Singing Voice Resynthesis using Concatenative-based Techniques

Nuno Miguel da Costa Santos Fonseca

Universidade do Porto

Faculdade de Engenharia

FEUP

Faculty of Engineering, University of Porto

Department of Informatics Engineering

December 2011

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Informáica

Singing Voice Resynthesis

using Concatenative-based Techniques

Nuno Miguel da Costa Santos Fonseca

Dissertação submetida para satisfação parcial
dos requisitos do grau de doutor
em
Engenharia Informática

Dissertação realizada sob a orientação do
Professor Doutor Aníbal João de Sousa Ferreira
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto
e da
Professora Doutora Ana Paula Cunha da Rocha
do Departamento de Engenharia Informática
da Faculdade de Engenharia da Universidade do Porto

Porto, Dezembro de 2011

This dissertation had the kind support of FCT (Portuguese Foundation for Science and Technology, an agency of the Portuguese Ministry for Science, Technology and Higher Education) under grant SFRH / BD / 30300 / 2006, and has been articulated with research project PTDC/SAU-BEB/104995/2008 (Assistive Real-Time Technology in Singing) whose objectives include the development of interactive technologies helping the teaching and learning of singing.

Author email: nuno.fonseca@ipleiria.pt

Abstract

Singing has an important role in our life, and although synthesizers have been trying to replicate every musical instrument for decades, is was only during the last nine years that commercial singing synthesizers started to appear, allowing the ability to merge music and text, i.e., singing. These solutions may present realistic results on some situations, but they require time consuming processes and experienced users.

The goal of this research work is to develop, create or adapt techniques that allow the resynthesis of the singing voice, i.e., allow the user to directly control a singing voice synthesizer using his/her own voice. The synthesizer should be able to replicate, as close as possible, the same melody, same phonetic sequence, and the same musical performance.

Initially, some work was developed trying to resynthesize piano recordings with evolutionary approaches, using Genetic Algorithms, where a population of individuals (candidate solutions) representing a sequence of music notes evolved over time, tries to match an original audio stream.

Later, the focus would return to the singing voice, exploring techniques as Hidden Markov Models, Neural Network Self Organized Maps, among others. Finally, a Concatenative Unit Selection approach was chosen as the core of a singing voice resynthesis system.

By extracting energy, pitch and phonetic information (MFCC, LPC), and using it within a phonetic similarity Viterbi-based Unit Selection System, a sequence of internal sound library frames is chosen to replicate the original audio performance.

Although audio artifacts still exist, preventing its use on professional applications, the concept of a new audio tool was created, that presents high potential for future work, not only in singing voice, but in other musical or speech domains.

Keywords: Resynthesis, Singing, Voice, Concatenative, Sampling, Audio Mosaic, Analysis, Feature Extraction, Synthesis, Pitch, Dynamics, Phonetics, Vocal, Speech, Music, Sound, Audio, Unit Selection, Music Transcription, Genetic Algorithms.

Resumo

A voz cantada sempre teve um papel importante nas nossas vidas, e embora há várias décadas que os sintetizadores tentam replicar todos os instrumentos musicais existentes, apenas nos últimos nove anos as primeiras soluções comerciais de síntese de voz cantada apareceram, permitindo a combinação de música e texto, ou seja, "cantar". Essas soluções podem criar resultados realistas em determinadas situações, mas requerem processos muito morosos e utilizadores experientes.

O objectivo deste trabalho de investigação é desenvolver, criar ou adaptar técnicas que permitam a re-síntese da voz cantada, isto é, que permitam ao utilizador controlar diretamente um sintetizador de voz cantada, usando a sua própria voz. Desta forma, o sintetizador deverá ser capaz de replicar, da melhor forma possível, a mesma melodia, a mesma sequência de fonemas e a mesma performance musical.

Inicialmente, algum trabalho foi feito na re-síntese de gravações de piano com abordagens evolutivas, usando Algoritmos Genéticos, onde uma população de indivíduos (potenciais soluções) representa uma sequência de notas que evolui ao longo do tempo, tentando replicar o áudio original.

Numa fase posterior o foco dirigiu-se para a voz cantada, explorando técnicas como *Hidden Markov Models*, *Neural Network Self Organized Maps*, entre outros.

Finalmente, uma abordagem concatenativa baseada na seleção de unidades (*Unit Selection*) foi escolhida para núcleo de um sistema de re-síntese vocal.

Extraindo a componente dinâmica (energia), altura (notas musicais) e informação fonética (MFCC e LPC), e usando um sistema de selecção de unidades e semelhança fonética, uma sequência de *frames* da biblioteca interna de sons é escolhida para replicar a performance do áudio original.

Embora ainda existam artefactos áudio, impedindo o seu uso em aplicações profissionais, foi criado o conceito de uma nova ferramenta áudio, que apresenta grande potencial para trabalhos futuros, não apenas para voz cantada, mas também para fala ou outros domínios musicais.

Palavras: Re-síntese, canto, voz, concatenação, amostragem, mosaicos áudio, análise, extração de informação, síntese, dinâmica, fala, música, som, áudio, seleção de unidades, transcrição de música, algoritmos genéticos.

Acknowledgments

The work behind this PhD dissertation was only possible, thanks to a group of people and organizations. As such, I would like to thank them.

To my PhD supervisors, Prof. Anibal Ferreira and Prof. Ana Paula Rocha, for their support and advice through these years.

To FCT for their PhD scholarship.

To Polytechnic Institute of Leiria – ESTG for reducing my work hours.

To EASTWEST, for contributing with their sampling material.

To Gustavo Reis, for the collaborative research work done in music transcription.

To Ricardo Sousa, for exchanging points-of-view and voice related references.

To Rui Pedro Paiva, for sharing his implementation of SACF.

To Valentim Emyla (and co-authors) for sharing their testing material/results regarding music transcription metrics.

To the listening tests users, for their time and patience doing tests.

To all other persons that directly or indirectly contributed to the work.

Last, but not the least, I would like to thanks my family (Nélia, Alexandre and Ana), for allowing me to take some hours of family time during the PhD work.

Thank you very much…

Table of Contents

List of Figures

List of Tables

Main Abbreviations

ACF – Autocorrelation Function

DAW – Digital Audio Workstation

DSP – Digital Signal Processor

FFT – Fast Fourier Transform

GA – Genetic Algorithms

GP – Genetic Programming

HMM – Hidden Markov Models

LPC – Linear predictive coding

MFCC – Mel-Frequency Cepstral Coefficients

MIDI – Musical Instrument Digital Interface

MIR – Music Information Retrieval

NN – Neural Network

PSOLA – Pitch-Synchronous OverLap and Add

SACF – Summary Autocorrelation Function

SOM – Self Organized Map

1. Introduction

1.1 Context

Since pre-historic times, music always had an important role on the human life. With time, better and more complex musical instruments were created, expanding the horizons of music.

With the birth of electronics, the first synthesizers started to appear, which are devices capable of creating new sounds or recreating existing musical sounds, and having a great impact on the way music was made on the second half of the XX century. With them, musicians have at their disposal a huge amount of sounds, without the need of using the actual physical music instruments or needing to master their playing techniques, in addition to the ability to recreate completely new sounds. With time, the quality and realism of these "virtual" instruments increased and, in many situations, it is impossible, even for professionals, to distinguish between the original sounds and the synthesized ones [NorthernSounds, 2003].

Nevertheless, one highly used musical instrument continued to provide many difficulties to its simulation with synthesizers: the human voice. Although many musical instruments present several different registers, the human voice presents a unique characteristic that is responsible for its major complexity: the ability to associate text with music, i.e., singing.

During the last decade, the first commercial solutions for singing voice synthesis were released [EASTWEST, 2001][EASTWEST, 2005][Yamaha, 2004][VirSyn, 2004]. Even though they presented some limitations, these solutions were finally able to associate text with music, through the simulation of a solo singer or a choir.

Although the first barrier of singing synthesis was overcome - the ability to merge text and music - the fact that the human voice allows so many different variations (musically and phonetically), creates a new problem for the user: how to instruct the synthesizer/computer to perform (sing) according to the wishes of the user? The way the text is pronounced; the duration of each phoneme; the transitions between phonemes; the melodic line; the desired music performance in terms of pitch effects (e.g. *vibrato* or *portamento)*; the music characteristics of the attack of each

phoneme; the sound intensity dynamics; etc. – all these options are only a small subset of the possible choices that are needed to be done to be able to achieve realistic and professional sounding singing performances.

Current solutions already allow some of those choices through graphical interfaces, but in most cases they represent a complex and time-consuming task even for experienced users, that may require several hours of work to achieve some seconds of realistic singing performance.

A possible way to highly decrease the complexity of such work, increasing the process productivity, may involve using the user's voice (or any other vocal recordings), to easily demonstrate how the singing performance should be obtained, creating a **resynthesis** process. For instance, a movie composer could use his/her voice to sing something, showing how his/her choir synthesizer should create the desired choir performance. It would be even possible to allow the editing of several parameters as a way to overcome some performance limitation of the users (out-of-tune notes, etc.).

Since the synthesizer is controllable by an audio stream, the system could also be used on an "Audio Effect" context, transforming the input voice into another one, as Figure 1.1 suggest, keeping the same musical and phonetic characteristics, but replacing the timbre/sound characteristics.

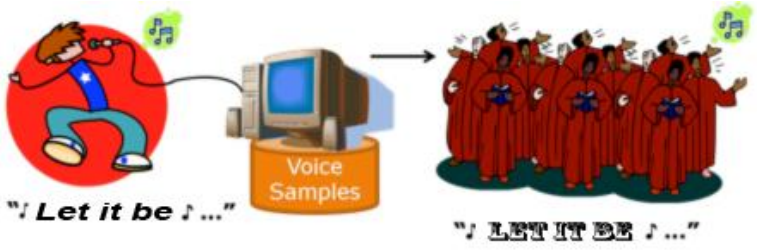

Figure 1.1 – Voice controlled singing synthesizer

1.2 Motivation

This PhD dissertation appears as a continuation of the work that the PhD student has been developing for some years now to the EASTWEST[1] Company (Hollywood, USA).

During 2001, EASTWEST released the "Quantum Leap - Voices of the Apocalypse" sound library [EASTWEST, 2001], with female and male choirs. Although the library contains vowels and consonant recordings, from the practical point-of-view, the user would use only one type of sound at each time (e.g. "ah" vowel).

After a post on an internet forum [NorthernSounds, 2002] in April 2002, on which the PhD student suggested the creation of a software application that allows users to enter the text that should be

[1] One of the most successful soundware companies in the world [EASTWEST, 2010].

sung by the virtual choir, a collaboration with the American company was started, although with some precautions from the company regarding the prospect of such tool.

During the first months, a prototype was evolved until being released as a freeware utility in the summer of 2002, designated as "Voices of the Apocalypse Utility" [Fonseca, 2002]. With the appearance of a single text editor (similar to Notepad), the user could enter the text that he wishes to hear, using a pre-defined syntax. The application receives the melody by MIDI in real-time, and based on the written text, it outputs, also in real-time, the required MIDI information to control the sampler[2] in the desired way. In the next year (2003) a second version of the application was available (this time with an associated cost) with several new features, including a graphical editor and an internal English-phonetic convertor [Fonseca, 2003][Fonseca, 2003b].

In 2005, EASTWEST released the "EASTWEST/Quantum Leap Symphonic Choirs" package [EASTWEST, 2005], with completely new choir recordings (*samples*), a sampler software and a new text-to-sing application. The new sound library[3] included boys/sopranos/altos/tenors/basses choir voices; the sampler engine was a customized version of the Kontakt sampler (from the German company Native Instruments); and the text-to-sing application, named WordBuilder™ (Figure 1.2) and developed by the PhD student, corresponded to an evolution of previous applications ("VotA Utility").

Figure 1.2 – WordBuilder™

These sound library packages, including its applications "VotA Utility" and "WordBuilder™", have received several awards [EASTWEST, 2001; EASTWEST, 2005] and were reviewed in several sound

[2] Device/software responsible for playing the pre-recorded *samples*.

[3] Recorded by Prof. Keith O. Johnson, two times Grammy award (Best Engineered Classical Recording).

magazines [SoundOnSound, 2002; SoundOnSound, 2005; Keyboard, 2003; EM, 2006;EM, 2006b; Keyboard, 2006] and books [Gilreath, 2004], being also displayed on international music fairs. They are currently being used by composers all over the world in movies, TV series and game soundtracks.

1.3 Goals, Challenges and Approaches

Goals

The goal of this PhD work is to develop, create or adapt techniques that allow the resynthesis of the singing voice, i.e., that allow the user to control a singing voice synthesizer using his/her own voice. The synthesizer should be able to replicate, as close as possible, the same musical performance, namely the same melody, same phonetic sequence, and the same personal music gesture[4].

Since many singing performances include non-existing words, one important requirement is that the system should be language-free (working with any language or with any phonetic sequence). Also, due to the nature of its application, the system will consider only monophonic audio material.

Challenges

The development of such a system presents several challenges:

Musical performance replication – to achieve its goal, the system must be able to extract and replicate the musical performance of the singer, which represent a challenging task.

Pitch extraction – although monophonic pitch extraction has been the object of intense research during decades, it still presents many issues, especially in the presence of noise or on unvoiced regions.

Phonetic replication – similar to the music domain, the replication of phonetic information is a challenging task.

Language Free – language free systems present two additional challenges: the need to handle rare or unknown phonemes; also, the fact that the system cannot benefit from language specific characteristics. Both of them limit the use of learning mechanisms.

Synthesis model – most synthesizer approaches are note-oriented, since they are to be used with MIDI or in similar environments. This type of approach is probably the best for most applications, but it faces difficulties when the origin of control is a real audio stream, which shows a "continuous" behavior instead of a "discrete" note behavior.

[4] Not the timbre characteristics of the voice, but the singer's interpretation of the song.

Timing - to reproduce the same musical/phonetic performance, the system must respect the original phonemes timings. This adds a new constrain to the synthesizer, by creating some time requirements for each phoneme.

Approaches

To accomplish the desired goals, the PhD work will try to explore the following approaches:

Merging speech and music research – The area of singing voice lays on the frontier between speech and music research areas. As such, this PhD work will try to merge concepts and techniques from both worlds: speech analysis with music analysis, and, speech synthesis with music synthesis.

Pitch extraction – Although pitch extraction has an important role on the system, several decades of research have been done on the subject of monophonic pitch extraction. As such, an existing monophonic pitch extraction system will be chosen and used, since trying to create new techniques could lead to the loss of focus regarding the PhD work.

Sampling-based synthesis - Although many interesting research work is being done on other areas of music synthesis (e.g. spectral modeling or physical modeling), most current high-end virtual instruments that replicate existing instruments are sample based. As such, and considering the PhD student past experience on sampling solutions (see section 1.2 “Motivation”), the work will be based on sampling-based synthesis.

1.4 Applications

The concept of using a human voice to control a singing synthesizer can have multiple applications.

Synthesizer User Interface – Singing synthesizers are becoming better, increasing the freedom of the user in terms of what the synthesizer can do. But this also increases the complexity of handling them, since there are too many parameters and choices to choose from. By controlling the synthesizer directly with the human voice, the job of the user is simplified, giving direct information on how the synthesizer should perform, and still giving the user the chance to make minor adjustment later on.

Audio Transformation Tool (Audio Effect) – Since the internal synthesizer is controlled by an audio stream, the system can be considered as an audio transformation tool or an audio effect, whose output is a transformed version of the input. In this scenario, the synthesis is not the goal, but a tool to achieve the goal – an audio transformation process – transforming an audio stream into another, allowing the change of timbre characteristics as: gender (e.g. male to female), age (e.g. adult to child), number of voices (e.g. solo to choir), style (e.g. normal to operatic voice), singer (e.g. from singer A to singer B), etc.

Deceased people performance – With the use of old recordings audio material as the internal sound library, new songs can be recreated with the voice of a deceased singer (e.g. Pavarotti, Sinatra). By using a living singer and replacing his/her performance with someone's old recordings, new audio material is created.

Although the main focus of this PhD work is singing voice, the same principles can be applied to any other musical applications, creating a new paradigm in audio effects. As such, a new type of audio effects, based on sampled audio material, can appear in the future, taking audio as input and replacing it with "similar" audio material. This could lead to a new world of applications:

Restoration – By accessing old recordings material, that presents degraded audio quality, and recreating the same music performance using high-end sample libraries. For instance, a 1940 recording of a solo violin piece, that is replicated using a sound library with a Stradivarius violin *samples*, recorded on a high quality concert hall, with high definition audio quality – same musical performance, but a much higher sonic quality.

Transynthesis – Transforming an instrument into another [Chang, 2006]. Either to achieve creative effects (e.g. using a violin to control a harpsichord sound, allowing *portamento* effects that the original harpsichord instrument does not allow), or as an orchestration tool (e.g. a trumpet player that uses his/her trumpet to create the music lines of each instrument of an orchestra passage), transynthesis can be an interesting application.

Instrument enhancer – Increasing the quality of an instrument recording, by overcoming the lack of sound quality of the instrument (e.g. plastic recorder[5] that is replaced with the sound of a custom-made wood recorder), or recording/environment limitations (e.g. low quality microphone, background noise).

Many of these applications depend on pitch extraction methods. Due to the low success rate of existing polyphonic pitch extraction methods, the use of monophonic audio recording may still be necessary during several years.

1.5 Main Contributions

The main contribution of this PhD work was the creation of a new method for singing voice resynthesis, allowing the computer to replicate a singing performance using an internal sound library with vocal *samples* (presented in section 4).

Other contributions were also made in the area of Sound and Music Computing:

- A new sample-based parametric synthesis method, mainly focused to the singing voice, allowing a finer control over pitch, loudness (Dynamics) and phonetic/timbre parameters (presented in section 4.2).

[5] Not a recording device, but the woodwind musical instrument.

- New developments on using evolutionary approaches, based on Genetic Algorithms (GA) for automatic music transcription and resynthesis (presented in section 3.3).
- Some developments on the performance optimization of applying Genetic Algorithms in sound processing applications (presented in section 3.4).
- A new metric to evaluate music transcription systems (monophonic or polyphonic), which better correlate with the human perception (presented in section 3.5).

1.5.1 Publications

Based on this PhD research, several works have been published, including 7 international review-based conference papers and 2 review-based national conference papers. Some of these works were reused or cited by other authors.

A journal submission was also done, currently waiting for its review.

During the same time period, some publications were also obtained, although not directly related with the PhD work, including 1 book, 1 journal white paper and 1 domestic conference paper.

Annex A presents the full list of publications and citations.

1.6 Outline of the Dissertation

The rest of this dissertation contains four chapters. Chapter 2 presents an overview of relevant background information, techniques and related state-of-the-art. Due to the multi-disciplinary area of Sound and Music Computing, introduction information is presented on related topics.

Chapter 3 proposes an initial resynthesis approach. Before entering the singing voice domain, some work was done on the resynthesis of piano recordings, based on evolutionary techniques (Genetic Algorithms): an initial note sequence that evolves over time to better match the original audio recording. Besides resynthesis, the chapter also presents work done on the optimization of Genetic Algorithms for music applications, and a new metric to evaluate music transcription systems.

Chapter 4 proposes a singing voice resynthesis system, and presents the work done during its development, such as: the creation of a new synthesis approach; the extraction of pitch from the original audio stream; the extraction and replication of sound intensity behavior (dynamics); and extraction and replication of phonetic content. It also presents an evaluation of the proposed resynthesis system done with human listening tests.

Chapter 5 completes the dissertation with the final conclusions and possible directions for future work.

Published material and related citations are presented in annex A.

2. Singing Voice: overview

2.1 Introduction

Most music produced and heard nowadays is centered on vocal performance, where the human (singing) voice appears as the leading music line. The singing voice has two important characteristics when compared to other musical instruments: it can handle text, allowing to transmit a message; and it is the most human related musical instrument, since it is built on the human body and is used by everyone since childhood.

This chapter presents an overview of the singing voice and voice-related techniques. Section 2.2 presents the human voice system; section 2.3 explores several existing approaches for synthesis, either for speech, music or singing; section 2.4 explores methods related to the transformation of audio, focusing on voice-related effects; and finally, section 2.5 presents several artificial intelligence related techniques used on signal processing.

2.2 The human voice

The human voice has a fundamental role in human life, since it is responsible for most of the communication between humans. Even today, with so many electronic forms of communication, the loss of the human voice, or even other mild or severe voice disorders, can easily lead to traumatic experiences. Voice is the natural acoustic support for speaking, singing, whispering, laughing and other related sounds. All these sounds are produced by the vocal organ, originated by an airstream created on the lungs that is processed by the vocal folds and modified by the vocal/nasal tract [Sundberg, 1987]. As such, we can divide the vocal organ in three different components: the breathing apparatus, the vocal folds and the vocal tract (see Table 2.1).

Table 2.1 – Vocal Organ

Function	Organ	Activity	Major Agents
Compressor	Lungs	Breathing	Abdomen and Diaphragm Muscles
Oscillator (source)	Vocal Folds	Phonation	Laryngeal Muscles, Aerodynamics
Resonator (filter)	Vocal Tract	Articulation	Lip, Tongue and Jaw Muscles

The lungs act as a compressor, using abdomen and diaphragm muscles to create an airstream. This airstream passes through the vocal folds at the glottis, where phonation takes place. Here, the vocal folds vibrate, opening and closing the passage to the airstream. This very quick movement creates an acoustic signal that excites the vocal and nasal tracts. However, between the moment this acoustic signal is generated at the glottis and the moment it is radiated to the acoustic space outside the mouth, the sound must travel through the vocal tract - the path between the vocal folds and the lips/nostrils (Figure 2.1). Due to the physical variations of the different sections inside the vocal tract, i.e., due to articulation, the acoustic signal suffers transformations, similar to the effect of a resonator (i.e. a filter). Since some of these sections can change its position or characteristics, a human is capable of controlling the way the voice sound is shaped.

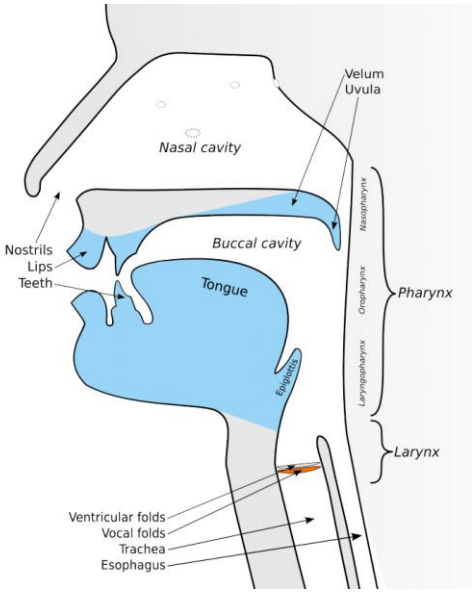

Figure 2.1 – Schematic view of a profile cut of the head (from [Degottex, 2010]

As long the vocal folds open and close the glottis at a constant time interval, the phonation will produce a periodic acoustic signal, designated as a voiced signal[6]. But this is not always the case. In some situations, the vocal folds might even not vibrate, with the airstream passing by, but due to turbulence effects, a noise-like acoustic signal is created, which is designated as unvoiced signal.

From the hearing point-of-view, it is important to understand what is the actual contribution of the phonation (vocal folds) and what is the contribution of articulation (vocal tract). Phonation will be responsible for two important behaviors: the choice between voiced/unvoiced sounds and, in the case of voiced sounds, the rate of the glottis vibration (fundamental frequency (F0) that is

[6] Characterized in the frequency domain by a series of harmonic partials at a decreasing amplitude of 12 dB per octave [Kent, 2004]

perceived as pitch). On a speech context, this frequency is used to express intonation[7], whereas in a singing context it is used to represent a melodic line.

On the other hand, articulation is responsible for shaping the sound in time and frequency, being the major responsible for the creation of phonemes. Although phonemes require a voiced or unvoiced phonation (for instance, all vowels are mainly voiced sounds), articulation is the main responsible for the differences between phonemes.

Figure 2.2 (from [Degottex, 2010]) presents an interesting schematic view of the voice production models.

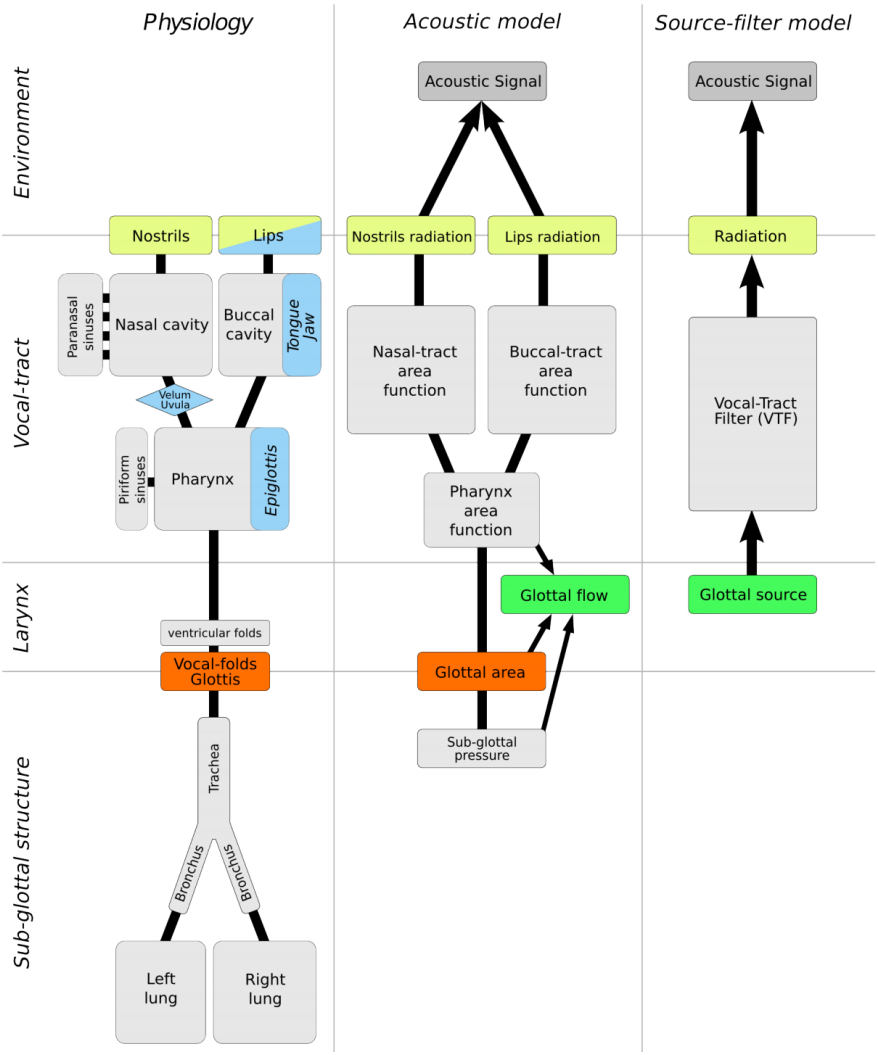

Figure 2.2 – Schematic view of voice production models (from [Degottex, 2010])

[7] For instance, if the pitch raises at the end of a sequence of words, it is probably indicative of a question.

2.2.1 Speaking vs. singing

There are important differences between the use of voice on a speech context and its use on a singing context.

Speech is characterized by a very smooth and continuous change of pitch, while singing uses a much more "jumping" behavior, since pitch is perceived as music notes. Also, singing (especially classic singing) is characterized by the usual presence of *vibrato* effects - a slight modulation of pitch, between 5 and 7 Hz, with up to ±2 semitones of pitch modulation [Kent, 2004]).

Another important difference between speaking and singing is the pitch range. Singing uses a much wider range of pitches, especially in higher frequencies. In some note/phoneme combinations, the fundamental frequency of the note might even be higher than the typical value of that phoneme first formant (F1). In those situations, the singer changes his/her F1 position (for instance, by opening the jaw) to move the first formant frequency above the F0 value [Kent, 2004], increasing its intelligibility. As such, the typical formant locations of speech phonemes may not apply to the singing voice.

Regarding the use of voicing, it is known that a singing voice uses much more voiced sounds than speaking voice (90% vs. 60% [Cook, 1990]), mainly due to the increase of vowel durations.

One interesting characteristic of singing, especially present on classic trained singers[8], is the existence of a "singer" formant [Sundberg, 1974], i.e., a special resonance of the vocal organ that allows the singer to increase the vocal loudness[9] at a particular frequency range (between 2.5 kHz and 5 kHz, depending on the singer), by clustering formants together (F3, F4 and even F5). This technique allows them to better detach themselves from the instrumental background (e.g. opera). Interestingly, this "formant" is only used for singing, and not with other vocal sounds.

It is also important to mention that sometimes the frontier between speaking and singing is not very clear, especially in some vocal performances as poetry, chant or rap music [Gerhard, 2003].

2.3 Synthesis

Synthesis, within an audio context, is the process of creating a new audio stream, usually based on symbolic or high level information, and it plays an important role in many music or speech applications. To better understand the process of singing resynthesis, it is important to comprehend the current approaches in music, speech and singing synthesis.

2.3.1 Musical sound synthesis

Musical sound synthesis can probably be split in two main classes: parametric synthesis and concatenative synthesis [Schwarz, 2004], as illustrated in Figure 2.3. Parametric synthesis (also

[8] Especially male singers and altos [Kent, 2004].

[9] Up to 20 dB higher than "normal" speech situations [Sundberg, 1987]

know as *Model based synthesis*) is based on models/parameters of the sound; concatenative synthesis is based on existing audio material.

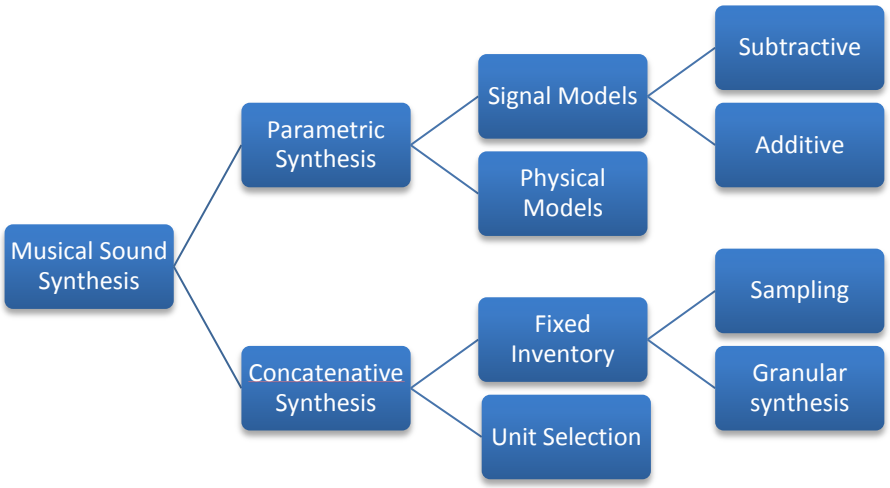

Figure 2.3 – Musical Sound Synthesis

Parametric synthesis can be subdivided in physical models and signal models. On physical modeling systems, the desired sound is obtained by modeling the physical characteristics of the instrument. For instance, to recreate a guitar sound, the idea is to model the behavior of a vibrating string, considering its mass, length, the acoustic response of the guitar resonance, among other physical behaviors. In signal systems, with well known approaches as additive synthesis or subtractive synthesis, the synthesis is obtained by using signal models of the sound. In the additive synthesis, the major goal is to add several single frequencies as a way to create the complex musical sound, by controlling the amplitude and frequency of each one. Subtractive synthesis starts with a complex waveform (e.g. a sawtooth signal) with many harmonics that passes by several filters that remove the unwanted harmonics.

Concatenative synthesis uses pre-recorded material (also known as *samples*[10]) to obtain the final audio stream. Granular synthesis uses very short *samples*, which will act as a sound grain, whereas Sampling uses longer recorded *samples* that recreate the original musical instrument. In Unit Selection systems, instead of a fixed inventory, audio fragments (units) are selected depending on the required target sound.

Although the main concepts of musical sound synthesis can be classified on such classes, some hybrid techniques may contain a mixed approach. For instance, Spectral Modeling Synthesis (SMS) [Serra, 1990], that may be considered both additive and subtractive synthesis by modeling sound as a combination of harmonic content (created by adding sinusoids as harmonics) and noise/residual content (created by filtering white noise with filters), can also use pre-recorded audio as the model baseline data.

[10] To prevent some confusion between the concepts of sample as a time instant value (digital processing context), and as a pre-recorded audio stream (music synthesis context), we will represent the second one with an italic format.

2.3.2 Speech synthesis

The first references to speech synthesis began in 1780[11] – von Kempelen has shown that the human voice could be modeled, creating a mechanical device that probably would be the first voice synthesizer of the world (Figure 2.4).

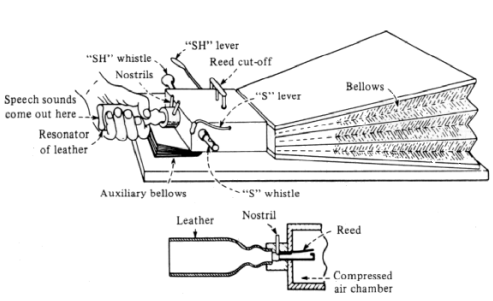

Figure 2.4 – von Kempelen's speaking machine

The next important milestone regarding voice synthesis was the presentation on the World Fair in S. Francisco in 1939, of the "Voder", a voice synthesizer created by Homer Dudley [Gold2000] (Figure 2.5).

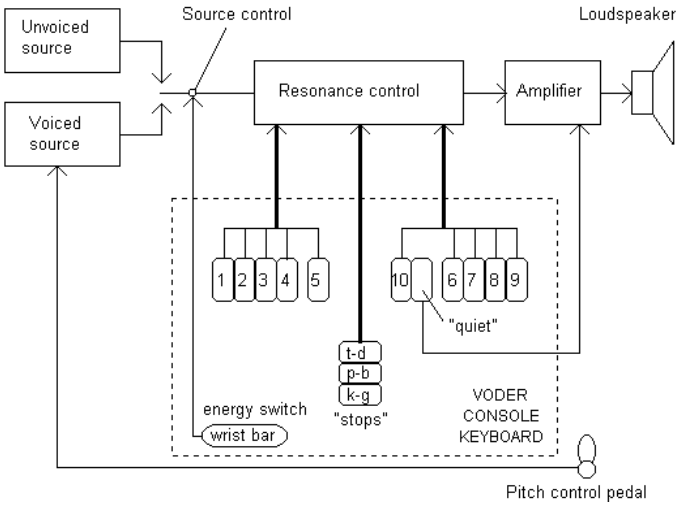

Figure 2.5 – Voder

With the appearance of digital electronics, and later, of computers, the technologies for voice synthesis have improved not only in terms of quality but also in terms of diversity of existing solutions. Nowadays, speech synthesis can be divided almost in the same groups as musical sound synthesis, as illustrated in Figure 2.6 [Schwarz, 2004].

[11] Although there was already some previous work, it was very limited and was mainly focused on vowels.

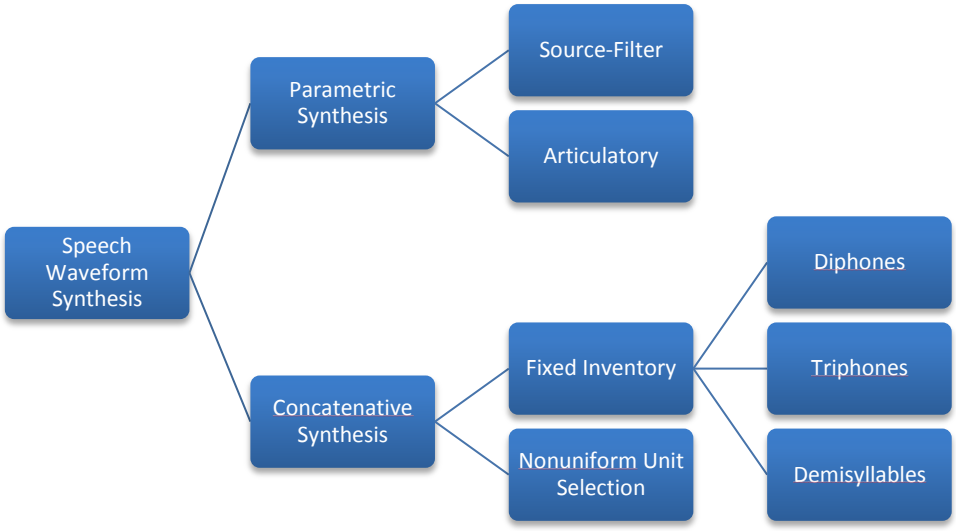

Figure 2.6 – Speech waveform synthesis

On the parametric side, source-filter synthesis (also known as *formant synthesis*) considers the human voice as a source-filter system, combining glottal pulse excitation (i.e. the source) and resonances of the vocal tract (i.e. the filter) [Holmes, 1983], similarly to a subtractive approach. Articulatory systems are based on the physical characteristics of organs producing the voice (i.e. physical modeling) [Shadle, 2002], using parameters directly related to the physical characteristics of the human vocal organ.

In concatenative systems, pre-recorded material is used. In fixed inventory, short audio fragments of audio are used, by dividing the audio material in diphones, triphones or demisyllables, whereas nonuniform unit selection allows the use of any fragment size (from single phoneme to entire words), as long it proves to be the best solution for the desired target sound.

2.3.3 Singing voice synthesis

Given that the application field for singing voice is music, most of the existing research and solutions came from the music research world. With the appearance of the electronic music synthesizers, people began trying to simulate the singing voice, although initially with only a single phoneme sound (e.g. “ah”, “oh”), allowing musicians to play melodies with a “vocal” kind of sound, but without any text. The ability to sing text would take several years to be implemented.

Singing voice synthesis can use almost the same segmentation as speech or music synthesis, with parametric approaches or concatenative approaches, although most often hybrid techniques are used.

The formant based CHANT project [Rodet, 1984], although expanded to general synthesis, was originally conceived for singing voice, based on Formant Wave Functions (FOF - *Forme d’onde formatique*) (Figure 2.7).

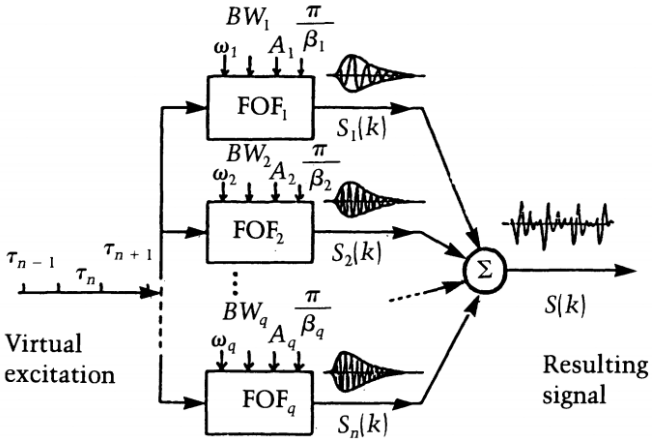

Figure 2.7 – FOF synthesizer [Rodet, 1984]

In SPASM [Cook, 1992] (Figure 2.8) a physical modeling approach is used, based on Digital Waveguide Filters (WGF) with shape parameters. The nasal tract is also based on WGF and glottal source pulses are stored and retrieved from wavetables. A noise component is also present (filtered pulse noise) to simulate the turbulence generated as air flows though the oscillating vocal folds.

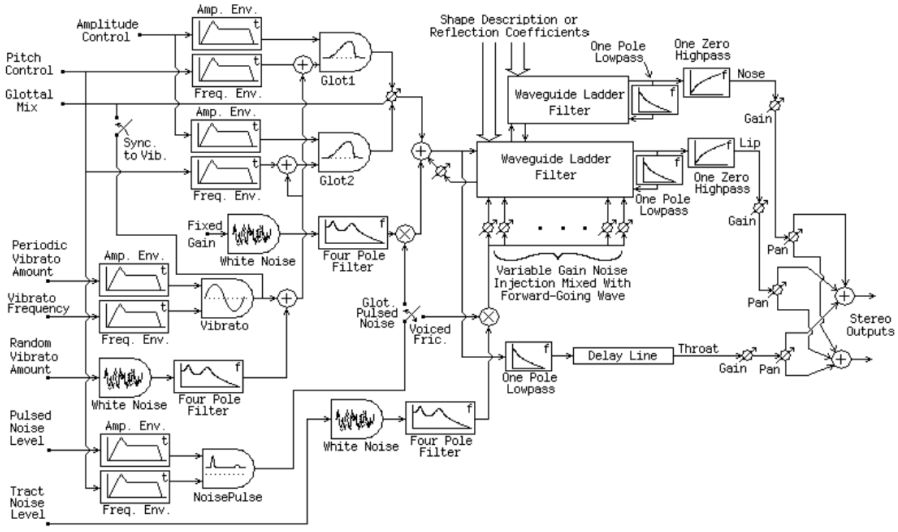

Figure 2.8 – SPASM [Cook, 1992]

The LYRICOS system [Macon, 1997] uses sinusoidal models based on pre-recorded material. It merges concatenation and unit selection with sinusoidal models that synthesize the output signal using overlap-add (OLA) (Figure 2.9). The system first selects the vowels and, on a second pass, fills up the consonants. Later, FLINGER (Festival Singer) [Macon, 2000] has taken the concepts of LYRICOS, while using the Festival TTS as its engine.

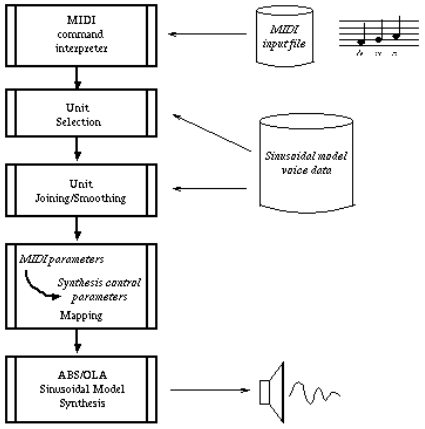

Figure 2.9 – LYRICOS [Macon, 1997]

In [Fonseca, 2003a], the author of this dissertation proposes a simple but effective concatenative approach for choir singing synthesis, with a pure sampling approach and without signal processing, other then simple cross-fades. By receiving MIDI information with the melody, and looking at the text and time editor's information, the engine activates or schedules *samples* on an external sampler (Figure 2.10).

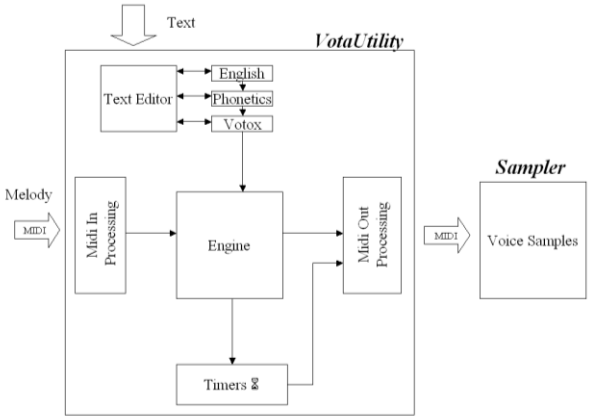

Figure 2.10 – VotA Utility engine [Fonseca, 2003a]

A significant milestone is the work of Jordi Bonada, by using performance sampling and spectral models [Bonada, 2007][Bonada, 2008]. Pre-recorded audio is used as models, and spectral processing allows its modification and concatenation. Figure 2.11 presents a simple vowel EpR (Excitation plus Resonances) synthesizer [Bonada, 2008] (the final system considers vowels and consoants). The system considers a sonic space as presented in Figure 2.12. During synthesis, information from several templates are combined (pitch/loudness/tempo/phonetic (A) + vibrato (B) + articulation (C)) to create the singing output.

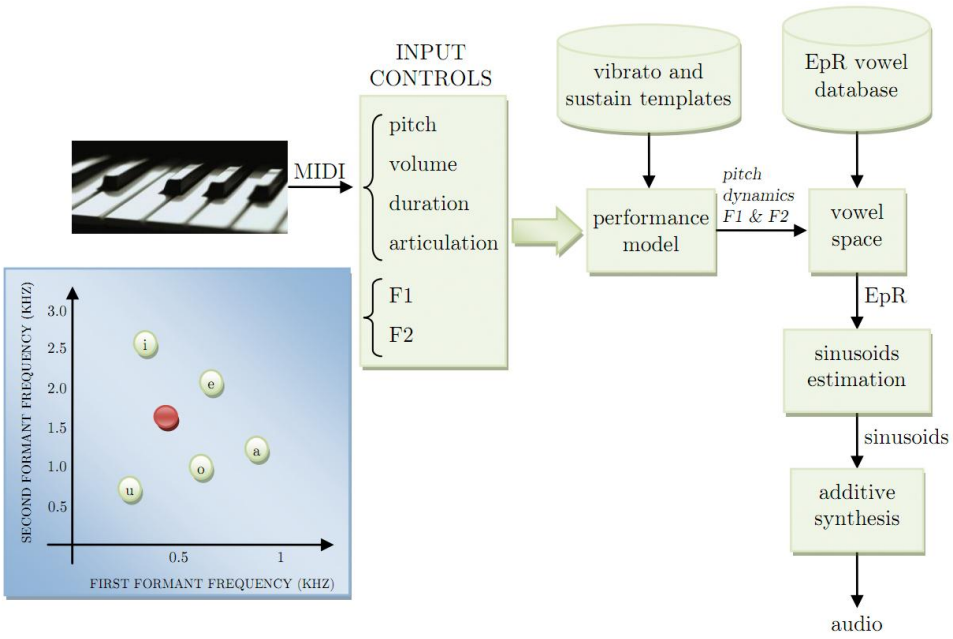

Figure 2.11 – A vowel synthesizer from [Bonada, 2008].

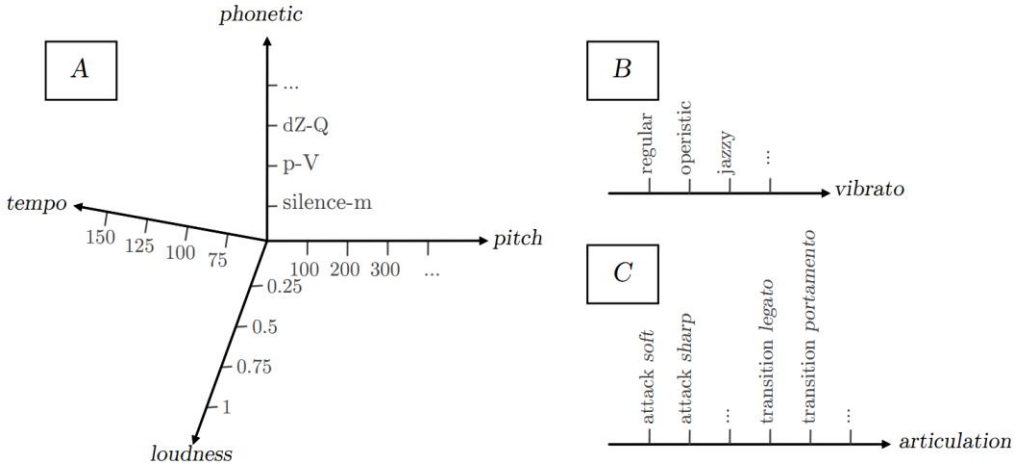

Figure 2.12 – Proposed subspaces of the singing voice sonic space [Bonada, 2008]

The commercial availability of research work is sometimes an interesting measure of its quality. The first commercially available singing synthesizer, with text building features, was probably "Voices of the Apocalypse" (VotA) [Eastwest, 2001] with its "VotA Utility", developed by the author of this dissertation [Fonseca,2002] and released in 2002, and it was able to recreate a choir sound with real-time support. Later "EW/QL Symphonic Choirs" and its "WordBuilder" software [Eastwest, 2005], also based on the work of [Fonseca, 2002] were released (also for choir), almost at the same time as Vocaloid [Yamaha, 2004], which was based on the work of Jordi Bonada and Xavier Serra [Bonada, 2007]. The Vocaloid engine was used on several sound libraries, mainly for solo voice. In Vocaloid 2, new features were added including real-time support. Another commercial solution, also available for solo voice, is "Virsyn Cantor" [Virsyn, 2004], although its internal concepts are not known. There are other virtual instruments with a limited number of

recorded syllables [VSL, 2010][Cinesamples, 2010], but they only allow a very small set of available words that can be sung.

It is also important to mention how user interface works with singing synthesizers, and what can be done to improve the interaction between man and machine.

For text singing, the virtual instrument requires both music information (e.g. musical notes) and text information (e.g. sequence of phonemes). Some systems, such as Vocaloid [Yamaha, 2004], handle both music and text together, being text just one additional characteristic of each note (Figure 2.13 – left). Other systems, such as Symphonic Choirs [Eastwest, 2005], use a text only editor (Figure 2.13 – right), with music information being playback by the DAW software or delivered in real-time.

Figure 2.13 – User interface for Vocaloid (left) and WordBuilder (Symphonic Choirs) (right)

Since most users do not have experience with phonetic alphabets, most solutions include phonetic translating tools, allowing users to enter English text that is internally converted to its phonetic equivalent, and still allowing the user to tweak the desired phonetic sound.

Although current singing synthesizers present interesting results with basic information (sequence of notes and words), these results are very unrealistic and unnatural, forcing the user to spend a lot of time making small adjustments, both regarding music information (e.g. adding vibratos or changing pitch attacks) and phonetic information (e.g. replacing phonemes or editing phoneme durations or intensities). Realistic or near-realistic results are usually possible (especially in the case of English, since almost used phonemes are covered), but it represents a very time consuming task, even for experienced users.

One of the most interesting solutions for such problem is the use of the human voice as an interface with the virtual instrument. By using his/her voice, the user could easily specify how such track could be sung.

The concept of using a human voice to control a synthesizer is not new. In [Janer, 2004] several examples are mentioned. In [Janer, 2006], the human voice is used to control a singing

synthesizer, but phonetic information must be manually inserted. In [Nakano, 2009], a similar system is proposed, where a singing voice is also used as the main source of information to control a singing synthesizer. Unfortunately, the system also requires the user to manually enter the phoneme sequence. Both systems extract pitch and dynamics information, and are able to align the manually entered phoneme sequence with the singing input stream.

Systems that allow the human voice to fully control a singing synthesizer (including phonetic information) are currently unavailable.

2.4 Audio transformation

Transforming an audio stream into another is a common situation in the life of any audio-related professional, especially when using digital audio effects. Modifying the sound characteristics of the input audio stream is the main goal of digital audio effects [Zolzer, 2002]. Effects as delay, reverb, equalization (EQ), dynamic processing and others, are widely used and usually handle any type of audio material. When it comes to processing the human voice, either speech or singing, all these generic audio effects continue to be used by professionals, but usually they are not able to modify specific voice characteristics.

To change voice characteristics, several techniques exist. One of the most common known examples of voice transformation is the vocoding [Zolzer, 2002], used for creating robot voices and similar effects. This effect takes two inputs and it creates a combination of both streams, by "spectrally shaping" the first sound by the second one, while preserving the pitch of the first sound. It is almost like applying the vocal tract of the second sound to the first sound.

Other common voice transformation effects are based on source-filter concepts and techniques [Verfaille, 2004][Robel, 2010]. The same way that human voice is produced by phonation (acting as source) and articulation (acting as filter), source-filter based effects try to separate the two components and change them independently. A common application is pitch-shifting with formant preservation. Traditional pitch-shifting techniques are able to change pitch, but will also change the location of formants, since all frequency information is "moved", i.e., equally displaced. For instance, increasing the pitch of a male voice can easily result on a sound similar to a female, child or even cartoon-type of voice, since the characteristics of his vocal tract are changed, acting as if the dimensions of the vocal tract were reduced. Source-filter approaches allow the pitch to be shifted while keeping the same vocal-tract (filter) characteristics, or vice-versa, by keeping the same pitch while changing the overall characteristics of the vocal tract. Examples of several representative methods can be found in [Zolzer, 2011].

Changing the number of perceived voices (e.g. changing a solo voice into a choir) is also a common voice transformation goal: from techniques that overlap several variations of the input stream, to traditional chorus effects (that use delay modulation to recreate the effects of slightly out-of-tune voices), or even more complex methods such as spectral processing [Bonada, 2005]. In fact, spectral processing has been responsible for many different techniques of voice-oriented effects [Mayor, 2009].

Other approach for transforming audio streams is resynthesis (the act of creating a new synthesis) - using an input audio stream and recreating a new stream based on the previous one. Resynthesis usually involves a two-step approach: first a feature extraction module, allowing to extract the relevant information from the original audio, and then a synthesis module, allowing to synthesize an audio stream based on the extracted information. The synthesis process of the Resynthesis can use any of the previously mentioned synthesis approaches. On the parametric side, some work was been done with additive synthesis [Klingbeil, 2005], additive/subtractive synthesis [Chang, 2006], spectral models [Bonada, 2005]. On the concatenative side, there is the work of [Schwarz, 2006], [Cardle, 2003], [Hoskinson, 2001], among others.

Some authors make a separation between Resynthesis and Transynthesis [Chang, 2006], considering Resynthesis when keeping the same type of timbre (trumpet → trumpet), and Transynthesis when timbre is not maintained (trumpet → violin).

When Resynthesis is implemented with concatenative techniques, it may be referred to as mosaic or similar terms (e.g. audio mosaicing) [Schwarz, 2005]. Following the concept of image mosaics, where images are created using small fragments of existing images, audio mosaics (or mosaicing) is the concept of concatenating audio fragments, considering a desired target. The desired target can be based on an existing audio recording or it could be a high level description given by the user.

The difference between concatenative synthesis and audio mosaic is essentially a difference of context. Although they share the same concepts, concatenative synthesis has the focus on synthesis (creating better synthesizers), whereas audio mosaic has the focus on the mosaic effect (not a perfect synthesis, but an interesting one).

2.5 Artificial Intelligence methods for signal processing

Significant research work has carried out by applying Artificial Intelligence methods to sound-related research. Due to its use in this dissertation, some approaches are presented in the next sections.

2.5.1 Neural Networks

A Neural Network (NN), also known as Artificial Neural Network (ANN) is a computational intelligence technique that is based on the concept of the human brain and its neural connections.

Figure 2.14 presents a simple perceptron, the first and simplest type of NN [Rosenblatt, 1958]. Considering inputs x_1 … x_n, the output is given by equation 2.1 and equation 2.2, where w_0…w_n represent the internal weights of the NN that need to be adjusted during training. Instead of a binary output, a continuous output can be used, by changing the activation function (e.g. tanh(x) instead of sign(x)).

$$o(x_1, \dots, x_n) = sign(w_0 + w_1 x_1 + w_2 x_2 + \ \dots + \ w_n x_n) \quad (2.1)$$

$$sign(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases} \quad (2.2)$$

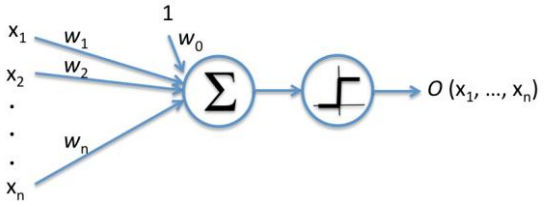

Figure 2.14 – A simple perceptron

The simple perceptron is very limited, but, by using several perceptrons (nodes) organized on more than one layer (Figure 2.15), highly complex systems can be obtained [Mitchell, 1997]. During training, the internal weights are adjusted to minimize the output error[12], by using Backpropagation or any other NN learning method[13].

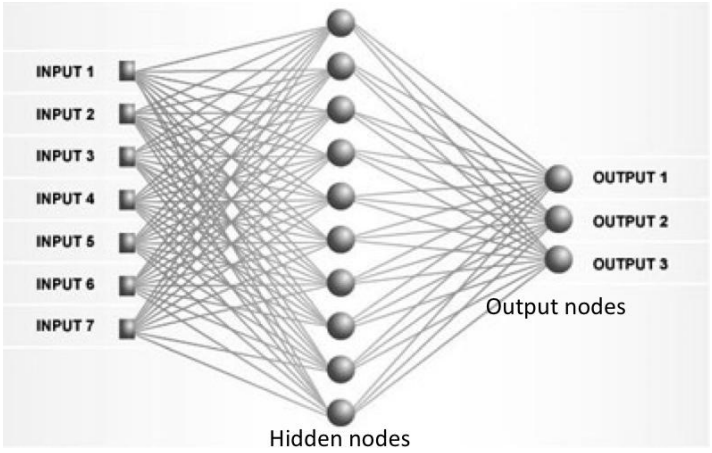

Figure 2.15 – MultiLayer Perceptron (MLP)

Other types of neural networks also exist [Haykin, 1994]. For instance, in recurrent neural networks, outputs or hidden nodes values act as inputs. This feedback connection allows the neural network behavior to be dependent of previous results, acting as a memory feature.

Self Organized Maps (SOM) [Kohonem, 1984] is a special neural network with unsupervised training. Contrary to supervised training approaches, where the system is fed with several training examples consisting of input data and the system's correct answer (target) for each situation, in unsupervised learning the target solution is not given or known. As such, the system must be able to improve its behavior, not directly, but indirectly, by using features like joining similar examples. In the case of self-organized maps, the neural network presents several outputs (for instance, representing a 2D mesh), and in each situation, only one output is activated, corresponding to the

[12] Difference between the output value and desired value.

[13] Backpropagation and most other NN learning methods are supervised, which means that the desired target is known during training.

output with the higher value ("the winner takes it all"). During the training, the internal weights are changed so that similar input data could fire the same outputs, or at least outputs located near each other. During the test phase, the output position can be used as data distance parameters, i.e., the distance between the outputs of two test data vectors can be used to measure the similarity between them. One of the first applications of such systems was in phonetics, where a SOM is used as a phonetic typewriter [Kohonem, 1988].

2.5.2 SVM

Support Vector Machines (SVM) is a learning approach based on statistical learning theory. This relatively new approach, introduced by Vapnik and co-workers [Boser, 1992] [Vapnik, 1998] is a very powerful method that outperforms most alternative systems in a variety of applications [Cristianini, 2000].

SVM uses a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory based on statistical learning theory. For instance, on an example as the one illustrated in the Figure 2.16, to classify a 2-class problem within a 2D space, some other learning method could stop their training with a situation similar to that illustrated in Figure 2.10 A, since all training examples get divided on their own classes (Figure 2.16 A). SVM, not only try to separate classes, but also try to find the hyperplane that maximizes the separation margin between classes (Figure 2.16 B). As such, some training examples are chosen as support vectors (the ones crossed by the dash lines in Figure 2.16 B).

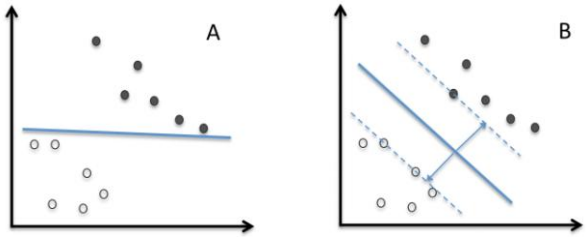

Figure 2.16 – Classification on a 2-class problem within a 2D space: some learning method (A) vs. SVM (B)

Since it is not always possible to use a hyperplane to divide classes on a linear space, kernel approaches can be used, transforming the data to a space where hyperplane separation is possible [Cristianini, 2000].

Although SVM uses a binary classification approach (only 2 classes), multiclass SVM can be easily obtained by dividing a k-class problem in binary approaches (e.g. one-against-all, pairwise classification) [Aisen, 2006].

2.5.3 HMM

Hidden Markov Model (HMM) is a statistical Markov model, but with hidden states [Rabiner, 1989]. A Markov chain (the simplest Markov model) models a system with a finite number of states (S_1… S_n), where the next state only depends on the current state and not the previous ones (Figure 2.17). The probability that the system changes from state *i* to state *j*, is given by a state

transition probability (a_{ij}). Unfortunately, in most systems that need to be modelled, the states are not directly visible. What is visible is what the state produces, that are designated as observations (v_i). The probability that a state *i* produces an observation *j*, is given by observation probabilities (b_{ij}).

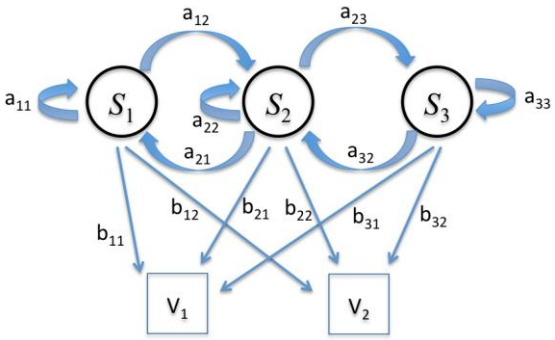

Figure 2.17 – Hidden Markov Model, with states (S), state transition probabilities (a), observations (v) and observation probabilities (b)

To create the model, the system tries to find the state transition probabilities and the observation probabilities that best models the training sequence(s). After this learning process, the model can be used. Given a sequence of observations (e.g. a signal), the system will find the best sequence of states that could present such observations (usually done with a Viterbi Algorithm [Rabiner, 1989]). For instance, on a phonetic recognizer where HMM states represent phonemes, the sequence of states correspond to the sequence of phonemes.

HMM are well known for their application on voice recognition [Becchetti, 1999], due to their ability to handle sequences of data.

2.5.4 Search approaches

When searching for a solution to a problem, several approaches can be used. If the search space[14] is relatively small, a full search can be made, by testing all available solutions and choosing the best one. Unfortunately, most problems present huge search spaces, preventing the use of a full search.

One possible approach to deal with huge search spaces is the use of heuristics. By creating a set of rules or a set of steps, the programmer creates a method that allows the computer to come up with a solution that hopefully will be near the optimum solution, in a small amount of time (when compared to a full search) by looking to a subset of the search space.

Another approach is to use local search methods, like Hill Climbing, on which a (random) candidate solution "looks" in its neighborhood for better solutions, improving its position iteration after

[14] The space spanning the universe of possible solutions

iteration until a maximum is obtained. Although it is quite effective in some kind of problems, it has the drawback of being stuck at local maxima, preventing it to find the global maximum.

A set of search approaches that are currently receiving particular attention from the research community are evolutionary approaches. Starting with a set of candidate solutions (in most cases, random solutions), several operations are created allowing those solutions to evolve iteration after iteration, searching for the best solution. Within evolutionary approaches, there are several well-known techniques like: genetic algorithms [Holland, 1992], memetic algorithms [Hart, 2004], Particle Swarm Optimization (PSO) [Kennedy, 1995], among others.

The concept of genetic algorithms [Holland, 1992] is based on the idea of natural evolution from Darwin. Better-prepared individuals are more likely to survive and reproduce more often, creating new individuals that will likely have the relevant genetic code of their parents. After several generations, the quality of the population increases.

From the computer science point-of-view, the idea is to start with an initial population of individuals, where each individual represents a possible solution to the problem. Generation after generation (iteration), individuals are evaluated and chosen as parents (selection) depending on their fitness (score): more fitted individuals will more likely be chosen as parents than less fitted individuals. By combining two different individuals, new individuals are created (recombination), that share their parents genes (information) and that are also susceptible of minor changes (mutation). After several generations, the best individual can be chosen as a solution to the problem.

Memetic algorithms [Hart, 2004] can be seen as an extension of Genetic algorithms that use local search features. For instance, after recombination, several different mutations can be tested and evaluated. Although Genetic Algorithms are known for their capabilities in finding good areas on huge search spaces, their capability for fine tuning their solution are not so great. Memetic algorithms allow a better local convergence, and in some specific practical problems are know for their success [Hart, 2004].

Swarm intelligence is an evolutionary approach that is based on the behavior of some social animals (e.g. antz, birds, fishes) that mix individual behavior with a group behavior. In the case of Particle Swarm Optimization (PSO) [Kennedy, 1995], a set of particles (each particle is a candidate solution) move around the search space where the direction/velocity depends on three factors: the global best (the best solution ever found by any particle), the neighborhood best (obtained by communicating with a subset of all particles) and the local best (the best solution ever found by that particle).

2.6 Summary

This chapter presented an overview of the singing voice, focusing both on the human vocal organ and on the synthesis of speech and singing materials. Initially, a brief presentation of the human vocal organ is presented, analyzing the differences between speaking and singing.

Later, several approaches were mentioned regarding the synthesis of speech, generic musical sounds, and singing, which will have an important role on this dissertation work. Besides synthesis, this chapter also presents a very summarized overview of the audio transformation of vocal sounds.

Finally, a small summary is presented regarding the use of some Artificial Intelligence approaches in signal processing, to better contextualize some of the work done in this dissertation.

3. Evolutionary Resynthesis of Musical Instruments using Genetic Algorithms

3.1 Introduction

Although the main goal of this PhD work is to achieve singing voice resynthesis, our work will begin by focusing on a simpler scenario – the resynthesis of a musical instrument. By disregarding the phonetic complexity, a simpler resynthesis approach can be tested and evaluated.

Of all available instruments, the piano seems an interesting choice: it is probably one of the most studied musical instruments and much research work is currently being done on analysis of piano recordings (mainly for polyphonic music transcription). From the technical point of view, it also presents some simplifications, especially regarding synthesis. Although the part of being a polyphonic[15] music instrument increases the analysis complexity, the synthesis results simplified since each note is independent of any other and there is no need to synthesize note transitions. By using a sampling approach, good synthesis results can be achieved by simply adding notes *samples* on top of each other.

This chapter presents the work being done trying to resynthesize piano audio recordings. Section 3.2 explores possible approaches to re-synthesis; section 3.3 explores a search space approach using Genetic Algorithms and presents relevant work done in this area; section 3.4 explores methods proposed to decrease the computational power required by such approaches; section 3.5 explores some side research done in the area of music transcription metrics; and finally, section 3.6 presents the final conclusions of the chapter.

It is also important to mention that some of the work presented in this chapter (section 3.3 and 3.4.1) was a collaborative research with another PhD student (Gustavo Reis) from Universidad de Extremadura, Spain.

3.2 Resynthesis

The concept of resynthesis can be achieved with two different approaches: a serial approach and a parallel approach. The serial approach consists of two independent modules: the feature

[15] able to play several notes at the same time

extraction (analysis) and the synthesis. First, a feature extraction process will extract, from the original audio stream, the relevant symbolic/semantic information (e.g. pitch, dynamics, phonemes when working with singing voice, etc.). With all that information, the synthesizer will then be able to recreate a similar audio stream. In this approach, the feature extraction module and the synthesizer module do not interfere with each other and are completely independent, it is just necessary that they agree on the information interface between them (for instance, using standard MIDI information). This means that it is possible to change one of the modules without affecting the other. Also, most of the current research can be applied individually to each module. For instance, state-of-the-art music transcription system can be applied to the first module, or a state-of-the-art music synthesis can be applied to the second module. Due to this module independency, we will designate it as a serial approach (Figure 3.1).

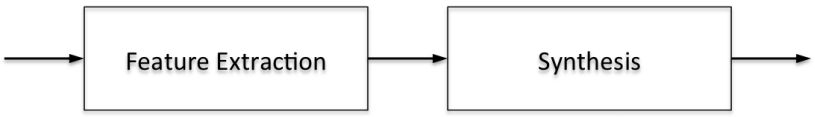

Figure 3.1 – A serial resynthesis approach

The other possible approach for resynthesis is based on the idea of integrating analysis and synthesis altogether, which means that analysis is no longer done independently. In this approach, that we will designate as parallel (Figure 3.2), analysis is mainly done by comparing the original audio with the synthesized one, looking for the best way to resynthesize a given sound, considering the existing synthesizer and its characteristics.

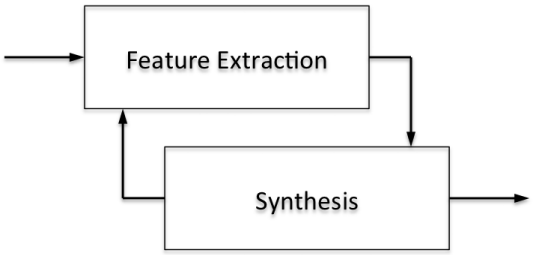

Figure 3.2 – A parallel resynthesis approach

Although widely used in many engineering fields such as speed/audio compression and adaptative filtering, this approach is much less explored by the research community devoted to music resynthesis, not due to worst results, but simply because the area of application is less common (many applications require only the analysis methods or only the synthesis methods). Also, it can be seen as an indirect approach (almost like reverse engineering) to the problem.

One interesting feature of this parallel approach is that if we output the analysis results instead of the synthesis results (without changing the internal structure), we change our resynthesis system into a music transcription system based on an analysis-by-synthesis approach, i.e., a music transcription system that uses an internal synthesis as a way to improve its analysis capabilities. Either as a resynthesis system or as an analysis-by-synthesis music transcription system, the

internal structure is exactly the same. The only difference is outputting the results from the internal synthesizer module or the internal analysis module.

3.3 A search space approach to resynthesis using Genetic Algorithms

It is possible to look at resynthesis as a search space problem, by searching for the best sequence of information that allows us to synthesize a similar audio stream. This last sentence highlights the three important concepts: search, synthesis and similarity. By using a search space method, the **analysis/synthesis** problems are converted on **search/synthesis/similarity** problems. Since piano synthesis can be easily achieved with sampling methods (by playing pre-recorded "samples" of each piano note) [Nemesys, 1998], the main remaining problems are **searching** and **similarity**.

To resynthesize a piano recording, we will want to search for the note sequence that best models (presents the best musical similarity with) the original audio recording. But this search-space approach has a major handicap – the size of its search space. Since we are dealing with so many complex dimensions (time, pitch that can be polyphonic, dynamics), the search space will be enormous. As such, the decision was made of using Genetic algorithms.

Genetic Algorithms (GA)[16] is a search space technique that is known for its ability to work on large search-spaces, by using only a very small subset of that search-space. In this case, the GA would be searching for the best sequence of notes that, by passing through the synthesizer, would create the most similar audio stream. Their use in a music transcription scenario was initially explored by [Garcia, 2001] and [Lu, 2006], although on simple situations.

But the use of Genetic Algorithms for this resynthesis task raises some questions:

- How should the information be encoded on the individuals (candidate solution)?
- How should *individuals* (*candidate solutions)* be evaluated?
- How should selection, recombination and mutation work?
- How do we create the initial population?

The following sections present information regarding the use of Genetic Algorithms in our resynthesis approach.

3.3.1 Encoding

Since each individual represents a candidate solution, it should be encoded as the sequence of events that will be presented to the synthesizer. As such, each individual will be a sequence of *n* note events (Figure 3.3). Since we do not know how many notes are present in the original audio recording, the number of note events is variable from individual to individual, and will probably change during evolution.

[16] More information in section 2.5.4

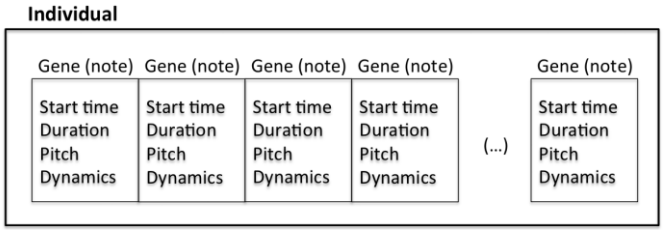

Figure 3.3 – Gene Encoding

Each note event will need the following information:

- Onset (the start time of the note)
- Duration of the note
- Note pitch
- Dynamics of the note (a.k.a. loudness, MIDI velocity)

This encoding allows us to represent a candidate solution for the problem, where each note event acts as a gene of the individual.

3.3.2 Selection, Recombination and Mutation

Genetic Algorithms use three main operations: Selection, Recombination and Mutation. The goal of Selection operation is to select parents to generate new individuals. The selection of parents should not be fully random, but based on the quality of existing individuals, i.e., the most fitted individuals[17] should have more probability to be selected as parents than less fitted individuals. But, since we want to overcome local maxima and explore different areas of the search space, we cannot use only the two best individuals to create all childs. From all traditional G.A. selection methods tested, we selected the method of Deterministic Tournament [Goldberg, 1991] with five individuals: five individuals from the population are randomly selected, and the one with the best fitness is selected as parent. This selection method offered good results and required less processing power than other alternative methods.

Unlike traditional GA implementations, that usually use a fixed number of genes, our encoding uses a variable number of genes (since we do not know how many notes exist on the audio stream), which will affect the recombination operator. So, instead of using the traditional n-point crossover, where one or more random points are chosen to split the parent's genes, we used what we called "one time point crossover" (Figure 3.4) - during recombination, a random point in time is chosen, splitting the parent's genes in two: the note events that occur before and the note events that occur after that time period. If a note is being played during that time point, the note event is split in two.

[17] Fitness measures the score of an individual regarding the problem. The most fitted individual is the one that presents the best solution for the problem.

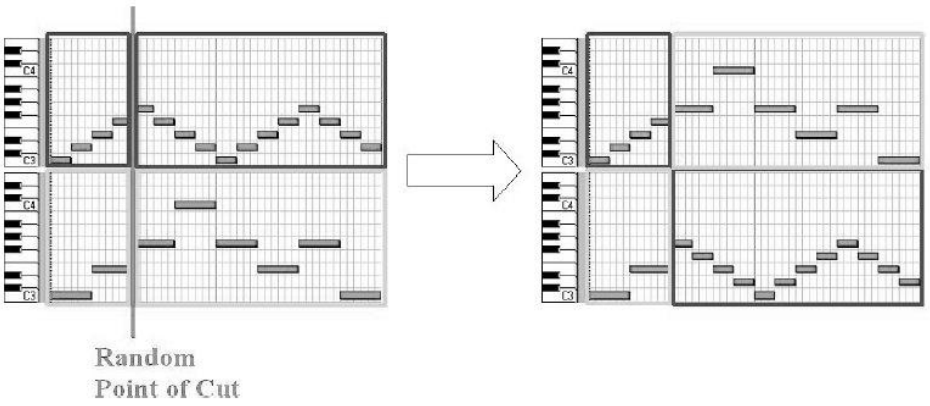

Figure 3.4 – Recombination

The mutation operator is responsible for adding slight changes to the individual genes, as a way to explore new areas of the search space that are not present at the parents genes, achieving a better search. In our system, instead of applying random bit changes, since our genes are very musical oriented (each gene have four musical parameters), we have decided to use music related parameter changes, like:

- Pitch shift – the note event changes its pitch value with ± 1 semitone.
- Octave shift – the note event changes its pitch value up to ± 2 octaves.
- Onset shift – the note events changes its onset value within a *x* second range.
- Duration – the note event increases or decreases its duration x%.
- Dynamics – the note event increase or decrease its dynamics *x*%.
- Note split – the note event is split on two with a random interval between them both.
- Note merge – two note events with the same pitch are merged into a single event.
- Delete event – the note event gets deleted.
- Random add – a new note is created. This note can be completely random or it can be a copy of an existing one with a different pitch.

3.3.3 Initial Population

Any Genetic Algorithm needs an initial population to start with. Although in most G.A. applications the system is initialized with a random population, which is good to explore completely different areas of the search space, presenting an initial population that is somehow related with the final solution improves immensely the processing time. As such, our initial population will not be randomly created, but created considering the FFT peaks of the original audio.

Using the original audio stream, for each time frame an FFT is calculated, the peak with the highest value is obtained. Based on the frequency of that peak, a note event is created (or a previous one is maintained, by increasing its duration) (Figure 3.5).

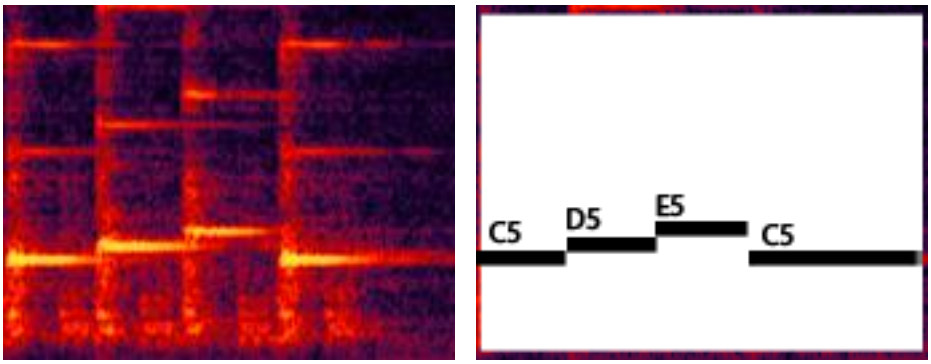

Figure 3.5 – Example of an original audio spectrogram and its initial individual

The use of FFT will, of course, create many errors. In most cases it can even present an error rate of 100% (e.g. missing fundamental, etc). Nevertheless, this is not an issue, since we are only creating an initial population. The Genetic Algorithm will be responsible to evolve this initial solution to a better one. Having an initial solution that is closer to the right solution than a pure random one, will result on a highly improvement of required computational power, since much less generations will be needed to achieve a similar performance.

Other approaches were also tested, but did not produce better results (worst precision and recall results):

- *k* FFT peaks – instead of using only the highest peak on each time frame, the system considers the *k* highest peaks, being *k* a user defined constant.
- Threshold FFT peaks – instead of using only the highest peak on each time frame, the system consider all peaks within a *x* amplitude range, being *x* a user defined constant.
- Cepstrum domain – using a similar approach, but applying a cepstrum analysis instead of an FFT.

To improve the quality of the first individual, trying to achieve a better fitness, several pre-generations are used: the individual passes through a hill-climber method that applies some special mutations equally to all genes, changing the notes dynamics, duration or onset.
Since we only have one individual and an initial population is needed, the additional individuals are mutation versions of this initial individual (by cloning it and applying *n* forced mutations).

3.3.4 Individual Evaluation (Measuring similarity)

Most of the concepts that were addressed so far regarding our GA system (encoding, operators, population, etc.) will affect essentially the search capability of the system: its speed, the ability to overcome local maxima, etc. But, probably the most important aspect of the system is the individual evaluation. And it is here where the concept of similarity plays a decisive role.

The system must be able to “look at” each individual note sequence and measure how close it is, from the musical point of view, to the original audio. To achieve that, each individual’s note

sequence must be converted first into an audio stream, using a synthesizer. Then, both audio streams (the original and the synthesized one) are compared. These two tasks are described next.

Synthesis

The internal synthesizer is based on sampling. The system has 30-second recordings of each piano note, taken at a medium dynamics (MIDI velocity 64), from piano *sample* libraries, that included a Steinway, a Bosendorfer and a Bechstein pianos. The user can choose which piano should be used as the internal synthesizer.

The release time of each note is created by applying a fade-out curve to the audio *sample* and the dynamic behavior of the synthesizer is done by controlling a gain parameter with a dynamic range of 50 dB.

Audio similarity

After transforming the individual note sequence into an audio stream, a comparison is made with the original audio. The way audio is compared has a fundamental role on the system. The goal is to have a method that gives importance to aspects that are perceptual and musically relevant, and disregard other aspects that are not perceptual or musically important. For instance, doing a simple sample comparison (comparing the error between each sample of the original stream and the synthesized one) will have bad results – even two identical signals, but with opposed phase, will present a very bad result.

To test the enormous amount of possibilities regarding audio similarity, a testing application was developed, using Microsoft VisualStudio, Intel C++ Compiler and Intel IPP libraries. Although several G.A. frameworks already exist, it was important to create everything from scratch, not only to allow us a better understanding of G.A. choices, but mainly to have full development flexibility and highly optimized code, since performance is a critical factor. The use of Intel C++ Compiler was important for performance gain and the use of Intel IPP libraries allows us to run signal processing tasks (FFT, etc) in a very efficient way.

The evaluation unit was done using an architecture comprising six modules (see Figure 3.6):

- Pre-Synthesis processing – responsible for processing notes before synthesis is done.
- Synthesis – responsible for converting the note sequence into audio.
- Framing – responsible for converting the audio streams into time frames
- Domain – responsible for converting each time frame to a specific domain (e.g. FFT)
- Post-processing – responsible for applying some kind of pos-processing to the obtained values
- Measuring differences – responsible for applying a numeric method to compare the obtained values (e.g. sum of the quadratic differences)

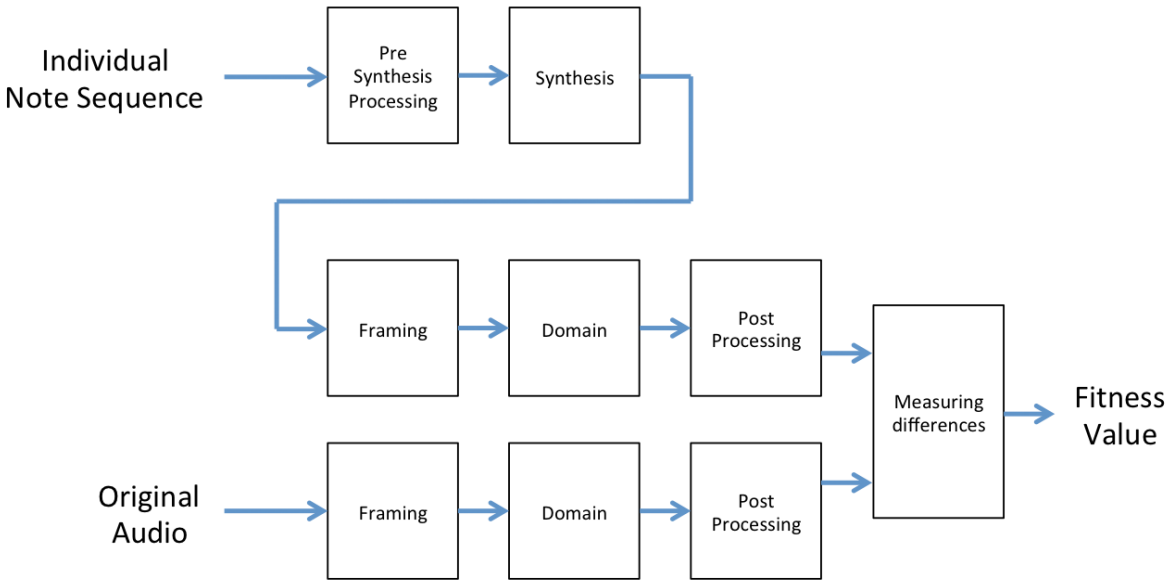

Figure 3.6 – Fitness evaluation unit

Pre-Synthesis processing

During pre-synthesis processing, several features were available to use:

- Discard notes with durations smaller than *n* ms;
- Discard notes with dynamic values *x* dB below the highest dynamic note on the note neighborhood;
- Dynamics Average, forcing all notes to share the same dynamics value (calculated by doing the average between all notes);
- Overlapping removal, preventing two notes with the same pitch to play at the same time;

Framing

Regarding framing, the user was able to choose the frame size, hop size and window type (Rectangular, Hamming or Hanning). The size of the time frame is, of course, a trade-off between temporal resolution and frequency resolution – smaller sizes improve temporal resolution, larger sizes improve frequency resolution.

Domain

Several domains were available:

- FFT with linear scale
- FFT with logarithmic scale
- Filter banks
- Cepstrum
- Hybrid (FFT and Cepstrum)
- Autocorrelation (ACF)
- Summary autocorrelation (SACF).

Post-Processing

During post-processing, several rules were available:

- Peaks only, considering only data from the local peaks
- Dynamic range, disregarding values *x* dB bellow the highest frame value
- Normalization, applying a gain factor (up to ±20 dB) to the synthesized frame data (as a way to decrease the impact of attack/decay/release differences between the original audio and the synthesized one). The gain could be calculated considering the average frame data, the frame data peak or iteratively trying several gains, with decreasing steps, until the best gain was detected to achieve the lowest error
- Data Blur. Bluring data by applying a low-pass filter to the frame data. The low-pass process was used by applying a window (rectangular, triangle, gauss), with a predefined *x* octave width, with symmetrical (same frequency width on both sides) or asymmetrical (same octave width on both sides) form.

Error Measurement

The final measurement was obtained by summing the frame errors over time, using frequency information from 27.5 Hz (lowest piano note) till the Nyquist frequency (i.e., half of the sampling frequency). The frame error could be obtained using one of the equations (3.1 – 3.9). The available equations range from traditional error measurements (equation 3.1 – 3.3), area interception (equation 3.6), correlation (equation 3.9) and other variations.

$$\sum |x_i - y_i| \quad (3.1) \qquad \sum |x_i^2 - y_i^2| \quad (3.2) \qquad \sum (x_i - y_i)^2 \quad (3.3)$$

$$\sum \frac{|x_i - y_i|}{\max(x_i, y_i)} \quad (3.4) \qquad \sum \frac{(x_i - y_i)^2}{\max(x_i, y_i)} \quad (3.5) \qquad \sum \frac{\max(x_i, y_i)}{\min(x_i, y_i)} \quad (3.6)$$

$$\sum \frac{\max(x_i, y_i)^2}{\min(x_i, y_i)^2} \quad (3.7) \qquad \sum log10\,(|x_i - y_i|) \quad (3.8) \qquad corr\,(x,y) \quad (3.9)$$

The measurement process also allows some features that could be applied, which change the way the error is obtained:

- Frequency normalization I. In most domains, different octaves have different number of bins (for instance, FFT doubles the number of used bins in each octave). As such, a normalization process can be easily obtained by dividing the data bin by the bin number, i.e., higher bins (that present less octave widths) will have less impact than lower bins.
- Frequency Normalization II. The same as previous but dividing the bin error, instead of dividing each data bin.
- Logarithmic intensity scale, obtained by applying a log10 formula to each data bin.

Instead of summing all frame errors, experiments were also done by calculating a 2D correlation in time/frequency between the original audio and the synthesized audio.

3.3.5 Initial tests and observations

Hundreds of tests were performed, looking for the best combination of parameters. Since synthesis was not a problem (our internal piano synthesizer (sampler) presented a good audio quality), we could focus on the analysis area, that would correspond to the system ability to search and measure similarity.

To evaluate the success of the tests, and since it was difficult to have non-human methods to evaluate the success of resynthesis itself[18], the choice was to evaluate the success of the system as a music transcription process, allowing to easily use numeric methods to measure its success. As such, the chosen metric was based on the onset-only Mirex metric [MIREX07], considering an onset time tolerance of ± 100 ms, using recall and precision values (comparing the original note sequence with the individual note sequence) to obtain a F-measure result. If the system was able to find a right note sequence, then the resynthesis would be successful.

Even from early tests[19], the system normally presents an evolution similar to the ones in Figure 3.7. The original notes would be detected (raising recall value), and precision values would rise for some time (deleting wrong notes) followed by a precision decrease (addition on new wrong notes).

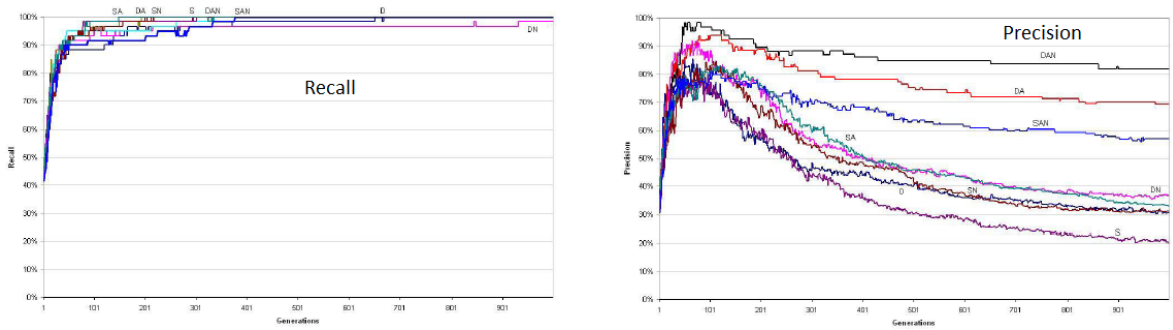

Figure 3.7 – Typical recall and precision evolution results over 1000 generations, with several tests with different configurations

Since our testing application allows the user to see, in real-time, the "piano-roll" of the original audio stream[20] as well as the piano-roll of the best individual of each generation, one of the issues observed was the fact that most errors were additional notes that were created in harmonic

[18] This is actually what we are trying to evaluate: comparing 2 audio streams in term of musical information similarity.

[19] Initial tests made during the development of the system, but not presented here.

[20] as long as there was a MIDI file with that information

locations of the original ones, usually with lower amplitudes and equal or smaller durations (Figure 3.8).

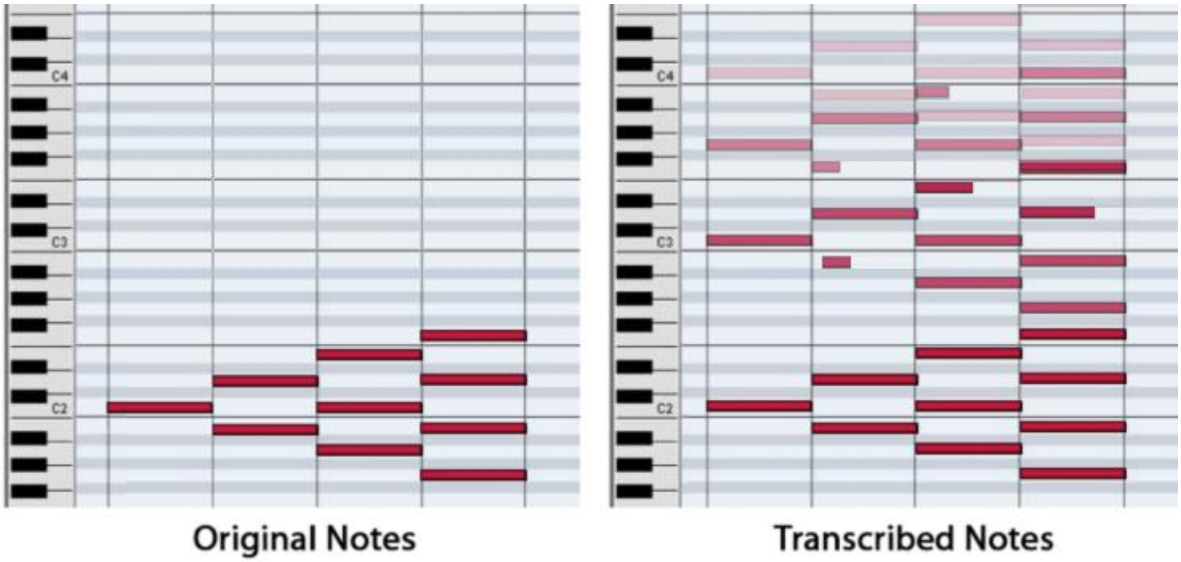

Figure 3.8 – Common errors occurring with additional low intensity notes on harmonic locations

Due to differences between the original timbre and the synthesized one, which would be responsible for differences on the amplitude of the harmonics, the system would create additional notes on those harmonic locations, usually with low amplitudes, as a way to decrease the differences between the original spectrum and the synthesized audio one.

We have designated this behavior as “harmonic overfitting”, because the system adds these notes as a way to decrease the error (fitting) on harmonic locations (harmonic), but using wrong notes (over-fitting) since it cannot decrease the error using only right notes.

This issue of harmonic overfitting will be probably one of the major issues that our system must handle, so several techniques were created to overcome it, and that was the reason behind many of the available pre-processing and post-processing features of our framework, always looking for new ways of preventing this harmonic overfitting behavior.

The following paragraphs present the main observations on the main modules of the system.

Pre-synthesis processing

The pre-synthesis processing revealed to be an important feature of the system, by cleaning up many garbage notes. Overlapping notes needed to be avoided, and notes with very small durations (< 50 ms) or small dynamics when compared to their neighborhood notes (<10 dB below) most probably exist due to harmonic overfitting. By applying these features, better results were obtained (a precision improvement around 15%).

Forcing all notes to have the same dynamics values, although this leads to better results when transcribing audio files created with a similar behavior, it is not a realistic feature, and presented worst results on more “real life” audio recordings.

Synthesis

Synthesis did not present a real impact on results. Of course, some internal pianos presented best results with one audio recording than another, but globally there was not any piano sound library that was able to systematically present better results than the others.

Framing

Tests were done with several settings, namely frame size from 512 to 8192 samples, hop sizes from 0% to 50%, and different windows (Hamming, Hann, rectangular, Gaussian, Blackman). The best results were obtained with a frame size of 4096 samples (92 ms at a sampling frequency of 44.1 kHz) and a hop size of 1024 samples (23 ms), showing that more resolution was needed in frequency than in time (although the offset tolerance of the music transcription metric could eventually benefit this situation). The Hann window presented the best results.

Domains

Regarding finding the best domain to compare audio, two interesting surprises were found. One of them was the fact that FFT with a linear scale on the magnitude axis presented best results than using a logarithm scale, especially considering that the human perception has a behavior much more similar to the logarithm scale than the linear one. Probably, the reason could be related to the fact that the linear scale increases the importance of higher amplitude frequency components when compared to a logarithm scale.

Some expectations existed regarding the SACF domain, since it is currently being used in many polyphonic music transcription and melody extraction systems ([Klapuri, 2008]), but, on our tests, FFT domain continued to present better results. As such, the domain with the best results was FFT with linear scale.

Post-processing

A post-processing feature that presented better results was the ability to limit the dynamic range. By considering only the frequency components on the top 40dB range, a better comparison is achieved, disregarding irrelevant frequency information.

The other available features, like “peaks only”, normalization and data blur did not present better results on our tests.

Error Measurement

The best results were obtained by using equation. 3.1 with frequency normalization II, i.e., like in equation 3.10:

$$Fitness = \sum_{t=0}^{t_{max}} \sum_{f=27.5\,Hz}^{fs/2} \frac{\left|\|X_{orig}(t,f)\| - \|X_{syn}(t,f)\|\right|}{f} \tag{3.10}$$

Calculating the 2D correlation between original audio and the synthesized audio, on a logarithmic FFT domain, presented interesting results, but still worst results than with the already mentioned measuring approach.

3.3.6 Harmonic Structure Evolution

On a second phase of the work, the architecture of the system went through several changes. The goal was to be able to evolve, not only the note sequence, but also the synthesizer characteristics to better model the original piano sound.

There are several aspects that can be different between two piano sounds. Thus, it is important to identify what sound characteristics can two piano sounds have, so that the synthesized piano sound could be changed into the original, if not completely, at least partially. Among them, we consider the following five domains:

- Time envelope – the behavior of the amplitude over time is an important factor. The way the attack occurs, how long it takes to decay and how it behaves on release (note release).
- Spectral envelope – what is the amplitude relation between harmonics.
- Inharmonicy/harmonic locations – harmonic are usually not perfectly aligned, i.e., harmonics are not always on multiple locations of the fundamental frequency.
- Dynamics – how time/spectral envelopes are affected by playing the same note at different dynamics.
- Static resonances/recording frequency response – how different is the behavior of the piano internal resonances and how different is the frequency response of the recording process (regarding microphone locations, frequency responses and directivity patterns).

All these five domains represent dimensions of the same sonic space (Figure 3.9) that cannot be handled separately. The time envelope depends on the spectral envelope, since each harmonic will have its own time envelope. But, once again, each harmonic time envelope will also depends on the dynamics, not only by a constant gain form, but because a “soft” note will have a different timbre than a “forte” note.

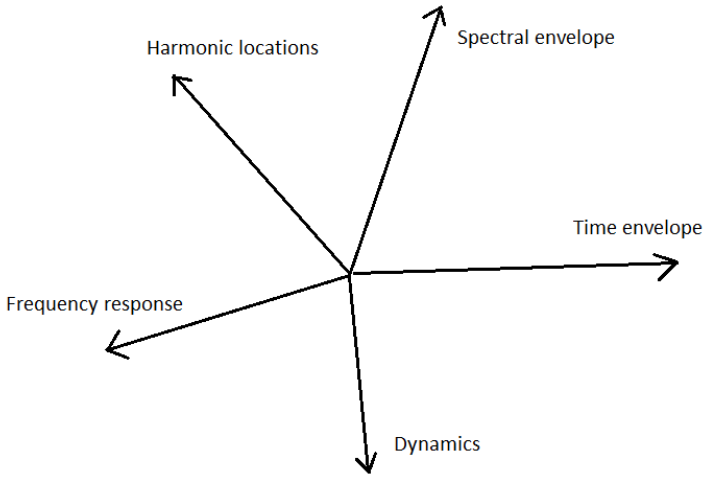

Figure 3.9 – Several dimensions of the sonic space

The initial approach was to begin by considering the spectral envelope and inharmonicity. As such, each individual will have, not only a note sequence as candidate solution, but also harmonic structure information (see Figure 3.10), consisting of 19 gains and 19 shifts[21], that would indicate to the synthesizer, which gain/shift should be applied to each of the first 19 harmonics above the fundamental frequency. The gain represents the gain (in dB) that should be applied to the harmonic FFT bin, and the shift represents the amount of bins that should be shifted.

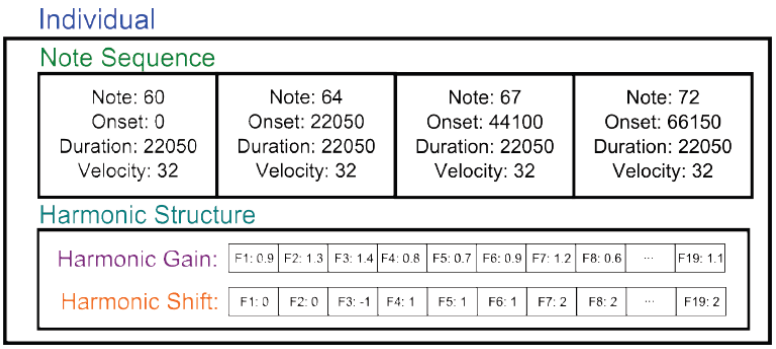

Figure 3.10 – Example of an Individual with a 4 note sequence and harmonic structure info

Since we are working on the FFT domain, it is simpler to apply these changes to the spectrum of each note/frame than using filters to accomplish such task. So, instead of applying filters to each note sample, add the resulting audio to create the final audio stream, and then create the FFT information based on the polyphonic audio, we use a different approach (Algorithm 3.1). For each time frame (n), the frames of the sound notes were converted to its FFT information, harmonic

[21]The main idea is to control the first 20 harmonic components of the sound. Since we consider that the fundamental frequency will have a fixed gain of 0dB and no shift, it is only important to evolve the remaining 19 harmonic components.

gains and shifts were applied to the FFT information, and then, the FFT information of each note was combined to create the final FFT data that was compared to the original FFT frame data.

```
Output_FFT[] = 0;

For each (sounding) note

      Note_FFT[] = CalcFFT(GetNoteFrame(note,n))

      Note_FFT[] = Apply(gains, shifts, Note_FFT);

      Output_FFT[] += Note_FFT[]
```

Algorithm 3.1 – Method for obtaining the FFT of frame n, by applying frequency gains/shifts on frequency domain

To combine the FFT data of the several notes, tests were made with and without phase information, i.e., to verify if just adding magnitudes was sufficient or if phase values should be taken into consideration, since they have an impact on the final magnitude (e.g. two signals with opposing phase result on zero amplitude). As expected, considering phase values presented a slightly better result (~3%).

With a new individual encoding scheme, it is required to change recombination and mutation operators to support it. As such, recombination, besides splitting the note sequence in two, will also split the gain vector and the shift vector (different random cut points for each vector). Regarding mutation, two new operations were added: a gain mutation, by applying a gain change within the range of ±0.5 dB; and a shift mutation, applying a change within ±3 to the shift value.

Tests were made with 30s audio from Mozart's Piano Sonata No. 17 in B flat (K570) using two different piano recording files as internal *samples* (Bosendorfer and Steinway). Table 3.1 and Figure 3.11 present the obtained results, considering a music transcription metric [MIREX 2007] with "onset only" and "onset/offset".

	Recall	*Precision*	*F-Measure*	*Overlap Ratio*
Bosendorfer "Onset only"	67.3%	72.6%	69.9%	63.1%
Steinway "Onset Only"	65.9%	74.5%	69.9%	63.1%
Average results "Onset only"	**66.6%**	**73.6%**	**69.9%**	**63.1%**
Bosendorfer "Onset/Offset"	36.9%	39.8%	38.2%	79.8%
Steinway "Onset/Offset"	36.8%	39.8%	38.3%	79.8%
Average results "Onset/Offset	**36.9%**	**39.8%**	**38.3%**	**79.8%**

Table 3.1 – Results (recall, precision,f-measure and overlap ratio) using Bosendorfer or Steinway piano *samples*, with "onset only" and "onset/offset" metrics.

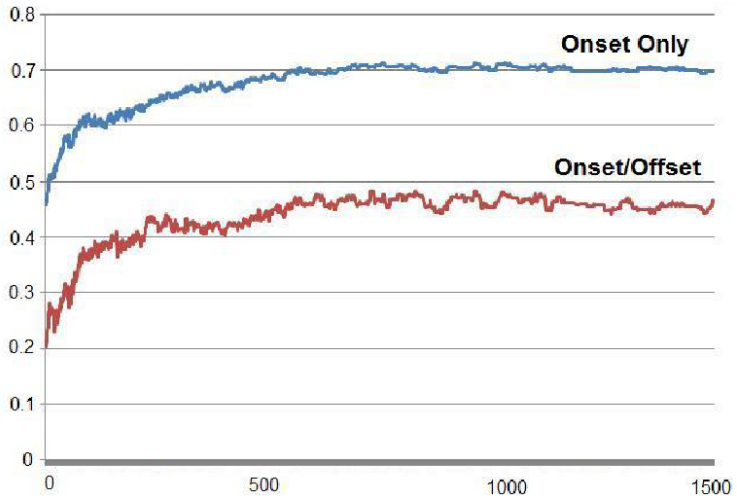

Figure 3.11 – Evolution of the f-measure results (average between pianos) over 1500 generations.

This harmonic structure evolution presented very similar results with different pianos, showing that different pianos were able to adjust to minimize timbre differences, making an approximation to the original piano sound.

The method and results presented in this section have been the object of a conference paper [Reis et al, 2007b].

3.3.7 Future directions

Additional work has been done with three new set of features: synthesizer layers evolution, onset detection and hostile evaluation. These features were implemented and preliminary tests presented very interesting results. However, since there was the need to refocus the PhD work back to the singing voice, these new features were not exhaustively tested, which did not allow us to take strong conclusions regarding them. Nevertheless, it may be important to mention them for future reference.

Synthesizer Layers Evolution

Instead of using only one pair of gain/shift vectors, several pairs were added to control different situations. For instance, four pairs of vectors may be used to represent four different situations in dynamics and pitch extremes (corners at Figure 3.12). Then, for each synthesized note, an interpolation is made between the four corners and the inside note position, to define the gain/shift vector values that should be used on that particular note.

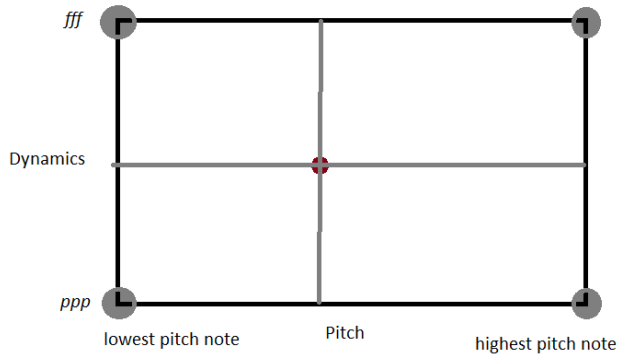

Figure 3.12 – Two layers (dynamics, pitch) for synthesis evolution.

The same concept can be increased with additional dimensions/layers. For instance, Figure 3.13 shows a three layer situation, adding a time dimension and four additional pairs of vectors.

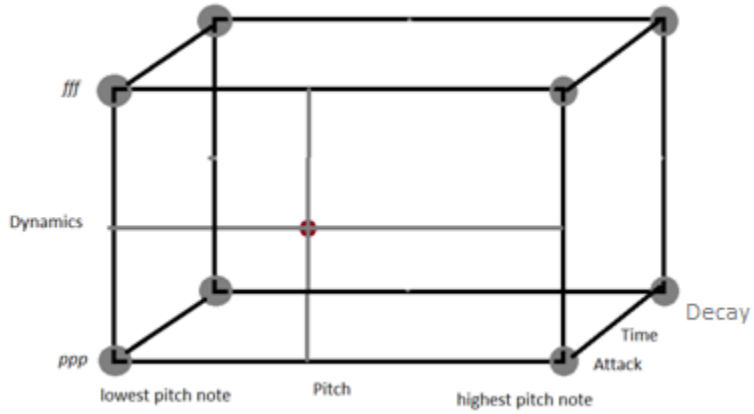

Figure 3.13 – Three layers (dynamics, pitch, time) for synthesis evolution.

Instead of using only two possible situations for each layer, more points can be added. A possible simplification can be accomplish, by handling each layer individually, and still interpolating values between layers, although losing some of its ability to handle the synthesizer sonic space.

Onset Detection

Since onset locations are one of the dimensions of our search-space, which we handle with a 1 ms resolution, a significant space reduction could be accomplished if we could know the location of onset events. Since we are working with piano recordings (that do not have slow-attack behavior), it is possible to try to use some sort of onset detection and force note events to start on that available onset positions. Even an onset detector with a very high rate of false positives, will present a search-space improvement.

Hostile Evaluation

The idea behind hostile evaluation is that the note sequence of each individual should not be evaluated using only one synthesizer, but using several ones. Since we want the resynthesis or music transcription system to focus on the music notes and disregard minor harmonic timbre differences between the original piano and the synthesized piano, this new feature would allow

each individual to be evaluated by using different pianos at the same time. During the evaluation of each individual, the system would create not one synthesized audio stream, but several audio streams (one for each available piano sound library). Each one of these synthesized streams would then be compared to the original audio stream and a fitness value would be obtained. The final fitness value would be obtained by combining all piano's fitness values. As already mentioned, this would allow the system to focus more on the harmonic structure of the pitches, and less on its spectral/timbre differences. It is important to refer that this is different from creating a new piano sound library where each note is a mix of all pianos *samples*. Doing this, we were simply creating a new piano sound, and harmonic overfitting would continue to easily exist.

3.4 Performance Enhancement on evolutionary resynthesis

One of the major drawbacks of search space approaches is their computational cost. Although Genetic Algorithms highly reduce the size of the used search space while finding for a solution, the need for a performance boost still exists.

3.4.1 Gene Fragment Competition

Genetic Algorithms are often very good at rapidly identifying good areas of the search space (exploration), but they perform poorly at refining near-optimum solutions (exploitation) [Hart, 2004]. One variation of Genetic Algorithms that presents a significant performance increase in some applications [Hart, 2004], by adding a local-search operation on the process, is called Memetic Algorithms. As can be seen in Figure 3.14, Memetic Algorithms use the same concept as GA, but consider an additional step that performs a "small" local search during the creation of each child. This local-search, that can consist, for instance, on a hill-climber, allows each child to perform better than using only recombination and mutation.

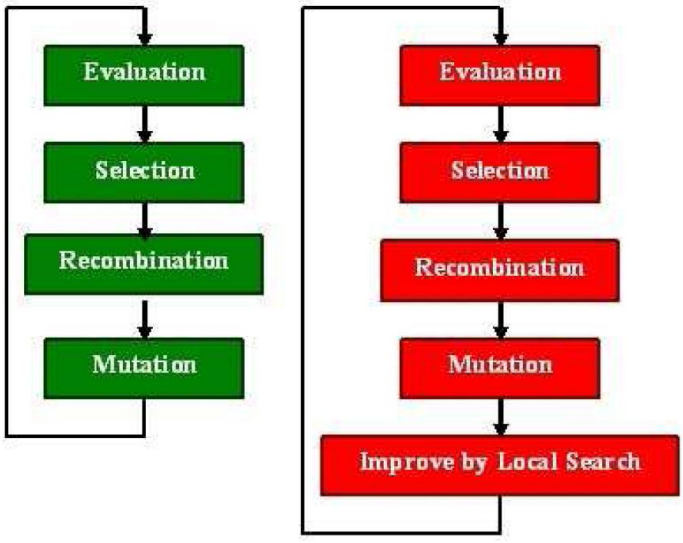

Figure 3.14 – Traditional Genetic Algorithms vs Memetic Algorithms

The main problem with the use of Memetic Algorithms in our application is that the main computational overhead happens during evaluation (synthesis, FFT calculation, etc.). Since local search methods require several evaluations, the amount of computational work would greatly increase. For instance, a Memetic Algorithm with a hill-climber that examines five neighbors would increase nearly five times the amount of computational cost, since each child would require not one evaluation (in G. A.) but six evaluations (original child plus five child neighbors in M. A.). It is important to discover a way to improve our local search ability without highly increase the computational cost.

In many applications, like our own, the fitness value of the individual is obtained by adding a sequence of numbers. In our case, the overall fitness is the sum of the frame's errors. This means that it is possible to fragment the genes of the individuals and even evaluate each fragment independently. On a traditional GA, an individual with an optimum gene fragment can be easily discarded if the remaining fragments are poor, because it only considers the overall fitness of the individual. Based on these ideas, we have created a new operator that was designated as Gene Fragment Competition, which replaces the traditional recombination operation.

Proposed Method

In the traditional GA approach, two parents are selected, their genes are recombined, creating two new individuals, that (after a mutation) are placed on the current population (Figure 3.15). We can consider that traditional recombination operators include two phases (Figure 3.15 a): fragmentation (parent genes are divided in two or more gene fragments) and merging (these gene fragments from parents are combined to create new individuals). The main concept of gene fragmentation competition is to include two additional steps inside recombination: fragment evaluation and fragment selection (Figure 3.15 b).

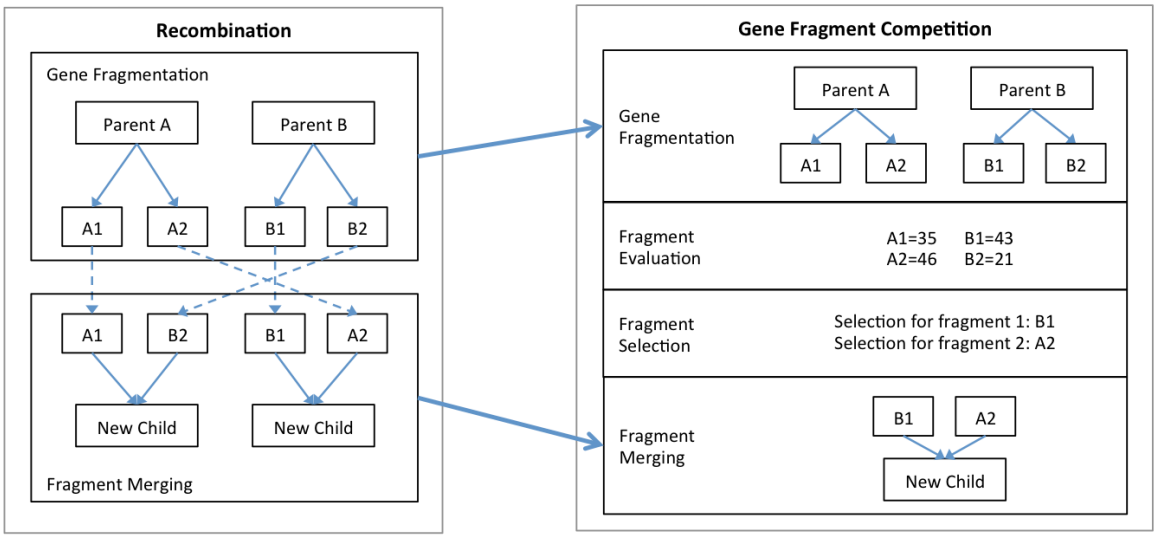

Figure 3.15 – Traditional Recombination (a) vs. Gene Fragment Competition (b).

During Gene Fragment Competition, the genes of the parents are split in *n* fragments. Then, each fragment is evaluated separately, and for each fragment location, a selection method is applied to choose the best gene fragment. The selected fragments are then merged to create the new individual. It is almost like creating a competition between gene fragments.

While traditional GA generates two new individuals during recombination, in this case, only one individual is created, having the best gene fragments of its parents. Also, with this approach, it is possible to have more than two parents. For instance, five parents may be combined to create a new individual (that will have the best gene fragments of the five parents).

To better explain the concept of Gene Fragment Competition, let us focus on a small example: the "OneMax" problem. This binary maximization problem is made of a sequence of *n* bits, and the fitness function is simply the number of genes set to "1". Of course, the perfect solution is a sequence of only 1's. Figure 3.16 shows an example of two parents.

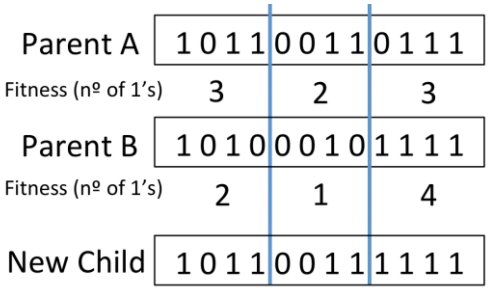

Figure 3.16 – Gene Fragment Competition on the OneMax problem, with 2 parents, considering 3 fragment

On the top of the Figure 3.16 the two parents A and B can be seen (in this case we are considering two parents, but it could be any number of parents). The first step is to fragment parent's genes (in this case we have considered fragments with four bits width). Next, each fragment is evaluated - the number bellow each fragment presents the fragment fitness value (its number of "ones"). Next, the fragment selection is made. In this case, we consider the best fragment but any traditional GA selection method [Goldberg, 1989] (roulette, tournament, etc.) could be used. Finally, the new individual is created using the best fragments from parents (the first two fragments from parent A, and the last fragment from parent B).

Instead of using a global search approach like traditional GA or different global/local search operators like Memetic Algorithms, Gene Fragment Competition performs a type of search somewhere in the middle (a quasi-global/quasi-local search).

Resynthesis application

Gene Fragmentation Competition is only possible on situations where the evaluation of a gene fragment is possible. For instance, on signal or image processing applications, individuals may be split in time fragments or spatial fragments. But this does not mean that a direct gene evaluation is required. In our case, genes represent note events and it is not possible to evaluate each note (gene) individually, especially in polyphonic music. However it is possible to evaluate time

fragments. So, our approach is to split the individual note sequence in small time fragments. If a fragmentation point occurs in the middle of a note, that note would be split in two (one for each side of the fragment point).

As mentioned before, in our resynthesis approach, most of the computational power is used on evaluation, due to the need of converting individual notes to an audio stream (synthesis), followed by audio analysis (ex: FFT differences). Although the Gene Fragment Competition approach increases the number of evaluations, and since we are evaluating both individuals (for parent selection) and individual fragments (for fragment competition), the overall required computational power does not increase significantly, due to the existence of a cache mechanism. During the individual evaluation, whose fitness value is the sum of frame errors, each frame error value is kept on a cache repository (which does not require a significant amount of memory). Later, during fragment evaluation, the fragment fitness is simply obtained by accessing the cache repository and summing the fragment frames errors, without needing a new synthesis or signal analysis process.

Tests

To analyze the impact of this method, tests were done comparing traditional GA approaches with Gene Fragment Competition. Table 3.2 and Table 3.3 show the used parameters.

Parameters	***Values***
Population/Offspring	100/100
Survivor Selection	Best 100 individuals (population size)
Crossover probability	75%
Mutation probability	1%
Note minimal duration	20 ms
Fitness	Equation 3.10

Table 3.2 – Common parameters

Traditional GA	
Selection	Deterministic Tournament (5 individuals)
Recombination	1-point time crossover (with eventual note split)
Gene Fragment Competition	
Individual Selection	Deterministic Tournament (5 individuals)
Fragment Selection	Non-deterministic Tournament (5 individuals)
Fragment size	5 seconds

Table 3.3 – Traditional GA parameters vs. Gene Fragment Competition parameters

Our concept of Gene Fragment Competition can be used with two different approaches regarding fragment sizes: the fragment sizes can be static (fixed) for all operations, which means that the

fragment points are always in the same locations through all generations; or, the fragment sizes can be dynamic, which means that fragmentation points are randomly selected each time a new child is created. We tested both situations.

The tests were performed trying to resynthesize/transcribe a 30s piano performance of Schubert's Impromptu No. 2 in E Minor. Each bank of tests were done with 1 000 generations and with four different runs (although results between runs were very similar, within a range < 3%). Table 3.4 and Figure 3.17 show the evolution of the obtained fitness over the 1 000 generations (choosing the fitness of the best individual in each generation/run and doing an average between the four runs). Once again, it is important to mention that in our system, since fitness is an error measurement, lower values mean better results.

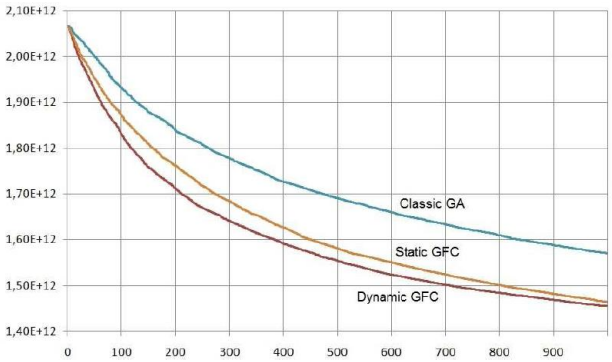

Figure 3.17 – Fitness value over 1.000 generations using traditional GA, using Gene Fragment Competition with static fragment size and dynamic fragment size.

Generation	*GA*	*Static GFC*	*Dynamic GFC*
250	1,808 e12	1,718 e12	**1,670 e12**
500	1,690 e12	1,580 e12	**1,554 e12**
750	1,621 e12	1,512 e12	**1,492 e12**
1.000	1,571 e12	1,464 e12	**1,455 e12**

Table 3.4 – Average fitness values over 250, 500, 750 and 1000 generations using traditional GA and Gene Fragment Competition with static and dynamic fragment size.

This first bank of tests shows that, apparently, the proposed method achieves a better performance than the traditional GA approach. Nevertheless, more situations and hypotheses needed to be tested before taking definitive conclusions.

A second bank of tests were made to check if changing the selection method of traditional GA from "Deterministic Tournament" to "Non-deterministic tournament" presents significantly different results (since "non-deterministic tournament" was used on fragment selection). The results were identical within a range <0.1%.

Since Gene Fragment Competition works with gene fragments smaller than the traditional GA approach, probably the mutation probability affects the results. As such, a third bank of tests was made to test other mutation probabilities. Figure 3.18 and Table 3.5 show the results.

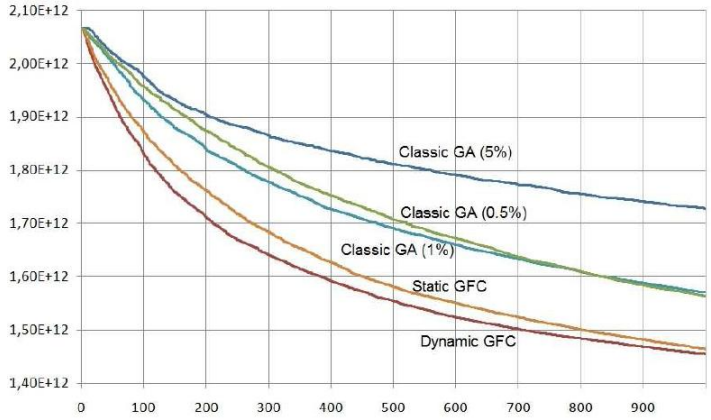

Figure 3.18 – Fitness value over 1.000 generations using traditional GA with mutation probabilities of 0.5%, 1% and 5% vs. using Gene Fragment Competition with static fragment size and dynamic fragment size.

Generation	GA (p_m= 1%)	GA (p_m= 0.5%)	GA (p_m= 5%)	Static GFC	Dyn. GFC
250	1,808 e12	1,839 e12	1,882 e12	1,718 e12	**1,670 e12**
500	1,690 e12	1,708 e12	1,811 e12	1,580 e12	**1,554 e12**
750	1,621 e12	1,624 e12	1,765 e12	1,512 e12	**1,492 e12**
1.000	1,571 e12	1,563 e12	1,728 e12	1,464 e12	**1,455 e12**

Table 3.5 – Average fitness values over 250, 500, 750 and 1000 generations using traditional GA with several mutation probabilities (0.5%, 1%, 5%) and Gene Fragment Competition with static and dynamic fragment size.

Once again, changing mutation probabilities did not present significant different results. A larger value (5%) results in worst values and a smaller value (0.5%) results on similar values.

Tests were also done with different audio recordings, but always achieving similar results. For instance, Figure 3.19 shows the fitness evolution when resynthesizing/transcribing 30s of Mozart's Piano Sonata No. 17 in B flat K570.

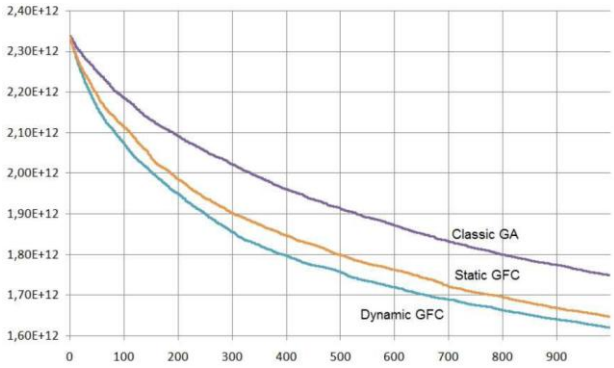

Figure 3.19 – Fitness value over 1 000 generations using traditional GA and using Gene Fragment Competition with static fragment size and dynamic fragment size – Mozart's Piano Sonata.

Tests have shown that Gene Fragment Competition could achieve similar results in 50% of the generations, which correspond to near 50% less computational power. The tests also shown that dynamic fragment sizes offers slight better results than static fragment sizes.

The method and results presented in this section have been the object of a conference paper [Reis et al, 2008a].

3.4.2 Fragment and frontier evolution

Although Gene Fragment Competition results on a computation improvement of about 50%, the need to improve the overall speed of the process continues to exist, especially to handle larger audio files. As such, additional work was done in this subject, by continuing to explore the advantages of fragmentation, but from a different perspective – fragmenting the problem instead of fragmenting the candidate solution.

Fragmentation can be a powerful tool for the optimization of search space problems like GA, since the simple act of fragmenting the problem into smaller parts, creating independent populations to solve each sub-problem, can boost the performance of the GA, since the decrease of the search space is quite dramatic. For instance, on a problem encoded as a 1024 bit sequence, the size of the search space is 2^{1024}, which is twice the size of a 1023 bit sequence (not of a 512 bit sequence). With the fragmentation of the 1024 bit sequence in two halves (with 512 bits), each fragment will have a search space reduction of 2^{512} times. Nevertheless, it may not be so simple, since problems could arrive, especially on the fragment frontier zones. To handle those issues, a method was created that tries to get the benefits of fragmentation, but still resolving the frontier issues.

The proposed method has two important requirements that are quite common in similar approaches: the problem that we want to solve must allow its decomposition on smaller problems, and an independent evaluation of each sub-problem must be possible.

The approach consists of several phases (see Figure 3.20). On the first phase, the problem (that we want to resolve) is fragmented on smaller problems. For instance, if we are dealing with an audio

file, the audio is split in small audio fragments; if it is an image, new smaller fragments of the image are created, etc.

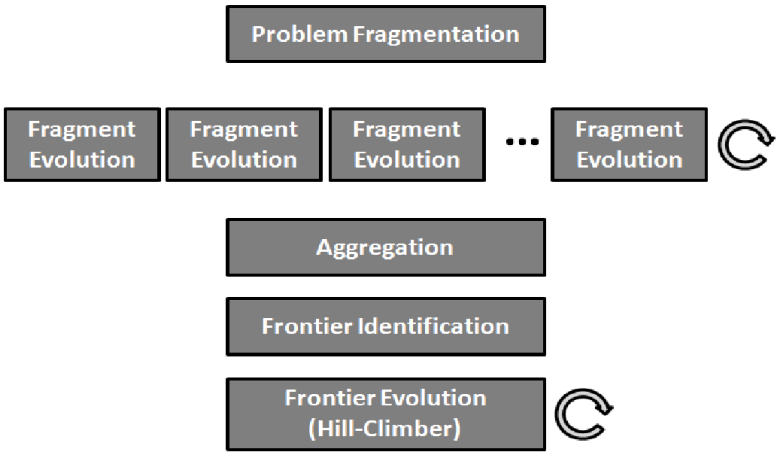

Figure 3.20 – Fragmentation and Frontier evolution

On the next phase, a population of GA individuals will be created for each fragment and will try to resolve that fragment problem "at its own", evolving during a defined number of generations using a traditional GA approach: Individuals are evaluated, parents are selected, new individuals are created and mutated, etc. During this phase, there is no interaction between different populations.

After the evolution of each fragment population (over a fixed number of generations), the phase of aggregation will occur: the best[22] individual of each fragment population is chosen (that we will designate as the fragment champion) to represent that fragment. All champions are merged to create our global candidate solution.

After this aggregation, the method will analyze all genes present on the candidate solution, and mark the genes that might have been affected by standing on a fragment frontier. This decision will be based on the problem, and could be as simple as defining a fragment frontier neighborhood distance, or a much more complex task where several factors are evaluated for each gene.

On the final phase, a frontier evolution using a hill-climber approach [Russel, 2003], performs a local search by applying special mutation operations only on the marked genes: during a defined number of iterations, a gene is randomly chosen between the genes that were marked on the previous phase (frontier genes), and a mutation is applied. If its overall fitness is better than that of the candidate solution, then it will become the new candidate solution. If the GA implementation supports several mutation operators, in this phase the system should only use mutation operators that might help to overcome problems created with the fragment cuts. This

[22] The individual with the best fitness value

last phase, responsible for the frontier evolution, has a crucial impact, since there are situations that simply cannot be resolved (or evolved) during a fragment evolution. For instance, applying a mutation operator that changes the note duration may help correct fragmentation issues, but using a pitch-related operator will probably not benefit the system, since fragmentation cuts will not change the pitch of existing notes.

As mentioned before, most of the required computational power is used on the evaluation of individuals. By creating N fragments, we are increasing N times the number of evaluations, but since each fragment will have 1/N of the original size, the required computational power to evaluate each fragment will decrease near N times when compared to the original audio. Since, most of the required computational power is used for evaluation, increasing the number of populations and its operations (selection, recombination, mutation, etc) will not represent a significant increase of CPU time. As such, the proposed method, although it may need some additional processing power, that will probably not be prohibitive or even significant.

Resynthesis application

To apply this proposed method to a resynthesis/transcribing application, some specificities must be taken into consideration.

During fragmentation (stage 1), the original audio is fragmented on smaller lengths. For instance, on three seconds blocks. Then, during fragment evolution, different GA populations are created for each audio block, which will evolve during many generations. In the end of this stage, the best individual of each population is chosen to represent its block and the candidate solution is created (aggregation). During frontier identification, the system will look at the candidate solution for note events that are located near the fragment frontiers. For instance, the system can identify notes that start or end on a 250ms range from the former fragmentation points (Figure 3.21).

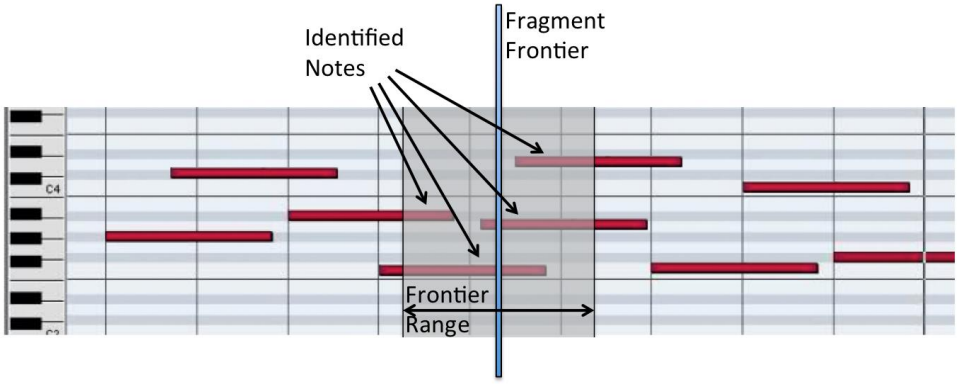

Figure 3.21 – Example of a Frontier identification process

During the last stage (frontier evolution), an iterative cycle occurs, where special mutation operations will be applied to one of the marked notes. Since the goal of this final stage is to resolve problems associated with the fragmentation (split notes, notes detected on one fragment but not on the other, etc.), the system must focus on mutation operations that can handle such

issues. For instance, creating a new note or changing a note pitch, although important during fragment evolution, will probably not resolve frontier issues.

Tests

Several tests were performed to help analyze the impact of the proposed method. The GA parameters used on the evolution of fragments were the same as in previous section (3.4.1), with some minor changes presented in Table 3.6. Since GA will work with smaller fragments, a higher mutation probability was chosen.

Population / offspring	*200/100*
Elite size	200
Mutation probability	10%

Table 3.6 – Used GA parameters.

In our test we used a 30s recording of Mozart's Piano Sonata No. 17 in B flat K570, and each test bank used four different runs (the presented results correspond to the average among the four runs). For frontier identification, the criterion was identifying notes that end at a 250 ms range of the frontier. During the last stage (frontier evolution), a 500 cycle was considered with two possible mutation operations: duration increase (up to 1 second) and note merging (merging the note with the next occurrence of the same pitch).

The following paragraphs present some of the test results.

GA Generations

The first bank of tests compares the traditional GA vs. the proposed method (with different fragmentation sizes), during 400 generations. The obtained fitness values are presented in Table 3.7 and Figure 3.22. The fitness values of the proposed method were obtained by summing the fitness value of the best individuals of the complete fragment population. Figure 3.22 also shows the fitness evolution during the frontier evolution.

	Fitness (25 gen.)	***Fitness (100 gen.)***	***Fitness (400 gen.)***	***Final Fitness***
Traditional GA	121.96	118.68	110.24	110.24
Proposed method (6s)	113.18	97.61	83.75	84.26
Proposed method (3s)	104.22	85.81	74.10	76.80
Proposed method (1.5s)	97.64	78.33	68.58	**73.24**
Proposed method (0.75s)	91.46	73.86	65.88	74.99

Table 3.7 – Fitness values after 25, 100 400 generations and the final values, considering traditional GA and the proposed method with fragment sizes of 6, 3, 1.5 and 0.75 seconds.

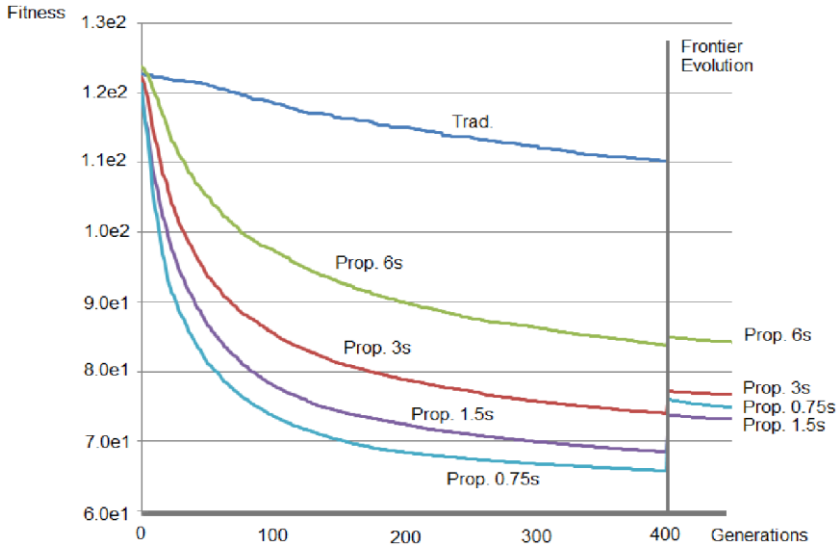

Figure 3.22 – Fitness evolution over 400 generations, using traditional GA and the proposed method with fragment sizes of 6, 3, 1.5 and 0.75 seconds.

As can be seen from Figure 3.22 and Table 3.7, for achieving some fitness value, the proposed method needs much less generations than the traditional approach.

Figure 3.22 shows a behavior that might look strange: the fitness value slightly increase in the transition between the fragment evolution and the frontier evolution. Although fragment fitness and global fitness are calculated in the same way (the sum of FFT differences over all frames), it is normal that value changes occur, since notes in one fragment can affect the next fragment with its release/decay behavior, situations that are not considered during the fragment evaluation.

It can also be seen that the value used for fragment size has an important role. By decreasing the fragment size, each fragment population can evolve much more quickly since the search space is substantially reduced. However, by decreasing too much the fragment size, the impact of fragmentation on the genes will increase. The smaller the fragment size, the larger the number of genes (notes) directly affected (split) or indirectly affected (genes near the fragmentation cut neighborhood). In this case, although the fragment size of 0.75 seconds presents the best evolution during the fragment evolution, as soon as the fitness is processed on a global base, the impact of the frontiers shows up. The global fitness is much more reliable that the sum of fragment fitness values, since it also considers the impact between fragments (e.g. a note from one fragment may continue to sound on the next fragment). And although apparently the smaller fragments (0.75 s) presented better fitness values during fragment evolution, since the moment global fitness is used, they change to second place.

Impact of mutation probability

The traditional approach must handle larger audio lengths, so its individuals have a larger number of genes. Considering the same mutation probability (as the proposed method) could be a wrong decision, since too much mutation may happen in each individual, which is bad from the performance point of view. To test if mutation probability values could be responsible for that

performance gap, several tests were made with the traditional approach, considering mutation probabilities of 10%, 3.3%, 1%, 0.33% and 0.1%. Results are presented in Table 3.8.

Mutation probability	*Final Fitness*
10 %	110.22
3.3 %	98.89
1 %	94.96
0.33 %	95.77
0.1 %	103.26

Table 3.8 – Test results with different mutation probabilities on the traditional approach.

As can be seen, although mutation probability could in fact affect the performance of the traditional approach, it still continues to have a performance evolution slower than the proposed method.

Frontier Evolution

Although the need of a frontier evolution could be validated from the theoretic point of view, since fragmentation might introduce issues that could not be resolved within the fragment evolution, it is important to analyze the importance of the frontier evolution from the practical point of view. Table 3.9 shows the fitness values before[23] and after the hill-climber process. Based on these two values, Table 3.9 also presents the number of generations (prior to hill-climber) needed to achieve the same amount of fitness improvement.

	Fitness before hill-climber	*Fitness after hill-climber*	*Generations needed to get the same fitness improvement*
Proposed method (6 s)	85.00	84.26	27
Proposed method (3 s)	77.26	76.80	32
Proposed method (1.5 s)	73.87	73.24	52
Proposed method (0.75 s)	76.13	74.99	125

Table 3.9 – Fitness values before and after the frontier evolution phase, after 400 GA generations.

As can be seen, the smaller the fragment size, the greater the importance of the hill-climber process to resolve frontier issues. If we consider that during the hill-climber phase (500 cycles), 500 individuals are created; that each GA generation creates 100 new individuals (by fragment); and that the major task (from the computational point of view) is individual evaluation, then the 500 hill-climber cycle will have a CPU requirement similar to 5 GA generations. As such, results show that the hill-climber phase contributes for a performance gain of the proposed method.

Table 3.10 shows the same parameters but considering only the 1,5 s fragment size, over 50 generations. In this case, with so little GA generations, the impact of the frontier evolution is very

[23] After aggregation stage.

modest, since there is probably much more work to be done on the global scale, than on the local scale of the frontiers.

	Fitness before hill-climber	*Fitness after hill-climber*	*Generations needed to get the same fitness improvement*
Proposed method (1.5 s)	88.57	87.05	5

Table 3.10 – Fitness values before and after the hill-climber phase, but after 50 GA generations.

CPU Time

Although the first tests compare the same number of generations between the traditional method and the proposed method, it is true that the fact that the proposed method has several genetic populations running, may add much more overhead. For instance, if the original problem is split in ten fragments, there will be ten times more recombination operations, ten times more mutation operations, etc. Therefore, it seems somehow unfair to take conclusions about its performance looking only to the same generation number. To analyze the real performance impact of the proposed method, an additional bank of tests was performed, testing different situations and considering different number of generations. Table 3.11 shows the obtained fitness values and the required CPU time (considering a Core 2 Duo processor - 2.0 GHz) for each one of these tests.

	Fitness value (lower is better)	*CPU time*
Test A – Traditional GA - 400 generations - Mutation prob: 1%	94.96	27 645 s (7:40:45)
Test B – Proposed Method (1.5s) - 30 generations - Mutation prob: 10%	93.37	1 518 s (0:25:18)
Test A – Traditional GA - 1000 generations - Mutation prob: 1%	83.44	72 268 s (20:04:28)
Test B – Proposed Method (1.5s) - 75 generations - Mutation prob: 10%	83.00	3 620 s (1:00:20)

Table 3.11 – Fitness values and required CPU time required for each test to complete

By comparing tests A and B, we can notice that even using the traditional approach with the best mutation probability (1%) during 400 generations, it takes only 5.5 % of the CPU time for the proposed method, with only 30 generations, to achieve a better fitness result. But, if we continue to increase the number of generations of the traditional approach (test C with 1000 generations), the performance gain continues to increase, with the proposed method needing only 5.0 % of the

CPU time (test D). The main reason is that the hill-climber phase requires the same amount of CPU time, independently of the number of previous generations. The greater the number of generations, the less impact will hill-climber has on the final CPU time, improving the performance gain of the proposed method.

The results show that the optimization is highly effective on the music transcription problem, requiring much less computational power. Some tests even suggest that performance benefits can achieve 20 times less CPU processing. Besides the performance boost, the method also takes special attention to internal damages created with its fragmentation. The advantages of the proposed method are maximized in the case of complex, high demanding, evaluation tasks, like this resynthesis/music transcription approach (using many FFT operations). If the evaluation is a simple process (from the processing point-of-view), although fragmentation could drastically reduce the search space, the overhead of creating additional population will have a stronger impact on the final performance.

The method and results presented in this section have been the object of a conference paper [Fonseca, 2008].

3.4.3 Conclusions and discussion

While GA only uses a small subset of the search-space, that subset continues to have an enormous size, requiring a lot of computational power. As such, every attempt to reduce the required CPU time is important.

Although the proposed approaches cannot be used in all kinds of problems, in the situations that allow fragmentation and that have complex evaluation tasks, both proposed methods could present an interesting optimization approach, especially the fragmentation and frontier evolution.

In the future, besides deeper and longer tests, many variations could be used with this last method: different local search methods in the last stage (frontier evolution); instead of using only the best individual of each fragment, several individuals may be chosen to form a population, which could evolve by the means of a memetic algorithm; overlapping frontier regions as a way to decrease the impact of fragment cuts.

3.5 Evaluation and Metrics for Music Transcription

As mentioned before, this proposed approach for resynthesis can be also used as a music transcription tool. Instead of using the created audio stream as the output of the system, by using the sequence of notes that originated that audio stream, and use it as output, we change from a resynthesis system to a music transcription system. And although much work is being done in music transcription research, the evaluation of music transcription techniques is less addressed by the research community and is even considered perhaps the weakest aspect of research on the

subject [Wang & Brown, 2006]. In most cases, the evaluation of music transcription results does not get much attention, besides brief paper paragraphs explaining how results were obtained.

Due to the existence of such lack of attention on this matter, some "side" research work has been done in this area, since it is fundamental to measure the results obtained by such systems, not only as a tool to improve them, but also as a way to compare the results of different approaches.

There are two main approaches for measuring the results of music transcription. The most common is note oriented - the first step is to identify what are the correctly transcribed notes, and the second step is to use that information for obtaining a final score. The other approach is time (frame) oriented, where original and transcribed "piano-rolls" are compared, frame-by-frame.

In note-oriented approaches ([Dixon, 2000], [MIREX 2007], [Ryynanen, 2005], [Reis, 2007]), notes are usually considered as correctly transcribed notes or as errors (there is no between). In order to be considered a correctly transcribed note, its parameters (pitch, onset and in some cases offset) must be within some tolerance values. Since most music pieces (especially western music) consider a musical resolution of a semitone, pitch tolerance of ± ½ semitone is widely accepted. Timing tolerances are more heterogeneous. Onset tolerances usually are between ± 25 ms and ± 150 ms.

In the less common time/frame oriented approaches, original and transcribed notes are represented as notes vs frames matrices, which are used as a mean to calculate their overlapping. The final value can be based on frame-level accuracy (use overlap values at frame level to calculate accuracy), or based on "speaker diarization error score" [Graham, 2007].

More recently, some interesting methods were proposed. For example, [Tavares, 2008] presents one idea to measure the distance between an original note and a transcribed one considering two Euclidean spaces. After calculating all the possible distances between each original note and each transcribed note, a one-to-one match process takes place. Finally, a performance evaluation array is created with diverse statistics information. Unfortunately, the method does not output a global measure for the music transcription.

In [Daniel, 2008], some human perceived tests were done and a new metric was proposed. It is still a note-oriented approach, but the transcription errors were divided in classes. Based on their human tests, each class of errors has an associated weight, depending on its perceptual impact on the music transcription. Although the amount of the error is not considered (still a binary approach: "correct note"/"error"), the impact of that type of error is considered, achieving, according to their tests, a metric closer to the human perception.

3.5.1 Impact of Transcription Errors

To be able to find good metrics for measuring music transcription results, it is important to identify what are the contributions to a good transcription and what are the common issues found on most current metrics.

Onset time tolerance

Everyone agrees that notes with onset values within a range of ±1 ms can be considered as successful transcriptions, but applying that small time tolerance to transcription systems will not provide us with real information about its transcription capabilities, since several "correctly" transcribed notes might not fall within such tight time tolerance. On the other extreme, applying a time tolerance of ±200 ms in transcription systems might tell us more information about their capabilities, but if a transcription achieves 100% success, we could not be sure that a perfect transcription was obtained. Some systems tend to use a time tolerance around 25-50 ms, probably because of the error margin of the ground truth and/or due to the proximity effect, also known as the Haas effect [Haas, 1972], since human note time resolution may be around 25-50 ms (sounds arriving to the ear with lower time intervals are perceptually merged).

In a music transcription, if a note is shifted by ±40 ms (regarding the original note), probably that shift is not even perceived. But it is important to analyze the impact of such time shifts, not only between original and transcribed notes, but also within transcribed notes. For instance, if two notes that should be played simultaneously are shifted respectively +40ms and -40ms, although they stay within a ±40 ms range of the original ones, they became apart by 80 ms, which can be easily perceived as a small transcript error.

Impact of the instrument time envelope

The behavior of the instrument time envelope (sound intensity behavior over time) can be an important factor in the music transcription process. For a better understanding of these issues, let us focus on two (extreme) types of musical instruments, which we will designate as "decay" instruments and "sustain" instruments (see Figure 3.23). Let's define "decay" instrument as a musical instrument with a very fast decay behavior (e.g. percussion, high notes from a harp, pizzicato violin, etc). And let's define "sustain" instruments as the ones characterized by having a constant energy/timbre behavior over the note duration (almost like "legato" woodwinds, strings, organ, but even with less timbre variations during attack).

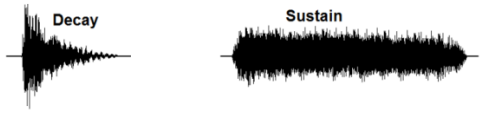

Figure 3.23 – A decay instrument and a sustain instrument

When transcribing music pieces of "decay" instruments, and depending on the instrument, the concept of offset might not even exist (e.g. glockenspiel), which means that to measure the transcription we should only focus on the note and on the onset. Even when the offset concept can be applied, and since most of the energy is concentrated on the initial moment of each note, from the perception point of view, the offset time value will not be as important as its onset time value. Also, in this type of instruments, the offset time is only important in a limited period of time, because after some point the sound energy will be below the threshold of hearing [Zwicker & Fastl, 1999], even if from the musician point of view the note is still playing. For instance, it is

almost impossible to detect or perceive if the higher C of a piano plays for three seconds or thirty seconds.

When transcribing "sustain" instruments, the importance of offset values highly increase (although it might continue to have slight less importance than onset values).

If offset values are not as important as onset values, it is important to find a way to take them into consideration but without giving them the same amount of importance as onset or pitch.

If we compare these differences between the transcription of decay and sustain instruments, with the differences of measuring transcription results using a note oriented approach or a time oriented approach, it may seem that decay instruments transcription may be better measured with note-oriented approaches, but sustain instruments transcription may be better measured with time/frame oriented approach. The major issue is that the majority of musical instruments are not fully "decay" or fully "sustain" based. Most instruments with a decay envelope end-up having a not so short decay time, and most "sustain" instruments, that may even have a constant energy behavior, end-up having an important timbre variation at the attack that may perceptually increase their impact compared to the rest of the note duration.

One-to-one mapping

Some works [Ryynanen & Klapuri, 2005], [MIREX 2007] refer the need of a one-to-one mapping when comparing transcription results, i.e., if a match is detected between an original note and a transcribed one, none of those can be used again on other note matches. The main idea is to avoid situations in which two different original notes could be matched to a single transcribed one (for instance, two quick notes being transcribed as one), or vice-versa, allowing a perfect score even in situations that have different number of notes. This rule is important, but it is usually incomplete, since it is not specified what should happen if an original note could be matched against two possible transcribed notes (or vice-versa).

For instance, on a 50 ms tolerance system, imagine an original note (W) at 960 ms and other original note (X) at 1020ms, that are transcribed as a 1000 ms note (Y) and a 1060 ms note (Z), like in Figure 3.24 (let's ignore offset values and consider same pitch for all notes). What should be the transcribed note pair of note X? The best match (Y, a 20 ms difference) or the first available (Z, since Y was first mapped against W – a 40 ms difference)?

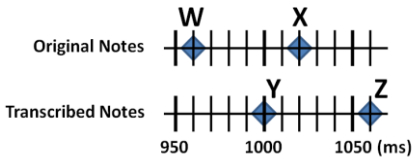

Figure 3.24 – Example of transcription (onsets only)

If we map X-Y, W and Z cannot be mapped because there is a 100 ms gap. If we map X-Z and W-Y, we get full mapping, although not mapping the best notes with X and Y. Of course, these types of

situations are probably rare on a global perspective, but they could be less rare within a special transcription system or within a special audio fragment.

Binary approach to transcription

Almost all metrics used on the subject employ a binary approach regarding the transcription of a note: the original note is perfectly transcribed or it is not transcribed at all. This binary approach has the advantage of being the simplest one, nevertheless, it might create some unfairness (regarding time tolerances, but also in others situations). Should all errors be treated equally? If some transcription errors have lower impact than others, should metrics take that into consideration?

For instance, splitting or merging notes are common situations: one note that is transcribed as a sequence of shorter notes (with same pitch), or a sequence of notes that are transcribed as only one note. Although these errors are easily perceived on decay sounds, it might be difficult to detect and perceive on sustain sounds (especially sounds with high release time values). Octave errors are also very frequent in music transcription systems and are usually treated as common errors. But although these octave errors are perceived by humans, their impact on perception is lower when compared to other pitch errors.

3.5.2 Proposed Method

Taking into consideration all of these issues regarding music transcription metrics, a new method is proposed, on which the final measure of a music transcription is obtained by the means of two different processes, based on the idea that “decay” and “sustain” sounds have different requirements regarding music transcription, and that most musical sounds have a combination of decay and sustain behavior.

In the first process, named “decay process”, the transcription is evaluated as if the musical sounds have all decay behavior, so only pitch and onsets are considered. On the sustain process, we consider that sounds are “sustain” based, and overlapping of both “piano-rolls” (original and transcribed) are obtained (considering pitches, onsets and offset).

Decay Process

In the decay process, only pitches and onsets are considered. Instead of a binary approach, made with “correctly” transcribed notes and “incorrect” transcribed notes, the system will produce a score for each pair of original/transcribed notes that will range from 0 (“completely incorrect” note transcription) to 100% (“completely correct” note transcription). The goal of this note score is to be able to obtain the maximum amount of information about the transcription performance of the system. Applying a threshold only, does not take into account several important aspects, so our approach considers that some transcription errors can be smaller than others.

If the pitch is correct (± ½ semitone), and if onsets are within a 25 ms interval, a full score is obtained (100%). If the onsets are separated by more than 200 ms, the score is 0. For situations between, a linear value is obtained as can be seen in equation 3.11 and in Figure 3.25. In the case

of octave errors (differences of ±1 octave between original and transcribed notes) a note score is also produced, but considering 30% of the values obtained with equation 3.11.

$$NoteScore = \begin{cases} 1 & , \Delta onsets \leq 25ms \\ \dfrac{200 - \Delta onsets}{200 - 25} & , 25ms < \Delta onsets \leq 200ms \\ 0 & , \Delta onsets > 200ms \end{cases} \quad (3.11)$$

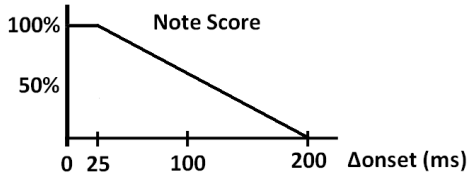

Figure 3.25 – Note Score

To prevent the issue of one-to-one mapping already mentioned, all original notes are mapped to the transcribed note that generates the higher NoteScore. On a separated process, all transcribed notes are mapped to the original note that generates the higher NoteScore. Mappings of n-to-1 and 1-to-n are allowed and there could be situations in which the mappings are not bidirectional (see Figure 3.26).

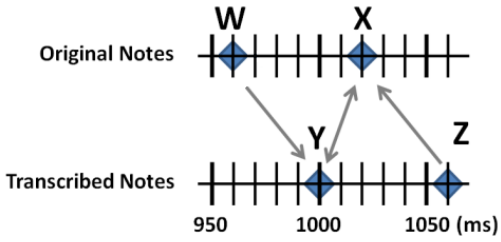

Figure 3.26 – The best mappings between original and transcribed notes

Each note (original or transcribed) will have an individual weight value (W_O or W_T) with a default value of 1. However, if the note (original/transcribed) ends-up never being picked as the best match for any other note (transcribed/original), their weight is reduced to 0.5. For instance, in Figure 3.26, notes W and Z will have weights of 0.5 because no other notes have chosen them as best match.

Although the application of the weights (W_O or W_T) might increase the complexity of the algorithm, they are responsible for three important features:

- Note sequences are commutative (exchanging original and transcribed note sequences will produce the same final score).
- Splitting/merging note errors are not so penalized.

- Since the proposed process does not have a binary approach to correct/incorrect transcribed notes, forcing one-to-one maps would require the existence of an important optimization process to be able the choose the right one-to-one mappings as a way to get the best overall score.

A variation of recall and precision is calculated (equation 3.12 and equation 3.13), where N_O and N_T represent the number of original notes and the number of transcribed notes. The DecayRecall just considers the best NoteScore of each original note, and DecayPrecision just considers the best NoteScore of each transcribed note.

$$DecayRecall = \frac{\sum(NoteScore(O_i) \times W_{oi})}{N_o} \tag{3.12}$$

$$DecayPrecision = \frac{\sum(NoteScore(T_j) \times W_{Tj})}{N_T} \tag{3.13}$$

Although it might look that the obvious choice for a final decay score would end-up on using the f-measure formula, the choice was made to use accuracy as final decay score, since it is a more linear behavior and most of f-measure advantages [Manning 2008] do not apply. By disregarding true negatives (TN), accuracy can be calculated as seen in equation 3.14.

$$Accuracy = \frac{1}{\frac{1}{recall} + \frac{1}{precision} - 1} \tag{3.14}$$

The final decay score is calculated as equation 3.15.

$$DecayScore = \frac{1}{\frac{1}{DecayRecall} + \frac{1}{DecayPrecision} - 1} \tag{3.15}$$

Sustain Process

In the sustain process, the main idea is to measure the overlapping between the original notes and transcribed ones, almost like analyzing the overlapping of their “piano-rolls”, considering the pitch, onset and offset of each note. For each note, up to a 25 ms tolerance on the onset and on the offset are considered. The idea is to be able to get a 100% score even if notes timings are slightly different (within a 25ms range). To achieve that, a value between 0 and 25 ms can be subtracted to the onset value or added to the offset value, in original or transcribed notes, as a way to increase their interception, i.e., the duration of the original note or the duration of the transcribed note can be increased by up to 25 ms on its onset and up to 25 ms on its offset (Figure 3.27).

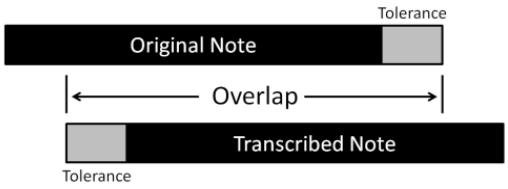

Figure 3.27 – Interception between an original note and a transcribed one

After this transformation on the timings of original notes and transcribed notes, a variation of recall and precision values are obtained using equation 3.16 and equation 3.17.

$$SustainRecall = \frac{\sum Overlaps}{\sum OriginalNoteDurations} \tag{3.16}$$

$$SustainPrecision = \frac{\sum Overlaps}{\sum TranscribedNoteDurations} \tag{3.17}$$

It is important to refer that the values used on equations 3.16 and 3.17 are the ones calculated after timing transformation (with added tolerances), and not based on the real original and transcribed notes.

The final sustain score, once again based on the concept of accuracy, can be calculated using its recall and precision values according to equation 3.18.

$$SustainScore = \frac{1}{\frac{1}{SustainRecall} + \frac{1}{SustainPrecision} - 1} \tag{3.18}$$

Octave errors are also considered, but once again with only 30% of the time value obtained with the interception. An original note fragment and a transcribed note fragment can be considered as an interception if they are one octave apart and if they do not intercept with other notes/fragments. If this is the case, 30% of their time value is considered as the interception, and the remaining 70% of the time value are considered as not intercepted value. As can be easily seen, on the sustain process, longer notes have a larger impact on the final score than smaller notes. The sustain process also allows to benefit some errors that otherwise would be considered as ordinary transcription errors like errors in missed onset determination on small-attack sounds, note merging/splitting, etc.

Final Score

The final result is the average of the values obtained on the decay process and on the sustain process (equation 3.19), and will range between 0 and 100 %.

$$FinalScore = \frac{DecayScore + SustainScore}{2} \tag{3.19}$$

From the analysis point of view, the process makes available seven different parameters that present important information about the transcription capabilities of the system: DecayRecall, DecayPrecision, DecayScore, SustainRecall, SustainPrecision, SustainScore and FinalScore.

Implementation

An implementation of the proposed method, for the Matlab Framework or C/C++, is freely available at http://www.estg.ipleiria.pt/~nfonseca.

3.5.3 Perception Tests

Although it makes sense to use a metric that considers different types of errors and different amounts of errors, it is important to compare the proposed method with other methods and also with human perception tests.

To validate our approach, we used the test materials created by [Daniel et al, 2008], consisting on three different music fragments (from Bach, Debussy and Mozart piano pieces) that were transcribed with four music transcription systems (15 files, considering the reference files also). A group of 31 listeners were asked to rate the discomfort of each transcription on a scale between 0 and 1.

To evaluate which metric correlates best with the human results, three methods were used (as presented in [Emiya, 2008]):

- Pearson linear correlation coefficient (prediction accuracy), which measures the linear correlation between two sets of values.
- Spearman rank order coefficient (prediction monotonicity), which disregards the actual values, considering only their order.
- Outlier ratio (prediction consistency), which validates that there is not any value too far apart from the correct value.

Table 3.12 shows the obtained results of [Daniel et al, 2008] (marked with *) and the ones of the proposed method (individual decay and sustain scores are also presented).

Metrics	*Prediction accuracy*	*Prediction monotonicity*	*Prediction consistency*
F-measure*	83.4 %	83.5 %	**0 %**
Perceptive F-measure*	84.1 %	84.9 %	**0 %**
PTD*	60.3 %	61.6 %	**0 %**
Perceptive PTD*	64.8 %	**89.6 %**	6.7 %
Decay Score	89,2 %	84,3 %	**0 %**
Sustain Score	85.0 %	80.0 %	**0 %**
DecaySustain Score	**90.5 %**	82.8 %	**0 %**

Table 3.12 – Prediction accuracy, monotonicity and consistency for several metrics when compared to the human listening test results

Discussion

The proposed method presents the best prediction accuracy with the human results presenting a significant improvement over the other methods. Like most other metrics, the method also does

not present outliers (prediction consistency = 0%). The only parameter with lower results is the Prediction monotonicity. Since this parameter is particularly used on non-linear situations, and since the linear correlation value is above 90%, it seems to us that its importance can be slightly reduced.

The results also show that although the human tests were done with piano transcriptions, that might be considered a “more” decay instrument, the combined use of decay and sustain methods gives the best score.

Although the test set could be used as a way to partially validate the proposed method, it does not have the necessary size or diversity to truly validate the proposed method, much less to tune its internal parameters. For instance, additional tests were made using the same test set and changing internal values of the proposed method, allowing better results. Nevertheless, those values are not presented since it is our opinion that those better results simply represent a best fitting with the small test set, and that would probably not improve the overall capabilities on other test sets.

3.5.4 Conclusions And Future Work

A different metrics approach to music transcription was proposed. Its main features are the following:

- music transcription errors are different from each other, either on type and on amount;
- offset has a smaller impact than onset (since it is considered only on “sustain” processes);
- timing variations have a more gradual impact (instead of a discrete impact);
- the process is commutative (exchanging original and transcribed note sequences generates the same final score);
- octave errors have a slight lower impact than other pitch errors;
- merging or splitting notes has a slight lower impact than in traditional metrics;
- slow-attack sounds are better handled than traditional metrics.

Nevertheless, the proposed method still presents some drawbacks: it is more complex than standard metrics (although a free implementation is available) and it still does not analyze the impact of note dynamics, which can be important in some music transcription scenarios.

In the future, more attention should be given to music transcription metrics by the research community. Besides pitch, onset and offset, future metrics should also be able to analyze other music features that future transcription systems may extract, like dynamics, timbre identification, pitch variations (vibrato, portamento, etc), etc. Regarding the proposed metric, larger perception tests are needed as a mean to better understand the impact of each type of transcription error, the impact on different types of instruments, and also as a way to allow tuning of metrics internal parameters.

Although these metrics were created for music transcription, this approach can be also used in other areas of MIR, like the measure of similarity between note sequences (e.g. measuring melody extraction, resynthesis).

The method and results presented in this section have been the object of a conference paper [Fonseca, 2009].

3.6 Final Conclusions

Although the evolutionary approach presents some interesting concepts to resynthesis (or to music transcription), it still possesses some characteristics that could act as obstacles to its adoption in real-life applications.

One of the main drawbacks is the required computational power. Even using a very small subset of the entire search-space (by employing Genetic Algorithms) and applying some optimization features (like fragmentation and frontier evolution), the amount of computational power required is still very high. Not only this prevents any type of real-time application, but also demands too much CPU power on off-line situations.

One characteristic of Genetic Algorithms, also present in many other computational intelligence methods, is its non-deterministic behavior. In this case, different runs of the same problem may present different solutions, due to the random evolution of the individuals. Although not being a problem at its own (since it presents a possible solution to the problem, probably not the optimal solution, but a near one), this type of behavior may not be seen as desirable on such systems.

Probably, these evolutionary approaches are not the best way to resolve problems from the theoretical point-of-view (non-deterministic behavior, not-optimal solution assurance, etc). The ultimate goal is to find methods that allow us to *always* get the *optimal* solution. Nevertheless, from the practical point-of-view, there will be always problems that due to its complexity or due to the lack of better solutions created by the research community, will push evolutionary methods as very interesting approaches, probably achieving better results, during some time period, than other methods. For instance, in applications like polyphonic music transcription, that probably will continue to be an unresolved problem for years or decades to come, evolutionary approaches may eventually present very good results until a satisfactory solution to the problem is found.

Since the main goal of this PhD work is the resynthesis of the singing voice, it is important to re-focus the scope of the research that eventually got a little deviated from its main focus. And although this line of research may not be continued by us, it is still important to mention what future work could be done in this area, that can be of interest to other researchers working on the subject.

The main problem with this type of approach regarding polyphonic music is the harmonic overfitting. Since there are always differences between the original musical instruments and the

internal synthesized ones, it is important to reduce such differences. And to achieve that, two concerns must be simultaneously taken into consideration: the synthesizer must be able to better imitate the original sound, and the way audio streams are compared must also try to minimize timbre/harmonic differences, focusing more on their perceptual similarity. Regarding the first goal (similarity between original sound and synthesized one), many techniques can be used, either evolving the synthesizer characteristics (changing its time, dynamics and spectral dimensions with greater resolution) or using different types of synthesis methods (for instance, using an additive approach). Regarding the second goal (audio comparison, by focusing on the sounds pitch-related harmonic structure and disregarding their timbre-related harmonic differences), a possible way could involve testing hybrid method that combine different domains (for instance, SACF as in [Klapuri, 2008] and FFT). An interesting approach could be the use of Genetic Programming, which would try to discover a better method for audio comparison, based on perceptual examples. In such system, programs evolve to better match the desired behavior. And eventually a completely new method or audio domain could be created, achieving a new paradigm in audio analysis.

Regarding the CPU power requirements of evolutionary approaches and its optimizations for signal processing, much work can be done, especially using fragmentation as a tool (as already mentioned in section 3.4).

3.7 Summary

This chapter presented the work done regarding an evolutionary approach to resynthesis.

Initially, some resynthesis concepts here introduced, including possible architectures.

Later, a Genetic Algorithms approach for resynthesis was proposed, with a focus on the resynthesis of piano recordings.

Due to the demanding computation requirements of the GA approach, two performance enhancement methods were proposed. A new metric for the evaluation of music transcription was also presented.

Finally, some conclusions were taken, to refocus the research on the main problem of the singing voice.

4. Singing Voice Resynthesis

4.1 Introduction

As mentioned in the first chapter, the main goal of this PhD work is to create a resynthesis approach for the singing voice. After some work on the resynthesis of piano recordings (chapter 3), the focus returns to the singing voice.

Within the singing voice resynthesis, there are essentially three domains that need to be replicated from the original to the resynthesized audio: Dynamics[24], with its time varying effects and performance (e.g. *crescendo*); Pitch, with its melodic line and fine pitch information regarding pitch related effects and performance (e.g. *vibrato* or *portamento*); and Phonetics, replicating the same phoneme sequence and allowing its intelligibility.

In this context, this chapter will present the work being done trying to resynthesize singing voice recordings. Section 4.2 explores a synthesis approach for singing voice; section 4.3 explores the extraction of pitch; section 4.4 explores the replication of dynamic information; section 4.5 focuses on the replication of phonetic information; section 4.6 addresses the combination of all modules to create a resynthesis system; section 4.7 presents some approaches to improve the audio quality of the system; and section 4.8 presents an evaluation of the obtained results.

4.2 Synthesis

A synthesizer is responsible for converting symbolic information into an audio stream, and it has a fundamental role on the resynthesis process. Most synthesizers receive that symbolic information using an event-based approach, where a sequence of discrete events (in most cases, note events) is fed to the synthesizer. Common events include starting notes (e.g. MIDI Note On), stopping notes (e.g. MIDI Note Off), changing instruments (e.g. MIDI Program Change) or changing parameter values of the synthesizer (e.g. MIDI Control Change). Several factors contribute to the use of an event-based approach: music sheets represent music events; most virtual instruments

[24] Musical concept of dynamics, regarding sound intensity.

handle MIDI, which is an event-based musical interface; most *Digital Audio Workstations* (DAW) base their user interaction on musical events.

Although this event-based approach presents many advantages, probably being the best approach for the interaction between user (musician) and synthesizer, it may presents some issues, especially when it comes to resynthesis. The main problem is that it may be difficult to identify where does a note event end and a new event starts. For instance, a *portamento* effect (a gradual shift between two different musical notes) is difficult to deliver and control on a note-based system. On a resynthesis system, where the synthesizer is controlled by an audio stream, it is more difficult to clearly identify the note events, being much easier and reliable to simply identify the evolution of different audio features. Instead of identifying note events (and the information of each event), a resynthesis system can extract the different domain information (pitch, sound intensity, timbre information) and transmit that information to the synthesizer. On this scenario, the synthesizer does not use an event-based approach, but a domain-based approach, receiving independent information for each domain (Figure 4.1).

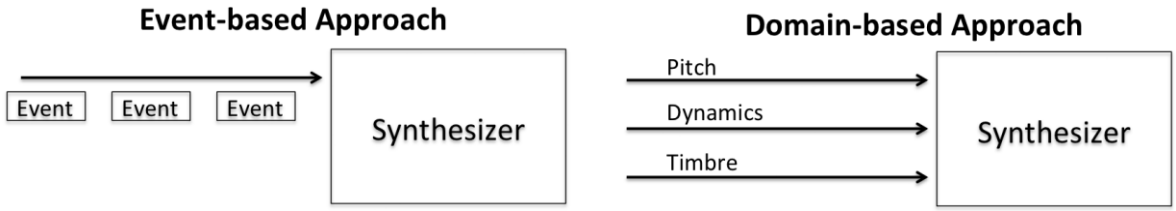

Figure 4.1 – Synthesis: Event-based approach vs. Domain-based approach

As mentioned in chapter 1, one of the goals of this PhD work was to continue the work that had been done by the author in the area of singing voice synthesis using concatenative approaches. Due to its virtual instrument and sampler-based nature, the original singing voice synthesizer [Fonseca, 2003a] used a note-based approach, and presented some limitations:

- *Portamento* effects (sliding from one note to another note) were almost impossible to reproduce, unless a pitch-bending mechanism was used, which would change both pitch and formants of the *sample*, creating a "Mickey Mouse effect" ([Zolzer, 2002]).
- Vibrato (low frequency variation of the pitch) was only controllable by a unique control, which would cross-fade between different *samples* with different vibrato amounts. Since the original singing synthesizer worked with choir voices, the higher number of voices would help to mask phase artifacts during transitions or when different *samples* were playing at the same time. With solo voices such artifacts would become audible.

With these issues, and considering the challenges of event-based synthesis on a resynthesis system, the decision was made to change the singing voice synthesizer to a domain-based approach. The synthesizer would still be based on sampling (but without modeling its behavior, like [Bonada, 2007]), and would try to minimize the amount of signal processing applied to the *sample*, keeping its full naturalness. As such, instead of an event-based input, the synthesizer

would use domain-based inputs, continuously receiving parametric information of the desired behavior in three different domains: pitch, sound intensity, timbre/phonetics.

To improve the system's sound quality, the decision was made to consider an offline resynthesis system (with an offline synthesis module) that would not work in real-time, allowing it to have the desired computational complexity (no time constraints for processing), no latency restrictions, and most important, allowing the system to analyze the full input audio stream to extract the best information and make the best decisions. Of course, such decision prevents its use in real-time applications like live concerts. Nevertheless, future work could later be done to adapt the system to such requirements.

4.2.1 Main concepts

The singing synthesizer will have three independent domain inputs that represent the desired output behavior in terms of pitch, dynamics (sound intensity) and timbre (phonetics).

- Pitch should indicate the exact pitch value of the output. Not only the actual music note, but also fine tune information. This input is the one to be used to create pitch related effects like *portamento* or *vibrato*.
- Dynamics should indicate the desired sound intensity of the output. Timbre differences that may occur due to the use of different dynamics (*fff* vs. *ppp*) should be covered by the timbre domain input.
- Timbre should indicate the desired characteristics of the timbre of the output. In the case of a singing voice, this domain also represents the phonetic information. Unlike the previous inputs that require a single value to represent them, this input may require a vector of values to represent the desired information.

The synthesizer will have an internal sound library with monophonic singing material. These recordings cover several different pitch values and different phonemes. As expected, the larger the amount of pitch values and the amount of phonetic situations, better results can be obtained.

Based on the domain inputs, frames from the internal sound library are selected and used as output frames. Some additional processing will probably be required, not only to comply with the desired dynamics and pitch values, but also to prevent sound artifacts. Nevertheless, this signal processing should be as minimal as possible, as a way to keep the naturalness of the used *samples*.

It is important to mention that the dynamics input will not affect the frame selection, and that the timbre input is used only for *sample* selection, not applying any additional signal processing to the chosen sample.

4.2.2 Dynamics (Sound Intensity)

Regarding dynamics (sound intensity), the synthesizer will simply use an ordinary gain module. Based on the difference between the selected frame energy and the desired energy, a gain module would apply a gain factor to the frame as a way to obtain such desired output result.

4.2.3 Sample Concatenation

The original choir-based singing synthesizers [Fonseca, 2003a][EASTWEST, 2001][EASTWEST, 2005] did not need to pay much attention to the way the concatenation was done. Applying a simple attack and release envelope would be enough to concatenate different *samples*. But to work with solo voice, concatenation needs to take some aspects into consideration, especially to prevent phase artifacts and ghost effects.

Overlapping plays an important role on the concatenation of audio. A too short or even inexistent overlap will probably create abrupt transitions, which will result on audible clicks and similar artifacts. A too long overlap will increase the chance of ghost effects[25], will increase the chance of phase misalignments between fragments (phase artifacts), besides decreasing the time resolution of events.

The concatenation of choir fragments will favor longer overlapping times: phase is not an issue, since tens of singers naturally create phase misalignments between each other; ghost effects are not significant, since the listener cannot identify the exact number of choir singers; and time resolution of events is not crucial, since tens of singers do not sing phonemes exactly at the same time. But with a solo voice, longer overlapping durations create issues: phase misalignment can be easily detected, ghost effects can also be detectable (listening two voices where should be only one), and loosing the voice ability to quickly change its characteristics especially during consonants.

In this context, a different approach was implemented for solo voice synthesis, using a short overlap between audio fragments (about 11ms, corresponding to 512 samples at a sample rate of 44.1 kHz). However, even with a relatively small overlap, phase artifacts may occur. To prevent them, a phase alignment process was implemented as illustrated in Figure 4.2. During concatenation, several offset values are tested for the chosen frame. The offset value that creates the highest correlation within the overlapping area is chosen and applied.

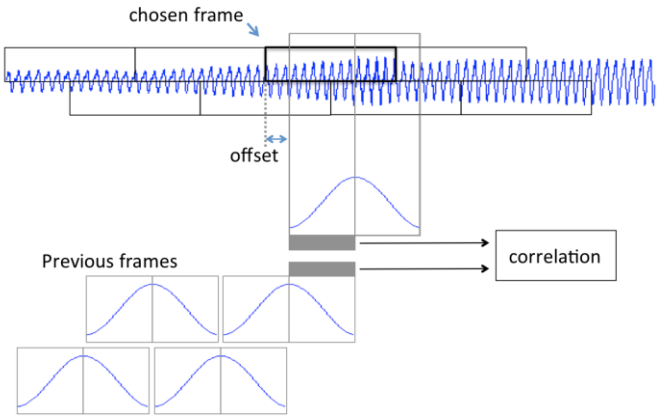

Figure 4.2 – Phase alignment, by using the offset that produces the strongest correlation

[25] doubling the number of perceived voices

Considering that most phase misalignment issues can be corrected within one period of the waveform, the maximum amount of the offset is equal to the number of samples required to obtain one period of the pitch. For instance, a frame with a pitch value of 440 Hz would consider a maximum offset of 101 samples (assuming a sample rate of 44.1 kHz).

This phase alignment process is not required on transitions between adjacent frames of the same *sample*, because in this situation frames are already naturally aligned.

Although correlation can become a heavy processing task, especially considering the number of concatenations that need to be done, the decision was made to currently disregard the CPU requirements of the chosen methods. Future work can be done to reduce the required processing power without affecting the quality of audio.

4.2.4 Controlling Pitch

Current high-end sampling libraries always present different recordings (*samples*) for each note. As such, apparently no pitch-shifting method would be required for changing the *sample* note. Nevertheless, since the synthesizer should be able to replicate all pitch related performance, including *vibrato*, *portamento*, pitch attack, and other pitch related effects, it is necessary to use one module to apply pitch changes to the *samples*. Also, in some situations, a pitch change could be required for correcting some internal *samples*. Even *non-vibrato samples*[26] may have slight pitch variations that need to be compensated, or internal *vibrato samples* that require to be flattened or have a different vibrato pattern.

There are many ways to change the pitch of an audio stream. The simplest way is to change both pitch and time, by simply accelerating or slowing down the audio stream. One of the first techniques for changing pitch consisted on recording audio on a magnetic tape recorder and playing it back with a different speed [Laroche, 1998]. The same can be done in the digital domain, by recording audio and playing it back with a different sample rate (or through resampling). The main problem with these types of approaches is that, by changing pitch, we are also changing time (making it faster or slower) and changing resonator (formants) frequency locations. For instance, increasing the pitch of a vocal recording in one octave, will double the speed of what is being said and will double the frequency of the formant locations (this is similar to reducing in half the physical dimensions of the vocal organs), creating a cartoon-like effect (also know as a “mickey mouse effect”).

To prevent these time and formant changes, other approaches are possible. For instance, using source-filter techniques, the vocal tract behavior could be extracted through the signal frequency envelope, and reapplied after changing the pitch, keeping the same resonating characteristics

[26] Audio material recorded from a singer with specified *non-vibrato* behavior

[Zolzer, 2002]. Methods like PSOLA[27] ([Zolzer, 2002]) allow the system to consider waveform frontiers and thus, being able to duplicate or remove cycles, maintaining the same time duration.

In the current context, the amount of the required pitch change is relatively small (typically within one semitone). This means that the impact of speed (time) changes or formant changes is also small. Although speech/formant changes still occur, one semitone pitch change will impact speech/formants in less than 6%, which in most cases is significantly relevant. Some informal tests were performed, by applying such pitch shifts, and the impact on speed variation or formant modifications were not considered relevant. As such, the decision was made to use simple frequency-time shifting mechanisms.

As mentioned before, such pitch changes can be done by continuously changing the output sample rate, which of course is not practical nor desirable (the output should use a fixed standardized sample rate). But the same goal can be obtained with two alternative methods: resampling or interpolation.

In resampling, the sample rate of the signal is increased P times (P is a integer), by simply inserting P-1 zero samples between the original samples. Then, a filter is applied to remove the new high frequency content (created with the new zero samples). On the final step, the sample rate is decreased Q times (Q is an integer), by simply extracting one sample within each Q interval. The amount of pitch change is defined by P/Q.

Interpolation tries to create signal values between the existing samples, using interpolation methods. For instance, based on the sample values at time instants 432 and 433, interpolation can predict what would be the signal value at time instant 432.7. By obtaining in-between sample values, interpolation can be used to change the original time/frequency characteristics of the original audio stream.

It is important to consider the impact of “continuity” during synthesis. Although the overlap-and-add approach can improve the continuity between frames, it is not sufficient to prevent some artifacts. Different frames will probably require different pitch values, and it is important that a smooth continuity is obtained, especially when using consecutive frames of the same internal *sample*. Imagining an internal *sample* of a long vowel that is used during several frames and needs its pitch changed to replicate a vibrato. If a pitch change is not required, the synthesizer would simply place the *sample* frames at the output, obtaining a perfectly natural sound, since it was totally based on a pre-recorded material without any type of signal processing, and the overlapping of frames would not create any artifact because both frames present the exact same audio content during the overlap time. But, if some pitch processing is required, it is important to keep that same continuity.

[27] Pitch-synchronous Overlap and Add

Returning to the resampling vs. interpolation discussion, although the process of resampling may eventually create better results (especially for higher frequencies), it presents a drawback: the amount of pitch-shift must be the same within the processed audio stream. For instance, if working at a frame level, the pitch shift must be the same within each frame. This means that during the overlap, it is not possible to have a perfect continuity between adjacent frames of the same *sample*, if they have been processed with different pitch values. One of the advantages of interpolation is that the amount of pitch-shifting can change within the frame. For instance, one frame may have its pitch changed from an initial value p1, to a different value p2, to a different value p3, within the same frame. By smoothly changing the pitch within the frame, any frame may present an initial pitch value (equal to that of the previous frame), a middle pitch value (the desired pitch value for the current frame), and a final pitch value (equal to that of the next frame), obtaining a perfect continuity, since during overlapping of adjacent frames of the same *sample*, the exact same audio is presented in both frames. Figure 4.3 presents an exaggerated illustration on this concept.

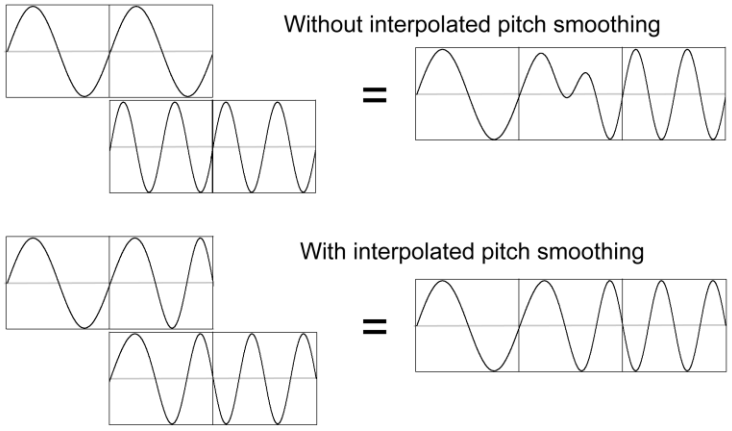

Figure 4.3 – Overlapping frames (Hann window) without interpolated pitch smoothing (top) vs. with interpolated pitch smoothing (bottom)

To better illustrate this pitch smoothing behavior, an alternative scenario is presented. A *sample* with a long vowel (MIDI note 60), non-vibrato, will be used during 20 frames. The desired pitch for the initial frames is 60.1, 60.3 and 60.4.

If using interpolation (with pitch smoothing), the first frame will consider an initial pitch shifting of +0.1 (no previous pitch value, so current pitch is considered), a middle pitch shifting of +0.1, and a final pitch shifting of +0.3. The second frame would have an initial pitch-shift of +0.1 (regarding the pitch value of the previous frame), a middle pitch shifting of +0.3 and a final pitch shifting of +0.4. One additional advantage of interpolation is that the frame can start anywhere, including in-between samples. In this case, the second frame would start exactly at the same point where the second half of the previous frame starts (regarding the original *sample*). During the overlapping between the first and second frames, both frames present the exact same signal: they apply the same gradual pitch change (from +0.1 to +0.3), and they use the same source material of the

original *sample*. Since the overlap is completely matched, a perfect continuity will occur since no artifacts (phase or any other type) will occur.

Using a resampling approach, the first frame would have a pitch change of +0.1 and the second frame will have a pitch change of +0.3, and during overlap minor artifacts could occur, since different material is presented in both frames with minor phase changes.

Within interpolation, several methods can be used, which present very different audio results in terms of noise and distortion. As such, several tests were done with MATLAB, choosing different interpolation methods for different situations. As expected, the higher the frequency, the higher the impact of the interpolation, since less samples exist per period.

Figure 4.4 represents results of one of these tests that show the output spectrum of a 1 kHz sinusoid with a pitch change of +1 semitone. The Figure 4.4 presents the magnitude spectrum of the original sinusoid, and the results of four different interpolation methods (nearest neighbor, linear, cubic, spline) and resampling (P=1000,Q=1059).

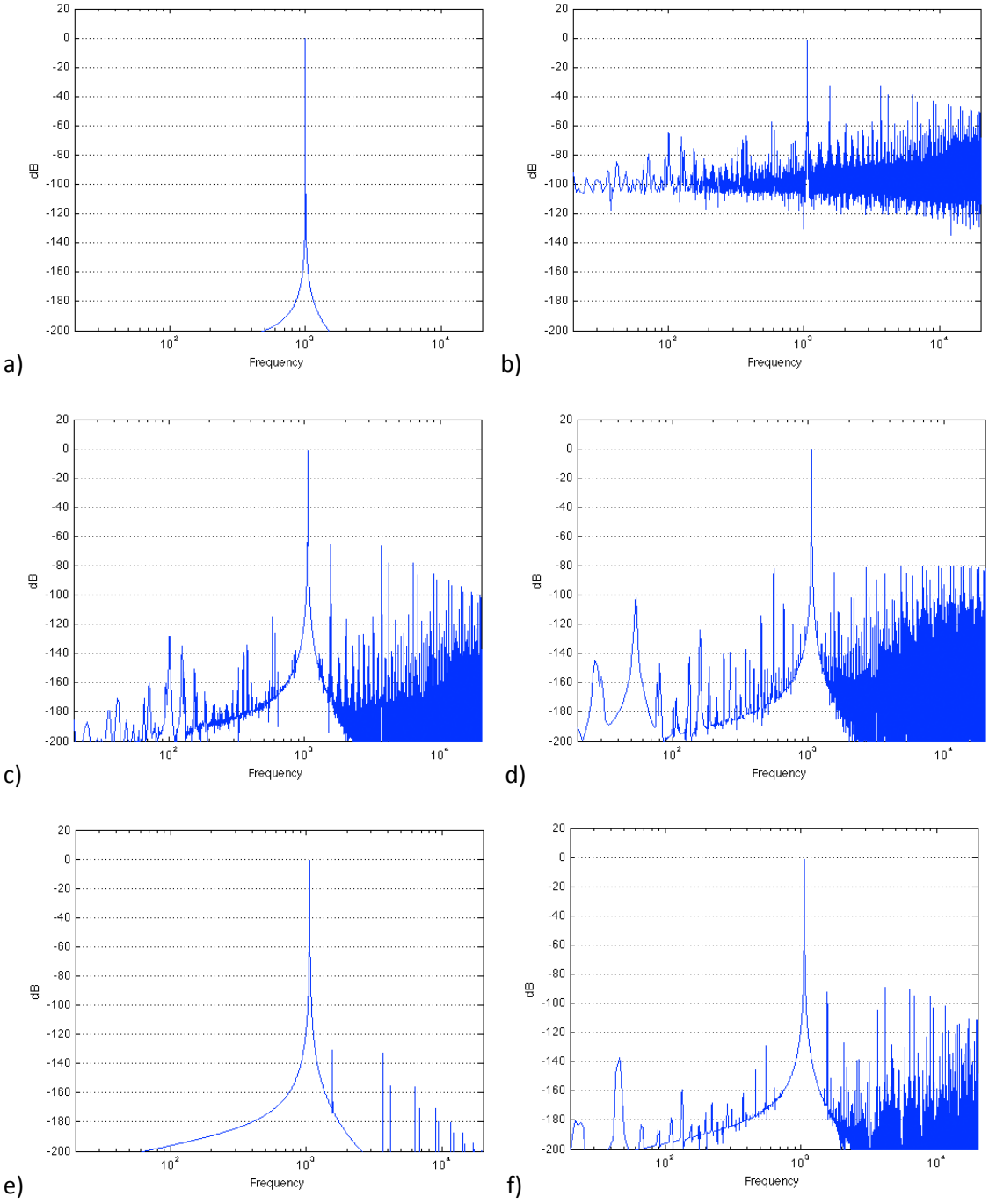

Figure 4.4 – Original sinusoid with 1 kHz (a) with a pitch change of +1 semitone using nearest neighbor interpolation (b), linear interpolation (c), cubic interpolation (d), spline interpolation (e) and resampling (f).

Analyzing the figure above, we conclude that spline interpolation (e) presents the best results in terms of noise and distortion.

Table 4.1 shows the dynamic range between the sinusoid amplitude and the frequency artifact with the highest amplitude, either for an original sinusoid of 1 kHz and for an original sinusoid of 5 kHz, considering a pitch change of +1 semitone.

Table 4.1 – Dynamic range for 1kHz and 5 kHz sinusoid, considering a pitch change of +1 semitone.

	Sinusoid 1 kHz	*Sinusoid 5 kHz*
Interpolation - Nearest Neighbor	32.12 dB	17,30 dB
Interpolation – Linear	64,27 dB	35,16 dB
Interpolation – Cubic	79,19 dB	38,06 dB
Interpolation – Spline	**129,69 dB**	70,89 dB
Resample	88,75 dB	**82,88 dB**

As can be seen in Figures 4.4 and Table 4.1, spline is the interpolation method with best results, presenting at some cases even better results that resampling.

Considering these results, and the benefits of interpolation, the choice was made to use spline interpolation as the pitch change mechanism of the synthesizer. It is important to mention that with same adjustments, resampling could offer the same benefits as interpolation. For instance, by having a large P value and a variable Q value within the frame, gradual pitch changes could be obtained within the frame. Nevertheless, since spline interpolation offers a good performance in terms of noise and distortion, such choice was not implemented.

4.3 Extracting Pitch

Since singing is usually used within a musical context, the extraction of pitch presents a fundamental role in the proposed resynthesis process. Although polyphonic pitch extraction methods exist (e.g. [Walmsley, 1999b], [Klapuri, 2008]), they still present high error rates. As such, and being the singing voice a monophonic musical instrument, the decision was made to consider only monophonic audio streams as inputs of the system. This limitation will highly decrease the number of pitch errors, without presenting a significant impact on the possible applications for the system.

Research on pitch extraction has being done for several decades and many methods exist today (e.g. [Noll, 1967], [Noll, 1969], [Sun, 2002], [Cheveigné, 2002]). Instead of creating a new method, the decision was made to evaluate existing methods and choose one of them, returning the research focus back to the resynthesis of the singing voice.

To choose the best way to extract pitch from the original audio stream, tests were made with several monophonic pitch extraction methods. Among these methods there were simple ACF, AMDF and SDF methods[28]; Cepstrum [Noll, 1967]; Harmonic Product Spectrum (HPS) [Noll, 1969] method with several variations; the SHRP [Sun, 2002]; unpublished Cepstrum-based method; and

[28] Looking for peaks of the autocorrelation function (ACF), average magnitude difference function (AMDF) and squared difference function (SDF).

the YIN method [Cheveigné, 2002]. Using monophonic singing voice recordings, these methods were tested and their results evaluated. Within the tested methods, YIN and SHRP were the ones that presented the best results.

Additional tests were also done, to achieve a better evaluation between both methods (YIN vs. SHRP). Table 4.2 presents the error rate using a song file ("Tom's Diner" – Suzanne Vega) and a sound library material ("EW/QL Voices of Passion"). Although the sound library material also presents consonants, the duration of the vowels is much longer, which results on a smaller error rate.

Table 4.2 – Test results for pitch extraction error rate, for YIN and SHRP methods (lower is better).

Method	***Error Rate ("Tom's Diner")***	***Error Rate (VoP sound lib)***
YIN	**10.07 %**	**2.53 %**
SHRP	11.66 %	2.74 %

YIN produced slightly better results. Probably by being a variation of auto-correlation (using square differences instead of multiplications), which is known to have good results in the presence of noise, allowed YIN to get good results with consonants that may present a noise-like behavior. Besides pitch, the YIN method also outputs an aperiodicity parameter that can be used as a measure of confidence or as a voicing measure.

To improve the output results, many pitch extraction systems use a smoothing approach, allowing them to recover from small pitch errors on isolated frames. For instance, if an isolated frame presents a pitch value that is significantly different from its two neighbors, the probability of being an error is very high. Most smoothing approaches are based on median smoothing: the pitch value of the frame *n* is equal to the median value of the available pitches between frame *n-k* and frame *n+k*. Also, when extracting pitch from a singing voice (or speech), most pitch errors occur during unvoiced situations, since the signal is so aperiodic that a pitch cannot be extracted.

Taking these situations into consideration, and based on experiments, the following pitch smoothing method was implemented:

1. Initially, pitch and aperiodicity values are extracted from the original audio stream, considering a hop size of 32 samples (window size of 1024 samples at 44.1 kHz sampling frequency). Pitch values where the aperiodicity is too high (>0.4) or pitch values outside the predefined singer range are discarded (receiving a NULL value).

2. Pitch values are gathered in groups of 16. Each one of the groups will output a single value for pitch, which means that the original hop size of 32 samples will act as a hop size of 512

samples. Within each group, if more than 8 pitch values are valid (not NULL), their median value is outputted, else the pitch value of the last group is maintained.

The median smoothing allows the system to correct some errors (especially on short unvoiced regions). Although pitch errors may continue to exist, their impact is minimized by using previous pitch values, which maintain the pitch context.

Figure 4.5 presents the extracted pitch for a singing fragment ("Tom's Diner" – Suzanne Vega), using SHRP (top), YIN (middle) and the proposed smoothing method on YIN (bottom). As can be seen, the proposed method presents a more stable behavior, overcoming the usual pitch errors of unvoiced passages.

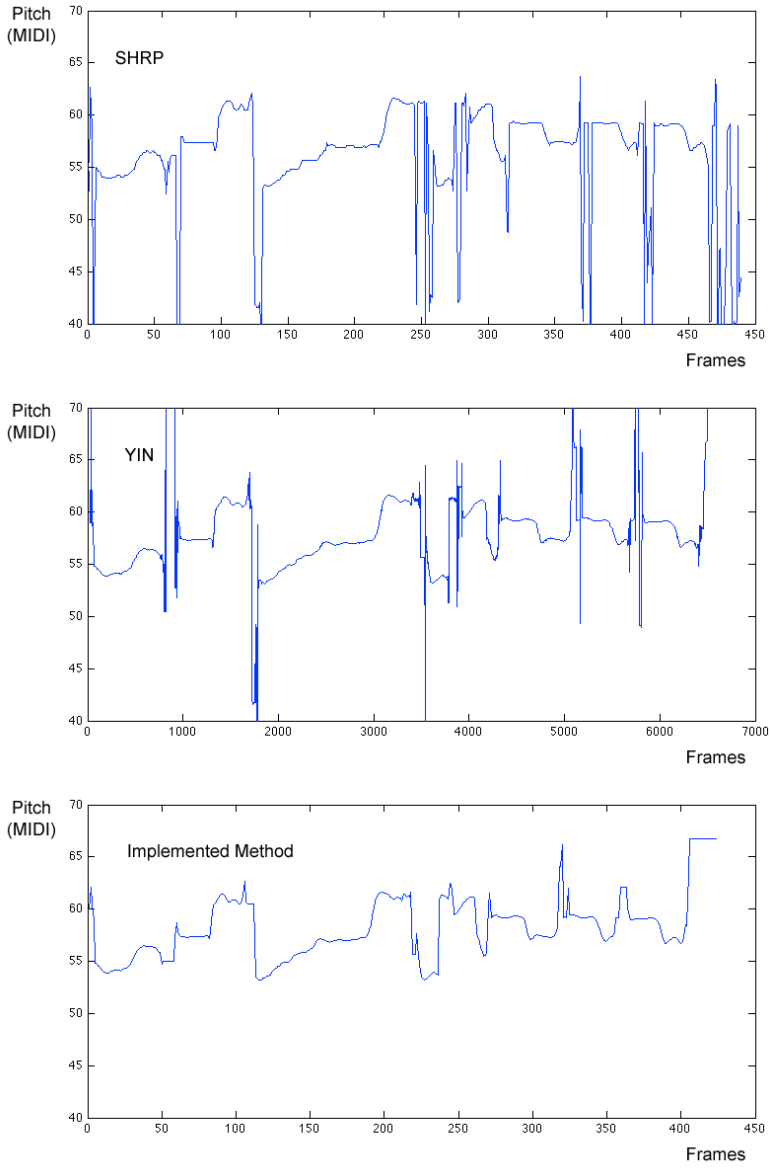

Figure 4.5 – Pitch extraction of "Tom's Diner" (Suzanne Vega) using SHRP (top), YIN (middle) and the final implemented method (bottom).

4.4 Replicating dynamic behavior

To obtain a good resynthesis of the original audio, it is important to replicate the same dynamic behavior, i.e., to be able to replicate the sound intensity behavior of the singing performance. Some signal processing parameters correlate well with the perception of loudness, energy being probably the most used one.

Some tests were done, trying to extract the dynamic behavior of a singing recording and to replicate the same dynamic behavior over different output audio materials (e.g. long vowels, sinusoids, noise).

During these initial tests, the energy of the original audio was extracted, considering several window sizes, and the same energy levels were reproduced on the output audio file, considering windows with the same size. As inputs, several different types of dynamic behavior were tested, including crescendos, diminuendos, staccatos, or ordinary singing performance.

As expected, the window size has an important role: too large windows do not capture the full dynamic behavior, especially in very quick *staccato* passages; and too short windows begin to create artifacts (especially distortion), by making changes at a waveform level. The best trade-off was obtained with window sizes around 23 ms (1024 samples at 44.1 kHz sampling frequency).

Regarding the hop size (overlapping), it does not seem to be too relevant. Tests with overlapping values between 99.9% (one sample hop size) and 0% were done, and as long as some overlap exists (e.g. 50%), the system presents equivalent good results.

Tests were also done using Hilbert absolute value, instead of energy, but it presented similar results.

Since the quality of the obtained results was more than satisfactory, tests with additional dynamic extraction approaches (for instance, considering attack and release times) were not implemented or tested.

4.5 Replicating phonemes

To be able to replicate the phonetic behavior, the system can use different approaches. Initially both “phoneme extraction” and “phonetic typewriter” approaches were tested, but better results were obtained using “phonetic similarity” approaches.

4.5.1 Phoneme extraction

Like the extraction of energy or pitch, the initial experiments on the phonetic domain followed the same approach, trying to detect, for each frame, what phoneme was been said or sung.

Since there is not available any data set with singing material and phonetic labeled information, the initial work was done with the TIMIT dataset [Garofolo, 1993], well known for its use in speech

research, which includes hundreds of speech phrases with labeled information regarding its phonetics.

Several phoneme classifiers were implemented, using different machine learning approaches. Since the extraction of phonemes cannot be done independently at a frame level (the information of previous frames is vital to the correct identification of some phonemes), special attention was given to the memory and feedback features of the used machine learning methods.

During these initial experiments, audio streams were converted to MFCC coefficients (C_1 .. C_{12} coefficients + C0) with first and second order derivates, a common step in several speech recognizer systems [Becchetti, 1999]. Regarding the used frameworks for each machine learning method, Matlab was used for Neural Networks (although Neural Networks with GA/PSO learning were implemented directly in C++), SVM^{ligh} was used for SVM [Joachims, 2002], and HTK was used for HMM [Cambridge, 2000].

Neural Networks

The first tests used neural networks (NN). Several models were used [Haykin, 1999], including simple MLP, RBF, recurrent models (Jordan, Eldman), with and without neural-network ensembles. Also, besides "traditional" NN learning methods, evolutive learning methods were used to evolve the weights of the neural network. Tests were done with Genetic Algorithms (GA) and Particle Swarm Optimization (PSO).

To have a memory feature within the Neural Network without the use of feedback (recurrence), a special architecture of neural networks was also developed. Within an MLP model, the "classic" perceptron unit is replaced with a perceptron pair with a D latch (Figure 4.6). The D latch will maintain its previous value, unless the Clock (CLK) input is activated, which outputs the D input. Instead of feedback, which tends to present a more short-term memory capabilities, the use of a latch would allow a better long-term memory. With the new architecture, classic back-propagation learning methods are no longer valid, but learning could be handled by evolutive approaches (GA or PSO). Nevertheless, the performance of this system was low (40% less correct phonemes) when compared to other "standard" NN architectures, so this idea was abandoned.

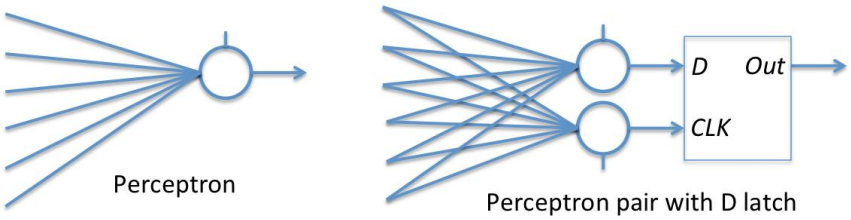

Figure 4.6 – A conventional perceptron and its replacement by a perceptron pair with a D latch.

Support Vector Machines

Support Vector Machines were also used, with different Kernels. Since SVM only considers binary class problems, some adjustments were needed to support multiple classes (multiple phonemes), as mentioned in section 2.5.2.

By using previous outputs as inputs of the system (feedback), a memory feature could be added. During training, considering the correct output of the previous frame as input, traditional SVM learning could be used. The outputs of the previous frames could be directly connected as inputs, or a binary processing could be added (transforming output values in the range [0 1] into a single bit (0,1)).

Hidden Markov Models

The commonly used Hidden Markov Models (HMM) were also tested, since HMM is probably the most common method for speech recognition. Although a tri-phone system presented a slightly better result (considering blocks of three phones), mono-phone was considered due to its better independency of the used language (tri-phone systems only consider the three phone situations encountered on the training set).

Singing material

After exploring many different approaches on the speech domain, using TIMIT, the focus was changed to explore the phoneme feature extraction on singing material. Although the methods allow relative good behavior extracting phonemes in speech materials (especially HMMs), their performance is reduced significantly when dealing with singing voice. For instance, HMMs, that presented a success rate of near 60% on speech material (TIMIT), dropped its performance to values near 15% on singing material. Either by training the HMM system with speech or training the system with singing, both presented a low success rate (>15%) on phoneme extraction on singing material.

4.5.2 Phonetic Typewriter

Due to the low success rate of phoneme extraction on singing material, other approaches were explored. One of such approaches was the concept of phonetic typewriter, proposed in [Kohonem, 1988], with a system that is able to represent, by activating points on a 2D mesh, what is being said. Depending on the phonemes, different points of the mesh are activated, being based on Self Organized Maps (SOM), an unsupervised neural network. Contrary to supervised learning, where the right “answer” is provided during training, in unsupervised neural networks the system simply analyzes the training examples and “distributes” them over its “output” surface. Similar inputs will “fire” similar outputs. This means that although the system does not know which phoneme is present at the input, it is able to automatically group similar phonemes together.

During the test phase, each frame (MFCC information) will fire one of the SOM outputs, that is identified by its space coordinates. Then, the system would look into the internal sound library for

the frame that, presented to the SOM, fired the output with the least distance from the original frame. The idea is to choose, from the sound library, the frame that is phonetically more similar to the original one. The chosen frame is then used on the output audio stream.

The best results were obtained with 2D triangle mesh and 1024 outputs, receiving MFCC coefficients (C0..C12) and first and second order derivatives as inputs. Tests were also done with other parameters, but similar or lower results were obtained.

4.5.3 Phonetic Similarity

Another possible approach to replicate the phonetic content of a singing recording is based on "phonetic similarity", using signal-processing parameters known to correlate well with phonetic information (e.g. MFCC, LPC, PLP). By analyzing the original audio stream with these signal-processing methods, the system can look on the internal sound library for fragments with similar content, using a distance or error measure.

As such, an Euclidean distance system was created, summing the square of the differences between the obtained coefficients. For each original audio frame, the system searches for the frame (from the internal sound library) that presents the least distance to the original one.

Many signal processing parameters were tested, including:

- MFCC, with 12 coefficients (C_1 … C_{12}), considering or not the C_0 coefficients, and considering or not the first and second derivatives;
- LPC coefficients;
- LPC frequency response, using LPC coefficients and converting them to their frequency response, with linear or a logarithmic scale;
- LPC Itakura-Saito distance, using LPC coefficients;
- PLP and RASTA-PLP;
- Spectrum;
- Filter banks.

By listening to the output audio streams, the ones that presented the best intelligibility were:

- MFCC, with C_1...C_{12} (no C_0, neither derivatives);
- LPC frequency response, with log amplitude scale;
- LPC Itakura-Saito distance.

4.5.4 Comparative resynthesis tests

To evaluate these possible approaches on a resynthesis scenario, tests were implemented. Using the work done on the synthesis module and on the extraction of pitch and dynamics, the phonetic approaches were tested side-by-side, trying to achieve a resynthesis system, allowing the evaluation of the final audio result.

Three banks of tests were implemented:

- Phoneme extraction approach
- Phonetic typewriter approach
- Phonetic similarity approach

Since there is not a method for numerically evaluate the audio quality of the results, either in terms of intelligibility or presence of artifacts, the audio results were evaluated by listening tests done by the author.

Phoneme extraction approach

Using some of the methods mentioned in section 4.5.1, the system would try to extract which phonemes were presented at the input audio stream. During synthesis, frame blocks with the same phoneme are replaced with audio fragments, from the internal sound library, with the identical phoneme.

The best results were obtained with HMM, either trained with TIMIT data set or with randomly created syllables from the internal sound library. But the obtained audio results were still very disappointing, especially in terms of intelligibility.

Phonetic typewriter approach

Using the implemented system of a "phonetic typewriter" (section 4.5.2), the resynthesis tests were evaluated. The obtained audio results were slightly better that the ones with HMM, although with more artifacts (HMM had transitions only between identified phonemes, where SOM may create much more transitions, since it works at a frame level). Nevertheless, the intelligibility was still very low (better, but low).

Phonetic similarity

Based on the concept mentioned on section 4.5.3, the resynthesis tests used an Euclidean distance within a MFCC domain. Each original frame was analyzed with MFCC (C_1...C_{12}, no C_0, neither derivatives), and replaced with the internal sound library frame with the lower Euclidean distance.

Although the obtained results presented more artifacts than previous approaches, the obtained intelligibility improved immensely, allowing the listener to perceive much more phonemes than in the case of previous approaches.

Based on the obtained results with all the approaches, the decision was made to use a phonetic similarity approach (base on signal processing parameters), in detriment of phonetic classifiers or unsupervised learning.

4.5.5 Measuring phonetic similarity

Using a phonetic similarity approach, the system must analyze the original audio frames, and select from the internal sound library, the frames that present the strongest similarity with the original ones. As such, the way phonetic similarity is measured will have a fundamental role and

impact on the system performance. Nevertheless, it is equally important that each audio fragment concatenates well with the previous one, or abrupt transitions and artifacts will appear. Both concerns are addressed with the concepts of “target cost” and “concatenation cost”.

Many concatenative synthesis systems, either for speech or music, use the idea of target cost and concatenation cost [Hunt, 1996]. To choose which audio fragment should be selected and placed on the output, these concatenative systems consider a target cost, responsible to ensure that the chosen fragment is similar to the desired sound (target), and a concatenation cost, responsible for ensuring that the chosen fragment does not present abrupt transitions regarding the previous fragment. Accordingly, the unit selection module searches for the best unit sequence (sequence of audio fragments), that presents the lowest overall cost (equation 4.1).

$$Overall\ Cost = TargetCost + ConcatenationCost \tag{4.1}$$

The same principles were implemented in the proposed system. Based on the original audio, the system will search from the best sequence of audio fragments (from the internal sound library) that presents the lowest overall cost, considering both target cost and concatenation cost. Target cost measures the distance between the original audio frame and the selected audio frame that will replace it. Concatenation cost measures the distance between a selected audio frame and the next selected audio frame. If two selected frames were adjacent frames on the internal sound file, a concatenation cost of zero is considered.

Target Cost

The initial tests mentioned in section 4.5.3 already presented some information regarding which signal based parameters would correlate well with phonetic information on singing material. Based on the three best signal parameters (MFCC, LPC frequency response, and LPC Itaruka-Saito distance), additional tests were performed.

By combining the three parameters on an Euclidean distance, better results were obtained. Better than using a single parameter or even a pair of parameters, the best results were found with the combination of all the three domains.

Since each domain has different value ranges, it was important to normalize their values to make sure that neither one of them masks the others. As such, normalization is obtained by dividing each domain by a *norm* value, which equals the mean error value within each domain.

Later, a fourth domain was added to the main equation: aperiodicity (voicing). Although aperiodicity by itself cannot be used as a phonetic distance measure, its addition to the Euclidean distance complements the final result, allowing the system to prevent voiced frames to replace unvoiced frames, and vice-versa.

As such, the final equation for Target Cost is presented at equation 4.2, considering an Euclidean distance within four domains: MFCC domain (equation 4.3); LPC Frequency response domain (equation 4.4), LPC Itaruka-Saito distance (equation 4.5), and the Aperiodicity domain (equation

4.6). The division by 2 in equation 4.2 acts as a normalization (*sqrt*(1+1+1+1)=2). MFCC domain considers 12 coefficients ($c_1 \ldots c_{12}$), disregarding c_0; LPC Frequency response considers the frequency response of 12 LPC coefficients (resampling at 10 kHz) with 128 bins and a logarithmic amplitude scale; LPC Itaruka-Saito distance considers the same 12 LPC coefficients (resampling at 10 kHz), but using a symmetrical version of the distance[29]; Aperiodicity is based on the YIN aperiodicity[30].

$$D_{i,j} = \frac{\sqrt{D_{MFCC}(i,j)^2 + D_{LPC\ resp}(i,j)^2 + D_{LPC\ dist}(i,j)^2 + D_{Ap}(i,j)^2}}{2} \quad (4.2)$$

$$D_{MFCC}(i,j) = \frac{\sqrt{\sum_1^{12}(c_n^i - c_n^j)^2}}{norm_1} \quad (4.3)$$

$$D_{LPC\ resp}(i,j) = \frac{\sqrt{\sum_1^{128}(X_n^i - X_n^j)^2}}{norm_2} \quad (4.4)$$

$$D_{LPC\ dist}(i,j) = \frac{\left[D_{IS}\big(LPC(i), LPC(j)\big) + D_{IS}\big(LPC(j), LPC(i)\big)\right]}{norm_3} \quad (4.5)$$

$$D_{Ap}(i,j) = \frac{|Ap(i) - Ap(j)|}{norm_4} \quad (4.6)$$

Concatenation Cost

Initially, the system considered the same formula (equation 4.2) for obtaining target cost and concatenation cost: Instead of comparing an original frame and an internal frame (target cost), the same formula was used to compare two internal frames (concatenative cost).

But the goal of concatenation cost is essentially to prevent abrupt transitions between frames, looking for smooth passages. Although a phonetic measure like equation 4.2 could be used for that purpose, a better method has been investigated.

Thus, additional tests were performed, trying to find signal parameters that could be better fitted for that purpose. One of the most interesting tests consisted in taking singing audio recordings, and see which signal parameter changed the less between adjacent frames when compared to random selected frames. Of all the tested parameters, the LPC frequency response was, by far, the method with the best results, showing that it might present a good candidate for a concatenative measure. The concatenation cost formula of the system changed from equation 4.2 to equation

[29] Original LPC Itakura-Saito distance is not symentrical: Dist(a,b) ≠ Dist(b,a) [Itakura, 1970].

[30] As mentioned in section 4.3, the YIN method [Cheveigné, 2002] also outputs an aperiodicity value.

4.4, and a slight improvement on the audio results were obtained. Other scenarios were also tested but without significant improvement.

As such, the final concatenation cost was based only on the LPC frequency response, as in Equation 4.4.

4.5.6 Searching for the best unit sequence

The main goal of the unit selection module is to select the sequence of internal units that will be used during synthesis. With the definition of a phonetic similarity distance (cost), the best unit sequence can be obtained as a result of a searching approach.

Although this search could be simple when considering only the target cost, the addition of the concatenation cost highly increases the complexity of the search process.

When considering only the target cost, search is a straightforward process: for each frame of the original audio (*i*), look for the internal audio frame (*j*) that presents the lowest cost (equation 4.7).

$$out(i) = \arg\min_{j} \; TC(i,j) \tag{4.7}$$

On an original audio stream with *n* frames and an internal sound library with *m* valid frames, an exhaustive search would be done with *n.m* comparisons. But, when concatenation cost is also taken into consideration, the search process becomes much more complex, since now all the sequence must be taken into consideration, which represents m^n possible sequences that need to be evaluated.

As such, two different approaches were used: Heuristics and Viterbi.

Heuristics

The initial approach used on the system was based on heuristics, i.e., a specific set of steps was chosen to find the best sequence. After many variations, the set of heuristics that presented the best results were:

Step 1 – Do a "target cost only" search: for each original frame, choose the frame from the internal library that leads to the lowest target cost.

Step 2 – Apply both "target cost" and "concatenation cost" to the initial candidate solution from step 1.

Step 3 – For each unit of the current candidate solution, try to replace it with every other frame from the internal sound library. If a better overall cost (target and concatenation) is obtained, then the candidate solution is changed to reflect that situation.

Step 4 – For each adjacent two units of the current candidate solution, try to replace them both with every other consecutive frame sequence from the internal sound library. If a better

overall cost (target and concatenation) is obtained, then the candidate solution is changed to reflect that situation.

Step 5 – Step 3 and 4 are repeated until no more changes on the candidate solution are obtained.

These heuristics allows the system to find an interesting solution for the problem. In most cases the system will not be able to find the optimal solution, due to being stuck at local minima, but it presents much better sound results than "target-cost-only" searches. Since the heuristics tend to merge frames together (step 4), using consecutive frames of the internal sound library, and once these frames are merged, the heuristics never split them apart, the output result presents an interesting smooth behavior.

Segmentation

In the context of the above heuristics, a segmentation approach was also tested. Instead of searching for all possible frame sequences, looking for the sequence with the lowest overall cost, a simplification could eventually be made. If the system was able to analyze the original audio stream and detect the boundaries of each phoneme, the resynthesis could eventually consider that within each phoneme, only one audio fragment would be required. This approach could bring several benefits: a simpler search process (only one audio fragment per phoneme, instead of one audio fragment per frame); and fewer transitions between units, creating less artifacts and a smoother concatenation.

In this type of scenario, with phoneme segmentation, it is important to analyze the impact of false positives and false negatives. A phoneme transition that is not detected (false negative) may present some problems, since the same audio fragment must be used for two phonemes, and considering the amount of possible phoneme pairs with the timing requirements, it may be difficult to find an internal fragment with such characteristics. Regarding false positives, a phoneme transition that is wrongly detected, is no relevant in terms of its impact, as the system is still able to ignore such transition and use a single fragment to cover many transitions. By adding a segmentation phase, we are simply reducing the search space. Instead of allowing new fragments to appear at each frame, we are reducing the number of locations were fragments may start.

Taking this into consideration - that is better to detect false transitions than undetecting real transitions - the decision was made to create a simple phoneme detection approach by looking at local peaks on the differences between adjacent frame information. By calculating the difference between the LPC frequency responses of the original adjacent frames (equation 4.4), a measure is created that represents the changes in what is being said. With that measure, local peaks will represent the locations where the phonetic changes are more accentuated. And although many false positives will exist, they would not create a problem. Besides LPC frequency response, other domains were tested but did not present significant perceived improvements on the final results.

State-of-the-art phoneme segmentation methods were not tested, since even manual segmentation did not present a significant role on the quality of the system[31].

Viterbi

The amount of all possible sequences is huge: m^n (where *m* represents the amount of valid internal frames available and *n* represents the number of frames of the original audio). For instance, a sound library with 100 seconds of audio material for each available pitch (about 8.600 audio frames), if used to replicate a 10 second original audio recording (about 860 audio frames), will create 8.600^{860} possible combinations. From a practical point-of-view, it is simply not possible to test each possible sequence individually. But it is possible to highly reduce the complexity of the search process.

The total cost of the best sequence (target cost + concatenation cost), is simply the best-accumulated cost at position *n*, being *n* the last position of the sequence. And, to obtain the best accumulation cost at any frame *i*, it is only required to know the best accumulated cost at frame *i*-1 for each possible value at *i*-1.

Figure 4.7 represents how the accumulation cost at frame *i* is obtained. The frame *i* can be replaced with any of *m* possible frames. The left side of the Figure represents all possible *m* possibilities at position *i*-1, with their accumulated cost and the sequence that originated such cost. To calculate the accumulated cost of using frame 1 at position i, considering a frame 2 at position i-1, the system just takes the accumulated cost of frame 2 at position i-1, adds the target cost between frame 1 and original frame *i*, and also adds the concatenation cost regarding frame 2 and frame 1.

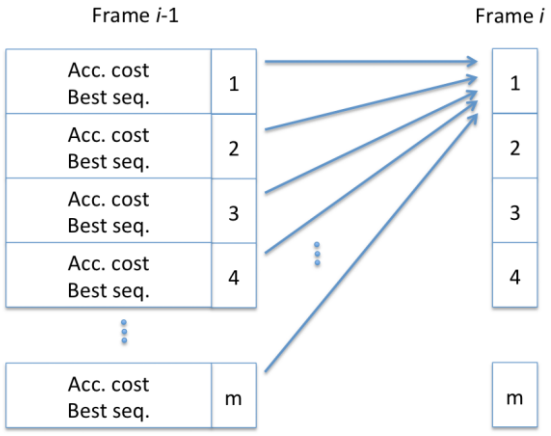

Figure 4.7 – Obtaining the accumulated cost at position i

By starting at original frame 1 and ending at original frame *n*, the final solution can be obtained by looking at the lowest accumulated cost at frame *n* and the sequence that originated such cost.

[31] Actually, the segmentation feature was later removed, and is not a part of the final system.

With this Viterbi approach, which is commonly used in similar problems (e.g. HMM training [Rabiner, 1989]), it is possible to find the optimal solution, without the need to individually test each possible solution.

It is important to mention once again that this simplification only works due to some assumptions. If the final costs included more complex methods than simple accumulations over time, or rules that did not comply with the Viterbi concept, such application could not be possible. For instance, a rule saying that a frame of the internal sound library could not be used more than 4 times over a period of 6 seconds, would create a problem, since the best accumulated cost at position *i* would not depend only at the best accumulated costs at position i-1, but eventually may depend on other past situations.

Pruning

Although the Viterbi approach highly reduces the search complexity, it still requires too much computational power and memory, due to the amount of possible frames (m) that need to be tested for each original frame.

To reduce the amount of possible frames for each situation, a pruning approach was implemented, forcing the system to only consider a subset of the available internal frames. The decision was made to use a value of 10% pruning, which means that only the 10% of the available internal frames are used during the search, instead of all available frames. As such, for position *i*, only the internal frames that present a target cost within the lowest 10% target costs for that position are used. For each position, the system analyze all target costs (between that original frame and any available internal frame) and extracts the value that makes the frontier between the best 10% rank, i.e., a kind of median calculation, but considering the 10% position instead of a 50% position. The obtained value is then used as a condition to decide if an internal frame should be used during the search on that frame position.

To obtain that 10% value, instead of sorting all target values and picking up the top 10%, a different method [Devillard, 1998] was used that requires must less operations, since a full sort is not required.

Tests were also done with other pruning values besides 10%, but lowering too much the pruning value would affect the quality of the results.

4.6 Putting all together

By combining the work done at each stage of the resynthesis process, a prototype of the system was implemented. Figure 4.8 presents the main stages of the system: Feature extraction, Unit selection, and Synthesis. Feature extraction is responsible for the extraction of the relevant information from the original audio stream; Unit Selection uses that information for selecting which frames from the internal sound library will be used during synthesis; and finally, Synthesis applies minor transformation to the selected audio frames and concatenates them to create the output result.

As mentioned before, the decision was made to consider an off-line system (not working in real-time), to prevent real-time requirements from affecting the quality of the results (future work can be done to achieve real-time support).

The system uses frame sizes of 1024 samples (which is equivalent to 23ms at 44.1 kHz sampling frequency) with hop sizes of 512 samples (50% overlap).

The Feature extraction and Synthesis modules were implemented in Matlab, and the Unit selection was implemented in C++, due to its CPU and memory requirements.

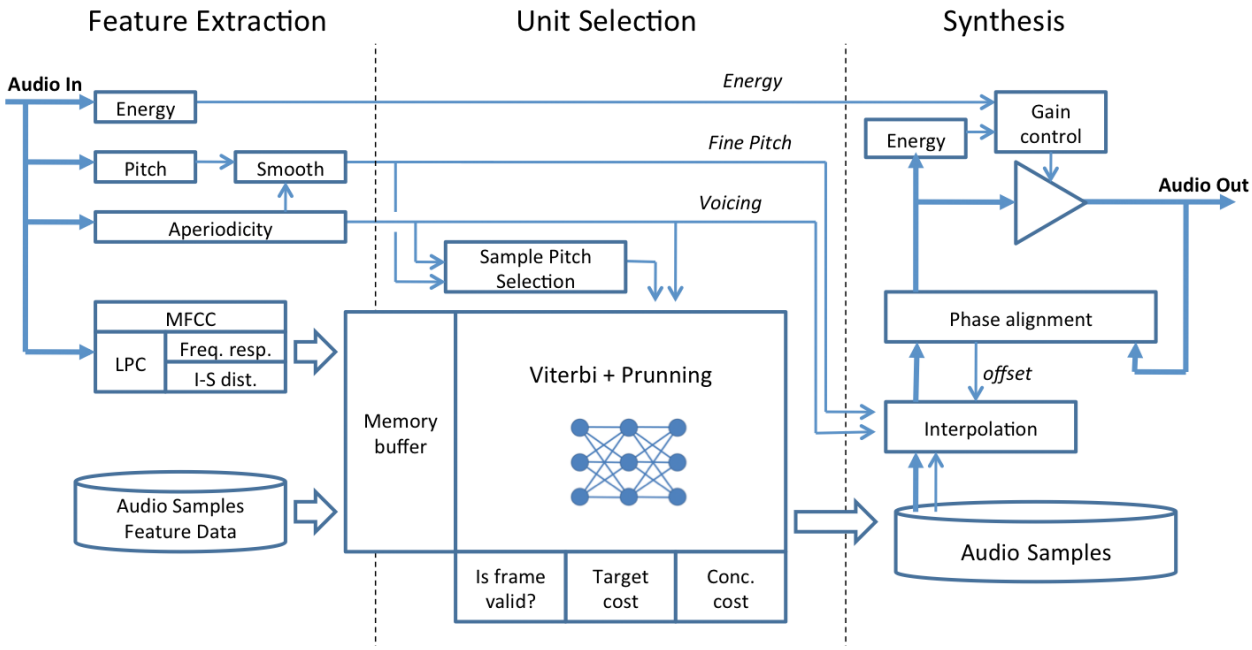

Figure 4.8 – Block diagrama of the Resynthesis system

Feature extraction

The Feature Extraction stage is responsible for the extraction of the relevant information from the original audio stream. The extraction of pitch and dynamic information (energy) was implemented as mentioned in the previous sections on the subject (section 4.3 and 4.4).

Regarding phonetic information, the system extracts MFCC coefficients and LPC coefficients (after a 10 kHz resampling). The LPC coefficients are then converted to their frequency response, using a logarithmic amplitude scale and 128 bins. Since LPC coefficients will also be used within an Itakura-Saito distance function, some initial preparation of the data is done to optimize the system. Aperiodicity information comes from the pitch extraction process (by using the YIN method).

The same type of information, which was previously extracted from the internal sound library, is also loaded on the system.

Unit Selection

The Unit Selection stage uses the extracted information for selecting which frames, from the internal sound library, will be used during synthesis. Since synthesis only applies a slight pitch transformation, for each original frame only internal frames with a similar pitch values can be considered. As such, a Sample Pitch Selection module will use the original frame pitch and voicing information, to filter out internal frames that do not present pitch values within the accepted tolerance (e.g. ± 0.5 semitones). During unvoiced passages, where pitch does not exist, any frame can be used.

A Viterbi-based process (with pruning) will then search for the frame sequence that presents the lowest overall cost.

Synthesis

The Synthesis stage applies minor transformation to the selected audio frames (interpolation and gain) and concatenates them to create the output result, as mentioned in section 4.2.

Internal Sound Library

The internal sound library consists of audio files with monophonic singing material. By working at a frame level, the system does not require any kind of segmentation or annotation, since unit frontiers does not need to be identified. As such, the system does not have any special requirement regarding the sound library – the audio files do not need any kind of annotation or any especial recording session procedure. Nevertheless, recording material with non-vibrato and sustained pitch singing could present slightly better results, since pitch changes could more easily force transitions to occur due to out-of-range pitch tolerance.

Increasing the size of the internal audio material, allows a better coverage of possible phoneme/pitch pairs, although it also increases the search space and the required computational power during unit selection.

The system uses two different sound libraries with singing *samples*. The first library corresponds to choir *samples*, from "EW/QL Symphonic Choirs" [EASTWEST, 2005], with 5 different ranges (Basses, Tenors, Altos, Sopranos, Boys), each file with a single phoneme/note, up to 1 seconds of duration. The library contains 8 vowels (*uh, ee, oo, ih, eh, oh, eu, ah*), 14 pitched consonants (*b, d, g, j, l, m, n, r, rr, th, v, w, y, z*) and 11 non-pitched consonants (*ch, f, h, k, p, q, s, sh, t, th, x*)[32].

The second sound library uses a completely different approach. It includes female solo recordings, from "EW/QL Voices of Passion" [EASTWEST, 2007], where each file consists on a word/note (46 words available), with durations up to 9 seconds. The used words were: *Bene, Breathe, Close, Dark, Death, Domine, Dream, Drown, Im, Fall, Fire, Fly, Gaia, Grass, Hasan, Hate, How, In, Len,*

[32] Each phoneme is not represented with the IPA notation, but using the notation from the original product manual [EASTWEST, 2005].

Love, Luxet, Ly, Mei, Ness, Of, Ooze, Pray, Priest, Row, Ruins, Run, San, Sing, So, Soft, This, True, Uram, Ventius, Ver, Vosh, Fortuna, From, Gravis, Is, Rain, The.

While the concatenation of choir *samples* tends to mask artifacts, solo voice *samples* are much more sensitive to the presence of such artifacts. Taking that into consideration, and also the fact that singing word recordings would better represent future applications of the system (instead of independent phoneme recordings), most work was done with the female solo sound library.

4.7 Improving the audio quality

Since concatenative approaches use pre-recorded material, they usually present a high sound quality within each unit. Most problems arise on the transition between units, changing from an audio fragment to the next, since a lack of continuity may appear. Unfortunately, these transitions problems cannot be solved simply by using cross-fading techniques.

To improve the quality of the resynthesis system, several features were added, especially regarding the transition between units.

4.7.1 Reducing the number of transitions due to pitch changes

Transitions may be originated due to phonetic or pitch changes. When pitch changes (e.g. a new note), the system will most likely consider a different unit (unless an audio fragment with a very similar phonetic and melodic behavior also exists on the internal sound library). But this forced transition may even occur with other pitch situations. Figure. 4.9 presents the pitch information extracted from a vibrato passage. As can be seen, the vibrato crosses many times the frontier between the E4 and the D#4 notes.

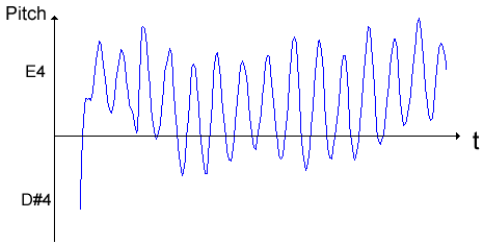

Figure 4.9 – Example of a vibrato passage that crosses the frontier between note E4 and note D#4.

Imposing a pitch tolerance of ± 0.5 semitones will force many transitions to occur on a passage like the one illustrated in Figure 4.9. Forcing different units each time the note frontier is crossed creates unnecessary transitions and less natural sounding audio. In contrast, increasing too much the pitch tolerance, although highly decreasing the number of transitions due to pitch changes, will increase the artifacts created by the pitch shifting itself. As such, a pitch tolerance of ±1.5 semitones presents a good trade-off: the number of transitions due to pitch changes may decrease significantly (less than 75% in some songs), especially during vibrato passages; and such pitch change will continue to present a very small amount of time/formants shifts during synthesis (±1.5 semitones creates a time/frequency change less than 9%).

But by increasing the pitch tolerance, the amount of internal frames that can be used in each

original frame also increases, forcing the unit selection process to require more time, memory and CPU, due to the increase of the search-space.

4.7.2 Reducing the number of transitions due to phonetic changes

In order to reduce the number of transitions due to phonetic changes, some methods can be used. For example, changing the weights assigned to target cost and concatenation cost, more emphasis can be placed on one of them. In this case, by increasing the weight of concatenation cost, the system will create much smoother results. But increasing too much the weight of concatenation cost will generate other problems, affecting the intelligibility of the results, with some phonemes (mostly consonants) being discarded in favor of better smooth transitions. Keeping that in mind, the concatenation cost had its weight slightly increased allowing smoother results without affecting the output intelligibility. On the proposed system, a value of 1.5 (as in equation 4.8) presents a good trade-off (values of 1; 1.25; 1.5; 1.75; 2; 3; were tested, looking for the best balance between intelligibility and audio artifacts)

$$Overall\ Cost = TargetCost + 1.5 * ConcatenationCost \quad (4.8)$$

Also, most unit selection systems, consider a concatenation cost of zero when joining adjacent frames[33]. On the proposed method, instead of a concatenation cost of zero, a negative cost is used, acting as an additional bonus for using adjacent frames. Of course, decreasing too much this negative cost will affect intelligibility. As such, a concatenation cost of -0,5 for adjacent frames has shown to present a good trade-off ensuring less transitions (and their artifacts) without affecting significantly the intelligibility of the output results.

4.7.3 Discarding low energy frames from the internal library

During the tests, it was noticed that some artifacts were created when using internal frames with a very small amount of energy, typically from “silence” periods or late reverberation reflections. These low energy frames, if chosen to replace original frames, besides not representing “true” vocal material, end-up being highly amplified, raising the noise level and decreasing the sound quality of the output results. As such, any frame from the sound library whose energy is more than 40dB below the highest frame energy on its audio file, is disregarded.

4.7.4 Preventing Frame Repetitions

Since the proposed method works at a frame level, a special phenomenon is likely to occur: in many cases, the same frame from the internal sound library would be used consecutively (for instance, 5 “times in a row”), which is not desired, because it will create phase issues and it does not provide the desired vocal continuity and naturalness. But the explanation for the phenomenon is quite simple - using the same frame repeatedly, originates a concatenative cost of zero, since there are not any differences between consecutive frames (it is the same frame over again). This problem does not exist on traditional concatenative systems, because they usually consider larger audio segments instead of frames, with differences between their first and last frame.

[33] If a concatenative system uses adjacent frames, then a perfect concatenation is naturally obtained since they are consecutive frames of the same recording.

To prevent this situation, an additional rule was imposed: the same internal frame cannot be used more than one time within a 10 frame interval.

4.7.5 Considering the effects of time shifts

During unit selection, for each original frame the system searches for the best internal frame that can replace it. But in reality, this one-to-one relation between original and internal frames may suffer from time shifts. The phase alignment process (within synthesis) may slightly shift the internal frame to achieve a better transition. For example, the unit selection may choose internal frame 466, but the synthesis module could end-up using the frame 466.2 (starting slightly after the beginning of frame 466 but before the beginning of frame 467). Also, the interpolation process that applies pitch changes (also within synthesis) creates slight time shifts, due to the compression or expansion of internal frames. For instance, if 15 original frames are replaced by 15 adjacent internal frames that require their pitch slightly increased, the synthesis module may eventually end-up using 16 internal frames (since to increase pitch much more samples are needed during the interpolation process). The accumulation of all these time shifts is likely to create a situation where the conditions taken into consideration by the unit-selection module may not be the actual conditions that occur during synthesis, since target costs and concatenations costs may not be the correct ones.

It is important to prevent this problem, but it is not practical for the unit selection process to calculate the exact information, which would require adding phase alignment processing to all frame combinations and calculating or accessing in-between frame information. Although the exact information cannot be accessed, an approximation can be made. Considering the original frame pitch, the system could easily calculate what is the Maximum Phase Offset (MPO) that could be required during the latter phase alignment process. Also, with the pitch values of the original frames and the pitch values of the internal frame candidates, it is possible to calculate the amount of Accumulated Time Shift (ATS) that would occur during the interpolation process.

With these time shifts estimations and a conservative approach (considering that the worst scenario is the one that occurs), the unit selection module quickly estimates realistic values for target cost and concatenation cost.

In this perspective, the Target Cost – TC(*i,j*) – will consider the Accumulated Time Shift (ATS) from previous frames and the worst scenario between no phase offset and Maximum Phase Offset (MPO) (equation 4.8)[34], where TC' represents the previously defined target cost (equation 4.2).

Concatenation Cost – CC(*i,j*) – will also consider ATS and the worst scenario between the four combinations of phase offsets (equation 4.9)[35], where CC' represents the previously defined concatenation cost (equation 4.4).

$$TC(i,j) = Max[TC'(i,j+ATS), TC'(i,j+ATS+MPO(j))] \quad (4.9)$$

[34] TC (*i,j*) – Target Cost between original frame *i* and internal frame *j*.

[35] CC (*i,j*) – Concatenation Cost between internal frames *i* and *j*.

$$\begin{aligned} &CC(i,j) \\ &= Max\big[CC'(i+ATS,j); \\ &CC'(i+ATS+MPO(i),j); \\ &CC'\big(i+ATS,j+MPO(j)\big); \\ &CC'\big(i+ATS+MPO(i),j+MPO(j)\big)\big] \end{aligned} \tag{4.10}$$

Since data in-between frames are not available, such values are obtained by linear interpolation, as shown in equation 4.10 and equation 4.11.

$$\begin{aligned} &TC'(i,j+\Delta) \\ &= TC'(i,j)*(1-\Delta)+TC'(i,j+1)*\Delta \end{aligned} \tag{4.11}$$

$$\begin{aligned} &CC'(i+\Delta_1,j+\Delta_2) \\ &= CC'(i,j)*(1-\Delta_1)*(1-\Delta_2) \\ &+ CC'(i+1,j)*(\Delta_1)*(1-\Delta_2) \\ &+ CC'(i,j+1)*(1-\Delta_1)*(\Delta_2) \\ &+ CC'(i+1,j+1)*(\Delta_1)*(\Delta_2) \end{aligned} \tag{4.12}$$

With this feature, the number of artifacts decreased, since unit selection is able to reflect the time shifts that might exist during synthesis, and take them into consideration with more realistic target and concatenation costs.

4.8 Resynthesis Evaluation

The evaluation of the results is always an important task in any research activity, and the same applies to this research work. The results were evaluated using listening tests with several users, allowing us to obtain a more representative and less biased assessment.

4.8.1 Listening Test Planning

Listening tests are a quite common tool in audio research, since many sound related behaviors can only be evaluated by humans. Although sound and psychoacoustic research continue to increase every day, there is still a long road before “audio quality” could be totally evaluated by non-human methods. As such, the decision was made of using listening tests to evaluate the resynthesis obtained results.

Test material

To test the proposed method, some audio fragments with solo female singing recordings (“*a capella*”) were used. Although there is not much audio material available, it was possible to gather audio fragments from some well-known female singers. Attention was made to choose voices with different voice styles from each other. The following material was used:

- Amazing Grace - LeAnn Rimes (duration of 16 seconds)
- Bohemian Rhapsody - Lauryn Hill (duration of 11 seconds)

- Frozen - Madonna (duration of 15 seconds)
- I Will Survive - Diana Ross (duration of 10 seconds)
- Tom's Diner - Susanne Vega (duration of 4 seconds)
- Whenever - Shakira (duration of 6 seconds)

Female voices were chosen due to the fact that the internal sound library also used female solo voices [EASTWEST, 2007]. Although the system also includes a choir sound library, most of the research was focused on the female solo sound library, since choir can more easily mask artifacts than a solo voice. For each song, the first vocal line was used, with the exception of Sharika's "Whenever" (the audio recording did not present a monophonic solo passage on the first line, so a different line was taken).

Each original audio fragment was used as the system input, and a resynthesized version was created.

Testing environment

Listening tests can be done remotely or implying physical presence in a special listening facility. In the latter case, all users share the same room (or similar ones) and are physically present during the tests. On remote listening tests, users are able to do the tests in their own listening space, and their physical presence is not a requirement.

Although presential listening tests offers some advantages over remote tests, in many situations it may be difficult or even impossible to gather a significant number of persons, with the required background, to do the listening tests. In our case, since it would be very complicated to gather several tens of users, preferably with audio or music backgrounds, to do the tests, and since in this case the listening environment does not have a significant influence on the results, the listening tests were conducted through a web page, allowing users from around the world to participate.

On the listening tests web page, users had access to the audio materials (original and resynthesized fragments), questions and response forms. All audio files (in .wav file format) were compressed on a .zip file (9.1 MB) that should be downloaded by each listener. The total amount of audio material was 124 second long, considering all original and resynthesized audio streams.

Listeners were invited to participate through audio and music-related mailing lists.

Questions

For each audio fragment, listeners were asked to answer the questions presented in Figure 4.10

1. Please specify the percentage of time that the resynthesized stream is similar to the original one, in terms of dynamic behavior (dynamics), melodic line (pitch) and phonetic content (phonetics).

Dynamics: ___ (%)

Pitch: ___ (%)

Phonetics: ___ (%)

2. Does the resynthesized audio stream present audio artifacts?

Please specify, for each type of artifacts, the impact on hearing discomfort, writing any value between 0 and 100, considering 0 (inaudible), 25 (audible), 50 (slight annoying), 75 (annoying), 100 (very annoying).

Impact of dynamics Artifacts (abrupt changes of levels, etc.): ___

Impact of pitch artifacts (wrong pitch notes, errors on melody line, abrupt changes in pitch, etc.): ___

Impact of wrong phonetics: ___

3. Does the resynthesized audio stream present noise? Please specify, the amount and impact on hearing discomfort. (AMOUNT: between 0 and 100, considering 0 [never], 25 [rarely], 50 [sometimes], 75 [often], 100 [all the time]; IMPACT: between 0 and 100, considering 0 [inaudible], 25 [audible], 50 [slight annoying], 75 [annoying], 100 [very annoying]).

Amount: ___ Impact: ___

4. Disregarding audio artifacts and timbre differences, how do you personally evaluate/score the resynthesized audio in terms of replicating the singer's unique performance (the way the singer sings the song)?

0 (worst case) - 100 (best case) _____

Figure 4.10 – Questions from the listening tests.

4.8.2 Listening Tests Results

The tests were available online for two months and had 35 participations, from heterogeneous backgrounds (researchers, audio professionals, amateur musicians).

Table 4.3 and Figure 4.11 present the results for the first question, regarding the percentage of time that the resynthesized audio is similar to the original audio, in terms of dynamics, pitch and phonetics.

Table 4.3 – Percentage of time that the synthesized audio is similar to the original audio, in terms of dynamics, pitch and phonetics (higher is better).

Song	*Dynamics*	*Pitch*	*Phonetics*
Amazing Grace	58.4 %	59.9 %	37.5 %
Bohemian Rhapsody	59.3 %	55.5 %	32.7 %
Frozen	72.6 %	75.6 %	62.8 %
I Will Survive	61.1 %	62.1 %	50.4 %
Tom's Diner	71.2 %	72.3 %	54.2 %
Whenever	53.0 %	56.2 %	40.5 %
Global	**62.6 %**	**63.6 %**	**46.4 %**

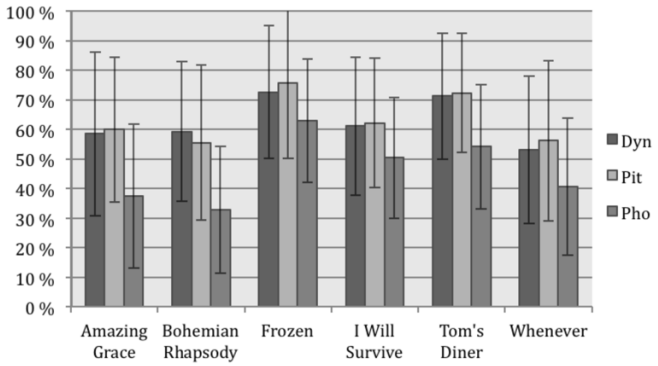

Figure 4.11 – Mean and standard deviation for the percentage of time that the synthesized audio is similar to the original audio, in terms of dynamics, pitch and phonetics (higher is better).

Some song fragments present better results than others. "Frozen" and "Tom's Diner" presented the best results, while "Bohemian Rhapsody" had the worst results. From the three domains, Pitch and Dynamics had consistently better results from the replication point of view than Phonetics, which is not a surprise due to the complexity of this last domain. The standard deviation (Figure 4.11) is quite high in all the three domains, which means that different listeners gave significant different answers.

Table 4.4 and Figure 4.12 present the results for the second question, regarding the impact on the hearing discomfort of several types of artifacts: dynamic artifacts, pitch artifacts, and phonetic artifacts.

Table 4.4 – Artifacts: Impact on hearing discomfort - dynamic artifacts, pitch artifacts, and phonetic artifacts (lower is better).

Song	*Dynamic*	*Pitch*	*Phonetics*
Amazing grace	49.4 %	51.1 %	57.2 %
Bohemian Rhapsody	38.7 %	40.7 %	55.4 %
Frozen	35.2 %	35.7 %	46.9 %
I will survive	39.9 %	46.9 %	54.1 %
Tom's Diner	44.0 %	49.1 %	53.0 %
Whenever	56.1 %	50.6 %	64.8 %
Global	**43.9 %**	**45.7 %**	**55.2 %**

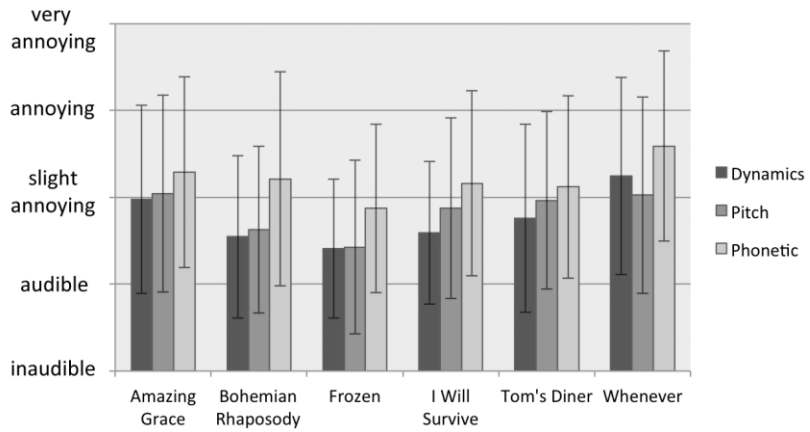

Figure 4.12 – Mean and standard deviation of the perceptual impact of artifacts: Impact on hearing discomfort - dynamic artifacts, pitch artifacts, and phonetic artifacts (lower is better).

Most fragments presented slightly annoying artifacts. Phonetic artifacts were the ones with stronger impact. Dynamic artifacts and pitch artifacts had slight smaller impact. "Frozen" presented the least amount of artifacts and "Whenever" presented the most artifact impact. The standard deviation continues to present high values.

Table 4.5 and Figure 4.13 present the results for the third question, regarding the amount of noise and its impact on the final audio stream.

Table 4.5 – Noise: Amount and impact on hearing discomfort (lower is better).

Song	*Amount of Noise*	*Impact of Noise*
Amazing grace	50.1	57.0
Bohemian Rhapsody	48.8	55.8
Frozen	38.7	45.1
I will survive	40.2	50.8
Tom's Diner	39.0	48.7
Whenever	53.9	60.8
Global	**45.1**	**53.0**

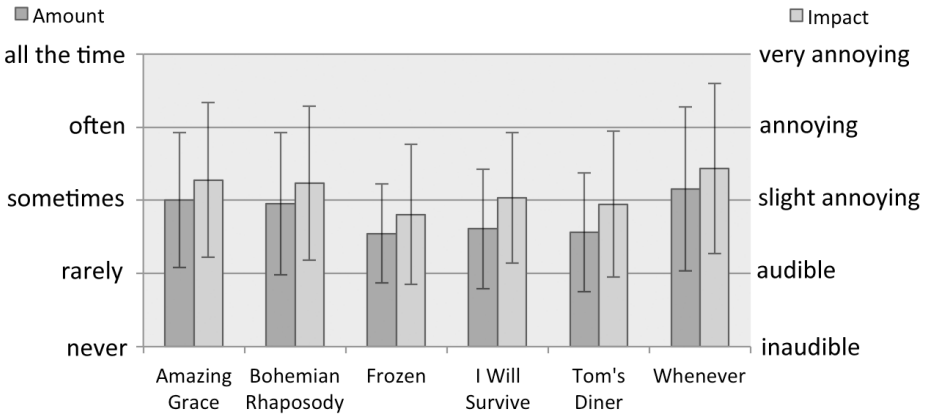

Figure 4.13 – Mean and standard deviation of the amount on noise and its impact on hearing discomfort (lower is better).

Listeners considered that "sometimes" the noise is present, creating a "slight annoying" impact on the hearing discomfort. Like the previous question, "Frozen" had the best results and "Whenever" presented the worst result.

Table 4.6 and Figure 4.14 present the results regarding capturing the singer's unique performance.

Table 4.6 – Capturing the singer's unique performance (higher is better)

Song	*Singer's performance*
Amazing grace	48.4 %
Bohemian Rhapsody	36.7 %
Frozen	67.0 %
I will survive	52.0 %
Tom's Diner	58.0 %
Whenever	43.0 %
Average	**50.8 %**

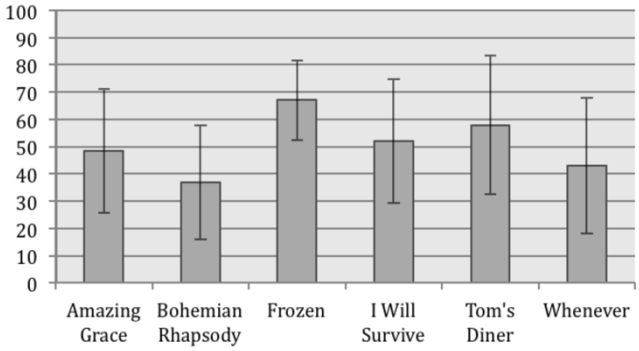

Figure 4.14 – Capturing the singer's performance (higher is better)

Although the system looses the timbre of the singers[36], users still gave a mid-range classification to the ability of capturing the singer's unique performance. "Frozen" and "Tom's Diner" presented the best results, and "Bohemian Rhapsody" presented the worst.

4.9 Discussion

Globally, from a proof-of-concept point-of-view the results were positive, although much work still needs to be done before a real world application is deployed on professional scenarios.

Being the most complex domain, it was no surprise that within the three domains (dynamics, pitch, phonetics), phonetics is consistently the one delivering the worst results.

Doing individual song evaluations, it is clear that some songs present better results than others. "Frozen" and "Tom's Diner" presented the best results, while "Whenever" and "Bohemian Rhapsody" presented the worst results. Probably factors like more nasal timbres, very quick phonetic sequences (that create too much transitions between units), or even too slow phonetic sequences (where transitions between units are more easily detected), are responsible for the lower scores of some songs.

Dynamics

The system seems to be able to capture the sound intensity behavior (dynamics) of the original signal, replicating well fast and slow passages, and most of the overall dynamics over the audio stream.

Nevertheless, the dynamic replication creates one special type of artifact when original voiced frames are replaced with unvoiced frames. Voiced and unvoiced frames tend to have different energy levels. With a more stable and harmonic content, voiced frames present higher energy levels. If an unvoiced internal frame is chosen to replace an original voiced frame, a high amount

[36] The goal of the system is to replace the original audio with different audio material, which has different timbre characteristics.

of gain is applied to maintain the same energy level, increasing the amplitude of very short transients and the background noise level. Although several attempts were made to handle this problem, including the addition of a voicing dimension on the phonetic target cost as mentioned in section 4.5.3, some artifacts still remain.

Pitch

The system behaves well regarding the extraction of the musical notes. There are not situations with octave errors; dissonant notes (where an original music note is replicated with a dissonant one, like a major second or a diminished fifth); or similar gross errors.

Most pitch related errors exist at a finer level, especially with pitch continuity between units. Due to the time-frequency resolution trade-off, the exact time-pitch information of the frame is not 100% correct, creating abrupt tune transitions. Although these pitch transitions are mostly within a semitone range, the artifacts are easily detected.

Phonetics

Most words are recognizable, but the system still presents limitations regarding the replication of phonetic content. Some consonants appear too “smoothed”, losing their fast transients, probably due to the weight of concatenation cost, responsible for decreasing the abrupt transitions between units. Other phonemes suffer from the presence of audible artifacts (e.g. abrupt transitions) that impact the phoneme perception. The worst scenario appears when a single phoneme is obtained with the concatenation of several different units within a short time period, loosing the phoneme identity from the perception point-of-view.

Music Performance

Although the system “replaces” the original audio stream with a new stream (loosing the original timbre), and even with the presence of some artifacts, the system is still able to capture the musical performance of the singer and to preserve a significant part of it on the output. In some songs, like Madonna’s “Frozen”, it is possible to identify the singer personal style on the output, even if the original audio is no longer there.

4.10 Summary

This chapter presented the work done during the development of a resynthesis approach for the singing voice.

Initially, the work was focused on the synthesis module, allowing it to better support a resynthesis environment.

Later, the replication of each domain (dynamics, pitch and phonetics) received attention, especially phonetics, which changed from a phoneme extraction approach to a phonetic similarity

approach. By testing different approaches within each problem domain, and combining music and speech techniques, in both analysis and synthesis fields, a prototype was created.

Some additional algorithmic decisions were taken to improve the audio quality and decrease the number and impact of the created artifacts.

Finally, an evaluation of the obtained audio results of the proposed resynthesis approach is presented using listening tests, including: how the listening tests were prepared, focusing on the audio material and questions; the results of the listening tests and their discussion, trying to better understand the major limitations of the proposed system. The results obtained showed that the system is able to replicate most of the information, although still presenting several audio artifacts.

The method and results presented in this chapter have been the object of several conference papers [Fonseca, 2010a], [Fonseca, 2010b], [Fonseca, 2011a], [Fonseca, 2011b].

5. Conclusions

Although the concept of resynthesis is simple, its implementation is very difficult, probably requiring several additional years of research work before achieving the required level of performance for professional applications. Nevertheless, it presents a high potential for many applications, from high-level semantic audio effects, to better control of virtual instruments.

In this dissertation, a singing voice resynthesis approach was proposed and implemented, allowing the user to control a singing synthesizer with his/her own voice. Although many artifacts are still present, preventing its use on real world scenarios, this PhD work has contributed to show that by merging analysis and synthesis methods from both speech and music research fields, new applications may be created on complex areas like the singing voice.

Originally, the goal was to use machine-learning mechanisms (or other intelligent approaches) to better achieve a resynthesis process. However, during the research, it became clear that lower level approaches presented better results, and most high-level approaches were abandoned.

The two major obstacles found during this research work were: probably the inexistence of a complete singing dataset, and the lack of a numeric metric for evaluating the system's results.

Regarding the singing dataset, the existence of a significant repository of singing material with pitch/phonetic annotations[37] would contribute for a better research work, probably allowing better results from machine learning approaches. Unfortunately, the required work and means for creating such dataset is beyond this PhD work. As such, a smaller and un-annotated dataset was used [EASTWEST, 2007].

A significant amount of work was done adjusting the parameters of the system, to achieve better results. By testing different target and concatenative costs, search methods, or synthesis approaches, and adjusting their internal parameters, many of these decisions were based on the personal listening evaluation of the author. Even with a trained ear, it is very difficult to identify and evaluate small differences between audio results, and to keep an unbiased opinion. On the other hand, it was also impractical to do frequent listening tests, with ear-trained people, for

[37] e.g. similar to TIMIT dataset [Garofolo, 1993], but with singing recordings

evaluating such decisions. Many attempts were done, trying to find[38] metrics capable of numerically evaluate the quality of the audio results (mainly in intelligibility and audio quality), but without success. The existence of such metrics would highly improve the research productivity on the subject.

As previously mentioned in section 1.5 ("Main Contributions"), this dissertation work resulted in nine conference papers (seven international conferences and two national conferences), a journal paper submission, an implemented prototype, and citations in books, PhD thesis, journal papers and conference papers[39].

5.1 Perspectives for Future Work

With the present system, there are many perspectives for future work, which could be grouped in four different areas: audio quality improvements; performance and real-time support; additional features; and non-singing voice applications.

5.1.1 Improving Audio Quality

On a system like the one addressed in this dissertation, errors are accumulated through the system, which means that all modules must perform very well. As such, every module should receive additional research attention, especially the phonetic related ones, which according to the listening tests, present more issues.

Regarding audio quality improvement, it is fundamental to decrease the artifacts created by the system, for allowing its use on professional applications. As previously mentioned, most of these artifacts appear due to abrupt transitions between audio fragments. One possible way to decrease these abrupt transitions may involve using PSOLA concepts. Instead of using fixed length frames, the system could use segments with a two period length, known to concatenate well in pitch-shift applications. To achieve that, the architecture of the system would require several changes in almost all modules, from feature extraction modules to synthesis.

To prevent artifacts with origin on the dynamics module, due to voiced/unvoiced mismatch, a better solution must be found to prevent the use of unvoiced frames to replace voiced ones, and vice-versa. Eventually, a set of rules might be enough.

Target and concatenation costs continue to require some additional attention, since they are responsible for the way frames are selected.

[38] either in existing research work or by creating new approaches

[39] Full list available at Annex A

5.1.2 Performance and Real-time Support

The actual system still requires a significant amount of computational power and memory. Although speed and memory requirements were not a concern during the PhD work, they should be taken into consideration in future work, especially to support real-time processing.

In reality, real-time support is not only a mater of speed processing. The current system considers an offline unit selection: searching for the full sequence of frames with target and concatenation costs. The future real-time support will require a different unit selection approach (eventually considering concatenation cost only between the current frame and the previous one). By loosing the overall sequence knowledge, it will be more difficult to choose the units for each situation, which might raise the number of existing artifacts.

Other performance bottleneck occurs on the interpolation and phase alignment during synthesis, where a more scalable approach should be found.

5.1.3 Additional features

Some additional features could bring significant improvements to the system.

Changing genre was not exhaustively tested during this work. Situations where the user is of a different genre from the internal sound library did not receive much attention during tests. Eventually, it may require some minor changes on the way the target cost is obtained (for instance, reducing the weight of LPC frequency response, or applying a frequency shift before using their values). A deeper attention is required to support such scenarios.

By adding editing capabilities, the user would be able to correct singing errors (e.g. out-of-tune problems) or achieve a more creative performance by adding some vocal effects (e.g. stable vibrato).

5.1.4 Non-singing voice applications

Although the research work was done on the singing voice, the same system could be used on non-singing material, either musical instruments or speech.

In speech, the system could be used to change the voice of the user on speaking scenarios.

On the musical area, the system could be used to replace (monophonic) music audio material with *samples* of other music instruments (as mentioned on the “applications” section of the first chapter). For instance, replacing one instrument recording with *samples* from some sound library.

With the present system, it is only a matter of using a non-singing sound library (speech dataset or a musical instruments sample library) and testing it with different audio files.

Probably some changes could be required regarding target costs, which may reveal to be more suitable for either speech or for music instrument resynthesis.

5.1.5 Future work summary

From all the ideas for future work, the PSOLA architecture will definitely be the first one to be implemented in a near future. Although it requires changes in all modules of the system, it may eventually present a solution for several issues: less artifacts due to better transitions, more scalable synthesizer (no interpolation and offset phase alignment), better support for a future real-time feature, and no time shifting issues.

List of References

[Aguirre, 1999] Aguirre, H.E., Tanaka, K., Sugimura, T.. Accelerated Halftoning Technique using Improved Genetic Algorithm with Tiny Populations. In: Proc. 1999 Conference on Systems, Man, and Cybernetics, Vol. 4, pp.905-910, 1999.

[Aguirre, 2000] Aguirre, H., Tanaka, K., Sugimura, T.. Accelerated Image Halftoning Technique using Improved Genetic Algorithm. In: IEICE Trans. Fundamentals, vol.E83-A, no.8, pp.1566-1574, Aug. 2000.

[Aisen, 2006] Aisen, B.. A Comparison of Multiclass SVM Methods. MIT final projects, 2006.

[Alender, 1995] Alender, J.T.. An indexed bibliography of genetic algorithms in signal and image processing. University of Vaasa, Department of Information Technology and Production Economincs, report 94-1-SIGNAL, 1995.

[Bay, 2009] Bay, M., Ehmann, A.F., Downie, J.S.. Evaluation of Multiple-F0 Estimation and Tracking Systems. In Proc. 10th International Society for Music Information Retrieval Conference (ISMIR09); Japan, 2009. (ismir2009.ismir.net/proceedings/PS2-21.pdf)

[Becchetti, 1999] Becchetti, C., Ricotti, L. P.. Speech recognition: theory and C++ implementation. John Wiley & Sons, 1999.

[Bello, 2003] Bello, J.P.. Towards the automated analysis of simple polyphonic music: A knowledge-based approach. PhD thesis, University of London, London, UK (2003)

[Berenguer, 2005] Berenguer, L.O., Quiros, F.C., Guijarro, S.T.. Multiple piano note identification using a spectral matching method with derived patterns. In Journal of the Audio Engineering Society, vol. 53, no. 1/2, pp. 32-43, January/Frebuary 2005.

[Bin, 2009] Bin, Z., et all. Frequency Calibration for on Chip Oscillator of AVR Using Genetic Algorithms. Research and Exploration, Vol 28 09, 2009. (http://d.wanfangdata.com.cn/Periodical_sysyjyts200909010.aspx)

[Bonada, 2005] Bonada, J.. Voice Solo to Unison Choir Transformation. In Proc. 118th AES Convention; Barcelona, Spain, May 2005.

[Bonada, 2007] Bonada, J., Serra, X.. Synthesis of the Singing Voice by Performance Sampling and Spectral Models. IEEE Signal Processing Magazine, 24, pp. 67-79, 2007.

[Bonada, 2008] Bonada, J.. Voice Processing and Synthesis by Performance Sampling and Spectral Models. PhD Thesis. Universitat Pompeu Fabra, Spain.

[Boser, 1992] Boser, B.E., Guyon, I.M., Vapnik, V.N.. A training algorithm for optimal margin classifiers. In D.Haussler, editor, Proceedings of the 5th Annual ACM workshop on Computational Learning Theory, pages 144-152, ACM Press, 1992.

[Cambridge, 2000] University of Cambridge. HTK. http://htk.eng.cam.ac.uk/

[Cardle, 2003] M. Cardle, S. Brooks, and P. Robinson. Audio and user directed sound synthesis. In Proc. ICMC, Singapore, 2003.

[Carreras, 1999] Carreras, F., Leman, M., Lesaffre, M.. Automatic harmonic description of musical signals using schema based chord decomposition. Journal of New Music Research, vol. 28, pp. 310-333(24), December 1999.

[Chang, 2006] Chang, W., Siao, Y., Su, A.. Analysis and Transynthesis of Solo Erhu Recordings using Addtivie/Subtractive Synthesis. In Proc. 120th AES Convention; Paris, France, May 2006.

[Cheveigné, 2002] Cheveigné, A., Kawahara, H.. YIN, a fundamental frequency estimator for speech and music. The Journal of the Acoustical Society of America, Vol. 111, No. 4. (2002), pp. 1917-1930.

[Cinesamples, 2010] Cinesamples. Voxos: Epic Virtual Choir. (Web page) http://cinesamples.com/products/voxos/ (as seen in January 2011).

[Collet, 2000] Collet, P., Lutton, E., Raynal, F., Schoenauer, M.. Polar {IFS}+Parisian Genetic Programming=Efficient {IFS} Inverse Problem Solving. In: Genetic Programming and Evolvable Machines, vol. 1, number 4, pp. 339 - 361, 2000.

[Cook, 1990] Cook, P.R.. Identification of Control Parameters in an Articulatory Vocal Tract Model, with Applications to the Synthesis of Singing. PhD thesis, Stanford University, 1990.

[Cook, 1992] Cook, P.. SPASM: a Real-Time Vocal Tract Physical Model Editor/Controller and Singer: the Companion Software Synthesis System. Computer Music Journal, 17: 1, pp 30-44, 1992.

[Cristianini, 2000] Cristianini, N., Shawe-Taylor, J.. An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press. 2000.

[Daniel, 2008] Daniel A., Emiya V., David B.. Perceptually-based evaluation of the errors usually made when automatically transcribing music. In Proc. of the Ninth International Conference on Music Information Retrieval (ISMIR).

[Degottex, 2010] Degottex, G.. Glottal source and vocal-tract separation – PhD Thesis. UPMC-Ircam, Paris, France, 2010.

[Devillard, 1998] Devillard, N.. Fast median search: an ANSI C implementation. July 1998, http://ndevilla.free.fr/median/median/index.html, (accessed Dec. 2010).

[Dixon, 2000] Dixon, S.. On the Computer Recognition of Solo Piano Music. Australasian Computer Music Conference, Brisbane, Australia, 31-37.

[Duan, 2010] Duan, Z., Pardo, B., Zhang, C.. Multiple Fundamental Frequency Estimation by Modeling Spectral Peaks and Non-peak Regions. IEEE Transaction on Audio, Speech and Language Processing (T-ASLP); Volume PP Issue:99, February 2010.

[Dunn, 2006] Dunn, E., Olague, G., Lutton, E.. Parisian camera placement for vision metrology. In: Pattern Recogn. Lett., vol. 27, number 11, pp. 1209 - 1219, Elsevier Science Inc., 2006.

[EASTWEST, 2001] EASTWEST. Quantum Leap Voices of the Apocalypse.

[EASTWEST, 2005] EASTWEST. EASTWEST/Quantum Leap Symphonic Choirs. http://www.soundsonline.com/Symphonic-Choirs-Virtual-Instrument-PLAY-Edition-pr-EW-182.html (Last accessed on Dec.09).

[EASTWEST, 2007] EASTWEST. EASTWEST/Quantum Leap Voices of Passion. http://www.soundsonline.com/Quantum-Leap-Voices-Of-Passion-Virtual-Instrument-pr-EW-174.html (Last accessed on Dec.09).

[EASTWEST, 2010] EASTWEST. About us (web page). http://www.soundsonline.com/About-Us-p-14.html (Last accessed on Dec.10).

[Emiya, 2008] Emiya V.. Automatic transcription of piano music. PhD Thesis, TELECOM ParisTech, Paris, France.

[Fonseca, 2002] Fonseca, N.. Voices of the Apocalypse Utility 1.0.

[Fonseca, 2003a] Fonseca, N.. VOTA Utility: Making the computer sing. In Proc. 114th AES Convention; Amsterdam, Netherlands, March 2003.

[Fonseca, 2003b] Fonseca, N.. Voices of the Apocalypse Utility 2.0.

[Fonseca, 2007a] Fonseca, N.. Introdução à Engenharia de Som (introduction to sound engineering). FCA, 1st – 5th edition. January 2007.

[Fonseca, 2007b] Fonseca, N.. Grapheme-to-phoneme conversion using recurrent neural networks. In Proc. CoMIC'07 - 2nd Conference on Methodologies for ScientifIC Research; Porto, Portugal, February 2007.

[Fonseca, 2008] Fonseca, N., Rocha, A.P.. Fragmentation and Frontier Evolution for Genetic Algorithms Optimization in Music Transcription. In Proc. Iberamia 2008 - 11th Ibero-American Conference on AI, Springer LNAI 5290; Lisbon, Portugal, October 2008.

[Fonseca, 2009] Fonseca, N., Ferreira, A.. Measuring Music Transcription Results Based on a Hybrid Decay/Sustain Evaluation. In Proc. ESCOM 2009 - 7th Triennial Conference of European Society for the Cognitive Sciences of Music; Finland, 2009.

[Fonseca, 2010a] Fonseca, N., Ferreira, A.. Singing Voice Resynthesis Using Vocal Sound Libraries. In Proc. 13th International Conference on Digital Audio Effects (DAFx-10); September 2010; Graz, Austria.

[Fonseca, 2010b] Fonseca, N., Ferreira, A.. A Singing Voice Resynthesis Approach. In Proc. 12th meeting of the Portuguese Section of Audio Engineering Society; October 2010; Aveiro, Portugal.

[Fonseca, 2011a] Fonseca, N., Ferreira, A., Rocha, A.P.. Concatenative Singing Voice Resynthesis. In Proc. 17th International Conference on Digital Signal Processing (DSP2011), July 2011, Corfu, Greece.

[Fonseca, 2011b] Fonseca, N., Ferreira, A., Rocha, A.P.. Re-síntese Concatenativa de Voz Cantada. In Proc. 3º Simpósio de Informática (inForum 2011), Nov. 2011, Coimbra, Portugal.

[Garcia, 2001] Garcia, G.. A genetic search technique for polyphonic pitch detection. In Proceedings of the International Computer Music Conference (ICMC), Havana, Cuba, September 2001.

[Garofolo, 1993] Garofolo, J.S., et al.. TIMIT Acoustic-Phonetic Continuous Speech Corpus, Linguistic Data Consortium, Philadelphia.

[Gerhard, 2003] Gerhard, D.B.. PhD Thesis - Computationally Measurable Temporal Differences Between Speech And Song. SIMON FRASER UNIVERSITY; 2003

[Gilreath, 2004] Gilreath, P.. The Guide To MIDI Orchestration. Musicworks Atlanta; 3 edition, 2004.

[Goldberg, 1989] Goldberg, D.E.. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional, January 1989.

[Goldberg, 1991] Goldberg, D. E., Deb, K.. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed., pp. 69–93, Morgan Kaufmann, San Mateo, Calif, USA, 1991.

[Gomez, 2003] Gomez, E., Klaupuri, A., Meudic, B.. Melody description and extraction in the context of music content processing. Journal of New Music Research 32(1) 2003.

[Gomez, 2006] Gomez, E., Streich, S., Ong, B., Paiva, R.P., Tappert, S., Batke, J.M., Poliner, G., Ellis, D., Bello, J.P.. A quantitive comparison of different approaches for melody extraction from polyphonic audio recordings. MTG-2006, April 2006.

[Goto, 2001] Goto, M.. A robust predominant-f0 estimation method for real-time detection of melody and bass lines in cd recordings. 2001.

[Haas, 1972] Haas, H.. The Influence of a Single Echo on the Audibility of Speech, JAES Volume 20 Issue 2 pp. 146-159.

[Harris, 1978] Harris, F. J.. On the use of windows for harmonic analysis with the discrete fourier transform. Proceedings of the IEEE, vol. 66, no. 1, pp. 51-83, 1978.

[Hart, 2004] Hart, W., Krasnogor, N., Smith, J.. Memetic Evolutionary Algorithms. In: Recent Advances in Memetic Algorithms. Springer (2004).

[Haykin, 1994] Haykin, S.. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA.

[Haykin, 1999] Haykin, S.. *Neural Networks: A Comprehensive Foundation.* 2nd edition. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[Holland, 1992] Holland, J.H.. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The MIT Press, April 1992.

[Holmes, 1983] Holmes, J.N.. Formant Synthesizers: Cascade or Parallel. In Speech Communications, Volume 2, pp. 251-273.

[Hoskinson, 2001] R. Hoskinson and D. Pai. Manipulation and Resynthesis with natural grains. In Proc. ICMC, Havana, 2001

[Hunt, 1996] Hunt, A.J., Black, A.W.. Unit selection in a concatenative speech synthesis system using a large speech database. ICASSP96, volume 1, Issue , 7-10 May 1996, pp. 373 - 376.

[Itakura, 1970] Itakura, F., Saito, S.. A statistical method for estimation of speech spectral density and formant frequencies. Electronics & Communications in Japan, 53A: 36-43, 1970.

[Janer, 2004] Janer, J. . Voice as a musical controller for real-time synthesis. Master Thesis, Universitat Pompeu Fabra, Spain.

[Janer, 2006] Janer, J., Bonada, J., Blaauw, M.. Performance-Driven Control For Sample-Based Singing Voice Synthesis. In Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06), Montreal, Canada, September 18-20, 2006

[Janer, 2008] Janer, J., de Boer, M.. Extending voice-driven synthesis to audio mosaicking. In Proc. 5th Sound and Music Computing Conference, 2008.

[Joachims, 2002] Joachims, T.. SVM light. http://svmlight.joachims.org/

[Kashino, 1995] Kashino, K., Nakadai, K., Kinoshita, T., Tanaka, H. . Organization of hierarchical perceptual sounds: Music scene analysis with autonomous processing modules and a quantitative information integration mechanism. In IJCAI, pages 158-164, 1995.

[Kennedy, 1995] Kennedy, J., Eberhart, R.. Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks. Vol. IV. pp. 1942–1948.

[Kent, 2004] Kent, R.D. (edited). The MIT Encyclopedia of Communication Disorders. MIT Press; 2004.

[Keyboard, 2003] Keyboard Magazine. Keyboard November 2003.

[Klapuri, 2000] Klapuri, A.P.. Qualitative and quantitative aspects in the design of periodicity estimation algorithms. In Proceedings of the European Signal Processing Conference (2000)

[Klapuri, 2004] Klapuri, A.P.. Automatic music transcription as we know it today. Journal of New Music Research, vol. 33, no. 3, pp. 269-282, 2004.

[Klapuri, 2008] Klapuri, A.. Multiplitch analysis of polyphonic music and speech signals using an auditory model. In: IEEE Transactions on Audio and Language Processing, 16(2):255-264, February 2008.

[Klingbeil, 2005] Klingbeil, M.. Software For Spectral Analysis, Editing, And Synthesis. In Proceding of ICMC 2005.

[Kohonem, 1984] Kohonen, T. *Self-Organization and Associative Memory. (3rd edition 1989)*. Springer, Berlin.

[Kohonem, 1988] Kohonen, T.. The 'Neural' Phonetic Typewriter. *Computer* 21, 3 (March 1988), 11-22, IEEE Computer Society Press.

[Laroche, 1998] Laroche, J.. “Time and pitch scale modification of audio signals,” in Applications of Digital Signal Processing to Audio and Acoustics, M. Kahrs and K. Brandenburg, Eds. Kluwer, Norwell, MA, 1998.

[Lu, 2006] Lu, D.. Automatic Music Transcription Using Genetic Algorithms and Electronic Synthesis. In: Computer Science Undergraduate Research, University of Rochester, USA, http://www.csug.rochester.edu/ug.research/dlupaper.pdf (Last accessed on Dec.09)., 2006.

[Macon, 1997] Macon, M.W., Jensen-Link, L., Oliverio, L., Clements, M., George, E. B.. A system for singing voice synthesis based on sinusoidal modeling,. Proc. of International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, pp. 435-438, 1997.

[Macon, 2000] Macon, M.. Flinger System Documentation (Web Page). http://cslu.cse.ogi.edu/tts/flinger/fldoc.html (as in December, 2010).

[Manning, 2008] Manning, C.D., Raghavan, P., Schütze, H.. An Introduction to Information Retrieval, Cambridge University Press, http://wwwcsli.stanford.edu/~hinrich/information-retrievalbook.html (Last accessed on Dec. 2010).

[Marolt, 2004a] Marolt, M.. Networks of adaptive oscillators for partial tracking and transcription of music recordings. Journal of New Music Research, vol. 33, pp. 49-59(11), March 01, 2004.

[Marolt, 2004b] Marolt, M.. On finding melodic lines in audio recordings. In Proc. of the 7th Int. Conference on Digital Audio Effects (DAFX-04), Naples, Italy, October 2004.

[Martin, 1996] Martin, K.D.. A blackboard system for automatic transcription of simple polyphonic music. Tech. Rep. 385, MIT Media Lab, Perceptual Computing Section, Tech. Rep., July 1996.

[Mayor, 2009] Mayor, O., Bonada, J., Janer, J.. KaleiVoiceCope: Voice Transformation from Interactive Installations to Video-Games. In Proc. AES 35th International Conference: Audio for Games, 2009.

[Mirex, 2007] MIREX. MIREX 2007 - Music Information Retrieval Evaluation eXchange, Retrived from http://www.musicir.org/mirex/2007/index.php/Main_Page (Last accessed on Dec. 2010), 2007.

[Mitchell, 1997] Michell, T.. Machine Learning. McGraw Hill.

[MMA, 1995] MIDI Manufacturers Association. The Complete MIDI 1.0 Detailed Specification. September 1995.

[Moorer, 1977] Moorer, J.A.. On the transcription of musical sound by computer. In Computer Music journal, 1(4):32-38, 1977.

[Nakano, 2009] Nakano, T., Goto, M.. VocaListener: a singing-to-singing synthesis system based on iterative parameter estimation. In Proc. SMC 2009, Porto, Portugal, July 2009.

[Nemesys, 1998] NemeSys Music Technology, Inc. Gigapiano.

[Noll, 1967] Noll, A. M.. Cepstrum pitch determination. Journal Acoustic Society Am. 41, 293–309.

[Noll, 1969] Noll, M.. Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate. In Proc. of the Symposium on Computer Processing Communications, pp. 779–797.

[NorthernSounds, 2002] NorthernSounds. Forum Post “Application to work with Voices of the Apoc. (VOTA)”. 2002. http://northernsounds.com/forum/showthread.php?p=27488 (as in April 2010).

[NorthernSounds, 2003] NorthernSounds. Forum post “REAL or SAMPLED CONTEST”. http://www.northernsounds.com/forum/showthread.php/9717-REAL-or-SAMPLED-CONTEST, (Last accessed on September 2010).

[Ouyang, 2009] Ouyang, B.H., Zhao, C.Y., Bihuan, O., Chunyu, P.. Current methods and development trend of harmonic detection of power network. Electronic Measurement Technology Journal; July 2009.

[Pertusa, 2010] Pertusa, A.. PhD Thesis “Computationally efficient methods for polyphonic music transcription”. Universidad de Alicante; 2010.

[Piszczalski, 1977] Piszczalski, M., Galler, B.A.. Automatic music transcription. Computer Music Journal, vol. 1, no. 4, pp. 24-31, 1977.

[Poliner, 2007] Poliner, G.E. and Ellis, D.P.W.. A Discriminative Model for Polyphonic Piano Transcription, EURASIP Journal on Advances in Signal Processing, Volume 2007, Article ID 48317.

[Rabiner, 1989] Rabiner, L.R.. A tutorial on hidden markov models and selected applications in speech recognition. In Proc. of the IEEE, 1989, pages 257-286.

[Rajeswari, 2010] Rajeswari, S., Karthiga, S., Geetha, T.V.. Fundamental Frequency Estimation of Carnatic Music Songs Based on the Principle of Mutation. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 7, July 2010.

[Raphael, 2002] Raphael, C.. Automatic transcription of piano music. In Proceedings of ISMIR 2002, 2002.

[Reis, 2007a] Reis, G., Fernandez, F.. Electronic synthesis using genetic algorithms for automatic music transcription. In GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation. New York, NY, USA: ACM Press, 2007, pp. 1959-1966.

[Reis, 2007b] Reis, G., Fonseca, N., Fernandez, F.. Genetic Algorithm Approach to Polyphonic Music Transcription, IEEE International Symposium on Intelligent Signal Processing (WISP 2007), Spain.

[Reis, 2008a] Reis, G., Fonseca, N., Vega, F., Ferreira, A.. Hybrid Genetic Algorithm based on Gene Fragment Competition for Polyphonic Music Transcription. In Proc. EvoIASP 2008, LNCS vol. 4974/2008, pp. 305-314.

[Reis, 2008b] Reis, G., Fonseca, N., Vega, F., Ferreira, A.. A Genetic Algorithm Approach with Harmonic Structure Evolution for Polyphonic Music Transcription. In Proc. ISSPIT 2008 - IEEE Symposium on Signal Processing and Information Technology; Sarajevo, Bosnia & Herzegovina, December 2008.

[Robel, 2010] Robel, A. Between Physics and Perception: Signal Models for High Level Audio Processing. In Proc. DAFx 2010, Graz, Austria, 2010.

[Rodet, 1984] Rodet, X., Potard, Y., Barriere, J.B.. The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General. Computer Music Journal, Vol. 8, No. 3 (Autumn, 1984), pp. 15-31, The MIT Press.

[Rosenblatt, 1958] Rosenblatt, F.. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.

[Russel, 2003] Russell, S. J., Norvig, P.. Artificial Intelligence: A Modern Approach. Prentice Hall, pp. 111–114, ISBN 0-13-790395-2.

[Ryynanen, 2005] Ryynanen, M.P. and Klapuri, A.. Polyphonic Music Transcription Using Note Event Modeling, 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, NY.

[Schwarz, 2004] Schwarz, D.. PhD Thesis - Data-driven concatenative sound synthesis. Université Paris 6; 2004.

[Schwarz, 2005] Schwarz, D.. Current Research In Concatenative Sound Synthesis. In Proc. International Computer Music Conference (ICMC 2005), Barcelona, Spain, September 5-9, 2005.

[Schwarz, 2006] Schwarz, D., Beller, G., Verbrugghe, B., Britton, S.. Real-Time Corpus-Based Concatenative Synthesiswith Catart. In Procedings DAFx 2006.

[Serra, 1990] Serra, X., Smith, J.. Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition. *Computer Music Journal* Vol. 14, No. 4 (Winter, 1990), pp. 12-24.

[Shadle, 2002] Shadle, C. H., Damper, R. I.. Prospects for articulatory synthesis: A position paper. In Proc. *4th ISCA Workshop on Speech Synthesis, August/September 2001*, Pitlochry, Scotland. pp. 121-126.

[SOS, 2002] Sound on Sound. Sound on Sound Magazine - October 2002, http://www.soundonsound.com/sos/Oct02/articles/sample_apocalypse.htm (Last accessed on April 2010).

[SOS, 2005] Sound on Sound. Sound on Sound Magazine - November 2005, http://www.soundonsound.com/sos/nov05/articles/ewsymphonic.htm (Last accessed on April 2010).

[Sun, 2002] Sun, X.. Pitch determination and voice quality analysis using subharmonic-to-harmonic ratio. Proc. of ICASSP 2002, Orlando, Florida, May 13 -17, 2002.

[Sundberg, 1974] Sundberg J.. Articulatory interpretation of the singing formant. J. Acoust. Soc. Amer. 55, 838-844; 1974.

[Sundberg, 1987] Sundberg J.. The science of the singing voice. Northern Illinois University Press; 1987.

[Tavares, 2008] Tavares T.F., Barbedo J.G.A., Lopes A.. Towards the evaluation of automatic transcription of music. In Proc. VI Congresso de Engenharia de Áudio 2008. p. 96-99.

[TCNAS, 2009] Bouillot, N., Cohen, E., Cooperstock, J. R., Floros, A., Fonseca, N., Foss, R., Goodman, M., Grant, J., Gross, K., Harris, S., Harshbarger, B., Heyraud, J., Jonsson, L., Narus, J., Page, M., Snook, T., Tanaka, A., Trieger, J., Zanghieri, U.. AES White Paper: Best Practices in Network Audio. In Journal of Audio Engineering Society, Volume 57 Issue 9 pp.; September 2009.

[Triki, 2008] Triki, M., Slock, D.T.M., Triki, A.. Periodic signal extraction with frequency-selective amplitude modulation and global time-warping for music signal decomposition. In Proc. MMSP 2008, 10th IEEE International Workshop on MultiMedia Signal Processing; Cairns, Queensland, Australia; October 2008, pp 972-977.

[Triki, 2009] Triki, M., Slock, D.T.M.. Perceptually motivated quasi-periodic signal selection for polyphonic music transcription. In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009); Tapei, Taiwan; April 2009.

[Vapnik, 1998] Vapnik, V.. Statistical Learning Theory. Wiley, 1998.

[Verfaille, 2004] Verfaille, V., Depalle, P.. Adaptative Effects based on STFT, using a Source-Filter Model. In Proc. DAFx 2004. Naples, Italy, 2004.

[VirSyn, 2004] VirSyn Cantor, http://www.virsyn.de/en/E_Products/E_CANTOR/ e_cantor.html (Last accessed on September 2010).

[VSL, 2010] Vienna Symphonic Library. Vienna Choir. (Web page) http://forum.vsl.co.at/forums/p/24818/166757.aspx (as in January 2011).

[Walmsley, 1999a] Walmsley, P., Godsill, S., Rayner, P.. Bayesian graphical models for polyphonic pitch tracking. In Diderot Forum, 1999.

[Walmsley, 1999b] Walmsley, P., Godsill, S., Rayner, P.. Polyphonic pitch tracking using joint bayesian estimation of multiple frame parameters. In Proc. IEEEWorkshop on Audio and Acoustics, Mohonk, New York, 1999.

[Wang, 2006] Wang, D., Brown, G.J.. Computational Auditory Scene Analysis, Principles, Algorithms and Applications: IEEE Press.

[Yamaha, 2004] Yamaha. Vocaloid. http://www.vocaloid.com/index.en.html (Last accessed on December 2009)

[Zils, 2001] Zils, A., Pachet, F.. Musical Mosaicing. COST-G6 Workshop on Digital Audio Effects, (DAFx-01), Limerick, 2001.

[Zolzer, 2002] Zolzer, U. (editor). DAFX - Digital Audio Effects. John Wiley & Sons, Ltd, 2002.

[Zolzer, 2011] Zolzer, U. (editor). DAFX - Digital Audio Effects. 2nd edition, John Wiley & Sons, Ltd, 2011.

[Zwicker, 1999] Zwicker, E., Fastl, H.. Psychoacoustics: facts and models, Springer series in information sciences, 22. Springer, Berlin; New York, 2nd updated edition.

Annex A – Publications and citations

This annex presents the publications and citations regarding this PhD work. The full articles (including the journal submission and the electronic version of this dissertation) can be accessed at "http://www.estg.ipleiria.pt/~nuno.fonseca/PhD".

The following nine papers were published:

- "Genetic Algorithm Approach to Polyphonic Music Transcription"; Gustavo Reis, Nuno Fonseca, Francisco Fernández de Vega; In Proc. WISP 2007 - IEEE International Symposium on Intelligent Signal Processing; Alcala de Henares, Spain, October 2007.
- "Hybrid Genetic Algorithm Based on Gene Fragment Competition for Polyphonic Music Transcription"; Gustavo Reis, Nuno Fonseca, Francisco Fernández de Vega, Aníbal Ferreira; In Proc. EvoIasp 2008 - 10th Workshop on Evolutionary Computation in Image Analysis and Signal Processing; Springer LNCS 4974; Naples, Italy, March 2008.
- "Fragmentation and Frontier Evolution for Genetic Algorithms Optimization in Music Transcription"; Nuno Fonseca, Ana Paula Rocha; In Proc. Iberamia 2008 - 11th Ibero-American Conference on AI, Springer LNAI 5290; Lisbon, Portugal, October 2008.
- "A Genetic Algorithm Approach with Harmonic Structure Evolution for Polyphonic Music Transcription"; Gustavo Reis, Nuno Fonseca, Francisco Fernández de Vega, Anibal Ferreira; In Proc. ISSPIT 2008 - IEEE Symposium on Signal Processing and Information Technology; Sarajevo, Bosnia & Herzegovina, December 2008.
- "Measuring Music Transcription Results Based on a Hybrid Decay/Sustain Evaluation"; Nuno Fonseca, Aníbal Ferreira; In Proc. ESCOM 2009 - 7th Triennial Conference of European Society for the Cognitive Sciences of Music; Finland, 2009.
- "Singing Voice Resynthesis Using Vocal Sound Libraries"; Nuno Fonseca, Aníbal Ferreira; In Proc. 13th International Conference on Digital Audio Effects (DAFx-10); September 2010; Graz, Austria.
- "A Singing Voice Resynthesis Approach"; Nuno Fonseca, Aníbal Ferreira; In Proc. 12th meeting of the Portuguese Section of Audio Engineering Society; October 2010; Aveiro, Portugal.
- "Concatenative Singing Voice Resynthesis"; Nuno Fonseca, Aníbal Ferreira, Ana Paula Rocha; In Proc. DSP 2011 - 17th International Conference on Digital Signal Processing; July 2011; Corfu, Greece.

- "Re-síntese concatenativa de voz cantada", Nuno Fonseca, Anibal Ferreira, Ana Paula Rocha, In Proc. inForum 2011; September, 2011, Coimbra, Portugal.

The following journal manuscript was submitted, and is waiting for review:

- "Singing Voice Resynthesis based on Feature Extraction, Unit Selection and Concatenative Approaches", Computer Music Journal, MIT Press.

During the duration of this PhD, the following items were also published, although not directly related with the PhD work:

- Book "Introdução à Engenharia de Som" (translation: Introduction to Sound Engineering); Nuno Fonseca; FCA; 1st - 5th edition; 2007.
- White Paper "AES White Paper: Best Practices in Network Audio"; Bouillot, N., Cohen, E., Cooperstock, J. R., Floros, A., Fonseca, N., Foss, R., Goodman, M., Grant, J., Gross, K., Harris, S., Harshbarger, B., Heyraud, J., Jonsson, L., Narus, J., Page, M., Snook, T., Tanaka, A., Trieger, J., Zanghieri, U.; In Journal of Audio Engineering Society, Volume 57 Issue 9 pp.; September 2009.
- Paper "Grapheme-to-phoneme conversion using recurrent neural networks"; Nuno Fonseca; CoMIC'07 - 2nd Conference on Methodologies for ScientifIC Research; Porto, Portugal, February 2007.

The PhD-related publications were referenced by other authors, including:

- Book "DAFX – Digital Audio Effects"; Edited by Udo Zolzer; 2nd edition; John Wiley & Sons; 2011.
- "Fundamental Frequency Estimation of Carnatic Music Songs Based on the Principle of Mutation"; Rajeswari Sridhar , Karthiga S and Geetha T V; IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 7, July 2010. [Rajeswari, 2010].
- "Multiple Fundamental Frequency Estimation by Modeling Spectral Peaks and Non-peak Regions"; Z. Duan, B. Pardo and C. Zhang; IEEE Transaction on Audio, Speech and Language Processing (T-ASLP); Volume PP Issue:99, February 2010. [Duan, 2010]
- "PhD Thesis - Computationally efficient methods for polyphonic music transcription"; Pertusa, A.; Universidad de Alicante; 2010. [Pertusa, 2010]
- "Current methods and development trend of harmonic detection of power network"; Bi Huan Ouyang, Chun-Yu Zhao, Ouyang Bihuan, Photo Chunyu; Electronic Measurement Technology Journal; July 2009. [Ouyang, 2009]
- "Perceptually motivated quasi-periodic signal selection for polyphonic music transcription"; Triki, M., Slock, D.T.M.; In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009); Tapei, Taiwan; April 2009. [Triki, 2009]

- “Evaluation of Multiple-F0 Estimation and Tracking Systems”; Mert Bay, Andreas F. Ehmann and J. Stephen Downie; In Proc. 10th International Society for Music Information Retrieval Conference (ISMIR09); Japan, 2009. [Bay, 2009]
- “Frequency Calibration for on Chip Oscillator of AVR Using Genetic Algorithms”; Zhou Bin et all; Research and Exploration, Vol 28 09, 2009. [Bin, 2009]
- “Periodic signal extraction with frequency-selective amplitude modulation and global time-warping for music signal decomposition”; Triki, Mahdi; Slock, Dirk T M;Triki, Ahmed; MMSP 2008, 10th IEEE International Workshop on MultiMedia Signal Processing; Cairns, Queensland, Australia; October 2008, pp 972-977. [Triki, 2008]
- “Melody and bass line estimation method using audio feature database”; Uchida, Yasunori; Wada, Shigeo; IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), 2011.
- “Influence de la decision voise/non-voise dans l'evaluation comparative d'algorithmes d'estimation de F0”; Francois Signol, Jean-Sylvain Lienard, Claude Barras; XXVIII Journées d'Étude sur la Parole (JEP 2010).
- “PhD Thesis - Estimation de fréquences fondamentales multiples en vue de la séparation de signaux de parole mélangés dans un même canal”; Francois Signol; 2009