Voice as a musical controller for real-time synthesis

by

Jordi Janer i Mestres

Submitted in partial fulfillment of the requirements for
the degree of Diploma of Advanced Studies.
Doctorate in Computer Science and Digital Communication
Department of Technology

Tutor: Dr. Xavier Serra

Universitat Pompeu Fabra

Barcelona, September 2004

Abstract

The aim of this document is to explore the singing voice as a valid musical controller for real-time synthesis. We address the identification and extraction of the voice features, and the mapping of those features to a synthesis engine. Since the advent of digital music synthesis and MIDI, many controllers have been developed, ranging from the traditional keyboard to sophisticated computer vision systems. We propose to employ an ubiquitous and highly expressive instrument: the voice. This work first reviews aspects of voice analysis, and the control of different synthesis techniques. Finally, we present a prototype of *Vocal Controller* that synthesizes a bass guitar using two techniques: Physical Models and Spectral Models.

The proposal for the PhD dissertation consists of extending the current prototype, allowing for real-time performance situations; and validating the system with two groups of subjects: naive users and skilled musicians.

Preface

The evolution of Computer Music has brought in past years advanced algorithms for synthesizing sounds in real-time. Some of these algorithms target to simulate existing acoustical instruments by using different techniques such as physical models or sample triggering. A second group of algorithms explores ways to create new *artificial* sounds, for instance granular synthesis. A problem that arises with all these synthesis techniques is how these algorithms can be controlled by a musician in a performance situation. One of the goals of any controller should be to capture the performer's execution and transmit it to the synthesis engine in order to produce a musical experience as expressive as it is with a real instrument. Most control interfaces for electronic instruments are replicas of traditional instruments provided with several sensors that get detail of the performer's execution. Typically, keyboards are used for controlling synthesizers but other attempts like wind controllers fall within this category. Finally, there are the so-called *Gestural Controllers*, in which the performer interacts with the instrument by means of body gestures that are captured by an interface such as a Computer Vision based system.

In this work we present an approach that exploits the human voice as a meaningful expression controller for synthesizing sounds in real-time. In other words, to consider the voice as a multi-parametric controller. Taking advantage of the past research done in the field of Voice Analysis/Synthesis, we can extract from the voice many nuances, more than just Loudness and Pitch, such as Timber Characteristic, Breathiness, Attack or Hoarseness. If we are able to map these descriptors meaningfully onto a synthesizer's engine, we will have an ubiquitous intuitive controller that drives the voice's expression to specific nuances of a target instrument synthesis. This lead us to think in two different situations, first, naive users could "play" an instrument just by controlling pitch and loudness. On the other hand, a very skilled singer can use his highly expressive performance for controlling, for instance, a trumpet sound. Another basic example would be to map the roughness factor of the input voice to the distortion amount in the extended plucked-string algorithm. An aspect tha plays an important role is the real-time performance. Our system can be considered altogether as a musical instrument, therefore it should react instantly to performer's gestures.

In the system we can identify three blocks: Analysis, Mapping and Synthesis; in addition there is a feed-back signal consisting of the output, which will help the user to learn from the sound. In this work, the research focuses basically on the first two blocks, and explore how they can be applied for synthesizing a bass sound in real-time using two different techniques physical-models and spectral-models.

Finally, this work could not be done without the collaboration and support of many people. I thank Professor Xavier Serra for giving me the opportunity to work at the Music Technology Group. Also I would like to thank for their knowledge and advise Perfe, Àlex, Lars, Jordi, Sergi, ÒscarM, Fabien, Pedro, Günter, Martin, Alvaro, Pau, Emilia, ÒscarC, ex-Cre@mworkers, Pierre Dutilleux and Paulo F.L.

Contents

Chapter 1

Introduction

In this chapter we introduce the problem faced in this work, as well as the context in which the research has been carried out. In addition, some preliminary discussions are addressed.

1.1 Context

1.1.1 Motivation

This work is part of the *Doctorate Program in Computer Science and Communication* of the Universitat Pompeu Fabra, Barcelona. With this document, we aim at setting the foundations towards the fulfilment of the PhD dissertation. We study the human voice as a meaningful musical controller, and the research proposal is to provide a model for transmitting features of the voice to a synthesis engine.

The human voice, and particularly the singing voice, is an enormously expressive instrument, which offers a wide range of possibilities. The author was enlightened by this during the work for another project related with the singing voice. In that case, the project consisted in developing a *Vocal Processor*, a system capable of changing the voice quality. Target users of the system were professional recording studios.

The initial idea of this work was to extract automatically features form the voice signal, and then, use those features to control the generation of sound. This led us to consider the voice as a musical controller, in other words, to present the voice as a substitute for the omnipresent keyboard or other controllers in the domain of digital music synthesis.

Moreover the voice offers several advantages. First, even an absolutely novel user is capable of control basic features of the voice such as pitch variation and loudness. More skilled individuals like trained singers can attain control over more interesting and complex nuances such as vibrato, breathiness amount, etc. This fact appears to be significant, since the proposed system can be

used by beginners, while maintaining interest for experimented users, who can accomplish a much richer musical experience. Second, the voice is ubiquitous, we *all* can sing without any extra equipment but a microphone. This relieves our system from expensive input electronic devices with plenty of sensors and microprocessors. Finally, a side benefit is that it can be employed by disabled people, who could "play" otherwise unplayable instruments.

Although there are certainly other systems that attempt to control digital synthesis by an audio stream, such as Pitch-to-MIDI converters, the author could not find in the literature references that explored the voice as a complex musical controller.

Our purpose is to use the *Voice Controller* for synthesizing any sound, primarily to control digital simulations of acoustic instruments, but to other extent also to explore any experimental synthesized sound. As we present in the chapter 4, a first prototype has been developed, which synthesizes a bass guitar. The main reason for choosing a bass guitar in the first prototype is our intention of integrate the system in a DJ-oriented application. In order to provide the DJ with new expressive resources, we thought of an application in which the DJ triggers a rhythm and "sings" the bass line on top. However, the bass sound has obvious drawbacks. First, is a less rewarding instrument, hardly used in solos and thus, apparently less enjoyable. It is usually employed as support instrument for accompanying the melody or keeping the rhythm in popular music. Second, using the bass as target instrument, we may narrow the expressivity present in the performers voice. The degree of expression is larger in the singing voice than in a bass guitar sound. Hence, our system will lose expression.

On the other hand, this is also an advantage, since we can concentrate on the main aspects of the system, seeking its viability as a real-time controller. The full potential of the voice expressivity can be addressed in a further step.

1.1.2 The Author

Here you find a brief description of my trajectory.

I graduated as *Technical Telecommunication Engineer* for La Salle School of Engineering, Universitat Ramon Llull, Barcelona in December 1997. The End-studies Report was *FIR Filters. Static and Adaptive Methods*, in Dep. of Signal Theory.

I graduated as *Electronic Engineer, Sound and Image Specialty*, for La Salle School of Engineering, Universitat Ramon Llull, Barcelona in July 2000. The End-studies Project was titled *AML — Architecture and Music Laboratory. Porting an interactive installation from NeXT/ISPW to Macintosh/Max-MSP*, and was achieved at ZKM — Institut für Musik und Akustik[1], Karlsruhe, Germany.

[1] http://www.zkm.de

Since December 2003 I am Researcher and PhD student at the Music Technology Group of the Audiovisual Institute of the Universitat Pompeu Fabra, Barcelona. In the year 2004, I was also Assistant Lecturer in Signal Processing at the ESUP, UPF.

From December 2000 until December 2003, I was employed as DSP Engineer in the R&D Department for the audio company Creamware Datentechnik GmbH[2], Siegburg, Germany.

Between the years 1999 and 2000, I did an 11-month training stage at the ZKM — Center for Art and Media, in the Institute for Music and Acoustics, Karlsruhe, Germany.

Apart from my professional and academic trajectory in the audio field, my passion for music (a reason for working at the MTG) started long ago. Besides being a very limited keyboard player, I enjoy listen to music, discovering new sounds, and Co-directing a weekly musical program in a local radio station.

1.1.3 The Place

This work has been carried out at the Music Technology Group (MTG) [3] of the Audiovisual Institute (IUA) [4], Universitat Pompeu Fabra (UPF) [5] 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, interactive systems and others.

Among the various areas of expertise within the MTG, two topics are specially related to our work:

- Spectral Voice Processing
- Musical Content Description

Referring to Voice Processing, research carried out in the MTG has been integrated in a Singing Synthesizer, commercialized by Yamaha Corp. under the name of *Vocaloid*. Other specific work related to the singing voice applied to Karaoke systems and Vocal Processors.

On the other hand, a rapid emergent topic in the Computer Music community is the so-called *Music Information Retrieval*. Within this area, the MTG is also specially active by participating in European projects such as *Cuidado*

[2] http://www.creamware.de
[3] http://www.iua.upf.es/mtg
[4] http://www.iua.upf.es
[5] http://www.upf.edu

or *SIMAC* [6]. Here the group's expertise lies in Audio Description Algorithms, which can be further integrated in larger systems. A sample of the potential integration of such algorithms is the SoundPalette [9], an application developed in collaboration with the German company Creamware for the Cuidado Project.

1.1.4 Past Projects

Here we mention the projects in which the author has been involved in the period 2003-2004 as part of his research activities

- CUIDADO. EU Project
  I was responsible for maintaining the SoundPalette Offline application, and building a ready-to-use installation package. This application will be shown at the 25th International AES Conference (AES-Metadata) [9].

- Voice Fx Vocal Processor. External project for Yamaha
  I joined this project in its final stage. My task was the development of voice analysis and transformation algorithms. Also, I was involved in the preparation of the demo sounds.

- OpenDrama. EU Project.
  One of the developed applications is the OperaTutor. This tool helps singing students in reaching the expressivity of a professional singer. I was responsible for the application's real-time processing, which consisted in the integration of several voice transformations algorithms.

1.1.5 Semantic HiFi Project

This work is partially carried out under the umbrella of the EU-Project *Semantic HiFi* [7], FP6 - IST - 507913, which started in 2004 and has as duration of three years. *Semantic HiFi* 's goal is to develop the HiFi system of the future, which is hard-disk based. 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. The partners involved in this project are IRCAM (Paris) as coordinator, Native Instruments (Berlin), Fraunhofer Institut (Illmenau), Universitat Pompeu Fabra (Barcelona), Sony CSL (Paris) and Ben Gurion University (Israel).

In addition, the project contemplates the development of several software applications targeted to the DJ community. Certainly, the work presented here fits in this part. As we have mentioned, our final goal is to come up with a system capable of using the voice as a musical instrument. From our perspective, this appears to be pretty fascinating, which can widen the musical expression of a DJ performance. Although the DJ began as a "pure decorative" figure in clubs and dance-halls, it has gained a major reputation during the last decades. With the emergence and popularity of certain electronic music styles (techno, house,

[6] http://www.semanticaudio.org/
[7] http://shf.ircam.fr/

d'n'b, etc.), the DJ has acquired in many cases the status of *rock'n'roll star*. The main topics involved in our work for the Semantic HiFi Project are:

- Audio Transformation
- Voice Control
- User Interaction devices

1.2 Sound Synthesis or Sound Transformation.

The main goal of this work is to use the singing voice for producing a different palette of sounds, or, in a broader sense, to use our voice for playing a trumpet, a guitar, drums, a flute, or any other, existing or non-existing, musical instrument. But before going into the actual research carried out, a preliminary significant question arises:

Are we controlling *Sound Synthesis* or are we doing *Sound Transformation ?*

The answer is neither evident nor totally objective, but will put us into context, and additionally, help us to define better the characteristics of the system. This discussion explores the evolution of some electronic musical devices, such as synthesizers or sound processors. We try also to point out, concepts and techniques in order to clarify this question.

By observing the proposed system, we find *sound* at both ends: the singing voice acting as input; and the synthesized sound as output. Hence, we could define it as a sound processor. If we look at it from another perspective, the generated sound has its source in the synthesis algorithm and not in the actual input voice, which only controls the algorithm. In this case, we could refer to the system as a voice-driven synthesizer. This question actually addresses to three different areas in the field of Computer Music: Digital Audio Effects, Musical Interfaces and Sound Synthesis. We find some emerging Conferences and Journals concerning these areas such as DAFX [44] for Audio Effects and NIME [42] for musical interfaces. Researchers strive to find new ways of manipulating sound, from building new instruments to transforming sonic material by using the current technologies.

1.2.1 Brief Survey of Processors and Synthesizers.

In this section, we pose the differences between sound transformation and synthesis control. As we have mentioned our particular approach can be observed from different perspectives. From one side, we can justify that we are transforming the input voice by altering its characteristics; from the other side, we can also argue that in our system, there is an explicit synthesizer's engine controlled arbitrarily by an input audio stream. In any case, *it is clear that the*

boundaries between transformation and synthesis blur. In order to demonstrate such a statement, we may have a look at the table 1.1. This table contains a list of several electronic music devices, sorted from keyboard synthesizers to general audio processors. Note here that we have omitted any historical classification.

Device	*Description*	*Products*
CONTROLLED SYNTHESIS		
Keyboard Synthesizers	External-controlled synthesis	MiniMoog, YamahaDX7
Gestural Controllers	Embodied-controlled synthesis	Theremin, EyesWeb
Pitch-to-MIDI	Basic Audio-driven synthesis	IVL-PitchRider, DigitalEar
Vocal Controller	Augmented Voice-driven Synthesis	(this work)
Vocoder	Cross-Synthesis	Roland VP330
Harmonizer	Synthesis of new voices	Eventide
Auto-Tune	Automatic pitch-shift	Antares' AutoTune
Voice Processor	Transform voice characteristics	TC-Helicon's VoicePrism
Compressors	Enhance the voice's quality	dbx 160A
Reverberators, EQ's, etc.	General-audio processing units	Lexicon 480L
PROCESSING		

Table 1.1: author's classification of electronic devices, from controlled synthesis to audio processing.

In the table 1.1, we put on one end standard controlled synthesizers, such as a keyboard. On the other end, we put general audio processing units such as reverberation or equalizers. On the other hand, this classification focuses on the singing voice. Both ends of the table converge in the middle with devices concerning the singing voice. We argue that the differences between an *Harmonizer* and a Pitch-to-MIDI based synthesizer are not extreme. For example, by experimenting with an *Auto-Tune processor*, the output voice can resemble to a guitar-like sound, if we sing with fast pitch changes and activate the *Pitch Quantize* option. However, Pitch-to-MIDI systems failed, at least so far, in achieving a relevant success. The causes should be further investigated, since they probably reveal us aspects to take into account in the design of our system. Regarding our approach of *Voice-Controlled Synthesis*, this is actually a particular case of Spectral Morphing. Here, we are not trying to go from one sound to another, what is commonly accepted as *Audio Morph.* Rather, we attempt to control the characteristics of a target sound by another. [13]

We have addressed an issue that can gain importance during the next years involving the design of new music synthesizers and audio effects. It seems clear that with the growing potential of real-time techniques, the boundaries between sound transformation and sound synthesis vanish. Specially, it becomes clearer using spectral processing techniques.

1.2.2 A New Musical Controller ?

We have seen that the boundary between sound transformation and sound synthesis can be fuzzy. Nevertheless, we decided to include our approach in the category of *Voice Controlled Synthesis.* We argue that our system consists of

two well-defined blocks: voice analysis and instrument synthesis; with an interface layer bridging both. The next paragraphs aim at justifying this decision.

Acoustical Instruments are designed in order to allow the performer to control the sound production mechanism. It requires years of practice before one is able to control the nuances of an instrument. In this case, the instrument's control and the sound production shares the same physical elements. For instance, in a violin, the performer moves the bow on the string, which starts vibrating and thus, generating sound.

A second type of instruments are organs. The organ is conceptually different, since the performer plays on a keyboard situated separately from the pipes. In this case, the keyboard switches air valves that open or close the air flow to the pipes. This switching system was originally mechanic, but was later replaced by a circuit of electronic switches. Later, appeared the first analog synthesizer that work in a similar way. By means of potentiometers or keys, an electric signal controls the oscillators generating the signal driven to the output.

The third category of instruments are the digital instruments of all variety. Here, we find a layer in the middle working as interface in form of a digital data stream. This layer *maps* input musical gestures into parameters of the synthesis algorithm.

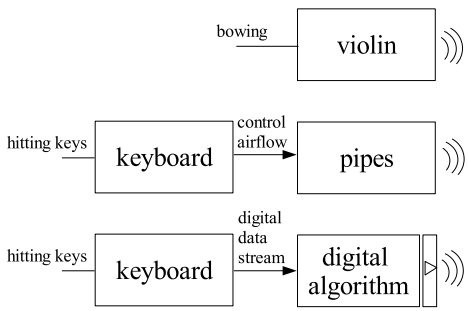


Figure 1.1: Three models of musical instruments. top: violin; middle: organ, bottom: digital keyboard synthesizer.

It is not necessary to say that our interest lies in the third model. The goal is to study how this model can be modified in order to have the human voice as input. However, it is interesting to observe that, from a certain point of view, our system could resemble the first model. The figure 1.1 shows the three models.

Concentrating on the third model, since the inception of MIDI in early 80's, this protocol has become the standard way of controlling digital music synthesis. Nowadays, there is probably none commercial synthesizer on the market without MIDI ports. Despite its popularity, MIDI presents also some limitations, mainly due to its reduced bandwidth. Note that, we are referring to a protocol specified

20 years ago, which has no adopted since then any modification. As we discuss later in section 3.3.1, our system does not incorporate MIDI communication, freeing ourselves of bandwidth limitations.

Our approach is still concerned with a "control plus synthesis" model, though. Musical gestures are captured by analysis algorithms that extract information from the input voice. A mapping layer, converts these features to parameters of a synthesis engine that generates the final sound. A first diagram of the functionality of the system is shown in the figure 1.2.

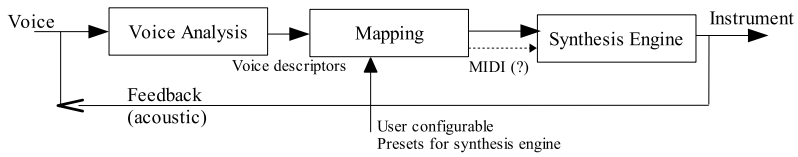


Figure 1.2: Functional block diagram: voice analysis, mapping and synthesis.

In the next chapters, we do a review of the main topics involving: Voice Analysis, Mapping and Synthesis. In the chapter 3, we present our contributions in considering the voice as a meaningful musical controller. Our goal will be to define which features are at the same time relevant and intuitive from performer's point of view.

Chapter 2

Background and Related Work

This chapter is structured in four sections that try to set the background topics for the research proposal. We give an overview of related concepts, techniques and systems, which in fact, is necessary before start doing any research activity. The analysis of the human voice is reviewed in the first section. Primarily, we will concentrate on the singing voice. In the second section, we explore the *musical controllers*, observing the evolution from acoustic instruments to the current devices of the digital era. The third section is dedicated to the sound generation process. We review different synthesis techniques, focusing on the *mapping* or how the algorithms can be controlled. Finally, the last section presents existing systems of different nature but with the common characteristic of generating sound from a singing voice's features.

2.1 Voice Analysis

In this section, we study the voice as a complex sound generator from different perspectives . The final goal of this work is to parameterize the singing voice in order to map the features onto a sound generator. Therefore, after an introduction of the physiology of the human voice, different techniques used in Voice Analysis/Synthesis and Speech Processing are reviewed. An important issue in our context is the fact that the extracted parameters are two-fold, these parameters must be musically-meaningful, and at the same time intuitive for the user *(e.g.producing a continuous pitch while changing from one vowel to another results in a sustained note with a varying timbre).*

2.1.1 Physiology of the human voice

We can arbitrarily 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,

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. In this part, we see 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 and the other parts that can be controlled by the singer.

2.1.1.1 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 $freq = 1/T_{pulse}$.

Now, we can observe in more detail why the vocal folds vibrate. In the vibration process, 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 there are much too slow for produce vibrations of hundreds 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 produce 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 frequency. Moreover, we are able to change this frequency, which is for us essential when dealing with 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 of photographic techniques, as it is explained in [61], 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. Now, if we observe the signal of the *Voice Source* in the spectral domain, we appreciate a more complex spectra than a single delta placed at the vibration frequency of the vocal folds. Certainly, the vibration produces a series of harmonic partials, multiple of the fundamental frequency or frequency of vibration f_1.

It has been shown that the level of this 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 of important deviations in the voice source spectrum of real subjects. Some experiments presented in [61], show that these deviations affect the difference of timbre between male and female voices. On the other hand, variations of phonation frequency and loudness result also in deviations of the aforementioned slope. Finally, it is necessary to mention that we find often in the literature the term *Voice Source* referred as *Glottal Source*. In this work, we will use both with no distinction.

We have seen how the vocal folds produce a pulsed excitation in form of variation 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 pulse, 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, emphasizing at the same time some frequencies *(resonance frequencies)*. In our case, the resonator is the vocal tract, and these frequencies are called *formant frequencies*. The four or the five first formants are the most relevant in the vocal tract. The two lowest are responsible for determining the vowel color, although all of them are of great significance to voice timbre. The partials generated in the glottal oscillator will be treated in various ways because of the formants, which will raise or attenuate certain frequencies. Therefore, we can conclude that the shape of the vocal tract is decisive to vowel quality and voice color.

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.

Once a configuration of articulators is set, the transfer function of the formants is defined, and hence the spectrum envelope. We have mentioned four or five predominant formant frequencies, which are placed between 200 Hz and

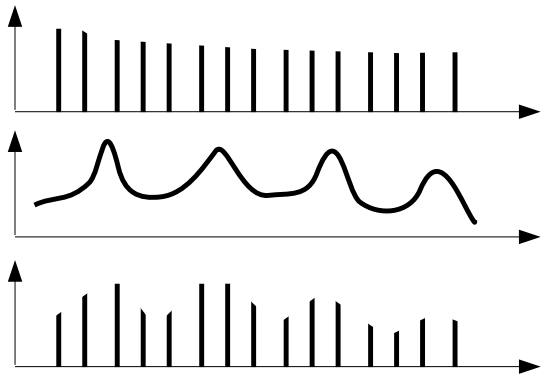


Figure 2.1: Spectrum of glottal source, formants and output spectrum.

$4kHz$. Due to the sound radiation characteristics and the high formant poles above 4 kHz, the amplitude of the formant frequencies is corrected with a high pass function. As we will see in the next part about control, we think of articulation in terms of the produced sounds rather than in positioning articulators.

2.1.1.2 Control

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 the parts of our phonatory system that are controllable, and how they are involved in the sound production. Later, it will help us in order to identify potential control parameters that can be mapped to the synthesis engine.

In the previous section, we presented the phonatory organ as a system consisting of two parts: *Source* and *Articulation*. When we produce a sound we are controlling elements of both parts simultaneously. Moreover, the necessary actions for this control are mostly performed unconsciously. For instance, and simplifying, in case of 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). It is obvious that when we speak or sing, we do not think of jaw's opening angle or muscle tension, instead we focus on the produced sound and adapt our phonatory system unconsciously.

However, there is a third participant, apart from Source and Articulation, which spectrum of glottal source, formants and output spectrum plays an important role in terms of control: Breathing. The vocal folds need an overpressure from the respiratory mechanism in order to start vibrating. This overpressure, called *subglottic pressure* [61] is significant for the amplitude and also, to some degree to the phonation frequency. It appears to be clear the importance of the breathing system, since these are two parameters over which a singer needs an excellent control. On the other hand, it explains the attention that is given to

the respiratory system in both therapy and training of the singing voice.

The sub-glottic pressure, air-pressure in the lungs, is raised when the abdominal muscles contract. It is related to the phonation loudness, as we can imagine. In reality, 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. From our point of view, we will consider the breathing system, and more precisely the sub-glottic pressure the main parameter for controlling the phonation 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 conform 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 glottal source, formants and output spectrum. We can all change the pitch of our voice. This is important not only in singing but also in speech, where the pitch variation may bring a lot of information. The prosody in questions differentiates from regular sentences by means of the pitch contour. Basically, when changing the pitch, the laryngeal musculature determines the length, tension and vibrating mass of the vocal folds. For raising or lowering the pitch, the musculature will determinate the properties of the vocal folds. They can be stretched to different degrees, becoming thiner 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. 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 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. Regarding voice disorders, our goal will be to identify them, and use them as control parameters. In the next sections 3.2.3, we present examples of this process.

Finally, in order to control the voice color, we need to deal with the formant frequencies of the vocal tract. As we have mentioned, the formant frequencies depend on the length and the shape of the vocal tract. The pronunciation of different vowel sounds is thus determined by the position of the articulators. 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 2.2.

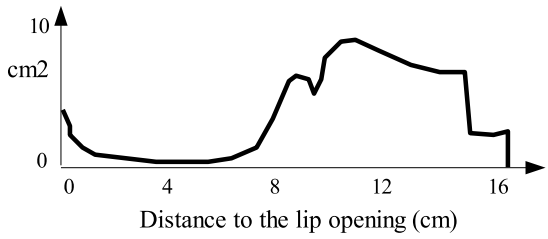


Figure 2.2: Approximative representation of the cross sectional area function of the vocal tract for the vowel */i/*, as shown in [61].

In an adult male, the vocal tract length is about 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 direct 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 vowel sounds, and intuitively we apply one of these patterns. These patterns are not identical in all languages. We may find vowel sounds in one language that are missing in others. Each articulation corresponds to the formant frequencies characteristic of that vowel. Basically, it is the two lowest formant frequencies that are important for the vowel quality. Furthermore, the vowels can be observed in a two-dimensional plane with the first and seconds formants frequencies as axes. The figure 2.3 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 vowel.

Summarizing this section, we have seen that the voice production is controlled in different ways in three different parts of the phonatory system: breathing, vocal folds and vocal tract.

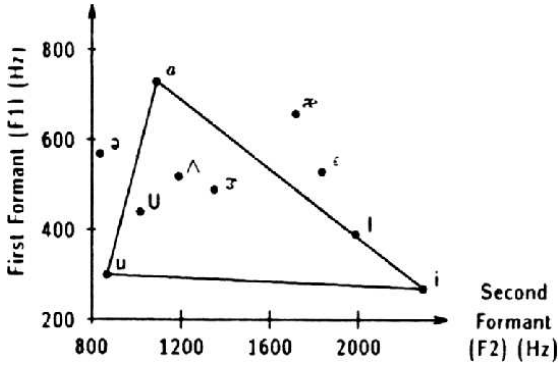


Figure 2.3: Vowel's triangle from [45]

2.1.2 Voice models

In this section, we review different approaches for the modelling of the singing voice, which were developed in the past years. Our study focuses on the voice's analysis. Nevertheless, we are aware of the fact that these models were conceived for Analysis/Synthesis systems. For many years, it has been the goal of many scientists and engineers to synthesize singing voice. It is necessary to mention that other relevant models particularity oriented to the synthesis of the singing voice are not discussed here. In this category we find the FOF(Formant Wave Functions) [51], which was used in the IRCAM's singing voice synthesizer called Chant. We identify three relevant approaches that tackle the analysis of the singing voice from different viewpoints:

- Formant Filter model
- Acoustic Tube model

Traditionally, the voice has been viewed as a source/filter system. An early but still prominent parametric model is the one proposed by G. Fant in [18]. By means of different sources and filters, we are able to generate a wide range of sounds with our voice. The *Formant Filter* model aims at identifying of filter's parameters and source's 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 2.1.3.2) analysis can be employed to automatically extract the parameters. On the other hand, 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 early 70's. Nowadays, it could not be considered for voice synthesis in terms of quality. Although, we do take into account the Formant Filter model, since it allows a very straightforward parameterization. The mapping from vowel's pronunciation to formant filters' frequencies, and then from formant frequencies to any control parameter, seems very promising.

The second model discussed here is the *Acoustic Tube* model. This can be considered completely as a *Physical Model*, since it simulates acoustically the vocal tract. In the section 2.1.1, 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 dimensions 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 $4KHz$ [13].

Previous work with the acoustic tube model in the Speech Synthesis field was carried out at Bell Labs. Kelly and Lochbaum developed a model for a smoothly varying tube [33], 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) [11] [59]. The tube is divided by sections of the same length. The acoustic impedance of each tube section is 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 [33]. The figure 2.4 shows the acoustic tube in its different forms: smooth varying, sampled version and the digital filter simulation.

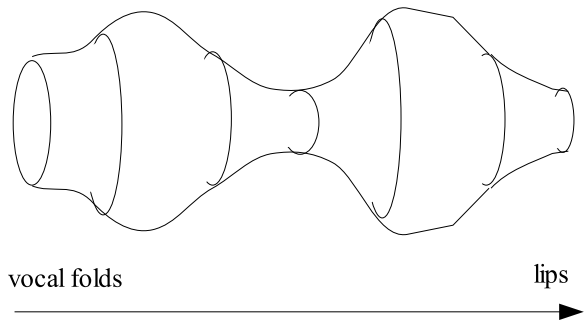


Figure 2.4: Acoustic tube model.

Additionally, this model has been used in the implementation of the singing synthesizer SPASM (Singing Physical Articulatory Synthesis Model) [12]. The system provided a graphical interactive environment, in which the user could

change the physical parameters of the vocal tract.

Now in the spectral domain, the Sinusoidal Model's approach presents yet another alternative for analyzing the singing voice. Here, the voice signal is modelled as a sum of sinusoidal components (2.1).

$$s(t) = \sum_{r=1}^{R} A_r(t)\ cos(\phi_r(t)) \tag{2.1}$$

In order to get the representation of the sound, we need to estimate the amplitude, frequency and phase of each sinusoidal. This estimation is done by first computing the SFTF (Short Fourier Transfer function) of the sound, and then, detecting spectral peaks. However, this representation would lack of realism, since in real voices there are more things than pure sinusoidals. A big improvement to this model is the *Sinusoidal plus Residual Model* [56]. In addition to the sinusoidal components, the analysis extract a residual signal $e(t)$, which in fact is assumed to be an 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 [64], including some practical applications. The most relevant processes are peak detection, pitch estimation and peak tracking.

For singing voice analysis, this model appears to be very convenient, since the extracted parameters can be easily mapped to the synthesis engine. Some of these parameters are: pitch, energy, sinusoidals spectral shape, residual spectral shape, sub-harmonicity, etc. We have commented that this spectral model performs the SFTF of the signal. It is necessary to mention, that the settings of the SFTF (window type, window size, zero-padding, etc.) are fundamental in order to attain good results. We review the characteristics of the Spectral Analysis more extensively in the section 2.1.4.

Furthermore, an application of this model for the synthesis of the singing voice can be found in [3]. In this approach, a Sinusoidal plus Residual model is combined with a newly developed model called *Excitation plus Resonance* (EpR).

2.1.3 Speech Processing

This work intends to present the voice as a valid musical controller. With this premise, we have considered through these pages the voice always as *sung voice*. At this point, we can not forget that most of the voice research carried out in the past is in the field of *Speech Processing*. Obviously, the impact of the

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 are considered as two complete different research fields. Nevertheless, we summarize in this section some Speech Processing techniques that will be of interest in our study.

Instead of doing speech recognition, this approach aims at using the vowel's timbre information for controlling parameters in a sound synthesis framework. Another important particularity is that in our case, the input signal is not speech but singing voice. We can extract certainly a large number of parameters (*descriptors*) from the analysis of the voice. However, since our system relates to user interaction, these descriptors should be equally intuitive. Therefore, it seems interesting to think of vowels as a control parameter. We present two different techniques used in formant estimation: Linear Prediction, Cepstral Analysis. Additionally, a novel approach using spectral centroids is presented in the section 3.2.2.1. The results are compared, keeping in mind the final goal of the system: to generate a control parameter that depends on the pronounced vowel. We will see that the resulting parameter is well represented by a point in a two-dimensional space, the *vowel's space*.

2.1.3.1 Vowel Estimator

The human voice has deserved a lot of studies in various scientific fields, such as medicine or signal processing. In the literature, we find our phonatory system often explained as a *Source/Filter* model. The *Source* are the vocal folds, and the vocal tract is a *Filter* with several resonances (also called *formants*). By changing the position of the articulators, we modify the shape and frequency of the formants and, in consequence, the characteristics of the produced sound. In general, if we assume that by estimating the frequencies of the first two formants, we can derive the vowel as shown in figure 2.5 from [45]. In this graph, the first formant frequency is displayed along the x-axis, and the second formant frequency along the y-axis.

In speech, we can basically differentiate two type of sounds: voiced and unvoiced. Among voiced sounds, we find in addition to vowels, also "sonorous consonants" (e.g. /m/, /n/, /l/, etc.). Moreover, each language has its own phonetics and a particular set of sounds associated. As we have mentioned, our system deals only with vowels, though. In fact, representing the vowels as a function of first and second formant in a two-dimensional space is our final goal. More precisely, to let the user determine the position of the *vowel point* inside the space by changing the articulators (i.e. the pronounced vowel). The implementation of visual feedback is also foreseen. Next, the output of the vowel estimator can be mapped to any control parameter of the synthesis algorithm, thus working as a controller. Here, we concentrate only on the formant estimation process.

Formant analysis can be traced back to the 1950's, and there are vari-

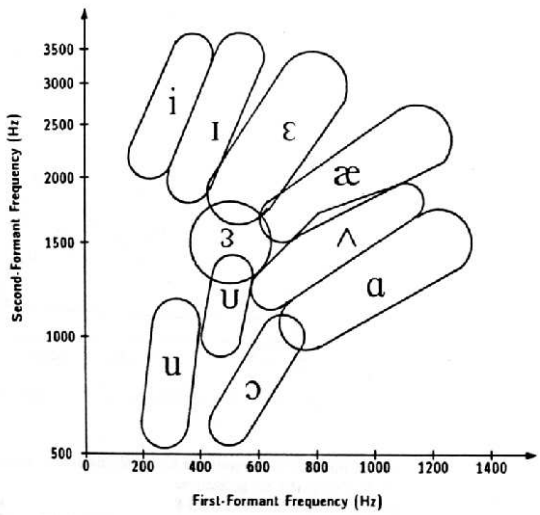


Figure 2.5: Vowels as a function of first and second formant. Extracted from [45]

ous existing techniques. Basically, these techniques achieve different types of "peak-picking" in the spectral domain. Some try to identify formants frequencies directly from the magnitude of the Discrete Fourier Transform. In other techniques, the "peak-picking" is performed on a smoothed spectrum such as Cepstral Analysis or LPC.

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 sung voice has a considerable higher pitch. On the other hand, formant analysis needs a low pitch in order to have the formant structure with enough resolution. The explanation is rather straight-forward: 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). Clearly, it is not feasible to determine the formant with just one peak value. Unfortunately, we do not present any alternative method to overcome this issue. Therefore, the system performance will be limited by the pitch.

2.1.3.2 Techniques

Modern methods commonly used nowadays in speech recognition systems are MFCC (Mel Frequency Cepstrum Coefficients) and PLP (Perceptual Linear Prediction). These techniques are based on two traditional approaches: Cepstral Analysis and Linear Prediction. In this section we review these two mentioned techniques for estimating the frequency of the first two formants. Moreover, a third approach, based on Spectral Peak Analysis, is presented.

Formant Estimation using LPC Linear Prediction 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.

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 [47].

$$\hat{s}[n] = \sum_{k=1}^{P} a_k s[n-k] \tag{2.2}$$

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. In equation 2.2, when the predicted sample equals the actual signal (i.e.,$\hat{s}[n]] = s[n]$, we have

$$S[z] = \sum_{k=1}^{P} a_k S[z] z^{-k} \tag{2.3}$$

$$S[z] = \frac{1}{1 - \sum_{k=0}^{P} a_k z^{-k}} \tag{2.4}$$

In order to find the prediction coefficients a_k, Rabiner and Juang [47] propose to use the autocorrelation signal r_{xx} with the correlation matrix R. The a_k are: $a_k = R^{-1} r_{xx}$. Since R is a *Toeplitz* matrix (it is symmetric with equal diagonal elements), the system can be optimally solved with the Levinson-Durbin algorithm. This is an iterative technique that allows to calculate the a coefficients without calculating the inverse of the matrix R. Although efficient, this algorithm does require numerical precision [20]. Once given the model structure, we need to know indeed the number of coefficients required, in order to represent the spectral resonances. Clearly, using a large number of coefficients will yield model spectra that better matches the original. However, this is not always an advantage. In our particular case, we want an estimation that is not affected by the pitch, thus the spectral envelope should remain unaffected by the harmonic's frequencies.

We implemented an LPC system in *Matlab* [26], using the built-in Levinson-Durbin algorithm. The complete function can be found in the appendix A. Additionally, we ran the system several vowel sounds at different pitch, yielding

the results presented in the following figures. For our particular case, the LPC estimator uses 14 coefficients.

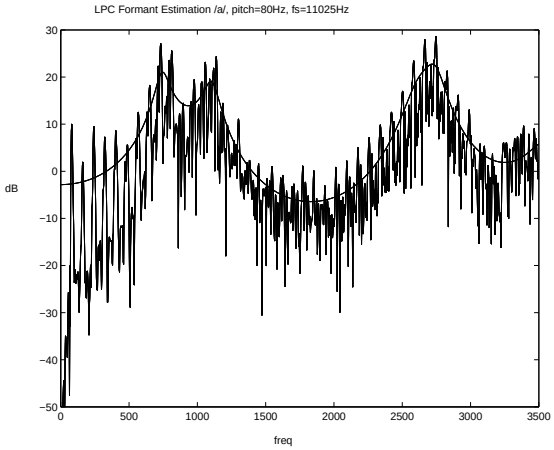


Figure 2.6: Spectrum of the vowel /a/ with $pitch = 80Hz$. Superimposed, the LPC approximation with 14 coefficients of the spectral envelope.

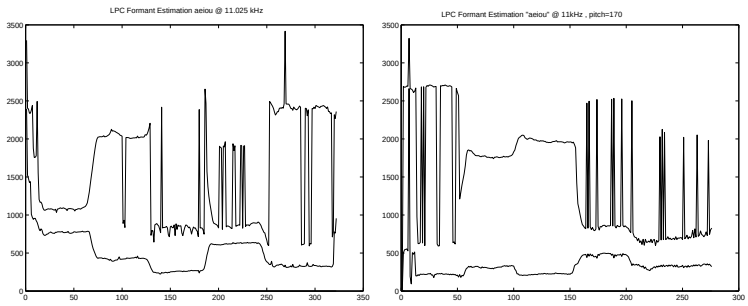


Figure 2.7: LPC. Formant frequencies $F1$ and $F2$ of a vowel sequence /a/-/ε/-/i/-/o/-/u/, with a)$pitch = 80Hz$ and b) $pitch = 170Hz$.

Formant Estimation using Cepstral Analysis The second technique presented is the formant estimation algorithm developed by Rabiner and Schafer based on cepstrum analysis and the chirp-z transform [54]. In our experiments, we ignored the chirp-z transform, though. 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 described in [54], the entire model can be expressed as,

$$H_{system}(z) = G(z)V(z)R(z) = (G(z)R(z)) \prod_{i=1}^{4} H_i(z) \tag{2.5}$$

note that $G(z)R(z)$ are grouped together because they are speaker dependant and can be approximated with a constant for all users with typical values of a and b,as written in equation 2.6. The vocal tract $V(z)$ is modeled as a cascade of resonators $H_i(z)$.

$$G(z)R(z) \equiv \frac{1 - e^{-aT_s}}{1 - z^{-1}\ e^{-aT_s}} \frac{1 - \ e^{-aT_s}}{1 - z^{-1}\ e^{-aT_s}} \begin{cases} a = 400\,\pi \\ b = 500\,\pi \\ T_s = samplingperiod \\ f > 5000Hz \end{cases} \tag{2.6}$$

The formant frequencies will corresponds to the poles in the $|H_{system}(\exp^{j\omega})|$ expression. Once we have seen the mathematical expression of our model, a block diagram of the complete formant estimation algorithm is shown in figure 2.8. The yielded result is the cepstrally smoothed spectrum of the input signal $\hat{F}[k]$.

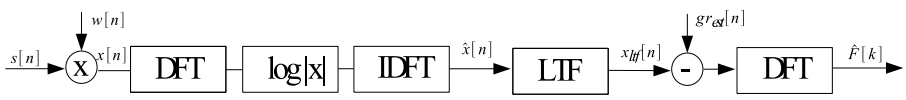


Figure 2.8: Cepstral Analysis. Block diagram of the entire algorithm

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* $\hat{x}[n]$.

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

For $t > T$, we can ignore the term $\hat{p}_\omega[n]$ from the expression, since it contains the periodicity information. Where T is the period of the signal, which is analyzed previously. Next, we subtract the glottal plus radiation component $\hat{gr_{est}}[n]$. The remaining signal is transformed to the frequency domain again using the FFT algorithm. The yielded result is $\hat{F}[k] = \sum_{k=1}^{4} \hat{H}_i[k]$. Where $\hat{H}_i[k]$ is one of the four first formants. At this point, the formant frequencies need to be estimated from $\hat{F}[k]$ with a peak-picking algorithm. Typically, for male voice, the first formant occurs in a range from 200 to $900Hz$, and the second from 550 to $2700Hz$.

Given an assumed frequency range, some rules help to find the formant frequencies $F1$ and $F2$ from $\hat{F}[k]$.

- $\hat{F}[k]/\hat{F}[k-3] > 1.005$
- $\hat{F}[k]/\hat{F}[k+3] > 1.005$
- $F1 < F2$
- $F2 - F1 > 200Hz$

We implemented this algorithm in Matlab [26], and ran it with different audio examples. Each analyzed example is a sequence of vowels with a constant pitch. The following figures (fig. 2.10) show the performance of the system with vowel sequences with a pitch at $80Hz$ and $170Hz$. The figure 2.9 depicts the spectrum of the vowel $/a/$ with a pitch of 80 Hz in a range from 0 to $3500Hz$. Superimposed we see the cepstrally smoothed log spectrum. By finding the two first relevant peaks of the smoothed spectrum, gives us the formant frequencies. On the other hand, in the figure 2.10, 4we observe the evolution of the calculated formant frequencies $F1$ and $F2$ in a sequence of vowels /a/-/ε/-/i/-/o/-/u/. Clearly, the "peak-picking" algorithms lacks of robustness when the pitch increases. As we mentioned, the original algorithm [54] is extended with the Chirp-z Transform in order to get a better frequency resolution. This can be useful if the poles are located too close together, making it impossible to identify the formant frequencies from the unit circle.

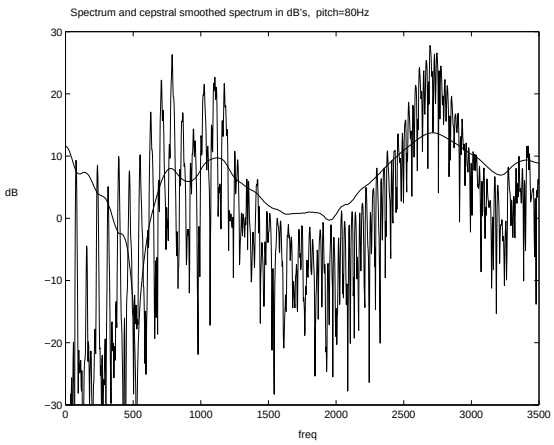


Figure 2.9: Spectrum and cepstrally smoothed spectrum of the vowel /a/ with $pitch = 80Hz$

2.1.3.3 Considerations

We have presented two different methods for estimating vowels in the special case of the singing voice. Assuming that a vowel can be represented by its first

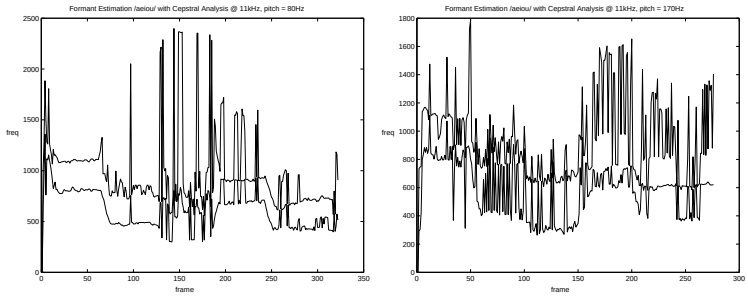


Figure 2.10: Cepstral Analysis. Formant frequencies $F1$ and $F2$ of a vowel sequence /a/-/ε/-/i/-/o/-/u/ with (a) $pitch = 80Hz$ and (b) $pitch = 170Hz$

two formant frequencies, we restrict our problem to a formant detection system. Traditionally, for extracting a model of the vocal tract, Linear Prediction and Cepstral Analysis have been used in multiple applications. We may find these approaches applied to either recognition or synthesis systems. Here, we discuss the differences encountered among the two implementations, and which method was finally chosen.

Linear Prediction is more efficient computationally than Cepstral Analysis, which needs several Fourier Transformations. The estimation of the first formant appears to be more stable, compared to the cepstral analysis. Anyway, in both techniques, the major problem encountered is the "peak-picking" algorithm, which should output stable values for the formant frequencies. A possible solution would be to build an additional Formant Tracking system, which should ensure the continuity and consistence of the formant frequencies. However, we decided to implement a new much simpler algorithm, Details about the implementation are found in section 3.2.2.1. We argue that even though we are losing accuracy, the results yielded by the *Dual Centroid Formant Estimation* are robust while maintaining enough consistency. As we have mentioned, the Formant Estimator is integrated in a larger framework. Although, the *DCFE* algorithm needs a pre-processing stage for spectral peak analysis, this is performed already within our framework. Thus, the added computational cost is minimum when compared to the other techniques, which would require a larger implementation.

Concluding, we studied two different methods for estimating formant frequencies in order to represent vowel sounds. Although, it lacks of certain precision, the *DCFE* method was chosen for its simplicity and robustness.

2.1.4 Spectral Analysis

Since early 90's Spectral Processing has gained a major role in the field of musical signal processing. An explanation for this success is related to psychoacoustics, since our auditory system works internally similar to a spectrum analyzer. A common utility of spectrum analysis is to identify timbres. How-

ever, spectrum is a physical property that can be characterized as a distribution of energy as a function of frequency; while timbre is a perceptual concept that allow us to differentiate between two sounds of same pitch and loudness.

There are several methods for analyzing the spectrum of a signal. Here in this work we describe 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.* 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.

Two different techniques are presented here. First, the Sinusoidal and the Spectral Modeling Synthesis [56], based on *Deterministic plus Stochastic Analysis.* Second the modified Phase-Locked Vocoder by Jean Laroche [34], which splits the spectrum in several harmonic regions in order to characterize the signal more accurately.

2.1.4.1 Sinusoidal Model

In the spectral domain, the Sinusoidal Model's approach presents an alternative for analyzing audio signals [39]. Here, the signal is modelled as a sum of sinusoidal components (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, we need to estimate the amplitude, frequency and phase of each sinusoidal. This estimation is done by first computing the SFTF (Short Fourier Transfer function) of the sound, and then, detecting spectral peaks. However, this representation would lack of realism, since in real voices there are more things than pure sinusoidals. A big improvement to this model is the *Sinusoidal plus Residual Model* [56]. In addition to the sinusoidal components, the analysis extract a residual signal $e(t)$, which in fact is assumed to be an 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 [64], including some practical applications. The most relevant processes are peak detection, pitch estimation and peak tracking.

For singing voice analysis, this model appears to be very convenient, since the extracted parameters can be easily mapped to the synthesis engine. Some

of these parameters are: pitch, energy, sinusoidals spectral shape, residual spectral shape, sub-harmonicity, etc. We have commented that this spectral model performs the SFTF of the signal. It is necessary to mention, that the settings of the SFTF (window type, window size, zero-padding, etc.) are fundamental in order to attain good results.

Furthermore, an application of this model for the synthesis of the singing voice can be found in [3]. In this approach, a Sinusoidal plus Residual model is combined with a newly developed model called *Excitation plus Resonance* (EpR).

The SMS model proposed by X. Serra [56] [57] assumes that the sound is composed by stable sinusoid (*partials)* plus noise (*residual component).* The analysis procedure studies the time-varying spectral characteristics of a sound. For pseudo-harmonic sounds, the harmonically related partials are grouped together, leaving the non-harmonic components and the noise as *residual.* As we can observe in the equation 2.9, the input sound is modeled by a number sinusoids plus a residual signal $e(t)$. This model assumes that the sinousids are stable partials of the sound, with slowly varying amplitude and frequency.

$$s(t) = \sum_{r=0}^{R} A_r(t) \cos\left[\phi_r(t)\right] + e(t) \tag{2.9}$$

In the analysis stage, a spectral frame can be classified either as *pitched* or *unpitched.* For the pitched frames, the separation of the harmonic and the noisy components relies on the peak-detection and continuations, and, on the fundamental frequency estimation. If we know the value of the fundamental frequency, we can set an appropriate window size that is related to a specific number of pitch periods.

Several methods for pitch estimation have been proposed in both frequency and time domain. In SMS, the fundamental frequency uses a similar method to the one proposed by Mahler and Beauchamp [37]. In the Two-Way Mismatch 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. For the implementation in SMS, the following modifications were introduced: pitch dependent analysis window, optimization of F_0 candidates, and optimization of the Peak Selection. A detailed description of this procedure can be found in [8].

The output data provided by the SMS analysis consists of the frequencies and amplitudes of the sinusoidal partials, plus the spectral shape of the residual component. Although, we overviewed here shortly the fundamentals of the Spectral Model's analysis, we address you to the references [56], [64] for further details.

2.1.4.2 Modified Phase-Locked Vocoder

The Phase Vocoder was first developed by Flanagan [19] in 1966 at Bell Laboratories, and was used for speech processing. Later, in its FFT form was brought to musical applications by J.A. Moore and M. Dolson. Primarily, this technique was used for modifying the time scale or the pitch of incoming voice or monophonic signals. A problem that arouse with the Phase-Vocoder algorithm is the loss of presence, or artificial reverberation after processing the sound. In order to overcome this, M. Puckette proposed the Phase-Locked Vocoder [46], which introduces phase-locking between adjacent pairs of FFT channels. Puckette's article gives a more thorough description about this technique.

A further step in the Phase-vocoder development is the new algorithm proposed by J. Laroche in [34]. This technique allows direct manipulation of the signal in the frequency domain. Some of the application are pitch-shifting, chorusing and harmonizing. The underlying idea behind the algorithm is to identify peaks in the Sort-Time Fourier Transform, and translate them to new arbitrary frequencies of the spectrum. Here, the spectrum is divided in several regions around peaks.

This technique was integrated in the *Spectral Peak Processing* [4] framework by Bonada and Loscos, which can be considered as a further step of the sinusoidal plus residual approach. It performs a frame based spectral analysis of the audio, giving as output of the STFT, the harmonic peaks and the pitch. Opposite to Laroche's approach, here the spectral peaks are calculated by parabolic approximation [56]. For the pitch detection, the same technique as in the SMS Analysis is used [8]. Basically, the SPP considers the spectrum as a set of regions, each of which belongs to one harmonic peaks and its surroundings. The main goal of such technique is to preserve the local convolution of the analysis window after transposition and equalization transformations. In SPP, any transformation applies on all bins within an harmonic region. Common transformations include pitch-shift and equalization (*timbre modification*). A block diagram of the overall process of SPP Analysis is depicted in the figure 2.11. Compared to the SMS model, it maintains more faithfully the characteristics of the original sound, overcoming the problem of the "too sinusoidal" sound usually found in SMS.

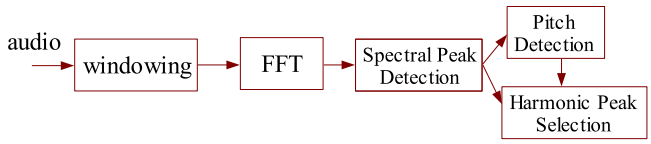


Figure 2.11: SPP Analysis process.

Note that this technique will be adopted as a analysis stage in our final implemented system, as we describe in the section 4. In addition, SPP Transformation/Synthesis framework is also used when the *Morphing* synthesis is applied.

2.2 Control of electronic instruments

The origins of electronic music can be traced back to the end of the 19th. Century with the first electromagnetic systems that generated sound. During the 20th. century some developments acquired relevance, such as the Hammond organs in the mid 1950's, which used Tone-Wheels to generate multiple harmonics.

The discovery of the transistor popularized the analog synthesizers, being Robert Moog's synthesizer the most venerated one. The next historical step is the birth of the digital synthesis with the appearance of the first computers. This fact supposed the emergence of a new research field: Computer Music. Electronic instruments have evolved rapidly, being now well accepted both musically and socially.

It is not our purpose to review here the evolution of the electronic music, instead, we discuss the control of interfaces of some electronic instruments.

With the introduction of electronic instruments, the synthesis layer became dissociated from the control layer. This fact turned out to be a very "exciting" issue addressed in the design of electronic instruments, since it gave a radically new perspective for building new musical instruments. Actually, one might assume that the notion of musical controller appears with the introduction of the first MIDI and digital controller and synthesizers.

In acoustic instruments, the designer was confined with several limitations, such as the physical dimensions of the sound generation mechanism, and the way the performer controls the instrument. Moreover, this control should allow some kind of expertise in order to create an experience musically interesting.

On the other hand, *Digital Lutherie* - term employed by Sergi Jordà for referring to the design of digital instruments - has explored multiple alternatives as input devices. Still far from being a closed issue, new control interfaces are continuously developed. Nowadays, these new interfaces are closely tied to the research in the *Human Computer Interaction (HCI)* field.

In this section, we pursue to give an overview of some key ideas, topics and main achievements within this field. Additionally, an extensively review of *musical input devices* can be found in [49], where a list with more than forty different musical controllers is presented.

2.2.1 Control of acoustic instruments

Before going into detail in the control of electronic musical instruments, we should take a look at the 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/plucked. A common fact in all acoustic instruments is that in all such cases the musician's energy is transformed into the output's sound energy [24]. Even in a piano, in which the interface (keyboard) is separated from the producing mechanism (strings), is the user's energy that produces the sound. Apart from putting

energy into the system, the performer controls the sound interacting with the sound production mechanism, as well. 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.

The most notable exception in traditional instruments is the organ, which has in many ways resemblances with electronic instruments. Firstly, the user's energy does not create the produced sound. Instead the player just control a keyboard and pedals that act as switches for several valves. The actual produced sound energy comes from the air generated by bellows that goes through the pipes. In the case of the organ, the interface is isolated from the sound producing system. Thus, actually, organs could have been build having a completely different user interface, maybe more innovative the omnipresent keyboard.

2.2.2 Regarding MIDI protocol

Despite its impact on the world of music in the last decades, MIDI is not a musical language, nor does not directly describe musical sounds. Rather, it is a communication protocol that permits the exchange of musical information by means of control signals. Actually, it was motivated by an agreement among manufacturer of music equipment, computers and software. The full *MIDI 1.0 Detailed Specification* was first released in 1983 [2].

MIDI brought a large number of possibilities to the electronic music. It separated the input device from the sound generator. This separation means that any input device (keyboard, breath controller, etc.) can control a synthesizer. Another important point is that device-independent musical software is easier to develop, since a MIDI application can work jointly with a bunch of different synthesizers.

The MIDI specification describes a language of message sent from device to device. A MIDI Message comprises one or more 10-bit *words*. For signaling the start of a musical event, MIDI uses the Note-On Message, which contains also the pitch information stored in 7 bits. That means that the pitch range extends over 128 notes, numbered from key 0 (C0) to the key 127 (G10). We find two categories in which MIDI Messages can fall: *Channel messages* and *System messages*. Channel messages target a specific channel. MIDI protocol supports up to 16 channels, which can be controlled differently. In contrast, System Messages refer to the whole system regardless of the channel. Usually these messages correspond to synchronization tasks.

As we have mentioned, the MIDI data does not provide any information about the actual sound(instrument, timbre, etc.); thus this 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. For example, selecting a given timbre (trumpet), it is expected that all instrument compliant with General MIDI produce a trumpet-like sound. Obviously, that does not guarantee in any case that what a manufacturer provides under

the *Trumpet* label, sounds actually as a trumpet.

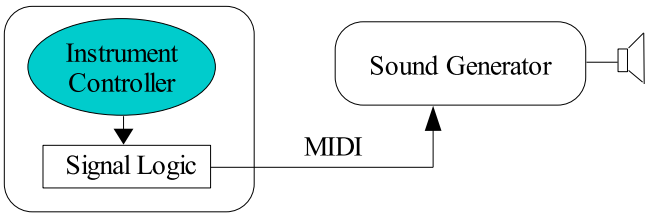


Figure 2.12: Standard MIDI controller/synthesizer system.

Timing is another issue addressed by the MIDI specification. It can be done by means of MIDI Clock Messages or MIDI Timecode. MIDI Clock allows synchronization with a sequencer by sending tempo information (song position). MIDI Timecode targets the film and video studios, and helped to synchronize music and images by sending absolute time signals (measured in hour, minutes, seconds, etc.).

Moreover, the MIDI protocol defines a specific format for interchanging data in files, the called *Standard MIDI Files (SMF)*. This let the user to store MIDI data from any input device, and later to interchange it with any software application available (e.g. sequencer).

In the context of this work, MIDI appears potentially as an appropriate signaling between our analysis and synthesis stages, thus being our *Mapping* protocol. Rather, MIDI presents some limitations that with the current technologies do not seem the best choice. As Curtis Roads points out in [49], 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 with a heavy use of continuous controllers such as foot pedals, or a breath controllers. The bandwidth would be for us a major limitation, since our system works on a frame by frame basis (usually at a rate of 172 frames per second).

The second constrain of the MIDI specification is the musical representation, since it was primarily conceived for equal tempered popular songs played on a musical keyboard. Digital Synthesis is now far beyond these boundaries, and therefore a new representation would be desirable. Probably, the main problem is the lack of timbre representation. Even with General MIDI, the timbre representation is minimal compared to the wide range of possible timbres found in the computer music. More detailed information on the MIDI can be found in [52], [2] and [49].

2.2.3 Traditional design in musical controllers

The reasons for building electronic musical instruments after traditional ones are several. First, it permits a musician to start playing it immediately, since hoe or she is already familiar with the instrument, and can apply his skills. In some cases, these electronic instruments try, at the same time, to imitate the original sound (e.g. electronic pianos). Another reason, although less scientific, is the potentially higher commercial success. We give an overview of two electronic musical controllers designed after traditional instruments: Embedded Keyboards and Breath Controllers. It is worth to mention that some of these controllers can be considered as *extended controllers*, when they are equipped with the add-on of extra sensors that extend the music control possibilities of the traditional instrument. In this category, we should include Tod Machover's *Hyperinstruments*[35], with the goal of building expanded instruments to give extra power to virtuoso performers. Some of the developments include percussion, keyboard and string instruments.

2.2.3.1 Embedded keyboards

As we have presented, playing a digital real-time musical instrument requires an input device, also called *musical controller*. There are certainly a large amount of different controllers, but without any doubt, most of the digital instruments have a keyboard. This fact was accentuated with the popularization of the MIDI 1.0 Specification. Above, we have already pointed that MIDI was designed with a keyboard performance in mind.

An electronic keyboard is the natural extension of an organ console. It is basically an array of on/off switches that indicate which notes are pressed. With the introduction of the MIDI standard, and the development of more sophisticated devices, the keyboards send also additional information about the performance, such as *note velocity* or *aftertouch*. This parameters are gathered by means of sensors placed along the keyboard.

In addition, some keyboards are equipped with other sensors in order to extent 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-beam. A list with the most common parameters sent by a keyboard follows:

- Note On/Off
- Note Velocity
- Aftertouch
- Pitch Bend

2.2.3.2 Breath controllers

With the emergence of breath controllers, wind players were able to use synthesizers. In the 70's, appeared the first devices that could capture the physical

gestures and send it to a synthesis engine. Later, in late 80's, Yamaha introduced the WX-7, the first breath controller to output MIDI, with a fingering layout close to a saxophone.

The breath controllers captures the player's gestures, that is, the state of the valves are replaced by switches, and the reed by breath and bite sensors.

Usually, these devices have aided researchers in the control of new Physical Modelling algorithms of wind instruments and other computer-based synthesis algorithms [53].

2.2.4 Innovative design in musical controllers

New Digital Music Controllers can be classified in many different ways. In the category of *innovative design*, we consider only Gestural Interfaces, which strive to capture the performer's gestures, either with the hands or with the entire body (dancers). It is worth to mention, though, that other non-gestural controllers such as Donald Buchla's [5] interface should also be included into a category of *innovative* controllers. In Buchla's designs, keyboards were replaced by pressure- and position-sensing touch-plates. Here, we describe briefly three different technological approaches towards the design of new musical instruments. An advantage of these innovative input devices is that they can exploit the possibilities of the digital music synthesis, since they are not bound up with any performance limitation.

2.2.4.1 Theremin

Although it was invented by Leon Theremin back in the 1920's, this wonderful instrument first acquired notable popularity in the decade of the 1960's. Conceptually, it can be considered as a new instrument, and not only as a controllers, since it generates the sound. It was the first musical instrument that could be played without physical contact. Pitch and amplitude were controlled by moving the hands through space over sensor plates and oscillators.

2.2.4.2 Body-Sensors

Sensing technology has brought a bunch of analog to MIDI converters, capable of transforming any physical gesture into a MIDI message. These interfaces have been widely used often in conjunction with software such as Max/MSP or Pd. Commonly these devices are found in real-time performances or interactive installations, some popular names are: Steim's Sensorlab, Infuncion System's ICube, Ircam's AtomicPro, and finally, La-Kitchen's Toaster and Kroonde.

A recurrent goal in gestural controllers is to provide ways of converting dancing events into music. In this sense, several *wearable* interfaces have been developed. A body suit incorporates several position and motion sensors that track the movement of the performer. Other applications of sensor interfaces is data-acquisition of musical expression. In the MIT's *Conductor Jacket* [38],

they extract parameters in order to describe the emotional state of an orchestra conductor.

2.2.4.3 Computer Vision - HCI

More recently, Computer Vision (CV), within the Human Computer Interaction (HCI) field, has also been employed as input device for musical performances, that is, using a camera for capturing body's gestures. Computer Vision consists basically of two parts. First, a camera, nothing else than a high-resolution light sensor; and second, image processing algorithms that extract some useful information data from the captured scene. Such a scenario is indicated for systems that allow to play with the whole body, like dancers. In this cases, the performer is freed of controllers, cables and sensors, thus, favoring the expressivity.

Certainly a very popular CV system for real-time musical performances has been the *Eyes Web*, developed by Antonio Camurri's group [7]. With the application, the user can program patches " *la Max*" 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 [1]. A set of built-in functions include several image and audio processing algorithms. Commonly, Eyesweb has been found in dancing performances for converting dancer's movements into sound.

On the other hand, Computer Vision has been used in the design of novel digital instruments such as the *reacTable** [31] developed at the UPF's Audiovisual Institute. This project under development leaded by Sergi Jordà, consists of a flat surface (table), upon which several objects are spread. It employs Computer Vision and Tangible Interfaces technologies, and aims at creating an instrument that is satisfactory for both kind of potential users: novice and trained musicians. One or more users move the objects, then , these actions are captured by the Computer Vision system that maps this data to the sound synthesis engine. The sound generation part is developed with Pd *(PureData)*.

[1] Open Sound Control. http://www.cnmat.berkeley.edu/OpenSoundControl/

2.3 Synthesis Control

In the previous section, we have presented an overview of acoustical and electronic musical instruments from the perspective of control. We have seen that particularly in acoustical instruments the performer's interaction with the instrument is done directly over the sound generation mechanism (e.g bowing a violin string). After the introduction of the electronic instruments first, and the posterior digital instruments, this tight relationship between *control* and *sound production* is no longer valid. Focusing now only on digital instruments, we can differentiate two layers *a)* user's interface and *b)* sound production or synthesis.

A definition for *Digital Instrument* will not provide us with much information about either the actual sound or about physical shape of the device. Instead, it only indicates that some step within the sound production's chain is done by manipulating 1's and 0's. Leaving apart philosophical discussions about digital signals, our interest in Digital Instruments is motivated by the broad range of techniques, sounds and devices that are involved. A general overview of different digital synthesis techniques can be found in [58]. In this article, Julius O. Smith presents a complete taxonomy for digital synthesis (see figure 2.13).

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 2.13: J. Smith's taxonomy of digital synthesis techniques [58].

In this section, we survey different synthesis techniques and their interface layer. More precisely, we will identify which parameters of the algorithms are controllable by the musician. The interface layer, known as *Mapping* differs obviously among the presented techniques, but also among the characteristics of the synthesized sound. We discuss first some aspects and trends of the mapping in the design of electronic instruments. Next, we review the most common controls for Abstract Techniques, Physical Models, Sample-Based and Morphing. However, we will not cover here controls for specific instrument sounds though.

2.3.1 Mapping layer

The playing interface in 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 has to be defined. The connection of these, traditionally inseparable, component in a real-time musical system is known as *mapping*. The mapping layer is very important [25] and in fact non-trivial, and 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 (Wanderley [62], Metois [40]) 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. A very promising approach is to model a mapping consisting of multiple layers. In the figure 2.14, 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.

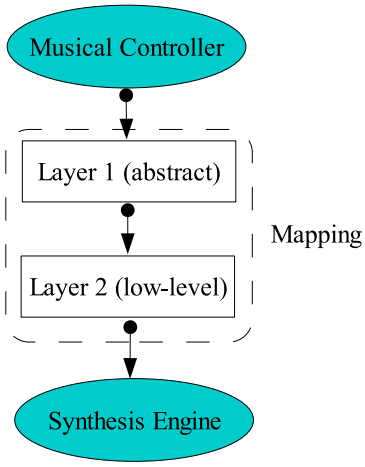


Figure 2.14: Block diagram of a two layer mapping.

As we will cover later (section 3.3 of this work), the multi-layer mapping approach is very well adapted to our goals. We pursuit to find a general mapping model for using the human voice, as musical controller, with different synthesis algorithms.

2.3.2 Abstract techniques

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

If we trace back to the origins of electronic music, we find that the first techniques employes for generating sound electronically were *additive* and *subtractive* synthesis. By the additive synthesis, a particular timbre is achieved by summing multiple sinusoidals of different amplitude and certain 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.

Opposite to the additive synthesis, there is the subtractive synthesis, which also allows a broad range of timbre. Here, a single highly harmonic 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. Looking at the spectrum of the generated signal, we observe that the harmonic components extend over the whole frequency range. Given one oscillator with a particular fundamental frequency, the next step is to generate a variety of timbres. To do so, the signal passes through filters that *subtract* energy from the original signal, and thus modifying the spectral information.

Probably the most referenced example of abstract technique is the FM[2]. Besides FM, we may find other techniques such as Granular or Waveshaping. FM produces a wide variety of complex timbres by rapidly varying the frequency of one oscillator in proportion to the amplitude of another oscillator. 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 2.11. 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.

$$x_{out}(t) = A(t)cos(\psi(t)) \tag{2.10}$$

$$\psi(t) = 2\pi f_c + I(t)cos(2\pi f_m t + \phi_m) + \phi_c \tag{2.11}$$

This technique was indeed a major achievement, having at the same time a lot of success in the commercial market when it was implemented in the massively sold Yamaha's DX7 Synthesizer.

A different and conceptually new approach is the granular synthesis. As Curtis Roads describes in [49]: " Granular Synthesis build up acoustic events

[2]Frequency Modulation synthesis, developed by Chowning in 1973 [10].

from thousands of sound grains. A sound grain lasts a brief moment (typically 1 to 100 ms), which approaches the minimum perceivable event time for duration, frequency, and amplitude discrimination."

The sonic grains are shaped by an envelope, which can take different forms. On the other hand, the content of the grain, i.e. the waveform inside the grain, can be of two types: synthetic or sampled.

In terms of control, the 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. The higher-level layer should generate automatically parameters to generate thousands of grains. Among various approaches developed to overcome this problem, we find the *Asynchronous clouds* that derives to the AGS (Asynchronous Granular Synthesis). AGS scatters grains in a statistical manner within a defined time-frequency region (the so-called clouds). This approach provides the following parameters for controlling the generated sound:

- Cloud's start time and duration.
- Grain duration.
- Density of grain per second.
- Bandwidth of the cloud.
- Amplitude envelope of the cloud.
- Waveform within the grain.

A practical application of the granular synthesis is the flexibility in the generation of sound textures such as explosions, rain, breaking glass, and the crushing of rocks, to name a few.

Finally, the third abstract technique is the Waveshaping Synthesis, first carried out by Jean-Claude Risset in 1969. Like FM this is musically interesting because of the wide ranges of possible timbres 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 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 [1], 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.

2.3.3 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 appeared the first predigital sampling synthesizer (Mellotron), it was many years later that this technique became very popular, primarily due to the lower prices of the memory-chips and to the many libraries of sounds available.

A major problem with the sampling synthesis technique, which tries to imitate existing instruments, is its lack of the so-called "prosodic rules" for musical phrasing. Individual notes may sound like realistic reproductions of traditional instrument tones. But when these tones are played in sequence, the nuances of the note-to-note transitions are missing. Partially, this problem is due to the little flexibility of the controllers. Most of the commercial synthesizers are keyboards that have very few controls, i.e. note on/off, pitch and velocity.

To summarize, the sampling synthesis technique can be well suited for certain experimental music. But for a musician, in terms of expression it is very far from the experience of playing an acoustic instrument.

2.3.4 Physical Modeling

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 mechanic-acoustic process. In other words, instead of trying to reproduce a waveform directly (as in the sampling synthesis), the Physical Modeling simulates first the producing 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. In any case, Yamaha VL1 included PhM algorithms, though.

Apart from recreating faithfully existing instruments, the PhM opens new ways for "unreal" sounds; for example, once we have simulated a cymbal, we can generate a sound of a cymbal with a diameter of 25 m.

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 filter and non-linear elements. These delay lines, also referred as *Digital Waveguides* [59], 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

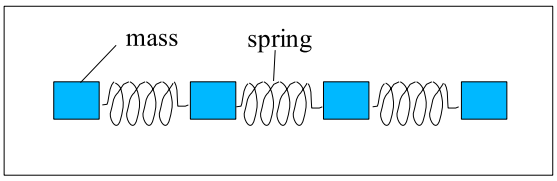


Figure 2.15: Simple Mass-Spring model for vibrating strings.

block is a pair of digital delay lines. This pair represents a wavefront travelling up and down the waveguide, due to the reflections. Basically, we have two waves travelling in opposite direction, which causes a resonances and interferences. Figure 2.16 depicts a general model for a waveguide instrument model that can simulate string and woodwind instruments. It is important to mention that Waveguide Synthesis is an efficient method that allows implementations in real-time.

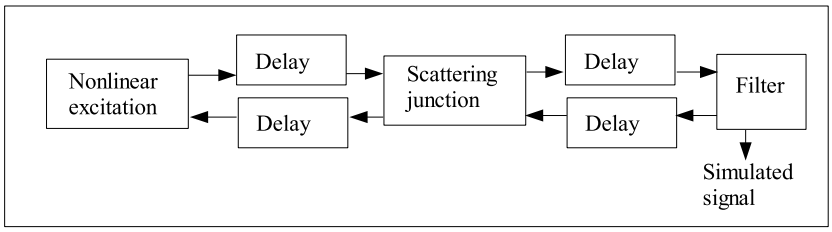


Figure 2.16: Generic waveguide instrument model. [49]

As we have mentioned, in the section 4.2.1 we describe the implementation of a virtual bass based on physical models. We use the Karplus-Strong algorithm, which simulates the sound of plucked-string instruments such as guitars, mandolins and harpsichords.

A drawback of physical models 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 very 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 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.

Finally, within the scope of Physical Modeling Synthesis we find also other algorithms that attain a great realism in generating non-musical sounds. A good example is the work achieved by Davide Rocchesso and others [50], who investigated the synthesis of the so-called “everyday contact sounds” by using

physical modeling. The models presented, impact and friction, are versatile enough to represent a wide range of contact sounds. For instance, the impact model can be tuned to render the material and size of the resonators. There are several ecological auditory phenomena, such as bouncing, breaking, rolling, crumpling are based on specific temporal compositions of impacts.

2.3.5 Spectral Models

Spectral Models attempts to characterize the sound in the frequency domain, which is more related to how our hearing system works. Roughly speaking, what Physical Models is to the sound generation mechanism, Spectral Models is to the perception organs. This approach decomposes the sound in two parts: deterministic and stochastic [56]. 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 also 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 2.17, the spectral synthesis needs various input parameters: sine frequencies, sine magnitudes, sine phases, and residual spectral data. Therefore, the next step is to see how all these data is generated, and how we modify it in order to attain particular sound transformation.

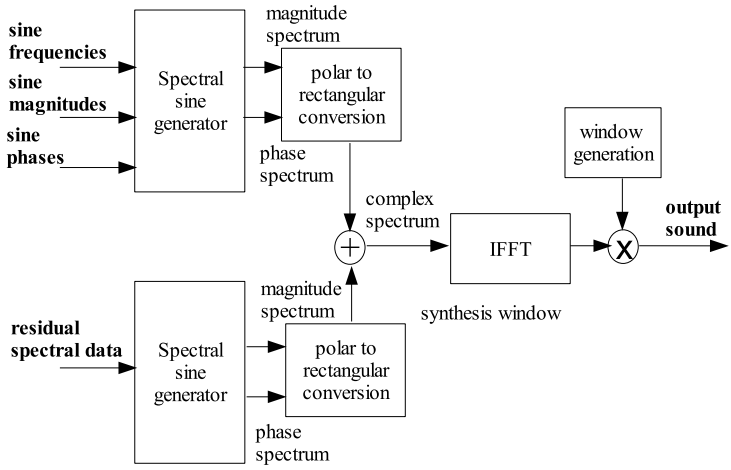


Figure 2.17: Block diagram of the spectral synthesis. [64]

The most remarkable advantage of Spectral Models over other synthesis techniques (e.g. Physical Models) is the fact that a number of musically meaningful parameters are closely related to spectral processing algorithms.

In the next paragraphs we describe *SALTO* [22], a practical implementation of Spectral Models for synthesizing a saxophone. Using Deterministic plus Residual components, it aims at reproducing a saxophone with a high degree of flexibility by using either a Breath-Controller or a Keyboard via MIDI. 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, SMS Spectral Analysis Data, consists of a complete frequency representation of the sound according to the Sinusoidal plus Residual Model.

SALTO's work-flow is the following. The MIDI Controller sends data to the input stage, which converts it to a *Input Timbre Vector* containing information about the pitch, loudness and attack. With this information one of the spectral segments of the database is selected. In the synthesis stage, the stored data provides frequency, magnitude and phase evolution for each sinusoidal track, plus the residual information, which generates a Synthesis frame. The next step consists of spectral transformations such as pitch shift or timbre morph. The timbre morph is achieved by interpolating the sinusoidal component of two Timbre Templates, and thus, following the performer's behavior in real-time. For instance, if the breath pressure increases (in case of using a breath controller), the synthesized spectrum should sound "brighter". The database holds about 72 samples in a range of two octaves at a semitone distance with three degrees of attack.

Additionally, other considerations are taken into account for synthesizing a single note. This process consists of three regions: attack, stationary and release. During the attack, the synthesis just takes the original information (Sinusoidal plus Residual) from the templates are re-synthesize the sound without any modification in order to preserve the quality of the note attack. In the stationary stage, it is possible to morph between two Timbre Templates, and the frames are continuously looped around two loop points. Finally, in the release stage, the frames are fade out within three frames in order to avoid clicks.

A key feature of this technique compared to the classical sample-based synthesis, is the modeling of the note transitions. Here, the *legato* can be achieved with certain realism by interpolating in a appropriate way the sinusoidal and residual components. Regarding the input MIDI data, it consists of two groups of information: *initial controls* (pitch, attack velocity, note on/off, and external controller data); and *continuous controls* (breath speed, and lip pressure). This information is mapped in a further step to the synthesis control parameters, as shown in the table 2.1.

In terms of control interface, this implementation does not propose nay new input device. Instead, it uses commercial MIDI Controllers, the Yamaha WX7 Breath Controller, which inherits the bandwidth limitations of the MIDI protocol.

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 [22]

2.4 Overview of related systems

This section is about various existing systems that are conceptually similar to this thesis' work. That is, controlling sound synthesis with the signing voice.

Historically, we can consider that the first attempt in using the voice for synthesizing 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 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. Long time after the analog vocoder, appeared the *Phase-Vocoder* [16]. This method based on the STFT[3] 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. synthesizer sound). It is based on the fact that we can multiply, point by point, the magnitude spectrum of two analyzed signals.

In any case, in this section we look at implementations that exploits the features of the voice to generate and control completely different sounds. We separated this section further into three parts. Firstly, two different interactive systems are reviewed. The *Singing Tree* is a public installation; on the other hand, *Auracle* combines voice interaction with collaborative music generation over the Internet.

A second type of systems refer to the most popular protocol for electronic music: MIDI[4]. Here, the goal is to convert an input audio stream, in our case *voice*, into musical events in form of MIDI messages. The main interest of these system is the capability of sending the extracted MIDI data to any compatible synthesizer module.

Finally, the third part is about synthesizers that are driven by an audio stream. Instead of having a physical interface (e.g. a keyboard), these two

[3]Short-Time Fourier Transform
[4]Musical Instrument Digital Interface

approaches analyze the input signal, extract some parameters and map them to the synthesizer's engine. As we will see, each system uses a completely different synthesis technique, and thus has a distinctive mapping strategy.

2.4.1 Voice-driven interactive systems

Human Computer Interaction (HCI) is a very challenging research field that involves many other disciplines. The evolution of the technology has brought, at the same time, various devices that capture human gestures. In a broad sense, *caption* can be either a mouse-click or a waving hand associated to a computer vision system. Although we review here audio system, we may find also research related to voice-interaction in the video field such as a face synthesizer driven by voice [23].

In the context of musical interactive systems, it is clear that an easy way for an individual to produce *music* is singing. Therefore, it seems convenient in an interactive environment to use the voice as input stream. Moreover, the human voice contains a lot of information, which can be analyzed and, in a further step, used for a new musical experience. Here, we describe briefly two systems that use the voice as interface for generating dynamic auditory scenes.

2.4.1.1 The 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*, which is related, to some extent, to the principles of this work: using the nuances of the voice for controlling the synthesis. The Singing Tree has only one microphone as input interface, an LCD screen and stereo headphones (plus speakers) as output device. Primarily, the quality of the voice is analyzed and a video sequence and a musical sequence are perturbed accordingly. The participant sings a pitch and, while singing, hears music and watches a video. In this case, the user's goal is to maintain a steady pitch, and the degree in achieving this goal is represented in both the auditory and visual feedback.

The Singing Tree has an unique video stream, consisting of different scenes. Each stream starts in an inactive state, for instance a sleeping human face. When the participant starts singing a steady pitch, the video 'wakes' and proceeds to an identifiable goal; for example, an eye opens and one zooms inside the eye to find a dancer spinning.

Regarding the auditory stream, this is also reward oriented. When the user proceeds to the goal, a beautiful and angelical music is generated, accompanied by vocal chorus in harmony with the singer. Otherwise, deviations produce an increase of dissonances, percussive sounds, generating a more chaotic auditory scene.

An issue of big interest for us is the mapping strategy. Here, the mapping is direct, as long as the user maintain a steady pitch, the video and musical sequences proceed towards 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. As they remark in an article on the Singing Tree [43], the mapping, from an artistic point of view, should be intuitive, while being at the same time, subtle and complicated enough to avoid a deterministic impression. The amplitude of the singer's is mapped to instrument's volume, and to a lesser extent to note density. Deviations of pitch were mapped primarily to note density, instrument choice, tonal coherence and rhythm consistency. Changes in the vowel's formants were analyzed and mapped to the scale on which the musical response is based.

Focusing on the voice's analysis, various vocal parameters are extracted by using Eric Metois' DSP Toolkit [40]. The pitch is analyzed in time domain. Additionally, the amplitude spectrum is also found in the frequency domain. For estimating the vowel's formants, a cepstrally smoothed spectrum analysis is achieved. Although the algorithm was not very reliable estimating the first three formant frequencies, it was very good in detecting change in the formant structure. Therefore, as they explain in [43], this was adequate for the type of mapping used. The Singing Tree integrates a music composition software as well. The mappings connecting the vocal parameters to the composition system (random generation engine) were based on fuzzy logic. This technique helps the system of being too deterministic.

The Brain Opera is currently installed at the House of Music in Vienna, Austria.

2.4.1.2 Akademie Schloss Solitude's *Auracle*

A more recent approach for using singing voice in an interactive environment is the *Auracle* system [60], developed at the Akademie Schloss in Stuttgart, Germany. In this case, instead of an interactive installation, they built a network-based instrument, which considers the participations of multiple users simultaneously. Indeed, Auracle is a group instrument controlled by the voice that works over the Internet. It has its origin in the interest of Max Neuhaus in participatory musical events. Back in the 60's and 70's, he created the radio-based *Broadcast Works*, realized over radio and telephone. Since collaboration at a distance requires a network, Auracle substitutes the prior broadcasting network by the Internet. We find a precise definition of the Auracle System in [6]:

"... an instrument that creates music by analyzing user's voices, transmitting the analysis information across the Internet to every participant, merging together this data from all participants, and using that to drive sound synthesis."

It is implemented in Java programming language and it runs in a web browser as a Java Applet. The implementation uses publicly available libraries for the sound generation and data transmission. These two libraries developed by Phil Burk are JSyn (for Sound Synthesis) and TransJam (for data transmission).

The figure 2.18, shows the architecture of the whole system, composed of five distinct modules. The input signal is handled by the analysis component, producing a gesture. This gesture is further analyzed and combined with other gestures by the coalescing component. Next, the information of each client is transmitted over the Internet by the network component. The mapping component is responsible for merging the state of all participants in a single model that is used to drive the synthesis engine.

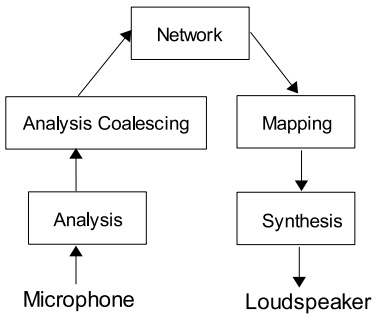


Figure 2.18: Block diagram of Auracle's architecture. [6].

For controlling the synthesis component with parameters, some characteristics of the incoming voice must be extracted. In the Input Analysis stage, the signal is downsampled to $8192Hz$ and split into blocks of $40ms$. Then, for each block some basic features of the signal are computed assuming that the signal is vocal. The considered features are: voiceness/unvoiceness, fundamental frequency, root-mean-square (RMS) amplitude and the first two formant frequencies. Concerning the formant frequencies estimation, they used Linear Prediction algorithm.

A distinctive feature of Auracle is the way it deals with the input voice in the *Coalescing Analysis* component. The voice is broken into gestures, defined as the data envelope from one block of silences to another. This leads to the extraction of higher-level descriptors for a given gesture. Furthermore, this reduces the network traffic, since the client only sends Gesture's information. Additionally, the information within a gesture is further compressed. As they suggest in [6], in practice gestures should have a limited duration due to playability and efficiency reasons. So, the limit was set to one second.

Another component that deserves our interest is the mapping. It converts gesture data to sound. In Auracle's approach there is not any constraint in the mapping from input data to generated sound, since the purpose of the system is not to recreate the original sound, but to produce interesting sounds. For instance, as input get louder, the output gets noisier. But the main issue here is the merging of the state of all participants into a single model that is used to drive synthesis. As the Auracle's team mention, the system should be transparent, that is, the user should be able to perceive his effect on the overall

sound output.

Finally, the synthesis component was totally implemented in form of a *JSyn* patch. During the prototyping, they implemented an interface protocol for using external sound engines with Auracle. The output of the mapping could be sent then to applications such as Supercollider or Max/MSP.

Regarding the user interface, Auracle differs from the typical interactive applications, since it is not controlled via mouse or keyboard. Instead, the Graphical User Interface is used to provide feedback about the state of the system to the participant. It displays information about the internal process and also about the state of other participants. The User Interface shows also a 2D projection of a Kohonen's Map, which is used in Auracle to classify input patterns.

From our point of view, Auracle appears as a very interesting approach to collaborative instruments. It attains a high degree of usability, since it uses the voice as unique input device.

Moreover, it shares some concepts with our proposed system. Discarding the collaborative architecture, we observe that the primary concept is to use the voice to control some synthesis engine. On the other hand, their approach defines the *Gesture* as the quantum of information, which consists of the voice between two silences. This leads obviously to a significant decrease of time resolution.

The main idea behind Auracle is to generate music in a collaborative way over the Internet. Thus, although it has some concepts in common, its final sonic results differs from our implementation (see chapter 4). As we will see later, our goal is to use the voice features to control the nuances of a synthesized instrument.

2.4.2 Voice to MIDI converters

Converting an audio signal to MIDI has been traditionally a challenging feature for many commercial audio software systems. We can differentiate two types of audio-to-midi conversion. Firstly, we find *Transcription* systems that try to extract the musical information from an audio stream and put it in form of a musical score. Actually, this is a research field with own entity, and goes beyond the scope of this work. Second, there are systems that use the extracted MIDI information to exploit the new possibilities for sound generation. This is in fact our case of interest, using the voice's features for synthesizing sounds. The advantages of converting an audio stream to MIDI are multiple. Basically, once we have MIDI messages, transformation on the MIDI score are cheap and will not affect the posterior audio quality. On the other hand, currently there are thousands of synthesizers available (for both hardware and software) that are driven by MIDI signals.

Although currently, there are audio-to-midi software working in real-time, the first programs ran off-line, converting basically an audio file to a MIDI file.

Many resources on the evolution of these systems can be found online at the *Electronic Musician* homepage [36].

Historically, the first Pitch-to-Midi hardware device that acquired notable popularity was the *IVL's PitchRider 4000*, which was introduced in May 1986. In 1995, the American company *Opcode* released Studio Vision Pro 3.0, its professional MIDI sequencing and digital audio recording software, which included a novel Audio-to-MIDI converter. Other applications, like *Melodyne*[5], 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. One feature of Melodyne is also to export the analyzed audio as a MIDI file.

However, note that, as we discuss in detail in the section 3.3.1, the MIDI protocol has known limitations; primarily due to the reduced bandwidth the MIDI messages.

In the next part, we review one existing software application that performs Voice-to-MIDI conversion in real-time, and therefore allowing to use the singer's voice as a musical controller. There are indeed many applications dedicated to audio-to-midi conversion. Here, we chose to review *Epinoisis' Digital Ear*, since it appears in many publications [6] as a good reference of the current state-of-the-art in similar commercial systems. However, we may find similar software systems converting audio to MIDI in real-time, such as e-Xpressor's *MIDI Voicer 2.0* (http://www.e-xpressor.com).

2.4.2.1 An example: Epinoisis' Digital Ear

Digital Ear can analyze a live or recorded solo performance (e.g. a singing voice, a saxophone solo, or any other musical instrument) and convert it into a standard MIDI file. The generated MIDI file can be imported in a general sequencer such as Cubase or Logic Audio. The input of the system can be standard audio files or a live microphone input.

The system tries to capture the nuances and expressive power of the vocalist, so that any vibrato, tremolo, pitch-bend, or portamento effects, are translated into MIDI events. It seems particularly interesting the capturing of detailed volume envelope and timbre dynamics events. These features can really boost the synthesizer's voice realism and enhancing at the same time the musical expression.

[5] http://www.celemony.com
[6] Electronic Musician magazine, Sept. 2001, Future Music magazine, Feb. 2000

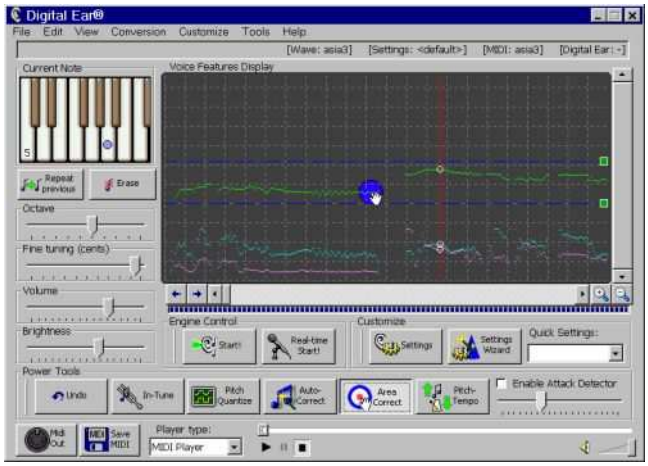


Figure 2.19: Graphical User Interface of Epinoisis' Digital Ear.

2.4.3 Audio controlled synthesizers

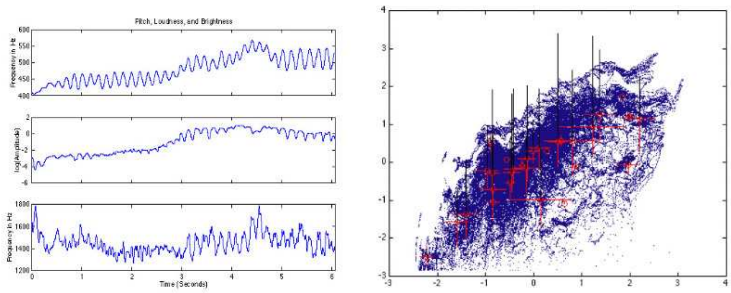
In this category, we find two systems that fairly correspond to our goals: to use singing voice to control meaningfully a synthesizer in real-time. Although, the synthesis techniques differ, these two systems share the same concept.

2.4.3.1 Audio-Driven Perceptual Timbre Synthesizer

Here we review again a system developed by the Hyperinstruments Group at the Massachusetts Institute of Technology, under the direction of Tod Machover. Probably, among all reviewed systems, this approach is the one that best matches the final goals of our work. The system was developed by Tristan Jehan, employing previous work done by Bernd Schoner [55] and Eric Metois [40]. Finally, the system's description was compiled as Master's Thesis [29].

Basically, the goal of this system is to use perceptual information of an arbitrary input audio stream for controlling a synthesis engine. The project was motivated by the need of providing a new synthesis technique for the *hyperviolin*. 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 Cluster-Weighted Modelling [55]. 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 the generate the new sound.



The specific model consists of three input parameters (pitch, loudness and brightness), and $2L$ ($= 20$ to 80) output spectral parameters, being L the number of modelled sinusoidals. The input vector is used with the predictor function on a frame by frame basis. The output vector is used then with the additive synthesis engine that generates the resulting signal in the time domain. In order to improve the sound quality, a noise component is added, following the SMS method [57]. For the noise component a second predictor is implemented, where the input vector consists of the perceptual parameters and, a nosiness estimator. In the synthesis stage, the model predictor generates polynomial coefficients, which are used to reconstruct the residual (noise) component for every frame. An Inverse-FFT transforms finally the noise spectrum to the time domain.

Although the mapping process is probably the most novel part of the system, it is not the purpose of this work to describe in detail the Cluster-Weighted Modeling (CWM). Shortly, CWM was first introduced by Neil Gershenfeld in 1999, and fully developed by Bernd Schoner in 2000. It is a probabilistic modeling algorithm based on density estimation around Gaussian kernels. More information about this technique can be found in [55].

Our impression is that this system is very powerful and contains some ideas that can be further developed in our framework. Basically, the non-linear mapping function using the Cluster-Weighted Modeling needs to be studied in detail in order to see if it is appropriate for our system. However, the presented system here lacks of generality, since it only deals with instruments with a non-quantized pitch such as singing voice, a violin, a trombone, etc.

2.4.3.2 Antares kantos 1.0

In 2002, the American company Antares presented *Kantos*, the first *Audio-Controlled Synthesizer*. It a commercial plug-in that can be integrated in most Digital Audio Workstations and work as a virtual instrument. 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

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.*

acceptable results, but other sound sources such as drum-loops can produce surprisingly interesting results, though.

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. Regarding the Source Signal, the system should work correctly with any monophonic material, with a reasonable uniform loudness for accurate pitch-tracking. The presence of background noise can be misleading for the analysis algorithms, as well. As we find in the product's documentation, it is suggested to feed *kantos* with voice or guitar sounds.

Looking at the block diagram, we find first the analysis module. It contains a function for controlling the triggering, the so-called *Gate Generator*. This function resembles a sophisticated noise gate with four parameters: Note on, Note Off, Floor and Hold.These parameters can be tuned by the user in order to adapt the *Gate Generator* to the input signal. However, finding adequate values for these parameters can be a bit tricky, nevertheless they depend on the characteristics of the input material. The generated signal is further used to trigger envelopes, to reset wavetables or for other tasks. In addition to the amplitude envelope, the system extracts the fundamental frequency and formant information, which characterizes the source's timbre. Note that, since we are here reviewing a commercial product, there is a lack of thorough descriptions about the employed techniques.

The next step is to map the extracted parameters to the synthesis engine. This process is executed by the *Modulation Matrix*, which is fully controllable by the user. As we have mentioned, the kanto's synthesis engine is quite simple. It consists actually of two wavetable oscillators, a noise generator, two filters, two LFO's and effects. The Modulation Matrix gains great interest because it allows to map the different input data to any of the synthesizers parameters. In the table 2.2, we see a list of the analysis parameters and some of the parameters to which they can be assigned. More detailed information about all possible combination in the *Modulation Matrix* can be found in *Antares Inc.* homepage [8].

In the synthesis block, we find the aforementioned elements (oscillators, filters, etc) plus a special sound processor called *Articulator*. This processor uses the harmonic and formant information detected in the analysis section, and applies that to the output of the wavetable oscillators and the noise generator. In

[8] http://www.anatrestech.com

other words, this process modifies the spectral envelope of the incoming signals. By means of one slider, the user controls the amount of applied effect.

Antare's Kantos is focused on generating traditional synthesizer sounds controlled by an audio input. We find that it is designed to be particularly effective with voice input, but with some adjustments the input section can be adapted to other sources such as guitar or drum loops.

In this work, we are specially interested in the mapping strategies employed by similar systems as Kantos. In this case, we have observed that the mapping is completely loose, since through the Modulation Matrix, the user can assign an input parameters to any of the provided synthesis parameters.

Chapter 3

Contributions

In the previous chapters, we have first presented the problem and then, provided research related to the addressed topic. The next two chapters intend to serve as a "proof of concept" for dedicate a PhD thesis to the proposed topic.

3.1 Initial Objectives

In the computer music community, most of the research related to voice had its goal in the singing voice synthesis. Alternatively, we aim at consider the voice as a musical controller, thus we focus on a particular set parameters that can be meaningfully mapped onto a synthesizer's engine. This is an ambitious project, which aims at providing ways for controlling any virtual musical instrument by the human voice. We consider that with the current state-of-the-art, there are sufficiently advanced techniques on both Singing Voice Analysis and Virtual Instruments Synthesis. So, the goal of this work is to settle a bridge between these two fields in order to generate sounds intuitively with the nuances of the human voice. From our point of view, this work covers many different areas in the field of computer music. Therefore, it seems necessary to limit the research topics, and to precise our final research goals. Consideration on the design of synthesis algorithms are beyond the scope of this work. We chose two techniques that with the current technology are capable of recreate sound with enough flexibility and realism.

The contributions presented in this work aim at filling the gap between Voice Analysis and Virtual Instrument Synthesis, as mentioned. Of course, this requires indeed a deep knowledge of both sides. From the input voice, we need to identify which characteristics are meaningful; likewise, on the synthesis engine's side, we need to set properly the algorithm's parameters. Finally, two topics appear to be meaningful to address:

1. Voice Features Extraction

2. Mapping Strategy

In the next sections, we find a thorough description of techniques that were developed within the framework of this project. Additionally, the next chapter (Chapter 4) explains the implemented system, which integrates the aforementioned techniques.

3.2 Voice Features Extraction

Our framework works in frequency domain, computing first the Short-Time Fourier Transform of the input voice. This process is inherently a block oriented process. We work in a frame-by-Frame basis, so we process the input signal at a particular frame rate, usually 172 frames/s. The next task is then to analyze input's spectrum and extract valuable features of the voice for each frame. In this section we describe a proposed set of voice descriptors, separated in different categories: excitation, vocal tract, vocal quality (disorders), and note.

Our research consisted in two parts. First, identify relevant features to be extracted and to choose the appropriate techniques, and second and more challenging, to develop novel algorithms for those unavailable.

3.2.1 Excitation Descriptors

When we listen carefully to a musical instrument playing some notes, we acquire intuitively some information. Apart from identifying the instrument, we perceive a note with a particular pitch, loudness and brightness. On the other hand, we agree that a musician controls the pitch, loudness and brightness of his or her instrument. Therefore, we can consider these descriptors to be of high relevance.

Although, we present the *Excitation Descriptors* in this chapter, it is necessary to mention that none of the extraction algorithms are new. Despite this fact, from our point of view, the contribution rests here on the selection of useful features and techniques.

3.2.1.1 Pitch information

Estimating the pitch from the voice might be a difficult task, since the pitch may present rapid and constant changes. On top of that, during the note attack (note onset), the signal may have a instantaneous pitch far away from the final stabilized pitch. Despite this, several methods attempted to extract the pitch from the voice. The first techniques were developed for speech processing [48], but we employ here a method primarily focused on the singing voice.

In our implementation, we used the Spectral Analysis framework developed at the IUA during the last years. It includes a Pitch Estimation method based on the Two-Way-Mismatch technique [8], as we have reviewed in the section 2.1.4. The pitch descriptor serves at the same time as *voiced/unvoiced* gate. When

the pitch takes the value 0, it means that the analyzed frame was unvoiced.

In addition, the pitch deviations might have also importance for controlling certain parameters of the synthesis engine. Hence, we include the *Pitch Gradient* as a complementary descriptor of the pitch. We calculate it as the variation from the last frame's estimated fundamental frequency, and the current one.

$$\Delta pitch = |\hat{f}_0[k] - \hat{f}_0[k-1]| \tag{3.1}$$

3.2.1.2 Loudness information

A good estimator of the loudness is the RMS (root-mean-square), which has been widely used in the audio industry. An approximation of the RMS in the frequency domain can be also calculated. Loudness is estimated by weighting the frequency bin k of the power spectrum by the coefficients W_k obtained from the interpolation of the Fletcher-Munson curves [29].

$$loudness = \sum_{k=2}^{N/2+1} W_k \cdot |a_k|^2 \tag{3.2}$$

In our case, the energy information is also included in the global *Energy Envelope* descriptors, explained in the page 67.

3.2.2 Vocal Tract Descriptors

By listening to a musical note, we usually identify which instrument is actually playing. The explanation is two-fold: firstly, we perceive a particular timbre; and second, we associate this particular timbre to a known instrument by means of our cultural background. Furthermore, a player modifies continuously the timbre of the instrument, adding expression to the performance. These variations are important on order to avoid a sensation of "steadiness".

It seems interesting for our work to extract timbre information from the input voice. Later we can use this attributes for modifying the timbre characteristics of the synthesized sound. In the human voice, the timbre is related to the pronounced vowel, which is determined by the formants. Hence, it will be very intuitive for the user to execute a *musical gesture* by changing from one vowel to another.

We provide here a method for estimating the formant frequencies that are necessary for identifying a vowel. In addition, a more detailed timbre's information is extracted with the spectral shape.

3.2.2.1 Formant Estimation. Vowel Vector

In the section 2.1.3, we reviewed different techniques used in Speech Processing for extracting the spectral envelope of voiced sounds. In the context of Speech Processing, LPC and Cepstral Analysis are applied to a variety of systems such

as speech recognizers or speech synthesizers. In our particular case, we intended to use these methods (*LPC and Cepstral Analysis*) for recognizing vowels. Since vowels can be approximately represented by the first two formants, our *vowel recognizer* is reduced to a formant estimator system. We observed, though, that the mentioned algorithms lacked of robustness when applied to the singing voice. This implies that a new approach needed to be developed in order to build a playable and stable system.

After analyzing our particular needs, we realized that the system could also retain its functionality with a less accurate algorithm for estimating the formant frequencies. Next, we present a new algorithm that separates the spectrum into two regions, one for the first formant and one for the second formant. This novel technique is called DCFE *Dual Centroid Formant Estimation*, and it is based on the spectral centroid calculation. The motivation for developing an alternative method, is the lack of robustness of the "peak-picking" algorithm in the two previous cases. As we have already observed in figure 2.7 and 2.10, the values of the formant frequencies are not very stable over time, even in a vowel steady-state. Notice that our goal is to use the values $\{F1, F2\}$ for controlling an external process. Therefore, these values should remain very stable within a vowel sound, even if the formant frequencies are not very accurate. Obviously, we are faced with a trade-off between stability and accuracy, but for our application keeping $\{F1, F2\}$ stable and controllable by the user is primordial.

Basically, the idea behind the algorithm is to define two regions in the spectrum corresponding to the first two formants. By observing the figure 2.5 from [45], which depicts the position of the vowels as function of the formants $Ffirst$ and $Fsecond$, we can make some assumptions. Typical values for $F1$ are between 200 and $800Hz$, and $F2$ goes from 800 to $2500Hz$. We define our *Vowel Vector* as the pair of formants frequencies $F1$ and $F2$. Next, we calculate the spectral centroid within each region, which will be directly our estimated values for $F1$ and $F2$. The Spectral Centroid is defined as:

$$f_c = \frac{\sum_{k=1}^{N} A[k]f[k]}{\sum_{k=1}^{N} A[k]} \tag{3.3}$$

where k is the spectral component, $A[k]$ is the amplitude and $f[k]$ the frequency. However, we use a slightly modified equation for calculating $F1$ and $F2$. Firstly, we consider only spectral peaks for $A[k]$ and $f[k]$; and secondly, the magnitude $A[k]$ is weighted by a "crossover-like" function $w_i[k]$ for splitting smoothly the two regions. Finally, we come up with two equations for calculating the values $F1$ and $F2$,

$$F1 = \frac{\sum_{k=1}^{N} A[k]w_1[k]f[k]}{\sum_{k=1}^{N} A[k]w_1[k]} \tag{3.4}$$

$$F2 = \frac{\sum_{k=1}^{N} A[k]w_2[k]f[k]}{\sum_{k=1}^{N} A[k]w_2[k]} \tag{3.5}$$

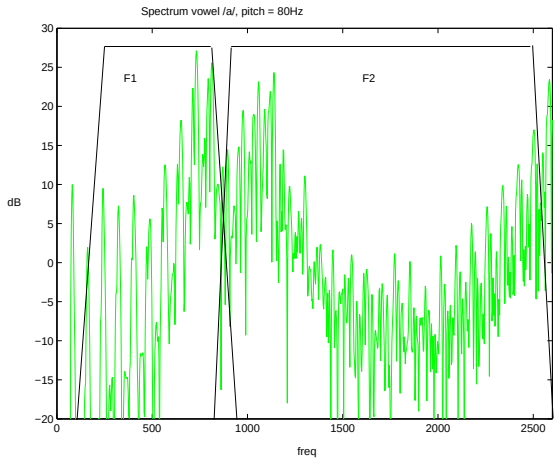


Figure 3.1: Spectrum of the vowel /a/ with $pitch = 80Hz$. Superimposed the weighting functions for finding $F1$ and $F2$.

The figure 3.1 shows the spectrum of the vowel /a/ with a pitch of $80Hz$, and superimposed, the weighting functions w_1 and w_2.

In fact, this method works pretty good for most vowel sounds, when the formants frequencies are separated and fall in the right 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 vowel /u/ and sometimes a *close* /o/. The solution appears to be quite simple. For sounds, in which the two first formants fall into the first region $[200 - 800Hz]$, the energy ratio ER_{12} in equation 3.6 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 formant frequencies $F1$ and $F2$ using two new weighting functions $w_{11}[k]$ $w_{12}[k]$.

$$ER_{12} = \frac{\sum_{k=1}^{N} A[k]w_1[k]}{\sum_{k=1}^{N} A[k]w_2[k]} \tag{3.6}$$

Finally, the *Vowel Vector* is defined as $\vec{V_f} = \{F_{first}, F_{second}\}$, which takes its values from $\{F1, F2\}$ or $\{F11, F12\}$ depending on the coefficient ER_{12}, which interpolates between the two. In addition, an averaging filter is placed at the output in order to smooth the transitions.

3.2.2.2 Brightness information

Another perceptual feature that characterizes a sound is the "richness of harmonics", namely the *brightness*. A good indicator of the instantaneous richness

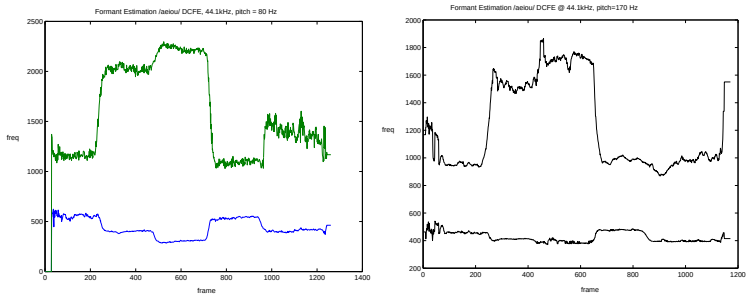


Figure 3.2: Centroid Analysis. Formant frequencies $F1$ and $F2$ of a vowel sequence /a/-/ε/-/i/-/o/-/u/ with a) $pitch = 80Hz$ b) $pitch = 170Hz$

of sound can be measured by the center of gravity of its spectrum. For a constant pitch, a sound with strong harmonics, will have a higher center of gravity. This value is also referred as the *spectral centroid* [63], or *first moment* of the spectrum. The final value for brightness is the result of the centroid expression (equation 3.7).

$$centroid = \frac{\sum_{k=2}^{N/2+1} a_k \cdot f_k}{\sum_{k=2}^{N/2+1} a_k} \tag{3.7}$$

Clearly, this result is correlated with the pitch. A way to evaluate the "harmonic richness", is to calculate the ratio of the centroid divided by the pitch.

3.2.2.3 Spectral Shape

In addition to the first two formant frequencies, as we have seen previously, the Spectral Analysis (see section 2.1.4.2) generates a more accurate description of the timbre. Using the harmonic peak detection algorithms, a 3rd-order spline-interpolation is calculated, providing a continuous envelope of the spectrum. It interpolates between data pairs of logarithmic magnitude and frequency of the selected harmonic peaks.

The generated curve can be used a posteriori for changing the timbre of the processed sound according to the input voice characteristics.

3.2.3 Voice Quality Descriptors

Before extracting features from the singing voice, it is necessary to discuss which parameters are meaningful to evaluate. We are particularly interested in parameters that are controllable by the singer (*user*). Obviously intentional vocal disorder is not the only possible expression in singing voice. Among many others there are vibrato, intonation and loudness. However, we focus here only these singing effects that can be considered as disorders.

A singers voice may have characteristics which on the one hand are described scientifically as vocal disorders, like a growl, creak, rough or hoarse voice. On the other hand, from the musical point of view, one can consider these characteristics as highly relevant expressiveness features, which creates its singers voice unique timbre. In this sense, we named this section "Voice Quality Descriptors". There are singers whose voices always have a timbre with vocal disorders, like Louis Armstrong, Janis Joplin and Tom Waits, but others control the amount of vocal disorders intentionally like Joe Cocker, Sting and Brian Adams.

Although in other disciplines such as medicine Voice Disorders are thoroughly classified, we distinguish as *Voice Quality Descriptors* only two classes of disorders. Firstly, in the larynx we find disorders due to a dysfunction of the vocal folds, which we will define as *roughness*. The second disorder considered here, relates to the air-flow that passes through the vocal folds without exciting them sufficiently. The acoustical effect of this disorder is an added noise component, known as *breathiness*. Precisely speaking, since in our case, these disorders are intentionally done by the performer, we should better use the term *effect* for referring the extracted attributes.

Here we present two novel methods for extracting these attribute in the spectral domain. Both are computed at a frame rate(usually 172 fps), that is, a new value for roughness and breathiness is provided each frame.

3.2.3.1 Hoarseness descriptor

Here, we focus on the particular case of voice disorder that provokes "hoarseness". This pathology is often referred in the medical literature as *Muscle Tension Dysphonia*. In a normal situation, the vocal folds vibrate at a certain vibrating frequency. It means that a constant *cycle* of opening and closing is periodically repeated. However, if this*cycle* does not remain constant, either in its amplitude or shape, new frequency components appear in the signal. These new components are in fact sub-harmonics of the fundamental vibrating frequency. The figure 3.3 shows clearly this effect.

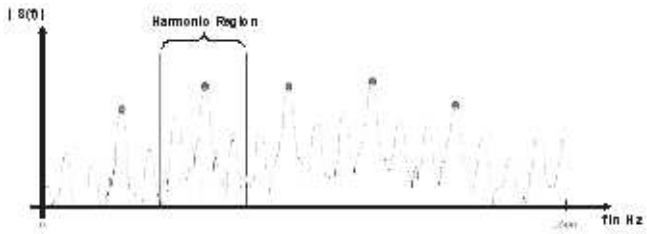


Figure 3.3: Example of vocal disorder. The spectrum (0-3500 Hz) shows clearly the presence of sub-harmonics, in addition to the harmonic partials (marked with dark circles).

Our first task is to define a method for identifying a voice with vocal disorders. In a growl or hoarse voice besides the harmonic partials additional

sub-harmonics are found. In the field of Perceptual Audio Coding appears often the idea of identifying the noisiness of an audio signal. Common methods are the Spectral Flatness Measure (SFM) and Tonality [30]. Here we use a similar concept for identifying the sub-harmonic components.

For evaluate the *Hoarseness* descriptor, we have defined the *Sub-Harmonic Factor* SHF [17]. As input data, we take values from the Spectral Analysis block: Spectral Peaks, Harmonic Peaks, Pitch and Excitation Slope. The Sub-Harmonic Factor is computed using a formula that derives from Harmonic Peaks and Spectral Peaks values, which are stored in arrays. Both arrays contain peak information of magnitude, frequency and phase. The first step is to divide the spectrum in regions around each Harmonic Peak [4]. We assume that peaks of frequency above 3.5kHz are not relevant for our estimation, thus we consider only the lower frequency range. In the figure 3.3, we observe the Harmonic Peaks, the Harmonic Region (centered on each harmonic partial), and all sub-harmonics with lower energy.

In a first trial, we attempted to compute the Sub-Harmonic Factor (SHF) using the Spectral Flatness Measure of each Harmonic Region. It is defined by Johnston [30] as the ratio between arithmetic and geometric means of the spectral power density function, and computed directly from the analyzed frame contained harmonics of high bandwidth, for instance in case of vibrato. Therefore, we developed a new approach considering for each region r, only the magnitude of the spectral peaks as valid data. In the equation 3.8, the Spectral Peaks are represented by the vector $SPeak[p]$, and the regions Harmonic Peak is represented by $HPeak_r$. We call it Region Sub-Harmonicity (RSH_r), which considers only the Spectral Peaks with index ranging from $[M_r$ to $N_r]$.

$$RSH_r = \frac{HPeak_r}{\sum_{p=M_r}^{N_r} SPeak[p]} \tag{3.8}$$

Then we calculate the Sub-Harmonic Factor (SHF) as the average of all Region Sub-Harmonicity (RSH_r) values. We only consider up to a frequency of $3.5kHz$ (Equation 3.9). Note that for frames with a high number of sub-harmonics, RSH_r tends to 0. Thus, in the final formula (Equation 3.9), SHF tends to 1.0 for a high sub-harmonicity, and is 0 in case of a signal with solely pure harmonics.

$$SHF = \frac{1}{R}\sum_{r=0}^{R-1}(1 - RSH_r) \tag{3.9}$$

Finally, our *Hoarseness Descriptor* is directly the computed SHF, filtered through a moving-average filter for smoothing the results.

3.2.3.2 Breathiness descriptor

Another commonly used singing effect is the *breathy* voice. The *breathy* voice is caused by a particular mode of phonation. The physiological explanation of

breathiness can be found in [61]. According to Sundberg, 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. Hence, our goal is to examine the harmonic content at high frequencies, and to extract a valid *breathiness* descriptors. This effect is only noticeable in the stationary part of a note, thus, in the calculation we do not take into account the transitions.

When we faced the calculation of a breathiness factor, several methodologies in the spectral domain seemed to be promising. Here, we compare three different techniques (listed below), justifying at the same time, the final implemented method.

- Harmonic-Peak Stability
- Band Spectral Flatness
- Peak-Envelope Differential Area

It is necessary to mention that all these methods take the output from the *Spectral Peak Processing*, reviewed in more detail in the section 2.1.4.2 of this work. It provides the signal's spectrum plus a list of the detected harmonic peaks, with corresponding amplitude, frequency and phase.

This method observes the trajectories of the detected harmonic peaks within a frequency range from 3 to $6kHz$. As we can see in the figure 3.4, due to the presence of noise, the harmonic peaks' frequencies suffer of great deviations in the mentioned range. For a frame k, we calculate a frequency stability factor (S_f), and phase stability factor (S_ϕ), 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 3.12, we wrap the resulting predicted phase.

$$\hat{f}_r[k] = n \cdot pitch \tag{3.10}$$

$$S_{fr}[k] = |f_r[k] - \hat{f}_r[k]| \tag{3.11}$$

$$\hat{\phi}_r[k] = \phi_r[k-1] + 2\pi \frac{f_r[k] + f_r[k-1]}{2} \Delta t \tag{3.12}$$

$$S_{\phi r}[k] = |\phi_r[k] - \hat{\phi}_r[k]| \tag{3.13}$$

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 3.16). The factor $d_{pitch}[k]$ will ensure that the final value is not affected by normal pitch variations such as in a vibrato.

$$S_{Peak,r}[k] = S_{\phi r}[k] + \frac{1}{40} S_{fr}[k] \tag{3.14}$$

$$d_{pitch}[k] = 1 - 100\,\frac{pitch[k] - pitch[k-1]}{pitch[k] + pitch[k-1]} \tag{3.15}$$

$$Breathy_{stab}[k] = d_{pitch}[k] \frac{1}{R} \sum_{r=0}^{R} S_{Peak,r}[k] \tag{3.16}$$

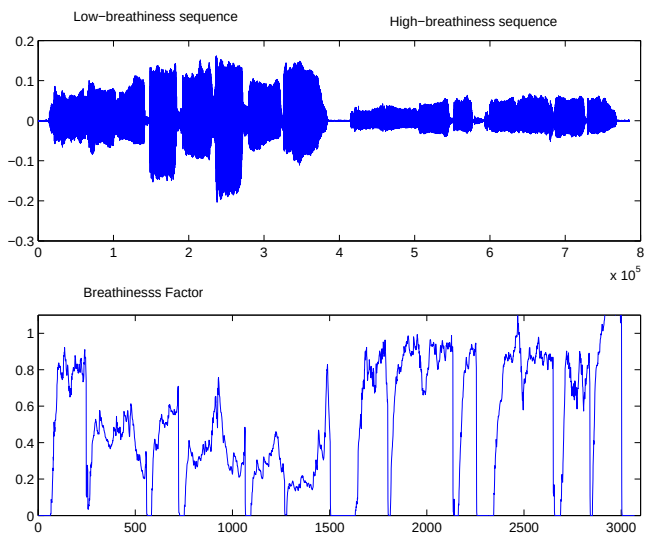


Figure 3.4: Breathiness Factor using the *Harmonic-Peak Stability* technique. The sound example is a musical scale with, first a regular, and then a breathy phonation.

The second method proposed is based on the *Spectral Flatness Measure* (SFM). It is defined by Johnston [30] 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 $2.5kHz$ to $9kHz$. Thus, in order to calculate the *Breathy Factor*, we compute the Spectral Flatness in the mentioned frequency band.

$$Breathy_{flat} = \frac{(\prod_{k=N}^{M} spec[k])^{\frac{1}{M-N}}}{\frac{1}{M-N}\sum_{k=0}^{N} spec[k]} \qquad \begin{array}{l} N = \frac{2500 FFTSize}{SampleRate} \\ M = \frac{9000 FFTSize}{SampleRate} \end{array} \tag{3.17}$$

Finally, the last technique (*Envelope Differential Area*) proposed takes also advantage of the Spectral Peak Processing. It computes the area between the harmonic peaks envelope and the actual spectrum, as depicted in the figure 3.5.

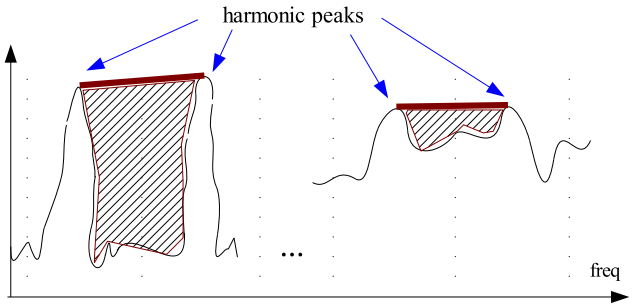


Figure 3.5: Envelope Differential Area. Linear interpolation between two harmonic peaks.

The equation 3.18 gives us the final formula for the *Breathy Factor*. In this case, experimental results showed that the best frequency boundaries were $4000Hz$ and $9000Hz$.

$$Breathy_{env} = 1 - \sum_{N}^{M} \frac{envelope[k] - spec[k]}{n\ bins} \qquad \begin{array}{l} N = \frac{4000FFTSize}{SampleRate} \\ M = \frac{9000FFTSize}{SampleRate} \end{array} \tag{3.18}$$

We tested these three methods, and some conclusions could be extracted. The first method is computationally the cheapest, since it just makes one calculation for each harmonic, and not bin by bin. Although, the three results appeared to be quite similar we found that the third method was rather influenced by the pitch. In the figures 3.6 and 3.4, a musical scale is sung twice, first in a regular way, and then with a high breathy phonation. For a real implementation, we argue that the first method (Harmonic-Peak Stability) attains good results while keeping a low computational cost.

3.2.4 Note Descriptors

Usually known in electronic synthesizers as *ADSR*(Attack, Decay, Sustain and Release), the note's energy envelope brings a lot of information about the instrument being played. It refers to the amplitude variation in time of a produced note. We can grasp better the meaning of the ADSR by observing graphically th waveform on a musical note in a sound editor. For instance, the differences between the amplitude waveform of a drum hit and a flute note seem quite straightforward. The former will have a fast impulsive attack and an exponential decay; the latter presents usually a softer attack and a long sustain.

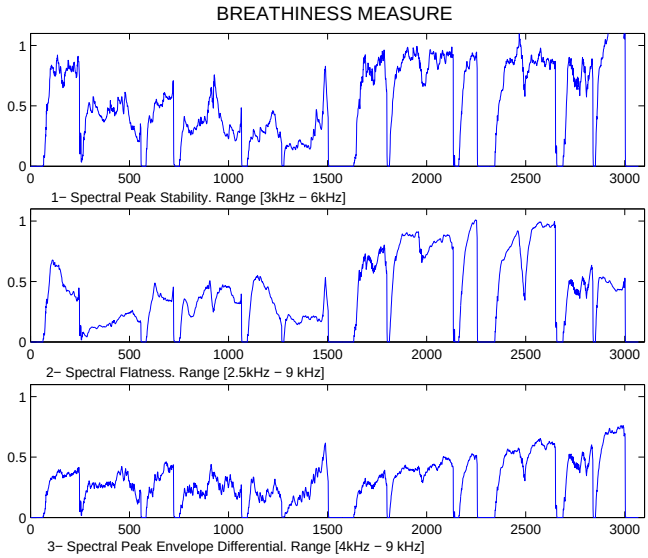


Figure 3.6: Comparative test. Top) Harmonic-Peak Stability; middle) Spectral Flatness; bottom) Envelope Differential Area.

As we have mentioned in the beginning of this section, in our framework, we process the input signal on a frame by frame basis. Therefore, we generate descriptors that are related to a particular spectral frame. In this category, we provide two attributes: *Energy Factor* and *Attack Factor*, which can be mapped onto the synthesizer's engine in order to determine the characteristics output sound. In addition, our timing description of the note envelope differs slightly from the classical ADSR approach, as we describe next.

3.2.4.1 Energy Factor

Intuitively, we expect any physical system (and musical instruments in particular) to be pseudo-linear in terms of input/output energy. If we hit a membrane with high energy, the sound energy produced will be proportionally high. Bearing it in mind, it seems convenient to track the energy parameter from the input voice.

The *Energy Factor* is calculated directly from input signal, as the sum of the sample's module, divided by the number of samples.

$$EnergyFactor = \frac{\sum_{n=0}^{N-1} abs(x[n])}{N} \tag{3.19}$$

3.2.4.2 Attack Factor

The second attribute proposed is the *Attack Factor*. Here we seek to capture the characteristics of the note's attack. In our context, we assume that a *note* consists of a voiced sound, usually a vowel, with a rather constant pitch. However, by extending the note with a consonant (unpitched) impulse preceding the sustained voiced sound, we can make use of additional information, and map it in a further step to the timbre of the synthesized sound. Hence, our model of note, will have two parts: an unpitched attack plus the actual pitched note. A practical example of the usability of this attribute is in the synthesis of string instruments; a *harsh* attack generates a plucked guitar with a pick, while if the attack is missing a soft finger-plucked guitar is synthesized.

Basically, the Attack Factor aims at extracting the harshness of the unpitched part in the note attack. It takes a block of samples, and computes the *Energy Factor* (described previously), and the *Zero-Crossing Rate*. The final formula is presented in the equation 3.20. Note that only unvoiced frames frame are considered (see section 3.2.1.1).

$$AttackFactor = \begin{cases} 0 & , pitched \\ \sqrt{EnergyFactor \cdot ZeroCrossing} & , unpitched \end{cases} \quad (3.20)$$

3.2.4.3 Timing description

Since our input signal is the singing voice, the *decay* part is hardly present. Hence, we propose a model consisting of just three regions: *Attack*, *Sustain* and*Release*. The figure 3.7 depicts the waveform of a note and the graphical representation of our model. We foresee that note's variations (vibrato, tremolo) can occur within the *sustain* region.

As we will see, the timing information is essential for the Morphing Synthesis' mapping (see section 3.3.4).

3.3 Mapping of the Extracted Features

Mapping is the other main research topic a dressed in this work. It has deserved also many studies in the computer music research community [62],[40]. In this section, we describe how the analyzed parameters are mapped to the two synthesis techniques here studied: Physical Models and Spectral Morphing. The achieved results refer to the synthesis of bass sounds, assuming, thus, that the input voice tries to emulate the target sound. This issue is of great relevance, since it will help to establish a consistent mapping, and on the other hand, it should be more intuitive for the user. For instance, in the synthesis of a plucked bass, we assume that the input is, to some extent, made of impulses, emulating the plucking process.

Our contribution in the area of mapping is to create a valid model for the voice in a two-layer mapping process. We aim at establish a set of valid voice

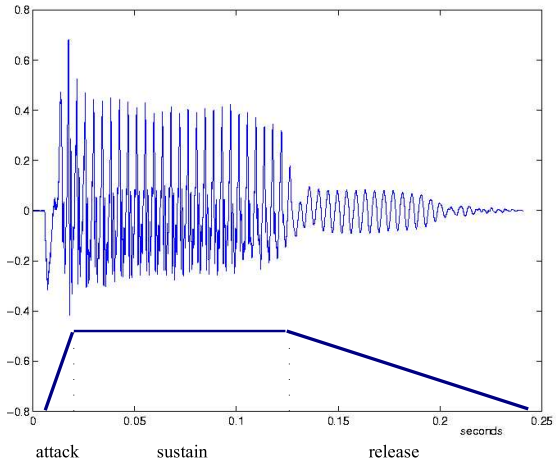


Figure 3.7: Waveform of a singing voice's note, and its note envelope model.

descriptors that are meaningful for the performer. The selection of the descriptors is a two-fold problem. On one side, they must be intuitive, and second they should transmit the expressivity of the singing voice to the synthesized sound.

We present a model for the first mapping layer, providing a set of relevant features. Additionally, a very preliminary first approach for the second mapping layer is introduced.

3.3.1 Justification for discarding MIDI

The MIDI Standard was first introduced in 1983. Since then, it has revolutionized, in many ways, the digital music synthesis. An evidence of MIDI's success, still nowadays, can be grasped by going into a musical instrument's store. Almost the totality of electronic instruments incorporate MIDI,

However, the MIDI protocol presents several limitations for controlling sound synthesis. Few years after its inception, Moore pointed out already some of these limitations [41].It is worth to note that with the evolution of the technology (digital synthesis algorithms to network topologies) in the past twenty years, this limitation have become more evident.

Maybe the most important drawback of MIDI is that it cannot preserve music's continuous, dynamic nature. This is due to basically data resolution and event frequency. Moreover, the number of assignable channels is 16 (a four bits field), making the protocol inextensible. We can identify also a lack of consistency in the system. Some functions such as pitch-bend, apply to a complete channel, making it impossible to pitch-bend a single note without affecting the rest. In contrast, in polyphony, playing several notes simultaneously results in multiple "Note On" messages, which provokes a audible delay between the first

and the last notes. Another fundamental problem is that MIDI is keyboard-centric. Non-keyboard instruments are penalized by coupling pitch, loudness and timing information. Brass or bowed instruments are not well described this way, which can lead to a saturation of MIDI messages if want to track the variations of the instrument. Although it is not of our concern, since our system consists of a software implementation, the network topology of MIDI is significantly outmoded. It uses a serial lines to daisy-chain several instruments, making the communication uni-directional, when Bi-directionality is needed.

Evidently, when we designed a system for using the singing voice as musical controller, the option of using MIDI was considered. However, apart from the inherent limitations presented above, MIDI does not adapt to our system for various reasons:

- The Singing Voice can hardly be modeled as a keyboard-oriented instrument.
- The low bandwidth restricts the amount of information sent per second that is required for our system.
- MIDI limits the data type to 7-bit integers, thus not providing ways to send arrays of floats (e.g audio samples, spectral envelopes, etc.)

In our implementation, we decided not to use any defined mapping protocol, and to postpone this issue for future studies.

3.3.2 Mapping Model for the Singing Voice

Several studies on mapping in a real-time synthesis situation, showed the convenience of a multi-layered mapping's approach [24]. In our context, we aim at presenting the voice as a generic musical controller. In other words, using the voice with a broad range of synthesis algorithms. Therefore, it is clear that by adopting a multi-layered mapping, we can separate the voice input from the synthesis engine.

In order to maintain the model as general as possible, we propose to use a *Two-Layer Mapping*, whereby the first is dedicated to voice analysis and features extraction; and the second is responsible for mapping the parameters accordingly to the synthesis engine. Our objective here is to define a general model for the first Mapping's layer, which can be reused in future research works. This model outputs a set of *abstract* parameters (musical oriented voice features such as brightness, attack harshness, pitch, loudness, etc.), which can be intuitively mapped to a generic synthesis algorithm. The figure 3.8 depicts the mapping layer.

The goal is to define a general model for the first mapping layer. Within this layer, low-level descriptors are converted to some other musically meaningful parameters. By looking at the figure 3.8, we see that the voice signal

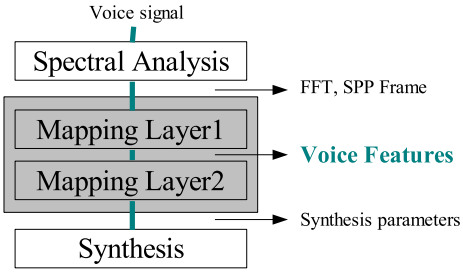


Figure 3.8: Two-layer Mapping. The first layer provides a set of abstract voice descriptors.

captured by a microphone goes into the *Spectral Analysis* stage. Here, two processes are pipelined, first, the Fast Fourier Transform (FFT) is executed, and second, the computed spectrum is then analyzed using the Spectral Peak Processing framework (SPP), see section 2.1.4.2. The result is a frame containing the spectrum plus the detected spectral peaks and other low-level descriptors (estimated pitch, frame energy, zero-crossing rate, and excitation slope). In a further step, we map this low-level data onto abstract descriptors, which give us voice information from a musical point of view. Although many parameters could be defined and extracted, we propose a fixed set of *Voice Features* that defines our model. In the table 3.1, we enumerate the input and output descriptors of the first mapping layer.

Input - *Low level Feat.*	Output - *Voice Features*
FFT Spectrum Estimated Pitch HarmonicPeaks array Frame Energy Zero-Crossing rate Excitation Slope	Pitch Pitch Gradient Loudness Brightness Timing Descriptors Breathiness Hoarseness Vowel Vector Attack Harshness

Table 3.1: Model of the first mapping layer for singing voice

By gathering the results of the *Feature Extraction* algorithms presented in the the previous section, we provide a list of meaningful descriptors for the singing voice. While some of these features suggest a direct mapping (e.g. *pitch* to *synthesis pitch*, *loudness* to*synthesis loudness*), other features offer a broader and more experimental mapping. We may imagine for instance, a mapping of the sung vowel (Vowel vector) to the amount of overdrive effect.

Up to this point, we have defined the First Mapping Layer of the archi-

tecture showed in the figure 3.8. The next step is to map these features onto synthesizer's parameters. This task is achieved in the Second Mapping Layer. It is necessary to mention that be focused our work in the first layer, thus only little experimentation has been done in the second mapping layer. The following sections describe a first approximation for the second layer in order to have a first functional application. As we discuss in the chapter 5, most of the research still to be done towards the completion of the PhD dissertation is in the second mapping layer. We strive to identify musical gestures from the extracted features and map them meaningfully to different categories of synthesis engine.

3.3.3 Mapping to a Physical Model Algorithm

This part is concerned with the second mapping layer, particularly for the case of a Physical Modeling algorithm, the Karplus-Strong. We have already mentioned that this is a very preliminary approach. Therefore, we are aware of the lack of generality of these results.

Basically, two parameters are important in the Karlplus-Strong algorithm. First, the delay line length, which will provide the synthesized pitch. Second, the initial content of the delay line for each plucked note. In the original algorithm [32], the delay line is initialized with noise samples that excites the model. In addition the output level and the frequency of a comb filter for simulating the plucked position on the string can be assigned.

In our approach, we map pitch and loudness directly to the corresponding counterpart. As excitation signal (initial values of the delay line), we use the block of input voice's samples corresponding to the attack. It is accomplished by buffering the input signal until a steady pitched note is detected. Then, the length of the delay line is set, and the attack samples are fed to the delay line, thus producing a new note. In order to modify the timbre of the sound, we use the Vowel Vector feature to control the Comb Filter Frequency. This makes the timbre dependant of the sung vowel.

Input Parameter	*Synthesis Parameter*
Pitch	Length of the delay line
Loudness	Gain
Attack(audio samples)	Delay line samples
Vowel Vector	Comb Filter Freq.

Table 3.2: Mapping of the input voice's parameters to the synthesis parameters of the Karplus-Strong algorithm.

Regarding the mapping of the pitch, since we are synthesizing a *bass* sound, depending on the note's pitch, the synthesized pitch is transposed down one octave in order to get a more realistic sound. Another alternative, not studied though, would be to quantize the pitch in order to simulate a bass guitar with frets.

3.3.4 Mapping to Spectral Model Algorithm (Morph)

The mapping strategy in our Spectral Model algorithm differs completely from the Physical Model presented above. Actually, this is a particular case of spectral modeling that can be handled as a sample-based synthesis in the spectral domain. On the other hand, this algorithm can also be seen as a kind of *Morph*. Here, we are not trying to go from one sound to another, what is commonly accepted as *Audio Morph*. Rather, we consider to control the characteristics of a target sound by another. More details about the actual implementation are explained in the chapter 4.

Our framework for spectral morph consists of three parts: first, taking frame by frame the spectrum of a stored sound template; second, transforming the spectrum of the template; and third, reconstructing the signal with the inverse FFT. A small-scale database containing pre-analyzed sound samples is available. The sound samples (instrument notes) are hand-labeled specifying their: *pitch, dynamics* and *attack type*.

The mapping problem is here two-fold. First, by using some extracted *Voice Features* from the First Mapping Layer, we need to select a sound template from the database. Second, map other features to the spectral transformation algorithms. The Input Vector contains three elements: *Pitch, Loudness* and *Attack Harshness*. Then, we use a retrieval method for selecting the sound template. By now, only *Euclidean distance* is used, which computes the square of the difference element-by-element. The figure 3.9 shows a representation of the database, with the sound templates spread over a 3D space whose axes are *dynamics*, *attack type* and*pitch*.

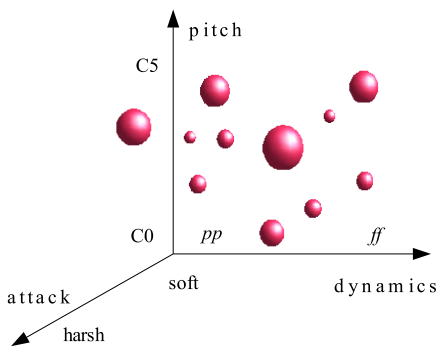


Figure 3.9: Spectral Morph Mapping. the spheres represent template sounds from the database distributed in a 3D space.

When a new note is triggered, we calculate which template sound in the database best matches our input vector. In a further step the spectrum of the selected template is transformed by different algorithms such as transposition or spectral shape modification. This ensures that the output sound follows the

performer's actions. In the table 3.3, we summarize the feature correspondence between the first and second mapping layer.

Input Parameter	*Synthesis Parameter*
Pitch Attack Harshness Loudness	Sound Template Retrieval
Pitch	Transposition
Vowel Vector	Spectral Shape Modifier
Breathiness	FX1
Hoarseness	FX2

Table 3.3: Model of First Mapping Layer for singing voice

In this section we have introduced a very basic mapping from *Voice Features* to a spectral model synthesis. In this approach an input parameters vector specifies the sound template by means of a retrieval method based on Euclidean distance. Other features are mapped to the transformation algorithms in order to control dynamically the output sound.

Chapter 4

Implemented System

In this chapter we present a system that integrates the result described previously. We developed a prototype consisting of a stand-alone software application. It must be assumed that this application is just a first experimental prototype with the goal of demonstrating the viability of such a system. The application consists of the audio in/out ports, processing algorithms and a graphical user interface.

4.1 Framework

The application was programmed in C++. For its development, we used the Integrated Development Environment (IDE) MS Visual Studio 7.1 [1]. Currently, it only runs on Windows operating system, but it can be complied for Linux and MacOSX.

The software was achieved within the CLAM framework, which is being developed and maintained at the IUA-MTG. Citing the CLAM's web page information [27]:

> CLAM is a full-fledged software framework for research and application development in the Audio and Music Domain. It offers a conceptual model as well as tools for the analysis, synthesis and transformation of audio signals. The original goal of the CLAM development framework was defined as: To offer a complete, flexible and platform independent Sound Analysis/Synthesis C++ platform to meet current and future needs of all MTG projects.

[1] http://msdn.microsoft.com/vstudio/

The three main axes of these goals were defined as:

- Complete: should include all utilities needed in a Sound Processing Project (input/output, processing, storage, display...)
- Flexible: Easy to use and adapt to any kind of need.
- Platform Independent: Compile under UNIX, Windows and Mac platforms.

These initial objectives have slightly changed since then mainly to accommodate to the fact that the library is no longer seen as an internal tool for the MTG but as a library that is published under the GNU-GPL in the frame of the Agnula IST European Project.

For our application, we took existing code for the Spectral Peak Processing Analysis (see section 2.1.4.2). All the code for control, user interface and synthesis algorithms was newly implemented.

4.1.1 Libraries utilized

Apart from using CLAM as software development framework, we utilized other libraries for specific tasks. Among the different alternatives available, we chose those that better adapt to our environment. In the next table 4.1, we enumerate the libraries and their functionality.

Library	*Functionality*
Trolltech's QT	Graphical User Interface
Synthesis Toolkit (STK)	Physical Modeling Basic Algorithms
RTAudio	Audio Real-time in/out Interface

Table 4.1: External Libraries and their functionality.

Specially, the *RTAudio* library presented severe difficulties for working in real-time for its long latency. Running under Windows, the latency of the system can reach 500ms., making it impossible to use it as a performance instrument. We studied already other alternatives, though. One is implementing the whole system as a VST plug-in. Another is compiling the application on Linux, since the RTAudio drivers offer a much shorter latency on this operative system.

4.1.2 Prototype

As we have mentioned, the mission of this first prototype is to proof the usability of the application, and help us during the development. The snapshot of the figure 4.1 is still a work-in-progress, but some parts are already fully functional. At the bottom, we find the Voice Features monitoring section. A set of level meters are updated at the frame rate, usually 172 frames/sec. They bring visual feedback to the performer, who can adapt his or her voice in order th get the desired values. On the right, by means of two combo boxes, the user select the

instrument and the synthesis technique employed. Currently, only an electric bass guitar is available, but a double bass sound will be also integrated for the spectral morph technique.

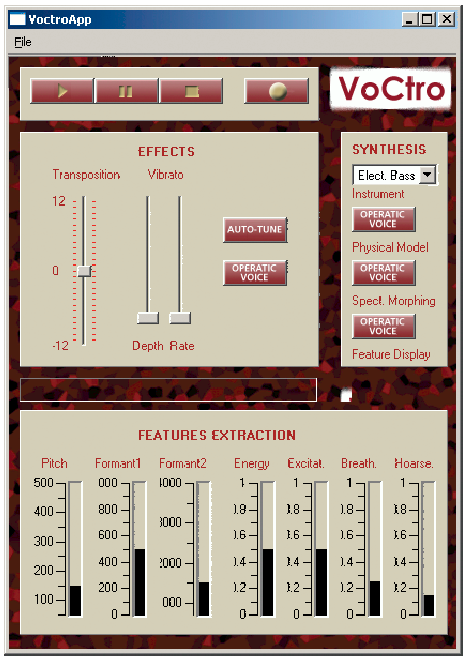


Figure 4.1: Prototype's User Interface. A set of level meters monitors the current state of the different features in real-time.

4.2 Synthesis of a Bass sound

From a musical point of view, the bass is not the most attractive instrument to emulate. The palette of sounds is reduced when compared to other solo instruments such as a trumpet or a violin. Moreover, since the input of our system is singing voice, we will experience that at the output the expressivity will be reduced. However, two reasons moved us towards this option. First, we plan to integrate the final application into a larger audio system targeted to DJs. This system aims at increasing the musical expressivity of the DJ performance. In a real situation, the DJ would trigger a rhythm loop and "sing" the bass line on top. Thus, the DJ can improvise with his or her voice, while keeping the hands on the turntables and mixing desk.

The second advantage is that using a "poor" sound (in terms of timbre space) such as the bass, permits us to focus on basic aspects of the system, pitch, loudness and note envelope.

In the following section we present the implementation of two different synthesis

algorithms for synthesizing a bass guitar, one based on physical models and the other based on spectral models. Again, the quality of the synthesis algorithms was not the goal of the system. Rather, our effort was put into the voice features extraction and the subsequent mapping.

4.2.1 Adapting the Karplus-Strong String Algorithm

This algorithm was invented by Kevin Karplus and Alex Strong , and was published in the Computer Music Journal in 1983 [32]. 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 with cheap processors. It is worth to mention that in 1983 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 have mentioned her 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 4.1), and writing the resulting sample in the wavetable. This can be seen, thus, as a delay line with length N.

$$Y_t = \frac{Y_{t-N} + Y_{t-N-1}}{2} \tag{4.1}$$

$$pitch = \frac{f_s}{N + 1/2} \tag{4.2}$$

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 4.2). This leads to a bad resolution for low values of N, quantizing the frequency for high pitches.

The physical interpretation, referring to the Digital Waveguide Theory [59], there are two waves travelling 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 4.3).

$$H(z) = \frac{1}{1 - \frac{1+z^{-1}}{2} z^{-N}} \tag{4.3}$$

The extensions developed by D. Jaffe and J.O. Smith [28] overcame several limitations of the original algorithm:

- Frequency Tuning to avoid quantization with an all-pass filter

- Decay Alteration: shortening low pitches and stretching for high pitches
- Dynamics control with a LPF
- Pick-Position with a comb filter
- Sympathetic String Simulation with a bank of strings

Detailed examples of these algorithms can be found in [14], and extended resources on *Digital Waveguides* are available online at J.O. Smith's web page[2].

Let us present here our practical implementation of a bass synthesizer using the Karplus-Strong algorithm. We designed two different approaches that satisfy the constraint of altering the timbre of the synthesized sound by the input voice's timbre. The figure 4.2 depicts a proposed algorithm that uses the transient of a sung note as the excitation signal. This excitation signal is fed into the Karplus-Strong delay line. In contrast, the figure 4.3 shows an alternative algorithms, where the excitation signal consists in a wavetable containing the recorded impulse response of the "plucking" action plus the body resonances of a real bass.

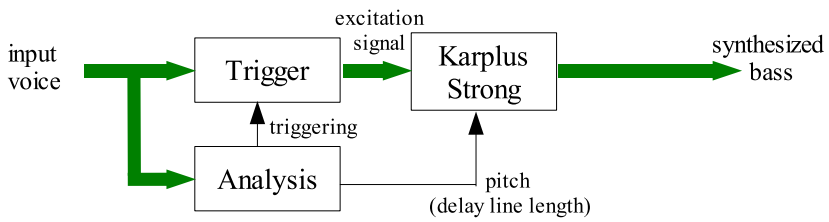


Figure 4.2: Bass synthesis with Physical Modeling. The input voice is used as excitation signal for the Karplus-Strong algorithm.

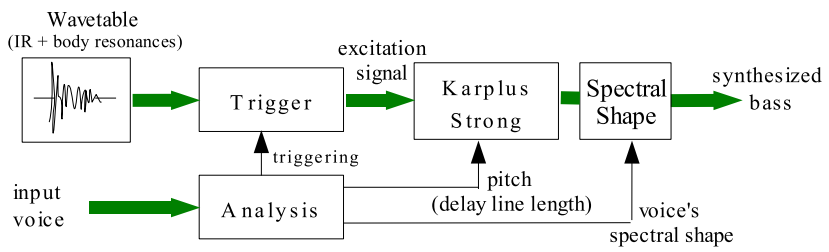


Figure 4.3: Bass synthesis with Physical Modeling. A wavetable with the impulse response and body resonances serves as excitation signal for the Karplus-Strong algorithm.

[2]http://ccrma.stanford.edu/ jos/wg.html

We chose to implement the first model presented (figure 4.2), where in practice, the attack of the note is used as excitation. Good results were achieved by attacking a note with a short 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 [15]. This Sound Synthesis Package provides a C++ class called *PluckedString.cpp*, that incorporates all mentioned extensions in the algorithm. From analysis stage, we get the estimation of the pitch, and the trigger control signal, which is related to the energy envelope of the user's voice. 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 4.2

During the design process, we noticed that continuation problems appeared in form of "clicks" in the output waveform, when changing from one note to another. This is caused while a note is fading out, and a new excitation signal is fed into the delay line. In order to minimize this effect, we introduce a second *string* in our model. The triggering stage send the excitation signal alternatively to one of the two strings. As we confirmed, the final synthesized sound was more natural, even though if a remaining note fading out was present.

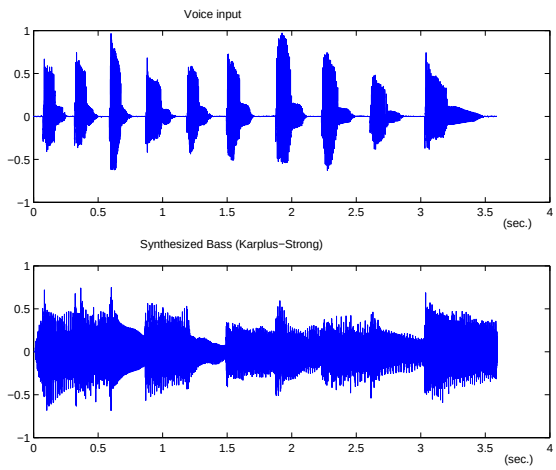


Figure 4.4: Input voice (top) and Karplus-Strong synthesized bass sound (bottom).

Finally, in the figure 4.4 shows the input voice's waveform and the synthesized bass sound. The current implementation uses a very simple *Mapping Layer*, taking only pitch information, formant frequencies and loudness from the input voice.

4.2.2 Spectral Morphing Model

In contrast to the Physical Model algorithm described above, our Spectral Model approach combines a sample-based synthesis (see section 2.3.3) with transformation, based in the Spectral Peak Processing (see section 2.1.4.2). A particularity of our sample-based algorithm is that it works in the frequency domain. Basically, depending on the input voice's parameters, a sound template track is selected from the database. Each template track contains all spectral frames from a single note. Then, we read periodically the spectral frames and transform some characteristics using the SPP framework. Finally, the processed spectral frames are converted to time domain through the inverse Fourier transform.

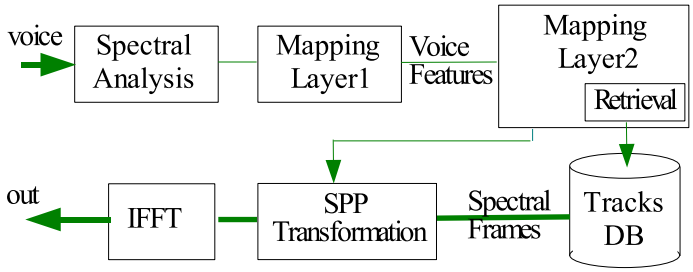


Figure 4.5: The voice features select a track from the database. The stored track's spectral frames are transformed and finally converted to the time domain.

For our bass synthesizer implementation, we set up a very small database consisting of 12 template tracks (a note's sample), six of electric bass, and six of double bass. The tracks are analyzed off-line in the spectral domain using the SPP model (see 2.1.4.2), and stored in the database in form of binary files containing spectral data. In a further step, all tracks have been labeled by hand according to its characteristics. Currently, only three features were annotated: *Pitch*, *Dynamics* and*Attack Type*. It results in a three dimensional space as it can be observed in the figure 3.9 of the previous chapter. The pitch values are specified in Hz, *Dynamics* and *Attack Type* range is [0..1]. In the case of Dynamics, 0 corresponds to a *pp* sound, and 1 to a *ff*. The attack type is a novel concept that we defined, and it is instrument dependant. Concerning bass sounds, we decided to classify two types of sounds: plucked and fingered, whose sounds are primarily related to the attack.

In the section 3.9, we have already presented the foundations of the mapping for the spectral model algorithm. A retrieval methods calculates the minimum Euclidean distance between the Input Vector (Pitch, Loudness and Attack Harshness) and the database elements. This outputs an ID corresponding to the selected track. Now, in a further step, we start reading the spectral frame of the track. Since we are dealing with a very small database, few combination of loudness and pitches are available. Therefore, some transformation has to be applied to the spectral frames. Basically, these transformation are of three types: transposition, gain and spectral shape modification. By now, only trans-

position and gain have been implemented. The transposition factor can easily calculated (equation 4.4).

$$transposition = \frac{pitch_{input}}{pitch_{template}} \tag{4.4}$$

Another factor that must be taken into account is the timing. The pre-analyzed template tracks have a certain duration, depending on the instrument and the playing properties. For each track, this is traduced into a certain number of spectral frames. In our system, though, the performer's voice controls the synthesized sound. Hence, is the input voice which decides the output duration. Our implementation uses a very "rudimentary" mechanism that deserves further research. Currently, we analyze the track by hand and set two loop points (*loopA* and *loopB*) within a steady region of the original sound. When a note is triggered, we start reading the template track frame by frame. When the point *loopB* is reached, we jump to the point *loopA* playing these frames in a loop. When the performer releases the current sung note, the last frames are played until the note's end.

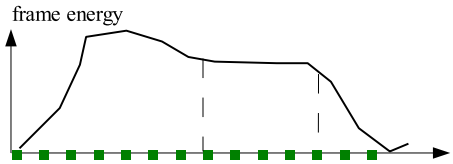


Figure 4.6: If the performer's note is longer than the stored template track, we loop the frames between the points A and B (specified by hand). The points on the x-axes represent the frames.

This implementation offers a poor sound quality due to the abrupt changes between loop points. Some ideas in order to overcome this problem are to set the looping points where the phase of the fundamental are coincident. Another variations is to take always the same frame and apply dynamic transformations a posteriori in order to keep the temporal expression. However, this model is our preferred implementation, as we discuss in the next chapter.

Chapter 5

Discussion and Future Work

5.1 Summary and conclusions

In this work we have introduced the voice as a musical controller for real-time synthesis. We strive to see if the singing voice through a microphone is a valid alternative to the current controllers, which includes keyboards, wind controllers and computer vision systems. A primary aim of any musical controller is to transmit the performer's expression to the synthesis engine. Thus, since the human voice is a highly expressive instrument, it seems adequate to employ it as a controller.

In the chapter 1, we introduced the problem, and addressed some preliminary questions on digital music synthesis. Particularly, we discussed our system from two different perspectives: as voice transformation, and as voice-controlled synthesis. The voice has been used since ancient times a musical instruments, and actually, any individual has innate control over his or her voice. However, this control varies among people, from a non-musician to a very skilled singer. In the chapter 2, we reviewed the voice as musical instrument, and presented some analytical models found in the literature. In addition, since our goal is to control digital music synthesis, we surveyed control issues on electronic instruments as well as on different synthesis techniques. Our contributions within this work are covered in the chapter 3. Basically, we distinguished two topics: Voice Features Extraction, and Mapping. Most of th results presented belong to the first topic. Thus, future research will focus on exploring the mapping layer. In order to test the usability and interest of the proposed system, we developed a prototype consisting in a software application. It takes an input audio stream (microphone), and outputs a synthesized instrument. In its first version, we integrated only guitar bass sound, but we plan to extend it to other instruments in the future. It is covered in the chapter 4. Different aspects of

the implemented algorithms are covered. Finally, the chapter 5 is dedicated to the conclusions and the intended future work.

The main outcomes of the present research work has been the following ones. Firstly, to derive a model that parameterizes the singing voice in order to use it as musical controller. And second, mapping these features to basic prototypes of two different synthesis engines. In this sense, we established a separation between two layers of mapping. The first layer analyzes the voice and outputs a set of high-level descriptors. And the second layer assigns these descriptors to parameters of the synthesis algorithms.

In our opinion, the results of the research for the first mapping layer are satisfactory. We proposed a set of meaningful descriptors for the singing voice, and extracted features by developing some novel algorithms. Nevertheless, there is a lack of high-level descriptors, which can help in describing the overall user's performance. Actually, finding new valid descriptors will be a significant task within the thesis work. By now, the preliminar list of current *high-level* descriptors is:

Fundamental Frequency, Fundamental Frequency Gradient,
Loudness, Brightness, Timing descriptors, Breathiness,
Hoarsensess, Vowel Vector and Attack Harshness.

On the other hand, we realized that the mapping to the synthesis engines appeared as a more complex task than we thought in the beginning. Although we achieved to synthesize a bass guitar sound from the input voice in real-time, the control is very limited. Between the two synthesis techniques implemented, Physical Model and Spectral Model, it is clear that the former is cheaper computationally, but in the case of the plucked-string, it does not permit many timbre variations. Another drawback of employing Physical Modeling as synthesis engine, is that it requires for each instrument a completely different model. Thus, making complex the integration of new instruments. The adaptation of existing algorithms or the development of novel ones, involves a strong expertise in the field. In contrast, our Spectral Model approach, which is based on pre-analyzed real samples, offers a major flexibility for integrating new instruments. Furthermore, the timbre characteristic of the output sound can be transformed using the Spectral Peak Processing framework.

Our conclusion regarding the two synthesis techniques is that the Spectral Model appears to be the better choice for further research. It allows to integrate easily new sounds, and we can make use of those existing spectral transformation algorithms developed at the IUA-MTG. In the spectral model, both processes *Analysis* and *Synthesis* use the same technique. It reveals a lack of substantial differences between processing and synthesizing in the spectral domain, as we have posed in the chapter 1. In our opinion, this hypothesis deserves a more thorough study that can be accomplished within the PhD thesis. In any case, the system appears to be rewarding, when a user is confronted with the microphone, but there is a learning time in which the user experiments with the sound

control. However, currently, the main negative point is the latency between input and output due to the computer's drivers, which we plan to overcome in next prototypes.

We can conclude that the proposed topic has a solid background in the field of computer music, and at the same time, it offers possibilities for important research contributions in both technological and musical aspects. This justifies from our point of view a PhD dissertation. Concerning the technological or scientific contributions, we can provide new signal processing techniques for analyzing the singing voice using spectral processing. These techniques intend to go beyond the classical *Singing Voice Analysis/Synthesis* framework, in particular by generating in real-time high-level descriptors of two kind. First, a set of instant descriptors such as breathiness; and second, performance's gestures descriptors, including mood, dynamics, intonation, etc. The latter descriptors look at the performance over time and may require aid from some machine learning techniques. Another field that can benefit of our work is the *New Musical Interfaces*. We will specify a model for mapping the extracted voice features to synthesis parameters of different virtual acoustic instruments. Our goal is to define a general model that maps meaningfully the voice to certain instrument family(string, percussion, wind, etc.). We will try to explore the underlying rules that allow an expressive and intuitive control of the synthesis algorithms. In addition, this mapping model can be used by music manufacturers as an add-on in their new synthesizers, which are likely to come into the market with new interaction features. In terms of technical viability, referring to the achieved results of the implemented prototype, the proposed system can work in real-time on a general purpose computer. That is fact is crucial in order to evaluate the usability of our system, since it must be used in a real-time performance situation.

On the other hand, the proposed thesis contributes also from a musical point of view. It explores the singing voice from another perspective, considering it as controller rather than as instrument. Some of the proposed high-level attributes can be useful as training information for singing students. Concerning real-time performance, different situations can benefit from such a system. First, let's imagine a keyboard player that has a more expressive alternative for controlling a trumpet sound. Second, we assume that most musicians have, to some extend, singing skills. Here, we consider that a trained musician can produce also satisfactory results on other instruments by controlling the synthesis with his or her voice. Third, although it is not one of our initial objectives, a voice-driven synthesizer can have in addition multiple applications in the field of experimental music. Finally, in our opinion, naive users and amateur music enthusiasts will take surely advantage of the system.

As we have mentioned previously, a PhD dissertation based on the presented research work satisfies three significant points: it is interesting, challenging and feasible.

5.2 Future work

This document serves as a research proposal for the completion of the PhD dissertation. We introduced the voice as musical controller, which appears to be an interesting topic for the PhD Thesis. It combines issues such as singing voice analysis, spectral processing, digital instruments control, mapping and synthesis techniques. In our opinion, the presented research has served as a proof-of-concept, demonstrating the suitability of the topic addressed. Summing up the different aspects that will be covered by the thesis, we suggest the following brief description:
Playing instruments with the voice: extraction of voice high-level attributes, and definition of mapping strategies for the synthesis of virtual musical instruments

A significant step of the thesis will be to address various conceptual issues regarding voice analysis and musical control. Some facets can be summarized as work hypothesis, which we will have to demonstrate. These hypothesis are listed below:

- The Singing Voice, not as musical instrument but as musical interface
- Beyond Pitch-to-MIDI. More than just a bandwidth limitation.
- Boundaries between Transformation and Synthesis using spectral processing techniques

The study of the aforementioned questions will put the foundations for the subsequent research activities, converging to the implementation of a usable real-time system. The work should focus on the voice analysis and the mapping layers, attaining good results in transmitting the performer's expression to the sound generation algorithms. In this section, we aim at specifying clearly the future research objectives and goals. We define our work in three main axes:

- Extraction of high-level features from the singing voice: Some high-level attributes of the voice have been already presented in this work. We plan to extend them by identifying new meaningful attributes. These a attributes should increase the expression capabilities for the performer. It includes the development of new signal processing algorithms that extract features of two different nature:
  - Frame descriptors
  - Gesture or Contextual descriptors

  Under *Frame descriptors*, we consider those attributes that refer to the analysis of one spectral frame. We will handle these descriptors as instantaneous information. The second group of descriptors intend to extract information of the temporal evolution of the performance. These features will describe aspects such as the phrasing, mood, dynamics, etc. The

study of these descriptors can derive in complex algorithms combining signal processing and machine learning techniques.

- Definition of a mapping model for different instrument categories: Our goal in the area of mapping will be to derive a general model for controlling different virtual acoustic instruments with parameters extracted from the voice. The mapping layer will take voice features as input and, according to the instrument nature, assign them meaningfully to the synthesis parameters. Here, our purpose will be to determine which parameters allow, for each instrument, a more intuitive and expressive performance. For instance, mapping the input frame energy to the synthesized output energy can have sense for a trumpet sound, but it is definitely meaningless for the synthesis of a plucked bass guitar, if we want to maintain some realism. In order to simplify the the range of musical instruments to control, we will cover only three categories of instruments:

  - Plucked instruments
  - Blown instruments
  - Struck instruments

  We will start by improving and extending the first prototype, which synthesizes a bass guitar(see section 4). In a further step, in the category of wind instruments, we plan to implement the control of a trumpet. Concerning percussion instruments, we consider the possibility of integrating timbre classification algorithms, which should discern between bass drum, snare drum and hi-hat.

- Development of a prototype and experiments with target users: In the chapter 4 we have introduced a first experimental prototype. The third part of our thesis work will be dedicated to the development of a prototype. An initial restriction of the prototype is that it must work in real-time. Once our prototype is fully functional, the next step is to evaluate it with users. We plan to carry out two evaluation sessions, each one with different purpose. First, to use these test as data-acquisition sessions. By analyzing the performance of distinct target users (non-musicians, skilled singers, etc.), we can extract useful information. Second, to validate the system as musical instrument, taking note of the subject's overall impression. With our implemented *bass synthesizer* system, we plan to test the system with three groups of users:

  - Naive users
  - Instrumentalists (for plucked, blown and struck instruments)
  - Singers

  Regarding the first test session, with naive users, we test the robustness of the system. From these experiments we can identify gestures performed by a bass player to control *his/her* instrument with the voice. On the

other hand, a professional singer can test the reliability of the proposed voice descriptors for controlling a synthesis algorithm.

The validation process will be carried out in the final evaluation session. The subjects will be given a questionnaire covering various aspects of the system such as usability, learning process, expressiveness, sound quality, response time, etc. In addition, we plan to perform a listening test comparing, in term of expressivity, the output of our system and the output generated by traditional controllers.

- Other potential applications: In addition to the described main research axes, we should also consider other possible future directions of the achieved research. Here we name some extensions and application that will deserve our attention. Our goal is to derive a general module for using the voice as musical controller. Hence, we should extend our prototype to other instruments. Within the Semantic HiFi Project we are required to provide some tools for controlling the synthesis of percussion instruments. We plan to adapt the current system in order to generate percussive sounds. In the hip-hop community, imitating the sounds of a complex rhythm track with the voice is a peculiar technique called *beat-boxing*. The strategy will be to map the voice's timbre to a fixed set of percussion sounds, so that a similarity algorithm decides dynamically which sound must be produced. In a further step, we propose to study the integration of other synthesis techniques such as abstract techniques (see section 2.3.2). Moreover, the voice features extracted could be used in a more general variety of applications. Particularly, it seems convenient to study the control of audio effects with the voice, for example, distortion effects, filters, etc. Also, in a completely other perspective, we plan to evaluate the integration of these voice features into other HCI applications such as video games. We can imagine a video character driven by the player's pitch. Additionally, disabled people might benefit from such as system. However, it requires of extended usability studies in the HCI discipline, which goes beyond the scope of our dissertation. Some new potential directions to take include:

  - Generalization to other synthesis techniques
  - Generalization for controlling Audio Effects
  - Voice Features as novel User Interface in HCI systems

Appendix A

In the appendix, we include two papers in which the author has participated. Since both papers were done with collaboration of other colleagues, We will point out the tasks in which the author was involved.

- Celma, O. and Gómez, E., Janer, J., Gouyon, F., Herrera, P. and Garcia, D. , *"Tools for Content-Based Retrieval and Transformation of Audio Using MPEG-7: The SPOffline and the MDTools"*, Proceedings of 25th International AES Conference, 2004. London, UK.

  Part of my work for the CUIDADO Project was the implementation of the *SoundPalette Offline*. This is a tool for musicians and sound engineers, in which the sound transformation was based on extracted audio metadata. In this paper, I described the functionality of the application.

- Fabig, L. and Janer, J., *"Transforming Singing Voice Expression - The Sweetness Effect"*, Proceedings of 7th. International Conference on Digital Audio Effects (DAFX), 2004, Naples, Italy.

  This paper presents some high-level transformation for the singing voice. It resulted from the development of a *Vocal Processor*. I contributed with an algorithm that detects the sub-harmonicity present in the voice. This parameter is used for applying "sweetness" only on certain regions, and thus, keeping as much as possible of the original sound.

.

.

.

.

.

.

.

.

.

.

.

.

Bibliography

[1] D. Arfib. Digital synthesis of of complex spectra by means of multiplication of non-linear distorted sine-waves. *Journal of the AES*, 27(10), 1979.

[2] The International MIDI Association. *MIDI 1.0 Detailed Specification.* http://www.midi.org.

[3] J. Bonada, O. Celma, A. Loscos, J. Ortolà, and X. Serra. Singing voice synthesis combining excitation plus resonance and sinusoidal plus residual models. In *Proceedings of International Computer Music Conference 2001*, Havana, Cuba, 2001.

[4] J. Bonada and A. Loscos. Sample-based singing voice synthesizer by spectral concatenation. In *Proceedings of Stockholm Music Acoustics Conference 2003*, Stockholm, Sweden, 2003.

[5] Buchla and Associates. http://www.buchla.com.

[6] Ramakrishnan C., J. Freeman, and K. Varnik. The architecture of Auracle: A real-time, distributed, collaborative instrument. In *Proceedings of New Interfaces for Musical Expression (NIME) 2004*, Hamamatsu, Japan, 2004.

[7] A. Camurri, S. Hashimoto, M. Ricchetti, R. Trocca, K. Suzuki, and G. Volpe. Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal*, 24(1):57–69, 2000.

[8] P. Cano. Fundamental frequency estimation in the SMS analysis. In *Proceedings of COST G6 Conference on Digital Audio Effects 1998*, Barcelona, 1998.

[9] O. Celma, E. Gómez, J. Janer, F. Gouyon, P. Herrera, and D. Garcia. 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, 2004.

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

[11] P.R. Cook. *Identification of Control Parameters in an Articulatory vocal Tract Model, with applications to the Sunthesis of the Singing.* PhD thesis, Stanford University, Stanford, 1991.

[12] P.R. Cook. Spasm: a real-time vocal tract physical model editor/controller and singer: the companion software synthesis system. *Computer Music Journal*, 17(1):30–44, 1992.

[13] P.R. Cook. Toward the perfect audio morph? singing voice synthesis and processing. In *Proceedings of the 1st. International Conference on Digital Audio Effects (DAFX)*, Barcelona, 1998.

[14] P.R. Cook. *Real Sound synthesis for Interactive Applications.* A.K. Peters Press, 2002.

[15] P.R. Cook and G. Scavone. *The Synthesis Toolkit (STK).* http://ccrma-www.stanford.edu/software/stk.

[16] M. Dolson. The phase vocoder: A tutorial. *Computer Music Journal*, 10:14–27, 1986.

[17] L. Fabig and J. Janer. Transforming singing voice expression - the sweetness effect. In *Proceedings of the 7th. International Conference on Digital Audio Effects (DAFX)*, Naples, Italy, 2004.

[18] G. Fant. *Acoustic Theory of Sech Production.* Mouton, The Hague, 1960.

[19] J.L. Flanagan and R.M. Golden. Phase vocoder. *Bell Systems Technology Journal*, 45:1493–1509, 1966.

[20] B. Gold and N. Morgan. *Speech and Audio Signal Processing.* John Wiley and Sons, New York, 1999.

[21] MIT Hyperinstruments Group. *The Brain Opera.* http://brainop.media.mit.edu.

[22] J. Haas. Salto - a spectral domain saxophone synthesizer. In *Proceedings of MOSART Workshop on Current Research Directions in Computer Music*, Barcelona, 2001.

[23] Y. Hang. Realtime face synthesis driven by voice. In *Proceedings of International Conference on Comuputer Aided Design and Computer Graphics 2001*, Kunming, China, 2001.

[24] A. Hunt and M. Wanderley. Mapping performer parameters to synthesis engines. *Organized Sound*, 7(2):97–108, 2002.

[25] A. Hunt, M. Wanderley, and M. Paradis. The importance of mapping in electronic instruments design. In *Proceedings of the Conference on New Instruments for Musical Expression NIME*, Dublin, Ireland, 2002.

[26] Mathworks Inc. *Matlab Documentation.* http://www.mathworks.com.

[27] Universitat Pompeu Fabra. IUA-MTG. *CLAM Documentation.* http://www.iua.upf.es/clam.

[28] D.A. Jaffe and J.O. Smith. Extensions on the Karplus-Strong plucked-string algorithm. *Computer Music Journal*, 7(2):56–69, 1983.

[29] T. Jehan. *Perceptual Synthesis Engine: An Audio-Driven Timbre Generator*, 2001.

[30] J.D. Johnston. Transform coding of audio signals using perceptual noise criteria. *IEEE on Selected Areas in Communications*, 1988.

[31] S. Jordà. Sonigraphical instruments: From FMOL to the reacTable*. In *Proceedings of 2003 International Conference on New Interfaces for Musical Expression*, Montreal, Canada, 2003.

[32] K. Karplus and A. Strong. Digital synthesis of plucked-string and drum timbres. *Computer Music Journal*, 7(2):43–55, 1983.

[33] J.L. Kelly and C.C. Lochbaum. Speech synthesis. In *Proceedings of the foruth International Congress in Acoustics.*, 1962.

[34] J. Laroche and M. Dolson. New Phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, 1999.

[35] T. Machover. *Hyperinstruments - a Composer's Approach to the evolution of Intelligent Musical Instruments.* MillerFreeman, Inc., 1992.

[36] Electronic Musician Magazine. http://www.emusician.com.

[37] R.C. Mahler and J.W. Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America*, 4:2254–2263, 1994.

[38] T. Marrin. *Inside the Conductor's Jacket: Analysis, Interpretation and Musical Synthesis of Expressive Gesture.* PhD thesis, MIT Media Laboratory, Massachusetts, 2000.

[39] R.J. McAuclay and T.F. Quatieri. Speech analysis/synthesis based on sinusoidal representation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):744–754, 1986.

[40] E. Metois. *Musical Sound Information: Musical Gestures and Embedding Synthesis.* PhD thesis, MIT Media Laboratory, 1996.

[41] F.R. Moore. The dysfunctions of MIDI. *Computer Music Journal*, 12(1):19–28, 1988.

[42] New Interfaces for Musical Expression NIME. http://www.nime.org.

[43] W. Oliver, J. Yu, and E. Metois. 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, 1997.

[44] Conference on Digital Audio Effects. http://www.dafx.de.

[45] D. O'Shaughnessy. *Speech Communication, Human and Machine.* Addison-Wesley, New York, 1987.

[46] M.S. Puckette. Phase-locked vocoder. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, New York, 1995.

[47] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition.* Prentice-Hall, Englewood Cliffs, 1993.

[48] L.R. Rabiner and R.W. Schafer. *Digital Processing of Speech Signals.* Prentice-Hall, 1978.

[49] C. Roads. *Computer Music Tutorial.* The MIT Press, Cambridge, Massachusetts, 1996.

[50] D. Rocchesso and F. (editors) Fontana. *The Sounding Object.* Mondo Extremo Publishing, 2003.

[51] X. Rodet. Time-domain formant-wave-function synthesis. *Computer Music Journal*, 8(3):9–14, 1984.

[52] J. Rothstein. *MIDI. A Comprehensive Introduction.* Oxford University Press, Oxford, 1992.

[53] G. Scavone. Modeling and control of performance expression in digital waveguide models of woodwind instruments. In *Proceedings of International Computer Music Conference (ICCM)*, Hong Kong, 1996.

[54] R. Schafer and L. Rabiner. System for automatic formant analysis of voiced speech. *The Journal of the Acoustical Society of America*, 47:634–648, 1970.

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

[56] X. Serra. *A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition.* PhD thesis, CCRMA, Stanford University, 1989.

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

[58] J.O. Smith. Viewpoints on the history of digital synthesis. In *Proceedings of the International Computer Music Conference (ICMC).*, pages 1–10, Montreal, Canada, 1991.

[59] J.O. Smith. Physical modeling using Digital Waveguides. *Computer Music Journal*, 16(4):74–91, 1992.

[60] Akademie Schloss Solitude. *Auracle.* http://www.auracle.org.

[61] J. Sundberg. *The Science of the Singing Voice.* Northern Illinois University Press, Illinois, 1987.

[62] M. Wanderley. *Performer-Instrument Interaction: Applications to Gestural Control of Music.* PhD thesis, University Pierre et Marie Curie - Paris VI, Paris, France, 2001.

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

[64] U. Zölzer. *DAFX - Digital Audio Effects.* John Wiley & Sons Publishers, New Jersey, 2002.