sopranos, C4 to C6; for mezzo-sopranos, A3 to A5; for altos, E3 to G5.

	note range	Hz
bass	E2 to E4	82.41 to 329.63
bariton	A2 to A4	110.00 to 440.00
tenor	C3 to C5	130.81 to 523.25
alto	E3 to G5	164.81 to 783.99
mezzo-soprano	A3 to A5	220.00 to 880.00
soprano	C4 to C6	261.63 to 1046.50

Table 3.1: Singers' typical pitch range.

Actually, in our context, vocal range can be easily mapped to the target range by transposing the incoming voice pitch. This modification could be user configurable, or event automatically detected with machine learning methods(see section 7.2). In addition to the physical properties of a particular voice, user dependency has other factors: the musical training. Musicians develop skills to control the expression of their instrument. This ability permits then to transmit a musical intention into sound accurately. Singers, thus, learn to control loudness, pitch stability, vibrato or more specific sound qualities like type of phonation or the singer's formant (Sundberg, 2001).

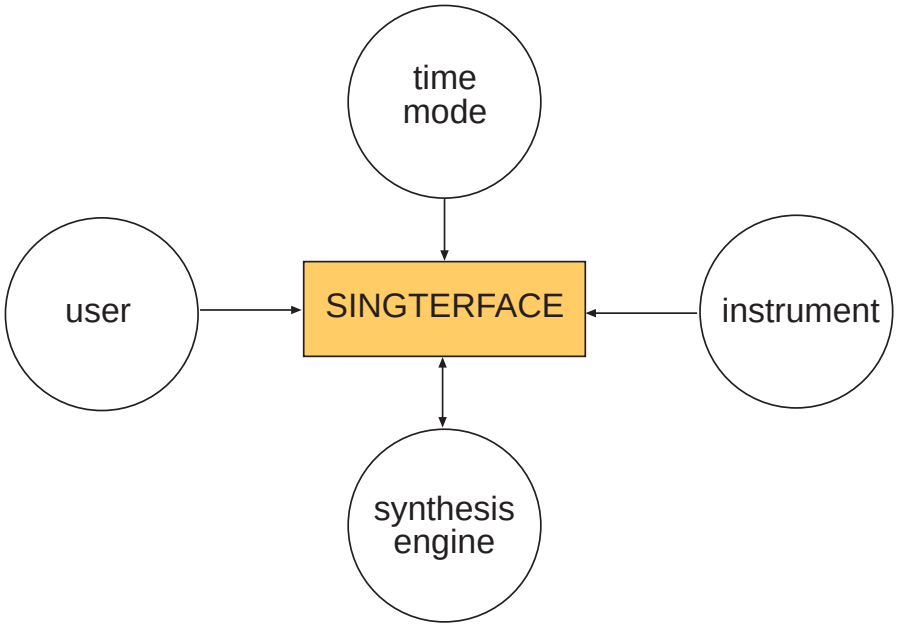

Figure 3.2: The design of a singing-driven interface (singterface) has to deal with four main dependencies: performer (user), imitated instrument (instrument), synthesis technique (synthesis engine) and operation mode (time mode).

Furthermore, by referring to the previously mentioned studies by Dalla Bella et al. (2007), considering that most of people is able to sing, we can endeavor to build interfaces that are accessible to a majority of users.

3.1.2.2 Imitated instrument factor

Each musical instrument has a set of controllers and sound features that can be controlled. This set of controls can be very heterogeneous among different instruments, although these will be similar for a given family of instruments. The sonic possibilities of an instrument, if we omit the attack, can be summarized as pitch, dynamics and timbre variations. These variations can be either continuous, discrete or unique, depending on the instrument. They allow to produce modulations such as vibratos, glissando, trills, for the pitch; tremolo and envelope control for dynamics; and are more specific for timbre (e.g mutes in brass instruments). Taking the pitch as an example, it will be discrete on a piano, while it varies continuously on voice or bowed string instruments. Also, in many instruments the pitch variation might be continuous only in a small range (e.g. guitar) allowing vibrato but not glissandi.

It seems clear that when controlling the synthesis of an instrument with the voice, we need to adapt the sonic possibilities of the voice to those of the imitated instrument. Therefore, by taking into

account the role of the synthesized instrument, we are able to adjust the mappings depending on the sound output. This constitutes a major difference over pitch-to-MIDI systems, which parameterize input signal regardless of the properties of the synthesized sound. Following with the example of the pitch, if we plan to control a piano sound, the pitch of the voice will have to be quantized and remain constant for the whole note, if we want to get a realistic synthesis of piano[4].

One perspective for looking at the instrument dependency is referring to *Instrumental Gestures* (Cadoz, 1988), as we see in chapter 5. By having a systematic description of the performer-instrument interaction, we would be able to establish mappings at a gesture level. In fact, recent trends in sound synthesis move in the direction of using gestural information as control.

3.1.2.3 Synthesis technique factor

In addition to the instrument to synthesize, different techniques for sound synthesis can be employed. The control layer differs also among them. Traditionally, how to convert the control signal into the synthesis parameters is known as *mapping*, as already introduced in section 2.1.4. Physical Modeling synthesis, for instance, has a set of parameters that relate to the actual mechanism of the acoustic instrument. One can specify the amount of blowing on a clarinet model, the bowing velocity on a violin model, etc. In contrast, in Sinusoidal Modeling synthesis amplitudes and frequencies of the partials have to be specified at a given frame rate.

Although in the prototypes of this dissertation (chapters 4 and 6), we use sample-concatenation, some preliminary experiments use Physical Modeling as a proof-of-concept. In the table 3.2, one can observe some examples of the influence of the target instrument and the employed synthesis technique.

Instrument factor	
Pitch	Tessitura (bass vs. violin) Pitch deviations (violin vs. piano)
Volume	Continuous (flute) Impulsive (plucked guitar)
Timbre variations	Instantaneous (piano attack) Continuous (violin)
Synthesis technique factor	
Simple oscillator	frequency and amplitude
Physical Models	length, excitation force, air-flow pressure, etc.
Spectral Models	sinusoidal partials amplitude and frequency, etc.

Table 3.2: The Mapping layer consists of two sub-layers, which adapt the input parameters depending on the instrument sound and the synthesis technique.

[4]It is necessary to mention that here the piano is considered played as a monophonic instrument.

3.1.2.4 Operation mode (real-time and off-line systems)

Although musical controllers are principally designed to work in real-time, we can devise situations in which working in an off-line manner[5] can improve the sonic results. As a result, throughout this dissertation we make reference on two time modes of voice-driven synthesis: off-line and real-time.

In contrast to off-line mode, the limitations of real-time are basically the introduction of latency; and the lack of context information. The former us caused by the voice analysis algorithms, for example, pitch detection needs some signal cycles to estimate the fundamental frequency. The latter might be specially important for some synthesis techniques than for others. For example, in sample-concatenation synthesis, contextual information allows to select longer samples from the database (covering several notes) according to a given voice input. If the database is sufficiently large, the picked sample will need less transformations, reducing the possible artifacts in the synthesized sound.

3.1.3 Limitations of real-time pitch-to-MIDI systems

When most MIDI-based synthesizers were keyboard-oriented, the introduction of first pitch-to-MIDI devices offered great expectations for non-keyboard performers to control sound synthesizers expressively. These pitch-to-MIDI devices acquire the sound by means of a microphone or a pick-up (see also section 2.2.1). However, it presented several persistent problems that limited its success. Basically, the generation of erratic spurious notes and the poor response for low pitched notes. In addition, there is the inherent limitations of the MIDI protocol when trying to parameterize non-keyboard music. Expressive resources such as portamento are difficult to code in real-time to MIDI messages. In some instruments, the performer has to play cleanly as precisely as possible to help the pitch analysis algorithm. Principal characteristics of MIDI and audio-to-MIDI converters are described in sections 2.1.1 and 2.2.1.

3.1.3.1 Case study: voice-to-MIDI bass guitar

If we plan to control a synthesizer, we will find that most of these devices use MIDI as input. Hence, it seems convenient to first consider the so-called *pitch-to-MIDI* systems as a possible path for our enhanced singing. This case study addresses the performance of a state-of-the-art real-time pitch-to-MIDI system for controlling a commercial bass guitar synthesizer. Given a target melody, we look at the MIDI transcription of a sung performance. Next, we compare the synthesized sound to an early prototype of voice-driven synthesis.

The next figures (figures 3.3 and 3.4) show the midi note sequence of a musical phrase, where the target note sequence is played with a keyboard and the transcribed MIDI transcription using the Digital Ear software in real-time mode. One can observe that the MIDI transcription presents two types of error: note onset detection error, and octave error in the pitch estimation[6].

[5] The term *off-line* is understood as a non-real-time processing.

[6] A remark to the transcription is that, although the pitch of the MIDI note does not always corresponds, the pitch

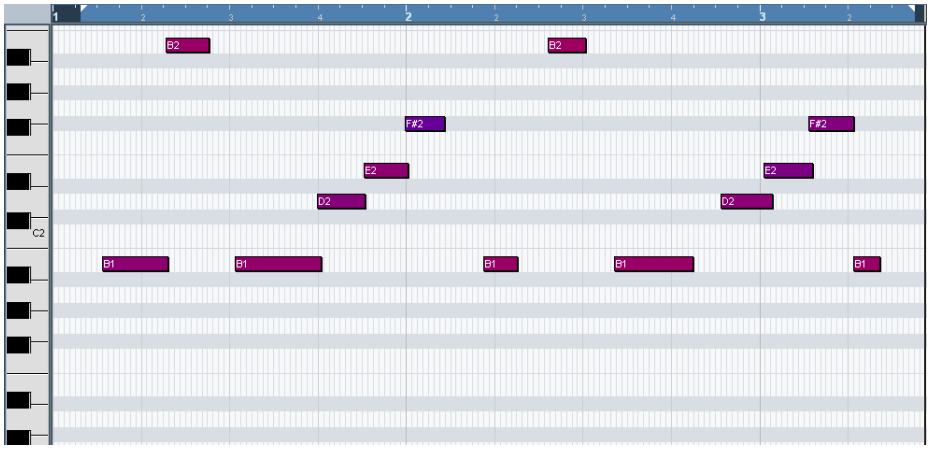

Figure 3.3: Target note sequence of a bass imitation using a keyboard musical controller.

We generated several sound examples: input voice (Audio 3.1) , DigitalEar (Audio 3.2) with Creamware's SixString synthesizer [7], Instrumentizer (Audio 3.3) (see section 1.1.3), and a keyboard (Audio 3.4) controlled synthesis with Creamware's SixString synthesizer.

3.1.3.2 Discussion

Summing up the limitations of using pitch-to-MIDI systems for voice-driven synthesis, we see first that, existing pitch-to-MIDI systems are designed to work with any monophonic musical audio input. In our approach, the input signal will be only a voice signal, which as we will see in section 3.3 possess special characteristics. Hence, in the design of our singing-driven interfaces, we can develop algorithms specially for this type of signals. Typically note onsets detection in pitch-to-MIDI systems looks at energy variation, considering a single note if the estimated pitch is continuous. For example, figure 3.5 shows the transcription of three variations of the same vocal melody (Audio 3.5) . For the first two variations, note to note articulations use voiced phonemes /na/ and /ra/, and the melody is considered as a single note. Ideally, we want to identify all the notes in the melody regardless of the phonetics used in the articulation.

Another problem of using pitch-to-MIDI systems for voice-driven synthesis is on the synthesizer's side. The flexibility that offers MIDI to control any compatible sound synthesizer becomes a limitation. In particular, the same sequence of MIDI messages may produce unlike sonic results setting the same instrument in the synthesizer, i.e. using the *General MIDI* recommendations. It seems clear that the pitch-to-MIDI process could be improved by having access to synthesizers characteristics such as the synthesis instrument. For example, if we are controlling a piano sound (discrete pitch), we should consider to create a new note event if the estimated pitch has a difference of one semitone

bend continuous control (note represented in the figure) might partially correct this.

[7] `http://www.creamware.de`

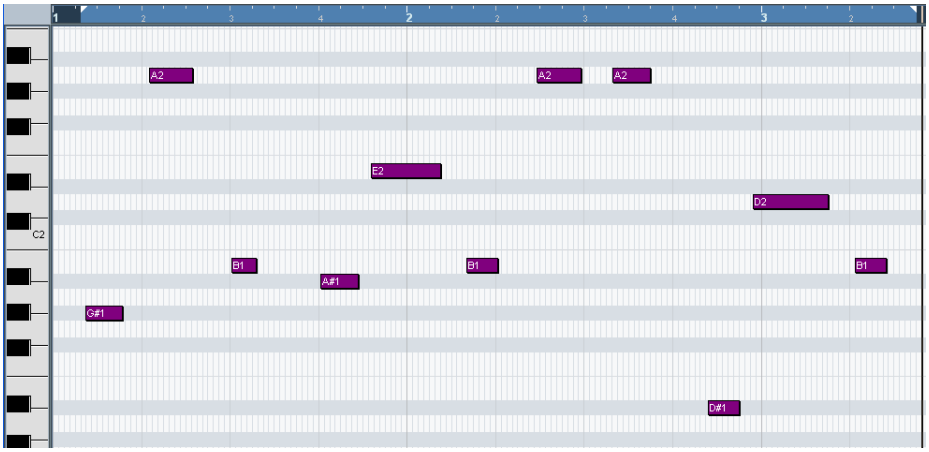

Figure 3.4: Pitch-to-MIDI transcription of a vocal imitation of a bass guitar using the Digital Ear software in real-time mode.

to the previous note. In contrast, if we are controlling a violin sound (continuous pitch), this might represent a legato articulation that can be correctly handled with the pitch bend controller.

The following list enumerates the proposed the limitations of current pitch-to-MIDI systems:

- MIDI is limited in its ability to represent continuously excitation instruments such as the voice.
- Analysis algorithms are not adapted to voice signals (and syllabling in particular).
- These systems are decoupled from the sound synthesis engine.

We propose to study the role of phonetics in the use of the voice as a control signal. Also, we strongly believe that pitch-to-MIDI systems have not been significantly successful because they are decoupled from the synthesis engine. Therefore, we propose to approach voice-driven synthesis, considering a closed system having complete control over the output sound. Our position agrees with the idea of "holistic mappings" Jordà (2005), arguing that new digital instruments can be better designed as a whole, and not as a chain of independent modules (controller and synthesis engine).

3.2 Preliminary experiments

We present in this section several experiments. The first experiment can be regarded as a proof-of-concept for the research on voice-driven synthesis. It describes how the timbre of the voice can be used to control the parameters of a FM synthesis. The second compares two synthesis techniques for bass guitar synthesis; and the third compares voice and a wind controller as input for a clarinet sound synthesizer.

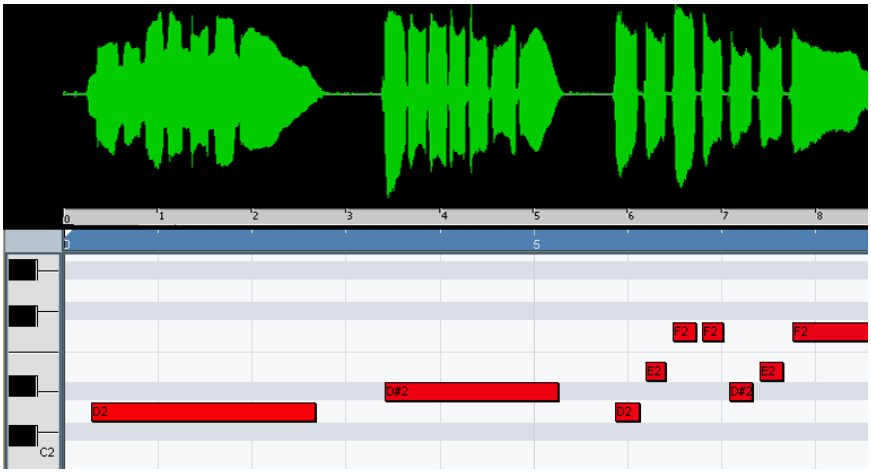

Figure 3.5: Waveform and midi transcription of the same vocal melody using different phonetics for note to note articualtions.

3.2.1 Voice-driven FM synthesis

Controlling a sound generator with a voice signal spans different degree of complexity. By using current graphical programming audio software, such as PureData[8], we can build very easily a voice-driven synthesizer using built-in analysis modules to control a single oscillator (see figure 3.6, (Audio 3.6)).

The voice offers a powerful control of timbre. We are able to either modify the timbre continuously from a more dark to a more bright sound by moving in a "vowel-space"; producing noisy sounds using fricatives; producing a distorted sound by adding roughness. All these control possibilities can be widely exploited in abstract synthesis algorithms such as subtractive or frequency modulation (FM) synthesis algorithms.

Another example is presented in figure 3.7, whereby singing voice attributes such as energy, pitch and formant frequencies drive the parameters of a simple FM synthesis engine (Janer, 2005a). The figure shows the path in PureData software. The timbre of the resulting sound was easily controlled changing the sung vowels. Although in that very first experiment, the estimation of some features lacked robustness, it showed that such a simple synthesis example can result in a rewarding musical experience for users.

It is important to remark that when controlling the synthesis of traditional instruments, we are somehow reducing the control possibilities of the voice. As explained above, the voice as a musical instrument has more timbre control than any other instrument, therefore, we will be reducing the dimensionality of the timbre control. In section 7.2, we outline some of the potential paths for exploiting the control possibilities that offers voice timbre.

[8] `http://www.puredata.org`

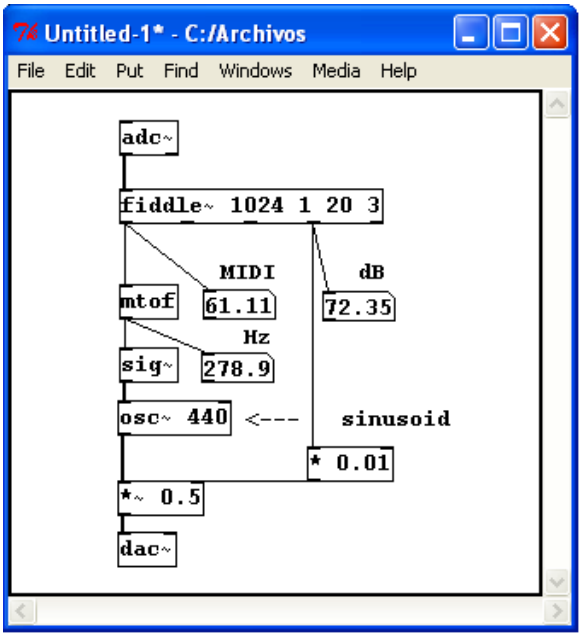

Figure 3.6: This patch shows an example of using built-in analysis modules in PureData to control the frequency and amplitude of a single oscillator.

3.2.2 Comparing techniques for bass guitar synthesis

In the next first experiment, we compare two synthesis techniques for bass guitar synthesis: physical models and spectral morphing. It represents a first proof-of-concept for voice-driven synthesis research, as reported in (Janer, 2004, 2005b).

A bass guitar is a discrete excitation instrument, producing an exponentially decaying sound. Due to the long note decay time, bass players usually damp the vibrating string with the finger before plucking a new note. Usually, either the double bass and the electric bass have four strings tuned at E1 (41.3 Hz), A1, D2 and G2. Therefore, the fundamental frequency remains below other instruments range, keeping a good timbre balance when playing together.

As we have introduced (see section 2.1.4), a general model of mapping should deal with different instruments and different synthesis techniques. For example, one can define a mapping process in which a first layer deals with the synthesized instrument, and a second layer with the synthesis engine. On this preliminary experiment, the instrument mapping layer is shared for both techniques. From the voice signal's energy, we defined a note onset signal that triggers a note depending on both the frame energy and derivative of the energy between two consecutive frames. The estimated *pitch* is transposed one octave lower and passed as a continuous parameter, thus allowing pitch bend. A pitch quantization function for simulating the influence of the frets was not integrated in the current implementation. Next we address the mapping particularities for each synthesis technique.

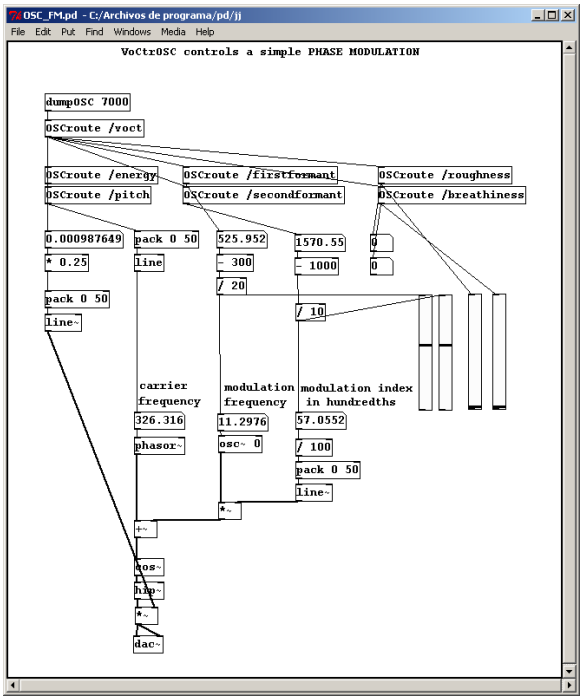

Figure 3.7: This patch shows an example of using external voice features for controlling a simple FM synthesis in PureData. Voice features are received in this case with OSC (Open Sound Control).

3.2.2.1 Approach based on Physical Modeling

This algorithm was invented by Kevin Karplus and Alex Strong (Karplus and Strong, 1983). Actually, this can be considered as the first attempt to the Physical Modeling synthesis, although this term did not exist yet in the computer music community. The success of such algorithm lies in its low-computational cost, which allowed at that time to synthesize in real-time a plucked-guitar with a substantial quality on cheap processors. It is worth to mention that 1983 was the time when the first Personal Computers were introduced; and the digital music synthesis in real-time required large and expensive computers only found in big research centers.

Although we refer here to Physical Models, originally this algorithm was based on the *Wavetable Synthesis* technique, which repeats number of samples generating a periodic signal. Instead, the Karplus-Strong algorithm introduces a variation by averaging two successive samples (equation 3.1), and writing the resulting sample in the wavetable. This can be seen, thus, as a delay line with length N.

$$Y_t = (Y_{t-N} + Y_{t-N-1})/2 \tag{3.1}$$

$$pitch = \frac{f_s}{N + 1/2} \tag{3.2}$$

It simulates a rigidly terminated string with losses, generating a periodic sound with decay. The delay line is initialized with random values (+A/-A) for simulating the *plucking* action. The pitch is determined by the length of the delay line (equation 3.2). This leads to a bad resolution for small values of N, quantizing the frequency for a high pitch. Note that in our case, in the pitch range of the bass this is not significant. The physical interpretation, referring to the Digital Waveguide Theory (Smith, 1992), there are two waves traveling in opposite direction along the string. The initial pluck position is the sum of these two waves. At CCRMA, researchers experimented simultaneously with the Karplus-Strong algorithm, proposing some extensions, and analyzing it in terms of a digital filter (equation 3.3).

$$H(z) = \frac{1}{1 - \frac{1+z^{-1}}{2} z^{-N}} \tag{3.3}$$

The extensions developed by Jaffe and Smith (1983) overcame several limitations of the original algorithm. It included a *Frequency Tuning* to avoid quantization with an all-pass filter. Also, a *Decay Alteration* make possible to shorten low pitches and to stretch for high pitches. The *Dynamics* are controlled by a Low-Pass Filter. The timbre variations due to changes in the Pick-Position are simulated with a comb filter. Finally, the possibility of simulating sympathetic strings was also tackled, which permits a complete guitar emulation.

We adapted this algorithm in order to be controllable by the voice, as depicted in the figure 3.8. Unlike the original algorithm, in which the delay line is filled by noise, we feed the Karplus-Strong delay line with the transient (attack) of the sung note. Good results were achieved by attacking a note with a plosive consonant, that is, in fact considering a note as a diphone. In our bass synthesizer system, the implementation of the Karplus-Strong Algorithm is taken from the STK Library (Cook and Scavone, 1999)[9]. From the first analysis module, we compute the onsets (triggering) signal, which are related to the energy envelope of the user's voice, and the presence of pitch. When the trigger is active, N samples of the input voice passes through and fill the delay line. The actual size of the delay line (N) is determined by the estimated pitch, as it is shown in the equation 3.2.

During the design process, we noticed that sample continuation problems appeared in form of "clicks" in the output waveform, when changing from one note to another. This is caused while, having a note that is still fading out, and a new excitation signal is fed into the delay line. In order to get rid of this effect, we introduce a second *string* in our model. The onset detector (triggering stage) sends the excitation signal alternatively to one of the two strings, getting a more natural sound, even though if a remaining note fading out was present. Finally, the figure 3.9 shows the input voice's waveform and the synthesized bass sound (Audio 3.7) . In this preliminary implementation, we used

[9] `http://ccrma-www.stanford.edu/software/stk`

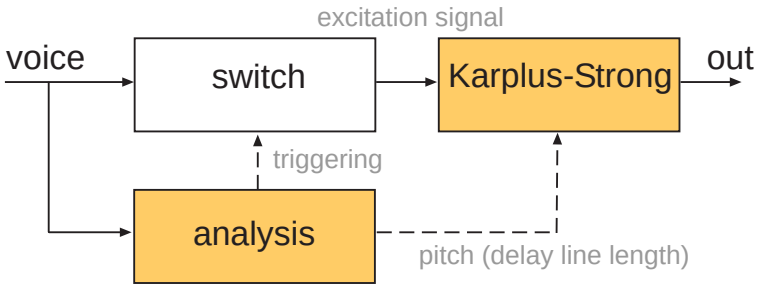

Figure 3.8: Bass synthesis with Physical Modeling. The input voice is used as excitation signal for the Karplus-Strong algorithm.

only pitch and loudness information from the input voice.

3.2.2.2 Approach based on Spectral Morphing

In contrast to the Physical Modeling algorithm, our Spectral Morphing approach combines a sample-based synthesis with transformations based on the Spectral Peak Processing (Bonada and Loscos, 2003). Briefly, depending on the input voice's parameters, a sound sample is selected from the database. Each sample contains a single note. Then, we read the sample sequentially frame by frame and transform some characteristics in the spectral domain. Finally, the processed spectral frames are converted to time domain through the inverse Fourier transform, as shown in figure 3.10.

For this preliminary experiment, we set up a very small database consisting of 12 samples (one note per sample), six of electric bass, and six of double bass. Samples were analyzed off-line, and stored in the database in form of binary files containing spectral data (complex spectrum, harmonic peaks, estimated pitch, etc.). In a further step, all samples have been labeled manually according to its characteristics. Only three descriptors (stored in a vector D_i) were used: *Pitch* (Hz), *Dynamics* ($0 \ldots 1$) and *Attack Type* ($0 \ldots 1$). In the case of Dynamics, 0 corresponds to a *pp* sound, and 1 to a *ff*. Concerning the attack type, we decided to classify two types of sounds depending on the plucking technique pick-plucked or fingered, whose sounds are primarily related to the attack.

In the mapping module, a retrieval function calculates the minimum Euclidean distance between the input observation o_i (Pitch, Loudness and Attack Type) and the descriptors D_i of the database elements. Since we are dealing with a very small database, few combination of loudness and pitches are available. Only transposition and gain transformations were implemented.

Another factor that must be taken into account is the timing. The pre-analyzed samples have a certain duration. In our context, though, the performer's voice controls the duration of the synthesized sound. A solution taken by most sample-based synthesizers set two loop points (A and B) within a steady region of the original sound. When a note is triggered, we start reading the selected sample frame by frame. When the point B is reached, we jump to the point A playing these

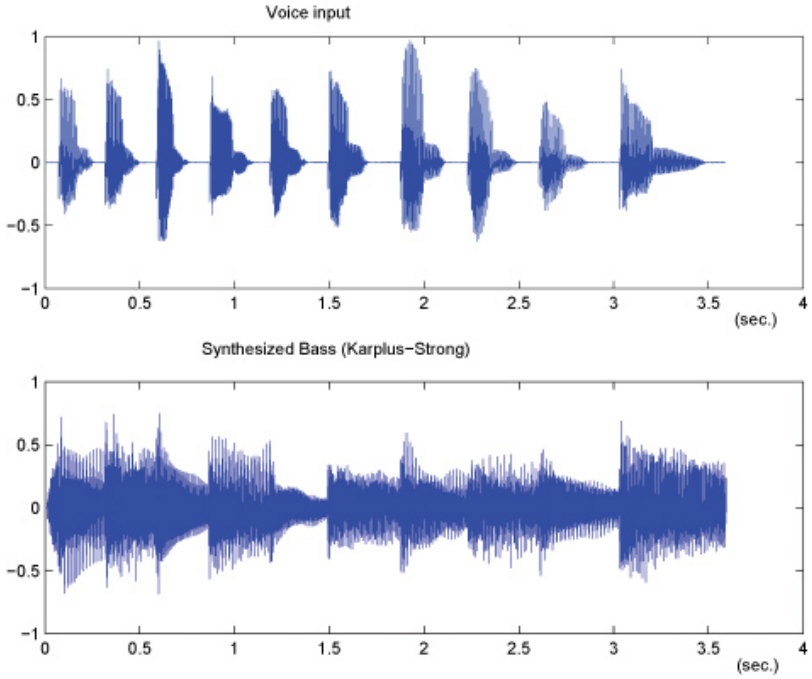

Figure 3.9: Waveforms of the input voice (top) and the Karplus-Strong synthesized bass sound (bottom).

frames continuously in a loop. When the performer releases the current sung note, the last frames are played until reaching the sample's end.

In sounds from discrete-excitation instruments such as bass guitar, the timbre is affected by the exponential decay of energy. High frequency harmonics decay more rapidly than low frequency ones. When using the looping strategy explained above, the resulting sound presents abrupt timbre changes. In order to overcome this problem and, at the same time, to get a duration independent of the sample's length, we tried an alternative approach. The attack portion is synthesized as it is in the original sample, until a frame labeled as "sustain" is reached. Then, this frame is repeated, applying phase continuation and modifying its energy depending on the input voice loudness. On the one hand, this approach permitted to produce interesting and *unreal* sounds, having a real bass guitar attack with a longer decay with amplitude modulation. On the other hand, first user tests showed that users felt more comfortable when keeping the original sample duration. Thus being able to control only sample onset and sample release as in this example (Audio 3.8) . The spectral morphing implementation here described was partially developed under the Semantic HiFi[10] project (see section 1.1.3). It was included in the final prototypes under the name of *Instrumentizer* as a real-time VST plugin.

[10] `http://shf.ircam.fr`

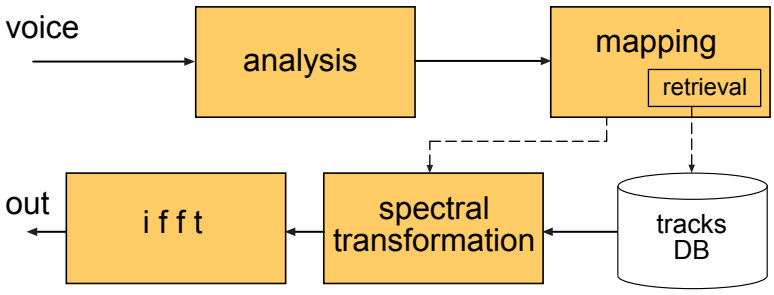

Figure 3.10: From the voice features, the mapping module selects a track from the database. The stored samples are transformed and finally converted to the time domain.

3.2.2.3 Discussion

For comparing the two synthesis techniques, we can look at different aspects: sound quality, computational cost and control flexibility. Next, we point out the most relevant characteristics of both techniques observed in this experiment.

- Concerning the sound quality, nowadays, both techniques can achieve very good results, which in some cases are undistinguishable from an acoustic instrument.

- If one compares the computational cost of both synthesis techniques in this preliminary experiment, the physical modeling algorithm is much more efficient than the spectral morphing. However, this can not be extended to all synthesis algorithms based physical modeling, since it depends on each particular algorithm. Although, for spectral morphing, the computational cost is high, it is fixed regardless of the instrument. Nevertheless, current off-the-shelf computers support the real-time synthesis using both techniques.

- Physical modeling synthesis reproduces an acoustic system (here an acoustic musical instrument), where its control parameters are counterparts of those of the simulated acoustic system. It means that, in our case, we have to derive the real performer's gestures (e.g. plucking velocity, breathing pressure, etc.) from the voice signal and map them to the input control of the physical model. In contrast, for the spectral morphing approach, there is a direct control of the synthesized output sound, without the intermediate step of estimating the controls of the acoustic instrument. In voice-driven synthesis, it is very likely that vocal imitation signal conveys already characteristics of the target output sound. This is closer to the spectral morphing approach presented. Moreover, for physical models each instrument has its algorithm, and therefore, its specific control parameters. It means that mapping strategies have to be redefined for every instrument. In the case of spectral morphing, the mapping strategies can be more general.

3.2.3 Comparing musical interfaces for clarinet synthesis

With the purpose of identifying the pros and cons of the voice as a musical interface, we ran an experiment that aimed to compare the voice with a MIDI wind-controller for modifying a perceptual attribute of a synthesizer. In our case, our goal is to control the timbre dynamics of a clarinet sound using a wind instrument synthesizer (*Ssynth*) developed at McGill University's Music Technology Area (Verfaille et al., 2006a). Although a valid evaluation requires at least a dozen of participants, for our experiment a smaller number of participants was available. We had five subjects in total, three singers using the voice as a controller; and two saxophone performers using the wind-controller. All participants come from the Faculty of Music, McGill University, Montreal. Next, we highlight relevant issues related to this experiment, a more extensive description can be found in (Janer, 2006).

3.2.3.1 Experiment description

The objective is to compare the capabilities of the singing voice with an existing controller dedicated to wind instruments such as the Yamaha MIDI WX5. Obviously, the singing voice has inherent limitations for synthesis control. For instance, playing a very rapid note sequence can be easily performed with the wind-controller fingering but is hard to achieve by singing. We are aware of that, thus, our mission is to explore the qualities of the voice for other expressive resources. The experiment set-up consists of two computers, one MIDI interface, a microphone and a windcontroller (Yamaha WX5). As shown in the figure 3.11, for voice control, the microphone captures the acoustic voice signal that inputs a laptop PC sound card. The voice analysis stand-alone application (Voctro), extracts a number of parameters from the voice and converts them as Open Sound Control(OSC) messages. Then, OSC messages are sent over the local network to the synthesizer. For the windcontroller, the WX5 sends MIDI messages to a MIDI interface that is connected via USB to a PowerPC G5. In the PowerPC, the MIDI messages are converted to OSC messages that meet the requirements of the internal protocol of the synthesizer.

The *Ssynth* is an additive synthesizer that uses a database filled with pre-recorded and pre-analyzed notes. It allows to generate sounds that interpolate between to original recorded instrument, dynamics and pitch. This document reports only the research carried out that focuses on the voice control. For further details on the synthesizer used in the experiment, we refer the reader to (Wanderley, 2001) or to the website of the Music Technology Area [11]. Concerning the parameters extracted from the voice signal were: energy, pitch and harmonic spectral centroid. The goal was to control the dynamics of the synthesized sound using the timbre of vowels. For this purpose, we designed a one-to-one mapping: energy controls the loudness level of the synthesizer, pitch the fundamental frequency of the synthesizer and the harmonic spectral centroid controls the dynamics.

Concerning the latency in our experiment, we only focus on the sustained part of a note. Therefore, we have room for latency, which is mainly noticeable during the note attack. Studies indicate

[11] `http://www.music.mcgill.ca/musictech`

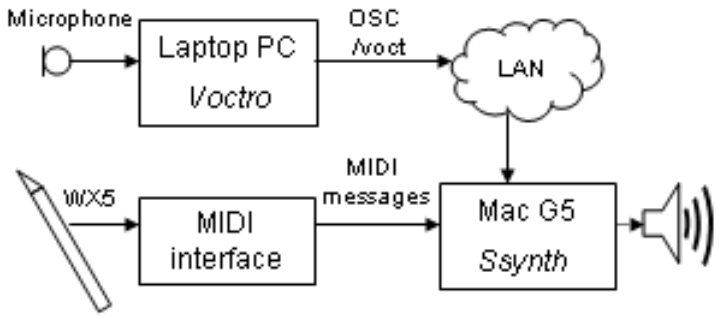

Figure 3.11: Set-up diagram. Microphone connected to a laptop sending OSC over a network. Windcontroller connected to a USB MIDI interface directly to the Mac G5 synthesizer computer.

that it should be kept below $30ms$ for percussive instruments.

3.2.3.2 Clarinet timbre control

The clarinet has a wider pitch range than the common pitch range of the human voice. Usually, humans are capable of producing sounds in a pitch range of usually 2 octaves (table with different registers). In the typical clarinet model (B♭), the note range is of almost 4 octaves, going from E3 ($147Hz$) to C♯7 ($1976Hz$). Depending on the selected piece and the singer register, it will imply to transpose the analyzed singer pitch to a higher frequency. This opens another issue to be tackled in future work, which relates to find other parameters of the voice for changing the register of the synthesized instrument. This would be a similar control as in wind instrument where a key allow a change of octave. Using phonetic information might be interesting but must be further studied.

Concerning the control of perceptual parameters of the synthesized sound, we ask the subjects to play the same melody with different mappings, allowing different degree of control of the synthesis parameters. The synthesizer input controls are high level parameters, also known as *abstract parameters* (Wanderley, 2001). In this experiment we are concerned only with the first mapping layer, which is responsible for converting controller parameters into *abstract parameters*. In the case of the MIDI WindController, the control parameters are MIDI messages, with three type of data. In the case of the singing voice, the controller parameters, which are specified in the following table, are converted to OSC messages at a rate of 86 frames per second.

The goal of this experiment is to evaluate the voice timbre for controlling a perceptual attribute of the synthesized sound (brightness). This task is *a priori* not intuitive for a singer, since it would tend to control the clarinet dynamics with the loudness, which relates to the actual acoustical behavior of clarinet. Nevertheless, our goal is to exploit the voice expressive capabilities, not in a traditional context, but rather to control sound generation in a wider scope. In this sense, the singer has to

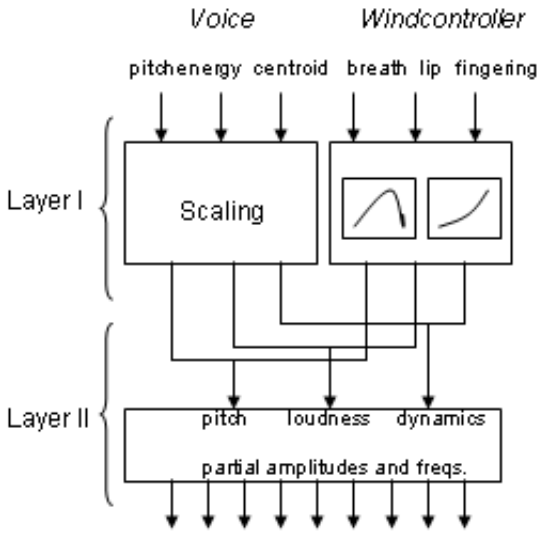

Figure 3.12: The diagram shows the different mapping layers. Both controllers share the synthesizer but the control signals follow different paths.

Control Parameters	Control Parameters	Abstract Parameters
WX5	*Voice*	*Ssynth*
breath	energy	Timbre dynamics
fingering	pitch	Loudness
lip pressure	brightness	Fundamental Frequency

Table 3.3: These tables show the control parameters for both interfaces and the synthesizers abstract parameters.

learn this new context to perform tasks that he or she is not used to.

As we mentioned, this experiment is a first attempt to compare the voice with existing musical controllers. In our case, a MIDI Windcontroller (Yamaha WX5). For the protocol, we defined eight tasks to accomplish by the participants. More specifically, we look at the control of a perceptual attribute of the synthesizer, the *brightness*, which is related to the timbre dynamics of the clarinet. For each task the procedure consists in listening a reference musical phrase of a real clarinet recording. Then, the subject has to repeat the same phrase by focusing on the *brightness* of the synthesized sound. Each task was a a sequence of notes with varying dynamics. The participant could listen to the reference phrase multiple times before performing his task with one of the controllers, the voice or the windcontroller. In both cases, the participant listened to the synthesized sound through headphones. For singers, it reduces the sound level of his own voice and can better concentrate on the synthesized sound.

3.2.3.3 Discussion

Here we provide some comments resulting from this experiment. This helps us to understand the posed problems and to find new strategies for a better voice control of musical instrument synthesis. As we noticed, the direct mapping between input and output fundamental frequency, sound level and brightness is not the ideal solution. The first reason is jitter on the fundamental frequency; the second reason is the attack quality; the third reason is adequacy of the proposed mappings for brightness control.

Pitch jitter: The fundamental frequency in the singing voice presents a much higher jitter than the clarinet. It is due to the different acoustical characteristics of both sound generation system. The acoustical operation of the vocal folds is much complex than the ressonant body of a bore . This jitter is perceived as not natural in a clarinet sound. We aim at identifying pitch contour patterns that can be automatically mapped to the synthesizer. As observed in figures 3.13 (Audio 3.9) and 3.14 (Audio 3.10) , the estimated pitch signal is much more stable and flat in the clarinet than in the voice for the same note sequence.

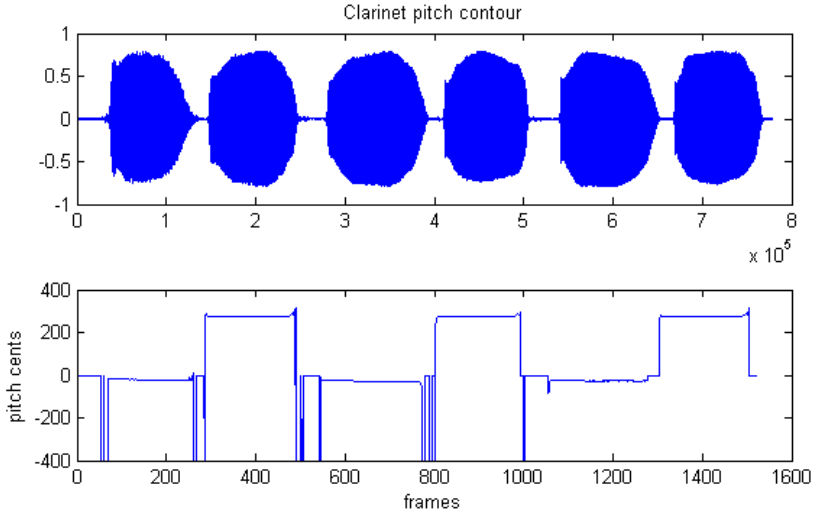

Figure 3.13: Estimated pitch of clarinet recording playing a note sequence in cents of semitones relative to A4. Waveform (top) and pitch contour (bottom). For clarity reasons in the representation, unpitched values are set to 0.

Amplitude Envelope and Attack: obviously, amplitude envelope is an important characteristic of any instrument timbre. The attack does not seem realistic for several reasons: the shape of the amplitude envelope (not smooth enough), and the fundamental frequency values during the first frames (the estimation is unstable during attack, resulting in big fundamental frequency variations). A solution consists in using a threshold on amplitude before starting a note, then using an expander to provide realistic smooth attack. By doing so, the instabilities of fundamental frequency happen

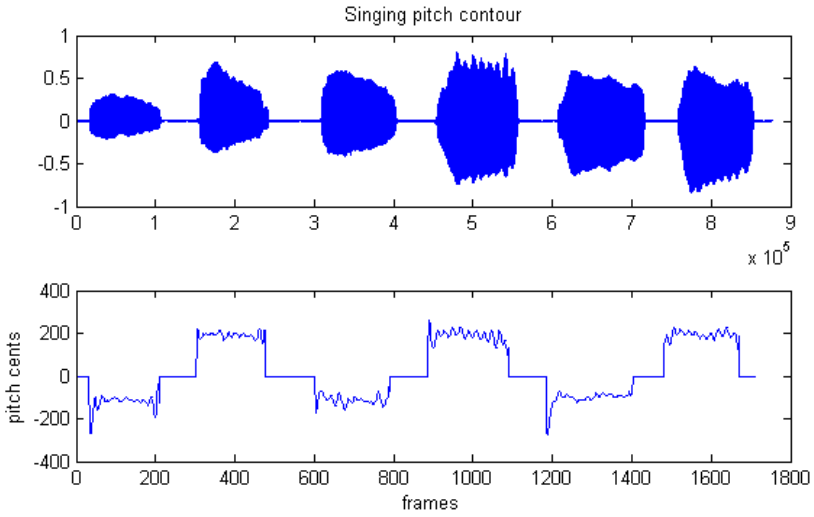

Figure 3.14: Estimated pitch of a recording of a singing voice note sequence in cents of semitones relative to A4. Waveform (top) and pitch contour (bottom). For clarity reasons in the representation, unpitched values are set to 0.

mainly when the amplitude is under the threshold, and cannot be heard. For this experiment we decided to focus only on the timbre of the sustain part of the sound.

Brightness control: some subjects reported that mapping vowel timbre to the dynamics of the clarinet was not intuitive for them. However, this mapping is not intuitive as long as a singer tries to do exactly what he learned to do before, e.g., use vowel to sing lyrics. But when the singer starts to really explore, he may be able to use vowels as a way to control brightness, as jazz singers do when scatting. Maybe our instructions were not clear enough, maybe our singers are not the appropriate subjects for this experiment). If we simplify too much the mapping or the task (for example asking the singer to sing louder so that, naturally, it sounds brighter), then we may limit the possibilities of control (e.g., how can we produce a louder sound that is darker, if loudness and brightness controls are not independent?).

One subject was able to control relatively the brightness but not in a gradual manner. This can have its origin basically in the centroid algorithm, since it gives a rapid varying signal since it uses only the harmonic peaks instead of the actual spectral bins. He divided the dynamics (brightness) parameter in three regions, one controlled by the sound /u/, the second by the sound /a/ and the third by the sound /i/. It allowed him to control the brightness consistently.

Another subject suggested that the mapping was absolutely not intuitive for a singer, indicating that “in singing the vowels are part of the linguistics and not of the music”. This is a good example that demonstrates that the singing voice as a control input has to be tackled differently than traditional singing. He added that it would seem a lot more obvious and intuitive to control the clarinet

dynamics with the "vocal effort". In terms of analysis parameters, this corresponds mainly to the glottis excitation level and probably a bit to the mouth opening. A combination of spectral slope and centroid might be a good descriptor. Subjects (singers and saxophonists, respectively) had to match the dynamics of a pre-recorded reference sequence.

- For singers, timbre dynamics was controlled with the centroid, and it was independent from the voice level. They had to learn it, since it was not "intuitive".
- For saxophonists, timbre dynamics was controlled mainly by the lip pressure and not really coupled with the breath pressure. This was not "intuitive" for the performers.
- Audio examples: Original clarinet (Audio 3.11) , Voice input 1 (Audio 3.12) , Voice control 1 (Audio 3.13) , Voice input 2 (Audio 3.14) , Voice control 2 (Audio 3.15) , WX5 control (Audio 3.16)

To sum up, this experiment addressed the voice control of a clarinet synthesizer input parameters. More precisely, the control of the timbre dynamics of the sound, ranging from a *pp* to a *ff*. Here, only the sustain part of a note was considered. For our specific task, the spectral centroid is not an ideal voice descriptor for the brightness control, since it depends on the timbre (vowel) as well as on the excitation slope, which is related to voice level. We believe that a more suitable control could be achieved by using formant frequencies as control parameter which is independent of the voice level.

3.3 Preliminary studies

This section addresses the characteristics of the voice in instrument imitation, looking at the role of the phonetics in nonsense text singing, referred also as *syllabling* (Sundberg, 1994). The study of syllabling will give us cues about the attributes of vocal gestures in instrument imitation, whose description will be the aim of chapter 5.

When facing the study of syllabling signals from a vocal instrument imitation, we foresee a link between phonetics and the articulation in musical instruments. This link appears to be stronger for some instruments than for others. For instance, in wind instruments, musicians use *tonguing* for enunciating different notes. This tonguing can be actually seen as a sort of syllabling but having an instrument coupled to the mouth. As an example, flutists might think of the words "dah-dah". Moreover, faster articulations in wind instruments are achieved with *double-tonguing*. It consists of alternating two syllables, which taking the previous examples would be "dah-gah-dah-gah".

For other instruments, such as bowed string instruments, vocal imitations will presumably try to match the acoustic properties of musical articulations. Legato articulations that are played in one bow, will probably be imitated by keeping the same vowel ("a-a") or using voiced consonants ("da-da" or "na-na"), while making use of voiceless consonants for staccato articulations. The results of the second case study will demonstrate some these assumptions.

3.3.1 Voice instrumental music

From a musicological perspective, nonsense text singing is often referred as *voice instrumental* music. We look here at different contexts where the voice has been employed just as a musical instrument, and not in a western traditional singing where lyrics play an important role. Although it is not tackled here, vocal imitation is also found in non-musical contexts, such as the imitation of everyday sounds (e.g. children playing mimicking the noise of a car).

3.3.1.1 Mother-infant non-verbal communication

During early stages of children infancy, parental communication is partially done by means of non-verbal speech close to singing. Moreover, this fact is widespread across cultures. In the literature, it is related to the characteristics of Infant-Directed (ID) speech, as opposed to Adult-Directed (AD) speech. In fact, ID speech is acoustically distinguishable from AD speech in several aspects (Nakata and Trehub, 2004). It includes a heightened pitch with exaggerated pitch contours, and usually presents a slow tempo. Being tolerant, one can consider this non-verbal speech as an example of voice instrumental, where musical characteristics such as pitch contour and tempo are of high significance (Fernald, 1989).

3.3.1.2 From non-western traditions to contemporary techniques

In *voice instrumental*, on the one hand, the voice can represent the aesthetic goal itself as using it as an instrument; on the other hand, the voice can be employed to imitate an instrument with pedagogical purposes. Referring to the former, all the possibilities of the voice, without using any meaningful text, are used with expressive intentions: timber, rhythm, pitch, dynamics, etc. Some manifestations of nonsense singing can be broadly found in contemporary classical music, such as Carl Orffs use of the voice, Arnold Schonbergs “Sprechstimme”, also used by Berg. Other composers such as Luciano Berio and Steve Reich used the voice as an integral aspect of experimental music. In traditional cultures, nonsense voice is used in Carnatic music of South India, in a form of voice improvisation called “Thillana”. Tuvinian throat singing often features wordless and improvised song and Hasidic Jews use a form of voice improvisation called “nigunim”. Also “cante jondo” can be included in this category.

Popular music, mainly jazz, uses the voice as an instrument, such as famous Louis Armstrongs and Ella Fitzgeralds “scat singing” (Kernfield, 1988). Scat singing is defined as a vocal improvisation using phonetic sounds similar to the instrumental sounds of jazz. Although, scat is believed to have its origins in African American ring shout ceremonies, it was first popularized by Louis Armstrong in a 1926 recording, when he dropped the lyrics and spontaneously substituted scat for the words. This particular singing style was adopted by other great jazz singers such as Ella Fitzgerald or Sarah Vaughan. Today, scat maintains its relevance with performers like Bobby McFerrin, who exploits his

vocal register producing a wide range of sounds. Also in more recent styles as "Hip-hop" or "dub", performers use "beatboxing" which involves creating beats, rhythms, vocal scratching and melodies using the human voice.

From the acoustic qualities of the voice in the mentioned voice instrumental examples, one could classify them in two main groups: syllabic (scat, carnatic, etc.) and vocalic (tuvinian, cante jondo, etc.). As we will justify later with some case studies in this chapter, our voice-driven synthesis approach considers syllabic voice signals.

3.3.1.3 Musicians non-verbal communication

Singing without words have been widely used as an aesthetic goal itself in performance, as explained above. Nevertheless, musicians also use syllabling as a practical communication language with pedagogical purposes, either in educational contexts, during rehearsals or even in performances. Referring to instrument imitation as a way to teach music, the best well-known example is the use of "Vayttari", some syllable commands used in the pedagogy of percussion Indian music (Hitchcock, 1986). Western music education makes use of *solfège*[12] when singing a score. Musicians, thus, learn to associate notes with phonetics. In less strict situations such as in rehearsals or master classes, musicians make use of syllabling as well for communicating musical intentions among them. As we will see next, the phonetics employed depend on the musical articulation, and in some cases, also on the imitated instrument.

3.3.2 Case study 1: Instrument imitation in music education

Instrument imitation with voice is as a multifaceted topic, which might encompass areas such as musical acoustics, musicology or phonetics. Also from a social and cultural point of view, it has its significance since most people have in some occasion imitated a musical instrument by singing. This first case study looks at the phonetics employed by performance teachers in an educational context. We believe that this topic deserves a more deep study from a musicological perspective. Many aspects remained unaddressed here, for instance, the cultural and linguistic differences in the choice of syllables.

To our knowledge, first scientific publications on syllabling refer to the work by Sundberg (1994). This preliminary study looked at the choice of syllables in informal nonsense text singing for six short melodic excerpts. As they report, most subjects used the same small set of syllables, being the syllable /da/[13] employed in the 34% of the cases and the syllable [di] in the 13%. Another result of this study is that the voiced consonant [m] was used to infer micropauses, and was often

[12] Solfège associates a syllable to every note in the scale *do, re, mi, fa, sol, la* and *si*.

[13] Syllables and phonemes are transcribed within slashes (//) as commonly used in linguistics. In our context, it is sufficient to find cues about the musical function of phonetics. A finer detailed transcription with allophones are represented with square brackets ([]), following the International Phonetic Alphabet (IPA) or its machine readable variant SAMPA format `http://www.phon.ucl.ac.k/home/sampa/`

Instrument	num. instances	(%)
violin	224	37
clarinet	192	22.7
double bass	103	17.1
oboe	39	6.5
fagot	31	5.1
horn	13	2.2

Table 3.4: Percentage of the master class recordings for different instruments.

followed by the voiceless stop consonant [p]. The reason of this later case was note grouping. In the action of syllable choicing, in addition to articulatory convenience, the study revealed that it also has a musical meaning. Also in a context of Query-by-Humming, we find work related to syllabling, here referred as *Query-by-Voice* (Lesaffre et al., 2003). In order to transcribe the voice query, they consider a syllable with the following structure: the onset (consonant), the nucleus (vowel) and the coda (final consonant). Yet from another perspective, Patel and Iversen (2003) study the choice of syllables for drum sounds imitation in the Indian tabla tradition.

3.3.2.1 Description

In contrast to Sundberg's study (Sundberg, 1994), where subjects where requested to sing a musical score with nonsense syllable, we analyze the syllabling produced in an educational context. In this environment, the teacher gives indications to students by singing, imitating instrument articulations with nonsense syllables.

The experiment data consists of manually annotated recordings of several master classes[14], covering various instruments (see table 3.4). Our data set consisted of 82 recordings with a total number of 604 syllables. The annotation process consisted in transcribing manually the sung syllables, as in this examples (Audio 3.17) , (Audio 3.18) .

3.3.2.2 Results

Results derive from two distinct analysis. A first part studies the relationship of syllable choice and the type of rhythmic articulation it tries to imitate in clarinet classes of different teachers. In this case, only a subset of recordings are considered. Despite cultural differences, we found some constants. Phonemes are found to have different functions depending on the position and the articulation type. Most syllables start with a consonant and they determine the articulation. /ta/ is used for normal attack, /da/ for a softer attack and /pa/ for staccato. The most common vowel is /a/; /i/ is used in high pitch sound and /o/ in dark timbers. At the end of the syllable, phoneme /m/ is used in long sounds to create resonance as found in Indian music. When two or

[14]The recordings were carried out at the ESMUC (Escola Superior de Música de Catalunya), in the frame of the eContent project HARMOS (Ref. 11189).

more syllables are linked in groups, /ra/ is often used in quick linking. This can be explained by articulatory phonetics, since the /r/ allow rapid transitions because of its place of articulation.

In the second part of the study, we analyzed recordings of syllabling in several instruments master classes. Our total data set consisted of 82 recordings summing up 604 syllables. Results indicate that regardless of the instrument and subject, a small number of syllables is used. The most uttered syllable is /ta/(20%), used at the beginning of note group or in staccato articulations. It is followed by the syllable /ra/(16%). Figures 3.15 and 3.16 show the results. Also from this case study, we find interesting commonalities with phonetics. It occurs, for instance, in the choice of the vowel /i/ for high pitched notes. In fact, this relates to the concept of *intrinsic pitch*, widely known in phonetics (Laver, 1994). Intrinsic pitch is the average fundamental frequency for a given vowel. Studies demonstrate that, it has a frequency of $186Hz$ for the vowel /i/, which is around two semitones higher than the intrinsic pitch for the vowel /a/. This would explain that the /i/ vowel is chosen unconsciously for reaching high pitches while saving effort.

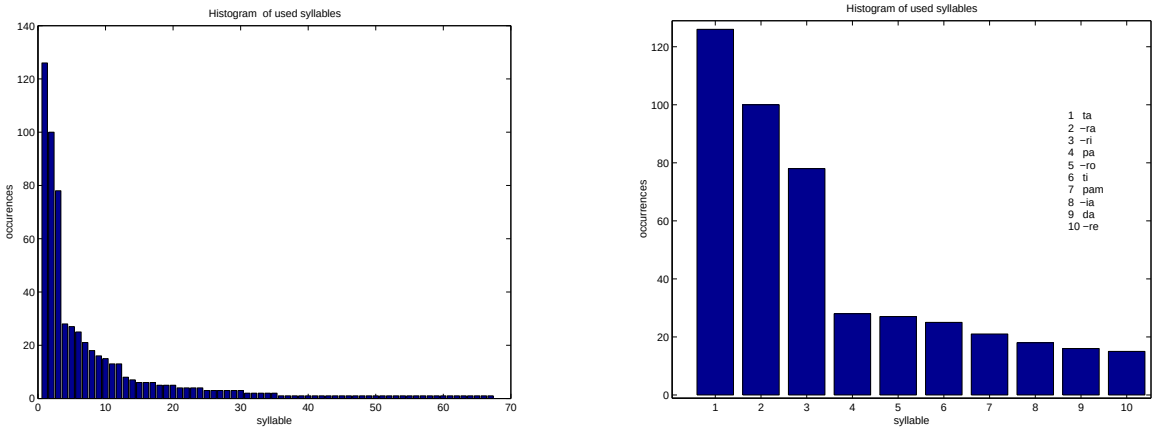

Figure 3.15: *Left:* Histogram of the choice of syllables in our data set. (e.g. syllable 1 corresponds to /ta/). *Right:* Reduced histogram showing only the ten most used syllables in our data set with the corresponding transcription.

Compared to the results presented by Sundberg (1994), we observe that in both contexts (nonsense text singing and instrument imitation) the phonetic characteristics of the singing voice is similar, consisting of syllables. Sundberg uses the term *syllabling* for describing this type of syllable, and we will using this term throughout this dissertation. A more detailed description of this study can be found in (Janer and Peñalba, 2007).

3.3.3 Case study 2: Web survey on instrument imitation

In order to provide results from a broader perspective of vocal imitations of instruments, we conducted a second case study. This study is more systematic, aiming to collect phonetic information

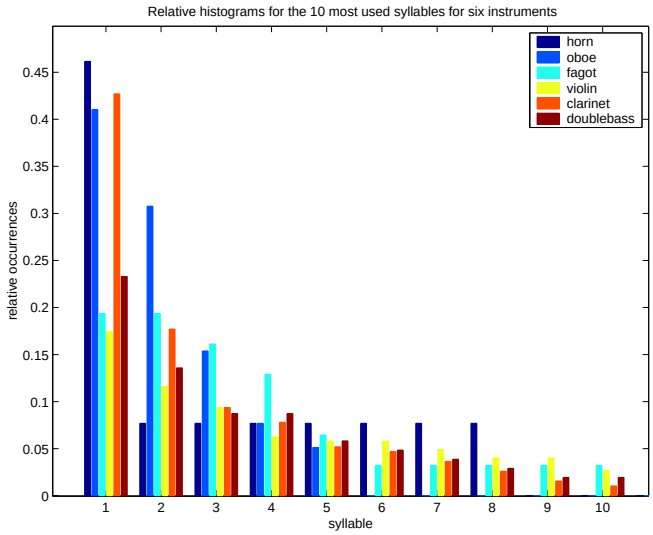

Figure 3.16: Relative histogram of syllable choice for six different instruments.

from people with different musical background, when imitating a reduced set of musical phrases.

3.3.3.1 Description

This case study consists of a web survey, where participants are presented with a number of short instrumental phrases (Audio 3.19) . Participants listened to the examples, imitated mentally the musical phrase and input the transcription of their imitation, typing the answer on a web questionnaire. Results are analyzed by means of text parsing techniques. Our goal was to identify quantitatively, not only which syllables are most used in instrument imitation, but also if the results are subject-dependent or instrument-dependent. Figure 3.17 shows the front-end of the web questionnaire. Participants indicated at the beginning of the survey their musical background (novice, amateur or professional), as well as their language. Each user had to transcribe 10 phrases, being it optionally extensible to 20 or 30 .

Musical phrases consisted of a nine-note sequence played on bass guitar, sax and violin; and variations on tempo and articulation. For the experiment, all recordings were normalized to an average RMS. Instruments were chosen to have distinct acoustic properties. Bass guitar is a plucked string instrument (with discrete excitation) and has a dark timbre. Sax is a wind instrument (and with continuous excitation) that has a "balanced" timbre (in an imaginary perceptual scale from dark to bright instruments). Finally, the violin is a bowed string instrument, also with continuous excitation, but with a brighter timbre. Variations were sorted randomly in order to avoid any strategy by subjects. The total number of variations is 36,covering:

1. *articulation* (3): legato, medium and staccato.

2. *instrument* (3): bass guitar, sax and violin.

3. *note interval* (2): low (2 semitones) and high (7 semitones).

4. *tempo* (2): slow and fast.

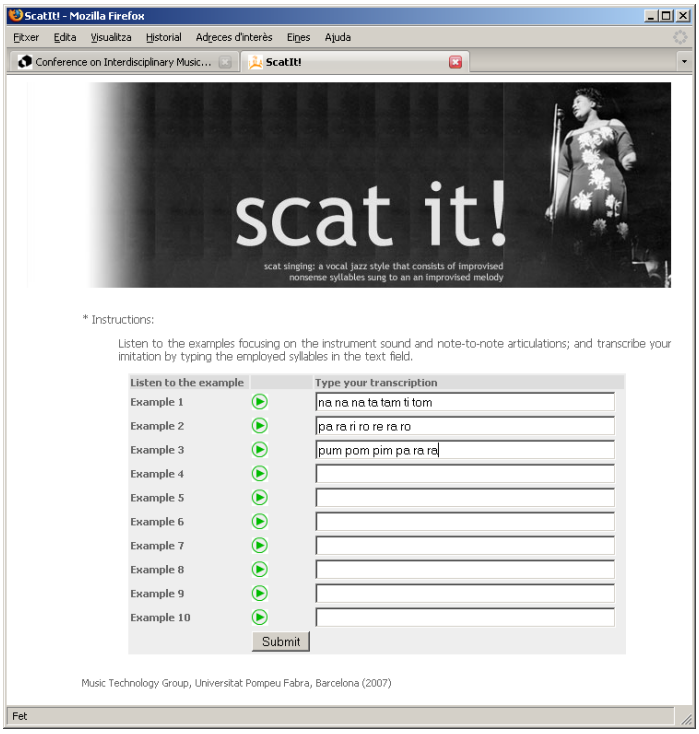

Figure 3.17: Screen-shot of the web survey *Scat It!*. Subjects listen to a series of musical phrases of saxophone, violin and bass guitar, which are randomly presented. The transcription of the vocal imitation is typed in the text field and submitted to the database.

Concerning the gathered responses, they have the following characteristics:

- Number of participants: 60
- Musical background: novice (8), amateur (39), professional (13).
- Language: romance (43), anglo-germanic (15) slavic (1) and other(1).
- Number of transcribed musical phrases: 869.

Also, the collected data, as shown in the next example transcriptions, consists of a sequence of syllables. Our interest is to compare the acoustic properties of the transcribed phonemes with the acoustic properties of the imitated musical phrases.

Pos.	*Syl.*	*Occur.*	Pos.	*Syl.*	*Occur.*
1	na	252	11	no	135
2	da	210	12	ti	131
3	ra	187	13	du	126
4	ta	181	14	do	118
5	ro	173	15	fa	110
6	ri	172	16	fo	109
7	la	161	17	ne	105
8	to	149	18	ba	105
9	dum	152	19	tu	99
10	ni	139	20	di	97

Table 3.5: List of the 20 most used syllables and the number of occurrences.

```
/dom dam di dam do dam di da dom/
/no na ni na no na ni na no/
/tululilutululilutuuu/
/jadadadadidadidada/
/dumdumdimdumdumdumdimdumdu/
/lalalilalilalilala/
```

3.3.3.2 Results

From the analysis of the collected data, we obtain interesting results. As we describe next, subjects choose the phonetics according to the acoustical properties of the imitated musical phrase. At the same time, we found a general trend in the choice of syllable that is independent of the musical phrase characteristics. Subjects tend to use the vowel /a/, and in particular, the syllable /na/ as a "default" choice. Table 3.5 lists the most common chosen syllables.

Next, we comment the following figures (Fig. 3.18 to 3.24), highlighting the influence of instrument, articulation, tempo and musical background of the subject in the vocal imitation. We consider a syllable to have a begin consonant, a vowel and an end consonant. In all figures, the first bar shows the average of all gathered responses. We observe that the most used syllables are voiced plosive (e.g. /d/ or /b/), while nasals, semivowels and unvoiced plosive are a bit below. Looking at the vowel choice, it is clear that /a/ stands out as the most used. Bars in the figures represent normalized histograms in percentage, i.e. all bars sum 100.

1. *Instrument* (Fig 3.18, 3.19 and 3.20): For the begin consonant, subjects use almost uniquely plosive consonants for the bass. Unvoiced fricatives (e.g. /f/) are relevant for the sax, when compared to the global figure. This is likely due to the blowing noise component of wind instruments. Violin notes are "attacked" with nasals (e.g. /n/) and semivowels (e.g /y/). Bass' vowels are primarily /o/ and /u/, easily linked to a dark timbre. For the sax, the vowel

choice are very similar to the average values. For the violin, although /a/ is most used, the choice of the vowel /i/ is also due to its brighter timbre[15]. Finally, comparing the chosen end consonant for the three instruments, we observe that it is present only for the bass guitar, where subjects choose a nasal to end the syllable (e.g. /pam/, /bom/).

2. *Articulation* (Fig 3.21 and 3.22): for the begin consonant, one can observe that the choice of unvoiced plosive consonant is related to the articulation, being patently important for staccato articulation. At the same time, for *staccato* nasal and semivowels are not common, while they appear for *medium* and *legato* articulations. Not remarkable differences are found for the chosen vowel and end consonant.

3. *Tempo* (Fig 3.23): Comparing the chosen begin consonant for fast and slow *tempi*, we do not observe relevant differences. Only a slight increase in the use of nasal and semivowels for fast tempo is identified.

4. *Background* (Fig 3.24): surprisingly, one can observe that *expert* users tend to use mostly voiced plosive consonant as begin consonant. In contrast, the choice is more spread for both amateur and novice subjects. A possible reason is that expert subjects would focus only on melody and timing, and not on the acoustic properties. However, a more extensive and formal study is necessary before extract some conclusions.

[15]The vowel /i/ has a brighter timbre because of the frequency of the second formant, which is around 3000Hz.

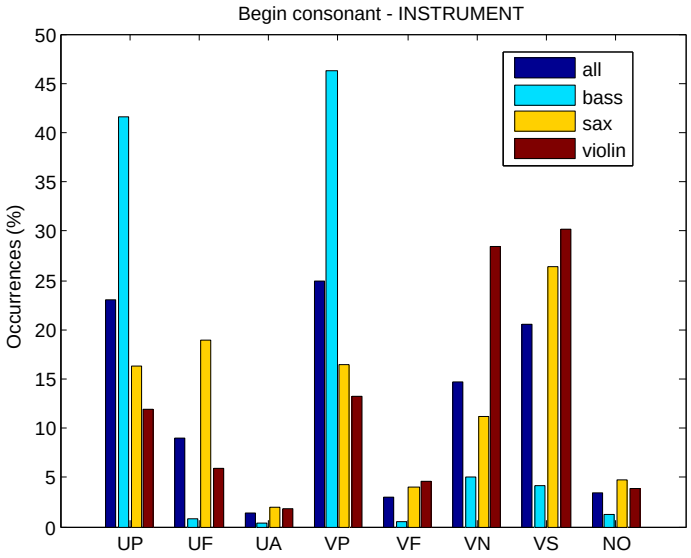

Figure 3.18: Normalized histogram of begin consonant class for different instruments. The broad phonetic classes, we use the following legend: UP-unvoiced plosive, UF-unvoiced fricative, UA-unvoiced affricative, VP-voiced plosive, VF-voice fricative, VN-voiced nasal, VS-voiced semivowel, NO-none.

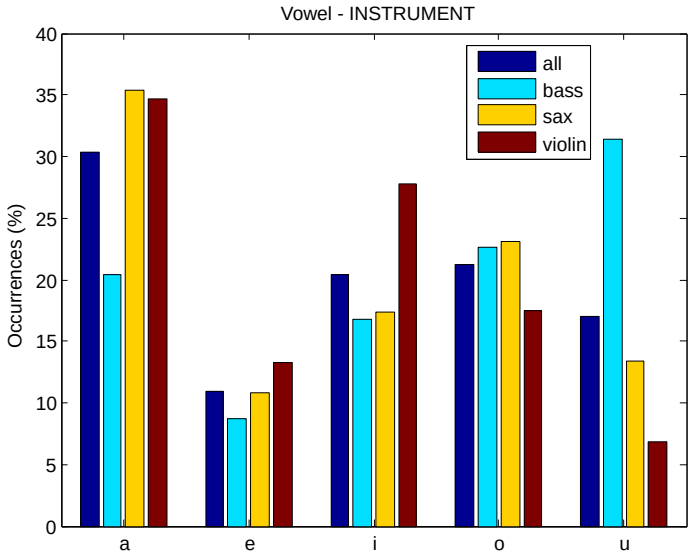

Figure 3.19: Normalized histogram of used vowels for different instruments. Only five vocalic sounds were considered.

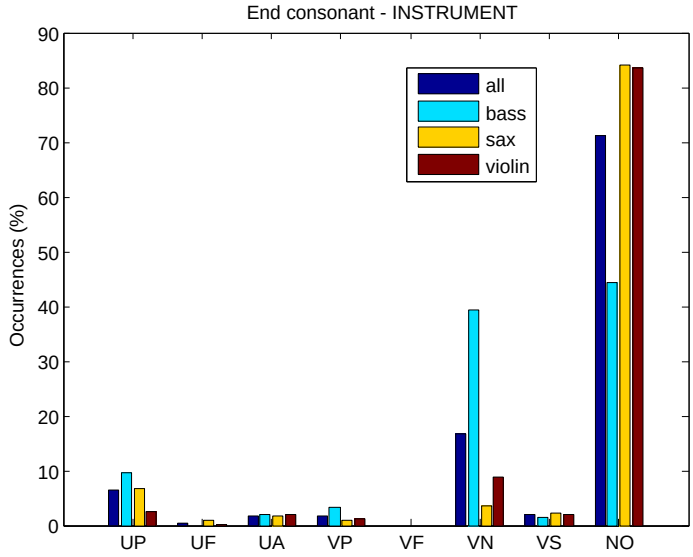

Figure 3.20: Normalized histogram of end consonant class for different instruments. The broad phonetic classes, we use the following legend: UP-unvoiced plosive, UF-unvoiced fricative, UA-unvoiced affricative, VP-voiced plosive, VF-voice fricative, VN-voiced nasal, VS-voiced semivowel, NO-none.

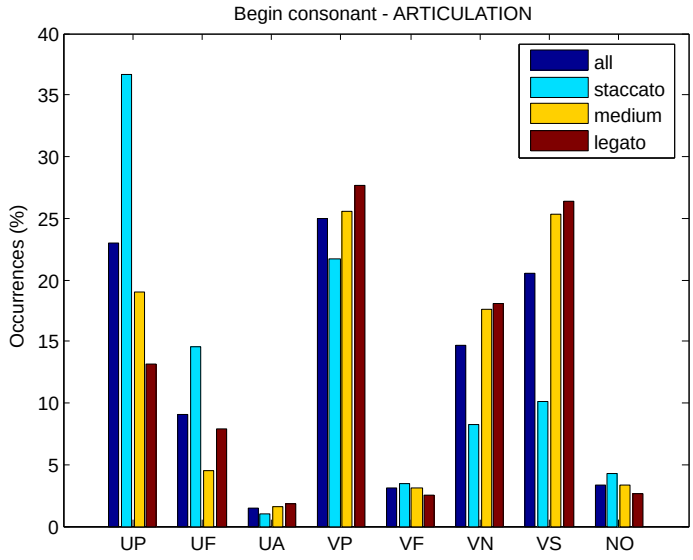

Figure 3.21: Normalized histogram of begin consonant class for different articulation (staccato, medium and legato). The broad phonetic classes, we use the following legend: UP-unvoiced plosive, UF-unvoiced fricative, UA-unvoiced affricative, VP-voiced plosive, VF-voice fricative, VN-voiced nasal, VS-voiced semivowel, NO-none.

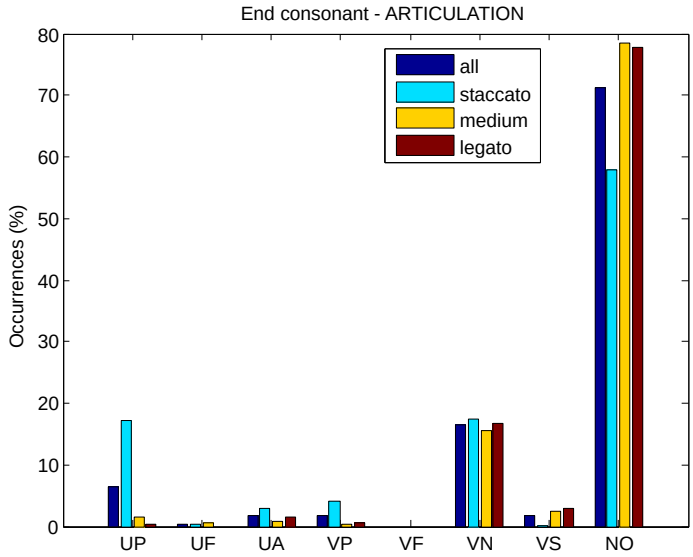

Figure 3.22: Normalized histogram of end consonant class for different articulation (staccato, medium and legato). The broad phonetic classes, we use the following legend: UP-unvoiced plosive, UF-unvoiced fricative, UA-unvoiced affricative, VP-voiced plosive, VF-voice fricative, VN-voiced nasal, VS-voiced semivowel, NO-none.

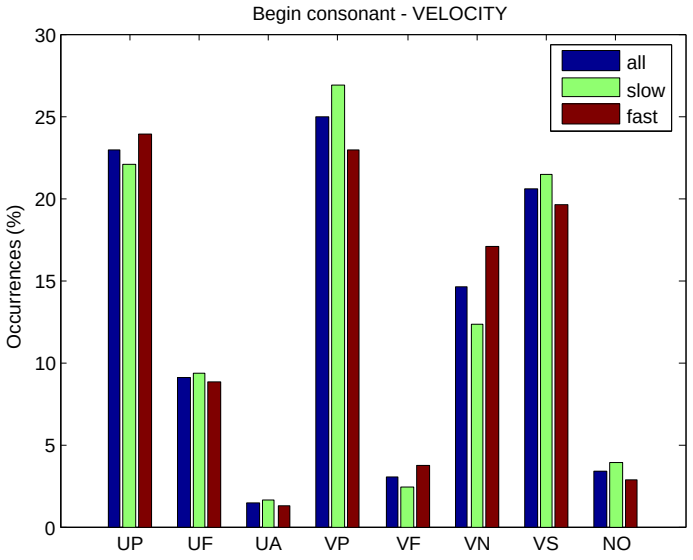

Figure 3.23: Normalized histogram of begin consonant class for different *tempi* (slow and fast). The broad phonetic classes, we use the following legend: UP-unvoiced plosive, UF-unvoiced fricative, UA-unvoiced affricative, VP-voiced plosive, VF-voice fricative, VN-voiced nasal, VS-voiced semivowel, NO-none.

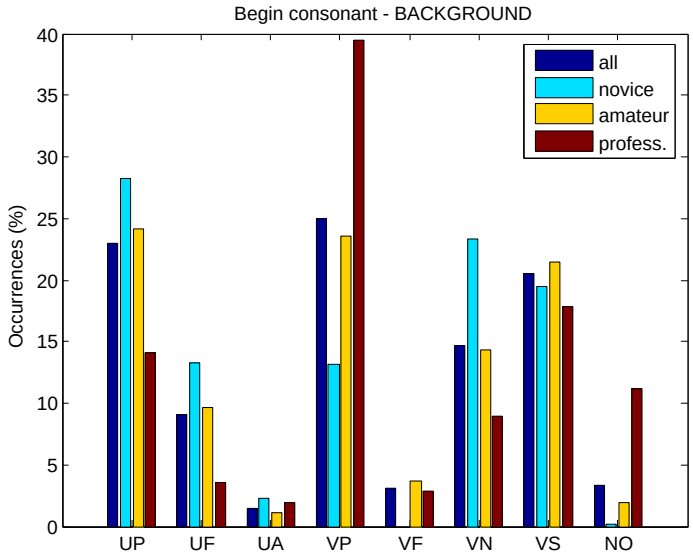

Figure 3.24: Normalized histogram of begin consonant class for subjects with different background (novice, amateur, professional). The broad phonetic classes, we use the following legend: UP-unvoiced plosive, UF-unvoiced fricative, UA-unvoiced affricative, VP-voiced plosive, VF-voice fricative, VN-voiced nasal, VS-voiced semivowel, NO-none.

3.4 Summary

In this chapter we have introduced the basic concepts of using the singing voice as a musical controller, which will be the basis for the research presented in the following chapters. We have first addressed the dependencies of a singing-driven interface when controlling a sound synthesizer: performer, imitated instrument, synthesis technique and use-case. Then, we identified the limitations of state-of-the-art pitch-to-MIDI systems, justifying the contributions of this dissertation. Three experiments on voice-driven synthesis, allows us to direct the focus of our research. Finally, two case studies address the relation of phonetics in a vocal imitation of musical instruments. These studies give us hints to develop more accurate analysis algorithms and mappings strategies.

The principal issues identified that will help us in the design of singing-driven interfaces for sound synthesis are:

- Voice might be an interesting approach to overcome the control limitations of current music synthesizers. This is one of the bottlenecks of nowadays sound synthesis as pointed out in the roadmap in Sound and Music Computing (Serra et al., 2007).

- Pitch-to-MIDI systems present several limitations. First, they are not designed specifically for voice signals, missing many nuances of the singing voice. Second, the decoupling from control (voice features) to synthesizer's parameters limits the design of mappings. For instance, voice input might be different depending on the instrument synthesized, requiring different mapping strategies. We propose a more *holistic* approach, where the interface is aware of the control input and the output sound.

- Given the four dependent factors (performer, instrument, synthesis technique and time mode), the design of a singing-driven interface will be specific for a given combination of these dependencies. In terms of implementation, it can be simplified using different configurations.

- Mapping voice features to control meaningfully an instrument synthesizer is not trivial, as shown for the case of the clarinet synthesizer when mapping timbre and pitch features.

- Comparing synthesis techniques, a sample-based concatenation synthesis allows us to define a general framework for the mapping functions, easily adaptable to several instruments while achieving high sound quality.

- Studies on vocal imitation of instruments indicate that voice will have special characteristics based on onomatopeic sounds. This signals are also referred as *syllabling*. Moreover, there is a link between phonetics and the acoustic and musical properties of the imitated sound.

From these issues, we structure the rest of this dissertation with the following chapters:

- Chapter 4 addresses the control of singing voice synthesizers (SVS). In this case, mappings are direct (voice-to-voice), in which timbre is specified by the lyrics. We use an existing sample-based singing voice synthesizer that uses spectral concatenation.
- Voice-to-Instrument synthesis will require analysis algorithms adapted to syllabling signals. Chapter 5 presents a number of voice analysis methods. Additionally, we put syllabling in the context of instrumental gestures, representing these signals as vocal gestures.
- Finally, from the results presented in chapter 5, chapter 6 addresses the design of mapping functions to control a synthesizer based on sample concatenation. Three different instruments are considered (bass guitar, sax and violin), which illustrate different instrument families.

Chapter 4

Controlling a Singing Voice Synthesizer

Up to this point, we have learned the principal issues involved in using the singing voice as a control input. The next step is to control a sound synthesizer in what we have called voice-driven synthesis. We envision uniquely the synthesis of traditional instrument sounds, which in the terminology of sound synthesis design is known as *instrument imitation* (Pressing, 1992). We believe that focusing on the sound of traditional instruments can help us in understanding advantages and drawbacks of using the voice for control tasks. Nevertheless, we are aware that some specific aspects (e.g. timbre control) could be also treated with non-instrumental sound synthesis, as we mention in section 7.2.

Before addressing the control of instrumental sounds in chapters 5 and 6, this chapter describes the control of a singing voice synthesizer. In this case, both input and output signals are of the same nature and mappings can be direct.

4.1 Introduction

Since the advent of Computer Music, several approaches have been proposed to produce a synthetic singing. Many early digital synthesizers, include *singing* as selectable sound preset, although in practice it only consisted of vowel-like ("ahs" or "ohs") sounds used as pads imitating a unison choir[1]. To recreate a complete singing (i.e. synthesizing lyrics), some approaches tackle it as an extension of speech synthesizers, such as those based on the source-filter model, e.g. Cook (1992). Other approaches undertake a completely specific approach for singing synthesis, e.g. Meron (1999).

In this chapter, we describe the design of an interface for a specific singing voice synthesizer (Bonada and Serra, 2007), referred here as *MTG-SVS* (Music Technology Group-Singing Voice

[1] Also pipe organs may include some preset with a voice like timbre.

Synthesizer)[2]. Some of the proposed ideas in this chapter are broad and applicable to other singing voice synthesizers (SVS). However, most of the research carried out is particular for this synthesizer that is based on performance sampling and spectral-concatenation of diphone samples. In this section, after highlighting the principal characteristics of the *MTG-SVS* synthesizer, we deal with specific control issues. In particular, two different approaches are proposed, one working in an off-line manner, and another one working in real-time.

4.1.1 Overview: performance sampling and spectral-concatenation

Basic sample-based synthesizers lack of expressivity and flexibility for continuous-excited instruments (such as bowed strings, blown, singing, etc.), when compared to discrete excited instruments (percussion, plucked strings, etc.). For the synthesis of singing, Bonada and Serra (2007)'s approach aims to overcome these limitations by means of performance sampling and by applying spectral techniques to the sample concatenation. Assuming that a performer has a *performance sonic space*[3], this approach builds a database by sampling this *space*. Then, the synthesis engine retrieves samples from the database and transforms them according to the score and a performer model.

Originally, the *MTG-SVS* synthesizer was designed to accept only a unique type of user input, working in an off-line manner. The input consists of a MIDI score plus lyrics, from which the system *renders* the output synthesized singing. The user can interact on a sequencer-like application, setting note and duration of the events on the sequencer sheet. For including the lyrics, each note has a editable field where the user writes the syllable. For each synthesized note, the system retrieves the most appropriate samples from the database based on a number of parameters (pitch, dynamics, adjacent notes, duration, etc.). Selected samples are time-scaled and concatenated in order to fit the specified note duration in the score. The time-scaling process affects only the vowel part of the diphone sample, leaving consonants with the original duration in the recording.

In addition to the score, other *MTG-SVS* control parameters can be manually edited on the GUI. The user can draw pitch and loudness contours, or add high-level expressive resources such as vibrato, or the type of articulation (e.g. "legato", "strong accent", etc.).

4.1.2 Control constraints

A first problem to deal with, is the lack of appropriate controllers for singing voice synthesizers[4], unlike for other instruments (such as wind-controllers for wind instrument synthesis). This fact restricts the generation of expressive MIDI score suited to the voice. But, what would be the

[2]Some technologies developed in the MTG-SVS synthesizer have been integrated in the Yamaha's Vocaloid Software Synthesizer. `http://www.vocaloid.com`

[3]The performance sonic space(Bonada and Serra, 2007) includes all possible sounds that a performer can produce when playing an instrument. Ideally, for creating a performer's database, this space is sampled with a sufficient resolution.

[4]Nevertheless, some gestural and imaginative controllers have been proposed in the past for controlling speech or singing voice (Fels, 1994; Kessous, 2004; Georgaki, 2004)

best controller for singing synthesis? Due to the unique characteristics of the performer-instrument interaction, the answer is not as evident as for other instruments. We depart from the assumption that a singing performance can be an appropriate control input, in what we call a *performance-driven* approach (Janer et al., 2006a). Performance-driven synthesis uses an input singing performance to infer the control parameters of the synthesizer (melody, dynamics and phonetic timing). Two operation modes are considered: real-time and off-line. As we have mentioned, the MTG-SVS was designed and developed as an off-line synthesizer in a sequencer environment, where the notes in the score contain the corresponding part of the lyrics, usually syllables. We encounter restrictions when using the MTG-SVS for performance-driven synthesis (both in offline an real-time), which can be summarized with the following points:

1. Fixed duration of consonant phonemes.
2. Latency in phoneme transitions due to the diphone samples.
3. Low-varying control envelopes.

On the one hand, the first restriction affects both operation modes, off-line and real-time. The target duration of a note is achieved by time-scaling an original sample in such a way that only the vowel part of the syllable is time-scaled. The consonant part of a sample keeps the original duration (see details in figure 4.1). It results in a loss of expression from the original performance, since the performer can only control timing at a syllable level and not at a phoneme level. On the other hand, the second limitation only affects the real-time mode. The problem is the trade-off between latency and sound quality. The concatenation of two diphone samples is achieved by joining the stable part of both samples. But in the real-time mode, the synthesizer only receives information from the phonetic alignment module when a new phoneme occurs. At that moment, the synthesizer can start the transition to the next phoneme. As we see in figures 4.1 (*bottom*), the rightmost part of the sample#1 has to be also synthesized to get a natural sound phoneme transition. Here, we can choose whether to compress the duration of the transition and get low latency, or to keep the original duration of the transition and get a natural sounding transition. In the figure 4.1 (*bottom*), the original duration is kept, introducing latency. The third constrain is related to the control envelopes, both in real-time and off-line. The MTG-SVS expects the control envelopes to be low-varying, since micro-variations of pitch and loudness are already present in the database sample. These micro-variations will include those dependent from the phonetics. It implies that the expression envelopes (pitch and dynamics) have to be low-varying but keeping the expression from the performer.

4.1.3 Proposed approaches

We propose two different approaches, one for off-line mode and one for real-time mode, that are based on the same block diagram (see figure 4.2). Our implementation consists of three independent

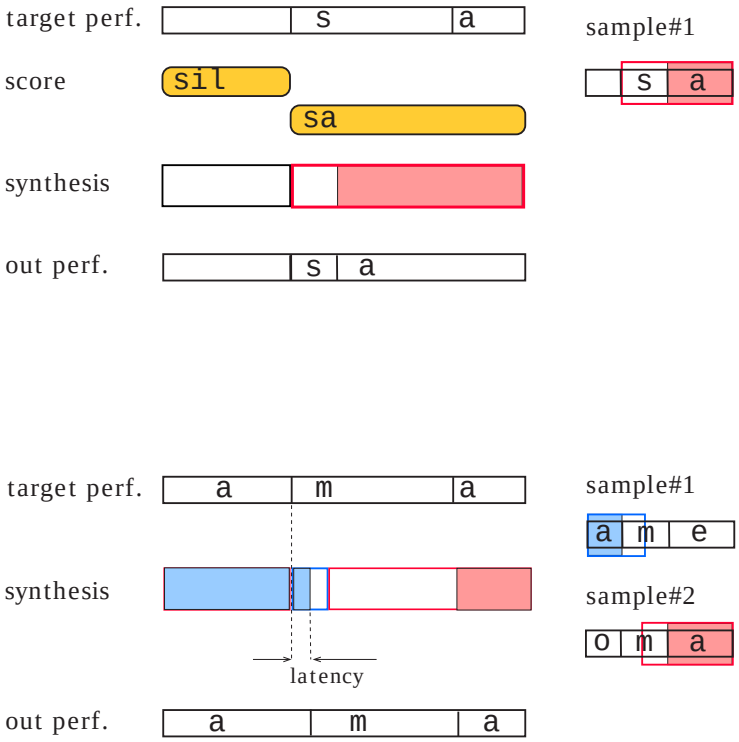

Figure 4.1: Control constraints of the MTG-SVS. On top, we see the effect of having a fixed duration for consonants. *Target performance* represents the phonetic timing of the performer. Score is the note sequence with corresponding syllable. *Synthesis* shows the sample concatenation, and *Out performance*, the final phonetic timing of the synthesis. At the bottom, we observe the effect of the latency due to the concatenation of two diphone samples. The latency corresponds to the portion of the transition /a-m/ corresponding to /a/.

modules: *Performance Analysis*, *Phonetic Alignment* and the actual *Singing Voice Synthesizer* (i.e. the MTG-SVS synthesizer). The inputs are a audio stream and a phonetic transcription of the lyrics in text format.

4.1.3.1 Off-line operation

In the off-line approach, the concept of latency disappears. Therefore, the sample concatenation between phonemes can be done smoothly, thus getting a better sound quality. Also, the results of the phonetic alignment is more robust, as we will see in section 4.3. Additionally, control envelopes for pitch and dynamics can be post-processed eliminating discontinuities (e.g. in note attacks).

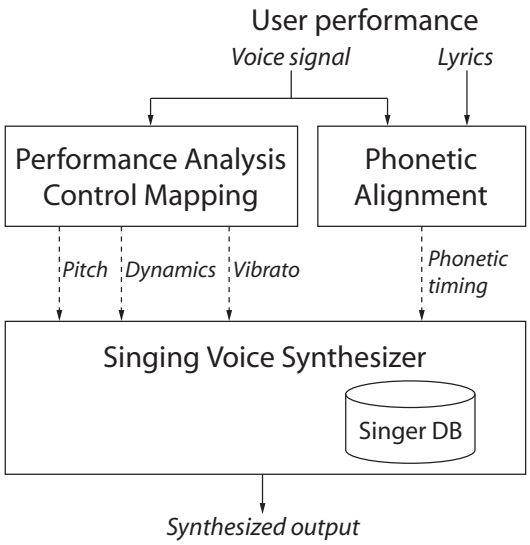

Figure 4.2: Block diagram of the complete system: Performance Analysis, Phonetic Alignment, and Singing Voice Synthesizer. In our implementation, vibrato information is conveyed by the pitch parameter.

4.1.3.2 Real-time operation

In terms of interaction, a digital musical instrument should work in real-time and with a latency as low as possible. Several studies have addressed the importance of low latency when designing digital instruments (Dahl and Bresin, 2001; Mäki-Patola and Hämäläinen, 2004). An otherwise good digital instrument may become totally unusable because of a large latency.

In real-time, the latency of the system has three sources, audio I/O, phonetic alignment and the internal synthesis latency. The first introduces 11 ms with an state of the art audio interface. The second is due to the MFCC computation process. Finally, as we can see in figure 4.1 (bottom), due to the diphone concatenation a small latency is introduced in phoneme transitions. For the current implementation, the latency is configurable ranging from $40ms$ to $250ms$, achieving acceptable sound quality with a latency of $75ms$.

4.2 Performance analysis

The *performance analysis* module extracts a set of descriptors that are mapped to the synthesizer's control parameters. Expression in singing voice could be principally described by loudness, fundamental frequency and spectral envelope, which all vary dynamically along time. In musical terminology, these descriptors can be referred as dynamics, pitch and timbre. In western traditional singing, timbre is mainly determined by the lyrics and the corresponding phonetics, and is addressed

in section 4.3.

4.2.1 Dynamics

As we have presented in section 5.3, *dynamics* is related to the loudness and it can be approximated from the signal's energy. Energy is computed directly as the root mean square of the windowed input voice signal. What one can observe by looking at the energy curve of a singing performance is that the instantaneous energy value is largely dependent of phonetics. Intuitively, we can see that an open vowel such as [a] will have more energy than a nasal consonant such as [m]. Actually it depends on how the placement of the articulators (vocal tract, tongue, mouth opening, etc.) affect the air stream circulation. Figure 4.3 depicts the energy envelope for the utterance [a-ma] (Audio 4.1) , representing two consecutive notes with the same pitch. When using energy as a estimation of dynamics, we are considering that the level of the input signal is constant. In reality, it might depend on the microphone distance and audio amplifier gain.

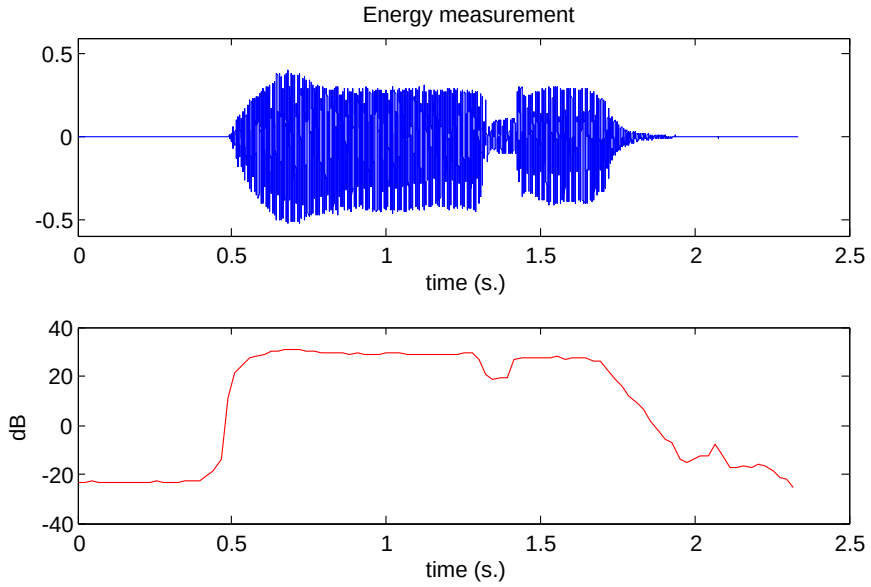

Figure 4.3: Energy measurement of the utterance. It decays for the phoneme [m], and rises again for the vowel [a]. Waveform (top) and energy plot in dB (bottom).

4.2.2 Pitch

Several techniques and methods for estimating fundamental frequency of audio signals have been extensively studied in the past. In our implementation, we use a the Two-Way-Mismatch (TWM)

technique as described by Cano (1998)(see also section 5.3). By looking at the pitch contour of the singing voice (figure 4.4), we can appreciate that, although the performer is here singing two notes of the same pitch, the pitch contour presents two type of perturbations. One type are the rapid variations known as *jitter*, which are provoked by the acoustic properties of the vocal folds vibration. The other perturbation on the pitch contour is due to the phonetics. In figure 4.4, the F0 estimation corresponds to the utterance [a-ga] (Audio 4.2) . Due to the position of the articulators for producing the sound [g], the pitch contour presents a valley before reaching the right pitch again on the second [a]. We explain is section 4.4.1 how to adapt this curve to the MTG-SVS pitch control parameter.

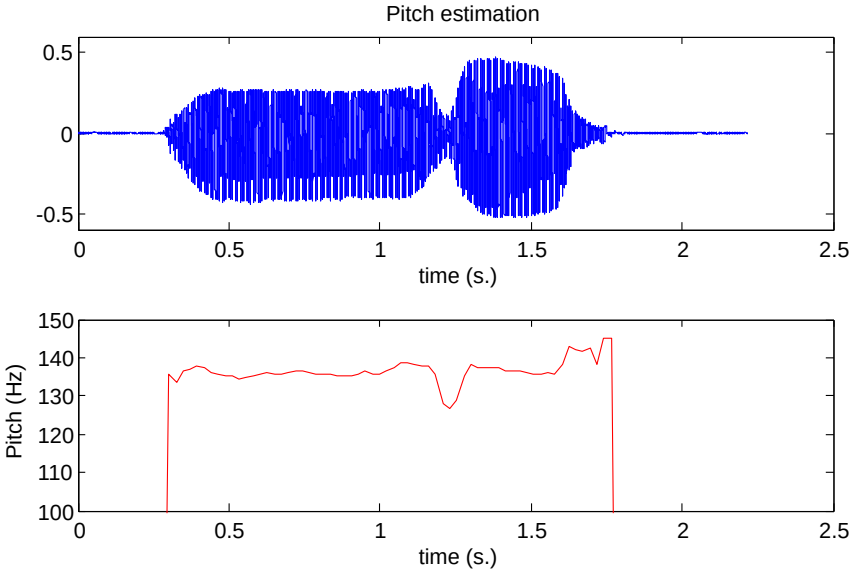

Figure 4.4: Although the two notes have the same pitch, due to the phonetics the pitch contour presents a valley during in the consonant [g]. Waveform (top) and pitch estimation in Hz (bottom).

4.2.3 Additional descriptors

Apart from dynamics and pitch, additional higher-level musical information can be extracted from the singing signal. One example is detecting the presence of vibrato in the signal by searching typical vibrato patterns (i.e. a quasi-sinusoidal component with a frequency in a range from $3Hz$ to $8Hz$) after filtering out slow fundamental frequency variations due to intonation, extracting vibrato rate and vibrato depth (Herrera and Bonada, 1998). Other examples include articulation aspects (e.g. portamento) and timbre aspects related to voice quality and not to phonetics (e.g. breathiness, roughness, etc.).

In our approach, though, we consider that performer expression is sufficiently conveyed by pitch and dynamics, where vibrato is taken directly from the pitch contour. Voice quality description is not addressed. As we mention in section 7.2, a higher-level description might be useful a context-aware voice-driven synthesis.

4.3 Phonetic alignment

Musical expression in the singing voice is described not only by pitch and dynamics but also by the lyrics. In western classical singing, scores usually assign a syllable to each note. For long notes, a vowel sound is sustained, while consonant sounds take place in the transitions. In our approach, the phonetic timing is also driven by the input performance. This section presents a method for phonetic alignment based on Dynamic Programming (Deller Jr et al., 1993)[5] that uses Mel-Frequency Cepstrum Coefficients (MFCC) as frame observations. At the same time, we introduce some modifications specially adapted to our context.

4.3.1 Dynamic Programming approach

The durations of the synthesized phonemes are derived from those in the input performance. We use a segmentation module The alignment module is based on Hidden Markov Models (HMM) and a Viterbi backtracking algorithm, as described by Rabiner (1989). Frame observations contain a vector of 25 MFCC ($c_1, \cdots c_{12}, \Delta\ c_0, \cdots, \Delta\ c_{12}$). Audio signal is resampled to $16kHz$, with a window of $25ms$. The method works with a frame rate of 86 frames per second (fps).

For each phoneme, there is an acoustic model, which is actually a hidden markov model of several states, in our case, three states. Most automatic speech recognition systems, use a library of acoustic models (AM) trained with several hours of speech recordings from a large number of subjects. In our framework, we would need acoustic models for Spanish, since the MTG-SVS uses a Spanish singer database. Unfortunately, all AM libraries in Spanish, we are aware of, are proprietary and not available[6]. Due to this limitation, we decided to use a free available Japanese AM library from the *Julius* Automatic Speech Recognition system (Lee et al., 2001). These AM's are coded as text in the HTK format[7]. Although the input singing performances are in Spanish, the Japanese AM's are also useful because the phonetics of both languages are somehow similar. However, it requires a previous step, where Spanish phonetics symbols are converted into the Japanese ones using SAMPA[8].

[5] Dynamic Programming is a mathematical concept for dealing with processes involving optimal decisions. In a discrete implementation, it seeks the best path (least-cost) between two discrete points or *nodes* in a grid. Deller Jr et al. (1993) describe carefully its applications to problems related to speech signals.

[6] Due to the huge amount of work and resources needed to record and annotate speech samples, AM are expensive and only affordable for big telecommunication companies or institutions.

[7] Cambridge University's Hidden Markov Model Toolkit (HTK). `http://htk.eng.cam.ac.uk`

[8] Speech Assessment Methods Phonetic Alphabet (SAMPA) is a computer-readable phonetic script based on the International Phonetic Alphabet (IPA).

The phonetic alignment task consists in, given a sequence of phonemes and an audio signal, finding the best path in a Finite States Network (FSN) that describes the audio signal. In figure 4.5, the sequence of states is distributed along the y-axis, while the x-axis represents the time, expressed in frame indexes. At each node (n, k) in the network, the cumulative cost function $C(n, k)$ can be expressed as:

$$C(n,k) = c_{local}(n,k) + \min_{m} \{c_{trans}\left((n,k),(m,k-1)\right) + c(m,k-1)\} \tag{4.1}$$

In the equation, c_{local} and c_{trans} are computed from the frame observations (MFCC vector) and the HMM model of a particular phoneme. For the off-line mode, we use the classical backtracking Viterbi algorithm (Rabiner, 1989). Each node stores the cumulative cost and its preceding node, so that starting from the last frame, the best path is found.

For the real-time mode, we rely on the Bellman's Optimality Principle (BOP) (Deller Jr et al., 1993)[9] to calculate the best path with L frames of delay. In the figure 4.5, from the best node found at frame k, we do the partial backtracking to select the best path at the decision frame $(k - L)$. In a first approach, we add a constraint by forcing the best path to be monotonically ascendant with transition of maximum one state. It means that from an arbitrary node (m, k), in the next frame it can only either stay in the same state $(m, k + 1)$ or jump to the next state $(m + 1, k + 1)$. The reason for this constrain was to give stability to the best path, avoiding jumps of several nodes, which would imply skipping some phonemes in the original sequence. In a low-delay situation, we observe that better results are achieved without this constrain (see section 4.3.2.2).

4.3.2 Modifications

For the performance-driven synthesis approach, we need to introduce modifications to the typical phonetic-alignment algorithm. First, in a performance situation, the singer follows the lyrics and the melody in a particular tempo. He adds then expression to his performance by introducing pitch and dynamics variation as well as controlling the phonetic timing, including the insertion of short pauses or silences. Unlike the first approach described before, where the performer is expected to exactly reproduce the sequence of phonemes as they are specified in the lyrics, we modify the algorithm to allow the insertion pauses between phonemes. Second, in a real-time situation, the latency has to be kept as low as possible. In this case, the best state in the FSN is computed without latency (input frame is the decision frame). Finally, the third modification for karaoke-like applications, we can improve the accuracy of the phonetic alignment algorithm by using a reference phonetic timing, manually annotated.

[9]The Bellman's Optimality Principle (BOP) is central to the Dynamic Programming algorithm. It was proposed by Richard Bellman and first employed in a speech processing problem by Sakoe and Chiba, as widely explained by Deller Jr et al. (1993).

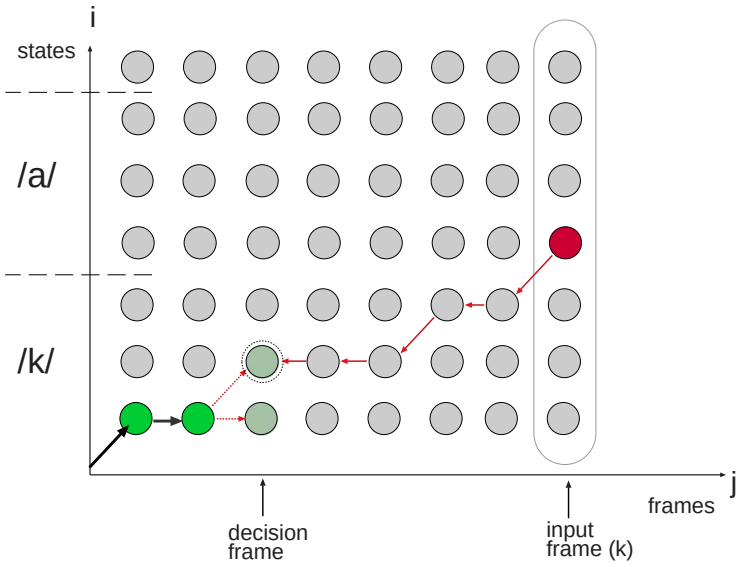

Figure 4.5: Finite state network for the phonetic alignment. From the best node in input frame k, we apply the backtracking until the decision frame $k - L$.

4.3.2.1 Insertion of silence

Basically, this modification consists in simply inserting a silence between two phonemes when building the Finite State Network (FSN). Then, the output state of one phoneme has two possible transitions, either to *silence* or to the next phoneme. Figure 4.6 shows the modified FSN. In fact this modification is of high importance to infer performer's expression in the phonetic timing of the synthesized singing. In singing performance, silences and micro-pauses are an essential part of the performer's style.

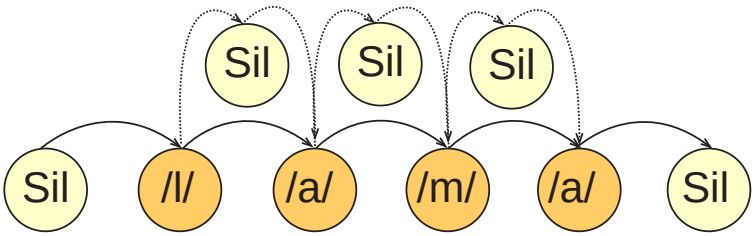

Figure 4.6: Example of phoneme transitions model for the utterance */lama/*. By inserting a silence model between two consecutive phonemes, pauses in the performance are detected.

4.3.2.2 Low-delay mode

Although in off-line mode, the classical backtracking algorithm is appropriate for the phonetic alignment, in real-time the latency should be kept as low as possible. The same problem has been previously addressed (Fonollosa et al., 1998; Loscos et al., 1999). To recall, it consists in setting the frames of delay to zero ($L = 0$), and thus not introducing latency. At each frame k, only the accumulated costs of the nodes of the previous frame $k - 1$ are kept. The best path is directly updated by finding the best node in the current frame k.

Moreover, in this mode we do not apply the monotonic constraint, so that from one state, the best path can jump to any state in the next frame, reducing error propagation. To improve the stability, we introduced a weighting function in the transition cost that penalizes large jumps in the FSN (see Fig. 4.7).

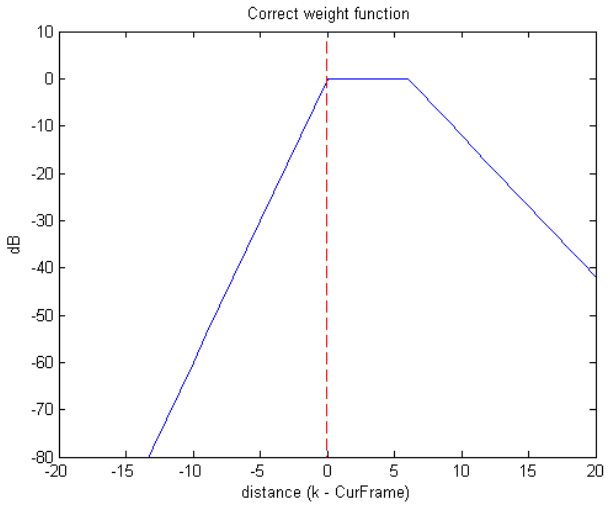

Figure 4.7: A weighting function penalizes distances from the current frame in the transition cost. Note that the transition to the next phoneme (distance ≤ 6 states) is not penalized.

4.3.2.3 Synchronized mode

As we stated before, one typical use case of our system is a karaoke-like application. Given a background music, the user replace the original singer by controlling a singing voice synthesizer. In those cases, we can improve the accuracy of the phonetic alignment module by forcing the best path to be in a certain region of the Finite States Network. This region corresponds to a time margin window from a reference path. This reference path, is either created from a manual annotation or from a MIDI score. The local cost probability of all nodes outside the region is set to zero ($-\infty$dB). Hence, the best path is forced to be inside the valid region, as depicted in figure 4.8.

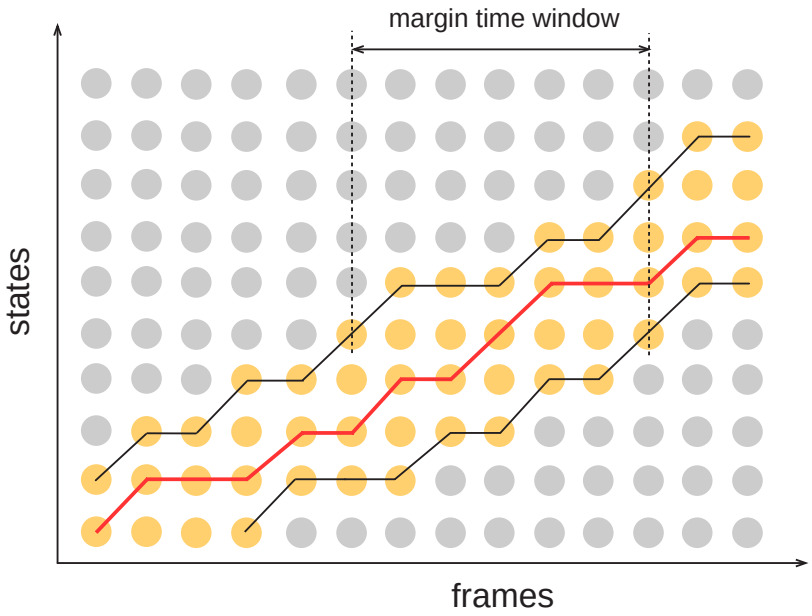

Figure 4.8: In the synchronized mode, all nodes outside a margin time window have are not considered as possible nodes. The reference path is marked with a red solid line.

4.3.3 Evaluation

This evaluation uses a test set of 116 short examples of a singing male voice in Spanish. All sound examples are taken from the same subject, having an average duration of 3.80 seconds with an average of 27.8 phonemes per recording. The recordings were carried out at the MTG.

To give a confidence measure of the phonetic alignment algorithm, we use two measures: *OverlapRate* (OvR) and *Accuracy*. The former is introduced by Paulo and Oliveira (2004) and this is the ratio between the number of frame that belong to that segment in both segmentation and the number of frames that belong to the segment in one segmentation. Accuracy is taken as the percentage of boundaries that are within a time window. Usually, this time error margin is about two frames long (in our case $23.2ms$)[10].

With relation to the presented results, we need first to make some remarks. These alignments implementation uses acoustic models of Japanese, while the data set is in Spanish. Although, we mentioned before that we can get acceptable results because of the relative similarity of Spanish and Japanese phonetics, the results cannot be compared to the state-of-the-art alignment systems. The main interest of our evaluations resides in observing the relative confidence measure for different configurations of our algorithm compared to the *Julius* stand-alone application, from which we take

[10] Although, this margin is slightly less restrictive than the typical $20ms$ at a frame rate of 100 frames per second, we use this in order to keep a resolution of two frames.

Configuration	Overlap Rate	Accuracy
Viterbi (Julius)	72.6123	76.9084
Viterbi	70.1935	73.6618
Look-ahead 278ms (24 frames)	55.9115	62.7402
Look-ahead 46.4ms (4 frames)	55.0630	63.9847
Look-ahead 11.6ms (1 frame)	54.1336	59.3800

Table 4.1: Offline alignment evaluation: accuracy and Overlap rate. Delta coefficients computation introduces a latency of two frames (23.2ms).

Configuration	Overlap Rate	Accuracy
Viterbi (Julius)	-	-
Viterbi	64.3987	64.8783
Look-ahead 278ms (24 frames)	39.5610	53.6739
Look-ahead 46.4ms (4 frames)	45.2747	57.2501
Look-ahead 11.6ms (1 frame)	49.6043	55.0553

Table 4.2: Offline segmentation Evaluation: accuracy and Overlap rate. Delta coefficients computation introduces no latency, resulting in a lower performance than the results from table 4.1.

the library acoustic models (AM). The approach of the Julius application uses a Viterbi algorithm with complete backtracking in an off-line manner and a frame time of 10ms. Our implementation uses a frame time of 11.6ms, which is the frame rate of the MTG-SVS. A different frame rate affects the computation of the delta coefficients ($\Delta\ c_{0\cdots12}$) and, thus, the performance of the algorithm is slightly degraded (see table 4.1).

Tables 4.1 and 4.2 show the results of alignment using the monotonic constrain. Offline alignment is compared to the Julius implementation and different latency configurations. In table 4.1, the delta coefficients consist of the first derivative of the MFCC, which uses the frames $k+i,\ i \in \{-2,-1,0,1,2\}$. In real-time, it introduces a latency of two frame (23 ms). In table 4.2, in order to reduce the latency, we compute the delta coefficients of a frame k using the frames $k+i,\ i \in \{-4,-3,-1,0\}$.

We have seen in the tables 4.1 and 4.2 that the performance of the proposed algorithm degrades for low latency conditions. Therefore, we introduced modifications (see section 4.3.2.2) where at each frame the cost probability is computed for all nodes in the network. From the results, the algorithm selects the best node independently of the current state. A drawback of this approach is that it is more computationally expensive.

Table 4.3 compares the low-delay mode to the offline alignment using a distance measure (Overlap distance). This distance is somehow related to the *Hamming Distance* used for string comparison. In our case, the *string* is a vector of a length equal to the number of frames. Then, for each frame we compute whether the phoneme is coincident with that of a reference vector, and get a percentage of correct frames for each audio example. The results shown in tables 4.3 and 4.4 are the average values of a data collection of 116 audio examples.

Configuration	Overlap Distance
Look-ahead 278ms (24 frames, monotonic)	61.27900
Look-ahead 46.4ms (4 frames, monotonic)	68.5986
Look-ahead 11.6ms (1 frame, monotonic)	69.4227
Low-delay 0ms (0 frames, without monotonic)	70.34867

Table 4.3: Low-delay evaluation using Overlap distance. The last configuration is without the monotonic constrain. Delta coefficients computation introduced a latency of 2 frames (23 ms).

Configuration	Overlap Distance
Look-ahead 278ms (24 frames, monotonic)	61.5853
Look-ahead 46.4ms (4 frames, monotonic)	64.1731
Look-ahead 11.6ms (1 frame, monotonic)	67.6128
Low-delay 0ms (0 frames, without monotonic)	68.3590

Table 4.4: Low-delay evaluation using Overlap Distance. Delta MFCC-delay is 0 frames and no heuristics are applied. Delta coefficients computation introduced no latency.

$$OverlapDistance = 100 \cdot \frac{CoincidentFrames}{TotalFrames}$$

In table 4.3, the internal MFCC delay is set to 2 frames, introducing a latency of 2 frames (23ms). In particular, this mode might be appropriate for those applications in which the latency is not critical. For the results in table 4.4, no latency is introduced by the MFCC computation. Figure 4.9 depicts the phonetic alignment of the manual annotated and automatic with the low-delay alignment mode (Audio 4.3) .

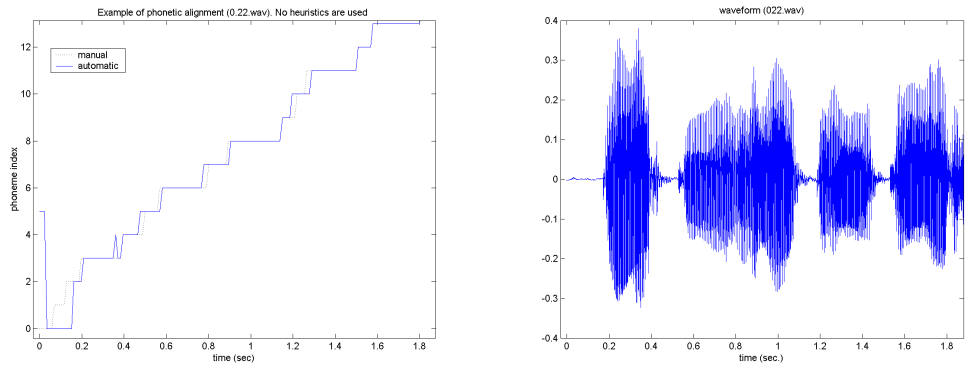

Figure 4.9: Example of phonetic alignment, with waveform (top), and the phoneme alignment with automatic and manual paths (bottom).

As a conclusion, we observe that for the low-delay situation the configurations with the monotonic constrain have a lower performance. Also, we see that by computing the MFCC vector without

latency, we can reduce the latency two frames (23 ms), lowering the performance only by a 2%. We build the observation vector with $c_1, \ldots, c_{12}$) from the current frame k and its first derivative, $\Delta c_0, \ldots, \Delta c_{12}$ from the frame $k - 2$.

4.4 Performance score

So far, we introduced the front-end modules of the performance-driven synthesis: performance analysis and phonetic alignment. This section addresses specifically the mappings to the synthesizer. Internally, the MTG-SVS has a set of low-level controls specifying frame by frame the synthesized output. The *phonetic timing* defines which phoneme sounds at which time and its duration. *Pitch* and *dynamics* determine target pitch and target singing intensity respectively and can be controlled directly or be derived from higher-level controls (e.g. MIDI notes and velocities) using models of pitch and dynamics curves. In our approach, pitch and dynamics are taken directly from the input voice. Expressive resources such as vibratos are already conveyed by the pitch curve.

From the *Performance Analysis* and *Phonetic Alignment* modules, we build an internal score, containing a sequence of samples and their transformation parameters for each frame. The optimal sequence of samples is chosen so the overall transformation required is minimized. The minimized cost function used is derived from: *1)* the compression or expansion of samples needed to fit the given phonetic timing, *2)* the pitch and dynamics transformations needed to reach the target pitch and dynamics curves and *3)* the continuity of the sample sequence.

4.4.1 Expression envelopes

The mapping layer consists basically in adapting the energy and fundamental frequency curves to the requirements of the synthesizer control layer by means of the *pitch envelope* and the *dynamics envelope*. The energy curve is smoothed in order to fit the *dynamics* control of the synthesizer. The fundamental frequency curve also has to be adapted. To retrieve a new sample, a distance is computed between the nominal pitch and dynamics values of the sample in the database and the pitch and dynamics envelopes of the internal score. This process is achieved internally in the MTG-SVS.

4.4.1.1 Pitch envelope

In section 4.2, we pointed out the characteristics of the F0 estimation curve. Actually, the MTG-SVS use samples that already contain pitch micro-deviations. Thus, it expects a smoothly varying the pitch curve. The first requirement for the pitch control parameter is to be continuous. Thus, in transitions and unvoiced phonemes when fundamental frequencies are missing, the curve is filled by smoothly interpolating the boundary values[11]. Also, the rapid and small fluctuations of fundamental

[11] In real-time when fundamental frequency is missing, the last non-zero value is kept.

frequency estimation not belonging to musical articulations, should ideally be removed. The reason is that these fluctuations are caused by the phonetics and how they are pronounced. Thus, they vary from one singer to another, and will be already present in the database samples. As a result, the computed pitch control is actually a smoothed and continuous version of the fundamental frequency estimation curve where musical gestures such as articulation and vibrato are kept.

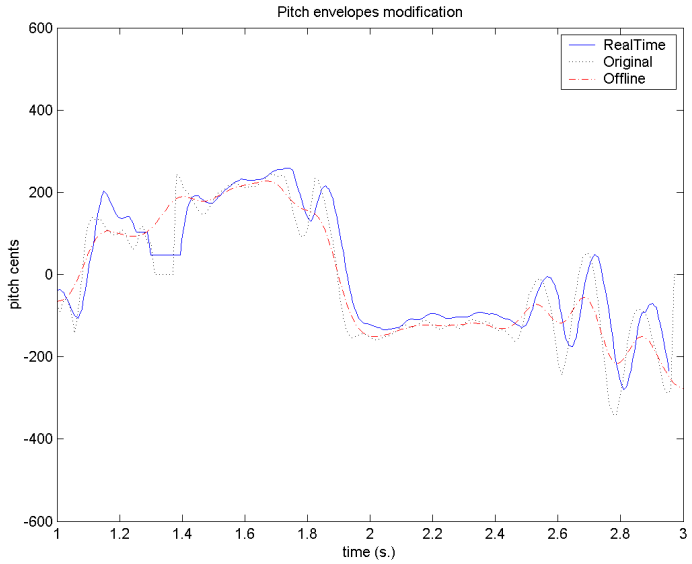

Figure 4.10: Pitch envelope[13]: pitch estimation, off-line envelope and real-time envelope. The latter is slightly delayed due to the smoothing.

4.4.1.2 Dynamics envelope

A similar approach is taken for the dynamics envelope. In the singer database, samples cover different dynamics, i.e. from *pp* to *ff*. Moreover, for dynamics, the amplitude of the synthesized sample is transformed according the the dynamics envelope. As shown in figure 4.3, phonetics affect the loudness estimation. In contrast, the synthesizer dynamics control parameter expects a value of dynamics, which is independent of the loudness of a given phoneme. Hence, we derive the dynamics envelope by smoothing the loudness curve in order to get rid of the phonetic variations. Note that the energy decay at the end is maintained.

4.4.2 Note onsets: voiced and vowel alignment

In western singing, notes in a musical score are often linked to a syllable. In this case, is the *vowel onset* that indicates the note onset time. Furthermore, singers often make use of expressive resources,

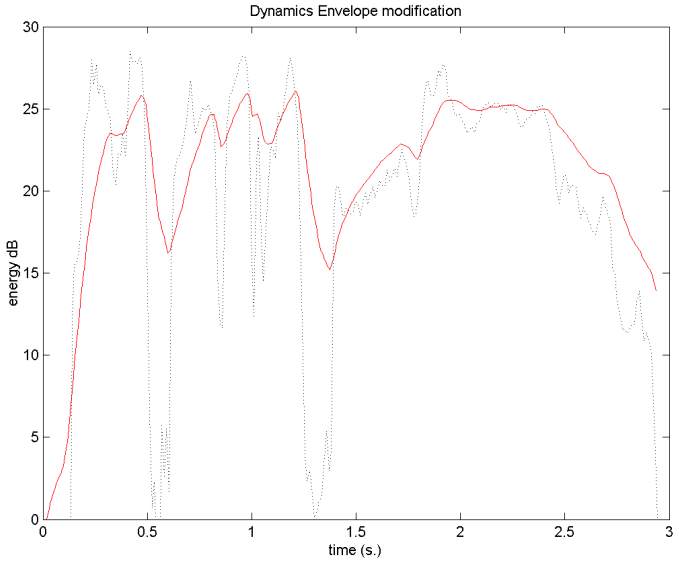

Figure 4.11: Signal energy in dB (dashed line) and the *Dynamics Envelope* (solid line), which is smoothed and slightly compressed curve.

such as portamento, that alter the alignment of the pitch contour and the uttered syllables from the original score.

The MTG-SVS presents some specific constraints, as described previously in section 4.1.2. Unvoiced samples cannot be time-scaled without introducing artifacts. Therefore, if we want to preserve the quality, only pitched (voiced) samples will be time-scaled in order to be aligned with the phonetics of the input, as we have seen in section 4.3.

We propose two alternative modes of generating the phonetic timing. In the first, voiced onset mode, only voiced phoneme onsets are aligned, while in the second, vowel onset mode, only vowel onsets are aligned. Both options ensure that unvoiced sounds have the same duration as in the database, i.e. they are not time-scaled. In the figure 4.12, we observe the waveforms of the input performance and the two proposed alternative phonetic alignments. From preliminary qualitative examples, one observes that with the voiced onset alignment the synthesized sound is closer to the input performance than with the vowel onset alignment. However, some artifacts are still noticeable in the transitions, affecting the sound quality. We argue that the vowel onset alignment offers an overall better trade-off between timing and sound quality.

In real-time, both the voiced onset and vowel onset alignment modes will introduce latency. For the former, unvoiced phonemes are first synthesized when a voiced onset occurs (e.g. /fa/). For the latter, the latency might be bigger since also voiced consonants are first synthesized when a vowel occurs (e.g. /ma/).

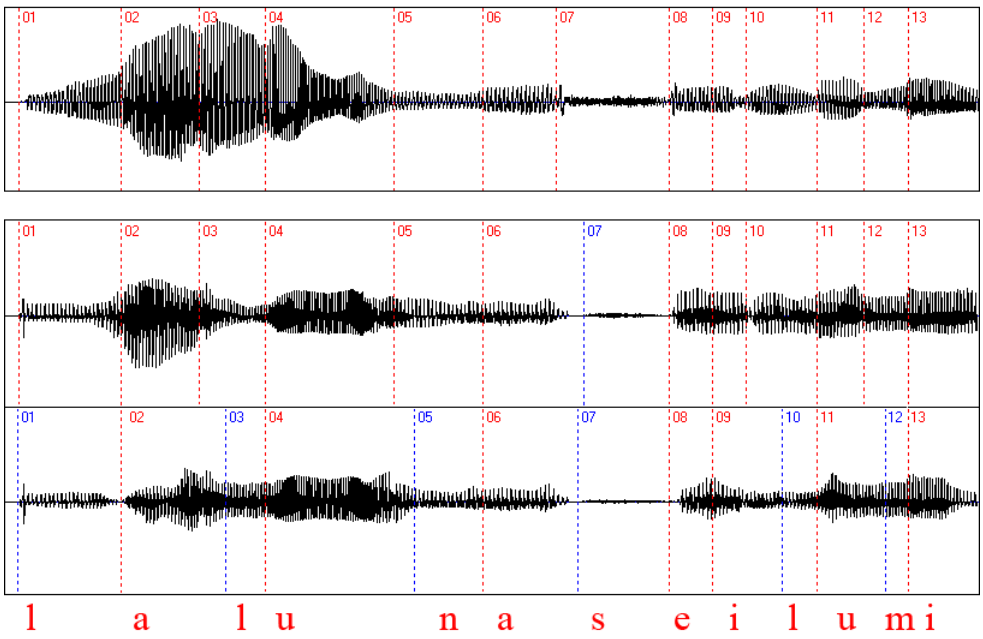

Figure 4.12: Two different modes of phonetic alignment: The second plot shows the synthesized output with all voiced phonemes aligned to the input segmentation (top plot). The bottom plot shows the synthesized output with only vowels aligned to the input segmentation.

4.5 Prototype

So far, we tackled the different steps involved in the design of a singing-driven interface for the MTG-SVS. This section presents a prototype that integrates both parts, interface and SVS, running in real-time. Two assessment sessions with experts are also described.

4.5.1 Implementation

For the control of the MTG-SVS in real-time, we implemented a plugin called *PDriven*[14]. Figure 4.13 depicts the GUI of the *PDriven* prototype. Features are summarized in the following list:

- VST plugin implementation: programmed in C++, it uses the VST API[15], which is a flexible framework for audio that supports real-time and works under different host applications and operating systems.

- Lyrics: the user has to previously input the lyrics, either by typing a text field on the GUI, or loading a text file. Currently, lyrics are required to be in SAMPA format.

[14]PDriven is a contraction of Performance-driven Synthesis
[15]VST is trademark of Steinberg, `http://www.steinberg.de`

- Singer database: the prototype synthesizes currently a single male voice in Spanish, although it can be easily extended with other singer databases.
- Functions:
  1. Start/Stop: the user decides when to start singing. A reset button allows starting the lyrics from the beginning again.
  2. Accompaniment: in karaoke-like situations the prototype allows playing back an audio file with background music.
  3. Synchronized mode: in karaoke-like situations with background music, the system can be synchronized with a reference phonetic alignment, previously annotated.
  4. Muted mode: the prototype can operate in silent mode, using only the phonetic alignment functionalities in other real-time applications such as animation or video games. Information is transmitted by means of OSC [16] messages.
- Configuration:
  1. Latency: the internal synthesis latency can be configured between 40 and 250ms.
  2. Synchronization: filename of the reference phonetic alignment and time window margin in seconds.
  3. OSC destination address: IP and port.

Figure 4.13: Screen-shot of the *PDriven* prototype. Users can type or load a text file with the lyrics in SAMPA format. In karaoke-like situations it can be supported with accompaniment music.

[16] Open Sound Control. `http://www.opensoundcontro.org`

Concerning the latency of the prototype, we have seen that the vowel onset alignment might delay note onsets. The synthesis engine has also an internal latency due to the sample concatenation process, which affects the sound quality of the synthesized singing. This internal latency is a configurable parameter, so that in off-line mode, it is set to a long value (e.g. 5 sec.), while for real-time mode, there is a trade-off between latency and sound quality (see section 4.1.2). In this latter case, a typical value is $70ms$. Smaller values imply that the phoneme co-articulation is shortened, which results in a less natural sound. The suggested latency for real-time is still bigger than the ideal value for a digital musical instrument.

4.5.2 Assessment

Evaluating musical controllers has typically been a controversial task, since it is difficult to define the tasks to accomplish. We already introduced this problem in section 2.1.3.4. In our context, a formal evaluation of the system should include tests with dozens of users of different background. For practical reasons, we opted to make an experiment with a reduced group of experts users. Basically, our evaluation is two-fold:

- *Perceptual experiment.* It consists of a subjective experiment, where users compare three synthesized versions of the same song fragment. In each version, pitch, dynamics and timing are controlled by different inputs corresponding to different instruments: voice, violin and piano.

- *Usability test.* We also conducted interviews, where users report their impression on different aspects of the prototype: latency, phonetic alignment and synthesis quality.

At the same time, we generated several examples demonstrating the control of the MTG-SVS. In this case, the singer database corresponds to a male in Spanish. In the first example, the voice input (Audio 4.4) is the same singer as the database used in the synthesized singing, for offline (Audio 4.5) and for real-time (Audio 4.6) . In the second example, a female voice (Audio 4.7) controls a male voice in the synthesized singing for offline (Audio 4.8) and for real-time (Audio 4.9) .

4.5.2.1 Perceptual experiment

For a given textual lyrics, the expression of singing synthesis is determined by the evolution of dynamics and pitch, as well as the timing. We aim to show here the advantages of inferring expression from a voice input. We compare three synthesized versions of the same song excerpt, inferring expression from voice (Audio 4.10) , violin (Audio 4.11) and piano (Audio 4.12) inputs (see figure 4.14). For the piano and violin inputs, the phonetic timing has been manually adjusted. The following audio examples were used for the perceptual experiment.

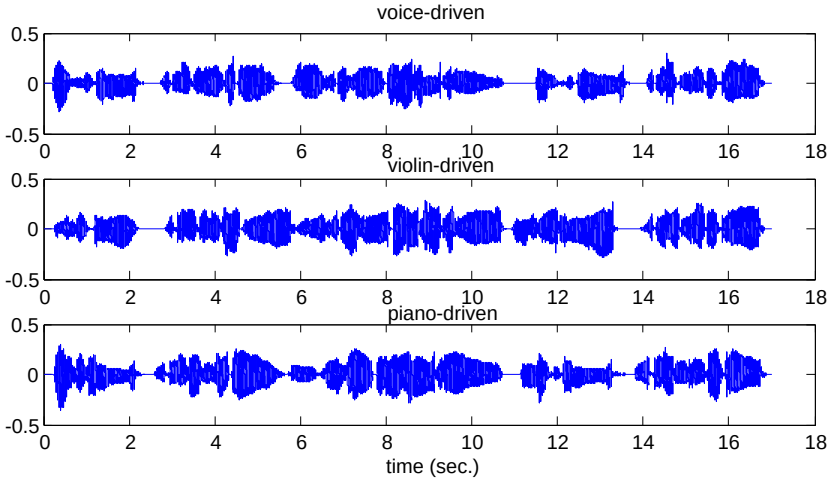

Figure 4.14: Expression assessment. Waveforms of the synthesized singing excerpt controlled by voice.

	Voice	*Violin*	*Piano*
Naturalness (1-5))	2.8	1.4	2.4
Correctly identified	3/5	3/5	5/5

Table 4.5: Results from the subjective experiment with 5 participants. Some participants misclassified the synthesis driven by violin and voice.

In a perceptual experiment, we asked five expert users[17] to first rate the "naturalness" of the three synthesized versions: voice (Audio 4.13) , violin (Audio 4.14) and piano (Audio 4.15) . Second, we explained the objective of the experiment and asked them to identify for each version the source instrument. Table 4.5 shows the results.

4.5.2.2 Usability test

A formal experiment for evaluating the *PDriven* prototype should include both professional singers and naive users, defining specific tasks to be accomplished. Although this cannot be considered a formal user test, we conducted a series of interviews with users in order to gather feedback about the limitations of the *PDriven* prototype and its components.

In the tests, we used the following set-up: a Pentium IV 2.4GHz, 1GB RAM (Desktop Computer), an Edirol UA25 (USB audio interface with ASIO drivers), a Shure SM58 (dynamic microphone), Edirol MA-7A (monitors) and Sennheiser eH 150 (headphones).

From the gathered answers, we identify two main problems when using it in a real-time situation.

[17] Participants in the listening experiment were music students from the ESMUC, Escola Superior de Música de Catalunya.

The first is the latency introduced by the synthesis engine (MTG-SVS). An acceptable sound quality is achieved with a latency of 70ms, while with a latency of 150ms the artifacts in sample concatenation are almost not noticeable. Obviously, a latency of 150ms is longer than the latency recommended for real-time musical system. A second problem is the errors in the phonetic alignment module. In the low-delay approach (section 4.3.2.2), phonemes might be skipped, altering the original lyrics in the synthesized sound. We have to stress that in the current prototype, the phonetic alignment module uses Japanese acoustic models to process a singing signal in Spanish. To improve the current results, we argue that a necessary step is to use Spanish acoustic models, either building our own database or acquiring a commercial database.

We believe that for a low-latency control of singing voice synthesis, the synthesizer should not be based on diphone samples, but having a more instantaneous response of the synthesis phonetics. Regarding the input voice analysis, in addition to phonetic alignment, a more continuous description of voice timbre such as formants, can provide an interesting control for the synthesized voice timbre.

4.6 Summary

The objective of this chapter was to address the use of a voice input to control a singing voice synthesizer. At the same time, it can be regarded as a first approximation to the general aim of this dissertation, i.e. controlling other instrument synthesizers. In this case, however, since both input and output are sounds of the same nature (singing voice), the involved mappings can be simplified. Expression in the input voice is represented by means of a pitch and dynamics evolution (expression envelopes), as well, as the phonetic timing (note onsets). For the latter, we introduced a low-delay phonetic alignment module for a real-time control. Finally, we developed a real-time prototype that integrates the contributed analysis methods with an existing singing voice synthesizer based on spectral concatenation. Further improvements involve the robustness of the low-delay phonetic alignment and the internal latency of the singing voice synthesis. The former would require the creation of acoustic models in Spanish, while for the latter, we should explore other synthesis techniques with a more instantaneous control of the synthesis timbre.

The next two chapters address the control of instrumental sound synthesis. We use the results from chapter 3, first, to design appropriate analysis methods for vocal instrument imitation signals; and second, to define the corresponding mapping strategies.

Chapter 5

Analysis of Vocal Gestures

The study of the vocal imitation of instruments in chapter 3 has given some cues about the characteristics of the voice input when controlling instrumental sound synthesis. In this chapter, we introduce the concept of *vocal gestures* as a way to describe this type of voice signals. It includes the development of voice analysis methods for formant tracking, signal segmentation and breathiness characterization.

5.1 Introduction

In the terminology of Phonetics, *vocal gestures* are related to the movement of the articulators (jaw, tongue, lips, etc.) in order to produce a sound (Lieberman and Blumstein, 1986). However, this chapter introduces the term *vocal gestures* in the same fashion that *instrumental gestures* describe performer actions in the literature (Cadoz, 1988). For describing *vocal gestures*, we employ signal analysis methods that extract attributes from the voice. Some of these methods represent original contributions, while in other cases we have implemented already existing techniques. Finally, we propose a representation of vocal gestures, which might allow more flexible and structured mappings to the sound synthesizers.

Instrumental gestures have deserved quite extensive attention for some years, both in musicology and from a technological standpoint. For the latter case, the objective has typically been two-fold, first to analyze gestural patterns in acoustic instrument performances, and second use this knowledge for controlling sound synthesis or effects in digital musical systems.

But the term *gesture* has various interpretations in the human-machine or human human literature, as Cadoz and Wanderley (2000) detail in relation to musical gestures. Actually, this term is seldom used in other research fields such as biomechanics where other terms such as *movement* or *action* are employed. However, there is a sufficient number of publications in the field of Music Technology that use the term *musical gesture*, which justifies the fact of referring to it also in this

dissertation. Métois (1996) gives a point of view regarding musical gestures that suits well in our purpose of addressing vocal gestures:

> *The gestures that are fed to the instrument are of physical nature (fingering, pressure, energy, etc.) whereas gestures resulting from our auditory perception are not. However, both present the ability to communicate musical intentions at a higher level than an audio waveform. The similarity of their level of abstraction motivated the author to label them both as* Musical Gestures.

Likewise, several authors attempt to describe musical gestures from different viewpoints. Ramstein (1991) proposes three approaches for instrumental gestures: phenomenological (descriptive), functional and intrinsic (from the performer point of view). Delalande (1988) suggests three levels: *effective* gestures (necessary to produce sound), *accompanist* gestures (body movements associated) and *figurative* gestures (perceived by an audience). We might find in the literature other nomenclature for the same classification:

- sound-producing or *effective*: those gestures necessary to generate sounds
- sound-accompanying or *accompanist*: ancillary conscious and unconscious movements.
- communication or *figurative*: directed to the audience or inter-performer communication.

Somehow related to Ramstein's *functional* approach, Cadoz (1988) considers only Delalande's *effective* or sound-producing gestures to describe and classify, what he calls, *instrumental gestures*. In the same publication, Cadoz proposes the following classification for instrumental gestures:

- excitation: provide energy to the instrument.
- modulation: modifies the properties of the instrument.
- selection: are related to a choice among similar elements of the instrument.

To better comprehend this classification, Wanderley (2001) presents several case studies. In the case of a cello, bowing is considered an *excitation* gesture, while the left hand is responsible for *modulation* gestures (string length) and *selection* gestures (string selection) at the same time. For a clarinet, breath pressure is an *excitation* gesture; lip pressure is a *modulation* gesture that changes vibrato; and fingering is a *selection* gesture. However, fingering in the case of *glissandi* could be also considered as a *modulation* gesture. Actually, as Cadoz and Wanderley (2000) admit, the classification of instrumental gestures is not trivial, and in many cases, gestures are interrelated, so that we can hardly separate them in different classes. It is the same situation for describing *vocal gestures*. In the context of voice-driven synthesis, we consider that by means of the voice a performer

Direct	*Indirect*
airflow pressure	energy
vocal folds frequency	pitch
articulators position	formant frequencies
phonation degree	breathiness

Table 5.1: Equivalence of direct and indirect acquisition parameters for vocal gestures.

is able to control wide range of sounds, in this case generated by a sound synthesizer. Therefore, we should be able to identify and describe *vocal gestures* following the categories as for other musical instruments: excitation, modulation and selection.

Besides the musical scope, in the field of Phonetics the term *vocal gesture* is commonly found in the literature. It describes those actions that take place in the voice organ to produce sound. It comprises the change of position of the articulators (tongue, jaw, lips. etc.) as well as actions in the glottis and the larynx. Concerning nonverbal communication in the scope of semiotics, *gesture* is addressed as body movements that may accompany speech in human interaction (Kendon, 2004).

Back again in a musical context, a relevant part of the study of instrumental gestures consists in the acquisition process. Gesture-acquisition may use a wide number of sensors and interfaces. Researchers distinguish two types of acquisition of sound producing gestures: direct (from sensor measurements), and indirect-acquisition (from an analysis of the acoustic signal) (Depalle et al., 1997). A good example is the work of Egozy (1995) on the clarinet and the analysis of timbre features in real-time. In our case, for extracting the sound-producing gestures of the voice, we will use indirect acquisition. It means that we only have access to the sound produced by the voice, which is captured by means of a microphone. Other alternatives to consider would be, for instance, to use EGG (electroglottography), which gives an accurate signal of the glottal activity. Although this type of acquisition is much more precise than indirect acquisition, we have also to bear in mind practical factors of the implications of our research. Patently, despite the limitations of indirect acquisition, using a microphone as unique input device appears the best option from both engineering and musical performance points of view.

All methods presented here intend to extract vocal gestures indirectly from a voice signal captured by a microphone. Thus, we do not describe the actual articulatory movements but their corresponding indirect equivalent referred throughout this dissertation as vocal gestures (see table 5.1). We consider vocal gestures as a subset of instrumental gestures, and in that sense, they can be grouped in three categories, following the classification found for instrumental gestures (Cadoz, 1988), i.e: excitation, modulation and selection.

It is important to remark that vocal gestures refer in particular to the nonsense text singing (or syllabling) related to the vocal imitation of instruments. Hence, the first consideration is to stress that in this work, vocal gestures do not intend to describe traditional singing. The second consideration relates to the classification in *excitation*, *modulation* and *selection* gestures. As

mentioned, among these categories there is a degree of interdependence. For instance, in classifying violin gestures, bow velocity *modulation* gesture is independent of string *selection* gesture. The same problem will arise for vocal gesture classification in section 5.3.

Yet another consideration concerns the acquisition method. Opposite to the majority of gesture based systems that use direct acquisition, vocal gestures derive from an acoustic signal, thus employing indirect acquisition. Despite these considerations, we find appropriate to employ the concept of vocal gestures in this research. First, it relies on the fact that we use the term gesture as a conscious action by the performer. Thus, we shall include in the voice gesture description those parameters that the performer is able to control. Second, by establishing a correspondence of vocal gestures and instrumental gestures for other instruments can facilitate the definition of mapping strategies, which becomes often an unorganized process.

5.2 Methods for voice description

As mentioned in the introduction chapter, there exists already a variety of audio analysis algorithms that can be directly employed to control sound synthesis. However, due to the nature of syllabling signals, these algorithms do not always offer the expected performance (e.g. onset detection). This section introduces new methods specifically designed for syllabling signals. First, we introduce a novel method for formant tracking, which should allow a dynamic controls of timbre. Next, we present a segmentation method for isolating vocal gestures, which is based on both phonetic and musical attributes. Finally, we introduce an algorithm to characterize the amount of breathiness in the voice signal.

5.2.1 Formant tracking

Our objective in the identification of formants is to get continuous descriptors of voice timbre. Actually, we simplify the problem by identifying only the first two formants frequencies. As we have earlier introduced in chapter 3, our input signals can be regarded as a sequence of syllables with a vowel as nucleus or central unit. With an automatic formant tracking, we aim to determine the vowel in order to control timbre attributes of the synthesized sound. We think that changing the vowels might be an appropriate control for manipulating the timbre of the synthesis. A preliminary method, based on the spectral centroid, shows an approximative estimation of the formant frequencies. Then, a novel approach for actual formant tracking is introduced. The method combines Discrete Cepstrum analysis and Dynamic Programming (DP). Compared to existing automatic formant tracking approaches, ours should fulfill specific requirements:

- It should work in real-time with low-delay
- It is oriented primarily toward vowel sounds

- It should favor robustness in continuity than in accuracy
- It should be consistent on high pitched voices

5.2.1.1 Preliminary timbre description

A low level feature commonly related to the perceived brightness of a sound is spectral centroid. Dealing with harmonic or pseudo-harmonic signals, as those presented in section 2.3.2.2, we can use a simplified formula. It takes the amplitudes a_k and frequencies f_k of the L harmonic partials (equation 5.1), and is referred in the literature as Harmonic Spectral Centroid (Peeters, 2003).

$$c_f = \frac{\sum_{k=0}^{L} a_k f_k}{\sum_{k=0}^{L} a_k} \tag{5.1}$$

By calculating the spectral centroid of the voice signal, one could map it directly to the brightness parameter in the synthesizer. Nevertheless, in order to have a higher level control of the timbre, a more refined strategy is to identify the two first formants, F1 and F2, defining thus a two-dimensional space. If we look at the characteristics of formant frequencies distribution, we find that typical values for $F1$ are between 200 and 800 Hz, and $F2$ goes from 800 to 2500 Hz. Having defined these two regions, we can calculate next the harmonic spectral centroid within each region, where $\tilde{F}1$ and $\tilde{F}2$ are an approximation to the formant frequencies.

However, we use a slightly modified equation for calculating $\tilde{F}1$ and $\tilde{F}2$. For L harmonic partials, the magnitude a_k is weighted by a "crossover-like" function $w_i(f)$ for splitting smoothly the two regions. The figure 5.1 depicts the spectrum of the vowel /a/ with a pitch of 80 Hz, and superimposed, the weighting functions w_1 and w_2. Finally, we come up with two equations for calculating the approximative values of $\tilde{F}1$ and $\tilde{F}2$,

$$\tilde{F}1 = \frac{\sum_{k=1}^{L} a_k w_1(f) f_k}{\sum_{k=1}^{N} a_k w_1(f)} \tag{5.2}$$

$$\tilde{F}2 = \frac{\sum_{k=1}^{L} a_k w_2(f) f_k}{\sum_{k=1}^{N} a_k w_2(f)} \tag{5.3}$$

This method works reasonably good for most vowel sounds, when the formants frequencies are separated enough and the second formant falls in the second region. However, we find also sounds in which the two formants are very close, and both in the first region. Particularly, this is the case of the vowels [o] and [u]. A partial solution implies to calculate an energy ratio between low and high frequency bands. For those sounds, in which the two first formants fall into the first region ($200-800$ Hz), the energy ratio ER_{12} in equation 5.4 between the two regions will be high. Thus, we can detect those cases and proceed with a recursive region splitting, that is, to find the approximate formant frequencies $\tilde{F}1$ and $\tilde{F}2$ using two new weighting functions $w_{11}(f)$ and $w_{12}(f)$.

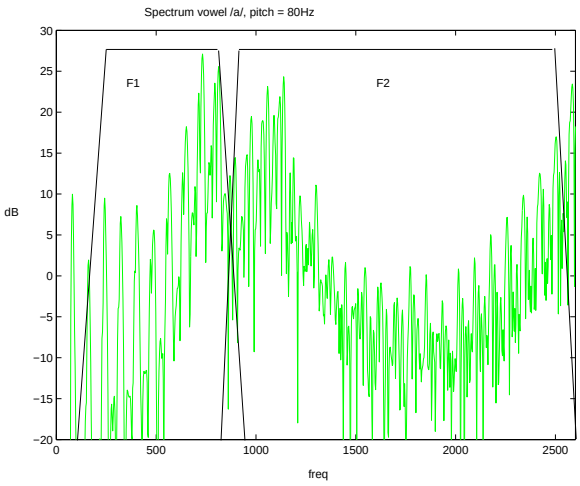

Figure 5.1: Spectrum of the vowel /a/ with $pitch = 80Hz$. Superimposed the weighting functions $w_i(f)$ for finding $\tilde{F}1$ and $\tilde{F}2$.

$$ER_{12} = \frac{\sum_{k=1}^{L} a_k w_1(f)}{\sum_{k=1}^{L} a_k w_2(f)} \tag{5.4}$$

The approximate formant frequencies, $\tilde{F}1$ and $\tilde{F}2$, take the centroid from the regions specified by $w_1(f), w_2(f)$ or $w_{11}(f), w_{12}(f)$ depending on the coefficient ER_{12}. At the end of the chain, an averaging filter is placed at the output in order to smooths the final trajectories.

Finally, it is important to stress that this preliminary experiment can only be considered an approximations to formant tracking. Actually, this algorithm based on spectral centroid describes the energy distribution and not the vocal tract resonances (formants). Next section, we introduce a method that indeed seeks the actual formant frequencies from the spectral envelope.

5.2.1.2 Method description

Several methods for parameterizing the spectral envelope of voice signals have been proposed in the past, such as Linear Prediction Coefficients, Cepstrum or the Discrete Cepstrum. Linear Prediction is an all-pole representation of the spectrum. Its main disadvantage is that for high pitched voices the spectral envelope fits only the first partials, which depends on the fundamental frequency. Also, spectral zeros are not estimated, which might be important on some cases. Cepstral Analysis models poles and zeros with equal importance. However, with small number of coefficients, Cepstral Analysis overestimates the bandwidth of the formants, which may lead to errors in the formant frequency tracking algorithm. Recent studies (Schwarz and Rodet, 1999) have shown that the most appropriate technique for getting the spectral envelope of periodic signals was the Discrete Cepstrum, first presented by Galas and Rodet (1990). This method takes a number of spectral

values, commonly, the amplitude of the harmonic partials, and calculates a spectral response that fits all the specified points. For voice signals with high fundamental frequency, among the other mentioned methods, this one appears to be the most convenient.

Concerning the smoothing and continuity of the estimated formant values, several approaches have been proposed. All these approaches rely on the fact that the vocal tract varies slowly compared to the analysis frame rate. Therefore, formant frequency values cannot vary drastically between two consecutive frames. Specially, Dynamic Programming has provided satisfactory performance for estimating formant trajectories (Talkin, 1987; Xia and Espy-Wilson, 2000). The trajectory is calculated by minimizing a cost function for a set of frequency candidates at each frame.

It is accomplished by providing several novel features:

- Cost functions use "anchor" formant patterns for typical vowels
- It uses Dynamic Programming to find the formant trajectories
- Spectral envelope is based on Discrete Cepstrum
- Transition cost is adapted to the spectral variation

The implemented method can be decomposed in the steps shown in figure 5.2:

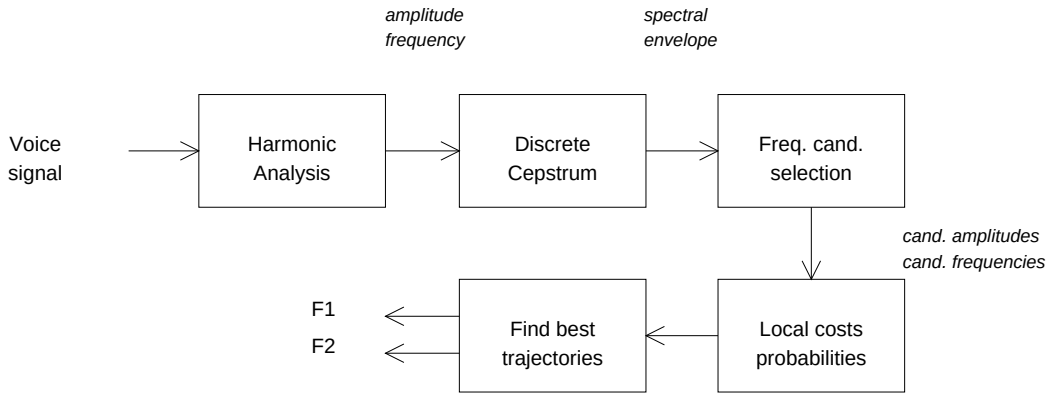

Figure 5.2: Block diagram of the formant tracking algorithm: *Harmonic Analysis*, *Discrete Cepstrum* Analysis, *Frequency candidates selection*, *Local cost computation* for each frequency candidate and *Trajectories computation* for F_1 and F_2 using transition cost probabilities.

Harmonic Analysis From the windowed signal, we compute the Short-Term Fourier Transform before that feeds harmonic analysis algorithm. Reliable results are attained with the Two-way mismatch method by Maher and Beauchamp (1994). Since frequencies of the first two formants lie below 4kHz, we limit the spectrum range from 0kHz to 4kHz. Currently, we use a window of 1024 samples, a hop size of 512 samples with a sample rate of 44100Hz.

Discrete Cepstrum Analysis We implemented the algorithm proposed by Galas and Rodet (1990). The principle is to calculate a set of cepstral coefficients so that the spectral response matches certain values, usually the harmonic partials. Given L amplitudes a_k corresponding to the frequencies f_k (usually harmonic frequencies), we find the real cepstrum coefficients $c_0, ..., c_n$, whose frequency response minimizes the error function in equation 5.5.

$$\epsilon = \sum_{k=1}^{L} |\log a_k - \log |S(f_k)||^2 \tag{5.5}$$

$$\log |S(f_k)| = c_0 + 2\sum_{i=1}^{n} c_i \cos(2\pi f_i) \tag{5.6}$$

Assuming $|S(f_k)|$ is real and symmetrical, its Fourier Transform is reduced to a sum of cosines. We have to minimize ϵ to find the vector c that contains the cepstral coefficients. We can write in form of a matrix:

$$\epsilon = ||\mathbf{a} - \mathbf{Mc}||^{\mathbf{2}} \quad \text{with} \quad a = \begin{bmatrix} \log(a_1) \\ \vdots \\ \log(a_L) \end{bmatrix} \tag{5.7}$$

$$\text{, and} \quad \mathbf{M} = \begin{bmatrix} 1 & 2\cos(2\pi f_1) & 2\cos(2\pi f_1 2) & \cdots & 2\cos(2\pi f_1 n) \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 2\cos(2\pi f_L) & 2\cos(2\pi f_L) & \cdots & 2\cos(2\pi f_L n) \end{bmatrix} \tag{5.8}$$

By zeroing the partial derivatives of the equation 5.6, the solution can be expressed as:

$$\mathbf{c} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{\mathbf{-1}} \mathbf{M}^{\mathrm{T}}\mathbf{a} \tag{5.9}$$

However, the resulting matrix $\mathbf{MM}^{\mathrm{T}}$ is generally badly conditioned and singular if the number of coefficients p is larger than the number of harmonic amplitudes L, ($p \geqslant L$), yielding meaningless results in some cases. Note that for high pitch, the value L(number of harmonic partials) might be low. To overcome this problem, Galas and Rodet (1990) proposed to replace each amplitude-frequency pair a_k, f_k by a cluster of points in order to add complexity to the minimization problem and finding a more consistent solution. In some cases, such as when in a broad frequency region no point is specified, this method was not satisfactory. Cappé et al. (1995) proposed a regularization technique that imposes additional constraints to the logarithmic envelope. This constraint seeks a *smooth* envelope, in other words, this is a penalization function that will be small if the estimated envelope is smooth and high if it contains high variations. Finally, the error function (equation 5.5)

is defined as:

$$\epsilon_r = (1-\lambda)\sum_{k=1}^{L} |\log a_k - \log|S(f_k)||^2 + \lambda\left(\int_0^1 \left[\frac{d}{df}\log|S(f_k)|\right]^2 df\right) \tag{5.10}$$

The parameter λ is the regularization parameter and controls the importance of the smoothness constraint. A typical value of λ is 10^{-4}. We can compute the final solution vector $\mathbf{c}$, which contains the cepstral coefficients as:

$$\mathbf{c} = \left[\mathbf{M}^T\mathbf{M} + \frac{\lambda}{\mathbf{1}-\lambda}\mathbf{R}\right]^{-\mathbf{1}}\mathbf{M}^T\mathbf{a} \quad \text{with} \quad \mathbf{R} = \mathbf{8}\pi^{\mathbf{2}}\begin{bmatrix} 0 & & & & \\ & 1^2 & & & \\ & & 2^2 & & \\ & & & \ddots & \\ & & & & n^2 \end{bmatrix} \tag{5.11}$$

Candidates selection From the cepstral coefficients, we can represent a smooth spectral envelope. In the next step, we select possible candidates from the set local maxima in the Discrete Cepstrum Envelope. In cases of a high pitched signal, when only few harmonic partials are available, the number of local maxima in the spectral envelope might be insufficient to get reliable results. In those situations, we add candidates by taking also the envelope inflection points.

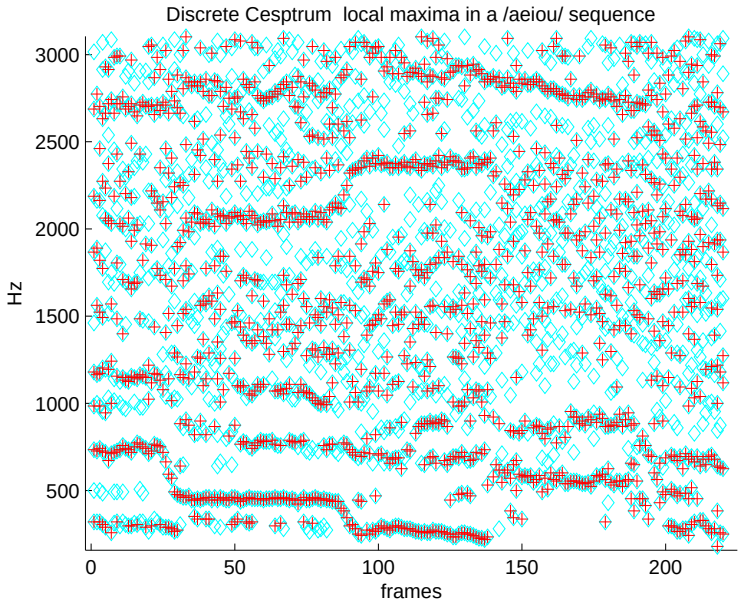

Figure 5.3: Voice sound example. Sequence of five vowels ([a],[e], [i], [o] and [u]). Frequency candidates are marked as crosses and local *maxima* of the discrete cepstrum as diamonds.

Vowel	F1	F2
[a]	750	1300
[e]	500	1800
[i]	300	2000
[o]	500	1000
[u]	300	700

Table 5.2: Typical first two formant frequencies for the selected five "anchor" vowel sounds.

Local cost probabilities Then, for all frequency candidates, we compute two local cost probabilities, one for F1 and one for F2 that evaluate the likelihood to be first and second formants. Local cost probabilities ($P_{cost,F1}$ and $P_{cost,F2}$) are computed using heuristic rules based on "anchor" patterns for typical vowels. The implemented heuristic rules cover mainly three aspects:

- Formant frequency in valid range $[200 - 2700]$Hz
- Local maximum in the spectral envelope.
- Sub-band energy distribution for typical vowel patterns.

From previous research on speech and the singing voice (Fant, 1960) we can assume that the first two formant frequencies will be bounded to a certain frequency range. For the first formant from $200Hz$ to $1500Hz$; and from $700Hz$ to $2700Hz$ for the second formant. Most formant tracking approaches rely uniquely on finding resonances in the spectral envelope. For our approach, though, in order to provide more robustness, it seems appropriate to look at the energy distribution in different frequency sub-bands for these vowels. As we already discussed in section 3.3, instrument imitation uses a reduced set of vowels. In syllabling, hence, we can assume that voice timbre will usually be around these "anchor" vowel patterns. In this sense, we chose only five typical vowel sounds that are common in several languages, which will give the reference formant frequencies for our estimator. The set of vowel sounds are [a],[e],[i],[o] and [u], as pronounced in Spanish. In the table 5.2, we can observe the typical values for the formant frequencies of the selected vowels.

Also, we divide the spectrum and calculate the band energy for four frequency sub-bands: 1)$[200-500]$Hz, 2)$[500 - 900]$Hz, 3)$[900 - 1700]$Hz and 4)$[1700 - 2700]$Hz. Then, we derive heuristic rules based on the probability that a formant frequency is present within the sub-band. These rules will rely on the typical energy patterns of the vowel sounds described before (table 5.2). The proposed rules for the sub-band energy distribution are summarized in the table 5.3. In order to get a more continuous behavior and make the algorithm more robust, the rules use gaussian probability density functions whose parameters σ, μ depend on each rule.

Trajectory calculation The estimated formant trajectories are based on Dynamic Programming, used already in formant tracking by Talkin (1987), where at each node the cumulative cost

band	vowel	F1	F2
1 $(200-500Hz)$	[i]	$E_1 > E_2 + E_3$	
	[e]	$E_1 > E_2 + E_3$	
	[a]	$E_1 > E_2 + E_3$	
2 $(500-900Hz)$	[u]	$E_4 > E_3$	
	[u]		$E_1 + E_2 > E_3 + E_4$
	[e]		$E_4 > E_3$
	[i]	$E_2 > E_1$	$E_2 > E_1$
3 $(900-1700Hz)$	[u]		$E_1 + E_2 > E_3 + E_4$
	[e]		$E_3 > E_2$
	[i]		$E_3 > E_2$
4 $(1700-2700Hz)$	[o]		$E_1 + E_2 > E_3 + E_4$
	[u]		$E_1 + E_2 > E_3 + E_4$
	[a]		$E_2 > E_1$

Table 5.3: Heuristic rules used for the local cost probabilities ($P_{cost,F1}$ and $P_{cost,F2}$) of each candidate. These rules are based on the sub-band energy distribution, and they differ depending on the frequency of the candidate.

function can be expressed as:

$$C(t,n) = c_{local}(t,n) + min_m \left\{ c_{trans}\left((t,n),(t-1,m)\right) + c(t-1,m) \right\} \tag{5.12}$$

At each frame, we update the grid of candidates and calculate the best path for F1 and F2. As shown in the figure 5.4, frequency candidates are distributed along the y-axis. The transition cost, c_{trans}, depends on the frequency difference:

$$c_{trans}\left((t,n),(t-1,m)\right) = 20 \log \frac{f(t,n)}{f(t-1,m)} \tag{5.13}$$

5.2.1.3 Evaluation

As a first step in the evaluation process, we test the algorithm qualitatively with real voice recordings. The sound examples consist of a vowel sequence test of a male voice at $120Hz$, and a short real performance excerpt. In the figure 5.3, we see the formant frequency candidates (crosses) and the local *maxima* (diamonds). A closer look allows us to identify five different regions that correspond to the five sung vowels. Figure 5.5 depicts the waveform of the sound example (Audio 5.1) and the estimated first two formant trajectories.

A first quantitative evaluation is achieved with a small set of syllabling signals, consisting of four phrases performed by one male and one female subjects. Table 5.4 shows the results of our

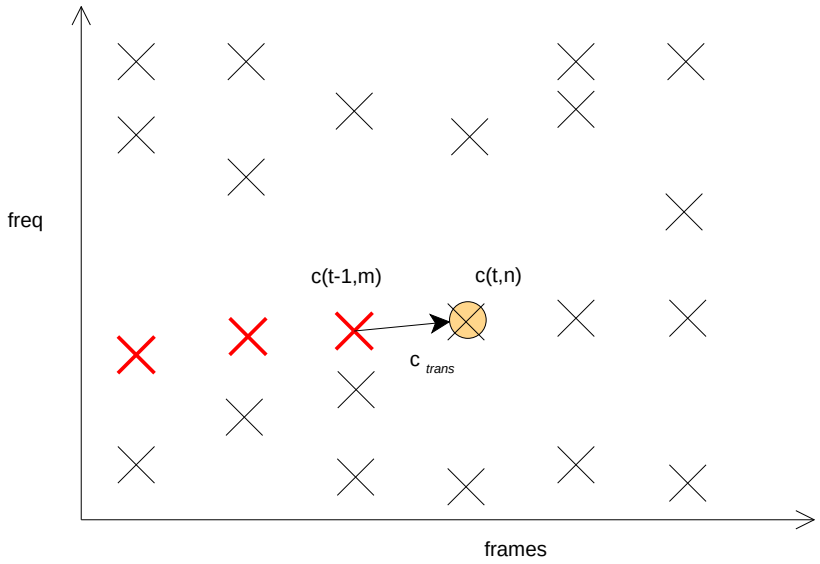

Figure 5.4: Crosses are formant frequency candidates with an accumulated cost $C(t,n)$, resulting from the sum of the previous one $C(t-1,m)$ plus the local and the transition costs. At each frame, formant trajectories minimize the cost function.

algorithm[1] compared to the algorithm integrated in the *Wavesurfer* software package[2]. By looking at the accuracy measures, both algorithms perform similarly for $F1$, while for $F2$ Wavesurfer's algorithm is more accurate. If we recall the results of table 5.5, we see differences between male and female results. While for male examples, both algorithms perform equally good, the accuracy degrades in female examples. The reason is that female singing signals are of very high pitch, resulting in a sub-sampled spectral envelope. In those cases, formant tracking algorithms tend to give the frequency of the second or third harmonic as second formant frequency. Moreover, ground-truth examples were generated using the Wavesurfer software, and by correcting manually the automatic formant estimation. It may explain the better performance of the latter, compared to our algorithm. Figure 5.6 depicts the results for both implementations for a sound example (Audio 5.2) .

Although the introduced algorithm for formant tracking is designed specifically to work with syllabling signals, we ran an extensive evaluation with speech signals. In this case, we use an annotated database with formant trajectories of speech signals. It was released in July 2006 by Li Deng and Abeer Alwan from Microsoft Research[3]. Speech data is taken from the TIMIT-ARPA

[1]We use the denomination of *Singterface algorithm* for the contributions of this dissertation, when they are compared to other implementations or systems.

[2]Formant estimation in *Wavesurfer* is achieved with a method based on dynamic programming(Talkin, 1987). `http://www.kth.se/wavesurfer`

[3]The MSR-UCLA VTR-Formant database is copyrighted by Microsoft Research. `http://www.microsoft.com/research/formants`

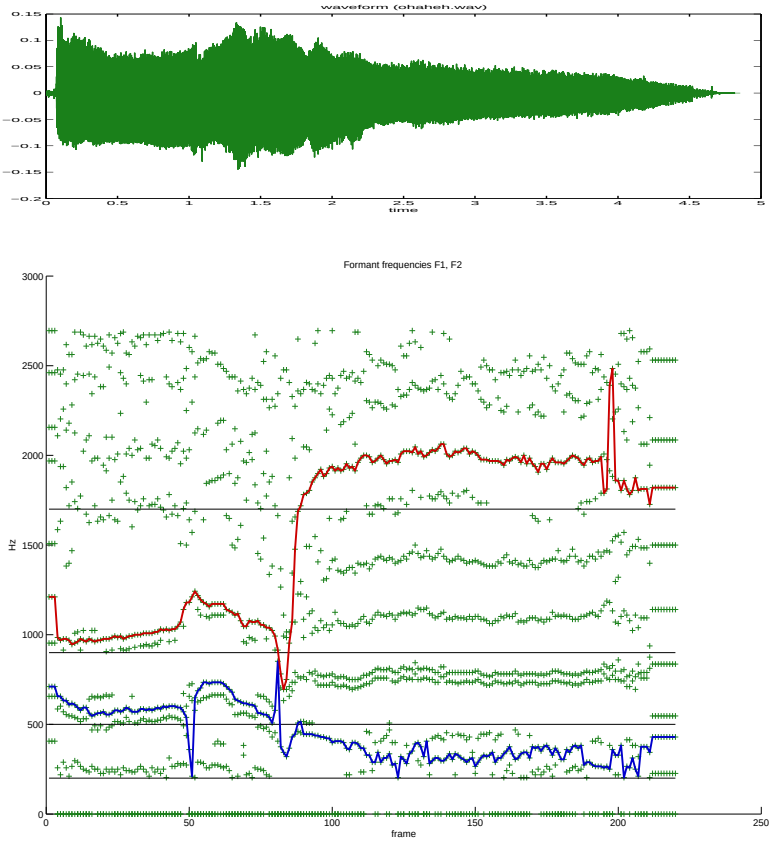

Figure 5.5: Real singing excerpt with the utterance [oh-ah-eh]. *top*) Waveform. *bottom*)First two formant trajectories (solid lines) are estimated based on the frequency candidates found for each frame (crosses).

speech corpus [4]. This corpus covers several American English dialects for both female and male, and the data format is PCM, 16-bit, $16kHz$.

The MSR-UCLA VTR-Formant database consists of two directories. A *Train* directory containing a total of 346 utterances in total, two utterances per speaker. A *Test* directory containing a total of 192 utterances in total, eight utterances per speaker. Typical evaluation measure for automatic formant tracking is the mean error, expressed in Hz. Another, informative measure is *accuracy*, which gives the percentage of frames in which formant frequency error is below a certain threshold (typically $200Hz$). In our evaluation, only frames detected as voiced were considered. Results are shown in table 5.6.

Clearly, the performance of our algorithm is not as good as expected when evaluated with a speech

[4]TIMIT speech corpus, NTISC, 1994

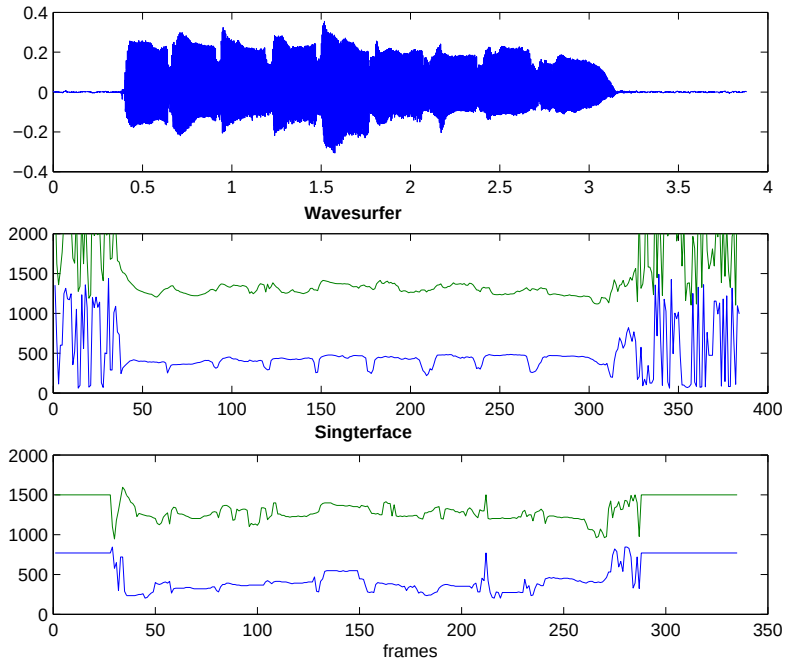

Figure 5.6: Automatic formant tracking for Wavesurfer and Singertface implementations. Audio excerpt consists of a note sequence, uttered with vowel [ae].

corpus. We identify two main reasons for such behaviour. First, our algorithm was designed for *syllabling* signals, which are significantly different from speech signals, as we have widely described in section 3.3. Basically, the ratio of vowel/consonants in number of frames (n_V/n_C) in syllabling is much higher than in speech, and our approach expects vowel sounds. Therefore, formant frequencies for voiced consonants such as nasals ([m],[n]), or liquids ([l], [r]), are not accurately estimated. And second, our implementation works in real-time with only one frame of delay. Other implementations based on dynamic programming, such as Talkin (1987), find the best formant trajectories by doing backtracking of the whole utterance. Backtracking provides robustness to the formant frequency estimation, and avoids errors in spurious frames.

5.2.2 Syllabling segmentation

In order to control a syntheszier, we are also interested in studying and describing the time structure of syllabling signals. Out of the scope of speech recognition, our aim is to exploit musical information conveyed by phonetics in vocal imitation of instruments. In the field of phonetics, we can find various classifications of phonemes depending on the point of view, e.g. from the acoustic properties

	Acc-200 (%)	Stdev	Error (Hz)	Stdev
Singterface				
F1	74.99	15.12	140.75	40.93
F2	38.16	24.64	602.375	398.44
Wavesurfer				
F1	76.48	20.21	128.68	81.28
F2	53.72	28.71	409.61	350.95

Table 5.4: Evaluation of automatic formant tracking with a small dataset of syllabling examples. Accuracy is computed with a frequency tolerance of $200Hz$. Results are compared to the Wavesurfer formant tracking algorithm (Talkin, 1987).

	Acc. male (%)	Stdev	Acc. female (%)	Stdev
Singterface				
F1	87.94	7.78	91.33	8.02
F2	69.06	25.67	28.37	11.01
Wavesurfer				
F1	80.28	1.89	89.91	15.79
F2	71.37	17.46	50.37	36.98

Table 5.5: Comparative table with accuracy measures for a *male* and *female* subsets. In this case, using a frequency tolerance of $300Hz$.

the articulatory gestures. A commonly accepted classification based on the acoustic characteristics consists of six broad phonetic classes (Lieberman and Blumstein, 1986): vowels, semi-vowels, liquids and glides, nasals, plosive, and fricatives. However, for our purpose, we might consider a new phonetic classification that better suits the acoustic characteristics of voice signal in our particular context. As we have learned from section 3.3, a reduced set of phonemes is mostly employed in syllabling. Furthermore, this set of phonemes tends to convey musical information. Vowels constitute the nucleus of a syllable, while some consonants are used in note onsets (i.e. note attacks) and nasals

	Acc-200 (%)	Stdev	Error (Hz)	Stdev
Singterface				
F1	63.238	14.338	174.1258	0.470
F2	38.105	18.949	381.215	1.3948
Wavesurfer				
F1	-	-	85	-
F2	-	-	149.66	-

Table 5.6: Evaluation of automatic formant tracking with the MSR-UCLA VTR-Formant database. Accuracy is computed with a frequency tolerance of $200Hz$. Error measures are compared to the Wavesurfer formant tracking algorithm (Talkin, 1987) in the experiments reported by Deng et al. (2006).

are mostly employed as codas. Taking into account syllabling characteristics, we propose a classification in: attack, sustain, transition, release and other (e.g. fricative). This classification allows us to translate phonetic information into musical information for the control of sound synthesis.

Traditionally, two different approaches have been used as a front-end for phonetic classification in automatic speech recognition systems (Lieberman and Blumstein, 1986):

- *Machine Learning*: statistical models such as Hidden Markov Models are trained with a large number of observations. Reliable results are usually achieved using Mel-Frequency Cepstrum Coefficients as input observations.

- *Heuristic Rules (Knowledge-based)*: this is a two-fold process. First extracting acoustic features from the signal; and second, to derive a set of rules using these features for achieving the classification.

In relation to traditional singing, the voice signal presents, in instrument imitation, distinct characteristics. Principal musical information involves pitch, dynamics and timing, which are independent of the phonetics. The role of phonetics is reserved for determining articulation and timbre aspects. For the former, we will use phonetics changes to determine the boundaries of musical articulations. Also, groups of notes can be articulated together, resembling *legato* articulations on musical instruments. Thus, we need to identify these grouped notes, often tied with liquids or glides. The figure 5.7 describes the model for syllabling segmentation. For the latter, phonetic aspects such as formant frequencies in vowels can be used to alter timbre of the synthesis in a continuous manner (e.g. controlling brightness).

CLASS	PHONEMES
Speech Phon. classes	
Vowels	[a] , [e] , [i], [o], [u]
Plosive	[p], [k], [t], [b], [g], [d]
Liquids and glides	[l], [r], [w], [y]
Fricatives	[s], [x],[T], [f]
Nasal	[m], [n], [J]
Syllabling Phon. classes	
Attack	[p], [k], [t], [n], [d], [l]
Sustain	[a], [e], [i], [o], [u]
Transition	[r], [d], [l], [m], [n]
Release	[m], [n]
Special	[s],[x],[T], [f]

Table 5.7: Typical broad phonetic classes as in (Lieberman and Blumstein, 1986), and proposed classification for syllabling on instrument imitation. In the second table, each class contains the phonemes that were most common in the web survey (see section 3.3.3).

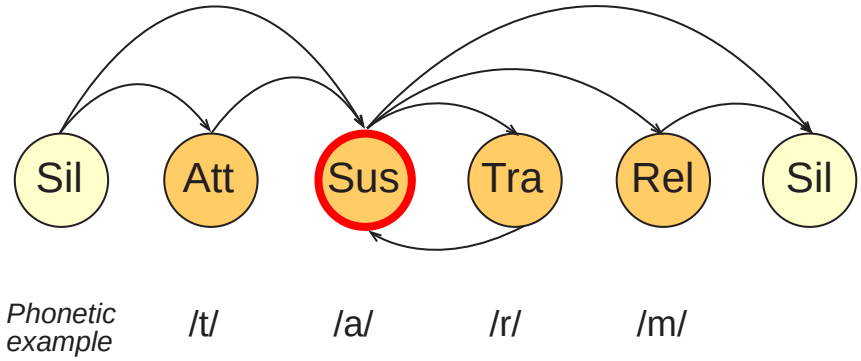

Figure 5.7: Model for the segment to segment transition for the different phonetic classes.

5.2.2.1 Method description

Our method is based on heuristic rules and looks at the timbre changes in the voice signal, segmenting it according to the phonetic classification mentioned before. It is supported by a state transition model that takes into account the behavior in instrument imitation. This process aims at locating phonetic boundaries on the syllabling signal, as shown in figure 5.8 (Audio 5.3) . Each boundary will determine the transition to one of the categories showed in table 5.7. This is a three step process:

1. *Extraction of acoustic features.*
2. *Computation of a probability for each phonetic class based on heuristic rules.*
3. *Generation of a sequence of segments based on a transition model.*

Concerning the feature extraction, the list of acoustic features includes: energy, delta energy, mel-frequency cepstral coefficients (MFCC), deltaMFCC, pitch and zero-crossing. *DeltaMFCC* is computed as the sum of the absolute values of the MFCC coefficients derivative (13 coeffs.) with one frame delay. Features are computed frame by frame, at a frame rate of 172 frames per second. This segmentation algorithm is designed for a real-time operation in low-latency conditions. From the acoustic features, we use a set of heuristic rules to calculate boundary probabilities for each phonetic class. In practice, we use five classes, where the *attack* class detects onsets with unvoiced plosive; the *sustain* class detects onsets with voiced phonemes; and the *transition* class detects note release with voiced phonemes. In addition to the *silence* class, a phonetic class labeled as *special* detects fricative sounds that might be used as effects.

In a first step, in order to generate continuous probabilities, and to attain a more consistent behaviour, we employ gaussian operators to compute a cost probability $f_i(x_i)$ for each acoustic feature x_i (see equation 5.14). Observe that for each acoustic feature x_i, function parameters μ_i and σ_i are based on heuristics. In the table 5.8, we list acoustic features used for the five considered

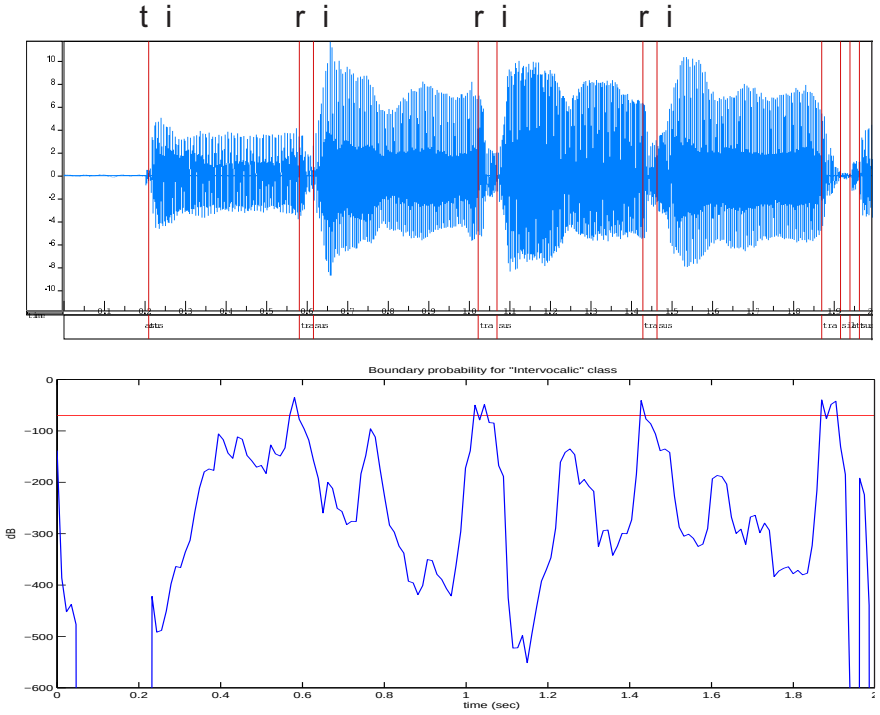

Figure 5.8: Syllabling Segmentation (from top to bottom): phonemes, waveform, labels and boundary probability for *transition* class (horizontal line representing the threshold b_{thres}).

boundary categories $B_j,\ j = \{1 \dots 5\}$. Then, for each boundary probability B_j, a weighted product of all acoustic feature probabilities is computed, with $\alpha_i = 1$ or $\alpha_i = 0$, whether a given phonetic class j is affected by a acoustic feature i or not respectively.

$$f_i(x_i) = \begin{cases} e^{\frac{-(x_i-\mu_i)^2}{\sigma_i^2}}, & x_i > \mu_i \\ 1, & x_i \leq \mu_i \end{cases} \tag{5.14}$$

$$B_j = \prod_i f(x_i)_i^{\alpha_i} \tag{5.15}$$

Boundary prob. (B_j)	*Acoustic features* (x_i)
Attack	energy, deltaEnergy, deltaMFCC, pitch
Sustain	energy, deltaEnergy, deltaMFCC, pitch
Transition	energy, deltaEnergy, deltaMFCC, pitch
Special	energy, zerocross, deltaMFCC
Silence	energy, deltaEnergy, pitch

Table 5.8: Description of the attributes use in the boundaries probability for each category.

This is a frame-based approach, computing at each frame k a boundary probability for each phonetic class j, $B_j(x[k])$. At each frame k, the maximum of all five probabilities (equation 5.16) is compared to a empirically determined threshold b_{thres}, to create a new segment boundary.

$$p(B) = \max_{1<5} [B_j(x[k])] \tag{5.16}$$

Finally, in order to increase robustness when determining the phonetic class of each segment in a sequence of segments, we use a finite state machine that determines the allowed transitions. The model of figure 5.9 is similar to the model in figure 5.7, but in this case, voiced attacks are labeled as *sustain*, and release as *transition*. For example, the utterance /na/ is transcribed as [*silence-sustain-sustain*]; and the utterance /pam/ will be transcribed as [*silence-attack-sustain-transition-silence*]. In section 6.3.1, we describe how to map the results of the segmentation to the musical articulation in the synthesis. For example, it permits to synthesize different sounds, distinguishing note onsets with a plosive (e.g. /ta/) or with liquids or nasals (e.g. /la/ or /na/). In addition, in off-line mode, if we want to be consistent with the first model shown (figure 5.7), one could apply a post-processing. This post-processing would look at the preceding and ending silence of note onsets and note releases, and change the labels accordingly.

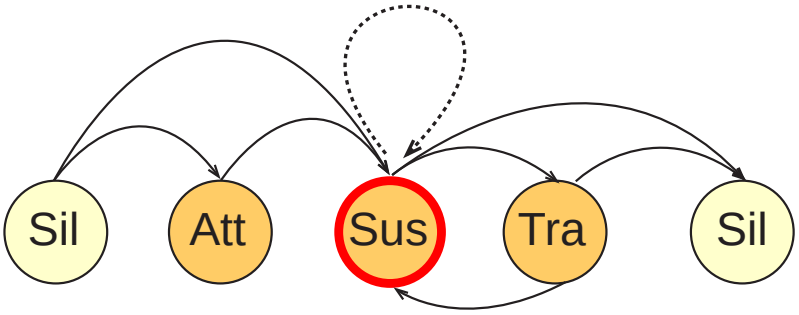

Figure 5.9: Implemented model for the segment to segment transitions, where the sustain to sustain jump (dashed arrow) is used for attacks with voiced phonemes. Examples: 1) /ta-ra/ is segmented as [*Sil-Att-Sus-Tra-Sus-Sil*]; 2) /na-na/ is segmented as [*Sil-Sus-Sus-Tra-Sus-Sil*]; and 3) /am/ is segmented as [*Sil-Sus-Tra-Sil*].

As an example of the performance of this algorithm compared to state-of-the-art pitch-to-MIDI system, figure 5.10 shows the segmentation of the same example (Audio 5.4) used in figure 3.5.

5.2.2.2 Evaluation

With the proposed method, we are able to segment effectively phonetic changes of voice signals in the context of instrument imitation. An evaluation of the algorithm was carried out by comparing

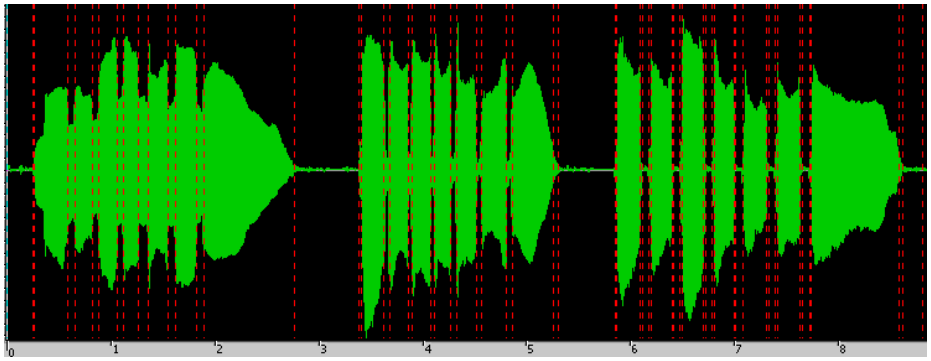

Figure 5.10: Segmentation of three variations of the same vocal melody using different phonetics.

automatic results with a manual annotated corpus (ground truth). The ground truth set consists of 94 syllabling recordings. Syllabling examples were voice imitations by four non-singers subjects of sax recordings with an average duration of 4.3*sec*. For the evaluation, onsets are considered those boundaries labeled as *vowel*, since it corresponds to the beginning of a musical unit. The annotated corpus is publicly available on the thesis website.

In the following tables (tables 5.9,5.10 and 5.11), the scores (%CD/%FP) are computed on each file, and then the average of the scores computed (each file has equal importance, regardless their length and the number of onset). For each algorithm the value "Averaged with equal onset weight" counts each onset (correct and false) across the whole collection and computes the overall score (each onset has equal importance). A difference of these two values might indicate that the performance varies depending on the number of onsets and the length of the file.

SINGTERFACE	Mean	Stdev
Correct detections (%)	90.68	15.38
False positives (%)	13.99	52.86
Deviation (ms)	−4.88	12.37
Absolute deviation (ms)	8.93	9.85

Table 5.9: Evaluation of the Singterface algorithm. Averaged with equal onset weight: Correct detection: 90.38 False positives: 9.78.

AUBIO	Mean	Stdev
Correct detections (%)	96.81	8.67
False positives (%)	109.81	105.69
Deviation (ms)	2.81	18.06
Absolute deviation (ms)	13.93	11.83

Table 5.10: Evaluation of the Aubio algorithm. Averaged with equal onset weight: Correct detection: 97.11 False positives: 99.52

ESSENTIA	Mean	Stdev
Correct detections (%)	89.20	14.56
False positives (%)	11.77	61.84
Deviation (ms)	17.44	20.29
Absolute deviation (ms)	22.78	14.04

Table 5.11: Evaluation of the Essentia onset detection algorithm. Averaged with equal onset weight: Correct detection: 89.57 False positives: 5.62

On the one hand, we observe that the general-purpose onset detection algorithm *aubio*[5] (Brossier, 2006) obtains higher correct detections that our algorithm, which denominated as *Singterface* in the results. However, comparing also the false positives, the *Singterface* algorithm obtains better results than the *aubio* algorithm (13.9% and 109.8% respectively). Therefore, considering both measures, the *Singterface* is more appropriate for our application since false positives would result in spurious notes in the synthesized sound. On the other hand, compared to the onset detector of the *Essentia* library [6], the results are very similar. Currently, the *Essentia* library operates only offline, therefore it is not appropriate.

Figure 5.11 depicts the performance of the three tested algorithms. Plots on the left represent the histogram of number of files with correct detections. Plots on the right represent the time deviations in milliseconds. The time deviation plot with sharpest peak corresponds to the *Singterface* algorithm.

[5] `http://aubio.piem.org`
[6] `http://mtg.upf.edu/pages/technology`

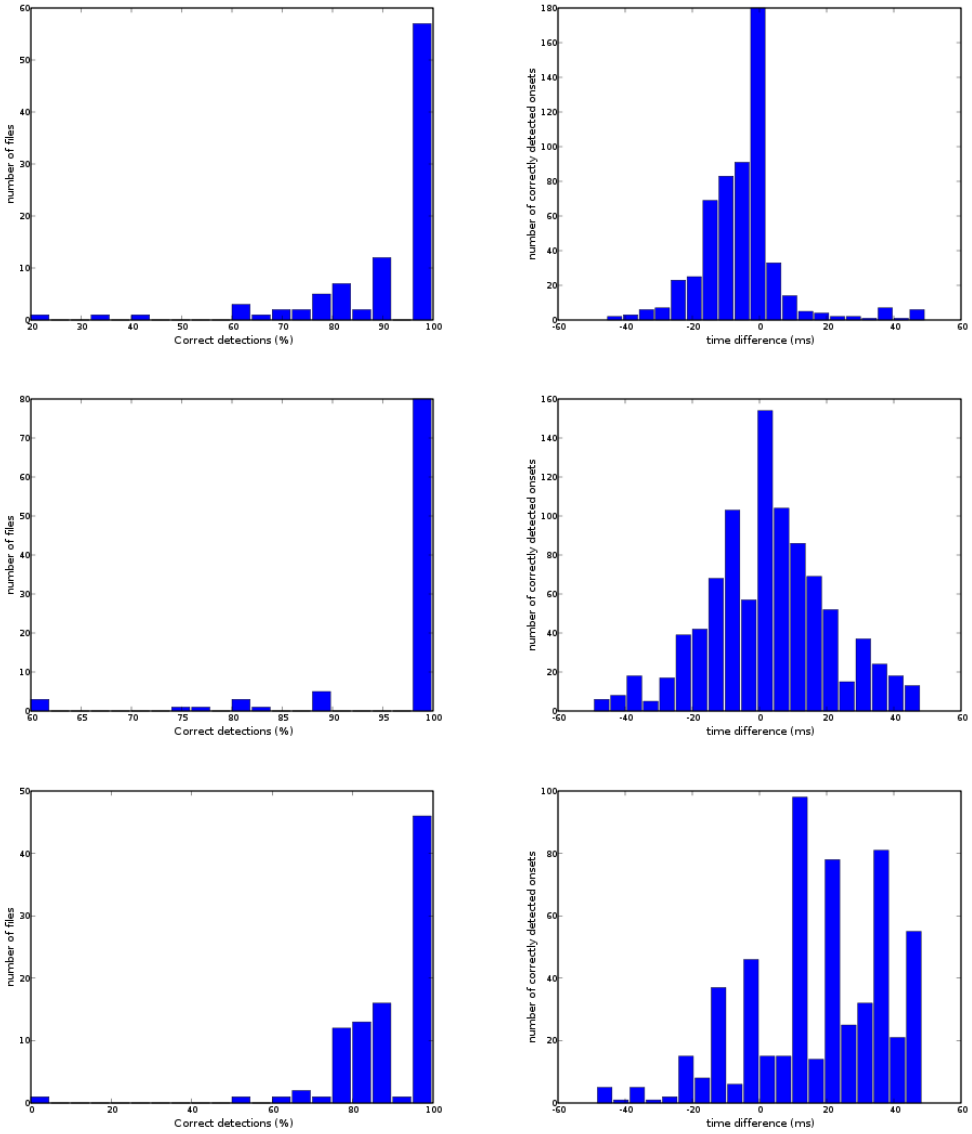

Figure 5.11: Histograms of the correct detections (*left*) and time deviations (*right*) for the three algorithms: singterface(top), aubio (middle) and essentia (bottom).

5.2.3 Breathiness characterization

A commonly used singing effect is the *breathy* voice. The breathy voice is caused by a particular mode of phonation. According to Sundberg (Sundberg, 1987), there is a *phonatory dimension* ranging from pressed to breathy phonation. When a high sub-glottal pressure is combined with a high degree of adduction, a pressed phonation occurs. If the sub-glottal pressure decreases, jointly with a lower adduction, it is called a *flow phonation*. Then, if the adduction force is still reduced, the vocal folds fail to make contact, producing the known breathy phonation. Perceptually, a breathy phonation results into an harmonic spectrum mixed with high-frequency noise, due to the air turbulence.

When we faced the characterization of breathiness, several methodologies in the spectral domain seemed to be promising. Here, we present and discuss three different techniques, carried out in collaboration with Loscos (2007).

- Harmonic-Peak Stability (HPS)
- Band Spectral Flatness (BSF)
- Harmonic to Spectral Envelope Area (HSEA)

It is necessary to mention that all these methods take the output from an spectral analysis. It provides signal's spectrum with a list of the detected harmonic partials, with corresponding amplitude, frequency and phase.

5.2.3.1 Method 1: Harmonic-Peak Stability (HPS)

This method observes the trajectories of the detected harmonic peaks within a frequency range from 3 to 6 kHz. As we can see in the figure 5.12 (Audio 5.5) , due to the presence of noise, the trajectories of the harmonic partials suffer of deviations in the mentioned range. For a frame k and an harmonic region r, we calculate a frequency stability factor (S_{fr}), and phase stability factor ($S_{\phi r}$), which are the difference between the measured frequency ($f_r[k]$) and phase ($\phi_r[k]$), and the predicted ones ($\hat{f}_r[k]$ and $\hat{\phi}_r[k]$). In the equation 5.19, we wrap the resulting predicted phase.

$$\hat{f}_r[k] = n \cdot pitch \tag{5.17}$$

$$S_{fr}[k] = |f_r[k] - \hat{f}_r[k]| \tag{5.18}$$

$$\hat{\phi}_r[k] = \phi_r[k-1] + 2\pi \frac{f_r[k] + f_r[k-1]}{2} \Delta t \tag{5.19}$$

$$S_{\phi r}[k] = |\phi_r[k] - \hat{\phi}_r[k]| \tag{5.20}$$

Finally, we add the two stability factors of all harmonic peaks, and correct the summation with the delta-pitch factor $d_{pitch}[k]$, being k the frame index (equation 5.23). The factor $d_{pitch}[k]$ will ensure that the final value is not affected by normal pitch variations such as in a vibrato.

$$S_{Peak,r}[k] = S_{\phi r}[k] + \frac{1}{40} S_{fr}[k] \tag{5.21}$$

$$d_{pitch}[k] = 1 - 100\,\frac{|pitch[k] - pitch[k-1]|}{pitch[k] + pitch[k-1]} \tag{5.22}$$

$$Breathy_{HPS}[k] = d_{pitch}[k] \frac{1}{R} \sum_{r=0}^{R} S_{Peak,r}[k] \tag{5.23}$$

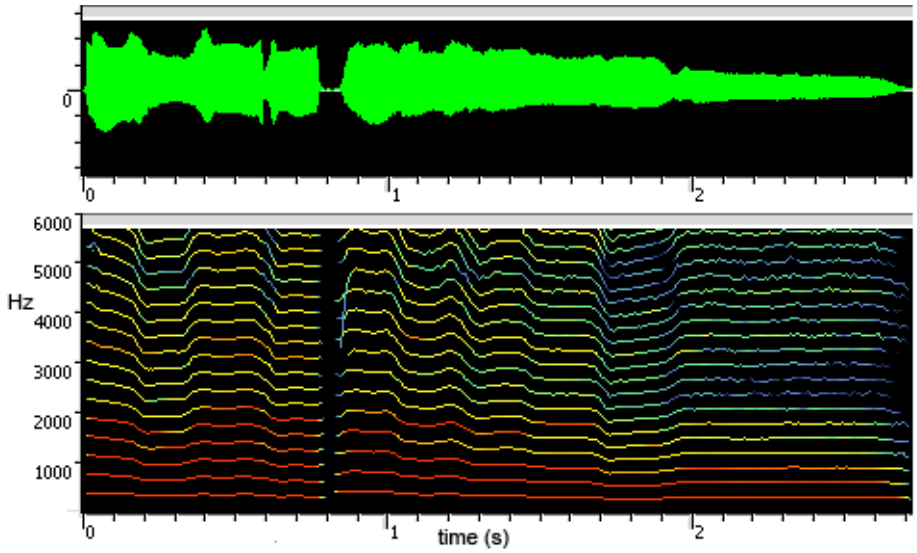

Figure 5.12: Evolution in time of the harmonic peaks in the frequency range from 1 to 6 kHz with signal's waveform on top.

5.2.3.2 Method 2: Band Spectral Flatness (BSF)

The second method proposed is based on the *Spectral Flatness Measure* (SFM). It is defined by Johnston (1988) as the ratio between arithmetic and geometric means of the spectral power density function, and computed directly from the FFT. After observing the spectral analysis of a breathy phonation, we realized that the spectrum had more presence of noise in the range from 2.5 kHz to 9 kHz. Thus, in order to estimate the amount of *breathiness*, we compute the Spectral Flatness in

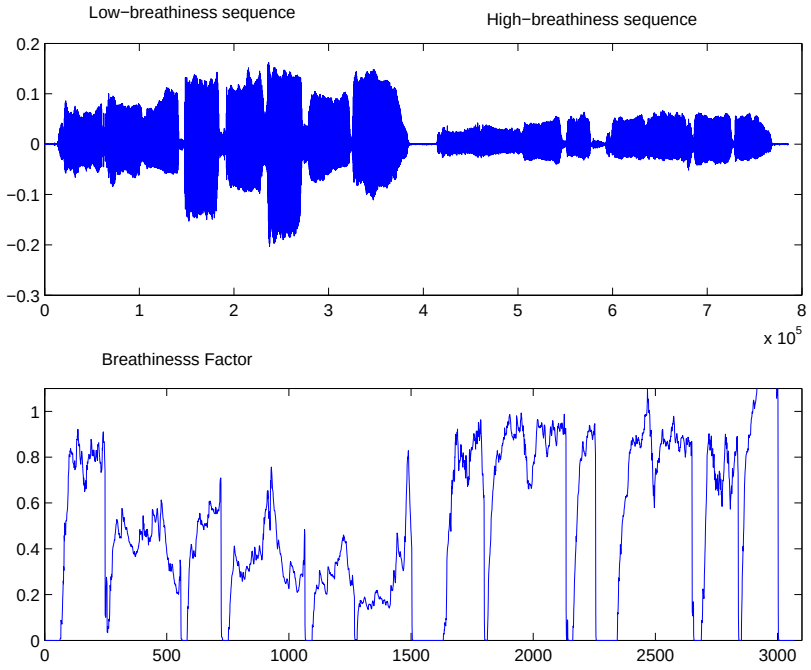

Figure 5.13: Breathiness Factor using the *Harmonic-Peak Stability* technique. The sound example is a musical scale with a regular and a breathy phonation.

the mentioned frequency band (equation 5.24).

$$Breathy_{flat} = \frac{(\prod_{k=N}^{M} spec[k])^{\frac{1}{M-N}}}{\frac{1}{M-N}\sum_{k=0}^{N} spec[k]} \qquad \begin{array}{l} N = \frac{2500 FFTSize}{SampleRate} \\ M = \frac{9000 FFTSize}{SampleRate} \end{array} \tag{5.24}$$

5.2.3.3 Method 3: Harmonic to Spectral Envelope Area (HSEA)

Finally, the last technique proposed takes also advantage of the harmonic analysis. It computes the area between the harmonic peaks envelope and the actual spectrum (*Envelope Differential Area*), as depicted in the figure 5.14.

The equation 5.25 gives us the initial formula for the amount of breathiness. In this case, experimental results showed that the best frequency boundaries were 4000 Hz and 9000 Hz. The estimated value of breathiness depends on the pitch and, for the example of figure 5.15 an additional correction function is used.

$$Breathy_{HSEA} = 1 - \sum_{N}^{M} \frac{envelope[k] - spec[k]}{n\,bins} \qquad \begin{array}{l} N = \frac{4000 FFTSize}{SampleRate} \\ M = \frac{9000 FFTSize}{SampleRate} \end{array} \tag{5.25}$$

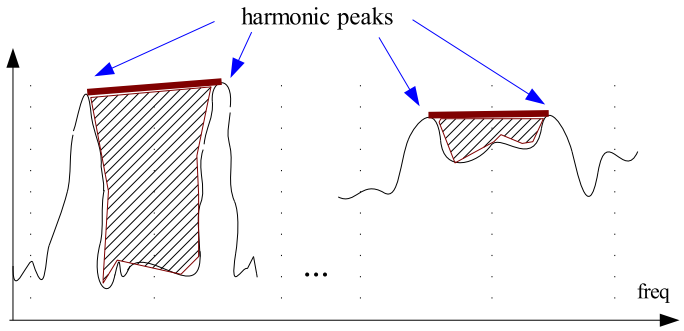

Figure 5.14: Envelope Differential Area. The Peak-Envelope consists of a linear interpolation between two harmonic peaks.

5.2.3.4 Results

The computational cost of the first method is proportional to the number of the harmonics in the spectrum, while for the second and third methods, it is proportional to the number of FFT bins. In a preliminary qualitative evaluation, the three methods are able to detect the presence of breathiness. However, they present different behaviour, as one can observe in figure 5.15, where a musical scale is sung twice, first in a regular way, and then with a high breathy phonation (Audio 5.6) . The third method is influenced by the pitch, presenting a slope due to the musical scale. Finally, although the first and the second methods present similar results, the first one (HPS) has the advantage of a lower computational cost.

5.3 Vocal Gestures representation

We introduced some specific methods for describing syllabling signals. This section intends to represents these signals as vocal gestures in the same fashion that instrumental gestures describe performer actions. Vocal gestures are therefore classified into three categories: excitation, modulation and selection. Also, we propose to code vocal gestures in the GDIF format [7], as a way to integrate the research carried out into other projects.

5.3.1 Excitation gestures

The source/filter model for voice production defines an excitation signal (a train of glottal pulses) that passes through a resonant filter (vocal tract). As observed in section 2.3.1, this excitation signal is pseudo-harmonic, and it depends on the airflow pressure and the vocal folds tension in the glottis. However, as explained before in this chapter, according to the terminology used for instrumental gestures, *excitation* refers only to those gestures that introduce energy to the acoustic

[7] GDIF stands for Gesture Description Interchange Format (Jensenius et al., 2006).

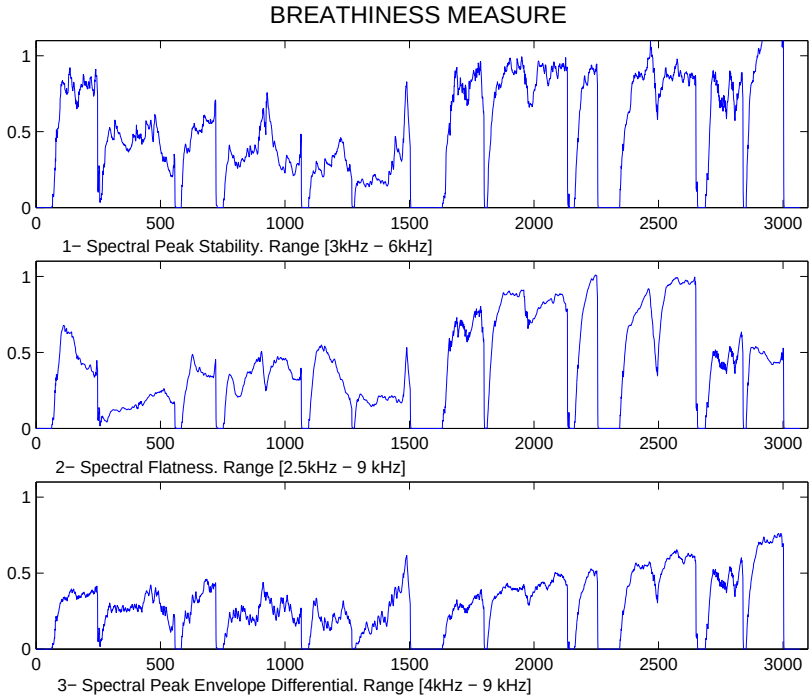

Figure 5.15: Comparative test. Top) Harmonic-Peak Stability; middle) Spectral Flatness; bottom) Envelope Differential Area.

system. An illustrative example is the bow velocity or bow pressure in the violin sound-producing gestures. When dealing with vocal gestures, though, our position will be to consider only *loudness* as an excitation gesture, and consider *fundamental frequency* as a modulation gesture.

5.3.1.1 Loudness

A simple way for estimating loudness derives it from signal's short-term energy. When using energy as a estimation of loudness, we are considering that the level of the input signal is constant. In reality, it might depend on the microphone distance and audio amplifier gain[8]. The energy estimation procedure takes a block of audio samples and calculates the mean square value after removing the mean or offset (μ).

$$E = \frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \mu)^2 \tag{5.26}$$

[8]In any case, we should not forget that singers use often the microphone distance as an expressive resource in a performance situation.

5.3.2 Modulation gestures

In terms of control, *modulation* gestures describe those attributes that change in a continuous manner. Specifically, the gestures considered are fundamental frequency, formant frequencies and breathiness.

5.3.2.1 Fundamental frequency

In voice signals, the fundamental frequency f_0 may present rapid and constant changes due to its production mechanism. Early techniques were developed for speech processing (Rabiner and Schafer, 1978), but we employ here a estimation method based on the Two-Way-Mismatch (TWM) technique as described by Cano (1998). It incorporates some modifications to the original algorithm by Maher and Beauchamp (1994) in order to improve f_0 estimation with a pitch dependent window size, and f_0 continuity. During the note attack (onset), the signal may have a fairly unstable instantaneous f_0. It is particularly critical in our context, since it may introduce latency in the sound output of a voice-driven synthesis system. These algorithms require a window comprising number of periods, or more robust period estimation. This fact is accentuated for lower pitch, since we need longer period before determining the pitch, leading to a bad time resolution.

Figures 5.16 and 5.17 compare a real-time implementation of the TWM algorithm with the YIN algorithm (de Cheveigné and Kawahara, 2002), both the original Matlab implementation (offline) and an implementation (real-time) developed by Kellum (2007). Basically, our interest is to observe the performance of these algorithms in terms of latency and accuracy.

As shown in figure 5.16 (Audio 5.7) , although the original implementation of the YIN algorithm is accurate, the real-time implementation used in this test presents "octave jumps". For a real-time system, we use the TWM method that has a latency of half a window size plus one frame size. In this case: $1024 + 256$ samples.

5.3.2.2 Formant frequencies

In speech and singing voice, a timbre description refers mainly to two facets, the quality of the voice (hoarseness, breathiness), and those related to the vocal tract. Concerning the latter, the formants are resonances that form an acoustic filter for the glottal pulsed signal. In the vocal tract, the two largest spaces are the throat and mouth. Therefore, they produce the two lowest resonances or formants. These formants are designated as F1 (the mouth) and F2 (the throat/pharynx). Upper formants (F3 to F5) are primarily dependent of the speaker's anatomy. We argue that controlling the F1 and F2 can be useful for an expressive control of a sound synthesizer. Formant frequencies are computed using the method described in section 5.2.1.2.

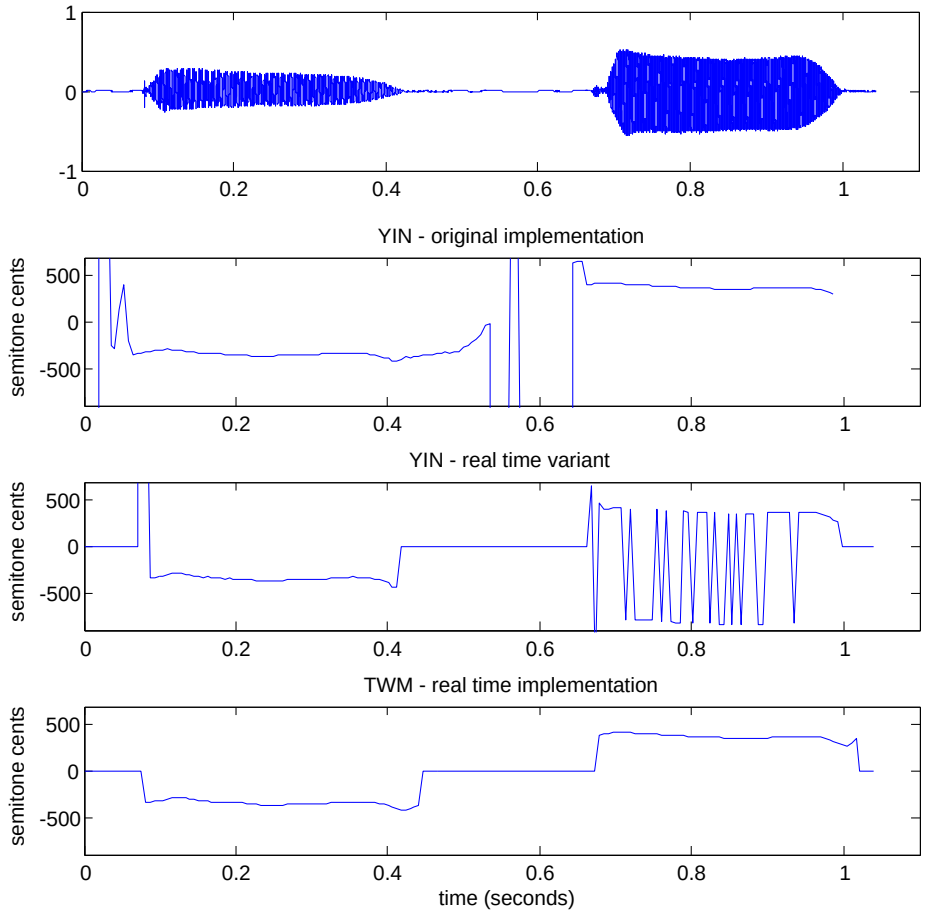

Figure 5.16: Pitch estimation of a female voice example with two consecutive notes expressed in semitone cents relative to A-440. The real-time implementation of the YIN algorithm presents octave jumps. Unvoiced pitch values are represented as 0 for clarity reasons.

5.3.2.3 Breathiness

A timbre attribute that can be modulated is the degree of phonation, referred throughout this dissertation as *breathiness*. This descriptor aim to measure the amount of air turbulence in the voice signal. We consider it as a modulation gesture and we will take directly the output of the breathiness algorithm presented in section 5.2.3.4.

5.3.3 Selection gestures

While in some instruments, *selection gestures* are clearly identified (e.g. the played string on a violin, register selection in an organ, etc.), instrumental gesture categories are much more interrelated for singing. However, in our context, we argue that the phonetic choice constitute a *selection gesture*.

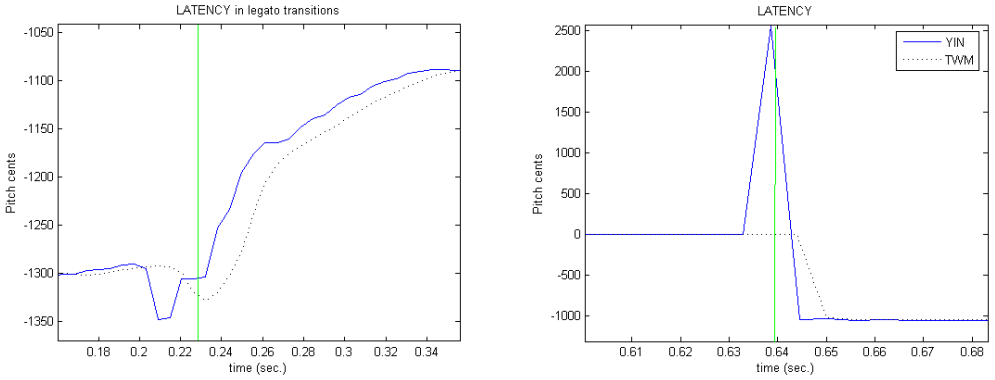

Figure 5.17: Two examples of fundamental frequency estimation for a voiced note onsets. Estimated values of f_0 stabilize after a number of frames. On the left a note-to-note transition. On the right, a silence to note transition for clarity reasons, pitch takes a value of 0 during silence for the middle and bottom plots.

Sound is determined by the position of the articulators. In our opinion, those are placed in a discrete manner (unconsciously) from a previous "mental" selection of the phoneme. One might find contradictions in this argument, though. For instance, vowels can be seen as points in a continuous space of two dimensions, first and second formants (see 2.3.1). And in fact, we included formants as *modulation gestures*. To remain consistent, the phonetic selection is not tackled at a phoneme level but at a broad phonetic class level, as introduced in table 5.7.

5.3.3.1 Phonetic classification

In section 5.2.2, we introduced a method for segmenting syllabling signals based on phonetics. Each segment is then labeled with a phonetic class in a classification based on its musical function. We include here the phonetic class as a *selection gesture*. As we describe in chapter 6, the phonetic class is used for mapping vocal gestures to the synthesizer controls. For instance, note onsets are determined if a gesture is labeled with the *sustain* phonetic class.

5.3.4 Coding vocal gestures

From the representation of vocal gestures, the next step is the coding in a specific format, allowing at the same time, the integration of the present research in other systems. Nowadays, a number of research labs in Music Technology study instrumental gestures with different purposes; from the control of sound synthesis or sound effects, to the analysis of performer gestures in traditional or digital instruments. Most of these research labs have joined and organized several workshops under the umbrella of the CONGAS European project[9].

[9]CONGAS (Gesture CONtrolled Audio Systems) is an EU COST action (2003-2007). `http://www.cost287.org`

As already expressed, in gesture-based audio systems, gestures are captured by some kind of sensor, and transmitted to a computer that processes this data either in real-time or in an off-line manner. In order to be able to share gesture data among labs, researchers at the University of Oslo (Jensenius et al., 2006; Jensenius, 2007) started an initiative to specify a standard format for gesture description. This format, still under development, is called GDIF (Gesture Description Interchange Format), and takes its name from SDIF (Sound Description Interchange Format). GDIF aims to accelerate/facilitate development of gesture-based systems, including the re-usability of measurement data and the exchange data among partners in collaborative projects.

The current GDIF implementation uses the Open Sound Control (OSC) protocol. In addition to acquisition data, the intent in GDIF is really to define how to describe mid- and higher-level features. A preliminary GDIF specification used for gesture control of spatialisation can be found in (Marshall et al., 2006). Further work on GDIF include a case study of the requirements needed to code *instrumental gestures* (i.e. sound-producing and sound-modifying gestures) in violin performance (Jensenius et al., 2007). We envisage two different types of usage when coding vocal gestures in the GDIF format:

- **Streaming:** Using GDIF in real-time applications (e.g. as an interface for a musical controller). Vocal gestures are coded as in the following examples:

```
/gdif/instrumental/excitation/loudness x
/gdif/instrumental/modulation/pitch x
/gdif/instrumental/modulation/formants x1 x2
/gdif/instrumental/modulation/breathiness x
/gdif/instrumental/selection/phoneticclass x
```

- **Storing:** Using GDIF for off-line post-processing (e.g. annotation, visualization, etc.), and as a sharing format for gestural data. For storing GDIF data, the SDIF format has been the proposed, since it is a binary file format that would allow an easy synchronization with sound.

5.4 Summary

This chapter has presented a number of voice analysis methods. From the findings of chapter 3, we observed the need of algorithms for describing syllabling signals. We introduced methods that describe three aspects of syllabling signals: formant tracking, phonetic segmentation and breathiness characterization. Next, we sum up the results achieved.

- Formant tracking algorithm: real-time method based on the discrete cepstrum and dynamics programming. It provides robust formant estimation with the objective to be used as a timbre control for voice-driven synthesis.

- Syllabling segmentation algorithm: it aims to exploit the phonetic characteristics of syllabling signals. It is based on the first derivative of timbre features (MFCC) and a probabilistic transition model.
- Breathiness characterization algorithm: it is an example of the timbre possibilities of the voice compared to other instruments. Three compared methods describe the type of phonation with a continuous value.
- Vocal Gestures is the term chosen to describe syllabling signals in our context. We represent vocal gestures in the same manner that instrumental gestures, as suggested by Cadoz (1988).
- GDIF is finally used as a format based on OpenSoundControl (OSC) to code and transmit the syllabling description, allowing a further use by other researchers.

Chapter 6

Controlling an Instrumental Sound Synthesizer

After addressing the control of a singing voice synthesizer in chapter 4, this chapter proposes control strategies for synthesizing other musical instruments. Our approach devises a set of mapping functions using the vocal gestures analysis of chapter 5, and puts examples of mappings for three families of instruments.

6.1 Introduction

Our goal is to convert acoustic features extracted from the voice signal to parameters of an instrumental sound synthesizer. Here, two concepts are closely related. On the one hand, we have to deal with vocal imitations of traditional instruments. We have addressed this topic in chapter 3. On the other hand, we have to cope with the typical problems of synthesizers control in instrument imitation, and its cognitive constraints. As Pressing (1992) explains, if one aims to play a plucked string synthesizer with a wind controller, one needs some "mental readjustment". In that sense, singing-driven interfaces will suffer the same limitation. We argue that, for the case of instrumental sound synthesis, cues of this readjustment will be found in the characteristics of the vocal imitation of instruments.

Also, related to the topic of gesture-based synthesis, for instance as described in (Perez et al., 2007), a future objective is to be able to define the mappings at a gesture level, linking the vocal gestures analysis (described in chapter 5) to the performer gestures on different instruments. For instance assigning voice loudness to the bow velocity control of a gesture-based violin synthesizer[1]. Instead, the presented results refer to mappings done using musical *intermediate parameters*: pitch,

[1]Note that *loudness* and *bow velocity* are excitation gestures in voice and violin respectively.

dynamics, timbre and articulation. This is actually closely related to the multilayer mapping strategies proposed by Arfib et al. (2002)[2]. As shown in figure 6.1, *performance analysis* corresponds to "gesture to gesture scaling", *mapping* corresponds the "gesture to abstract parameters" layer, and *score creation* corresponds to "abstract to synthesis parameters" layer. After the performance analysis, vocal gestures are first converted to intermediate parameters by means of different mapping functions (section 6.3). In a second step, the score creation module (section 6.4) uses these intermediate parameters to select the most appropriate sample from the database and to derive the required transformations.

In the first part, we investigate mapping strategies from two different perspectives: phonetic-based, and instrument-based. The former describes research study toward a user-adapted mappings. It aims at establishing a link between phonetics by users and musical articulation. The latter is derived from musical acoustics and musical performance. It gives hints on how continuous voice features such as pitch or dynamics should be mapped according to the characteristics of the instrument sound.

In the second part, the outputs of the mapping functions are applied to the creation of an internal score for a synthesizer based on sample-concatenation. The score contains frame by frame information of the selected sample and the transformation parameters. Finally, in section 6.5 we introduce a prototype that incorporates the presented research results.

As we previously indicated, the factors involved in the design of singing-driven interfaces for sound synthesizers, are: performer skills, imitated instrument, synthesis technique and time mode (real-time and off-line). The implemented system assumes that the performer has enough musical skills at least to sing in tune and to control the expression. We choose a sample-based synthesis technique with spectral concatenation. Regarding the synthesized instrument, we put examples of three instruments that are representative of different instrument families: plucked string, bowed string and blown instruments.

6.2 Performance analysis

From the analysis of voice signals in the context of instrument imitation, we have observed (see chapter 3) that they can be described as *syllabling*. Also, we proposed to represent syllabling signals as *vocal gestures*, in the context of instrumental gestures (a subset of musical gestures). Instrumental gestures is a term widely used in the literature to describe performer actions. In order to describe the input voice, we use the output of the vocal gestures analysis, as presented in the previous chapter. This brief section only recalls the vocal gestures representation, which consists of:

- Excitation:

[2] Arfib et al. (2002) proposes a three layer mapping consisting of a first "gesture to gesture scaling", a second "gesture to abstract parameters" layer and a third "abstract to synthesis parameters" layer. In our approach, we use the term "intermediate" instead of "abstract".

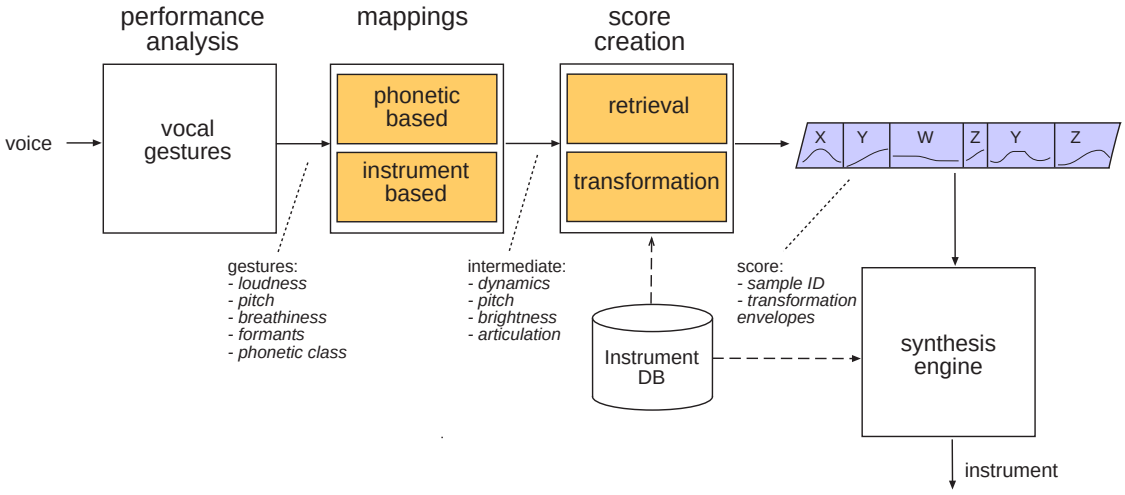

Figure 6.1: Diagram of the voice to instrument synthesis framework. *Performance Analysis*, as described in section 6.2, refers to vocal gesture analysis. *Mappings* is achieved with two functions that convert vocal gestures to intermediate parameters, phonetic-based and instrument based (section 6.3). Finally the *score creation* module (section 6.4) uses the intermediate parameters to find the most appropriate sample $(X, Y, Z, \ldots)$ from the database and create the transformation envelopes used in the synthesis engine.

  - Loudness: it is a continuous feature related to the air-flow pressure and is derived from the signal's energy.

- Modulation:
  - Pitch: it is a continuous feature that is derived from the signal's fundamental frequency.
  - Breathiness: it is a continuous feature that is related to the type of phonation and is derived from the amount of noise in an voiced signal.
  - Formants: it consists of a continuous pair of values related to the vocal tract position, the pair of values are the first and second formant frequencies.

- Selection:
  - Phonetic class: it is a discrete value related to the phonetics and the musical function. Changes in the phonetic class are later used to detect onsets in the voice signal.

The performance analysis module receives a voice signal and outputs a sequence of vocal gestures, as extensively described in chapter 5. This process can be considered as a first mapping layer. The following sections (section 6.3 and 6.4) address one layer for converting vocal gesture to intermediate parameters, and one layer for converting intermediate parameters into the synthesizer parameters, in our case an internal score.

6.3 Studies on mapping

Proposed mappings receive vocal gestures and output *intermediate parameters* that will feed the score creation module for the concatenative synthesizer. As a first step toward gesture-based mappings, we opt to define mapping functions from vocal gestures to perceptual musical parameters, here referred as *abstract*. Future research should strive to allow direct mappings to gesture controls of a gesture-based synthesizer.

Two studies are presented (sections 6.3.1 and 6.3.2), which handle the role of user phonetics and the imitated instrument. First, in the section entitled *Phonetic-specific mappings*, we describe an experiment whose objective was to learn the phonetic employed by several subjects for imitating note-to-note articulations of different instruments. This experiment attempted to learn the performer dependency when using singing-driven interfaces. Next, under the name *Instrument-specific mappings*, we present a study of instrument dependency from a theoretical standpoint, proposing and justifying a set of mapping functions.

6.3.1 Phonetic-specific mappings

Phonetic-specific mappings control the type of note-to-note articulation of a sample-concatenation synthesizer. For the type of articulation we might think of a continuous value, as described in (Maestre and Gómez, 2005), ranging from staccato to legato. The underlying idea is to derive the type of articulation from the phonetics in the voice signal.

In chapter 3, we observed that one can find some constants in vocal instrument imitation. Fundamentally, the phonetics in instrument imitation has relation to the musical and acoustic properties of the original instrument performance. Yet a more flexible alternative would be to use *user-adapted* mappings, where the system, given a few examples by the user, adapts automatically to the subject's own preference in the choice of syllable.

This section reports of an experiment that given a vocal instrument imitation signal, classifies the type of articulation as staccato or legato. We compare two approaches: a user-dependent approach based on supervised-learning and a user-independent approach based on heuristic-rules. The first approach aims to automatically learn the behaviour for a particular user using Machine Learning (ML) techniques. The input data for the classifier consists of the phonetic class (see section 5.3.3) from the vocal gesture representation, plus the first 5 Mel-Frequency Coefficients (MFCC) as low-level timbre features. The second approach uses simple rules to classify between staccato and legato articulations. The input data consists only on the phonetic class of the vocal gestures representation.

In our experiment, subjects were requested to imitate real instrument performance recordings, consisting of a set of short musical phrases played by saxophone and violin professional performers. We asked the musicians to perform each musical phrase using different types of articulation. From each recorded imitation, our *imitation segmentation module* automatically segments note-to-note

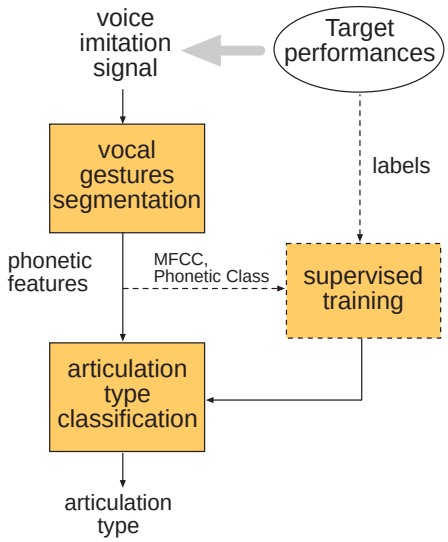

Figure 6.2: Overview of the proposed system. After the imitation segmentation, a classifier is trained with phonetic low-level features feature and the articulation type label of target performance.

transitions. After that, a set of low-level descriptors, mainly based on cepstral analysis, is extracted from the audio excerpt corresponding to the segmented note-to-note transition. Then, we perform supervised training of the *articulation type classification module* by means of machine learning techniques, feeding the classifier with different sets of low-level phonetic descriptors, and the target labels corresponding to the imitated musical phrase (see figure 6.2). Results of the supervised learning are compared to classifier of articulation type based on heuristic rules.

6.3.1.1 Classification of musical articulations

The mapping task consist here in associating phonetics to different type of musical articulations in note to note transitions. For the voice imitation recordings, we asked four volunteers with diverse singing experience to listen carefully to target performances and to imitate those by mimicking musical articulations. Target performances are saxophone and violin recordings, in which performers were asked to play short phrases in three levels of articulation. There are 24 variations covering:

- *articulation* (3): legato, medium and staccato.
- *instrument* (2): sax and violin.
- *note interval* (2): low (2 semitones) and high (7 semitones).
- *tempo* (2): slow and fast.

All target performance recordings were normalized to an average RMS level, in order to remove the influence of dynamics and to concentrate on articulation aspects. Subjects were requested to naturally imitate all 24 variations with no prior information about the experiment goals. Variations were sorted randomly in order to avoid any strategy by subjects. The process was repeated twice, gathering 48 recordings per subject.

In table 6.1, we can observe the results of user-dependent supervised training for the four subjects, using two (*staccato* and *legato*) and three (*staccato*, *normal* and *legato*) classes for articulation type is presented. The classification algorithm used in our experiments was the J48, which is included in the WEKA data mining software [3]. Due to the small size of our training set, we chose this decision-tree algorithm because of its interesting properties. Namely, because of its simplicity, this algorithm is more robust to over-fitting than other more complex classifiers. The attributes for the training task include phonetic features of note-to-note transitions. Three combinations of phonetic features within a transition were tested: 1) MFCC(1-5) of the middle frame; 2) MFCC(1-5) of the left and right frames; and 3) difference between the MFFCs of the left and the middle frames, and between the MFCCs of the right and the middle frames.

In addition, we present the results of a user-independent classifier (2 classes) based on heuristic rules in table 6.1. The rules derive from the phonetic class information from the *vocal gestures segmentation* module. When a new *sustain* segment is preceded by a *transition* segment, then the articulation type is classified as legato, else it is classified as staccato. Table 6.2 shows the mean percentage of correctly classified instances using different phonetic features as input attributes, and in the last row the results using heuristic rules.

6.3.1.2 Discussion

In a qualitative analysis of the imitation recordings, we observed that phonetics are at a phoneme level indeed user-dependent. However, some constants are found at a phonetic class level, similar to the results of the web survey presented (see section 3.3.3). Not all subjects were consistent when linking phonetics to articulation type on different target performances. Moreover, none of the subjects were able to distinguish three but only two types of articulation in the target performances. Also, we are aware that the number of subjects taken for this experiment was insufficient to extract some general conclusions. However, the results provided hints about how to handle articulation in vocal imitation for singing-driven interfaces.

From the quantitative classification results, we can extract some conclusions. Similar results were obtained classifying in two and three classes, when compared to the baseline. When looking at the dependency on the imitated instrument, better performance is achieved by training a model for each instrument separately. It indicates some correspondence between imitated instrument and phonetics. Concerning the phonetic features used as input attributes for the classifier, results are very similar

[3] `http://www.cs.waikato.ac.nz/~ml/weka/`

Supervised training: 3 classes baseline = 33%	
description	*correct* (%)
subject1- sax	57.727
subject1- violin	44.5455
subject1- sax-violin	51.5909
subject2- sax	67.281
subject2- violin	67.2811
subject2- sax-violin	51.2415
subject3- sax	41.7391
subject3- violin	48.7365
subject3- sax-violin	40.2367
subject4- sax	41.7722
subject4- violin	42.916
subject4- sax-violin	38.3648

Supervised training: 2 classes baseline = 66%	
description	*correct* (%)
subject1- sax	83.1818
subject1- violin	71.3636
subject1- sax-violin	78.6364
subject2- sax	93.5484
subject2- violin	67.699
subject2- sax-violin	80.5869
subject3- sax	70.4348
subject3- violin	72.2022
subject3- sax-violin	69.0335
subject4- sax	64.557
subject4- violin	73.3333
subject4- sax-violin	66.6667

Heuristic rules: 2 classes baseline = 66%	
subject1- sax-violin	82.2727
subject2- sax-violin	79.684
subject3- sax-violin	76.3314
subject4- sax-violin	78.1971

Table 6.1: Results of the supervised training with 3 classes(staccato, normal and legato) and 2 classes (staccato and legato) using ten-fold cross-validation. MFCC (first five coefficients) are taken as input attributes. Results of a classifier based on heuristic rules with 2 classes(staccato and legato).

attributes	*sax*	*violin*	*sax+violin*
1	77.930	71.698	73.730
2	80.735	72.145	74.747
3	81.067	72.432	75.742
4	–	–	79.121

Table 6.2: Mean percentage for all subjects of correctly classified instances using: *1)*MFCC (central frame); *2)*MFCC+LR (added left and right frames of the transition); *3)*MFCC+LR+DLDR (added difference from left to central, and right to central frames); *4)* Heuristic rules.

(see table 6.2). The heuristic-rule classifier uses the output of the imitation segmentation module. When a new note onset occurs (a vocal gesture is labeled with the *sustain* phonetic class), the rule looks at the phonetic class of the preceding gesture[4]. If the phonetic class of the previous gesture is a transition, the the articulation type is legato, else the articulation type is staccato. This simple rule performed with an accuracy of 79.121%, combining sax and violin instances in the test set.

Comparing the overall results of the user-dependent supervised training, we can conclude that there is no significant improvement over the user-independent classifier based on heuristic rules, which also has a more direct implementation. Hence, in our final implementation the type of articulation derives from an heuristic rule that only looks at the phonetic class of the vocal gestures segmentation.

6.3.1.3 Mappings to intermediate parameters

From the vocal gestures segmentation, the *phonetic-specific mappings* generate one parameter to describe articulation, which is included in the category of *intermediate parameters* (see figure 6.1). The *articulation* parameter is a continuous value in a range $[0 \cdots 1]$, that corresponds to the grade of legato (0 for an impulsive staccato articulation, and 1 for a smooth legato articulation). Within this range, we distinguish two types of note to note articulations whether the note is preceded by a rest ($x < 0.5$) or not ($x \geq 0.5$). Moreover, from the vocal gestures segmentation we compute the continuous value, which is derived from the phonetic class and the loudness of the preceding gesture. For example, in the the utterance /t a n a/, two onsets are detected with the following gestures segmentation $Sil, Att, Sus, Tra, Sus, Sil$. The first articulation will have an articulation value near to 0, it comes from a rest, and is is impulsive since it is preceded by an *attack* segment. The second onset, will be classified as legato, and the articulation value will range from 0.5 to 1.0, depending on the loudness of the *transition* segment.

In our framework, the articulation will influence the sample selection process. In other words, in a sample concatenation synthesizer, we will seek *samples* (i.e. the musical unit to be concatenated) that belong to a specific musical context. *Samples* are notes, and the musical context refers to the articulation to the preceding and following notes[5]. In offline mode, with a good sample selection strategy, we can improve the realism of the synthesized sound, as in the following audio examples: staccato voice input (Audio 6.1) , staccato violin synthesis (Audio 6.2) , legato voice input (Audio 6.3) and legato violin synthesis (Audio 6.4) .

[4]The phonetic class of a vocal gesture is detailed in section 5.2.2 and listed in table 5.7. Based on its musical functions, vocal gestures are labeled as attack, sustain, transition, release, special and silence.

[5]However, current trends in sample concatenation synthesis aim to consider as *samples* as long as possible encompassing complete musical phrases in order to keep the expression of the original recording.

6.3.2 Instrument-specific mappings

The second study on mappings looks at the role of the synthesized instrument sound. The instrument acoustic properties as well as its control constrains, will help us in defining meaningful control strategies.

Playing an instrument requires transforming musical ideas to physical actions. In section 3.3, we have observed how the term *instrumental gesture* has been addressed in the literature to express those actions by a performer. One can clearly see that each instrument has a number of control variables that a performer utilizes to produce sound. Actually, as suggested by Bonada and Serra (2007) in a framework for the singing voice, one can talk of *sonic spaces*, and more precisely of a sonic space of one instrument/performer combination. Thus, the performer navigates in this space, varying the characteristics of the produced sound. This space is multidimensional and will be bigger for a skilled performer than for a novice. We opted to deal only with the dimensions of this sonic space that are the most perceptually relevant (pitch, dynamics, color and articulation). Other expressive resources such as vibrato and how they are performed in different instruments are thus not considered[6]. This practical constraint let us to focus on the control possibilities of the voice, which is the main contribution of this dissertation. In other contexts such as the development of high-quality instrument synthesizers, these dimensions of this space merit further exploration.

6.3.2.1 Characteristics of different instrument families

In the area of Musical Acoustics, instruments can be classified according to its excitation mechanism, either continuous or discrete (Fletcher and Rossing, 1998). A further classification might include pitch contour behavior in note-to-note articulations, as summarized in table 6.3. Note that one could extend it with an additional classification depending on the capability of producing polyphony. As we already mentioned, this research deals only with monophonic synthesis[7]. Focusing on monophonic instruments, and for the case of synthesizers based on sample concatenation, we propose a number of mapping functions.

Instrument-specific mappings consist in defining a set of functions that, given one voice descriptor (e.g. pitch) and the synthesized instrument (e.g. violin), appropriately generates *intermediate parameters*. Specifically, we define five variables (pitch, dynamics, timbre, articulation, and timing), and a set of functions for these variables. Instrument-specific mappings aim to establish the type of function used for each variable and the different instrument categories listed in the table 6.3. The next sections describe in more detail the mappings for three instrument families: plucked string, bowed string and blown instruments.

[6] Actually, some expressive resources, both for analysis and synthesis, are more appropriate for off-line synthesis. For instance, detecting the presence of vibrato implies a latency of hundreds of milliseconds. In an off-line sample-based synthesis, one can select a sample containing a real vibrato recording to improve the realism. The same can be applied to note to note articulations such as portamento.

[7] Here, we might think on controlling piano chords or guitar strumming with the voice, using pitch to select the note and timbre features such as vowel to select the chord: major, minor, 7th., etc.

Classification	*Category*
Excitation	continuous excitation discrete excitation
Pitch contour	continuous pitch discrete pitch

Table 6.3: Instrument classification according to musical acoustics.

Category	*Examples*
A) Cont. excited and cont. pitch	singing voice, violin, viola, etc.
B) Cont. excited and discr. pitch	clarinet, sax, flute, etc.
C) Discr. excited and cont. pitch	double bass, guitar, etc.
D) Discr. excited and discr. pitch	piano, vibraphone, harp, etc.

Table 6.4: Examples of instrument for different categories combining continuous/discrete excitation and continuous/discrete pitch control .

Before specifying mapping strategies, we need to define the sonic space of the instrument/performer combination. In order to identify the axes of this multidimensional space, we interviewed some experienced performers of bass guitar, violin and saxophone. We asked them to identify the control parameters they use to produce and modify the instrument sound. Next, we look at the characteristics of three instrument families, focusing at the same time on vocal control issues.

Plucked string instruments This category cover a wide range of musical instruments, which despite their variety in timbre, can be characterized as decaying sounds with short attacks. Most of the plucked string instruments are polyphonic, and both, the repertoire and typical playing styles make extensive use of this property (e.g. guitar strumming). In our context, we only consider plucked strings as a monophonic instrument, so many of the reproduction of such effects are beyond this work.

Concerning the control, the following aspects for the synthesis parameters should be contemplated. *Dynamics*: a key-velocity control is appropriate, while using a breath controller would require a mental readjustment. The same occurs in our case, for singing-driven interfaces, where the user often imitates the plucking action with a syllable starting with a plosive. *Vibrato*: in the traditional instrument with frets, the string is bent away from its normal position. With fretless instruments, the left-hand controls the vibrato as in the bowed strings. *Timbre*: some timbre effects such as the playing position (near the bridge or the fingerboard) can be exploited by using some voice features such as vowels. *Articulation*: for muting/damping the string, the imitation will create a rapid decay to zero. This is easily implemented in our synthesis engine.

Table 6.5 summarizes the performer/instrument dimensions for the case of an electric bass guitar. Although, the guitar is probably the most popular plucked-string instrument, we found the vocal control of bass guitar more appropriate for a number of reasons. This is a plucked-string instrument

Abstract parameters	*Gestures*	*Range*
Dynamics	plucking velocity	*pp-ff*
Pitch	string and fingerboard position	E1-G3
Color	plucking position	bridge-neck
Attack	plucking finger action	soft-normal-slap

Table 6.5: Proposed control dimensions of the performer/instrument sonic space for bass guitar

that is mostly played as monophonic. Also for its potential use in real performances, a non-soloist instrument can be equally interesting, in the sense that traditional instrument performers such as guitarists or pianists can create voice-driven bass lines as accompaniment.

In a plucked strings imitation, the dynamics parameter is taken from the voice loudness during the attack and kept constant for the whole note, so that the sample-based synthesized sound will reproduce exactly the plucked string decay. In contrast, the pitch parameter tracks the smooth variations of the input voice pitch in order to simulate pitch-bend. Even, synthesizing a fretted bass guitar, control over the pitch bend results in a more natural sound. Additionally, using spectral transformations, we can synthesize "unrealistic" sounds by modifying the amplitude envelope of the plucked string decay, e.g. making an infinite sustain note by looping a short fragment using spectral concatenation synthesis technique.

Bowed string instruments The sound produced in bowed string instruments is generated by the vibrations of the strings, but opposite to plucked strings, here strings are excited by a continuous applied force. Due to the friction forces between the exciter (bow) and the resonant element (string), the latter starts vibrating.

By looking at the control characteristics, the following considerations for the synthesis parameters should be contemplated. *Dynamics*: here, bow velocity is the input parameter that most influences the dynamics. Typically brightness increase with volume. *Articulation*: an extensive sampling is required for reproduce changes of bow direction, etc. *Vibrato*: opposite to blown instruments, here the vibrato affects only frequency modulation. *Pitch micro-variations*: they are almost not present, therefore voice pitch has to be flattened to eliminate the jitter. *Timbre*: in violin, timbre control is multidimensional (bow force, bow angle, etc.). However, some basic timbre effects such as the playing position *sul ponticello*, or *sul tasto* can be exploited by using voice formants.

Bowed string instruments fall in the category of continuous excitation instruments. Moreover, the pitch control is also continuous. Hence, pitch and dynamics parameters can be directly related to those in the singing voice. As expressed previously, micro-variations will be already reproduced by the synthesizer. Therefore, the mapping functions for pitch and dynamics have to filter out the micro-variations of pitch and dynamics.

Abstract parameters	*Gestures*	*Range*
Dynamics	bow-bridge distance, bow velocity, bow force	*pp-ff*
Pitch	string and fingerboard position	G3 to D7
Color	bow-bridge distance, bow velocity, bow force	
Attack	acceleration and force	

Table 6.6: Proposed control dimensions of the performer/instrument sonic space for violin.

Abstract parameters	*Gestures*	*Range*
Dynamics	air pressure	*pp-ff*
Pitch	fingering and vocal cavity volume	A2-E5 (tenor sax)
Color	fingering, (1)lip position and (2)glottis	(1) normal, subtone; (2) growling
Attack	tonguing	phonetics (/t/, /d/)

Table 6.7: Proposed control dimensions of the performer/instrument sonic space for saxophone.

Blown instruments Blown instruments can differ either in the type of excitation (e.g. single or double-reed vibration, lips vibration, etc.) or in the note selection mechanism (holes, keys, etc.). Using the clarinet as a general case, one can consider that breathing pressure control the dynamics, while the pressed keys control the pitch in a discrete manner. In addition, pitch bend is achieved by modifying the lips pressure on the reed.

For controlling a synthesized blown instrument, the following characteristics can be taken into account. *Dynamics*: by default, brightness increase with volume. With a continuous controller such as breath controller or, as in our case, the voice, we overcome the control limitations of keyboard controllers that use only key-velocity. *Articulation*: in the traditional instrument, the attack includes a noise component that can be achieved by pronouncing the consonants /t/, /k/ or /d/ in the embouchure. Legato articulations are determined by the absence of such consonants. *Vibrato*: it can be different from a synthesis point of view or from a performing point of view. In the former, it affects only to frequency modulation, while in the latter in might contain also amplitude modulation, depending on the instrument. *Pitch micro-variations*: usually, it is unlikely to maintain the exact pitch over a sustained note. Also, effects such undershoot or overshoot may appear in the traditional instrument sound. *Use of breath*: pauses are necessary to allow air to be taken in.

Since we use voice as a control signal, many of the effects that we encounter in blown instrument are also in the singing voice (e.g. breathing pauses). Hence, a seamless control can be achieved. Specifically, the dynamics parameter results from filtering out the micro-variations in the loudness curve of the voice. However, the pitch parameter will require a more precise adaptation. To simulate the discrete notes, we implement a non-linear mapping function to quantize the voice pitch curve to fixed values at the exact pitches of the tempered scale. For note to note articulation, we can benefit from the phonetic class in the vocal gestures analysis.

6.3.2.2 Proposed mapping functions

Here we present a set of mapping functions that convert *vocal gestures* into *intermediate parameters*. A first general description of this functions will lead us, in a further step, to adjust these functions specifically for a given instrument, in what we call *instrument-specific mappings*. Intermediate parameters are later used to create the internal score, as shown already in figure 6.1. Let us consider the following general mapping functions for:

- Pitch: estimated voice signal pitch may present unwanted irregularities (e.g. jitter on blown instruments).
- Dynamics: in some instruments, it will be not realistic to synthesize amplitude variationss (e.g. tremolo on a plucked string instrument).
- Timbre: voice timbre is more flexible than in other instruments. We propose to map voice timbre, determined by the formant frequencies, to brightness as output abstract parameter.
- Time-scaling[8]: the sample playback process will use different strategies depending on the instrument (e.g. it is not realistic to loop a sustain of a plucked string instrument).

Pitch functions From the performance analysis module, we have access to the instantaneous estimated pitch of the voice signal. Due to the nature of the voice signal and possible errors introduced by the pitch estimation algorithm, we apply a post-processing consisting of three functions: *envelope*, *quantization* and *transposition*. The three function can be configured depending on the characteristics of the synthesized instrument. Figure 6.3 shows the proposed mapping functions and the configuration parameters.

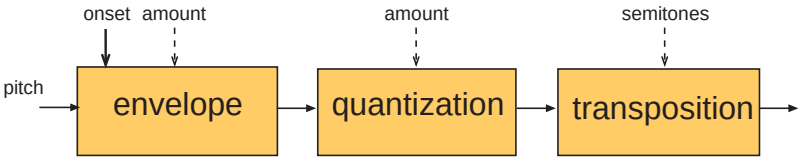

Figure 6.3: Mappings for pitch, consisting of *envelope*, *quantization* and *transposition* functions.

Envelope is a smoothing function ($y[n] = (1-a) \cdot y[n-1] + a \cdot x[n]$), in which the value of the factor a depends on a control parameter (amount), and a trigger signal (an onset detector derived from the vocal gestures segmentation). If *amount* is set to 0, this function behaves as a "sample and hold" operator, where $a = 1$ when an onset occurs and $a = 0$ for the rest. If *amount* is set to 1.0, it operates as a bypass ($a = 1$). For *amount* in a range between 0 and 1, the factor a takes a

[8]This mapping function is specific of sample-based synthesizers.

value of $1 - amount$ during an attack[9] time, and after that attack time changes smoothly to a value of $a = amount$. With this behaviour, we obtain a rapid response in note onsets allowing to recover from error in the pitch estimation. For a real-time operation, there is a trade-off in note onsets between the accuracy of the pitch estimation and the introduced latency. The *Envelope* function solves the problem only partially, since rapid variations of the mapped pitch might introduce artifacts in the synthesis. The *Quantization* function is also adjusted, obtaining either a stepped function at regular semitone intervals or a more smooth function, as shown in figure 6.4. An offset($[0 \ldots 100]$ semitone cents) can be added to the *Quantization* function, permitting a different tuning. Finally, *Transposition* adds an offset in the range -12 and $+12$ semitones.

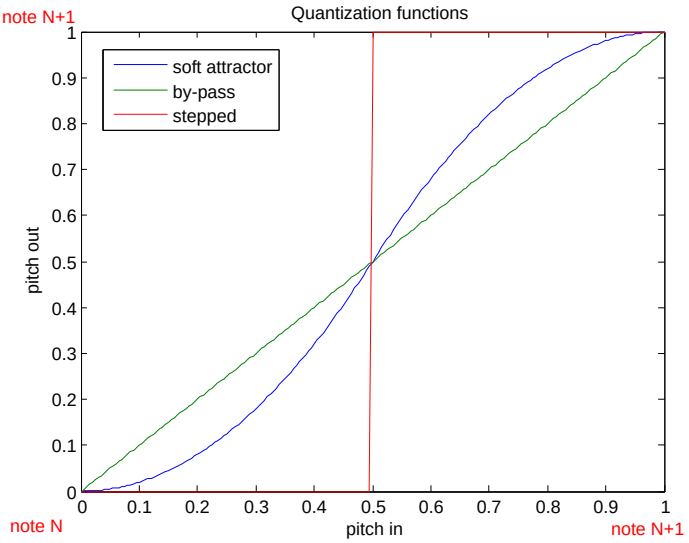

Figure 6.4: Pitch quantization function, showing different plots depending on the quantization amount. For amount is set to one, the quantization function is stepped; when amount is set to zero no quantization is applied (bypass).

Dynamics functions Similar to the pitch estimation, the performance analysis module provide information about the dynamics. The mapping functions are in fact equivalent, *compression* reduces the dynamic range of the input dynamics contour (see equation 6.1), and *gain* applies an offset, acting in a similar way as the *make-up gain* in traditional audio compressors. Figure 6.5 show the mapping functions.

$$dyn_{comp} = dyn^{(1-comp)} \tag{6.1}$$

[9]The attack time is fixed with typical values below 100 ms.

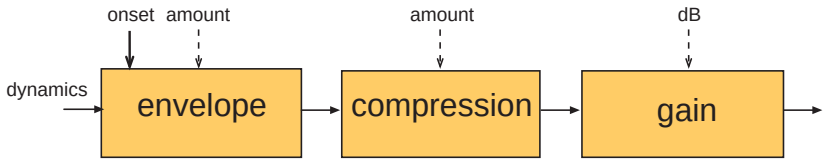

Figure 6.5: Mappings for dynamics, consisting of *envelope*, *compression* and *gain* functions.

Timbre functions In addition to pitch and dynamics, voice timbre constitute an interesting control for the synthesis. It can be handled in many different fashions, either from the voice features taken (e.g. spectral centroid, MFCC, formants, etc.) or from the derived mappings (e.g. brightness, noisiness, etc.). As a mapping example of voice timbre to *intermediate parameters*, formant frequencies (F1 and F2) are converted to brightness as shown in figure 6.6. Several strategies can be taken for this two-to-one mapping. We propose to derive *brightness* from the vowel sound in a scale from front to back in a range $[0 \cdots 1]$ (see equation 6.2). Taking the typical vowel sounds, this scale would be: /u/, /o/, /a/, /e/ and /i/. For the synthesis of saxophone sounds, we can control the amount of perceived amount of air, as in the following example: input voice (Audio 6.5) , without timbre control (Audio 6.6) and with timbre control (Audio 6.7) .

$$brightness = \frac{F2 - F1}{2000} \tag{6.2}$$

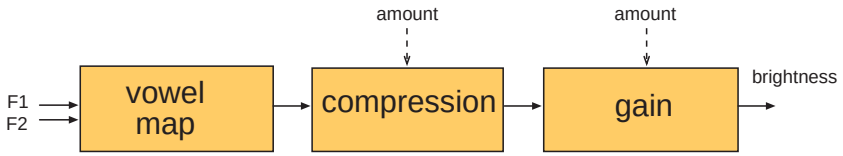

Figure 6.6: Mapping functions for timbre, where brightness is derived from the first two formant frequencies. *Compression* is used to reduce the timbre variations; and *gain* adds an offset.

Time-scaling functions Finally, the last mapping considered is time-scaling, which is very specific of sample-based synthesis. Depending on the acoustic characteristics of the sample different types of time-scaling can be applied to get a natural synthesis without artifacts. Actually, the appropriate type of time-scaling is related to the musical instrument family. For example, for discrete-excitation instruments such as a guitar, it has no physical sense to time-scale a sample since it will sound unnatural. For sustained instruments such as a flute, doubling the duration of a sample is absolutely meaningful. Table 6.8 lists the proposed types of time-scaling.

The strategies taken for the time-scaling will vary whether working in real-time or not. It is

Time-scaling type
Sustain-stretch
Sample-stretch
Sustain-frame
None

Table 6.8: Four types of time-scaling applied to in a sample-concatenation synthesizer.

assumed that the database contains annotations of the sustain part of the audio samples. Next equations use the following notation for the database sample: D_a^s (attack duration), D_s^s (sustain duration), D_r^s (release duration) and D_T^s (total duration); and the following for the input (target) durations: D_a^i (attack duration), D_s^i (sustain duration), D_r^i (release duration) and D_T^i (total duration).

- For the *Sustain-Stretch* type, the time scale factor (TS) is:

$$TS = \begin{cases} 1 & t < D_a^s \\ \frac{D_T^i - D_a^s - D_r^s}{D_s^s} & D_a^s < t < D_a^s + D_s^i \\ 1 & D_a^s + D_s^i < t < D_a^s + D_s^i + D_r^s \end{cases} \tag{6.3}$$

  In real-time, a looping strategy is necessary, which can also be achieved back and forth to avoid timbre discontinuities. In this case the sample selection process can be improved if the user sets the tempo (or is automatically learned), so that in real-time, lopping cases are minimized.

- For the *Sample-Stretch* type, the time scale factor (TS) is:

$$TS = \frac{D_T^i}{D_T^s} \quad t < D_T^i \tag{6.4}$$

  In real-time, the same looping strategy as in the Sustain-Stretch is applied.

- For the *Sustain-Frame* type, during the attack the original duration is kept and a single frame is repeated for the rest of the input duration.

$$TS = \begin{cases} 1 & t < D_s^s \\ 0 & D_s^s < t < D_T^i \end{cases} \tag{6.5}$$

  For real-time, the same time scaling function is used. This types produces a static sound in terms of timbre, resulting less realistic.

Category	*Abstract Par.*	*Mapping functions*
Plucked strings (bass guitar)	Dynamics	Env. = 0.00, Compr. = 0.50, Gain = 0
	Pitch	Env. = 0.15, Quant. = 0.00, Tran. = -12
	Time mode	None
Bowed strings (violin)	Dynamics	Env. = 0.75, Compr. = 0.00, Gain = 0
	Pitch	Env. = 0.50, Quant. = 0.00, Tran. = 0
	Time mode	Sustain-stretch
Blown (sax)	Dynamics	Env. = 0.75, Compr. = 0.25, Gain. = 0
	Pitch	Env. = 0.15, Quant. = 0.75, Tran. = +12
	Time mode	Sustain-stretch

Table 6.9: Description of the instrument-specific mapping functions for the intermediate parameters and the three instrument categories listed in table 6.4.

- For the *None* time-scaling type, the type of time-scaling where the duration of the original sample is kept the time scale factor is $TS = 1$.

6.3.2.3 Mappings to intermediate parameters

Finally, for each instrument category in table 6.3, we propose different configurations of the mapping functions for pitch, dynamics and time mode. From the three example instruments (bass guitar, saxophone and violin) as examples, from which we should be able to extrapolate for the three instrument families: plucked strings, blown and bowed strings.

In figure 6.7, we depict the pitch and dynamics intermediate parameters for different mapping configurations depending on the instrument. One can observe that the intermediate parameters envelopes eliminate the micro-variations present in the pitch and loudness analysis of the input voice (Audio 6.8) . Concerning the instrument sound characteristics, for the bass guitar, we see that the *dynamics* envelope is only updated at every onset, while it evolves continuously for the sax and violin. For the *pitch*, we observe that in the violin it is a continuous smoothed envelope, while it is quantized for sax and bass guitar.

6.4 Performance score

From the intermediate parameters, the synthesis control involves the creation of an internal score. In a further step, the synthesis engine reads the internal score, which contains information about the picked sample and the transformation parameters. The synthesis engine is then responsible for generating a continuous audio stream by concatenating different samples from the database and applying the necessary transformation based on spectral processing. Spectral processing transformations include transposition, spectral equalization and time-scaling.

An important constraint to take into account when creating the internal score is the time mode,

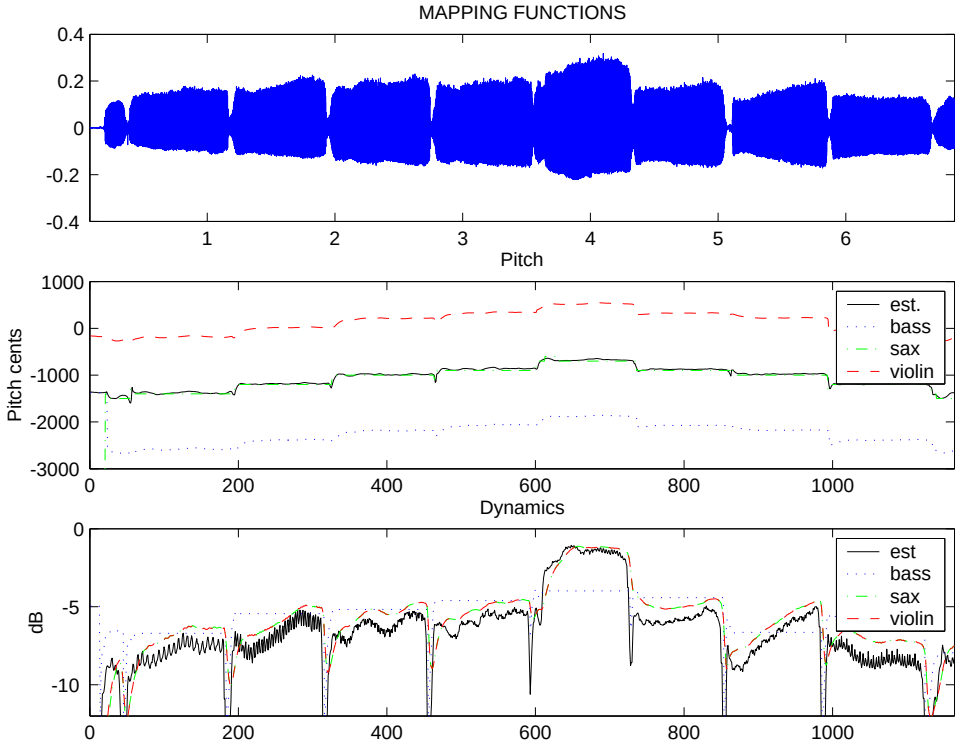

Figure 6.7: Example of the intermediate parameters output envelopes for pitch (middle) and dynamics (bottom). Note onsets are marked in the waveform (top). Mapping functions use different configuration depending on the instrument (bass, sax and violin).

i.e. real-time and off-line. As mentioned in several parts of this dissertation, real-time implies latency constraints and a primary goal is to achieve a fast responding system. On the contrary, in an off-line mode, the mapping functions can look over a longer time frame, considering *contextual* information (e.g. sequences of notes). This can be particularly useful in sample-based synthesizers whose database contains not only single notes but long musical phrases played with different expression (e.g. dynamics, tempo, etc.). The sample selection process can, thus, consider this contextual information in order to find the samples that require less transformations. This strategy is described for the case of the singing voice synthesis by Bonada and Serra (2007), and for the case of violin synthesis (Maestre et al., 2007). The present section addresses first the steps involved in building a sample database. Then, we describe the third mapping layer (*intermediate parameters* to *synthesizer parameters*), in our case, an internal score. Finally, we give a brief overview of the sample concatenation process.

6.4.1 Sample database creation

In order to search the appropriate sample in the database, the query targets have to be labeled with a meaningful description. In addition to the actual audio data, other analysis data is stored for each sample in the database. As graphically shown in figure 6.8, the database consists of three different types of files: audio (WAV), description (XML) and analysis (binary). Audio files contain performance recordings in PCM format, which might consist of musical phrases with several notes (Audio 6.9) , here referred as *samples*. Description files are XML files that contain the sample segmentation (begin and end times), sample descriptors (pitch, dynamics, articulation, context), and references to the pre-analysis binary files. In order to optimize the sample retrieval process, the system loads in memory a tree of samples, parsing all XML files. Figure 6.8 shows the database creation process.

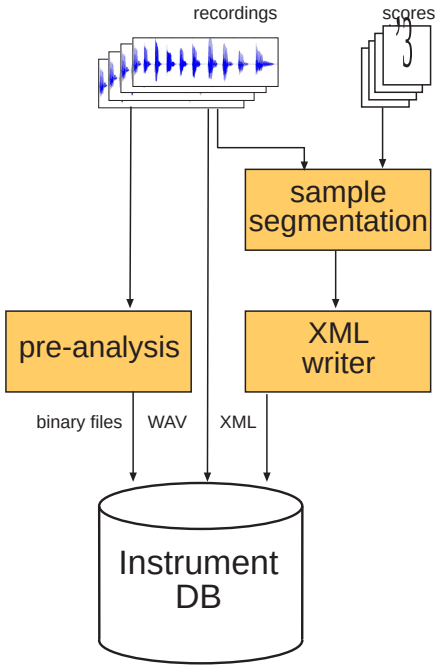

Figure 6.8: Block diagram of the database creation process. From the recordings and the scores of instrument performances, the database includes: audio in WAV format, pre-analyzed binary files with spectral information and XML files with a description of the sample.

As we see in the following example of an XML description file, the sample description makes references to a number of external binary files. These files includes pre-analyzed data in order to speed up the synthesis process. For all frames in the sample, it stores: *pitch* (the estimated pitch), *dynamics* (containing the frame energy), *peaks* (spectral peaks) and *mfpa* (maximally flat phase

alignment)[10]. In addition to the reference to the binary files, the XML file contains the nominal pitch for all samples contained in this phrase. Also, the sample segmentation times are provided, which are used by the synthesis engine to concatenate samples. Articulation is specified with the labels *from_sil* and *to_sil*, indicating whether the previous and the following notes are silences (rests) or not.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<concat_db_entry>
<external_references>
<audio_sources>
<source name="synthesis" file="2.wav">
<cdescriptor name="pitch" file="2_pitch.dat"/>
<cdescriptor name="peaks" file="2_peaks.dat"/>
<cdescriptor name="mfpa_env" file="2_mfpa.dat"/>
<cdescriptor name="dynamics" file="2_dynamics.dat"/>
</source>
</audio_sources>
</external_references>
<sample_segmentation>
<sample begin_mus="0" end_mus="1" est_reliable="true">
<note_context from_sil="true" to_sil="true" legato_left="0.5" legato_right="0.5"/>
<note midi_note="33" is_rest="no" />
<dynamics value="p"/>
<articulation type="staccato" />
<energy mean="0.5" min="0.1" max="0.8" begin_value="0.2" end_value="0.5"/>
<pitch mean="140" min="135" max="160" begin_value="137" end_value="158"/>
<spectral_centroid freq="350"/>
<attack seg_begin="0.1" seg_end="0.2610663" dyn_init="0.1" dyn_end="0.5"/>
<sustain seg_begin="0.2610663" seg_end="4.2591809" dyn_init="0.5" dyn_end="0.3"/>
<release seg_begin="4.2591809" seg_end="4.4160025" dyn_init="0.3" dyn_end="0.1"/>
</sample>
...
</sample_segmentation>
</concat_db_entry>
```

6.4.2 Sample retrieval and transformation parameters

As we have introduced in the beginning of this chapter (see figure 6.1), the intermediate parameters are fed to the score creation module. This module has two functionalities: first, to select the most appropriate sample from the database (retrieval); and second, to generate the transformation envelopes that are used by the synthesis engine to achieved the sample concatenation. Concerning the sample retrieval, it relies on calculating a cost function for all samples in the database given the input intermediate parameters. The sample having the lowest cost is then selected and appended to

[10]This technique was first developed for voice analysis/synthesis by Bonada and Loscos (2003) and is used for waveform preservation.

the score. The cost function is multidimensional, whose dimensions are derived from the intermediate parameters (pitch, dynamics, brightness, articulation and duration).

Intermediate Parameter	*Type*	*Range*
Loudness	envelope	[0..1]
Pitch	envelope	[−12000..12000]
Brightness	envelope	[0..1]
Articulation	float	[0..1]
Duration	float	*seconds*

Internal Score Parameter	*Type*
SampleID	integer
Gain	envelope
Transposition	envelope
Timbre mapping	envelope
Frame index	envelope

Table 6.10: List of input parameters and internal score parameters.

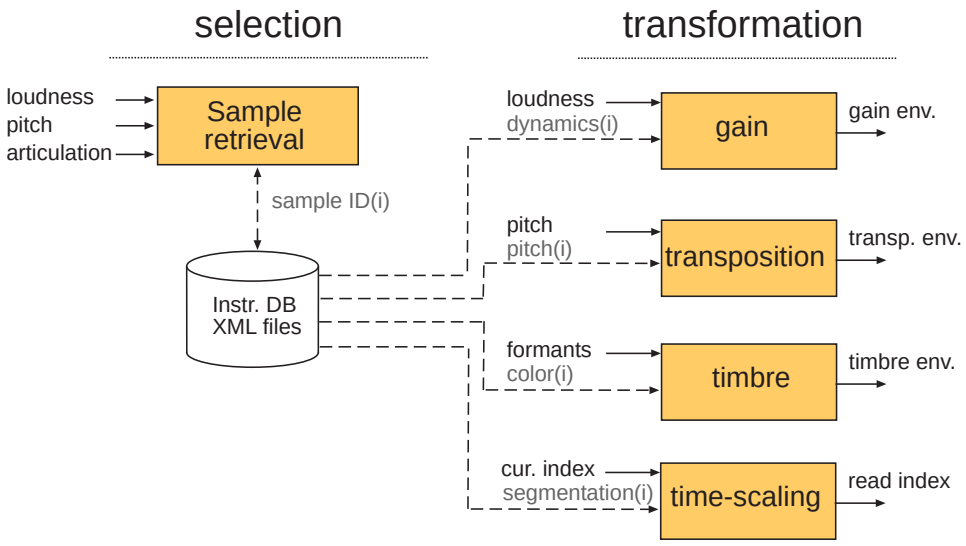

Figure 6.9: Diagram of the score creation process, divided in two main functions: sample selection, and transformation. On the left, the sample retrieval process takes nominal (one single value per parameter) values from the intermediate parameters. A cost function takes the nominal values to find the best sample from the database. On the right, the transformation functions use the descriptors and the sample i and the input envelopes of intermediate descriptors to compute the transformation envelopes for gain, transposition, timbre and time-scaling.

6.4.2.1 Sample retrieval

From the intermediate parameters, we have to find the sequence of samples from the database that best fits the voice input. This process is referred to here as *sample retrieval*. As previously indicated, the process of sample selection is different whether we work in real-time or in off-line mode. In real-time mode, intermediate parameters are updated frame by frame. At the very instant that a new onset occurs, we do not have information about the time elapsed until the next onset (it is a causal system). It implies that the duration of the sample in the database cannot be considered for finding

the best sample. An off-line operation permits us to take this into account, reducing the time-scaling applied to the sample.

For each new onset, a new sample is appended to the internal score. We compute a distance function between the input intermediate parameters and the sample descriptors. It is derived from a set of nominal values A_{nom} taken from the intermediate parameters (pitch and dynamics envelopes and the type of articulation). Nominal values in real-time take the instantaneous values for pitch and dynamics. In offline mode, they are computed as a weighted average ($A_{nom} = \sum_{k=N_i}^{N_{i+1}} w_k \cdot a[k]$). For pitch, w_k is the first derivative of the pitch. For dynamics, w_k is the dynamics value in order to favor high energy levels. Finally, the distance between input parameters and each database sample D_i is expressed as a weighted distance function $\rho(A, D; w)$, where Q is the number of parameters.

$$\rho(A, D; w) = \sum_{j}^{Q} |w_j(A_j - D_j)|$$

If we develop the previous equation for the case of real-time, the distance ρ_i for a sample i is:

$$\rho_i = w_{pitch} \cdot (A_{nom,pitch} - D_{i,pitch}) + w_{dyn} \cdot (A_{nom,dyn} - D_{i,dyn}) + w_{artic} \cdot (A_{nom,artic} - D_{i,artic})$$

Additionally, in off-line mode, the sample retrieval distance function can look at contextual information. In fact, introducing the sample duration in the distance function allows to reduce the time scaling applied.

6.4.2.2 Transformation parameters

Once a sample is selected from the database, the next step is to transform it according to the input intermediate parameters. We consider four different transformations: gain, transposition, timbre and time-scaling. As shown in figure 6.9, each transformation has a control envelope, determined from the abstract parameter and the nominal value of the sample. Samples are assumed to be constant in dynamics (absence of crescendo or decrescendo) and pitch (absence of glissando)[11] over the whole sample duration. Therefore, the transformation envelopes contained in the internal score will directly consist of the difference between the abstract parameter from the voice input and the nominal value of the sample. Transformation parameters are:

- Gain: it consists of a continuous envelope expressed in dB in a range $[-12..12dB]$.
- Transposition: it consists of a continuous envelope with the transposition factor (f_{out}/f_{in}).
- Timbre: it consists of a continuous envelope in the range $[-1..1]$ corresponding on the transformed spectral slope ($\pm 3dB/octave$).

[11] However, the presence of vibrato is correctly handled, since the perceived pitch is stable.

- Time scaling: it consists of a sequence of frame indexes of the read sample[12].

6.4.3 Sample concatenation

The synthesis is achieved by reading one sample from the database, frame by frame, and transforming it using spectral techniques for transposition, timbre equalization and time-scaling. In addition, the transition between two different samples is achieved using sample concatenation, as described in (Bonada and Loscos, 2003). Figure 6.10 shows the concatenation of two samples, where the sustain part is time-scaled. The main advantage of using a spectral-concatenation technique, is that it allows to "connect" two different samples applying pitch and timbre preservation. Therefore, the artifacts of the transition from one sample to the next is less noticeable.

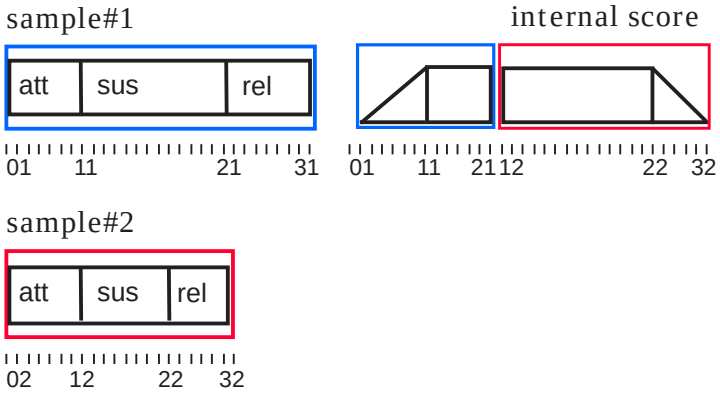

Figure 6.10: Example of the concatenation process. From two samples (#1 and #2), the internal score contains the frame indexes for each sample, allowing time-scaling and concatenation of the sustained parts of a sample.

As we have mentioned, it is not the aim of this dissertation to go into detail into the synthesis process. For extensive readings on spectral transformation and concatenation, we suggest to refer to (Laroche and Dolson, 1999; Bonada and Loscos, 2003; Bonada et al., 2006; Bonada and Serra, 2007).

6.5 Prototype

Different mapping strategies have been studied throughout this chapter, emphasizing the role of phonetics and the synthesized instrument sound. The actual process of controlling a sample-based concatenative synthesizer has been also described. In this section, we present a prototype that incorporates the proposed mapping functions. The prototype includes the creation of three instrument databases, one with bass guitar and two with saxophone sounds. The implementation of the

[12]This sequence of frame indexes is not necessarily neither linear nor continuous.

prototype and the instrument databases are used then to assess different aspects of voice-driven instrument synthesis. However, it is worth to point out that the achieved sound synthesis quality cannot be compared to state of the art synthesizers. The development of a high-quality sample-based concatenative synthesizer involves the creation of large databases, recording and annotating a variety of dynamics, articulations, playing modes, etc. This process is typically done manually and is beyond the scope of this dissertation.

6.5.1 Implementation

For the control of the instrumental sound synthesis, we have implemented a real-time plugin, called VDriven[13]. The architecture of the prototype is shown in figure 6.11. The VST plugin (VDriven) communicates to the analysis libraries that extract voice features. These features are then passed to the *internal score* module, which reads from the instrument database and applies the corresponding mappings. The synthesis engine reads the score and sends the output audio back to the VST.

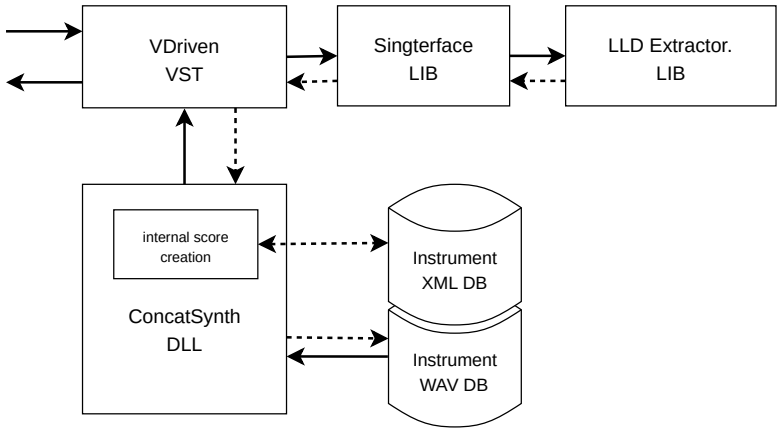

Figure 6.11: Software architecture of the developed prototype. Solid lines represent audio stream and dashed lines represent data stream.

6.5.1.1 Features

This prototype provides voice controlled synthesis of instrumental sounds in real-time. The system is expendable with new instrument databases. At the same time, it can be used as a singing-driven interface in other applications than synthesis, using only the vocal gestures analysis. The system should provide a low-entry fee, in the sense that it should be easy to produce sound. However, skilled singers will get the most of it thanks to their accurate control of voice. Figure 6.12 depicts the GUI of the *VDriven* prototype. Features are summarized in the following list:

[13]VDriven is a contraction of Voice-driven Synthesis.

- VST plugin implementation: programmed in C++, it uses the VST API[14], which is a flexible framework for audio that supports real-time and works under different host applications and operating systems.
- Instrument: user can choose different instrumental sounds. Currently, one bass guitar and two saxophone sound databases are included, but it can be extended with other instruments.
- Mappings: user can adjust the mappings to the synthesizer parameters according to his preferences and the instrument sound.
- Accompaniment: the prototype permits to play back background music, e.g drum loops.
- Configuration: the user can store and load different mapping configurations, including OSC destination address IP and port.
- Analysis mode: the prototype can operate only as an analysis tool, sending the vocal gestures description using OSC protocol.

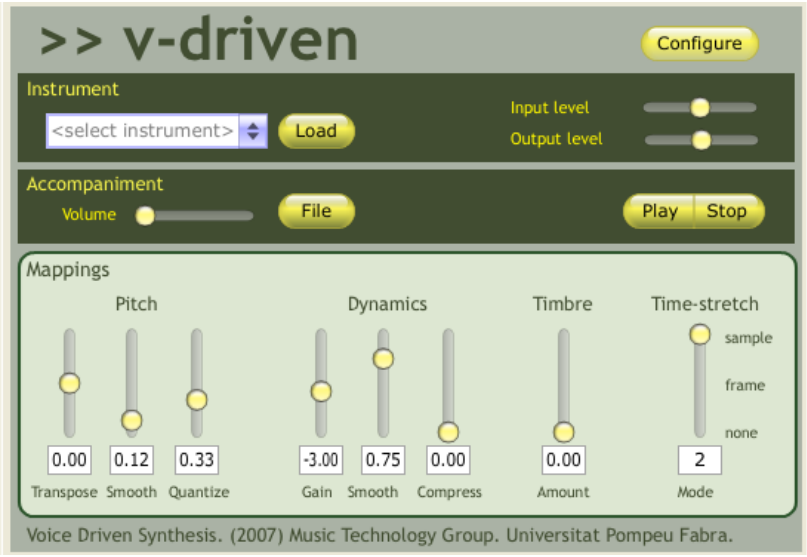

Figure 6.12: Screen-shot of the *VDriven* prototype.

The prototype works with blocks or *frames* of 256 samples at a sample rate of 44100 Hz. In terms of latency, due to the internal processing the prototype adds a latency of 29.02 ms to the latency of the audio interface (11.6 ms). The overall latency of the systems is therefore 40.6 ms. The internal processing needs half a window size (1024 samples) for the spectral analysis algorithms

[14]VST is trademark of Steinberg, `http://www.steinberg.de`

used for pitch estimation. In addition, the segmentation algorithm takes one additional block size to determine note onsets.

Concerning the synthesized sounds, we created three databases. One with bass guitar and two with saxophone sounds. The first one (bass guitar) contains three notes per octave; three dynamics: piano, mezzo and forte; and two articulation types: staccato and legato. This database does not cover different playing styles (e.g "slap") or timbre variations (e.g. varying the plucking position). The second database contains saxophone sounds. The samples belong to recordings of musical phrases (with an average of 16 notes per phrase), performed at different tempo. The dynamics in this database was constant, and most notes are played legato. The third database contains also saxophone sounds. In this case, the database contains single notes covering the pitch range at semitone intervals, and it contains different dynamics: piano, mezzo and forte.

All three databases are limited in terms of sampling the performer/instrument space. It affects the quality of the synthesized sound and the control possibilities, when compared to state-of-the-art synthesizers. Despite these considerations, we think that this prototype effectively demonstrates the possibility of controlling realistic instrumental sounds with the voice in real-time. Further work on the synthesis engine and on the databases creation shall improve the synthesis quality.

6.5.1.2 Adding new instruments

Our intention is to build a extensible system, able to incorporate new instrument databases. Using a synthesizer based on sample-concatenation eases this process. Basically, it suffices in creating a new database and defining the mapping functions, as explained in section 6.3.2. Theoretically, for a sample-concatenation synthesizer, the larger the number of recordings (database size), the better synthesis quality can be achieved. Large databases imply less transformations, avoiding unnatural artifacts. However, we argue that a minimal database should cover at least:

- three notes per octave;
- three dynamics: piano, mezzo and forte;
- two articulation types: staccato and legato.

Finally, for a given musical phrase recording, the process of adding a new instrument can be summarized with the following steps:

1. *Definition of recording score.* The score should include melody, dynamics, tempo and articulation. Additionally, other expressive resources such as vibrato, crescendos can be included.
2. *Performances recording.* The performer should play the score focusing on the indications written in the score. Recordings are stored as WAV files.

3. *Manual annotation.* Using an audio editor (e.g. Wavesurfer[15]), the user loads the musical phrases and place segmentation markers, labeling each segment as silence, attack, sustain, release or transition. The output are text files containing a row for each segment with *beginTime, endTime, label*.

4. *Pre-Analysis files.* The four binary files (pitch, peaks, dynamics and phase-alignment) are created using an external stand-alone analysis application.

5. *XML writer.* By means of a scripting language, we can create the XML file.

6. *Mapping definition.* Following the indications exposed in section 6.3.2, the mapping functions should be adapted for the new instrument.

6.5.2 Assessment

As we mentioned in section 4.5.2, a formal evaluation of the system should include tests with several users. For practical reasons, we opted to make an experiment with a reduced group of experts users. We describe here two assessment experiments with subjects: perceptual and usability. Both experiments compare different mapping configurations for the three instrument sound databases. At the same time, we generated several examples demonstrating the control of instrumental sound synthesis for: bass guitar, solo (Audio 6.10) and with accompaniment (Audio 6.11) ; bass guitar, solo (Audio 6.12) and with accompaniment (Audio 6.13) ; and finally violin, solo (Audio 6.14) and with accompaniment (Audio 6.15) . For the last examples of violin sound synthesis are generated with an off-line violin synthesizer under development at the MTG, which is also based on sample-concatenation. These examples demonstrate the potential of voice-driven synthesis when using a synthesizer with higher quality. In this case, apart from working off-line, the violin synthesizer uses a larger sample database and the transformation algorithms are specially adapted to the violin sound.

6.5.2.1 Perceptual experiment

In this perceptual experiment, users compare different versions of synthesized sounds. Each version uses different mapping configurations. Participants judge the "naturalness" of each version from 0 (very bad) to 5 (very good).

Table 6.11 shows the results of the experiment. For the bass guitar sound, the highest rating is obtained using a transposition of one octave and smoothed envelopes for pitch and dynamics. For the case of the saxophone, subjects perceived more natural the synthesis using the database (DB2), which contains single notes, instead of long phrases. We argue that this is due to the better sound quality of the recordings of the DB2 compared to the DB1. The highest rating is obtained using a smooth envelope for pitch but with rapid responding envelope for dynamics.

[15] `http://www.speech.kth.se/wavesurfer`

	Rate (0-5)	PT	PE	PQ	DG	DE	DC	TM
Bass guitar								
Configuration 1 (Audio 6.16)	2.8	−12	0.1	0.2	−6	0.25	0	0
Configuration 2 (Audio 6.17)	1.6	−12	1.0	0.0	−6	1.0	0	0
Configuration 3 (Audio 6.18)	2.2	0	0.1	0.0	−6	0.0	0	0
Configuration 4 (Audio 6.19)	2.0	0	0.1	0.0	−6	1.0	0	0
Sax (DB1)								
Configuration 1 (Audio 6.20)	1.2	0	0.15	0.0	0	0.0	0	2
Configuration 2 (Audio 6.21)	1.0	0	1.0	0.0	0	1.0	0	2
Sax (DB2)								
Configuration 1 (Audio 6.22)	3.0	0	0.1	0.3	−12	0.75	0	2
Configuration 2 (Audio 6.23)	2.6	0	1.	0.0	−2	1.0	0	2

Table 6.11: Results from the subjective experiment with 5 participants. Different mappings are compared for bass and saxophone synthesis. Rightmost columns detail the mapping for each configuration. PT (Pitch transposition), PE (Pitch envelope), PQ (Pitch quantization), DG (Dynamics gain), DE (Dynamics envelope), DC (Dynamics compression). For saxophone synthesis, two different databases are compared, one containing phrases (DB1) and one containing single notes (DB2).

6.5.2.2 Usability test

The goal of this experiment is two fold, on the one hand, to compare different mapping configurations for different instrument sound databases, one for bass guitar, and two for saxophone; and on the other hand, to gather feedback about the limitations of the current prototype. Four subjects with professional experience on real-time digital musical systems participated in the experiment. Each session lasted 25 minutes, consisting of three parts: introduction , evaluation, discussion. We used to following set-up: Pentium M Centrino 2.0GHz, 1GB RAM (laptop computer), Edirol UA25 (USB audio interface with ASIO drivers), a Shure SM58 (dynamic microphone), Edirol MA-7A (monitors) and Sennheiser eH 150 (headphones). Subjects preferred the use of headphones, monitoring also the input voice.

In the introductory part, we explained the purpose of the experiment, and let users to get familiar with the system. In the evaluation part, we asked users to compare three mapping configurations for each instrument database. Table 6.12 presents the results, where users had to rate the sense of control and the sound quality of the synthesized sound from 0 (very bad) to 5 (very good). From the results, we observe that users mostly prefer a rapid control of pitch and dynamics (configuration 2), without applying quantization or smoothing. An interesting result is that this contradicts the results of the perceptual experiment, where most subjects preferred the configuration with smoothed pitch and dynamics envelopes. Also, we see that the synthesis of bass guitar sound is in general more convincing that for the saxophone.

Finally, in the discussion part, we identify the following main issues:

	Control (0-5)	Synthesis (0-5)	PT	PE	PQ	DG	DE	DC	TM
Bass guitar									
Configuration 1	2.0	2.5	−12	0.1	0.15	−6	0.0	0.6	0
Configuration 2	4.25	3.5	−12	1.0	0.0	−3	1.0	0	0
Configuration 3	2.0	2.25	−12	0.55	1.0	−3	0.5	0.75	0
Sax (DB1)									
Configuration 1	2.75	2.75	0	0.1	0.15	−6	0.0	0.6	2
Configuration 2	3.5	2.5	0	1.0	0.0	−3	1.0	0	2
Configuration 3	2.5	1.75	0	0.55	1.0	−3	0.5	0.75	2
Sax (DB2)									
Configuration 1	3.0	2.75	0	0.1	0.15	−6	0.0	0.6	2
Configuration 2	3.0	2.25	0	1.0	0.0	−3	1.0	0	2
Configuration 3	2.25	1.5	0	0.55	1.0	−3	0.5	0.75	2

Table 6.12: User test with 4 participants. Different mappings are compared for bass and saxophone synthesis. Rightmost columns detail the mapping for each configuration. PT (Pitch transposition), PE (Pitch envelope), PQ (Pitch quantization), DG (Dynamics gain), DE (Dynamics envelope), DC (Dynamics compression). For saxophone synthesis, two different databases are compared, one containing phrases (DB1) and one containing single notes (DB2).

- *Latency*: as we mentioned before, the system introduces a latency of 40.6 ms. Although all participants noticed the latency, the majority considered it acceptable for a system with such characteristics. Further strategies for reducing the latency, might involve the combination of the current approach with time-domain processing during note onsets. For instance, transforming the input voice signal, with filters or amplitude envelope modifiers, during a few initial milliseconds.

- *Synthesis control*: the most criticized aspect of the system is the sound quality of the synthesis. Subjects considered that, while for the bass guitar sound synthesis the results were convincing, the saxophone sound synthesis lacked of realism. We argue that the control of timbre in the synthesized sound is difficult for small databases. This is specially noticeable for dynamics, where, if only three dynamics are sampled (e.g. *p,mf,f*), the timbre of the output sound can be vary largely for similar values of input dynamics in the sample retrieval process. In a similar way, subjects mentioned that lack of controlling note attacks, for instance, having "slap" bass guitar sounds. To improve the control of the sound synthesis using a sample-concatenation technique implies using larger databases, e.g. covering the dynamic range with more resolution and recording a variety of playing styles. An alternative approach would consider the use of different synthesis techniques with an appropriate timbre model (see section 7.2).

- *Feedback*: in our experiment, the performer has two auditory feedback streams (voice and synthesis), which some users considered disturbing. This brings an interesting discussion. When

playing musical instruments, the performer can receive different types of feedback: auditory and non-auditory. The auditory feedback is the performance sound (e.g. violin sound), and the non-auditory is usually haptic (e.g. resitance force during bowing). In recording studios or in amplified live performances, monitoring is employed to ensure the auditory feedback to help the performer. When using the voice to control a synthesis sound, the performer has two auditory feedback streams that may differ in its characteristics (e.g timbre, pitch quantization). In our opinion, a performer should be able to learn the behaviour of such systems. At the same time, we propose to explore this phenomena in future research experiments (see section 7.2).

- *Learning curve*: the last comment concerns the entry fee of the system before the user can produce a musically interesting result. Subjects that were not confident with singing found the system more difficult to control. We believe that due to the particular characteristics of this system, the user requires some training to learn the latency and feedback constraints. Nevertheless, users with singing experience will achieve a more precise control, for instance getting a stable pitch, producing vibratos, etc.

6.6 Summary

In this chapter, we addressed the vocal control of instrumental sounds, focusing on mapping aspects. We devised three mapping layers, a first *Performance Analysis* (voice to vocal gestures), a second layer generally called *Mapping* (from vocal gestures to intermediate parameters), and a third layer that fills the synthesizer's *Internal Score* from these intermediate parameters. The main contribution of this chapter refers to the second layer. We differentiated two different types of functions: phonetic-based and instrument based. The former determines the articulation, using the phonetic classification from the vocal gestures analysis, ranging from staccato to legato articulations. The latter determines the intermediate parameters' envelopes (pitch, dynamics, timbre and time-scaling) according to the characteristics of the synthesized instrument sound. We showed the importance of these mappings in the context of voice-driven synthesis, which represents an improvement over systems based on pitch-to-MIDI converters. Finally, we developed a real-time prototype that demonstrates the obtained results if this dissertation. This tool can be used as a VST-plugin in various audio software applications, either as a compositional tool or for live performing.

Chapter 7

Conclusion

This dissertation addressed the use of the singing voice as a control source for sound synthesizers. Nowadays, state of the art synthesizers are able to generate realistic high-quality sounds, using physical models or advanced sample-based techniques. However, major challenges in current digital musical instruments are on the control side. Since the advent of MIDI, a wide variety of musical interfaces (musical controllers) have been proposed to control synthesizers, including pitch-to-MIDI converters. A main criticism of these controllers is the lack of integration with the sound engine. This research has explored a path toward more integrated digital musical instruments, in which the design of the interface and its mapping functions are adapted the characteristics of the synthesized sound. The design of a *singing-driven interface* should take into account four different factors: performer, imitated instrument, synthesis technique and time mode (real-time and off-line).

From a number of case studies and experiments, we learned the main issues involved in using the singing voice as a control input. On the one hand, by controlling a singing voice synthesizer, we showed that the expression can be effectively transmitted into the synthesized singing. In this specific example (voice input to voice synthesis) mappings can be direct. On the other hand, from the research on vocal imitation of instruments, we found out that the voice signal consists of sequences of syllables, also referred as syllabling signals. By putting syllabling in the context of instrumental gestures, we suggest the concept of vocal gestures, whose phonetics can be classified according to a musical function. The study of syllabling permits, in a further step, to design specific methods for feature extraction and to define appropriate mappings. Finally, we developed two software prototypes that incorporate the obtained results. These real-time modules demonstrate the usability of voice-driven synthesis.

This research can impact different areas of *Sound and Music Computing* application. It extends the list of musical controllers that together with high-quality instrument synthesizers can be employed either in a live performance or in a composition context. Since voice is universal, singing-driven interfaces would allow also non-skilled users to experiment with digital instruments in a direct

way (e.g. "singing a violin"). Moreover, with this work we expect to encourage further research on the use of singing-driven interfaces. This can be later applied either to the musical arena with more experimental-oriented performances, or in interactive media (e.g. video-games).

From the initial aims of this dissertation, we think that the vocal gestures analysis represents a significant step towards the use of the voice as a musical controller. Concerning the study of mapping strategies, from voice features to synthesizer parameters, they can be further explored specially on timbre aspects. We consider that the control in the developed prototype for instrumental sounds is still limited. The creation of extensive sample databases (with manually annotated, high quality recordings) might overcome this limitation, improving the synthesis quality. Next, we describe the main contributions of this dissertation.

7.1 Summary of contributions

Review of sound-driven sound systems Most common real-time digital music systems are signal processors and synthesizers. In the former a sound stream is transformed in a certain way (e.g. distortion, reverberation, pitch shifting, etc.). In the latter, an input device (e.g a keyboard, wind controller, etc.) sends control information to a synthesis engine. The topic of this dissertation relates to a third category of music systems, those referred as sound-driven sound systems. In section 2.2, we provided a detailed survey of systems in which the sound input is used as a control signal.

Study of vocal imitation of musical instruments An important contribution of this dissertation is a novel study about the role of phonetics in a vocal imitation of musical instruments. We carried out a web survey collecting more than 850 transcriptions of a set of musical phrases performed on sax, violin and bass guitar. A first conclusion is that people typically use one syllable per note, containing a central vowel and optional begin and end consonants. Also, besides some general trends (e.g. people choose mostly the vowel /a/), we identified a correspondence between the timbre and articulation of the imitated instrument and the employed phonetics. Section 3.3.3 presents the detailed results of the survey.

Low-delay Phonetic Alignment Controlling a singing voice synthesis with an input performance can be achieved by means of phonetic alignment of the audio signal and the lyrics. We implemented a low-delay alignment algorithm that allows real-time interaction. The principal limitation of our approach is due to the different languages used in the phonetic alignment, Japanese for the trained acoustic models and Spanish for the voice input and synthesis, which reduced the accuracy of the algorithm. A future step is to integrate trained acoustic models in Spanish, which have been unfortunately not available for this research.

Formant Tracking Algorithm Voice timbre can be easily modulated by varying the position of the articulators. In the signal, these variations correspond to changes in the resonances (formants). We introduced a new method of formant tracking, specifically adapted to vocal control in real-time. The proposed algorithm is based on the Discrete-Cepstrum. From a set of frequency candidates, it uses Dynamic Programming, in which cost functions are based on "anchor" patterns for five typical vowel sounds. To increase the robustness, transition cost depends on the spectral variation.

An initial evaluation with *syllabling* examples, our algorithm performs satisfactorily. However, when evaluated with a speech corpus and compared to a state-of-the-art algorithm based on Linear Prediction (LP), the performance of our algorithm is lower. One reason is that since our approach expects vowel sounds, formant frequencies for voiced consonants such as nasals or liquids, are not accurately estimated. Also, the LPC-based method is computed off-line, achieving backtracking of the whole utterance.

Syllabling Segmentation Algorithm In the study on instrument imitation, we observed that voice signal consists of syllables with a musical meaning. We developed a segmentation algorithm especially designed for this type of signals. We proposed a phonetic classification based on its musical function. The segmentation algorithm relies on heuristic rules that primarily look at timbre variations. An evaluation of the algorithm showed that it performs clearly better than general-purpose onset detection algorithms. The latter gave a lot of *false positives* since it is not adapted to voice signals. An additional contribution is a corpus of syllabling recordings with annotated segmentations, which is publicly available on the thesis website.

Mapping strategies for voice to instrument We described in chapter 5 a three-layer mapping, which converts a voice signal into the control parameters of a sample-concatenation synthesizer. A first layer analyzes a voice signal and outputs vocal gestures, as described in chapter 3. In a second layer, we proposed a set of functions for converting vocal gestures into a set of perceptual abstract parameters. We specified at the same time, the appropriate configuration for three instrument families. The third layer is involved with the creation of an internal score from the abstract parameters.

***PDriven* Prototype** The *PDriven* prototype is a VST plugin that allows the use of a voice as input for controlling a Singing Voice Synthesizer (SVS). It features phonetic-alignment and expression analysis in real-time. We control an existing SVS in Spanish that is based on sample concatenation. We believe that the developed prototype can represent an interesting add-on tool for state-of-the-art SVS's. It facilitates the process, now tedious, of adding expression to a SVS. For singers, it might be as way to experiment with new voices and timbres, without using less-adapted interfaces such as keyboards.

Main limitations of the prototype are the latency and the phonetic alignment accuracy. The former is mainly due to the internal architecture of the synthesis engine, which concatenates diphone

samples. Detected phoneme changes in the input voice correspond actually to the beginning of the transition of the diphone sample in the synthesis.

***VDriven* Prototype** The *VDriven* prototype is a VST plugin that allows the use of a voice as input for controlling a the synthesis of other musical instruments. It features vocal gestures segmentation and it allows different mapping configurations. It incorporates the research carried out on mapping voice features to the control parameters of a sample concatenation synthesizer. We have built databases for two instruments, but the system can be easily extended with new instruments.

Initial evaluation experiments of the system indicate that users are engaged with the possibility of controlling an instrument with their voice. The principal drawbacks encountered are the latency and the limited timbre space of the synthesized instrument. The former is specially noticeable in note attacks. The latter is motivated by the reduced size of the instrument database. We argue that current databases should be extended, covering sufficiently the instrument sonic space (e.g. several note to note articulations, changing the plucking position on a bass guitar, or generating roughness on a sax).

***Singterface* Library** Although the analysis methods are already integrated in the *VDriven* prototype, we compiled these algorithms in a software library to be used by third-parties. The library includes: syllabling segmentation, formant tracking and breathiness analysis. In terms of technology transfer activities, the *Singterface* library is being integrated in an music education application by an educational software company.

7.2 Further research directions

In the first chapter, we have already mentioned that the topic of this dissertation is quite broad and it can be addressed from different perspectives. We undertook an engineering approach, resulting in the development of two prototypes, one controlling a singing voice synthesizer and one controlling the synthesis of other musical instruments.

We believe that this dissertation motivates further research on voice-driven interfaces. Specifically, we propose here a number of further research directions that address: the role of voice timbre, automatic expression description and studies on users' behaviour.

7.2.1 Context-aware voice-driven synthesis

In chapter 6, we addressed the design of a number of mapping functions that convert voice features into control parameters for the synthesizers. Although, we have constantly identified the differences

between real-time and off-line use cases, the advantages of the off-line synthesis in terms of expressivity can be extended. An off-line analysis eliminates causality and latency constraints, thus context information (looking at the expression over a sequence of notes) may improve the results. This is particularly useful in synthesizers based on sample-concatenation. The sample retrieval process can include searches taking into account specific expressive resources (e.g. type of vibrato, portamento, crescendo, etc.). In the case of singing voice synthesis, it can be extended with timbre aspects related to voice quality (e.g. breatghiness, growl, etc.). It brings at the same time, the need of extending the database with an extensive sampling of the performer sonic space (Bonada and Serra, 2007). Concerning the analysis methods, machine learning techniques can be used to characterize musical gestures (Maestre and Gómez, 2005; Maestre et al., 2006).

7.2.2 Timbre-oriented voice-driven synthesis

In this dissertation, we focused on controlling pitch, dynamics and articulation, where the control of the synthesis timbre is not fully explored. In our experiments, timbre was primarily determined by the instrument, and in the case of the singing voice, it was determined by the lyrics. Of course, the voice presents enormous timbre possibilities that can be further exploited in a voice-driven synthesis context. As a further research direction to explore, we propose "Timbre-oriented Synthesis", which can use the proposed vocal gestures analysis to control traditional sound synthesis techniques such as subtractive, or amplitude and frequency modulation (Roads, 1996). It should also study perceptual aspects of different kinds of mapping. This type of synthesis can be appropriate in the arena of electronic music where more experimental artists are eager to discover new controllers. From early works on timbre in the Computer Music field (Grey, 1975, 1977), more recent approaches employ machine leaning techniques using spectral processing (Wessel et al., 1998; Jehan and Schoner, 2001b).

7.2.3 Performer gestures with cross-instrumental auditory feedback

This dissertation includes a few basic user experiments with the developed voice-driven synthesis protoypes. We strongly think that an study of the role of the performer on voice-to-instrument synthesis, should be extended taking into account also cognitive aspects. We argue that it would constitute a research topic in itself, and it would need the collaboration of experts in musicology and cognition to define the experiments. The objective of this research is to study performer's behaviour when controlling the sound of different instruments. Under the term *cross-instrumental auditory feedback*, we refer to a situation in which the performer is playing one instrument (voice) but listening to the synthesis of another instrument. This research could be equally extended to other input instruments than voice.

For carrying out such experiments would include quantitative (based on signal processing) and qualitative (based on user surveys) evaluations. For the former, machine learning techniques could be here also used to characterize musical gestures (Maestre and Gómez, 2005; Maestre et al., 2006).

7.2.4 Voice-driven sound retrieval applications

Besides controlling traditional sound synthesis techniques, another interesting path to explore related to timbre is using *mosaicing* synthesis techniques. In this case, timbre similarity methods can be used to navigate in sound databases (either consisting of a small set of loops or big sound repositories, e.g. The FreeSound Project[1]) using the voice as a control. This would lead to new horizons for voice-driven synthesis, incorporating research in the area of Music Information Retrieval and audio *mosaicing* applications (Schwarz, 2005; Zils and Pachet, 2001).

7.2.5 Generalization to instrument-driven sound synthesis

Finally, a natural progression of the present research is to extend this to allow any type of instrumental signal as input. This would create a many-in, many-out system with very interesting properties. Imagine a duet of violin and flute, where the violin controls a flute synthesizer, while the flute controls a violin synthesizer. Due to its popularity, a particularly interesting case is to have an guitar as input instrument to control a variety of synthesized sounds. The goal would be to study the use of a guitar as a polyphonic controller, including the implementation of analysis methods for both monophonic and polyphonic signals. Also, it should encompass the implementation of a guitar-controller synthesizer including sample database creation and the appropriate mapping functions.

[1] `http://freesound.iua.upf.edu`

Appendix A

Related Publications

In this annex, we provide a list of publications of relevance to this dissertation in which the author has participated. Abstracts and electronic versions of most of these publications are available at `http://www.mtg.upf.edu`.

1. **Janer, J. Maestre, E. 2008.** *Mapping phonetic features for voice-driven sound synthesis*, E-Business and Telecommunication Networks - Selected papers from ICETE 2007. CCIS Series. Springer-Verlag.

2. **Janer, J. Maestre, E. 2007.** *Phonetic-based mappings in voice-driven sound synthesis*, Proceedings of International Conference on Signal Processing and Multimedia Applications - SIGMAP 2007; Barcelona, Spain.

3. **Janer, J. Peñalba, A. 2007.** *Syllabling on instrument imitation: case study and computational methods*, Proceedings of 3rd Conference on Interdisciplinary Musicology; Tallinn, Estonia.

4. **Janer, J. Bonada, J. Blaauw, M. 2006.** *Performance-driven control for sample-based singing voice synthesis*, Proceedings of 9th International Conference on Digital Audio Effects; Montreal, Canada.

5. **Janer, J. Bonada, J. Jordà, S. 2006.** *Groovator - an implementation of real-time rhythm transformations*, Proceedings of 121st Convention of the Audio Engineering Society; San Francisco, CA, USA.

6. **Janer, J. 2005.** *Feature Extraction for Voice-driven Synthesis*, Proceedings of 118th Audio Engineering Society Convention; Barcelona.

7. **Janer, J. 2005.** *Voice-controlled plucked bass guitar through two synthesis techniques*, Proceedings of 2005 International Conference on New Interfaces for Musical Expression; Vancouver, Canada.

8. **Janer, J. Loscos, A. 2005.** *Morphing techniques for enhanced scat singing*, Proceedings of 8th Intl. Conference on Digital Audio Effects; Madrid, Spain.

9. **Fabig, L. Janer, J. 2004.** *Transforming Singing Voice Expression - The Sweetness Effect*, Proceedings of 7th International Conference on Digital Audio Effects; Naples, Italy.

10. **Janer, J. 2004.** *Voice as a musical controller for real-time synthesis*, Doctoral Pre-Thesis Work. UPF. Barcelona.

Appendix B

List of Audio Examples

In this appendix, we provide the list of audio examples referenced in the dissertation. All audio files are available online at: `http://www.mtg.upf.edu/~jjaner/phd`.

Chapter 3	
Audio 3.1.	Input voice for the bass guitar synthesis using pitch-2-MIDI.
Audio 3.2.	Bass guitar synthesis with DigitalEar (pitch-2-MIDI transcription) as input.
Audio 3.3.	Bass guitar synthesis with the Instrumentizer prototype.
Audio 3.4.	Bass guitar synthesis using a keyboard MIDI input.
Audio 3.5.	Input voice example used in the transcription example of figure 3.5.
Audio 3.6.	Simple voice-driven synthesis in PureData, using built-in analysis modules.
Audio 3.7.	Bass guitar synthesis using Physical Modeling. Input voice and synthesis sound.
Audio 3.8.	Bass guitar synthesis using Spectral Morph. Input voice and synthesis.
Audio 3.9.	Clarinet sound example used to observe the pitch contour in figure 3.13.
Audio 3.10.	Voice example used to observe the pitch contour in figure 3.14.
Audio 3.11.	Clarinet synthesis experiment. Original clarinet phrase to be imitated.
Audio 3.12.	Clarinet synthesis experiment. Voice input 1.
Audio 3.13.	Clarinet synthesis experiment. Synthesized clarinet 1.
Audio 3.14.	Clarinet synthesis experiment. Voice input 2.
Audio 3.15.	Clarinet synthesis experiment. Synthesized clarinet 2.
Audio 3.16.	Clarinet synthesis experiment. Synthesized clarinet with WX5 input.
Audio 3.17.	Syllabling in music education. Master classes recording (clarinet).
Audio 3.18.	Syllabling in music education. Master classes recording (violin).
Audio 3.19.	*Scat it* web survey. Example musical phrase to be transcribed.

Chapter 4

Audio 4.1.	Loudness variation due to phonetics for the utterance [a-ma].
Audio 4.2.	Pitch variation due to phonetics for the utterance [a-ga].
Audio 4.3.	Voice used in an example of the low-delay phonetic alignment algorithm.
Audio 4.4.	Input male performance, with the same singer used to create the MTG-SVS.
Audio 4.5.	Input male performance. Offline synthesis.
Audio 4.6.	Input male performance. Real-time synthesis.
Audio 4.7.	Input female performance.
Audio 4.8.	Input female performance. Offline synthesis.
Audio 4.9.	Input female performance. Real-time synthesis.
Audio 4.10.	Instrument to singing. Voice input used for singing voice synthesis example.
Audio 4.11.	Instrument to singing. Violin input used for singing voice synthesis example.
Audio 4.12.	Instrument to singing. Piano input used for singing voice synthesis example.
Audio 4.13.	Instrument to singing. Voice-driven singing voice synthesis example.
Audio 4.14.	Instrument to singing. Violin-driven singing voice synthesis example.
Audio 4.15.	Instrument to singing. Piano-driven singing voice synthesis example.

Chapter 5

Audio 5.1.	Voice used in an example of the formant tracking algorithm.
Audio 5.2.	Voice used in an example of the formant tracking algorithm.
Audio 5.3.	Voice used in an example of the syllabling segmentation algorithm.
Audio 5.4.	Voice used in an example of the syllabling segmentation algorithm.
Audio 5.5.	Example of breathy voice.
Audio 5.6.	Musical scale sung with normal and breathy phonation.
Audio 5.7.	Voice used in an example for comparing pitch estimation.

Chapter 6

Audio 6.1.	Voice to show the micro-variations present in the pitch and loudness analysis.
Audio 6.2.	Phonetic-specific mappings. Staccato voice input.
Audio 6.3.	Phonetic-specific mappings. Staccato violin synthesis.
Audio 6.4.	Phonetic-specific mappings. Legato voice input.
Audio 6.5.	Phonetic-specific mappings. Legato violin synthesis.
Audio 6.6.	Timbre mappings. Input voice.
Audio 6.7.	Timbre mappings. Saxophone sound synthesis without timbre control.
Audio 6.8.	Timbre mappings. Saxophone sound synthesis with timbre control.
Audio 6.9.	Mappings to abstract parameters envelopes. Voice example.
Audio 6.10.	Database file example: a musical phrases with several notes.
Audio 6.11.	Demo 1. Bass guitar synthesis example with voice (left), Bass guitar (right).
Audio 6.12.	Demo 2. Bass guitar synthesis example with accompaniment.
Audio 6.13.	Demo 3. Saxophone synthesis example with voice (left), saxophone (right).
Audio 6.14.	Demo 4. Saxophone synthesis example with accompaniment.
Audio 6.15.	Demo 5. Violin offline synthesis example with voice (left), violin (right).
Audio 6.16.	Demo 6. Violin offline synthesis example with accompaniment.
Audio 6.17.	Perceptual experiment. Bass guitar sound synthesis(proposed mapping).
Audio 6.18.	Perceptual experiment. Bass guitar sound synthesis(Variation 1).
Audio 6.19.	Perceptual experiment. Bass guitar sound synthesis (Variation 2).
Audio 6.20.	Perceptual experiment. Bass guitar sound synthesis(Variation 3).
Audio 6.21.	Perceptual experiment. Sax sound synthesis with DB1(proposed mapping).
Audio 6.22.	Perceptual experiment. Sax sound synthesis with DB1(Variation 1).
Audio 6.23.	Perceptual experiment. Sax sound synthesis with DB2(proposed mapping).
Audio 6.24.	Perceptual experiment. Sax sound synthesis with DB2(Variation 1).

Bibliography

Amatriain, X., Bonada, J., Loscos, A., Arcos, J. L., and Verfaille, V. (2003). Content-based transformations. *J. New Music Research*, 32(1):95–114.

Amatriain, X., Bonada, J., Loscos, A., and Serra, X. (2002). *DAFX - Digital Audio Effects*, chapter Spectral Processing, pages 373–438. U. Zoelzer ed., J. Wiley & Sons.

Arfib, D. (1979). Digital synthesis of complex spectra by means of multiplication of non linear distorded sine waves. *J. Audio Eng. Soc.*, 27:250–65.

Arfib, D., Couturier, J.-M., Kessous, L., and Verfaille, V. (2002). Strategies of mapping between gesture parameters and synthesis model parameters using perceptual spaces. *Org. Sound*, 7(2):135–52.

Baecker, R. M. and S, B. W. (1987). *Readings in Human Computer Interaction: A Multidisciplinary Approach*, chapter The Haptic Channel, pages 357–365. Morgan Kaufmann, San Mateo, California.

Bevilacqua, F., Müller, R., and Schnell, N. (2005). MnM: a Max/MSP mapping toolbox. In *Int. Conf. on New Interfaces for Musical Expression, Vancouver*, pages 85–8.

Bilbao, S. (2005). A finite difference plate model. In *Proc. Int. Comp. Music Conf. (ICMC'05), Barcelona*.

Bonada, J., Blaauw, M., and Loscos, A. (2006). Improvements to a sample-concatenation based singing voice synthesizer. In *Proceedings of 121st Convention of the Audio Engineering Society*, San Francisco, CA, USA.

Bonada, J. and Loscos, A. (2003). Sample-based singing voice synthesizer by spectral concatenation. In *Proceedings of Stockholm Music Acoustics Conference 2003*, Stockholm, Sweden.

Bonada, J. and Serra, X. (2007). Synthesis of the singing voice by performance sampling and spectral models. *IEEE Signal Processing Magazine*, 24(2):67–79.

Brossier, P. (2006). *Automatic annotation of musical audio for interactive systems*. PhD thesis, Centre for Digital music, Queen Mary University of London.

Buchla, D. (2005). A history of buchla's musical instruments. In *Int. Conf. on New Interfaces for Musical Expression, Vancouver*, page 1.

Cadoz, C. (1988). Instrumental gesture and musical composition. In *Proc. Int. Comp. Music Conf. (ICMC'88), San Francisco*, pages 1–12.

Cadoz, C. and Wanderley, M. M. (2000). *Trends in Gestural Control of Music*, chapter Gesture-Music, pages 71–94. M.M. Wanderley and M. Battier, IRCAM.

Camurri, A., Hashimoto, S., Ricchetti, M., Trocca, R., Suzuki, K., and Volpe, G. (2000). Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal*, 24(1):57–69.

Canazza, S., De Poli, G., Drioli, C., Rod, A., and Vidolin, A. (2004). Modeling and control of expressiveness in music performance. *Proceedings of the IEEE*, 92(4):286–701.

Cano, P. (1998). Fundamental frequency estimation in the SMS analysis. In *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98), Barcelona*, Barcelona.

Cano, P., Loscos, A., Bonada, J., de Boer, M., and Serra, X. (2000). Voice morphing system for impersonating in karaoke applications. In *Proc. Int. Comp. Music Conf. (ICMC'00), Berlin*, pages 109–12.

Cappé, O., Laroche, J., and Moulines, E. (1995). Regularized estimation of cepstrum envelope from discrete frequency points. In *Proc. IEEE Workshop on Applications of Digital Signal Processing to Audio and Acoustics*.

Cardenoso-Payo, V. and Escudero-Mancebo, D. (2002). Statistical modelling of stress groups in spanish. In *Proceedings of Speech Prosody 2002, Aix en Provence*, pages 207–210.

Celma, O., Gómez, E., Janer, J., Gouyon, F., Herrera, P., and Garcia, D. (2004). Tools for content-based retrieval and transformation of audio using mpeg-7: The spoffline and the mdtools. In *Proceedings of 25th International AES Conference*, London, UK.

Chowning, J. (1971). The synthesis of complex audio spectra by means of frequency modulation. *J. Audio Eng. Soc.*, 21:526–34.

Clarisse, L., Martens, J., Lesafre, M., Baets, B. D., Meyer, H. D., and Leman, M. (2002). An auditory model based transcriber of singing sequences. In *Proceedings of the ISMIR 2002, 3th International Conference on Music Information Retrieval, Paris*, pages 116–123.

Cook, P. (1991). *Identification of Control Parameters in an Articulatory vocal Tract Model, with applications to the Synthesis of the Singing.* PhD thesis, Stanford University, Stanford.

Cook, P. (1992). Spasm: a real-time vocal tract physical model editor/controller and singer: the companion software synthesis system. *Computer Music Journal*, 17:30–44.

Cook, P. (1998). Toward the perfect audio morph? singing voice synthesis and processing. In *Proceedings of the 1st. International Conference on Digital Audio Effects (DAFX)*, Barcelona.

Cook, P. (2001). Principles for designing computer music controllers. In *Int. Conf. on New Interfaces for Musical Expression (NIME 2001)*, pages 1–4, Singapore, Singapore. National University of Singapore.

Cook, P. and Scavone, G. P. (1999). The synthesis toolkit (STK). In *Proc. Int. Comp. Music Conf. (ICMC'99), Beijing*.

Dahl, S. and Bresin, R. (2001). Is the player more influenced by the auditory than the tactile feedback from the instrument? In *Proc. of the COST-G6 Workshop on Digital Audio Effects (DAFx-01), Limerick*, pages 194–197.

Dalla Bella, S., Giguère, J., and Peretz, I. (2007). Singing proficiency in the general population. *The Journal of the Acoustical Society of America*, 121:1182.

Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, pages 357–366.

de Cheveigné, A. and Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111:1917.

Delalande, F. (1988). *La gestique de Gould: élements pour une sémiologie du geste musical*, chapter Glenn Gould, Pluriel, pages 83–111. Louis Corteau Editrice Inc.

Deller Jr, J., Proakis, J., and Hansen, J. (1993). *Discrete Time Processing of Speech Signals*. Prentice Hall PTR Upper Saddle River, NJ, USA.

Deng, L., Cui, X., Pruvenok, R., Huang, J., Momen, S., Chen, Y., and Alwan, A. (2006). A database of vocal tract resonance trajectories for research in speech processing. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'2006), Tolouse*.

Depalle, P., Tassart, S., and Wanderley, M. M. (1997). Instruments virtuels. *Résonance*.

Ding, W., Kasuya, H., and Adachi, S. (1996). Multiphonic note identification. *IEICE Trans. Inf. and Syst.*, E78-D(6):738–743.

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

Egozy, E. B. (1995). Deriving musical control features from a real-time timbre analysis of the clarinet. Master's thesis, Massachusetts Institut of Technology.

Fant, G. (1960). *Acoustic Theory of Speech Production*. Gravenhage, Mouton.

Fels, S. (1994). *Glove Talk II: Mapping Hand Gestures to Speech Using Neural Networks*. PhD thesis, University of Toronto.

Fernald, A. (1989). Intonation and communicative intent in mothers speech to infants: Is the melody the message? *Child Development*, 60:14971510.

Fletcher, N. H. and Rossing, T. D. (1998). *The Physics of Musical Instruments*. 2nd ed. Springer-Verlag, Berlin and New York.

Fléty, E. (2002). Atomic pro: a multiple sensor acquisition device. In *Int. Conf. on New Interfaces for Musical Expression (NIME 2002), Singapore*.

Fonollosa, J. A. R., Batlle, E., and Mario, J. B. (1998). Low delay phone recognition. In *EURASIP IX European Signal Processing Conference. EUSIPCO, Rhodes, Greece*.

Friberg, A. (2006). pdm: an expressive sequencer with real-time control of the kth music performance rules movements. *Computer Music J.*, 30(1):37–48.

Friberg, A., Bresin, R., and Sundberg, J. (2006). Overview of the kth rule system for musical performance. *Advances in Cognitive Psychology, Special Issue on Music Performance*, 2(2-3):145–161.

Galas, T. and Rodet, X. (1990). An improved cepstral method for deconvolution of source-filter systems with discrete spectra: Application to musical sounds. In *Proc. Int. Comp. Music Conf. (ICMC'90), Glasgow*, pages 82–8.

Georgaki, A. (2004). Virtual voices on hands: Prominent applications on the synthesis and control of the singing voice. In *Proceedings of Sound and Music Computing 2004, Paris*.

Gershenfeld, N., Schoner, B., and Metois, E. (1999). Cluster-weighted modelling for time-series analysis. *Nature*, 397:329–332.

Gillet, O. and Richard, G. (2005). Drum loops retrieval from spoken queries. *Journal of Intelligent Information Systems*, 24:2/3:159–177.

Gómez, E. (2006). *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra.

Grachten, M. (2006). *Expressivity-aware Tempo Transformations of Music Performances Using Case Based Reasoning*. PhD thesis, Universitat Pompeu Fabra, Barcelona.

Grey, J. M. (1975). *An Exploration of Musical Timbre*. PhD thesis, Stanford University.

Grey, J. M. (1977). Multidimensional perceptual scaling of musical timbres. *J. Acoust. Soc. Am.*, 61(5):1270–7.

Grey, J. M. and Moorer, J. A. (1977). Perceptual evaluation of synthetic music instrument tones. *J. Acoust. Soc. Am.*, 62:454–62.

Haas, J. (2001). Salto - a spectral domain saxophone synthesizer. In *Proceedings of MOSART Workshop on Current Research Directions in Computer Music*, Barcelona.

Hämäläinen, P., Mäki-Patola, T., Pulkki, V., and Airas, M. (2004). Musical computer games played by singing. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-04), Naples*.

Haus G., P. E. (2001). An audio front end for query-by-humming systems. In *Proceedings of the ISMIR 2001, 2th International Conference on Music Information Retrieval*, pages 65–72.

Herrera, P. and Bonada, J. (1998). Vibrato extraction and parameterization in the spectral modeling synthesis framework. In *Proc. COST-G6 Workshop on Digital Audio Effects (DAFx-98), Barcelona*.

Herrera, P., Serra, X., and Peeters, G. (1999). Audio descriptors and descriptor schemes in the context of MPEG-7. In *Proc. Int. Comp. Music Conf. (ICMC'99), Beijing*.

Hitchcock, H. (1986). *The New Grove Dictionary of American Music*. Wiley and Stanley Sadie eds., Grove Dictionaries of Music, Inc., New York.

Hunt, A. and Wanderley, M. M. (2002). Mapping Performer Parameters to Synthesis Engines. *Org. Sound*, 7(2):97–108.

Hunt, A., Wanderley, M. M., and Paradis, M. (2002). The Importance of Parameter Mapping in Electronic Instrument Design. *Org. Sound*, 7(2):429–40.

Infusion Systems Ltd. (2002). I-CubeX, http://www.infusionsystems.com.

Jaffe, D. and Smith, J. (1983). Extensions on the Karplus-Strong plucked-string algorithm. *Computer Music Journal*, 7(2):56–69.

Jameson, J. W. (2002). Voice-controlled electronic musical instrument.

Janer, J. (2000). AML - Architecture and Music Laboratory. Porting an interactive installation from NeXT/ISPW to Macintosh/Max-MSP. Master's thesis, La Salle, Universitat Ramon Llull, Barcelona.

Janer, J. (2004). DEA: Voice as a musical controller for real-time synthesis. Master's thesis, Universitat Pompeu Fabra.

Janer, J. (2005a). Feature extraction for voice-driven synthesis. In *118th Conv. Audio Eng. Soc., Barcelona*, Barcelona.

Janer, J. (2005b). Voice-controlled plucked bass guitar through two synthesis techniques. In *Int. Conf. on New Interfaces for Musical Expression, Vancouver*, pages 132–134, Vancouver, Canada.

Janer, J. (2006). Estudi de casos pel control de síntesi dinstruments musicals virtuals mitjanćant la veu cantada. Technical report, Agència de Gestió d'Ajuts Universitaris i de Recerca, Generalitat de Catalunya.

Janer, J., Bonada, J., and Blaauw, M. (2006a). Performance-driven control for sample-based singing voice synthesis. In *Proceedings of 9th International Conference on Digital Audio Effects*, pages 41–44, Montreal, Canada.

Janer, J., Bonada, J., and Jordà, S. (2006b). Groovator - an implementation of real-time rhythm transformations. In *Proceedings of 121st Convention of the Audio Engineering Society*, San Francisco, CA, USA.

Janer, J. and Peñalba, A. (2007). Syllabling on instrument imitation: case study and computational methods. In *Proceedings of 3rd Conference on Interdisciplinary Musicology*, Tallinn, Estonia.

Jehan, T. and Schoner, B. (2001a). An audio-driven perceptually meaningful timbre synthesizer. In *Proc. Int. Comp. Music Conf. (ICMC'01), Havana*.

Jehan, T. and Schoner, B. (2001b). An audio-driven, spectral analysis-based, perceptually meaningful timbre synthesizer. In *110th Conv. Audio Eng. Soc.*, Amsterdam, Netherland.

Jensenius, A., Camurri, A., Castagné, N., Maestre, E., Malloch, J., McGilvray, D., Schwarz, D., and Wright, M. (2007). The need of formats for streaming and storing music-related movement and gesture data. In *Proceedings of International Computer Music Conference 2007*, Copenhagen, Denmark.

Jensenius, A. R. (2007). *Action Sound, Developing Methods and Tools to Study Music-Related Body Movement*. PhD thesis, University of Oslo.

Jensenius, A. R., Kvifte, T., and Godoy, R. I. (2006). Towards a gesture description interchange format. In *Int. Conf. on New Interfaces for Musical Expression, Paris*, pages 176–179. IRCAM; Centre Pompidou.

Johnston, J. (1988). Transform coding of audio signals using perceptual noise criteria. *IEEE on Selected Areas in Communications*.

Jordà, S. (2005). *Digital Lutherie: Crafting musical computers for new musics performance and improvisation*. PhD thesis, Universitat Pompeu Fabra.

Jordà, S., Kaltenbrunner, M., Geiger, G., and Bencina, R. (2005). The reactable*. In *Proceedings of International Computer Music Conference 2005*, Barcelona.

Kapur, A., Benning, M., and Tzanetakis, G. (2004). Query-by-beat-boxing: Music retrieval for the dj. In *ISMIR-2004*.

Karplus, K. and Strong, A. (1983). Digital synthesis of plucked-string and drum timbres. *Computer Music J.*, 7(2):43–55.

Kellum, G. (2007). Violin driven synthesis from spectral models. Master's thesis, UPF. Barcelona.

Kelly, J. L. and Lochbaum, C. C. (1962). Speech synthesis. In *Proc. Fourth Int. Congress on Acoustics*, volume 4, pages 1–4.

Kendon, A. (2004). *Gesture: Visible Action as Utterance*. Cambridge University Press.

Kernfield, B. (1988). *The New Grove Dictionary of Jazz*. Grove Dictionaries of Music, Inc.

Kessous, L. (2004). Gestural control of singing voice, a musical instrument. In *Proceedings of Sound and Music Computing 2004, Paris*.

Klapuri, A. (2006). Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of the ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, Canada*.

La Kitchen, P. (2002). Kitchen.lab, http://www.la-kitchen.fr/kitchenlab.

Laroche, J. and Dolson, M. (1999). New phase vocoder technique for real-time pitch-shifting, chorusing, harmonizing and other exotic audio modifications. *J. Audio Eng. Soc.*, 47(11):928–36.

Laver, J. (1994). *Principles of Phonetics*. Cambridge University Press.

Lee, A., Kawahara, T., and Shikano, K. (2001). Julius — an open source real-time large vocabulary recognition engine. In *Proc. European Conference on Speech Communication and Technology*, pages 1691–1694.

Leman, M., editor (2007). *Embodied Music Cognition and Mediation Technology*. MIT Press, Cambridge, Massachussetts.

Lesaffre, M., Tanghe, K., Martens, G., Moelants, D., Leman, M., Baets, B. D., Meyer, H. D., and Martens, J. (2003). The mami query-by-voice experiment: Collecting and annotating vocal queries for music information retrieval. In *Proceedings of the ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore*.

Lieberman, P. and Blumstein, S. E. (1986). *Speech physiology, speech perception, and acoustic phonetics*. Cambridge University Press.

Loscos, A. (2007). *Spectral Processing Of The Singing Voice*. PhD thesis, Universitat Pompeu Fabra, Barcelona.

Loscos, A. and Aussenac, T. (2005). The wahwactor: a voice controlled wah-wah pedal. In *Proceedings of 2005 International Conference on New Interfaces for Musical Expression*, Vancouver, Canada.

Loscos, A., Cano, P., and Bonada, J. (1999). Low-delay singing voice alignment to text. In *Proc. Int. Comp. Music Conf. (ICMC'99), Beijing*, Beijing, China.

Loscos, A. and Celma, O. (2005). Larynxophone: Using voice as a wind controller. In *Proceedings of International Computer Music Conference 2005*, Barcelona.

Machover, T. (1992). *Hyperinstruments - a Composer's Approach to the evolution of Intelligent Musical Instruments.* MillerFreeman, Inc.

Maestre, E. (2006). *Coding Instrumental Gestures. Towards a quantitative description of instrumental gestures in excitation-continuous musical instruments.* PhD thesis, Universitat Pompeu Fabra, Barcelona.

Maestre, E., Bonada, J., Blaauw, M., Perez, A., and Guaus, E. (2007). Acquisition of violin instrumental gestures using a commercial emf device. In *Proceedings of International Computer Music Conference 2007*, Copenhagen, Denmark.

Maestre, E., Bonada, J., and Mayor, O. (2006). Modeling musical articulation gestures in singing voice performances. In *Proceedings of 121st Convention of the Audio Engineering Society*, San Francisco, CA, USA.

Maestre, E. and Gómez, E. (2005). Automatic characterization of dynamics and articulation of monophonic expressive recordings. *Procedings of the 118th AES Convention.*

Maher, R. C. and Beauchamp, J. (1994). Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *J. Acoust. Soc. Am.*, 95(4):2254–63.

Mäki-Patola, T. and Hämäläinen, P. (2004). Latency tolerance for gesture controlled continuous sound instrument without tactile feedback. In *Proc. Int. Comp. Music Conf. (ICMC'04), Miami.*

Marrin, T. (2000). *Inside the Conductor's Jacket: Analysis, Interpretation and Musical Synthesis of Expressive Gesture.* PhD thesis, MIT Media Laboratory, Massachusetts.

Marshall, M. T., Peters, N., Jensenius, A. R., Boissinot, J., Wanderley, M. M., and Braasch, J. (2006). On the development of a system for gesture control of spatialization. In *Proc. Int. Comp. Music Conf. (ICMC'06), New Orleans.*

Mathews, M. V. and Guttman, N. (1959). Generation of music by a digital computer. In *Proc. of the Third Int. Congress on Acoustics*. Amsterdam: Elsever Publishing Company.

Mathews, M. V. and Moore, F. R. (1970). GROOVE - a program to compose, store, and edit functions of time. *Communications of the ACM (CACM)*, 13(12):715–21.

Mayor, O., Bonada, J., and Loscos, A. (2006). The singing tutor: Expression categorization and segmentation of the singing voice. In *Proceedings of 121st Convention of the Audio Engineering Society*, San Francisco, CA, USA.

McAulay, R. and Quatieri, T. (1984). Magnitude-only reconstruction using a sinusoidal speech model. *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, pages*, 27(127.6):4.

Meron, Y. (1999). *High Quality Singing Synthesis using the Selection-based Synthesis Scheme*. PhD thesis, Universitiy of Tokyo.

Métois, E. (1996). *Musical Sound Information: Musical Gesture and Embedding Synthesis*. PhD thesis, Massachusetts Institute of Technology.

Miranda, E. and Wanderley, M. (2006). *New Digital Musical Instruments: Control and Interaction beyond the Keyboard*. A/R Editions.

Moore, F. (1988). The dysfunctions of MIDI. *Computer Music Journal*, 12(1):19–28.

Moulines, E. and Charpentier, F. (1990). Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Com.*, 9(5/6):453–67.

Nakata, T. and Trehub, S. E. (2004). Infants responsiveness to maternal speech and singing. *Infant Behavior & Development*, 27:455464.

Oliver, W., Yu, J., and Metois, E. (1997). The Singing Tree: design of an interactive musical interface. In *Proceedings of the conference on Designing interactive systems: processes, practices, methods, and techniques.*, Amsterdam, The Netherlands.

O'Shaughnessy, D. (1987). *Speech Communication, Human and Machine*. Addison-Wesley, New York.

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

Patel, A. D. and Iversen, J. R. (2003). Acoustic and perceptual comparison of speech and drum sounds in the north indian tabla tradition: An empirical study of sound symbolism. In *Proc. of the 15th International Congress of Phonetic Sciences, Barcelona*.

Patten, J., Recht, B., and Ishii, H. (2002). Audiopad: A tagged based interface for musical performance. In *Int. Conf. on New Interfaces for Musical Expression (NIME 2002), Singapore*.

Paulo, S. and Oliveira, L. C. (2004). Automatic phonetic alignment and its confidence measures. *Advances in Natural Language Processing*, 3230:36–44.

Peeters, G. (2001). *Modèles et modélisation du signal sonore adaptés à ses caractéristiques locales*. PhD thesis, University of Paris VI.

Peeters, G. (2003). A large set of audio features for sound description (similarity and classification) in the cuidado project.

Perez, A., Bonada, J., Maestre, E., Guaus, E., and Blaauw, M. (2007). Combining performance actions with spectral models for violin sound transformation. In *Proceedings of 19th International Congress on Acoustics*, Madrid, Spain.

Pressing, J. (1992). *Synthesizer performance and real-time techniques*. Oxford Uninversity Press, Oxford.

Proakis, J. and Manolakis, D. (1983). *Digital signal processing: principles, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Puckette, M. (1995). Score following using the sung voice. In *Proc. Int. Comp. Music Conf. (ICMC'95), Banff*.

Rabiner, L. and Juang., B. H. (1993). *Fundamentals of Speech Recognition*. Prentice-Hall, Englewood Cliffs.

Rabiner, L. and Schafer, R. (1978). *Digital Processing of Speech Signals*. Englewood Cliffs, New Jersey: Prentice-Hall, 19.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 77(2):257286.

Ramakrishnan, C., Freeman, J., and Varnik, K. (2004). The architecture of Auracle: A real-time, distributed, collaborative instrument. In *Int. Conf. on New Interfaces for Musical Expression, Hamamatsu*, Japan.

Ramstein, C. (1991). *Analyse, représentation et traitement du geste instrumental*. PhD thesis, Institut National Polytechnique de Grenoble, France.

Roach, P., Stibbard, R., Osborne, J., Arnfield, S., and Setter, J. (1998). Transcription of prosodic and paralinguistic features of emotional speech. *Journal of the International Phonetic Association*, 28:83–94.

Roads, C. (1996). *Computer Music Tutorial*. The MIT Press, Cambridge, Massachusetts.

Rodet, X. (1984). Time-domain formant-wave-function synthesis. *Computer Music Journal*, 8(3):9–14.

Rowe, R. (1993). *Interactive Music Systems: Machine Listening and Composing*. MIT Press, Cambridge,MA.

Rubine, D. and McAvinney, P. (1990). Programmable finger-tracking instrument controllers. *Computer Music J.*, 14(1):26–42.

Salembier, P., Sikora, T., and Manjunath, B. (2002). *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley and Sons.

Schafer, R. and Rabiner, L. (1970). System for automatic formant analysis of voiced speech. *The Journal of the Acoustical Society of America*, 47:634–648.

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

Schwarz, D. (2005). Current Research in Concatenative Sound Synthesis. *Proceedings of the International Computer Music Conference (ICMC)*.

Schwarz, D. and Rodet, X. (1999). Spectral envelope estimation and representation for sound analysis-synthesis. In *Proc. Int. Comp. Music Conf. (ICMC'99), Beijing*.

Serra, X., Leman, M., and Widmer, G., editors (2007). *A Roadmap for Sound and Music Computing [Online], http://smcnetwork.org/roadmap*. The S2S Consortium.

Serra, X. and Smith, J. O. (1990). A sound decomposition system based on a deterministic plus residual model. *J. Acoust. Soc. Am., sup. 1*, 89(1):425–34.

Smith, J. (1991). Viewpoints on the history of digital synthesis. In *Proc. Int. Comp. Music Conf. (ICMC'91), Montréal*, pages 1–10, Canada.

Smith, J. (1992). Physical modeling using Digital Waveguides. *Computer Music Journal*, 16(4):74–91.

Steim, A. (2002). SensorLab, http://www.steim.org/.

Steiner, H.-C. (2005). [hid] toolkit: a unified framework for instrument design. In *Int. Conf. on New Interfaces for Musical Expression, Vancouver*, pages 140–3.

Sundberg, J. (1987). *The Science of the Singing Voice*. Dekalb, IL: Northern Illinois University Press.

Sundberg, J. (1994). Musical significance of musicians' syllable choice in improvised nonsense text singing: A preliminary study. *Phonetica*, 54:132–145.

Sundberg, J. (2001). Level and center frequency of the singer's formant. *J. Voice*, 15(2):176–869.

Sundberg, J., Askenfelt, A., and Frydén, L. (1983). Musical performance: A synthesis-by-rule approach. *Computer Music J.*, 7:37–43.

Talkin, D. (1987). Speech formant trajectory estimation using dynamic programming with modulated transition costs. *J. Acoust. Soc. Am.*, 82(1):55.

The International MIDI Association (1983). *MIDI 1.0 Detailed Specification.* http://www.midi.org.

Van Nort, D., Wanderley, M., and Depalle, P. (2004). On the choice of mappings based on geometric properties. *Proceedings of the 2004 conference on New interfaces for musical expression*, pages 87–91.

Verfaille, V. (2003). *Effets Audionumériques Adaptatifs : Théorie, Mise en Œuvre et Usage en Création Musicale Numérique.* PhD thesis, Université de la Méditerranée (Aix-Marseille II).

Verfaille, V., Boissinot, J., Depalle, P., and Wanderley, M. M. (2006a). Ssynth: a real time additive synthesizer with flexible control. In *Proc. Int. Comp. Music Conf. (ICMC'06), New Orleans.*

Verfaille, V. and Wanderley, M. M. (2005). Mapping strategies for gestural control of adaptive digital audio effects. *in preparation.*

Verfaille, V., Zölzer, U., and Arfib, D. (2006b). Adaptive digital audio effects (A-DAFx): A new class of sound transformations. *IEEE Trans. on Acoustics, Speech, and Signal Processing.*

Wanderley, M. (2001). *Intéraction Musicien-Instrument : application au contrôle gestuel de la synthèse sonore.* PhD thesis, Université Paris VI, IRCAM.

Wanderley, M. M. and Depalle, P. (2004). Gestural control of sound synthesis. *Proc. IEEE, Special Issue on Eng. and Music - Supervisory Control and Auditory Communication*, 92(4).

Wanderley, M. M. and Orio, N. (2002). Evaluation of input devices for musical expression: Borrowing tools from HCI. *Computer Music J.*, 26(3):62–76.

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 *Proc. Int. Comp. Music Conf. (ICMC'98), Ann Arbor*, pages 62–65.

Widmer, G. and Goebl, W. (2004). Computational models of expressive music performance: The state of the art. *J. New Music Research*, 3(33):203–216.

Wright, M., Chaudhary, A., Freed, A., Wessel, D., Rodet, X., Virolle, D., Woehrmann, R., and Serra, X. (2000). New applications of the sound description interchange format. In *Proc. Int. Comp. Music Conf. (ICMC'00), Berlin.*

Xia, K. and Espy-Wilson, C. (2000). A new strategy of formant tracking based on dynamic programming. In *Proc. Sixth International Conference on Spoken Language Processing (ICSLP'2000), Beijing*, pages 53–58.

Zils, A. and Pachet, F. (2001). Musical Mosaicing. In *Proc. of the COST-G6 Workshop on Digital Audio Effects (DAFx-01), Limerick*.

Zölzer, U., editor (2002). *DAFX - Digital Audio Effects*. U. Zoelzer ed., J. Wiley & Sons.