

Independent Formant and Pitch Control Applied to Singing Voice

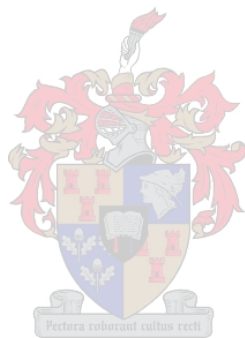
WIETSCHÉ R. CALITZ



*Thesis presented in partial fulfilment of the requirements for the degree
Master of Science in Electronic Engineering
at the University of Stellenbosch*

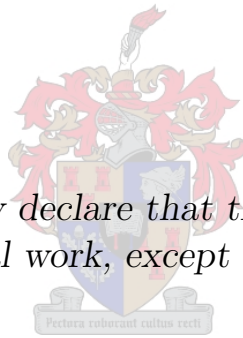
SUPERVISOR: Prof J.A. du Preez

December 2004



Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, except where stated otherwise.



SIGNATURE

DATE

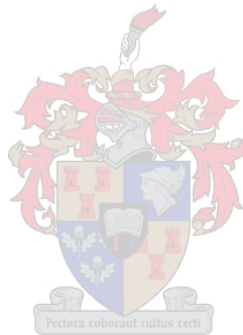
Abstract

A singing voice can be manipulated artificially by means of a digital computer for the purposes of creating new melodies or to correct existing ones. When the fundamental frequency of an audio signal that represents a human voice is changed by simple algorithms, the formants of the voice tend to move to new frequency locations, making it sound unnatural. The main purpose is to design a technique by which the pitch and formants of a singing voice can be controlled independently.



Opsomming

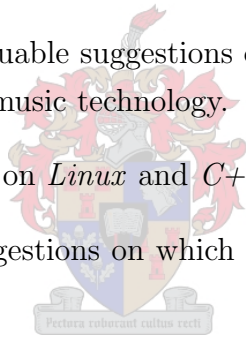
Onafhanklike formant- en toonhoogte beheer toegepas op 'n sangstem: 'n Sangstem kan deur 'n digitale rekenaar gemanipuleer word om nuwe melodieë te skep, of om bestaandes te verbeter. Wanneer die fundamentele frekwensie van 'n klanksein (wat 'n menslike stem voorstel) deur 'n eenvoudige algoritme verander word, skuif die oorspronklike formante na nuwe frekwensie gebiede. Dit veroorsaak dat die resultaat onnatuurlik klink. Die hoof oogmerk is om 'n tegniek te ontwerp wat die toonhoogte en die formante van 'n sangstem apart kan beheer.



Acknowledgements

I would like to thank the following people:

- *Prof Johan du Preez* for his academic guidance and general support over the past years.
- *Jeanne Hugo and Frouwien du Toit* for lending me their singing talent. I used recordings of short performances by them for my development.
- *Christo Viljoen* for recording my audio samples and for his short performances.
- *Peter Matthaei* for his valuable suggestions on my development and for joining me in exploring the world of music technology.
- *Johan Cronje* for his help on *Linux* and *C++*.
- *Albert Visagie* for his suggestions on which academic papers to consult and where to find them.
- *Gert-Jan van Rooyen* for his highly appreciated \LaTeX help.
- *The National Research Fund* for two years of financial support and *Prof Ben Herbst* for the administration of the funding.
- *Wilken Calitz* for his help with MIDI and for all he taught me about music.
- My dear parents *Estian* and *Karin Calitz* for their love and support that I had known since I can remember.
- Him, to whom I owe everything.



Prologue

I, the author of this thesis, went on a very stimulating journey while writing this text. With strong coffee as a companion, a general background in speech processing and an interest in music, I started pursuing information on the less documented topic of singing processing - all in the hope of finding information to solve the problem as stated in the topic of this thesis. After some reading and discussion with colleagues I started getting a faint idea of the scope of the problem.

I turned to the biggest information source I know: the Internet. It offered vast amounts of information on different types of phase vocoders and their applications, as well as many papers on pitch-trackers. Many of my early ideas were formed by websites providing such information and one after the other I discovered undocumented papers of individuals claiming their phase vocoder or pitch-tracker to be better than the next.

Numerous code examples and papers are available on the time-scaling application of the phase vocoder but very little on its application for pitch-shifting, which was a problem that kept me busy for some time. Some papers that claim to cover this topic do not go into the detail of the implementation, therefore I hope that this thesis could be of use to someone who wishes to implement a phase vocoder-based pitch-shifting algorithm.

After spending some time on reading and studying code examples, I started implementing my own code and evaluating it by comparing the results to the commercial standards of a number of companies devoted to processing of the singing voice. They do not make their algorithms public, but their results are available on websites where sound-files can be downloaded. Listening to their results and comparing it to my own made me ecstatic at times and miserable at others. As one would guess, their algorithms are very well guarded secrets and I spent my time emulating what they had achieved without having an idea how they did it. It offered me an excellent opportunity to experiment with my theoretical knowledge of speech, music and electronic engineering.

The contents of this thesis reflect some of the things I learned from other persons and publications and some of my own ideas. Although not listed in my Bibliography, I give credit to the hobbyists and scientists who took the trouble to put their ideas on a website - for me to learn from.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Key concepts | 2 |
| 1.2.1 | Formants | 2 |
| 1.2.2 | Pitch | 2 |
| 1.2.3 | The phase vocoder | 2 |
| 1.2.4 | Non-linear smoothers | 3 |
| 1.3 | Objectives | 3 |
| 1.4 | Contributions | 3 |
| 1.5 | High level overview | 4 |
| 1.5.1 | Background | 4 |
| 1.5.2 | Main algorithms | 4 |
| 1.5.3 | Applications of main algorithms | 6 |
| 1.6 | Implementation and audio results | 7 |
| 2 | The Singing Voice | 8 |
| 2.1 | Introduction | 8 |
| 2.2 | Brief notes on the history of singing | 8 |
| 2.3 | Speech production | 9 |
| 2.4 | Singing: a special form of speech | 11 |
| 2.5 | Mathematical model | 15 |
| 2.6 | Summary | 18 |
| 3 | “LULU”: a Non-Linear Smoother | 19 |
| 3.1 | Introduction | 19 |
| 3.2 | Basic non-linear smoother concepts, notation and terminology | 20 |
| 3.2.1 | Notation and terminology | 20 |
| 3.2.2 | The median filter | 21 |
| 3.3 | Smoothers: \mathcal{L} and \mathcal{U} | 22 |
| 3.4 | Smoothers: \mathcal{UL} and \mathcal{LU} | 24 |
| 3.5 | Summary | 24 |

| | | |
|----------|--|-----------|
| 4 | The Phase Vocoder | 26 |
| 4.1 | Introduction | 26 |
| 4.2 | A unity phase vocoder | 27 |
| 4.2.1 | Analysis | 27 |
| 4.2.2 | Synthesis | 28 |
| 4.2.3 | Window choice | 29 |
| 4.3 | Time-scaling | 30 |
| 4.3.1 | A simple approach | 30 |
| 4.3.2 | Using the phase vocoder for time-scaling | 30 |
| 4.3.3 | Applications of time-scaling | 35 |
| 4.4 | Summary | 36 |
| 5 | Pitch Detection | 37 |
| 5.1 | Introduction | 37 |
| 5.2 | Harmonic extraction | 38 |
| 5.3 | Pitch calculation | 39 |
| 5.4 | Constructing the pitch-track | 43 |
| 5.5 | Post-processing | 43 |
| 5.6 | On silence | 43 |
| 5.7 | Summary | 44 |
| 6 | Pitch-shifting | 46 |
| 6.1 | Introduction: failure of an obvious approach | 46 |
| 6.2 | Source-filter decomposition | 48 |
| 6.2.1 | Splitting the signal | 48 |
| 6.2.2 | Linear Prediction-model | 49 |
| 6.2.3 | Direct-model | 52 |
| 6.2.4 | Comparing <i>LPC</i> to direct-modelling | 58 |
| 6.3 | Modifying the excitation | 59 |
| 6.4 | Breathing life into the excitation | 62 |
| 6.4.1 | More than pitch matters | 62 |
| 6.4.2 | Re-applying a <i>LPC</i> -model | 63 |
| 6.4.3 | Re-applying a direct-model | 64 |
| 6.5 | A note on phase | 66 |
| 6.6 | Summary | 66 |
| 7 | Artificial singing voices | 67 |
| 7.1 | Introduction | 67 |
| 7.2 | Analysis | 69 |

| | | |
|-----------|---|------------|
| 7.3 | Pitch-tracking | 69 |
| 7.4 | Pitch-shifting | 69 |
| 7.5 | β calculation | 70 |
| 7.5.1 | Scale notation | 70 |
| 7.5.2 | Harmony | 70 |
| 7.5.3 | Pitch correction | 71 |
| 7.5.4 | Constant pitch-shift | 71 |
| 7.6 | Unvoiced and silence detection | 72 |
| 7.7 | Synthesis | 73 |
| 7.8 | Windowing issues | 73 |
| 7.9 | Summary | 74 |
| 8 | Further Applications | 75 |
| 8.1 | Introduction | 75 |
| 8.2 | Wave synthesis | 75 |
| 8.3 | Gender transformer | 76 |
| 8.4 | Summary | 78 |
| 9 | Results | 79 |
| 9.1 | What to measure | 79 |
| 9.2 | Accuracy and resolution of the pitch-tracking algorithm | 80 |
| 9.3 | Accuracy of the pitch-shifting algorithm | 82 |
| 9.3.1 | Evaluation of the algorithm | 82 |
| 9.3.2 | Results from pitch-shifting applications | 85 |
| 9.4 | Synthesis | 86 |
| 9.5 | High level evaluation | 87 |
| 10 | Final comments | 89 |
| 10.1 | Conclusions | 89 |
| 10.2 | Future work | 90 |
| 10.3 | Last thoughts | 91 |
| A | Linear Prediction Analysis | 94 |
| B | Basic Perspective on Musical Scales | 97 |
| C | Pitch Cents | 99 |
| D | Autocorrelation function pitch detection methods | 100 |
| D.1 | Autocorrelation Basics | 100 |
| D.2 | Improving on Basic Autocorrelation Method | 101 |

D.3 Average magnitude difference function (AMDF) 102



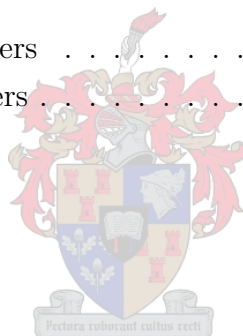
List of Figures

| | | |
|-----|---|----|
| 2.1 | Diagram of the vocal tract [3] | 10 |
| 2.2 | <i>LPC</i> spectrum of a vowel (/Λ/) | 11 |
| 2.3 | Spectra of a sung and spoken vowel | 12 |
| 2.4 | Trained and untrained singing of a vowel | 12 |
| 2.5 | Formant frequencies: speech and singing | 13 |
| 2.6 | Ranges of voice types | 15 |
| 2.7 | Excitation spectrum of a vowel | 16 |
| 3.1 | Median filter versus linear smoother | 21 |
| 3.2 | Steps of an \mathcal{L} -smoother | 22 |
| 3.3 | Steps of a \mathcal{U} -smoother | 23 |
| 3.4 | Spectrum and its smoothed version | 25 |
| 4.1 | Short-time Fourier transform illustration | 28 |
| 4.2 | A unity phase vocoder | 29 |
| 4.3 | Mapping during time-stretching | 31 |
| 4.4 | The importance of phase continuity | 31 |
| 4.5 | Time-scaling algorithm | 34 |
| 4.6 | Spectrogram of an original audio signal | 35 |
| 4.7 | Spectrogram of a time-scaled audio signal | 36 |
| 5.1 | Spectral peak spanned by a plateau | 38 |
| 5.2 | Filtering at two different orders | 40 |
| 5.3 | Pitch-track of a scale by a male vocalist | 43 |
| 5.4 | Silence detection | 45 |
| 5.5 | Entrance and exit of the silent state | 45 |
| 6.1 | Spectrogram of a male singing voice | 47 |
| 6.2 | Spectrogram of a signal evaluated at $\frac{4}{3}F_s$ | 47 |
| 6.3 | Diagram of the working of a pitch-shifter | 48 |
| 6.4 | A spectrum before (top) and after (bottom) whitening | 50 |
| 6.5 | Normalised error | 51 |

| | | |
|------|---|-----|
| 6.6 | Optimal <i>LPC</i> model | 52 |
| 6.7 | <i>LPC</i> over modelling | 52 |
| 6.8 | A single B-spline segment | 54 |
| 6.9 | Cubic spline demonstration | 54 |
| 6.10 | Steps of direct-modelling | 55 |
| 6.11 | Spectral whitening through direct-modelling | 56 |
| 6.12 | Formant shifting | 57 |
| 6.13 | <i>LPC</i> -modelling and direct-modelling | 59 |
| 6.14 | Spectral manipulation | 61 |
| 6.15 | Excitation spectrum, before and after pitch modifications | 62 |
| 6.16 | Enlarged extractions from Figure 6.15 | 63 |
| 6.17 | Formant restoration using <i>LPC</i> | 64 |
| 6.18 | Formant restoration using direct-modelling | 65 |
| 7.1 | High level diagram of the non-causal pitch-shifting system | 68 |
| 7.2 | A pitch-track an a scale corrected version | 71 |
| 8.1 | Wave synthesis | 77 |
| 9.1 | Spectrogram of a synthetic signal | 80 |
| 9.2 | Pitch-track of a synthetic signal | 81 |
| 9.3 | Pitch-track noise around the true pitch | 81 |
| 9.4 | Pitch-tracks of an original signal and of five pitch-shifted copies | 84 |
| 9.5 | Plots of the actual pitch shifting factor β' | 84 |
| 9.6 | Spectrograms of pitch-shifted signals | 85 |
| 9.7 | Pitch correction | 86 |
| 9.8 | Harmonisation | 87 |
| 9.9 | Synthesised waveform | 88 |
| B.1 | A piano octave | 98 |
| D.1 | Periodic time signal | 100 |
| D.2 | Autocorrelation of time signal | 100 |
| D.3 | Autocorrelation-based pitch detecting system | 101 |
| D.4 | A signal and its power-raised version | 101 |
| D.5 | A signal and its centre-clipped version | 102 |
| D.6 | A signal and its tertiary-clipped version | 102 |
| D.7 | AMDF | 103 |

List of Tables

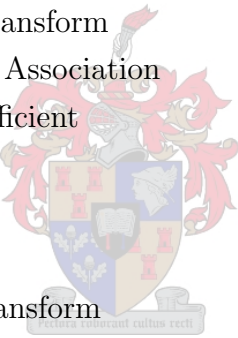
| | | |
|-----|--|----|
| 2.1 | Formant frequencies and amplitudes for spoken vowels | 14 |
| 2.2 | Formant frequencies and amplitudes for sung vowels | 14 |
| 2.3 | Vowel substitutions when singing | 14 |
| 2.4 | High-pitch vowel substitutions | 14 |
| 4.1 | Information on different window types | 29 |
| 7.1 | β calculation rules | 72 |
| 9.1 | Pitch-tracking parameters | 82 |
| 9.2 | Pitch-shifting parameters | 83 |



Nomenclature

Acronyms

| | |
|------|------------------------------------|
| ACF | Autocorrelation Function |
| AR | Auto-Regressive |
| DFT | Discrete Fourier Transform |
| FFT | Fast Fourier Transform |
| IDFT | Inverse Discrete Fourier Transform |
| IFFT | Inverse Fast Fourier Transform |
| IPA | International Phonetic Association |
| LPC | Linear Prediction Coefficient |
| LTI | Linear Time Invariant |
| MSE | Mean Squared Error |
| RMS | Root-Mean-Square |
| STFT | Short-Time Fourier Transform |
| FM | Frequency Modulation |



Notation convention

The notation in this text follows a formula that is important to note. The content deeply involves short-time analysis and therefore nearly all properties under discussion are functions of time. Say we measure property δ of signal $x(n)$ at time m . Usually δ would also be dependent on a second (usually frequency domain) parameter - let us denote it as k . As a convention we would express δ in terms of its parameters as: $\delta(m, k)$.

The text typically involves expressions such as $\phi(t_a(u), \Omega_k)$, which means ϕ is measured at time $t_a(u)$ and is further a function of Ω_k . When we omit the time parameter of a time dependent property, such as in $\phi(\Omega_k)$, we refer to a general ϕ - one that is not tied to a specific point in time.

Symbols

We list important symbols below followed by their units of measurement in square brackets, if applicable.

| | |
|------------------|---|
| n | Discrete-time index |
| s | Seconds |
| k | Discrete-frequency index |
| u | Frame index |
| $t_a(u)$ | Analysis time-instant [n] (or [samples]) |
| $t_s(u)$ | Synthesis time-instant [n] (or [samples]) |
| $t_p(u)$ | Pitch time-instant [n] (or [samples]) |
| R_a | Analysis hop-size [n] (or [samples]) |
| R_s | Synthesis hop-size [n] (or [samples]) |
| R_p | Pitch hop-size [n] (or [samples]) |
| Ω_k | Discrete-frequency [rad/sample] |
| ω | Frequency [rad/sample] |
| $x(n)$ | Discrete-time signal representing an original voice |
| $x_w(t_a(u), n)$ | A windowed portion of $x(n)$, taken at time $t_a(u)$ |
| $y(n)$ | Discrete-time signal representing a synthetic voice |
| $y_w(t_s(u), n)$ | A windowed portion of $y(n)$, taken at time $t_s(u)$ |
| $e(n)$ | Excitation signal |

| | |
|-----------------------------|---|
| $e'(n)$ | Artificial excitation signal |
| $\phi_h(n)$ | Phase of the h-th harmonic of $x(n)$ at time n [rad] (under steady state conditions) |
| $\theta_h(n)$ | Phase of the h-th harmonic of $e(n)$ at time n [rad] (under steady state conditions) |
| ω_h | Frequency of the h-th harmonic of $x(n)$ [rad/sample] (under steady state conditions) |
| $X(t_a(u), \Omega_k)$ | Short-time Fourier transform of $x(n)$ at time $t_a(u)$ |
| $\phi(t_a(u), \Omega_k)$ | Phase of $X(t_a(u), \Omega_k)$ |
| $\omega(t_a(u), \Omega_k)$ | Instantaneous frequencies of $X(t_a(u), \Omega_k)$ |
| $X(\Omega_k)$ | Short-time Fourier transform of $x(n)$ at no specific time |
| $Y(t_s(u), \Omega_k)$ | Short-time Fourier transform of $y(n)$ at time $t_s(u)$ |
| $\phi(t_s(u), \Omega_k)$ | Phase of $Y(t_s(u), \Omega_k)$ |
| $Y(\Omega_k)$ | Short-time Fourier transform of $y(n)$ at no specific time |
| $E(\Omega_k)$ | Excitation spectrum |
| $E'(\Omega_k)$ | Artificial excitation spectrum |
| $\widehat{M}(\Omega_k)$ | Direct-model of $X(\Omega_k)$ |
| \mathcal{S} | Generic scale associated with $x(n)$ |
| \mathcal{S}_j | Element number j in scale \mathcal{S} |
| $P_x(t_p(u))$ | Pitch of $x(n)$ at time $t_p(u)$ |
| $P_x^{\mathcal{S}}(t_p(u))$ | $P_x(t_p(u))$ corrected to the closest values in \mathcal{S} |

| | |
|------------------|--|
| \mathcal{H}'_n | Spectral peaks extracted from $X(t_a(u), \Omega_k)$. Each element in [rad/sample] |
| \mathcal{H}_n | Refinement of \mathcal{H}'_n . Each element in [rad/sample] |
| \mathcal{J}_q | “LULU”-smoother of order $q + 1$ |
| F_0 | Fundamental frequency |
| F_1 | First harmonic |
| α | Scaling factor |
| β | Spectral resampling factor |



Chapter 1

Introduction

1.1 Motivation

The beauty of a singing voice is due to its richness and complexity: a raw sound with many subtle nuances that can differ vastly from one individual to the next, making it a very unique instrument. In this thesis we investigate some of the engineering properties behind this mystique art.

Two very important properties of a singing voice are the pitch and the formant structure. They give a voice character and are the reason why individual voices sound different. Common agreement exists in the speech processing literature that these two properties are physiologically (nearly) independent and that the individual can change one without affecting the other [3]. A series of interesting problems arise when we change any of these properties artificially i.e. after the sound leaves the singer's mouth. The human voice is a very fragile sound and the slightest uncalculated manipulation by means of a digital computer can make it sound highly unnatural. The main topic of this thesis is to develop a method to manipulate the pitch of a singing voice in such a way that the subtleties of the voice do not get lost in the transformation process.

Singing voice - instead of speech - is used as a vehicle for the ideas developed, since the results are more interesting: By changing the pitch of a singing voice we can create new melodies from existing ones which can be used to harmonise with the original. The pitch can also be manipulated to be clamped to certain allowable frequencies, forcing the singer's voice to stay within a prescribed musical scale, thus giving a singer better intonation. If we change the pitch and the formant structure simultaneously - one independent of the other - it may be possible to transform a male voice to a female voice, and vice versa.

These transforms, or effects, find application in the world of music technology. Instead of spending hours to train two or more singers to sing in harmony, a music producer can now use software or hardware to produce backing vocals for a lead voice. Today, many singers of popular music use intelligent machines to assist them in keeping the correct

pitch throughout live shows and in the studio.

Whether these techniques are ethical or not may be debated by musicians and music producers, but from an engineering point of view it is a remarkable achievement. We proceed in the following chapters to investigate in detail how this can be done.

1.2 Key concepts

Certain terminology and concepts from the literature are used in this chapter. This section explains their meaning and functioning.

1.2.1 Formants

While singing, the throat, oral and nasal cavities of a singer are excited by an acoustic wave generated by the diaphragm and modulated by the *larynx*. The excited cavities have certain natural resonances that can be controlled by the individual. These resonances are called formants and are necessary for vowel intelligibility, uniqueness of different singers and voice projection. Their locations in the frequency domain are very important and should be well controlled during voice manipulation. Table 2.2 (p14) gives the approximate frequencies where the first three formants occur for different sung vowels.

1.2.2 Pitch

Pitch is the human ear's measure of the frequency of a sound and in most cases is the fundamental frequency. A few exceptions distinguish pitch from the fundamental frequency. Certain “psychoacoustic” effects can cause a perceived pitch to be different from the actual content of the sound. For example, when the fundamental frequency is missing from the sound, a listener would still perceive it as if it exists. The ear also perceives very high or low frequencies differently at different volume levels.

1.2.3 The phase vocoder

The phase vocoder is a way of representing a time signal. A series of overlapping frames are taken from the signal, windowed and represented in the frequency domain by a magnitude and a phase quantity associated with each frequency bin. The name “phase vocoder” comes from its use for vocal coding. A signal represented by a phase vocoder can be perfectly reconstructed under the condition that successive frames overlap. Each frame is transformed back to the time domain by an inverse Fourier transform after which the original signal can be synthesised through a overlap-and-add procedure [12]. The phase vocoder is best known for its application in the electronic transmission of speech and for

its ability to perform high quality time-scaling on signals. If a different overlap value is used during the overlap-and-add process (synthesis), the signal can be stretched or compressed in time, leaving the frequency content unmodified. It is also a very useful tool for pitch modification techniques that leave the time duration unaltered. We describe the finer workings of the phase vocoder in detail in Chapters 4 and 6.

1.2.4 Non-linear smoothers

A non-linear smoother is an algorithm applied to a signal to filter out all ill-behaved values. These data points are usually replaced by better behaved neighbours, or a weighted combination of them. Non-linear smoothers work very well for impulsive noise, in which case linear filters usually fail. Because of the non-linearity of the smoothers, the result cannot be written in closed form and is mathematically complicated. This forces us to deal with the smoothers heuristically.

We make use of a combination of two unsymmetric smoothers to devise a powerful smoothing algorithm. The algorithm, introduced by Rohwer[16], is nicknamed “*LULU*”. (For more details see Chapter 3.)

1.3 Objectives

The objectives of this study are:

- To investigate the properties of singing from an engineering perspective.
- To design and implement a robust pitch-tracking algorithm that specialises in singing voice.
- To design and implement an algorithm for manipulating the pitch of a singing voice while leaving the formants in tact.
- To design design and implement a technique for artificial formant manipulation.
- To describe in detail the solutions to the problems we encounter while aiming for these objectives.

1.4 Contributions

The following are the contributions made by this study:

- The application of a “*LULU*” non-linear filter and some basic signal processing on a spectrum made a powerful spectral peak extractor. See Chapter 3 and section 5.2 of Chapter 5.

- The reliable knowledge of the peak locations in a spectrum made it possible to devise a robust pitch-tracking algorithm specialising in singing voice. See Chapter 5.
- We designed a method for modelling the magnitude of a spectrum. See Chapter 6, specifically section 6.2.3.
- A detailed description on modifying an excitation signal, a topic we identified as a shortcoming in most papers on phase vocoders. Here we present experimental details based on our own experience.
- We designed a system that creates harmonies from an existing recording which may be used to harmonise with the original. The same system can be utilised to do pitch correction on a recording. See Chapter 7.
- A synthesis system that uses a short vowel to create long notes of which the frequencies are controlled by a MIDI-keyboard. The result is a sound reminiscent of the singing technique of a choir member. See Chapter 8, section 8.2.
- A voice gender transform algorithm. We attempt to devise an algorithm that changes a recorded vocal performance into performance by the opposite sex. See Chapter 8 section 8.3.

1.5 High level overview



1.5.1 Background

Chapters 2 to 4 serve as a background for the subsequent chapters that follow them, and present a detailed discussion of the concepts introduced in 1.2.

1.5.2 Main algorithms

Bearing in mind the key concepts introduced in section 1.2, we designed two main algorithms detailed in Chapters 5 and 6.

1. Pitch detection algorithm
2. Pitch-shifting algorithm

Pitch detection algorithm

To estimate the fundamental frequency of a signal, we divide it into overlapping frames and transform them by means of an FFT. A non-linear smoother is used to filter out unwanted noise, leaving (ideally) only peaks caused by the excitation. For a single frame,

the exact frequencies of the most prominent of these peaks are determined by calculating a frequency offset from the centre of the bin where the peaks occur. The offset is determined by calculating the deviation from the expected phase advance between frames for a certain peak. The expected phase advance is the phase advance between two frames for a sinusoid having the same frequency as the centre of the bin it falls into. The phase deviation from the expected phase advance is used to calculate the frequency deviation from the bin centre. This exact frequency is called the *instantaneous frequency*, of which the detail can be found in Chapter 4, section 4.3.2.

Once the refined frequencies of the peaks are known, we do a power calculation. We sum the power of each peak and its harmonics (or rather locations where harmonics are expected). This process is known as harmonic summing. The peak and its harmonics that give the highest power sum is regarded as the pitch of the frame. This technique is robust and will discard any spurious peaks that survived the smoothing process that are likely to be mistaken for the pitch. (At a later stage, we will address the case where the frequency estimate is lower than the true pitch. In such cases the power sum will be higher than the true power sum. We introduce simple check in Chapter 5 to avoid such confusions.)

A pitch value for each frame makes up an array of frequency values called a pitch-track. Figure 5.3 (p43) shows such a pitch-track of a scale sung by a male vocalist. Chapter 5 provides detail on this pitch detection algorithm.

Pitch-shifting algorithm

Pitch-shifting refers to a process that changes the fundamental frequency of a signal, leaving the spectral shape unaltered. In terms of a human voice, it means changing the pitch without moving the formants. See Figure 6.18 (p65) for spectra of an example where changing the fundamental frequency of a short signal has little effect on the formant locations.

Say we have a windowed frame of voiced singing that is in steady state. The pitch of the frame is called the *source pitch* and the *target pitch* is the pitch that the frame should have after pitch-shifting. Once these two values are known, we can calculate a measure of pitch shift called β , where:

$$\beta = \frac{\text{source pitch}}{\text{target pitch}}. \quad (1.1)$$

A unity value should leave the pitch unmodified, while a value of 0.5 doubles the pitch and a value of 2 halves the pitch. Before shifting the pitch of a frame, the excitation and the spectral envelope should be separated. The excitation may then be pitch-shifted by means of a spectral resampling process. Once this is done the spectral envelope is restored. This process changes the pitch of a frame in a very natural way.

1.5.3 Applications of main algorithms

A detailed discussion of the applications of these algorithms can be found in Chapters 7 and 8. The applications are summarised below.

Synthetic harmony vocals for an original voice

We have a signal $x(n)$, a recording of a vocal performance, and would like to create a synthetic signal $y(n)$ having a different pitch-track. After a pitch-track has been calculated for the signal $x(n)$, we divide $x(n)$ into overlapping frames and form a phase vocoder. The pitch curve is then interpolated so that each frame (or analysis instant) of the phase vocoder can be associated with a certain frequency value. These values are the source pitches of the frames. Next we need to calculate target pitches for the frames, which are the pitch values the frames should have in order to synthesise $y(n)$ through overlap-and-add.

The values of the target pitches are governed by a *rule*, specified by a user. The rule acts as a mapping function for the source pitch: For each source pitch value, there exists one target pitch value. Chapter 7 provides more detail about the target pitch calculation. Once each frame has a source- and target pitch, a measure of pitch shift β can be calculated for each frame, as in equation 1.1.

By pitch shifting each frame taken from $x(n)$, we get a series of new frames, which can be overlapped and added to form $y(n)$. (It is very important to restore the original phase to avoid phase distortion.) If the *rule* is governed by music theory, $y(n)$ may be a complementing harmony to $x(n)$.

Pitch correction on an original voice

If we apply the above method and calculate the target pitches so that they are frequency values from a musical scale closest to the source pitch values, we can improve $x(n)$. The result should be a version of $x(n)$ with much less deviations from the scale in which it was performed.

Vocal wave synthesis

Based on a few stationary frames of a sung vowel, we can create a signal $y(n)$ that has arbitrary length and pitch (within bounds), but has the same spectral envelope as the original vowel. Before synthesis, we calculate the pitch of the frame and use it as the source pitch. The target pitch comes from a MIDI-keyboard played by a musician. The keyboard streams information, indicating the key that is pressed and the duration of the note.

The original frames are copied and pitch-shifted to the desired frequencies to form synthesis instants after which the instants are overlap-and-added to form $y(n)$.

Gender transformation

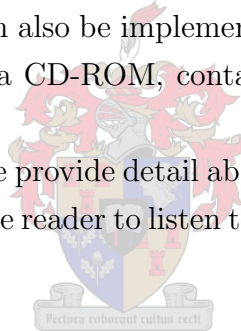
We investigate gender transformation by pitch-shifting a voice by a constant factor and by shifting the formants to new frequency locations. We describe the formant shifting process in section 6.2.3.

1.6 Implementation and audio results

The systems we described in section 1.5 are from our own design and we implemented the theory in a number of computer programs. We used MATLAB as a design tool and did the final implementation using GNU C++. Apart from our own designs, we also wrote a phase vocoder program to demonstrate its working.

The programs deliver audio results which serves as proof that this research is not only academic of nature, but can also be implemented to provide pleasing sound samples. The samples may be found on a CD-ROM, containing an HTML presentation which is included with this document.

In the chapters that follow we provide detail about the concepts and designs mentioned in this chapter. We encourage the reader to listen to the sound samples as they will provide some extra insight.



Chapter 2

The Singing Voice

“Music expresses that which cannot be said and on which it is impossible to be silent.”

- *Victor Hugo*

2.1 Introduction

We give a broad introduction to singing, covering the history, the physiological processes and mathematical properties.



2.2 Brief notes on the history of singing

Singing, the vocal production of musical tones, is the oldest known musical instrument and predates the development of spoken language. This ancient art fulfilled many important functions to the individual and the social group in the areas of entertainment, communication and religion. We surmise that early singing was individualistic and random and a simple imitation of the sounds heard in nature. It is not clear at what point it became meaningful and communicative. Reconstructing history on the basis of cross-sectional observations, thus comparing primitive singing with more advanced musical structures, suggests a possible scenario of musical development which started with simple melodic patterns based on several tones. A logical phase to follow would be several persons singing in unison with matching pitch movement, which gave rise to the melodic and harmonious patterns governed by the scales we know today.

The cradle of modern-day singing is undoubtedly the opera. For a long time, the opera was very experimental and the idea of singing in a key took some time to be established. The opera made people aware of the beauty and the complexity of the human voice, after which more composers wrote music for the voice in the standardised notation that was then developed in other areas of music.

One of the historical benchmarks for singing was Beethoven's famous ninth symphony performed in 1824 for the first time. It was the first time that singing was included in a symphony - something unheard of at that time. The last movement of the symphony is a powerful combination of orchestral music joined by a big choir and solo vocal performances.

So-called classical singing was very popular for centuries but phonograph recordings and radio broadcasting brought new styles of music into people's houses. Blues, jazz and swing became very popular and finally mankind saw the dawn of *popular music*, which is the backbone of the modern-day industry. Most modern day music styles rely heavily on singing and without the human voice it would be dull, empty and without emotion [6].

It may seem ironic that the human voice, the oldest known instrument, is less well understood than most other present-day instruments. This is due to the unaccessibility of the various physiological components used by the singer. Rossing [18] compares the study of the physics of singing to studying a violin which is played from behind an opaque screen with only a small hole to peek through!

The mysterious and interesting art of singing is an important part of the world we live in. It is worth studying not only its history, but also its physical and mathematical properties. This is the purpose of the rest of the chapter.

2.3 Speech production

Since singing is a well-controlled form of speech, we will first consider the generally accepted speech production model after which we will discuss the differences between speech and singing. It is useful to consider singing as a special form of speech because speech processing is a well developed science on which many publications are available.

The speech waveform is an acoustic pressure wave, originating from voluntary movements of anatomical structures which make up the human speech production system. The most important parts of the system are the *lungs*, *trachea* (windpipe), *larynx* (organ of voice production), *pharyngeal cavity* (throat), *oral cavity* (mouth) and *nasal cavity* (nose). The *pharyngeal cavity* and the *oral cavity* are grouped together to form the *vocal tract*, and the *nasal cavity* is also called the *nasal track*. Finer anatomical features that are critical to speech production include the *vocal folds*, *velum* or *soft palate*, *tongue*, *teeth* and *lips*.

The three main cavities of the speech production system are exited by the *lungs* and *diaphragm*. The produced acoustic wave, also called the excitation waveform, is filtered by these cavities before leaving through the mouth and nose. The *vocal cords*, found inside the *larynx* vibrate because of a stream of air from the lungs, pressed upwards by the *diaphragm*. The rate of the vibration is determined primarily by the mass and tension

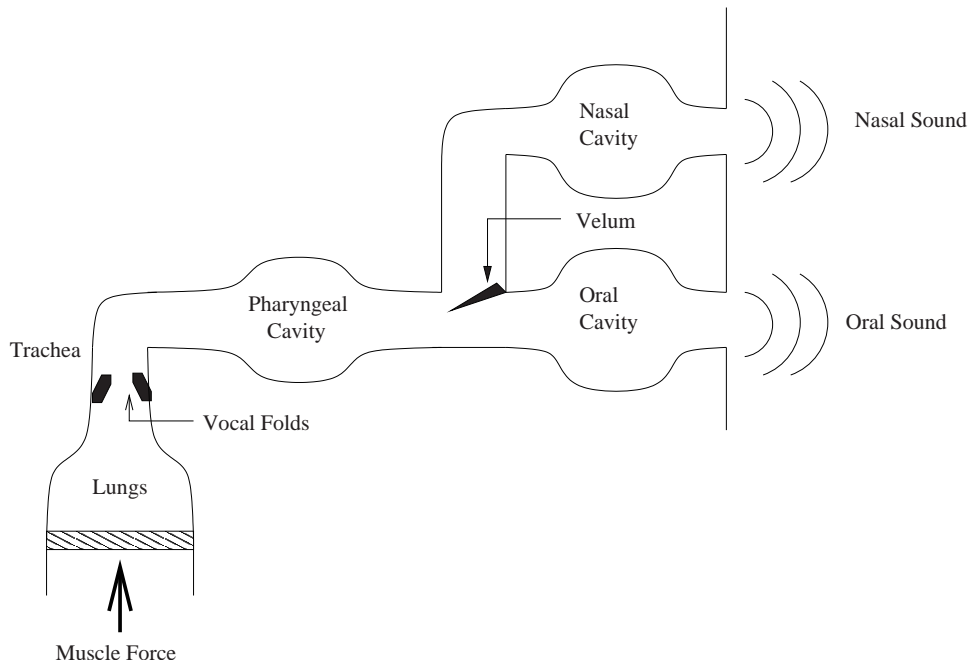


Figure 2.1: Diagram of the vocal tract [3]

of the *vocal cords*. In the case of adult males, these cords are typically longer and heavier than in the case of females and therefore vibrate at lower frequencies. A simplified model of the vocal tract is shown in Figure 2.1.

The vocal track introduces resonating frequencies that humans control to pronounce different vowels and voiced consonants. These resonances - theoretically independent of the excitation - cause local maxima in the spectral envelope and are called formants. A spectrogram is a convenient way to view formants. It is a plot of time versus frequency of a time signal where dark areas indicate the power. Figure 6.1 (p47) is a good example of a spectrogram of a male voice. The thick dark lines indicate the formants. Another way to see the formants is to use *linear prediction*. If we can isolate a stationary part of a signal, we can calculate the *linear prediction coefficients (LPC)* of the section. The *LPC*-coefficients form an all-pole filter that approximates the behaviour of the signal it was derived from. A frequency sweep of the filter produces an approximation of the spectral envelope. We will return to *LPC*-analysis in Chapter 6 and the detail about the calculation of the coefficients may be viewed in Appendix A. Figure 2.2 shows the response of an *LPC* filter derived from a stationary section of the voiced part of “us” and the formants are clearly visible.

The positions of the formants are important for the listener to recognise different speakers as well as different vowels, voiced consonants and diphthongs. The first three formants are crucial for vowel recognition while the higher formants, among other properties, enable a listener to distinguish one speaker from another [18].

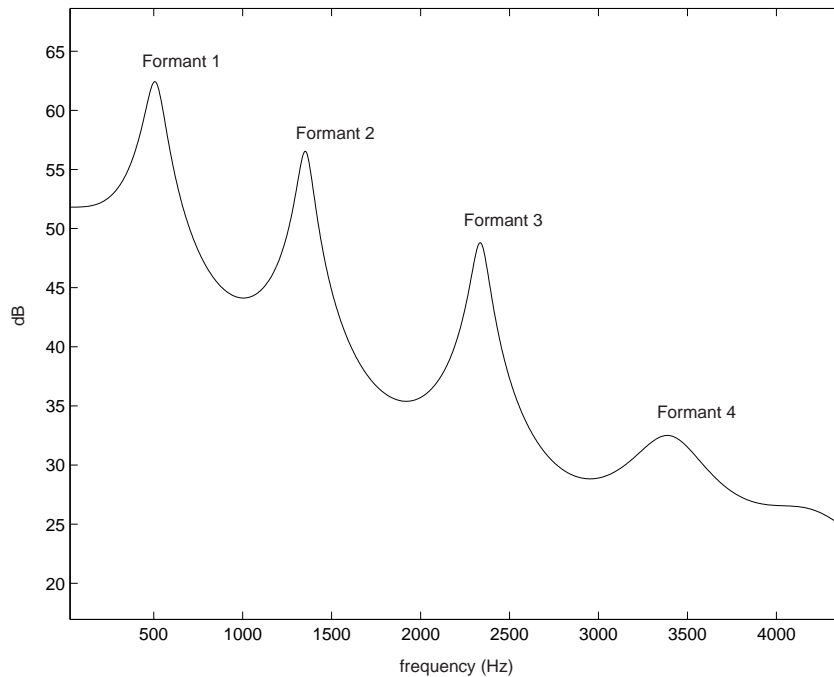


Figure 2.2: LPC spectrum of a vowel (/Λ/)

2.4 Singing: a special form of speech

In both speech and singing, there is a division of labour between the vocal cords and the vocal tract. The vocal cords control the pitch of the sound, whereas the vocal tract determines the vowel sounds through its formants. The pitch and the formant frequencies are nearly independent but trained singers, especially sopranos, may tune their formant frequencies so that they match one or more of the harmonics of the sung pitch.

Sung vowels are fundamentally the same as spoken vowels, although singers do change a few vowel sounds in order to improve the musical tone. An analysis of individual vowel formants reveals some substantial spectral changes. Figure 2.3 shows spectra of the same vowel /æ/, sung and spoken. Note that the first formant hardly moves but the second formant is significantly lower in frequency. The third and fourth formants are unchanged in frequency, but are significantly stronger when the vowel is sung.

Four significant articulatory differences between speech and singing are [18]:

- the *larynx* is lowered,
- the jaw is opened larger,
- the tongue tip is advanced in the back vowels /u/, /o/ and /a/; and
- the lips are protruded in the front vowels.

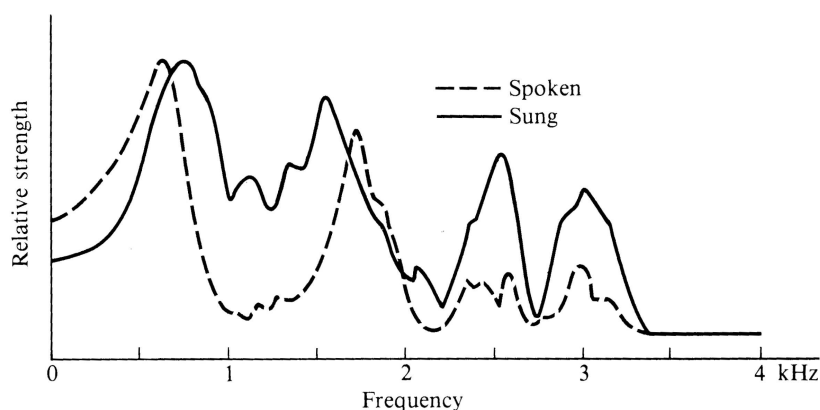


Figure 2.3: Sung and spoken vowel (/æ/) [18]

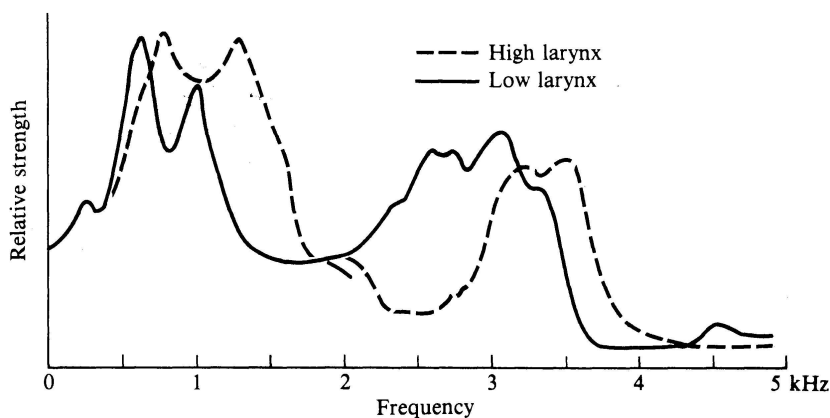


Figure 2.4: Trained and untrained singing of a vowel (/a/) [18]

Trained singers, especially male opera singers, show a strong extra formant somewhere around 2500-3000Hz [18]. This is called the “singer’s formant” and is more or less independent of the vowel being sung. This formant gives carrying power and brilliance to the male voice. An interesting act of nature is that the singer’s formant is near one of the resonant frequencies of the human ear canal, which gives an additional auditory boost.

The reason for the extra formant is attributed to a lowered *larynx*, which along with a widened *pharynx*, forms an additional resonating cavity. Untrained singers tend to raise their *larynxes* as they raise their pitch and this is why popular singing sounds different from operatic singing. (It is interesting that so-called untrained singing has become a standard in its own right in the genres of rock and popular music.)

Figure 2.4 shows the spectrum of a trained and an untrained voice singing the same vowel and it is evident that the formants are significantly higher when the *larynx* is raised. The result is a high frequency boost, which is why popular singing sometimes sound hoarse and “whispery”. The formant frequencies of long Swedish vowels were

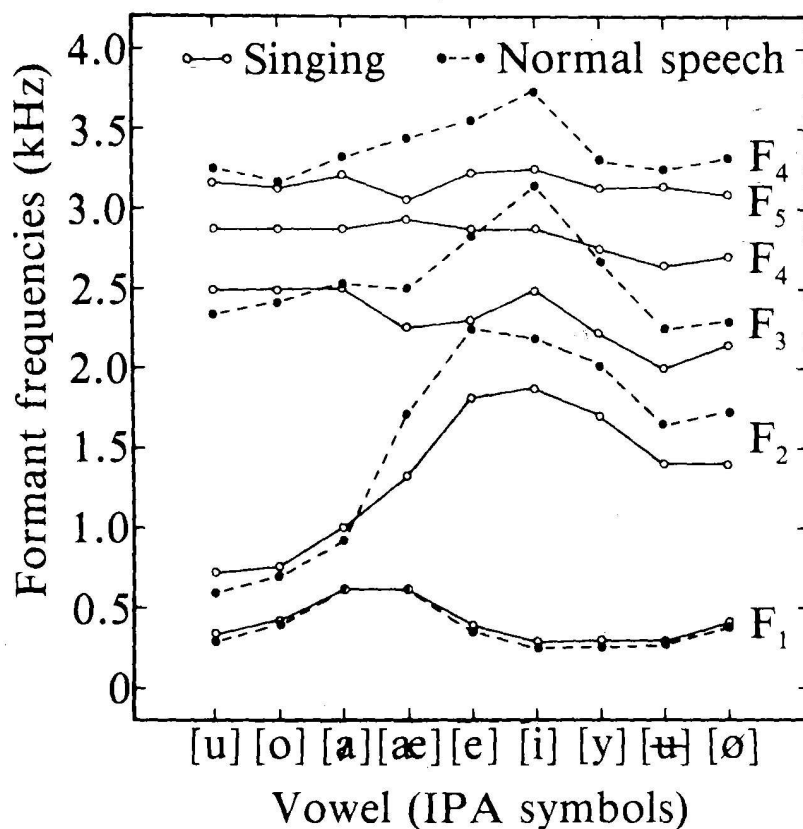


Figure 2.5: Formant frequencies of long Swedish vowels in normal male speech and in professional male singing [18].

calculated for normal male speech and singing and are displayed in Figure 2.5. Note that the first formant hardly differs from speech to singing and that the fourth singing formant (the singer's formant) is rather constant compared to the other singing formants.

For a further comparison between sung and spoken vowels, Rossing [18] provides two tables, one for speech and one for singing, indicating the average frequencies of the first three formants of the basic vowels. See Tables 2.1 and 2.2.

As might be noted, a few substitutions were made in the two tables, the reason being that some vowels are pronounced differently when sung so that the formants may support the pitch and are therefore denoted by different IPA symbols. These substitutions are listed in Table 2.3. Singers also find it convenient to substitute certain vowels when the sung pitch rises. The new vowels are chosen so that the formants support a higher pitch. This technique is a trade-off between intelligibility and sound projection and helps opera singers to rise above the orchestra. The substitutions are listed in Table 2.4. A further point to note is that the average female pitch is more or less twice the average male pitch, corresponding to an octave difference in musical terms, but the formants are on average only 25% higher [18]. This is important when designing software to change

Table 2.1: Formant frequencies and amplitudes for spoken vowels

| Formant | /i/ | /I/ | /ε/ | /æ/ | /a/ | /o/ | /U/ | /u/ | /Λ/ | /ʒ/ |
|---------|------|------|------|------|------|------|------|------|------|------|
| F1 ♂ | 270 | 390 | 530 | 660 | 730 | 570 | 440 | 300 | 640 | 490 |
| ♀ | 310 | 430 | 610 | 860 | 850 | 590 | 470 | 370 | 760 | 500 |
| F2 ♂ | 2290 | 1990 | 1840 | 1720 | 1090 | 840 | 1020 | 870 | 1190 | 1350 |
| ♀ | 2790 | 2480 | 2330 | 2050 | 1220 | 920 | 1160 | 950 | 1400 | 1640 |
| F3 ♂ | 3010 | 2550 | 2840 | 2410 | 2440 | 2410 | 2240 | 2240 | 2390 | 1690 |
| ♀ | 3310 | 3070 | 2990 | 2850 | 2810 | 2710 | 2680 | 2670 | 2780 | 1960 |

Table 2.2: Formant frequencies and amplitudes for sung vowels

| Formant | /i/ | /I/ | /ε/ | /æ/ | /a/ | /ɔ/ | /ʊ/ | /u/ | /Λ/ | /ʒ/ |
|---------|------|------|------|------|------|------|------|------|------|------|
| F1 ♂ | 300 | 375 | 530 | 620 | 700 | 610 | 400 | 350 | 500 | 400 |
| ♀ | 400 | 475 | 550 | 600 | 700 | 625 | 425 | 400 | 550 | 450 |
| F2 ♂ | 1950 | 1810 | 1500 | 1490 | 1200 | 1000 | 720 | 640 | 1200 | 1150 |
| ♀ | 2250 | 2100 | 1750 | 1650 | 1300 | 1240 | 900 | 800 | 1300 | 1350 |
| F3 ♂ | 2750 | 2500 | 2500 | 2250 | 2600 | 2600 | 2500 | 2550 | 2675 | 2500 |
| ♀ | 3300 | 3450 | 3250 | 3000 | 3250 | 3250 | 3375 | 3250 | 3250 | 3050 |

Table 2.3: Vowel substitutions when singing

| | | |
|--------|-----|-----|
| spoken | /o/ | /U/ |
| sung | /ɔ/ | /ʊ/ |

Table 2.4: High-pitch vowel substitutions

| | | | | | | |
|--------------|-----|-----|-----|-----|-----|-----|
| Normal range | /i/ | /ε/ | /æ/ | /a/ | /ɔ/ | /u/ |
| High range | /I/ | /a/ | /a/ | /Λ/ | /Λ/ | /U/ |

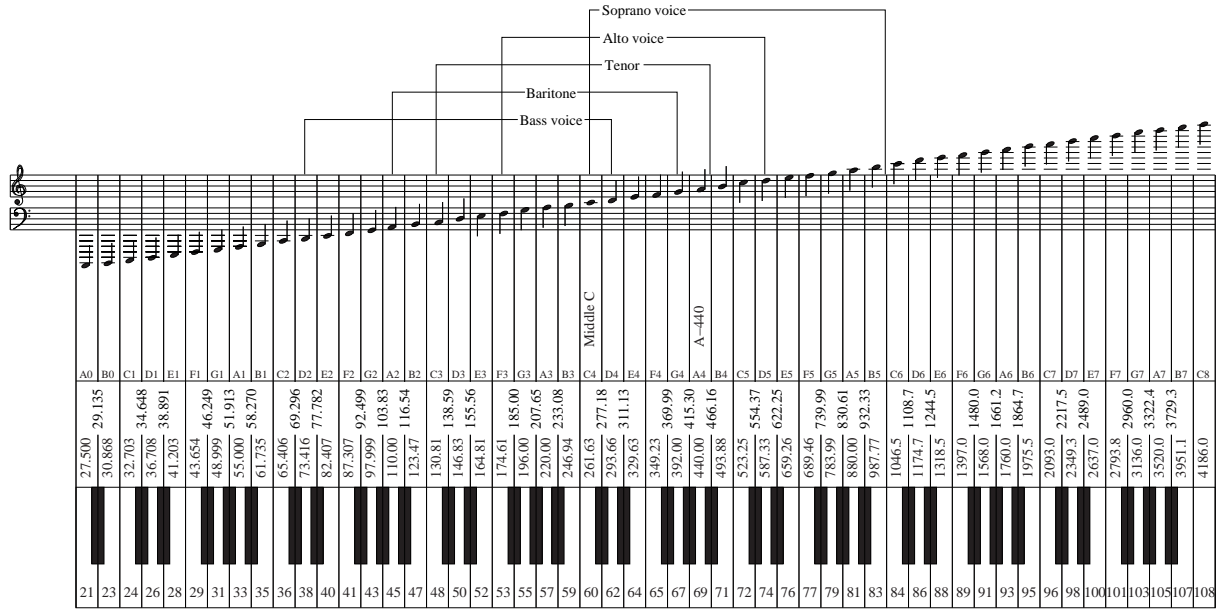
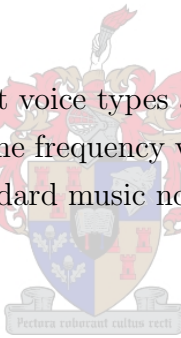


Figure 2.6: Ranges of voice types

the gender of a voice.

The possible ranges of different voice types are given in Figure 2.6 along with a comparison of the range of a piano. The frequency values are indicated above the piano keys. The figure further provides a standard music notation expression for the voice and piano ranges.



2.5 Mathematical model

The generally accepted speech production model is modelled as the output of a time-varying linear filter driven by an excitation signal $e(n)$. The excitation could be a sum of harmonic related narrow-band signals, which is useful for modelling voiced speech segments. The harmonic relation between the narrow-band signals is clear from Figure 2.7. It could also be a stationary random sequence with a flat power spectrum, which is used for unvoiced modelling.

The filter parameters accounts for the identity (spectral characteristics) of the sound for the two different types of excitation [10]. The time varying filter approximates the effect of the transmission characteristics of the *vocal tract* and *nasal cavity* combined with the shape of the glottal pulse. The input-output behaviour of the system is characterised by its impulse response $s_n(m)$, defined as the response of the system at time n . We can view $s_n(m)$ as a snapshot of the vocal tract at time n , where m is the time index of the impulse response. An equivalent description is given by the Fourier transform of $s_n(m)$

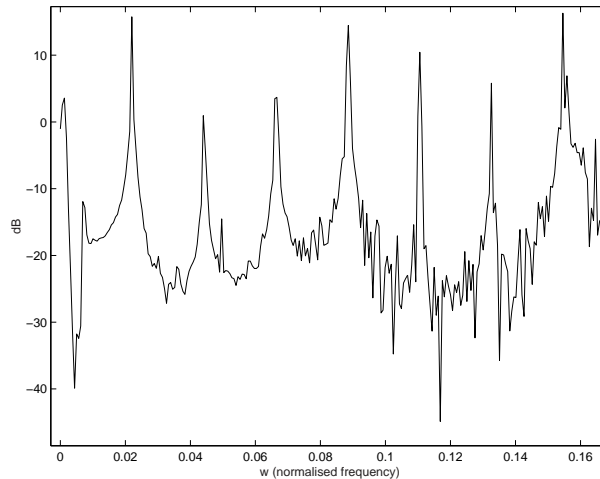


Figure 2.7: Excitation spectrum of a vowel

with respect to m :

$$\sum_{m=-\infty}^{\infty} s_n(m) e^{-j\omega m} = S(n, \omega) e^{j\psi(n, \omega)}. \quad (2.1)$$

$S(n, \omega)$ and $\psi(n, \omega)$ are referred to as the time varying amplitude and phase of the system. The non-stationarity of $s_n(m)$ depends on the movements of the physical articulators and is slow compared to the time variation of the speech waveform. Therefore we may say that $s_n(m)$ is a *quasi-stationary* system. For voiced speech or singing, the excitation signal $e(n)$ may be represented by a sum of harmonic related complex exponentials with unit amplitude and zero initial phase, since the impulse response of the vocal tract accounts for the phase and amplitude. For the sake of simplicity, we assume that the excitation always has P harmonics, including the fundamental frequency. We write:

$$e(n) = \sum_{h=0}^{P-1} e^{j\theta_h(n)}, \quad (2.2)$$

where $\theta_h(n)$ is the excitation phase of the h -th harmonic. Each harmonic has a frequency and we denote the frequency of the h -th harmonic by w_h - a constant under stationary assumptions. If we assume *quasi-stationarity*, we can state that

$$\theta_h(m) = \theta_h(n) + (m - n)w_h \quad (2.3)$$

for small $|m - n|$, since phase advance is the product of frequency (w_h) and time duration ($m - n$).

The amplitudes of all the pitch harmonics are equal and $S(n, \omega)$ accounts for the magnitude of the spectrum. The pitch harmonics have zero initial phase because $\psi(n, \omega)$ alone is responsible for the phase of the pitch harmonics.

According to filter theory [14], we can now write a standard speech waveform as:

$$x(n) = \sum_{m=-\infty}^{\infty} s_n(m)e(n-m), \quad (2.4)$$

Equation 2.4 is a convolution of the excitation signal with the vocal tract filter. If we assume stationarity for the duration of $s_n(m)$, we can replace the excitation with its local sinusoidal representation. Thus we substitute equation 2.2 into equation 2.4:

$$\begin{aligned} x(n) &= \sum_{m=-\infty}^{\infty} s_n(m) \left[\sum_{h=0}^{P-1} e^{j\theta_h(n-m)} \right] \\ &= \sum_{m=-\infty}^{\infty} \sum_{h=0}^{P-1} s_n(m) e^{j\theta_h(n-m)} \end{aligned} \quad (2.5)$$

Using equation 2.3 we can rewrite $\theta_h(n-m)$ as $\theta_h(n) + (-m)\omega_h$ and substitute the result. With the above in mind we change the order of the summations:

$$\begin{aligned} x(n) &= \sum_{h=0}^{P-1} \sum_{m=-\infty}^{\infty} s_n(m) e^{j\theta_h(n)} e^{(-m)j\omega_h} \\ &= \sum_{h=0}^{P-1} \left[\sum_{m=-\infty}^{\infty} s_n(m) e^{-jm\omega_h} \right] e^{j\theta_h(n)} \end{aligned} \quad (2.6)$$

We note that the expression inside the brackets is the Fourier transform of the impulse response of the vocal tract. Substituting equation 2.1 into equation 2.6 yields:

$$\begin{aligned} x(n) &= \sum_{h=0}^{P-1} S(n, w_h) e^{j(\theta_h(n) + \psi(n, w_h))} \\ &= \sum_{h=0}^{P-1} A_h(n) e^{j\phi_h(n)} \end{aligned} \quad (2.7)$$

where $A_h(n) = S(n, w_h)$ and $\phi_h(n)$ is the sum of the excitation phase and the system phase:

$$\phi_h(n) = \theta_h(n) + \psi(n, w_h) \quad (2.8)$$

$\phi_h(n)$ is often referred to as the *instantaneous phase* of the h -th harmonic. Because the system phase $\psi(n, w_h)$ is slow varying with respect to n we may develop $\phi_h(n)$ in the neighbourhood of n according to equation 2.3:

$$\phi_h(m) = \phi_h(n) + (m-n)\omega_h \quad (2.9)$$

for small $|m-n|$.

We showed that, according to equation 2.7, a vocal sound's harmonics amplitudes are controlled by the vocal tract alone and that the phase of the harmonics is a result of both

the excitation phase and the phase of the vocal tract's filter. This model only applies for voiced speech, but since singing relies almost completely on its voiced sections, we will use this model throughout this thesis. The above explanation is based on a proof found in [12].

The importance of *instantaneous phases* will become clear in Chapter 5 where we will use them to determine fundamental frequencies of signals.

2.6 Summary

We described how singing developed from a pre-speech time to the highly acclaimed form of art we know today. The production of singing can be compared to that of speech since the same anatomical elements are used by the individual, but controlled differently. On an auditory level, the main difference is that the pitch and projection are the most important properties of singing while intelligibility is the most important property of speech.

The pitch is an exponent of the modulation of the air flow from the diaphragm by the vocal folds. A constant modulation at the correct frequency is called a note. The individual controls the projection and intelligibility by changing the frequency behaviour of the vocal tract, causing certain frequencies to resonate. The airflow from the diaphragm can be expressed as a sum of complex exponentials at frequencies of the pitch or its harmonics. This signal, called the excitation, is filtered by the vocal track which can be expressed as a time-varying filter. The sound leaving through the nose and mouth can be expressed as a convolution between the excitation and the time varying vocal tract filter.

We have shown that the resulting sound's harmonic amplitudes are controlled by the vocal tract alone and that the phase of the harmonics is a result of both the excitation phase and the phase of the vocal tract's filter. This combined phase is called the *instantaneous phase*.

Chapter 3

“*LULU*”: a Non-Linear Smoother

3.1 Introduction

This thesis relies heavily on the output of a Fast-Fourier-Transform (FFT). Because of short-term analysis we are forced to window a signal into time frames. The spectral characteristics of the windowing function manifests as artifacts in the spectral domain that can be viewed as noise, corrupting the true spectrum. These artifacts and other non-harmonic content in the spectrum make it difficult to extract useful information from the FFT. Aiming to solve these problems, we investigate non-linear smoothers.

The theory of linear smoothers is well developed. A linear smoother (also referred to as a linear filter¹) relies on the principle of replacing a certain data point with a weighted average of its closest neighbours. The order of the smoother specifies how many neighbours are taken into account. Linear smoothers are good for smoothing data that is well behaved, e.g. data corrupted by Gaussian noise. In the case of impulsive noise with unreasonable amplitude, a linear smoother will not succeed. It will merely spread the impulsive noise over the time domain. In cases like these we have to turn to the family of non-linear smoothers.

The theory behind non-linear smoothers are mathematically complicated and incomplete but we can understand and implement them heuristically [17]. Non-linear smoothing means we replace unacceptable outliers with better behaved neighbouring points.

Rohwer [16] applied a pair of unsymmetric² smoothers (\mathcal{L} and \mathcal{U}) for the purpose of filtering data corrupted by impulsive noise. In the sections that follow, we will describe how to combine these smoothers to form a “*LULU*” smoother, as it is nicknamed, and

¹The technical difference between a smoother and a filter is that a smoother uses past, present and future values relative to the point being calculated as opposed to filter using only past and present values. In this chapter we use the two terms interchangeably.

²Smoothing either upward or downward outliers

compare it to a median filter.

Strong relationships exist between the components of “LULU” and the morphological filters used in image processing, such as found in [5] and [20]. To maintain clarity, these relationships will be pointed out.

3.2 Basic non-linear smoother concepts, notation and terminology

3.2.1 Notation and terminology

We adopt the notation and terminology used by Rohwer [16] and Marquardt [11]:

- A series of data points is denoted by a lower case letter, e.g. x . Individual elements are denoted by $\dots x(i-1), x(i), x(i+1) \dots$
- A set of points obtained by selecting a subinterval of a series is denoted by $x(s, t)$, where $x(s, t) = x(s), x(s+1), \dots, x(t-1), x(t)$, given that $t \geq s$. **This subinterval notation applies to this chapter only.**
- A smoother, represented by an uppercase letter, e.g. R , is defined as an operator or transform, that maps each point in the input sequence to a point in the output sequence. Therefore, the statement $y = Rx$ represents the operation of smoother R on sequence x , giving sequence y . Individual elements can be addressed by indexing: $y(i) = Rx(i)$.³
- $y = R_m x$ signifies a smoother R with window size $m + 1$ operating on x to give y . If m is omitted, the window size is arbitrary.
- Certain non-linear smoothers are called *rank-based selectors* since data is smoothed by replacing a point with a better natured one inside a given window (including the particular point). The point is chosen on its relative rank amongst the other points inside the window. For example, a median filter selects the centre point in a sorted list.
- The concatenation, or combination, of two smoothers refers to a smoother operating on the output of another smoother. If S and R are two smoothers operating on x (in the given order) to give y , we can state $y = RSx$. If both R and S use windows

³Expressing individual elements in this manner may be confusing. We must point out that $y(i)$ cannot be calculated from $x(i)$ alone, but needs a window of points around $x(i)$. Whenever we use an expression like $y(i) = Rx(i)$, and R is a smoother, we imply that points around $x(i)$ are also considered.

of size $m + 1$, it is clear that an element of the resulting output is a point selected from $2m + 1$ points. This total is termed the *support* of the resulting smoother.

- Smoothers can be ordered by a comparison of their output and input. For example, consider the two smoothers, R and S . If $Rx(i) \geq Sx(i)$ for all valid values of i , we can state that $R \geq S$.
- A smoother is *idempotent* if it does not change its own output, i.e. $RRx(i) = Rx(i)$ or $RR = R$.

3.2.2 The median filter

A *median filter*, say M , is a well known non-linear smoother. Mx selects the median⁴ from the current window, and replaces the point around which the window is centred. If $y = Mx$ and the window size is $2m + 1$, then $y(i)$ is the median of $x(i - m, i + m)$.

Let us consider an example where the median filter would prove useful. As mentioned in 3.1, non-linear smoothers are good for filtering impulsive noise. Figure 3.1 shows a steady sloping curve corrupted by impulses, and the results after being filtered by a linear smoother as well as a median filter. The two smoothers have the same window size.

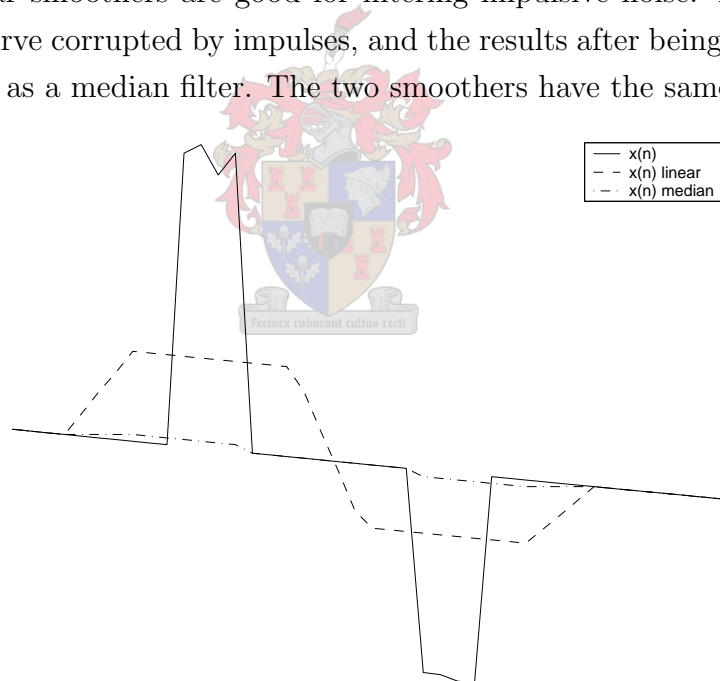


Figure 3.1: Median filter versus linear smoother

It is clear that the linear smoother only spreads the impulse, while the median filter suppresses it very successfully. Median filters are very useful, but they have significant drawbacks:

⁴Sorts the elements and selects the centre one.

- The rank selection can only be performed once the elements in the window are sorted. Sorting is computationally expensive since sorting algorithms can have \mathcal{N}^2 -complexity in the worst case.
- Median filters are not *idempotent*, which is a very desirable property for a smoother. This means that different resulting signals can be obtained from the same smoother and input signal, by applying the smoother repeatedly.
- It can be very hard to predict the result of median smoother, and after smoothing, we have no idea how the smoother achieved the result. Rohwer describes a median filter as having “enigmatic” behaviour.

3.3 Smoothers: \mathcal{L} and \mathcal{U}

Let us consider a well behaved sequence with an occasional outlier in the upward direction. We can remove the upward pulses from the sequence by applying a running minimum. In morphology, this process is referred to as *erosion*. Erosion will succeed as long as the widths of the impulses are less than the window length. If the sequence also contains downward pulses, they will be widened. This is a shortcoming that can be overcome by applying a running maximum (called dilation in morphology) after the running minimum. This will restore the downward pulses to their original width and will also preserve downward trends. To summarise, a running minimum, followed by a running maximum, will:

- remove upward impulses
- retard upward trends
- advance downward trends.

Rohwer calls this an \mathcal{L} -smoother, where \mathcal{L} denotes the above described operation on a given sequence. In morphology, this corresponds to a process called *opening*. Figure 3.2 illustrates the steps of such smoother.

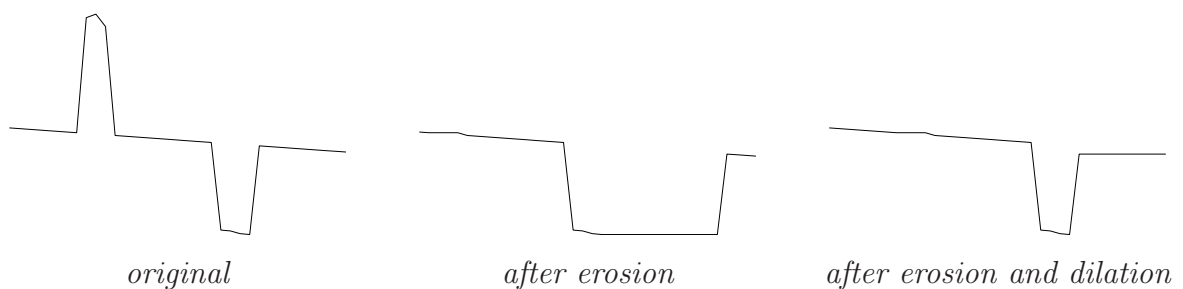


Figure 3.2: Steps of an \mathcal{L} -smoother

Following the same line of thought, we can state that a running maximum, followed by a running minimum, will:

- remove downward impulses
- retard downward trends
- advance upward trends.

The above process is called a \mathcal{U} -smoother by Rhower and called *closing* in morphology. The working of a \mathcal{U} -smoother is illustrated in Figure 3.3.

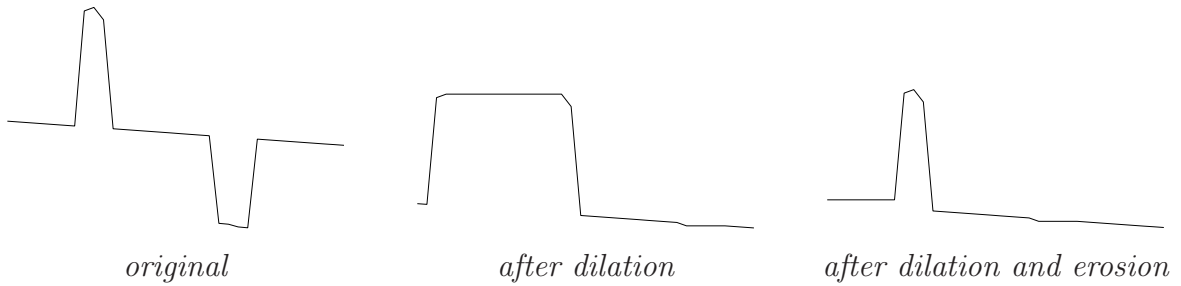


Figure 3.3: Steps of a \mathcal{U} -smoother

Given the notation in section 3.2, we can write the equations for \mathcal{L} - and \mathcal{U} -smoothers with window size n :

$$\begin{aligned}
 \mathcal{L}x(i) &= \mathcal{L}_m x(i) \\
 &= \max\{\min\{x(i-m, i)\}, \\
 &\quad \min\{x(i-m+1, i+1)\}, \\
 &\quad \dots \\
 &\quad \min\{x(i, i+m)\}\}
 \end{aligned} \tag{3.1}$$

$$\begin{aligned}
 \mathcal{U}x(i) &= \mathcal{U}_m x(i) \\
 &= \min\{\max\{x(i-m, i)\}, \\
 &\quad \max\{x(i-m+1, i+1)\}, \\
 &\quad \dots \\
 &\quad \max\{x(i, i+m)\}\}
 \end{aligned} \tag{3.2}$$

Note that the support for both \mathcal{L} and \mathcal{U} is $2m+1$.

\mathcal{L} and \mathcal{U} can be shown to be *idempotent*, i.e. $\mathcal{L} = \mathcal{L}\mathcal{L}$ and $\mathcal{U} = \mathcal{U}\mathcal{U}$, which is an improvement on a median filter. If \mathcal{L} , \mathcal{U} and M (a median filter) are of the same support, we can say that $\mathcal{L} \leq M \leq \mathcal{U}$, because of the different nature of the rank selection in each smoother.

3.4 Smoothers: \mathcal{UL} and \mathcal{LU}

Concatenations of \mathcal{L} and \mathcal{U} give \mathcal{LU} and \mathcal{UL} and can be proved to be *idempotent*, as done by Rhower. From $\mathcal{L}x \leq x \leq \mathcal{U}x$, it follows that $\mathcal{LU} \leq \mathcal{U}$ and $\mathcal{UL} \leq \mathcal{L}$ and therefore:

$$\mathcal{U} > \mathcal{LU} \geq M \geq \mathcal{UL} \geq \mathcal{L},$$

i.e. \mathcal{LU} and \mathcal{UL} are narrower bounds on M than \mathcal{U} and \mathcal{L} . In practice, either \mathcal{LU} or \mathcal{UL} is used as a smoother or the average of the result of both smoothers is used.

Rhower suggests that the smoothers should be implemented successively with increasing order, stopping at the required order. Suppose we have a discrete-time signal $x(n)$ that we would like to filter by a \mathcal{LU} - or \mathcal{UL} -smoother of order m . If we denote the smoother by \mathcal{J}'_m , the successive filtering can be expressed as:

$$y(n) = \mathcal{J}'_m \mathcal{J}'_2 \dots \mathcal{J}'_1 x(n), \quad (3.3)$$

and to simplify the notation we define:

$$\mathcal{J}_m \triangleq \mathcal{J}'_m \mathcal{J}'_2 \dots \mathcal{J}'_1. \quad (3.4)$$

Throughout the rest of the thesis we will refer to the successive process in equation 3.3 as a “LULU”-smoother of order m and denote it by \mathcal{J}_m . The actual smoother \mathcal{J}' can be \mathcal{LU} , \mathcal{UL} or a combination of them.

Consider Figure 3.4, part of the spectrum of a voiced segment taken from a singing voice recording. Because of the windowing, side-lobes appear on either sides of the main lobe, thus cluttering the spectrum. A “LULU”-smoother is applied to the spectrum in the hope of clearing it up from the windowing side-lobes.

Note the successful attenuation of the side-lobes and the trend preserving property of the “LULU”-smoother. The smooth spectrum’s local maxima form plateaus that, with very few exceptions, span the excitation peaks in the real spectrum. This immensely simplifies the task of extracting spectral information.

3.5 Summary

We described techniques for smoothing data that is corrupted by impulsive noise. These techniques are referred to as non-linear smoothers. The techniques are also used in digital image processing and are referred to as mathematical morphology.

We discussed the median filter as well as a nonlinear smoother called “LULU”, the latter of which is more computationally efficient and a bound on the median filter. We illustrated its application to a spectrum corrupted by the side-lobes of the windowing function.

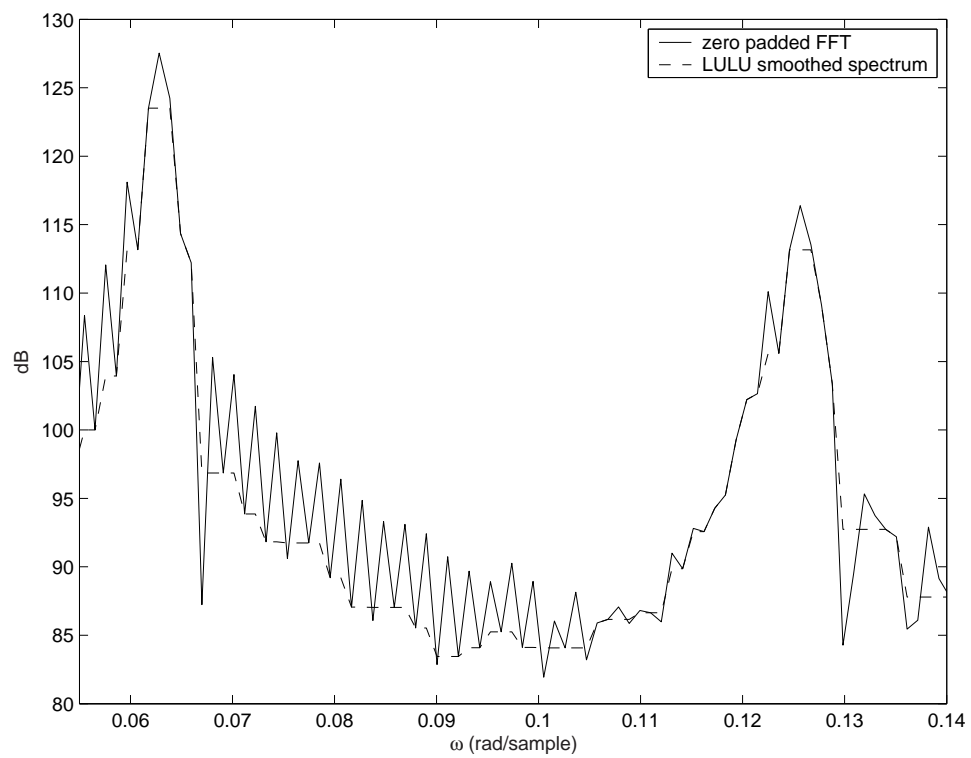


Figure 3.4: A zero padded spectrum, before and after smoothing

Chapter 4

The Phase Vocoder

4.1 Introduction

The representation of a signal in terms of its short-time Fourier transform can serve as a means of manipulating basic speech parameters. These parameters may include pitch, formant structure and speed of articulation. Systems that are based on such a representation are often referred to as phase vocoders, since magnitude and phase are the defining parameters. The “phase vocoder” was first introduced by Flanagan and Golden [4] in 1965. Their aim was to devise a way to encode speech so that communication bandwidth could be utilised more economically. As a side topic, they investigated the phase vocoder’s ability to stretch and compress the time duration of speech signals. In the words of the abstract paragraph of their original paper:

“A vocoder technique is described in which speech signals are presented by their short-time amplitude and phase spectra. A complete transmission system utilising this approach is simulated on a digital computer. The encoding method leads to an economy in transmission bandwidth and a means for time compression and expansion of speech signals.”

Schafer and Rabiner [19] took the phase vocoder to a next level by introducing a method based on the fast Fourier transform (FFT), greatly reducing the amount of computation needed. The first direct implementation of a digital phase vocoder was introduced by Portnoff [13] in 1976 and became a classic reference for future research.

Apart from its role in speech coding and transmission economics, the phase vocoder has found plenty of applications in music. The two best known applications are pitch-shifting and time-scaling. In this chapter we will discuss the latter of these methods, preceded by a detailed description of a unity system. By a unity system we mean that the input and output are theoretically identical and is equivalent to the system introduced by Portnoff [13]. We describe pitch-shifting in detail in Chapter 6.

4.2 A unity phase vocoder

Portnoff introduced a unity phase vocoder that represents a signal $x(n)$ in terms of its short-time phase and magnitude by means of short-time Fourier transforms, after which the short-time spectra are used to resynthesise the original signal. Here we develop a unity system that is, at a higher level, identical to that of Portnoff.

4.2.1 Analysis

We denote the successive *analysis* time-instants (where *analysis* windows start) by

$$t_a(u) \triangleq uR_a \quad (4.1)$$

where R_a is a fixed integer increment that controls the analysis rate. R_a is also referred to as the analysis *hop-size* and u is called the frame index. We can write the short-time Fourier transform, evaluated at discrete frequencies, as:

$$X(t_a(u), \Omega_k) = \sum_{n=0}^{N-1} h(n)x_w(t_a(u), n)e^{-j\Omega_k n} \quad k \in [0, N-1], \quad (4.2)$$

where

$$x_w(t_a(u), n) \triangleq x(t_a(u) + n) \quad n \in [0, L-1]. \quad (4.3)$$

$X(t_a(u), \Omega_k)$ is the short-time *analysis* spectrum of the signal at time $t_a(u)$, where $h(n)$ is a windowing function and where

$$\Omega_k \triangleq \frac{2\pi k}{N} \quad n \in [0, L-1]. \quad (4.4)$$

If we consider the discrete Fourier transform as a series of bandpass filters, then the values Ω_k are the centre frequencies of each band or “bin”. In practice, the STFT is calculated by means of a FFT of length N . The time domain window has a length of L where $L \leq N$. If $L < N$, the time domain windows must be “zero-padded” by adding a tail of zeros so that it has a length of N .

We can express $X(t_a(u), \Omega_k)$ in polar coordinates with magnitude M and phase ϕ :

$$X(t_a(u), \Omega_k) = M(t_a(u), \Omega_k)e^{j\phi(t_a(u), \Omega_k)} \quad (4.5)$$

Figure 4.1 illustrates the STFT principle. We refer to such short-time spectra as *analysis* instants.

This STFT procedure, concerning a phase vocoder, is generally known as *analysis* and gives us perspective on the signal in both time and frequency and is therefore an ideal tool to change frequency or time parameters. Since we are discussing an unity system, no modifications are performed. This may seem like a redundant exercise, but once we can move from the time domain to short-time spectra and back again, we could introduce frequency domain modifications before *synthesis*. Therefore the unity phase vocoder serves as a foundation for all our singing-parameter modifications.

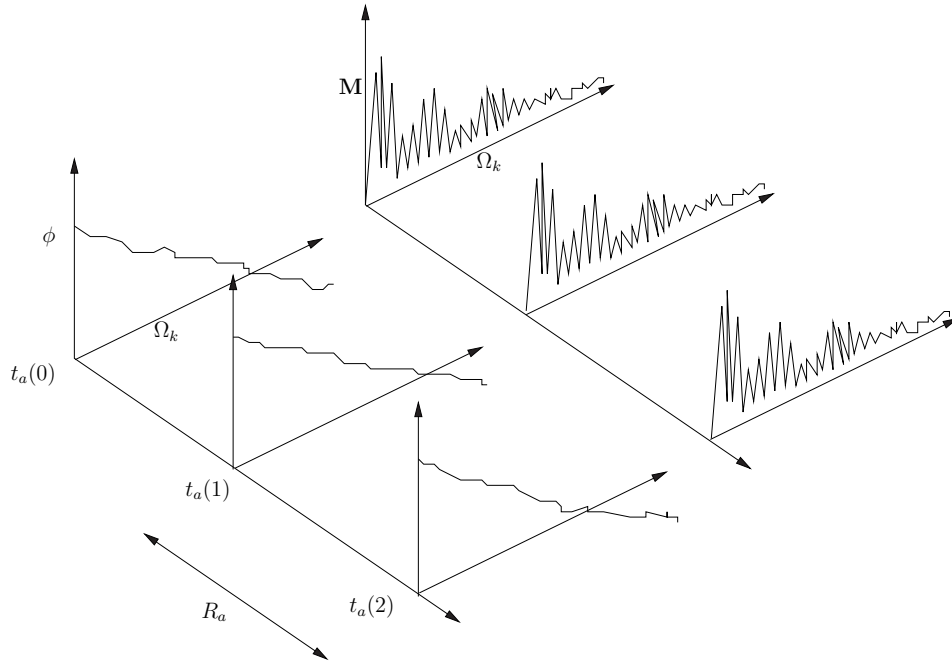


Figure 4.1: Short-time Fourier transform illustration

4.2.2 Synthesis

Synthesis is the process of combining the short-time spectra in order to return to the time domain. Figure 4.2 illustrates the process. Each short-time spectrum goes through a synthesis process in order to construct the desired signal $y(n)$. Short-time spectra used for synthesis are called *synthesis instants* and are denoted by $Y(t_s(u), \Omega_k)$ and may be expressed as:

$$Y(t_s(u), \Omega_k) = M(t_s(u), \Omega_k) e^{j\phi(t_s(u), \Omega_k)} \quad (4.6)$$

where $t_s(u)$ is called the *synthesis* time-instants and is defined as

$$t_s(u) \triangleq uR_s \quad (4.7)$$

where R_s is an integer called the *synthesis* hop-size. Moulines and LaRoche [12] prove that these *synthesis* instants can be combined by means of a weighted overlap-and-add procedure, giving a minimised square-error result implying ideal reconstruction. The overlap-and-add formula - or the *synthesis* formula - is:

$$y(n) = \frac{\sum_u y_w(t_s(u), n - t_s(u))}{\sum_u h(n - t_s(u))} \quad (4.8)$$

where

$$y_w(t_s(u), n) = \frac{1}{N} \sum_{k=0}^{N-1} Y(t_s(u), \Omega_k) e^{j\Omega_k n} \quad n \in [0, L - 1], \quad (4.9)$$

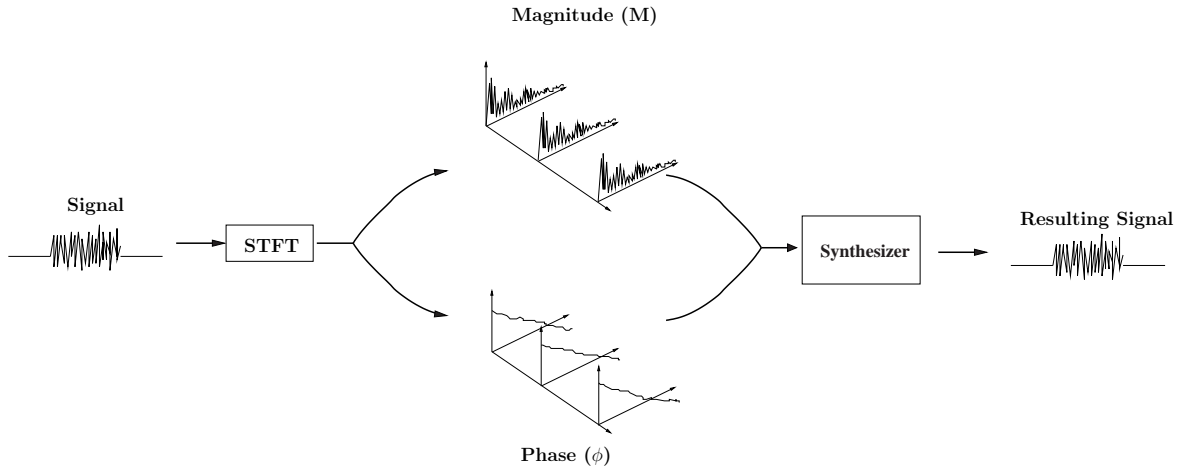


Figure 4.2: A unity phase vocoder

which is the inverse Fourier transform of the *synthesis* instants.

In the case of a unity system, where we do not induce modifications, perfect reconstruction of the original signal is possible, as long as overlapping time-domain windows are used. This concludes the mathematical derivation of a unity phase vocoder. Now we turn the attention to the window type.

4.2.3 Window choice

Typical weighting or windowing functions are the well known windows *Hamming*, *Hanning*, *Blackman* etc. Each of these have different spectral characteristics and must be selected carefully. We know that framing a signal with a rectangular window introduces unwanted noise in the spectrum, therefore windowing schemes with better signal-to-noise (S/N) ratios have been developed. Frequency resolution and S/N ratio are inversely proportional and most windows are the result of a payoff between these two extremes.

Windowing in the frequency domain is a convolution of the window's spectrum with the spectrum of the signal. Therefore, the spectrum of the ideal window is an impulse, i.e. a main lobe having zero width and side-lobes with zero amplitude. Table 4.1 provides these properties of popular windowing functions.

Table 4.1: Information on different window types

| Type of window | Approximate main-lobe width | Relative peak side-lobe (dB) |
|-----------------|-----------------------------|------------------------------|
| Rectangular | $4\pi/N$ | -13 |
| <i>Hanning</i> | $8\pi/N$ | -32 |
| <i>Hamming</i> | $8\pi/N$ | -43 |
| <i>Blackman</i> | $12\pi/N$ | -58 |

From the above table we conclude that *Hamming* and *Hanning* windows are best suited since they have narrow lobes compared to *Blackman* and have low side-lobes with low amplitudes compared to Rectangular windows.

4.3 Time-scaling

4.3.1 A simple approach

The simplest method of time-scaling is to evaluate a signal, say $x(n)$ sampled at F_s , at a different sample rate, say $\frac{1}{\alpha}F_s$, to get $y(n)$. α is a scaling factor by which the time duration of a signal will be scaled. The simplicity is attractive, but the artifacts of such a stretching/compression scheme are very significant.

Performing such an operation will cause the whole spectrum to stretch or compress by a factor of α . This is very undesirable, since a formant at ω_f would move to $\alpha\omega_f$ after the time-scaling. Formant drifting is a serious artifact that we can circumvent by taking a more involved approach towards time-scaling by employing the phase vocoder.

4.3.2 Using the phase vocoder for time-scaling

Background

The need very often arises to slow signals down (or speed them up) while leaving the harmonic content unchanged. As explained in section 4.3.1, time and frequency behaviour cannot easily be controlled independently.

If we have the signal presented by a phase vocoder, the tempo of the time domain behaviour can be changed by using a different hop-size for *synthesis* than used for *analysis*. If we want to scale the signal by a factor α , we calculate R_s as

$$R_s = \frac{1}{\alpha}R_a. \quad (4.10)$$

Each event taking place in $x(n)$ maps to a different time in $y(n)$, as in Figure 4.3. Applying the scaling factor α this way will reduce artifacts dramatically.

We perform no other modification to the signal and therefore the magnitudes of the *synthesis* instants are equal to that of the *analysis* instants:

$$|Y(t_s(u), \Omega_k)| = |X(t_a(u), \Omega_k)| \quad \text{for all values of } u. \quad (4.11)$$

The phases of the *synthesis* instants need attention. Since $R_a \neq R_s$, the phase between frames are no longer continuous because they no longer stand in a natural time relationship to each other. We need to force the phase of the original signal onto the synthesised one. A popular way to do this is to adjust the sinusoidal components of the first *synthesis* instant to take on the phases of the components in the first *analysis* instant. Now $x(n)$ and $y(n)$

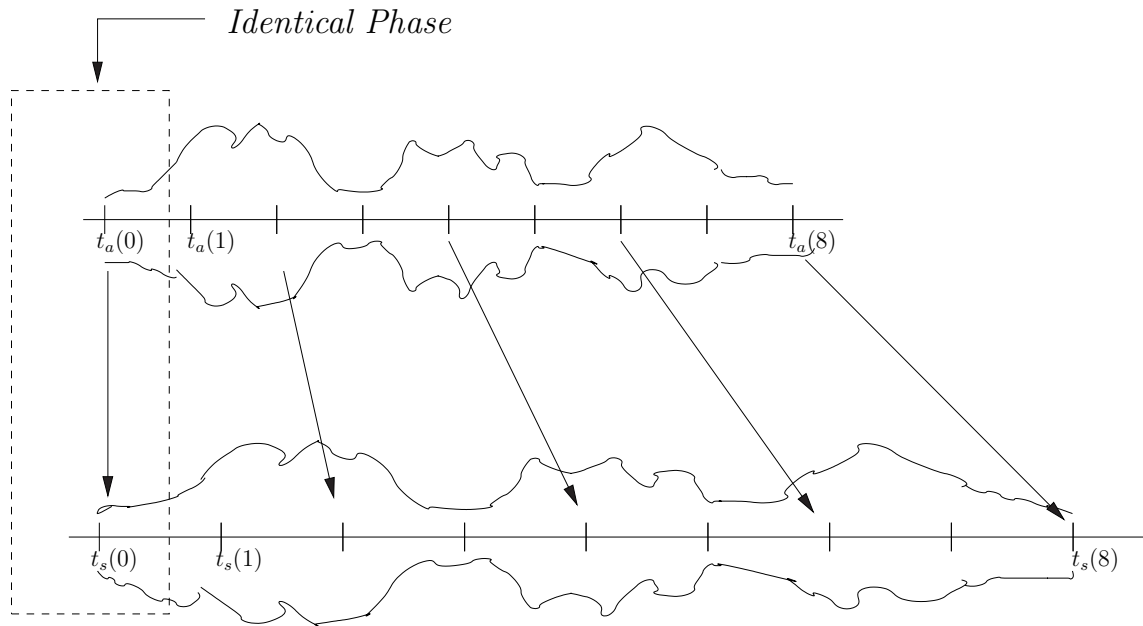


Figure 4.3: Mapping during time-stretching

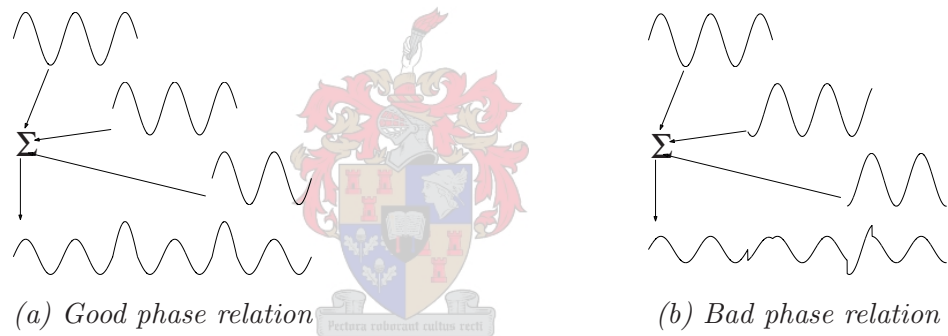


Figure 4.4: The importance of phase continuity

will start off with the same phase values for their sinusoidal components, as indicated by the rectangle in Figure 4.3.

Once the phases for the first *synthesis* instant is known, we can calculate the phases for the next instant by calculating phase advances. It is very important to have phase continuity between successive frames. The serious effect of bad phase relations when adding frames, is illustrated in Figure 4.4. Phase distortion, as it is labelled, is a very audible and disturbing artifact, especially in harmonically rich signals such as speech and singing.

To calculate the exact phase advances between the *synthesis* instants, we need to know the synthesis hop-size value R_s and the exact frequencies of all components present in the signal. (We assume that a single sinusoid is present for each value of Ω_k .) This exact frequency of a single component is called its *instantaneous frequency*. We will first show how to calculate the *instantaneous frequency* and then return to the phase problem.

Instantaneous frequency calculation

The FFT has a resolution of $1/F_s$, which can be increased by zero-padding, but still gives only average resolution which is not good enough to use for the calculation of the phase advance of a sinusoid over a time fraction. We can achieve very high resolution by calculating the *instantaneous frequency* of a sinusoid.

The phase $\phi(t_a(u), \Omega_k)$, carries information on the *instantaneous frequency* $\omega(t_a(u), \Omega_k)$, the exact frequency of a sinusoid in frequency-bin k in the *analysis* instant $X(t_a(u), \Omega_k)$. If the successive *analysis* time instants, $t_a(u)$ and $t_a(u - 1)$ are close enough so that equation 2.3 applies, we can determine $\omega(t_a(u), \Omega_k)$.

$\Delta\phi(t_a(u), \Omega_k)$ is defined as the phase increment of a sinusoid, falling in frequency bin k , between times $t_a(u)$ and $t_a(u - 1)$:

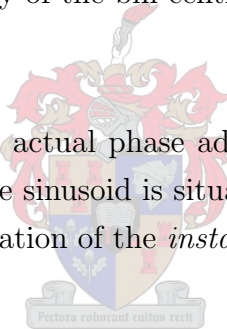
$$\Delta\phi(t_a(u), \Omega_k) \triangleq \phi(t_a(u), \Omega_k) - \phi(t_a(u - 1), \Omega_k). \quad (4.12)$$

We also define an expected phase advance, which is the phase advance a sinusoid would have if it had the exact frequency of the bin centre, Ω_k :

$$\Delta\phi^e \triangleq R_a \Omega_k. \quad (4.13)$$

Now, the difference between the actual phase advance and the expected phase advance gives an indication of how far the sinusoid is situated from the bin centre, Ω_k .

We can summarise the calculation of the *instantaneous frequency* as follows: (see [12] for a detailed proof)



1. Calculate the phase increment $Z(t_a(u), \Omega_k)$, the difference between the expected phase advance and the actual phase advance:

$$Z(t_a(u), \Omega_k) = \Delta\phi(t_a(u), \Omega_k) - \Delta\phi^e. \quad (4.14)$$

2. Modify the phase increment so that the result, \bar{Z} , lies between $-\pi$ and π , by adding and subtracting multiples of 2π .
3. Calculate the *instantaneous frequency* at time $t_a(u)$ in bin k , $\omega(t_a(u), \Omega_k)$,

$$\omega(t_a(u), \Omega_k) = \Omega_k + \frac{\bar{Z}(t_a(u), \Omega_k)}{R_a}. \quad (4.15)$$

The calculation of *instantaneous frequencies* is based on certain implicit assumptions [12]:

- For the STFT-analysis to be able to resolve individual pitch harmonics, the *analysis* window's bandwidth, ω_b , must be less than half the spacing between successive harmonics.
- The length of the *analysis* window, L , must be small enough so that the amplitudes of the *instantaneous frequency* of each harmonic can be considered constant for the duration of the window.

Phase Adjustment

Figure 4.4(b) shows the result of an overlap-and-add procedure for a single sinusoid where the three *synthesis* instants have bad phase relations among them. These phase relations are corrected in Figure 4.4(a), giving an undistorted result. We need to apply the same principle for signals containing many sinusoids.

Now that we know how to calculate the *instantaneous frequency* of any sinusoidal component, we can calculate the phase advance the component undergoes from the current instant to the next one. Knowing the values of the phases in the first *synthesis* instant and by calculating the phase advances, we can create a series of *synthesis* instants, $Y(t_s(u), \Omega_k)$, through a recursive process. These instants are identical to $X(t_a(u), \Omega_k)$ in magnitude but with phase values that make the synthesis of $y(n)$ possible. We initialise this recursive process by giving the first *synthesis* instant the same phase values as that of the first *analysis* instant. We provide a summary of the process in the next section.

Time-scaling algorithm

We summarise the steps of calculating $Y(t_s(u), \Omega_k)$, the *synthesis* instants needed to create $y(n)$ in Figure 4.5. The procedure is recursive, since the phase in the previous *synthesis* instant must be known to calculate that of the next.

Testing the algorithm

The inverse Fourier transform of the *synthesis* instants gives short-term time signals $y_w(t_s(u), n)$. We apply an overlap-and-add-procedure (equation 4.8) and calculate $y(n)$. We put the algorithm to the test by stretching an audio signal and by inspecting the spectrograms before and after stretching. The result should be two identical spectrograms, over different time durations. An inspection of the spectrograms in Figures 4.6 and 4.7 proves that the frequency content does not change, while stretching the signal. In other words, the pitch and the formant locations are constant. These results are similar to those obtained by Flanagan and Golden [4].

Not only is the algorithm theoretically successful, it is also very pleasing on an auditory level. The time-scaled results sound surprisingly natural. A signal can easily be scaled to

1. Transform $x(n)$ into a series of *analysis* instants, $X(t_a(u), \Omega_k)$
2. $u = 0$:
Initialisation: $\phi(t_s(u), \Omega_k) = \phi(t_a(u), \Omega_k)$
3. Increment u :
Compute the instantaneous frequency, $\omega(t_a(u), \Omega_k)$ for each bin, using equations 4.12 to 4.15. (Step through k)
4. Compute $\phi(t_s(u), \Omega_k)$ for each bin (step through k):
$$\phi(t_s(u), \Omega_k) = \phi(t_s(u-1), \Omega_k) + R_s \omega(t_a(u), \Omega_k)$$
5. Compute $Y(t_s(u), \Omega_k)$, the *synthesis* instant (step through k):
$$Y(t_s(u), \Omega_k) = |X(t_a(u), \Omega_k)| e^{j\phi(t_s(u), \Omega_k)}$$
6. Return to step 3 until the maximum value for u is reached
7. Calculate a series of short time signals $y_w(t_s(u), n)$ through equation 4.9
8. Synthesise $y(n)$ through equation 4.8

Figure 4.5: Time-scaling algorithm

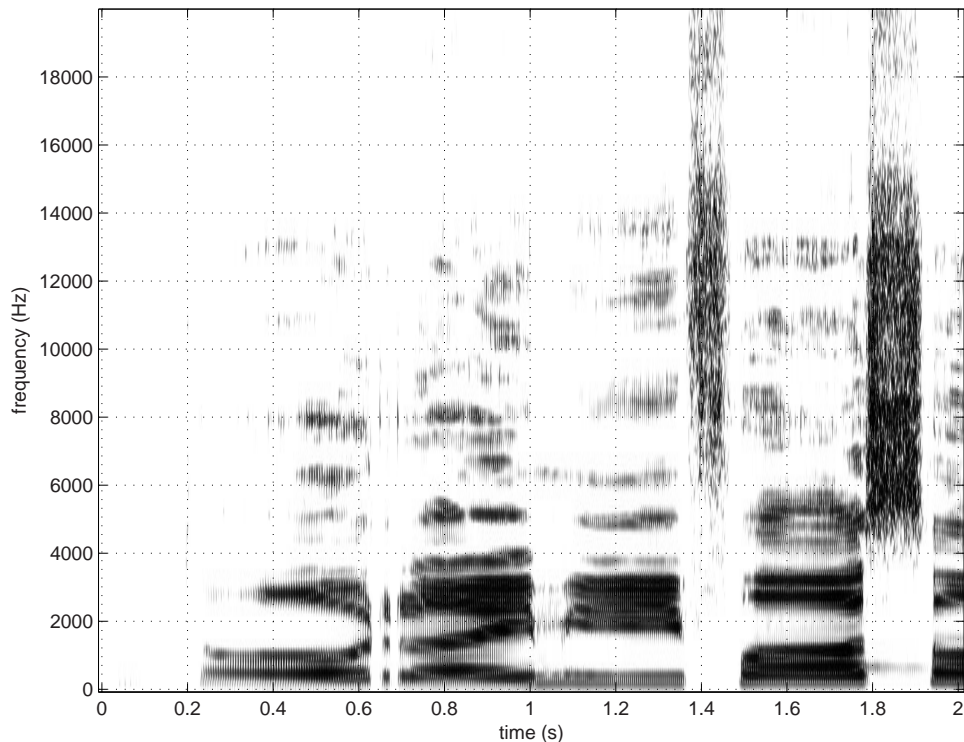


Figure 4.6: Spectrogram of an original audio signal

150% its original length and will fool any unsuspecting listener.

(The reader may refer to the included HTML document. The link to Chapter 4 examples provides audio examples of the phase vocoder’s time-scaling abilities.)

4.3.3 Applications of time-scaling

The time-scaling method we described is used widely in the music industry because of its remarkable ability to retain spectral behaviour while compressing or expanding the time duration. A further property is that part of the original signal can be modified in duration and the result can be seamlessly integrated with the rest of the original signal. Typical applications are rhythm matching and phrase training.

Rhythm matching is a handy tool when a music producer has two audio tracks that are out of synchronisation because of bad timing in the one track. Bad timing is the cause of incorrect rhythm for parts of the audio signal resulting in events taking place before or after they should. The phase vocoder can be used to delay or accelerate a given event to match that of the correct signal. The correct signal can be a MIDI track, which is machine generated and therefore has perfect rhythm. A human generated track does not have perfect rhythm, but the parts that are audibly out of synchronisation with the MIDI track can be fixed with a phase vocoder based program.

Phrase training is a musical exercise by which musicians master a specific note pro-

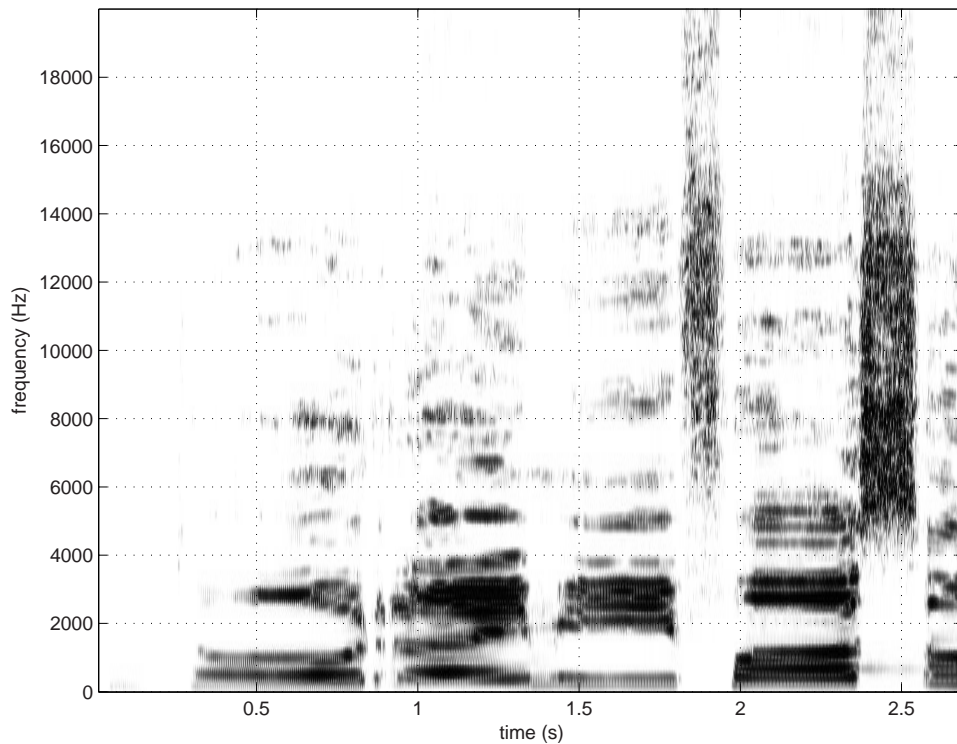


Figure 4.7: Spectrogram of a time-scaled audio signal

gression. The progression, or phrase, is repeated until the musician is satisfied. This can be a daunting task when he or she has no sheet music and has to learn the progression by listening to an audio track. Furthermore, if the progression is very fast it becomes a cumbersome routine of listening again and again. A phase vocoder based program can record the phrase and repeat it at a speed the musician finds comfortable.

4.4 Summary

We discussed the working of a phase vocoder, which transforms an audio signal into a representation giving both time and frequency information. If no modifications are performed on the signal, it may be synthesised to give a result that is identical to the input. This type of representation makes it possible to perform frequency or time modifications (or both) to the original signal. In this chapter, we described how to scale the time duration of an audio signal. The result is a high quality signal, identical to the input in frequency behaviour, but of a different time duration. We concluded with examples of typical uses for time-scaling.

Chapter 5

Pitch Detection

5.1 Introduction

Pitch is the perceptive fundamental frequency. The human ear perceives frequency as the spacing between the harmonics in a signal, rather than the lowest harmonic related sinusoid denoted by F_0 . F_0 and the pitch will have the same value in most cases, but there are times when F_0 is missing or very weak. In such circumstances F_1 is often mistaken for the pitch, while the true pitch might be $\frac{F_1}{2}$.

The aim of this chapter is to design an efficient frequency domain pitch-tracking algorithm, using a discrete signal $x(n)$ representing a singing voice. The result is a sequence of frequency values - or a pitch-track - denoted by $P_x(t_p(u))$. The subscript x indicates the signal from which the pitch-track is derived and $t_p(u)$ are the time instants where the pitch of $x(n)$ is calculated.

This is a short-term analysis problem and we are forced to make quasi-stationary assumptions about $x(n)$. We know that speech and singing are mixtures of periodic, aperiodic and stochastic signals but we assume that they are stationary over a short time interval. This interval is typically 10 – 30 milliseconds for speech but can be extended for singing since a vocalist keeps his or her pitch constant to achieve musical notes.

For pitch detection, we rely on the sliding window principle: $x(n)$ is windowed into a series of frames that overlap and that have constant lengths of value L . L should be chosen so that the frame is short enough for the signal to be stationary across its length, but long enough so that the signal properties can be measured, i.e several pitch periods long. The start of the overlapping regions are determined by the pitch hop-size, R_p , so that

$$t_p(u) = uR_p. \quad (5.1)$$

We found it convenient to express the time signal as short-time spectra, like in the *analysis* stage of a phase vocoder. Each *analysis* instant is the Fourier transform of a sliding window $x_w(t_p(u), n)$ with a spectrum $X(t_p(u), \Omega_k)$. The pitch we calculate from a

short-time spectrum $X(t_p(u), \Omega_k)$, represents the pitch for the duration of the window. In the next two sections we will show how to calculate the pitch for a single frame. Forgetting time for a moment, we consider a spectrum $X(\Omega_k)$.

(Although not discussed in this chapter, autocorrelation based pitch detection techniques are quite popular for the simplicity and robustness. Appendix D gives a high level overview of such techniques. For more detail on ACF the reader may consult [3].)

5.2 Harmonic extraction

The excitation manifests as a set of harmonic related impulses in the spectrum, spaced F_0 apart. See Figure 2.7 (p16) for an example. Pitch detection in the frequency domain consists of determining the frequency spacing between harmonics. The first task is to extract harmonics from the magnitude spectrum $|X(\Omega_k)|$.

There exists well documented techniques on peak extraction as described in papers on phase vocoders such as [8] and [7], where a peak is defined for a discrete magnitude spectrum as any value for $|X(\Omega_k)|$ that is larger than any of its two neighbours on both sides. This criterion does not eliminate peaks caused by an underlying sinusoid caused by the analysis window's side-lobes [9]. We introduce a peak extraction technique aiming to eliminate these “possible confusions” as they are labelled in [9].

In order to extract harmonics reliably, we need to filter the spectrum so that the only remaining local maxima are plateaus spanning the fundamental or any of its harmonics. Once we have the locations of these plateaus, we can simply find the local maximum that is spanned by the plateau. With a high degree of certainty, it can be said that these peaks are the excitation harmonics. Figure 5.1 shows a harmonic peak in a spectrum overlaid by an ideal filtered version.

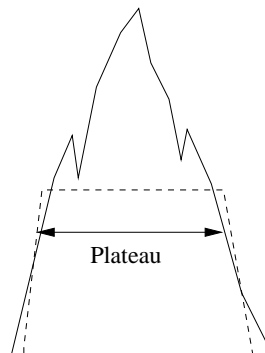


Figure 5.1: Harmonic peak in a spectrum, overlaid by an ideal filtered version

Suppose we have a square windowed frame of singing or speech of length L and zero-pad it to length N , with $r = \frac{N}{L}$ any sensible integer, i.e not bigger than e.g. 10. The

FFT of this frame produces a spectrum of length N . We know that a square window has very noisy side-lobes in the frequency domain (the peak side-lobe is only 13dB down), but they are also narrower than the main lobe and can therefore be filtered out. The side-lobe width¹ is r , and therefore we employ a *LULU*-filter of order r , \mathcal{J}_r . This filter will suppress any peak with a width of r or narrower and should remove the side-lobes from the spectrum as illustrated in Figure 3.4 (p25).

Figure 5.2 demonstrates a successful smoother; however, this is not necessarily the case since there may be spurious peaks that have widths wider than that of the standard side-lobes. We could provide for this possibility by raising the filter order, but it is important to note that the width of the plateau is proportional to the order of \mathcal{J} . If the plateau becomes larger, the area in which the peak should be found also increases.

This is a situation that asks for a play-off between two extremes:

- a low filter order for \mathcal{J} , meaning a narrow bound on peaks with the possibility of spurious peaks, being spanned by a “false” plateau.
- a higher filter order, meaning a wide bound on peaks with very low possibility of spurious peaks being spanned by a plateau.

Another parameter to consider is computational cost, since it becomes very high for large filter orders because of the concatenation in equation 3.3 (p24). Even though a second order filter succeeds for $\frac{N}{L} = 2$, we will typically raise the order to provide a safety margin.

We express the filtering process as:

$$\mathcal{J}_q|X(\Omega_k)| \quad \text{Pectus roburant cultus recti} \quad (5.2)$$

where q is the filter order.

We introduce a symbolic operator \mathcal{P} , a peak extractor. $\mathcal{P} \cdot \mathcal{J}_q|X(\Omega_k)|$ signifies the extraction of peaks in $X(\Omega_k)$ that are spanned by the plateaus in $\mathcal{J}_q|X(\Omega_k)|$.

5.3 Pitch calculation

Once we have the extracted harmonics we calculate the spacing between them, thus calculating the pitch. We define the sequence of extracted harmonics as:

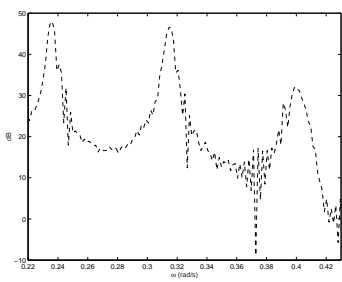
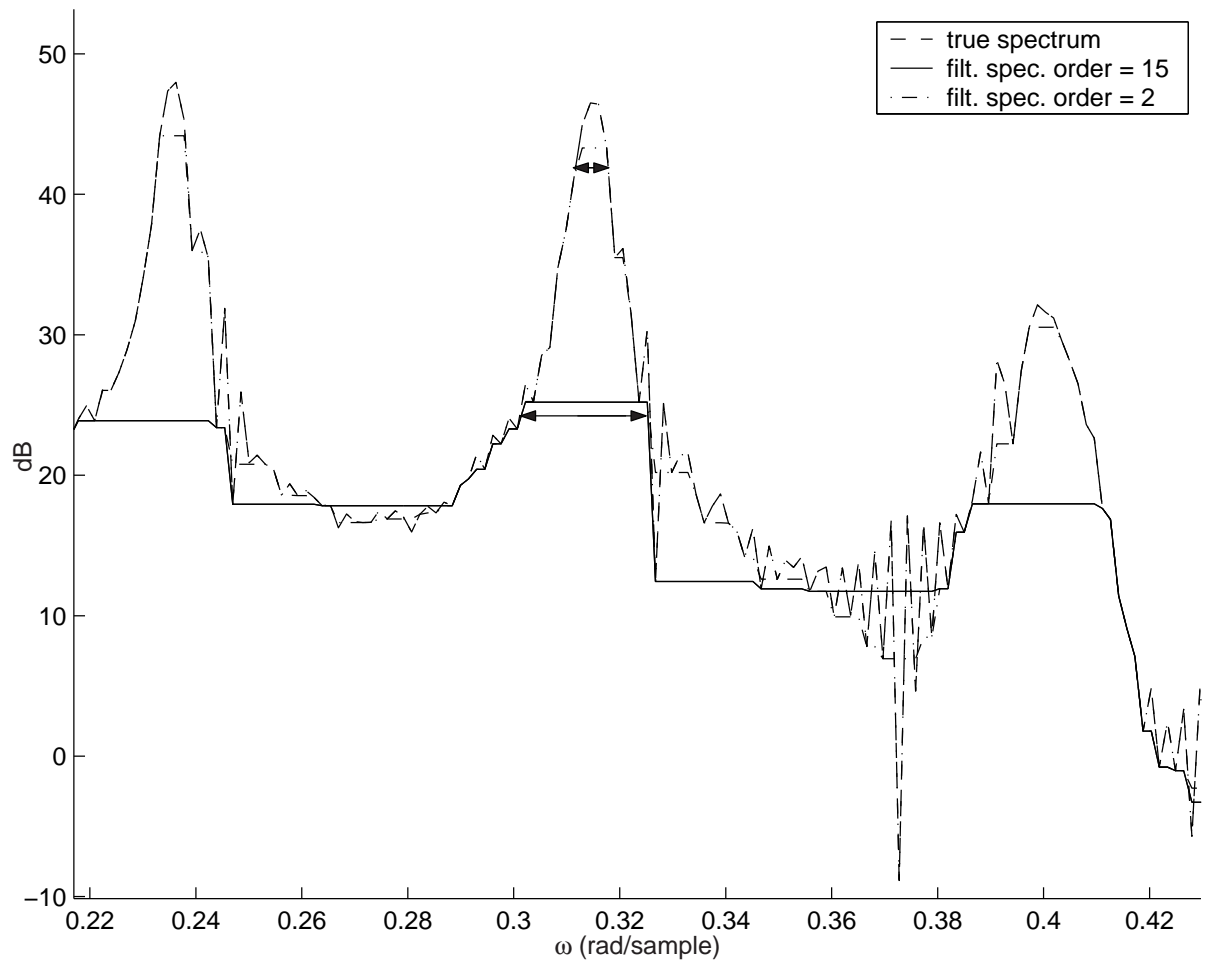
$$\mathcal{H}'_n \triangleq \mathcal{P} \cdot \mathcal{J}_q|X(\Omega_k)|, \quad (5.3)$$

where \mathcal{H}'_n is the frequency location of the n -th extracted peak.

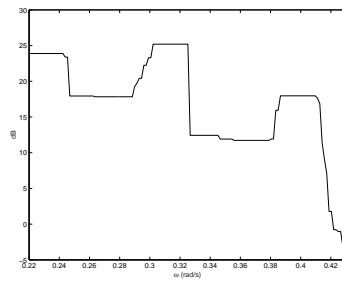
The FFT can be viewed as a series of bandpass filters with bandwidth

$$\omega_b = \frac{2\pi}{N}, \quad (5.4)$$

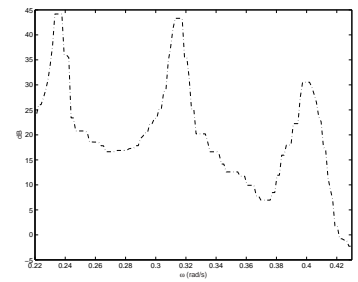
¹The width is given in samples.



True spectrum



*Filtered spectrum:
order = 15*



*Filtered spectrum:
order = 2*

Figure 5.2: Filtering at two different orders

meaning that a peak contained in \mathcal{H}'_n can be the result of a sinusoid with a frequency that can deviate from the peak location up to half the bandwidth ω_b . To calculate the pitch accurately, we need to know the exact frequencies of the sinusoid responsible for the peaks in $|X(\Omega_k)|$. There are two possible ways to solve the problem:

- We can zero-pad the short-time signals before Fourier transforming them. This will increase the frequency resolution by the same integer factor that the short-time signal is elongated. We can get significant frequency resolution increase, but computational cost becomes very high for large values of N .
- We can calculate the *instantaneous frequency* as described in Chapter 4, giving extremely high frequency resolution.

The latter of these solutions are preferable since it outperforms zero-padding in terms of resolution and computational cost. We found that a combination of the two methods is also a good choice. First the resolution is increased by a small integer factor $\frac{N}{L}$ after which the *instantaneous frequencies* of the peaks are calculated. The series of instantaneous frequencies are denoted by \mathcal{H}_n and is a correction of \mathcal{H}'_n . Since this method of pitch detection is dependent on the phase advance between two successive frames, it is based on the implicit assumption that a sinusoid producing a peak in one frame, produces a peak in the same bin in the next frame. This is a strict quasi-stationary assumption, since we expect two successive frames to be identical in terms of harmonic peaks. When transients occur between frames, the instantaneous frequency cannot be calculated. It is useful to check for transient errors in a post-processing stage.

If the peak picking stage was perfect, we could calculate the pitch by simply subtracting successive harmonics:

$$\mathcal{H}_{n+1} - \mathcal{H}_n. \tag{5.5}$$

Although we have confidence that most harmonics will be extracted, we cannot be certain that:

- F_0 is included in \mathcal{H}_n ;
- all the harmonics are extracted; and
- every value in \mathcal{H}_n is a harmonic, since it may include spurious peaks.

For these reasons we have to use a more robust approach than equation 5.5 to calculate the harmonic spacing. We base the approach on a method called harmonic summing.

To start off, let us assume F_0 is included in the sequence \mathcal{H}_n . We seek the value for n that maximises the equation²

$$\sum_{q=1} \left| X(\mathcal{H}_n \times q) \right|. \quad (5.6)$$

This means we treat every element in \mathcal{H}_n as F_0 and sum the magnitudes of the spectrum where its harmonics should occur. If a specific element is the fundamental frequency, equation 5.6 should be maximised. We can state that $F_0 = \mathcal{H}_{k_0}$ if

$$k_0 = \operatorname{argmax}_n \sum_{q=1} \left| X(\mathcal{H}_n \times q) \right|, \quad (5.7)$$

where q runs from 1 up to the maximum number of harmonic locations allowed by the sampling rate. However, this is true only if F_0 is extracted and included in \mathcal{H}_n . In some cases, especially when the recording bandwidth is limited, \mathcal{H}_n may not include the fundamental. If we assume that the first harmonic, F_1 , is included in the list of peaks, we can repeat equation 5.7 with each value in \mathcal{H}_n assumed to be F_1 , and therefore divided by two to get the fundamental:

$$k_1 = \operatorname{argmax}_n \sum_{q=1} \left| X(\mathcal{H}_n \times \frac{q}{2}) \right|. \quad (5.8)$$

The evaluation of equation 5.8 serves as a safety net for equation 5.7. We can say that $F_0 = \frac{1}{2} \times \mathcal{H}_{k_1}$ under two conditions:

1. $k_0 = k_1$; a check to make sure that the only other possible option for the fundamental is one half of the first estimate. In other words the first estimate mistook F_1 for the pitch.
2. The total sum of the harmonics that maximises equation 5.8 should be more or less twice as high as that in equation 5.7, since we would expect twice as many harmonics:

$$\sum_{q=1} \left| X(\mathcal{H}_{k_0} \times q) \right| \approx 2 \sum_{q=1} \left| X(\mathcal{H}_{k_1} \times \frac{q}{2}) \right|. \quad (5.9)$$

This idea can be extended to a case with any number of lower harmonics missing, however it is in the first place unlikely for F_0 to be missing in high quality recordings and most unlikely for F_1 to be missing.

²Because \mathcal{H}_n are *instantaneous frequency* values, they do not necessarily correspond to the discrete frequency values, Ω_k . Therefore the value of Ω_k that is closest to $\mathcal{H}_n \times q$ is used in equations in the likes of 5.6 and 5.7.

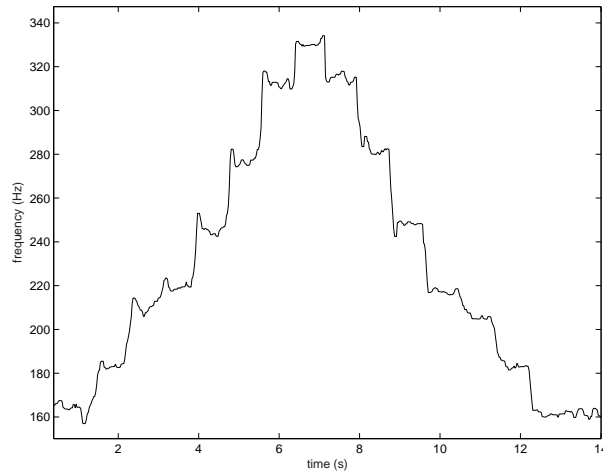


Figure 5.3: Pitch-track of a scale by a male vocalist

5.4 Constructing the pitch-track

By calculating F_0 of each instant $X(t_p(u), \Omega_k)$, we can construct a pitch-track $P_x(t_p(u))$, containing the fundamental frequencies of $x(n)$. The method explained above gives the pitch in units of radians/sample, and we convert it to Hertz by dividing by 2π and then multiply by the sampling rate.

5.5 Post-processing

Because of unpredictable transients between frames, we have to post-process the pitch-track to have a smooth contour representing the pitch movement of $x(n)$.

The pitch-track in Figure 5.3 is the result of a raw pitch-track filtered by a median filter of order 5. The “LULU”-smoother introduced in Chapter 3 is also very useful to produce a smooth pitch-track.

For certain applications it might be useful not to do post-processing, since abnormal high pitch values are good indicators of unvoiced areas or silence. (The vocalist that sung the scale in Figure 5.3 used vowels only.)

5.6 On silence

Sometimes the need rises to know when the signal we study is in the state of silence. If such a state occurs it is redundant to process that part of the signal. We devise a simple but effective way to label such regions and include the result into the pitch-track by giving $P_x(t_p(u))$ a value of zero if $x(n)$ is in a silence state at time $t_p(u)$.

The same frames used to calculate the short-time spectra in section 5.1 can be used to label regions as silent. For each frame we calculate the RMS power:

$$r(t_p(u)) = \sqrt{\sum_{n=0}^{L-1} x_w^2(t_p(u), n)}. \quad (5.10)$$

If K is the number of frames we average the RMS values to get m :

$$m = \frac{1}{K} \sum_{u=0}^{M-1} r(t_p(u)). \quad (5.11)$$

Empirically, we found that a hysteresis approach with two thresholds based on m gives satisfactory results. We devise a state machine that can be either in *non-silence state* or *silence state*:

- Moves from *non-silence state* to *silence state* when $r(t_p(u)) < 0.15m$. As long as this state occurs, $P_x(t_p(u))$ becomes zero.
- Moves from *silence state* to *non-silence state* when $r(t_p(u)) > 0.3m$. $P_x(t_p(u))$ keeps its original value for the duration of this state.

We do not introduce a new notation for pitch-tracks that were processed to “zero” the silences. We will mention it explicitly when this process is applied to a pitch-track.

Figure 5.4 shows how the RMS values of frames of $x(n)$ can be used to “zero” the pitch in areas of silence. To better indicate the entrance and exit of a *silent state*, we display an enlarged region of Figure 5.4(b) in Figure 5.5.

5.7 Summary

This chapter described the design of a robust pitch-tracking algorithm. The algorithm employs the “LULU”-smoother described in Chapter 3 and relies on the principle of harmonic summing. We can adapt the algorithm for band limited signals where the possibility exists that F_0 is missing, in which case the algorithm should still detect the correct pitch. It is further robust since it discards spurious spectral peaks such as those induced by windowing side-lobes or other unwanted noise.

We also gave a suggestion on how to detect silence in a signal. The silence detection relies on a hysteresis principle using RMS power calculations. It may be useful to know where silences occur, since it is redundant to process such parts of a signal.

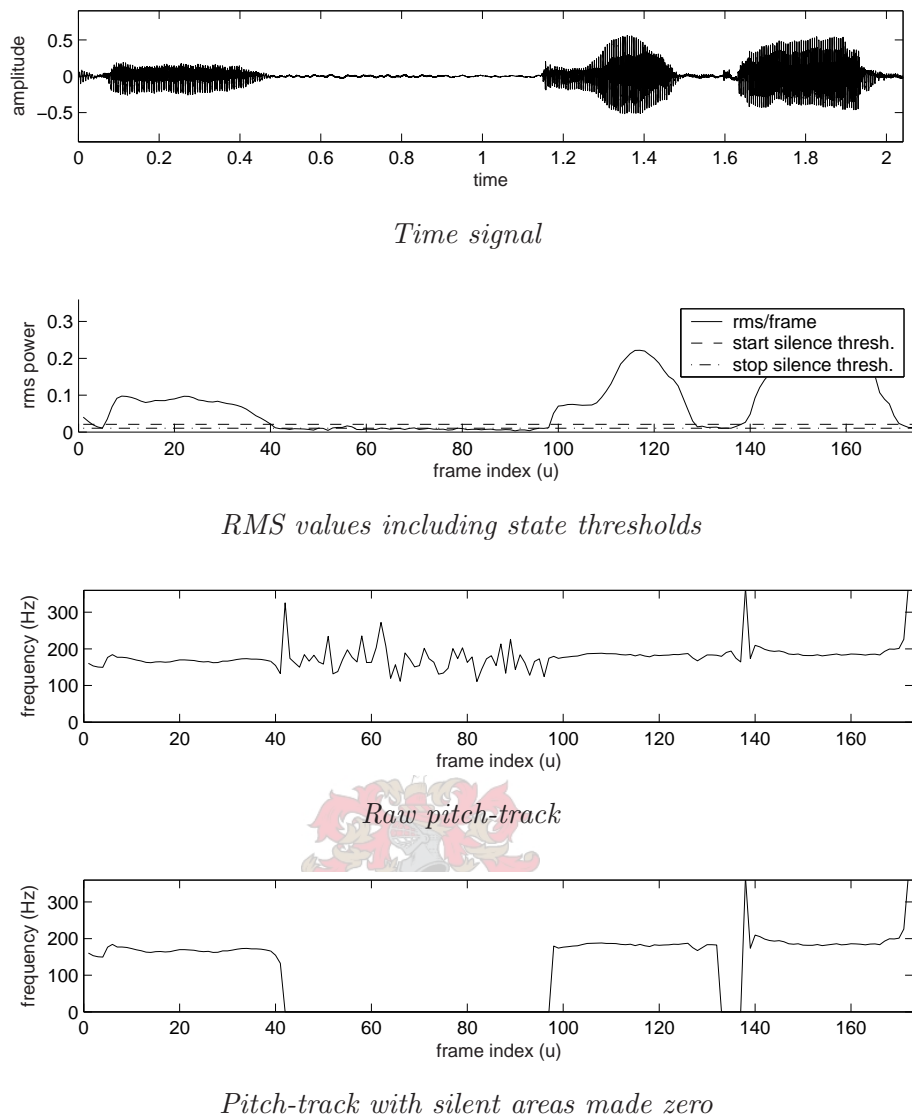


Figure 5.4: Silence detection and integration into a pitch-track

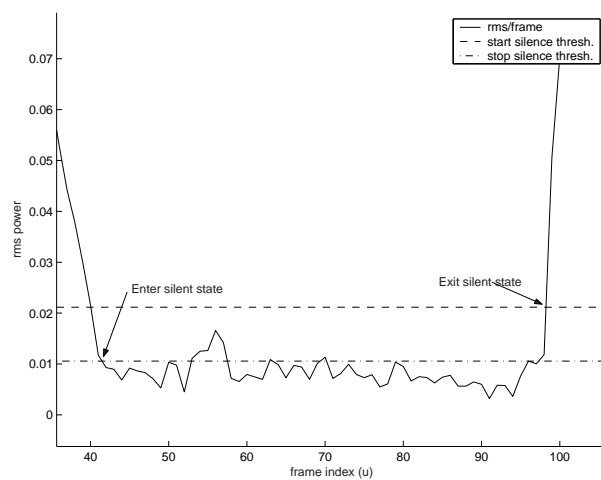


Figure 5.5: Entrance and exit of the silent state

Chapter 6

Pitch-shifting

NOTATION:

For this chapter only, we make a few slight notational changes.

In this chapter we will be considering a single short-time signal $x_w(t_a(u), n)$ and how to change its pitch. To simplify notation we will simply refer to it as $x(n)$ and its discrete spectrum $X(\Omega_k)$. The short-time pitch-shifted signal $y_w(t_a(u), n)$ is referred to simply as $y(n)$ with spectrum $Y(\Omega_k)$.

The excitation signal of $x(n)$ is denoted by $e(n)$ and has a spectrum $E(\Omega_k)$. The excitation signal of $y(n)$ is denoted by $e'(n)$ and has a spectrum $E'(\Omega_k)$.

6.1 Introduction: failure of an obvious approach

Pitch-shifting refers to a process that changes the fundamental frequency of a signal while leaving the spectral shape unaltered. In terms of a human voice, it means changing the pitch without moving the formants. The obvious way of changing pitch is to use the artifact of evaluating an audio signal at an incorrect sampling rate $\frac{1}{\alpha}F_s$, as done in section 4.3.1 (p30), where α is a constant. The fundamental frequency will change by a factor $\frac{1}{\alpha}$, but the formants will move. A formant that was centred around ω_f will now be centred around $\frac{1}{\alpha}\omega_f$. The formant drifting has been described as the “singing rodent” effect, which gives an indication of the seriousness of the artifact.

Figure 6.1 is the spectrogram of a voiced signal (a vocalist singing *do re me fa ...*). If we evaluate the same signal at $\frac{4}{3}F_s$, we can clearly see in Figure 6.2 that the formants moved higher with a factor of $\frac{4}{3}$ **and** the total signal is compressed in time with a factor of $\frac{3}{4}$. The fundamental frequency is at the desired frequency, but the artifacts are too serious to ignore. Therefore, this approach cannot be regarded a true *pitch-shifting* algorithm.

We conclude that a simple sampling rate change scheme is not sufficient. The rest of the chapter explains in detail how we can change the pitch of a single frame that will

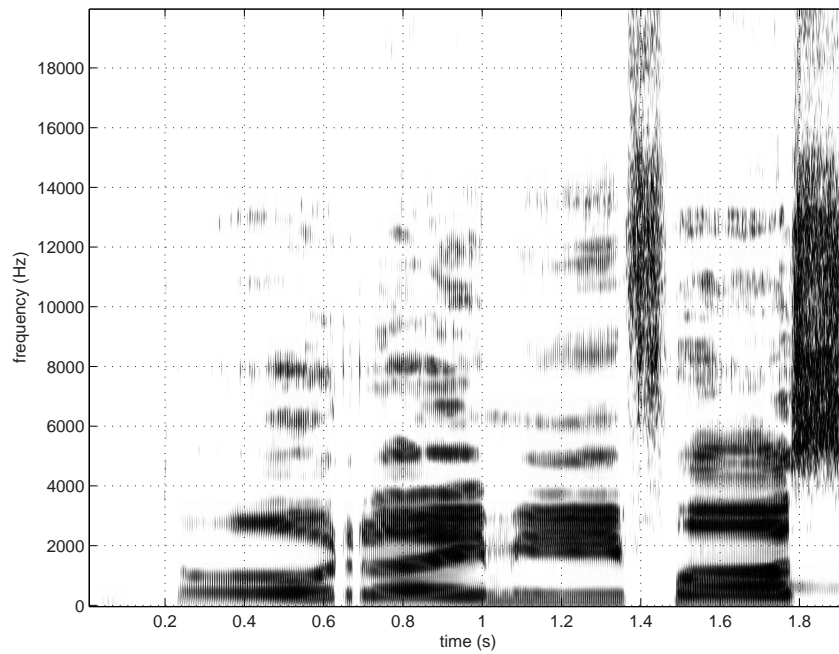


Figure 6.1: Spectrogram of a male singing voice

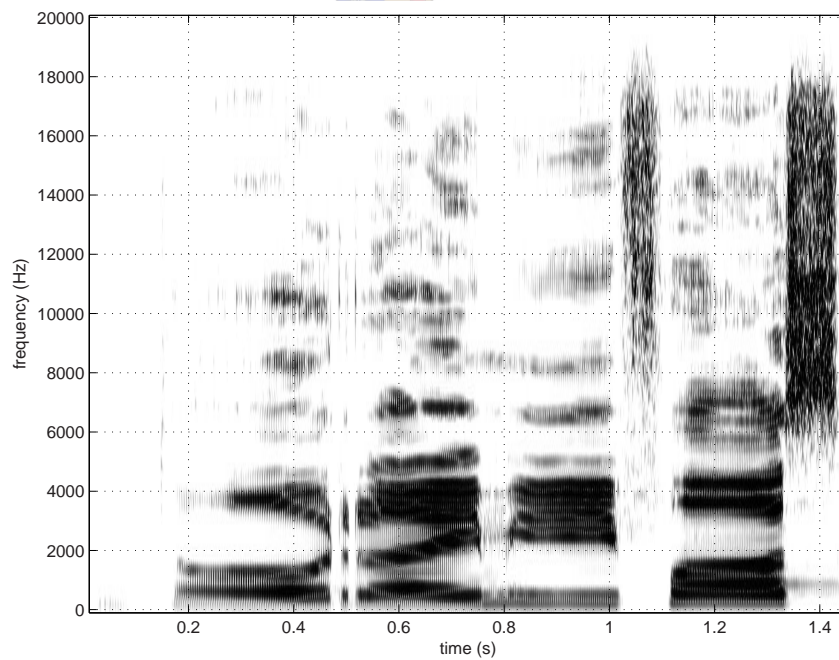


Figure 6.2: Spectrogram of a signal evaluated at $\frac{4}{3}F_s$

- theoretically - not introduce artifacts. In Chapter 7 we will combine the techniques described here with a phase vocoder to introduce a complete, frame-based, *pitch-shifter*.

6.2 Source-filter decomposition

6.2.1 Splitting the signal

In Chapter 2 we described the generally accepted model for speech production. Singing, from a signal processing point of view, can be seen as a time dependent filter driven by an excitation signal. If we can split the filter and the excitation for a single frame, we can manipulate the excitation signal $e(n)$ and apply the filter afterwards to restore the original resonances. We need not think of a filter in the *linear time invariant* sense, as we will shortly prove. To make the concept more generic, we use the term *model* to denote any set of parameters that describe the signal's behaviour in time or frequency.

An *inverse filter* can be derived from the model and be applied to the signal, thus removing the resonances, or formants, to produce the excitation. After the excitation has been modified, the same or a different type of model can be applied to restore the resonances of the original signal. By changing the pitch in this manner, we will end up with the formants in their original position, but excited by a different fundamental frequency. This technique theoretically delivers a true *pitch-shifting* algorithm. Figure 6.3 summarises the above. Note that *modelling technique A* and *modelling technique B* may be identical.

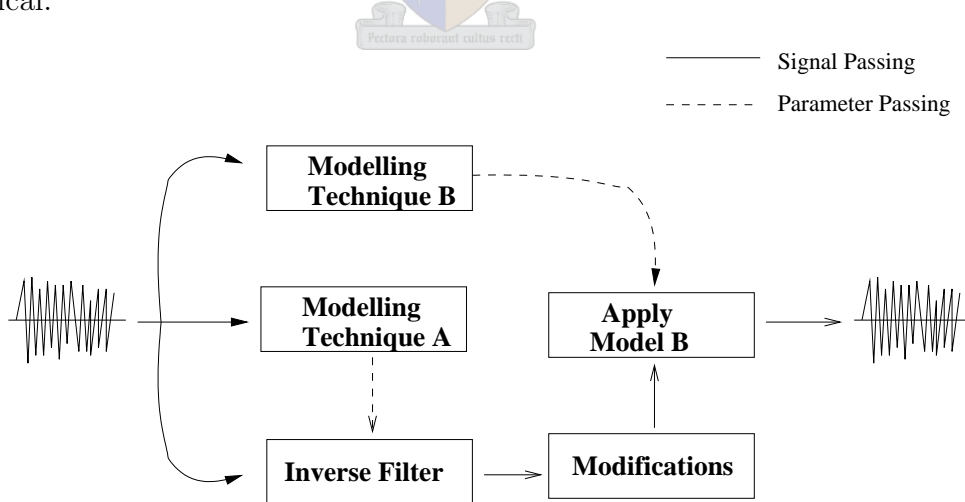


Figure 6.3: Diagram of the working of a pitch-shifter

We will discuss two modelling techniques, namely *linear prediction* and *direct-modelling*.

6.2.2 Linear Prediction-model

Through linear prediction analysis, we can estimate an all-pole filter from a frame, under stationary assumptions for the duration of the frame. The filter $H(z)$ will approximate the behaviour of the time signal, where

$$H(z) = \frac{G}{1 - \sum_{i=1}^p a_i z^{-i}} \quad (6.1)$$

where G is a constant gain term.

$H(z)$ can be viewed as the z -transform of the impulse response of the vocal tract $s_n(m)$ as described in section 2.5 (p2.5).

There exist several well documented techniques for the calculation of a_i , the filter parameters. The best known and a well used one is the autocorrelation method, of which the derivation is given in detail in Appendix A. This method is very attractive because the produced filter is always stable and requires no checking, as opposed to the covariance estimation procedure [3]. (See [10] and [3] for detailed discussions on linear prediction.)

We can determine the excitation signal by filtering the frame with its inverse filter, $H(z)^{-1}$, thus removing the formants, after which we normalise the spectrum. Figure 6.4 shows the lower part of the spectrum of a sung vowel (top figure), and the spectrum after inverse filtering, i.e. the excitation spectrum (lower figure). The formants were removed successfully, giving a spectrum with harmonic related peaks of near equal amplitude. The process of removing resonances from a signal is referred to as whitening, since it transforms the spectrum's envelope to a line, reminiscent of white noise spectra.

The advantage of isolating the excitation lies in the fact that there are no natural resonances in the spectrum, therefore it can be modified without the danger of formants that move. After the modifications are applied, the original envelope can be restored by re-applying the model $H(z)$. If the excitation spectrum is not altered, the original signal will be produced, since $H(z)^{-1} \times H(z) = 1$.

The quest for the optimum *LPC* order is somewhat intuitive in the literature. Moulines and Laroche[12] gives a crude figure and states that either 10 or 20 poles should be used, according to the sampling rate. Makhoul [10] gives more thought to the subject: the higher the number of poles, the better the signal fits the model. The question is where to stop. A normalisation error V_p , is introduced and proves useful for estimating the optimum filter order. (The subscript p indicates the amount of poles.) The normalisation error is defined as “the ratio of the minimum modelling error to the energy in the signal”. Makhoul proves that:

$$V_p = \frac{e^{\left(\frac{1}{2\pi} \int_{-\pi}^{\pi} \log_{10} \hat{P}(\omega) d\omega\right)}}{\frac{1}{2\pi} \int_{-\pi}^{\pi} \log_{10} \hat{P}(\omega) d\omega}, \quad (6.2)$$

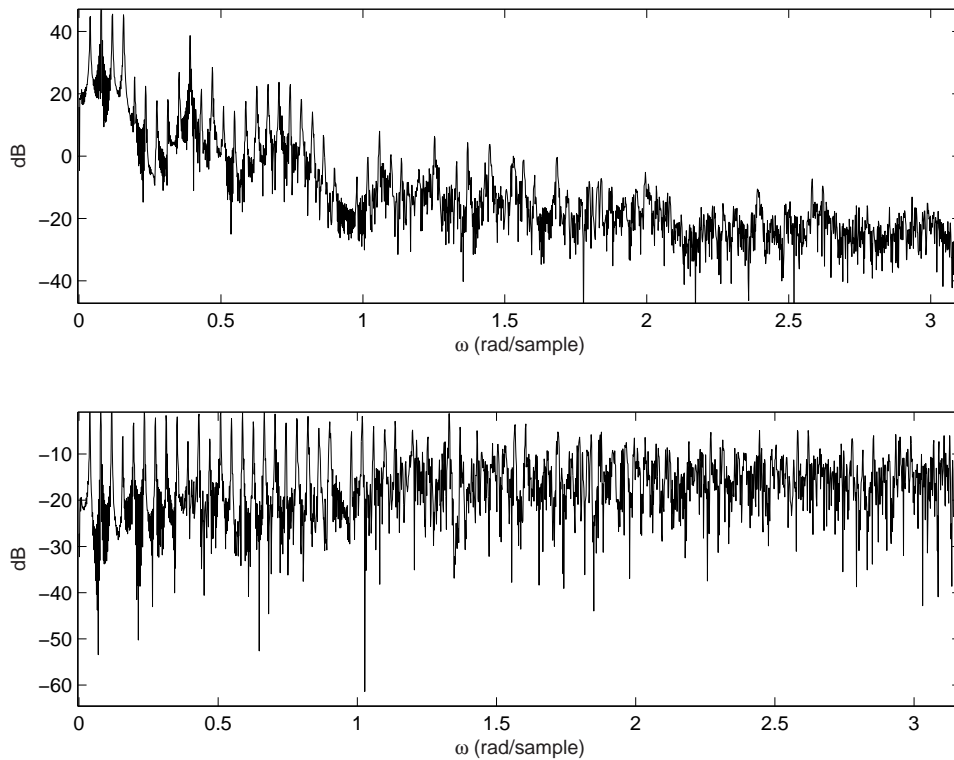


Figure 6.4: A spectrum before (top) and after (bottom) whitening

where $\hat{P}(\omega)$ is the estimated, or modelled, power spectrum. We can calculate this power spectrum by evaluating the square of equation 6.1 on the unit circle between $-\pi$ and π and substitute z for $e^{j\omega}$:

$$\hat{P}(\omega) = \frac{G^2}{\left|1 - \sum_{i=1}^p a_i z^{-i}\right|^2} \Big|_{z = e^{j\omega}} \quad (6.3)$$

If the spectrum that we try to estimate is that of an all-pole AR-process with p_0 poles, then V_p will become flat as $p \rightarrow p_0$. In the real world, we are seldom confronted by true AR-processes but a well controlled assumption that they are, can help us estimate the optimum number of poles. This number p can be calculated from the inequity

$$1 - \frac{V_{p+1}}{V_p} < \delta, \quad (6.4)$$

with δ a small number. If equation 6.4 holds true for several consecutive values of p , we can state that the curve has flattened out - indicating an optimal value for p .

Figure 6.5 shows V_p for several values of p , using the same time-domain signal used for Figure 6.4. The curve has flattened out after $p = 60$, and gives hardly any improvement after $p = 80$. We used an 80th order filter to produce the lower spectrum in Figure 6.4.¹

¹The sampling rate of the time signal was 44.1kHz.

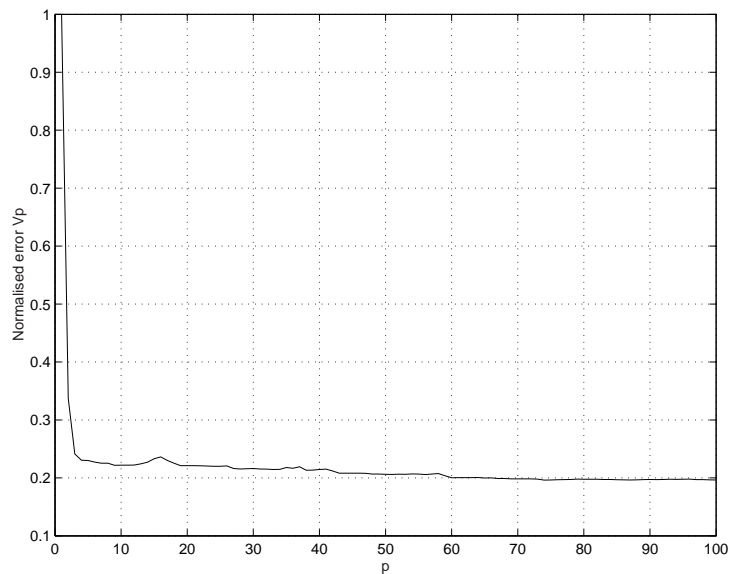


Figure 6.5: Normalised error V_p , vs number of poles

In practical applications an order of around 30 would be sufficient.

A new problem features when the order, p , becomes very high. The individual pitch harmonics are being modelled instead of the spectral envelope. It also features when a female sings very high, because the harmonic spacing becomes of the same order as the spacing between formants. This is called *over-modelling*. Therefore we should be careful when raising the order.

Figure 6.6 shows an *LPC* approximation at an optimal order, while Figure 6.7 shows a model of an order that is too high. Of course, in the second case V_p would also be flat but the modelling properties at high orders become less useful for formant analysis.

To summarise the answer to the question on how to determine the optimal number of poles:

1. Calculate the normalised error curve with equation 6.2.
2. See that equation 6.4 holds true for a number of consecutive values of p .
3. Make sure that the spectrum is not over modelled as in Figure 6.7. In general, V_p should be flat long before *over-modelling* becomes an issue.

We finish the discussion on Linear Prediction with a few notes on the gain term G in equation 6.1. G may be calculated so that the filter approximation includes the volume of the signal it was derived from. Along with the derivation of the linear prediction

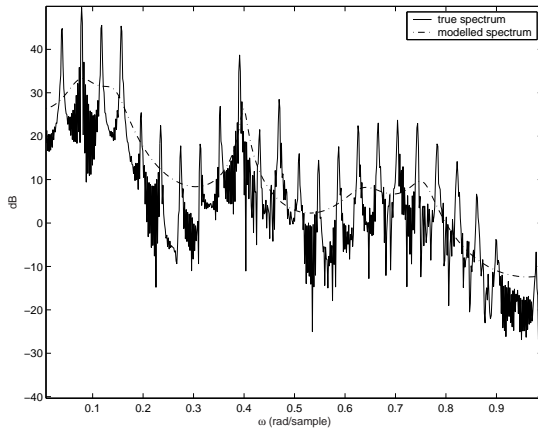


Figure 6.6: LPC spectral approximation at an optimal order: $p = 30$

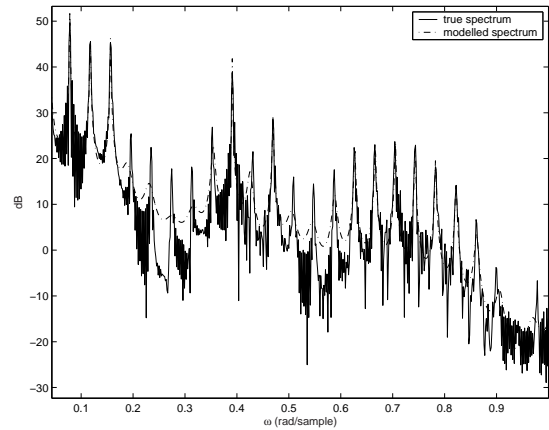


Figure 6.7: *LPC* over modelling: $p = 150$

coefficients a_i , found in equation 6.1 through the autocorrelation method, we can calculate the power of the prediction error - of which the square root is used for G . (See Appendix A for details.)

When we inverse filter a series of short-time signals (to get excitation signals) using equation 6.1 with G calculated as above, all the excitation signals will have identical power. If a different model is used to restore the formant locations, that model must not only provide the formant locations but also the correct gain.

If we wish to keep the volume of a short-time signal in its excitation signal, G must simply have a unity value during inverse filtering and the model used to restore the formants should therefore also have unity gain.

6.2.3 Direct-model

Introduction

In section 6.2.2 we discussed a time domain method to model the behaviour of a frame of speech or singing. We now introduce a method to model the frequency behaviour directly from the spectrum, hence the term *direct-model*. The idea is to estimate the magnitude spectrum's envelope by interpolating the peaks formed by the excitation harmonics. We divide this modelling technique into three schemes: filtering, peak extraction and interpolation. The result is a smooth curve that approximates the spectral magnitude envelope.

Filtering and Peak Extraction

In Chapter 5 we described in detail how to extract peaks from a magnitude spectrum after being smoothed by a “*LULU*”-filter. We repeat the symbolic equation used to extract

the peaks:

$$\mathcal{H}'_n = \mathcal{P} \cdot \mathcal{J}_q |X(\Omega_k)|. \quad (6.5)$$

\mathcal{H}'_n is a series of the locations of prominent peaks in the magnitude spectrum.

Interpolation

Once we know the amplitudes and locations of the spectral maxima, we can interpolate them to give a smooth continuous approximation of the magnitude spectrum. We denote this magnitude approximation by $\widehat{M}(\omega)$.

We take a quick detour to explain the interpolation process. A well documented form of interpolation is the so-called cubic B-spline, which finds application in computer graphics. It is used to give animated characters a smooth human appearance and to create smooth landscapes like the ones found in computer games. Four control points, called the knot sequence, are needed to draw a smooth curve near the middle two points. This curve will not pass through any points in the knot sequence and is a third order polynomial that can be expressed as a matrix equation:

$$P(t) = [1 \quad t \quad t^2 \quad t^3] M \begin{bmatrix} P_k \\ P_{k+1} \\ P_{k+2} \\ P_{k+3} \end{bmatrix} \quad (6.6)$$

where t can be swept between 0 and 1, inclusive, at any resolution and where

$$M = \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ 1 & 3 & -3 & 1 \end{bmatrix} \quad (6.7)$$

and $P_k \dots P_{k+3}$ is the knot sequence. Figure 6.8 shows a knot sequence with its associated B-spline curve. In this case P_k, \dots, P_{k+3} are vectors of order 2, in order to denote Cartesian co-ordinates. But they can be of any order since each dimension gets interpolated independently.

In practice, a complex interpolation with a long knot sequence is done piecewise, selecting four points at a time and calculating the B-spline curve. The index, k , is then increased by one to calculate the next piecewise curve. A simple but elegant way to force the smooth curve to finish and start at the same points as the knot sequence, is to simply repeat the first and last entries in the sequence. (If the smooth curve must pass through any point in the knot sequence that is not the first or last one, that point must be repeated three times.) Figure 6.9 shows the piecewise interpolation of a complicated knot sequence.

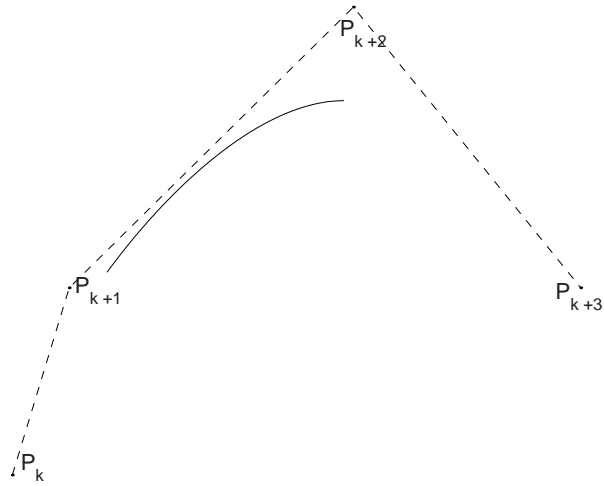


Figure 6.8: A single B-spline segment

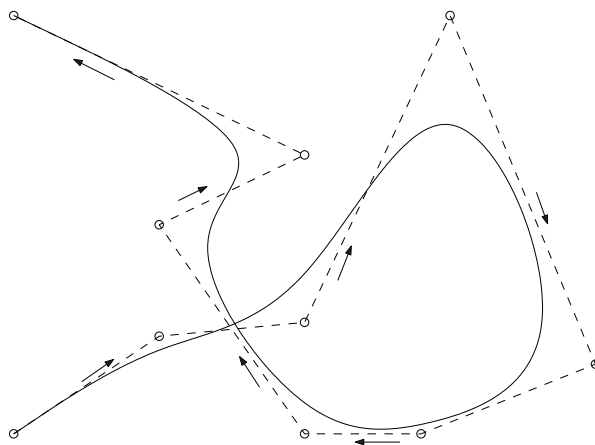
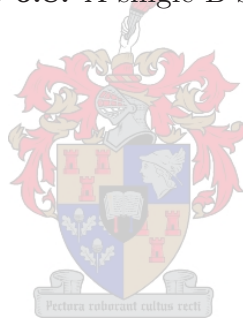


Figure 6.9: Cubic spline demonstration

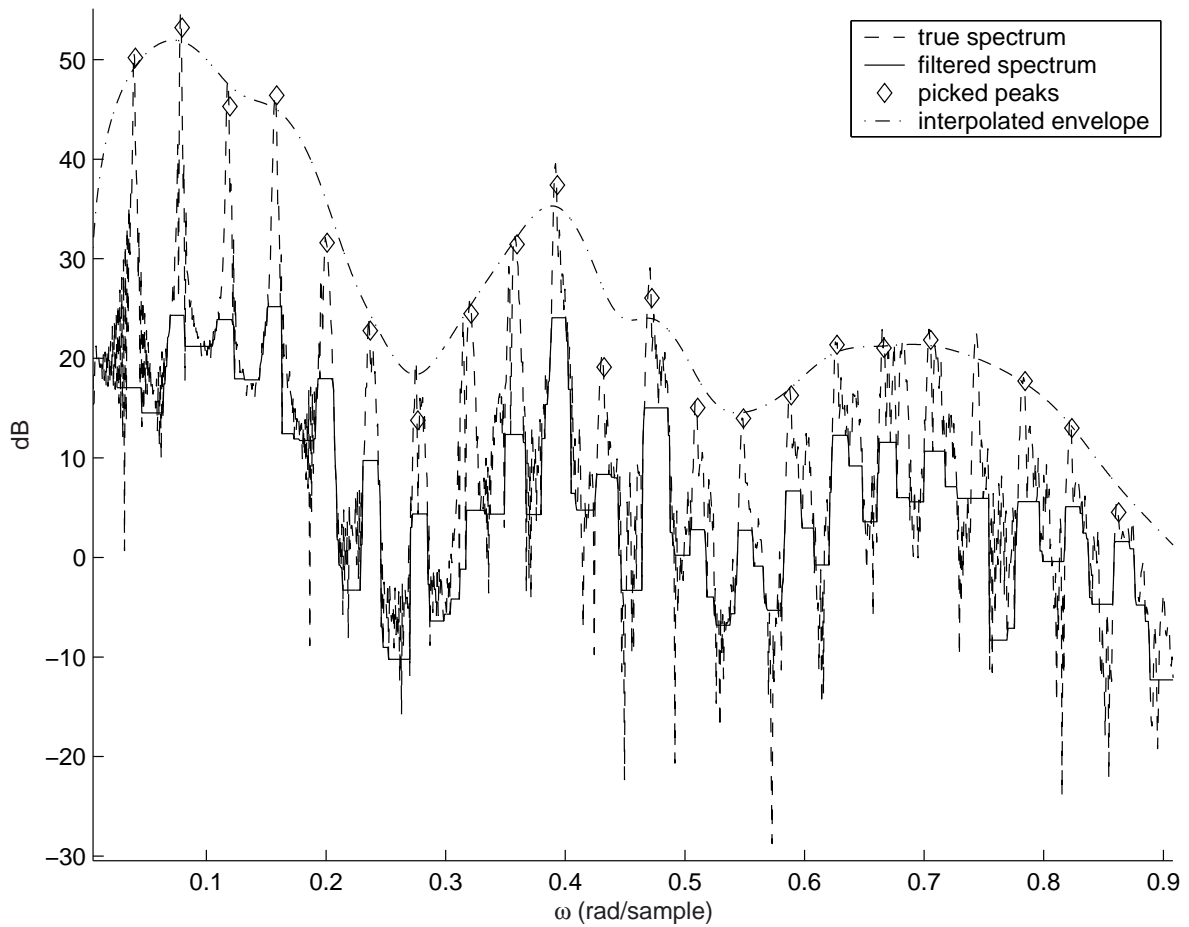


Figure 6.10: Steps of direct-modelling

We proceed to apply the piecewise cubic B-spline for direct-modelling. The interpolation of the spectral peaks caused by the excitation harmonics gives a smooth spectral magnitude estimate that can be compared to a magnitude spectrum obtained through *LPC*-analysis. We introduce a new symbolic operator \mathcal{I} , signifying the interpolation stage. We can express magnitude approximation $\widehat{M}(\omega)$ as:

$$\widehat{M}(\omega) = \mathcal{I} \cdot \mathcal{P} \cdot \mathcal{J}_q |X(\Omega_k)|. \quad (6.8)$$

Note that the interpolation gives a continuous spectrum. In practice, $\widehat{M}(\omega)$ has to be sampled again to be used as a discrete spectrum: $\widehat{M}(\Omega_k)$.

Figure 6.10 illustrates the steps towards a direct-model. The interpolation stage follows the peak extraction stage and draws a smooth curve with the peaks as “guidelines”, to approximate the spectral envelope.

Once we have a direct-model of a spectrum $|X(\Omega_k)|$ we can use it to whiten the spectrum to get the excitation magnitude spectrum $|E(\Omega_k)|$, where

$$|E(\Omega_k)| = \frac{|X(\Omega_k)|}{\widehat{M}(\Omega_k)}. \quad (6.9)$$

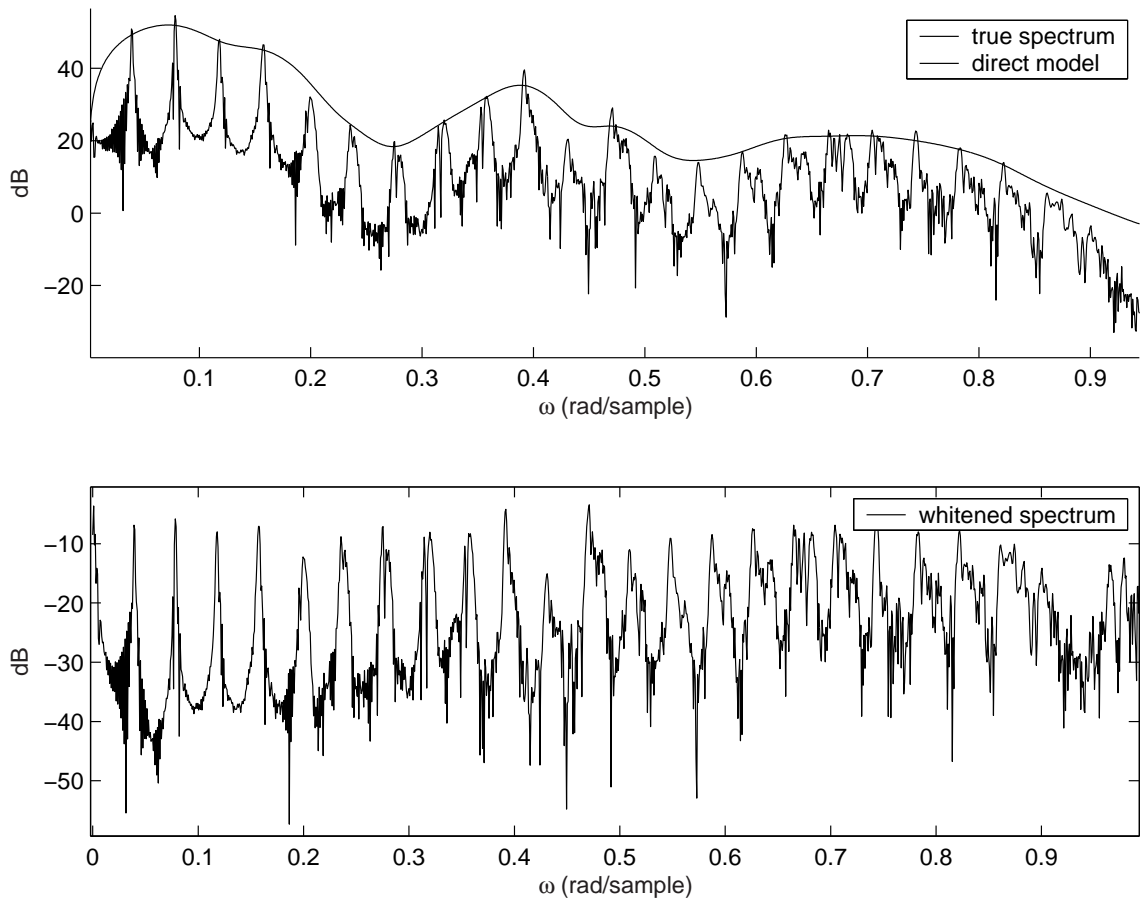


Figure 6.11: Spectral whitening through direct-modelling

Figure 6.11 shows how this formula can whiten a spectrum, giving us the normalised excitation spectrum. Since it is normalised, the model being applied to restore the formants must account for the volume (or power).

Formant shifting

By using direct-modelling, we have a numeric representation of the spectral envelope of a certain frame and therefore it is trivial to change the shape of the envelope. This gives us total control on how we want to modify the spectrum, which makes it a handy tool for formant shifting. Suppose we have a spectral envelope $\widehat{M}(\Omega_k)$ of magnitude spectrum $|X(\Omega_k)|$. Shifting the formants means moving the spectral envelope which presents two problems: shifting the formants higher or lower in frequency. When moving the formants artificially, we encounter missing bands in the spectrum. In section 6.3 we will discuss the process of filling empty frequency bands but for the moment we give a short summary of how to solve the problem:

- When the formants are moved lower, a gap appears in the high frequency areas. In order to utilise the maximum bandwidth supported by the sampling rate, we

repeat the high frequency values in the envelope to regenerate the missing ones. At high sampling rates the higher parts of the spectrum has noise characteristics and therefore the envelope as well and we can copy parts from the high end of the envelope to fill the missing band.

- When the formants are moved higher, we insert low level noise between 0Hz and the start of the original envelope to fill up the new envelope.

In both cases above, values are added to the discrete envelope $\widehat{M}(\Omega_k)$, thereby increasing its length. Since the sampling rate only supports a specific number of values in the discrete spectrum², we need to discard the same number of values that were added. In both cases, we discard values on the opposite side from where the new values were added. Figure 6.12 illustrates this principle.

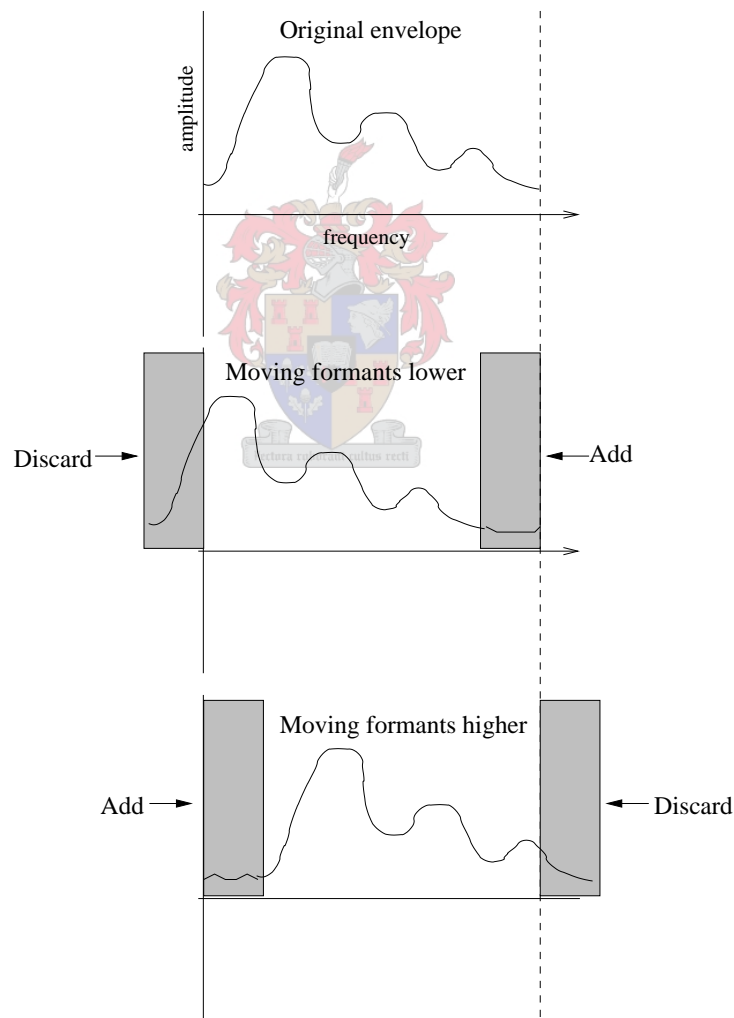


Figure 6.12: Formant shifting

²Dependent on the window size and the sampling rate.

We will put this technique to use in Chapter 8 where we will attempt to change the gender of a singing voice.

6.2.4 Comparing *LPC* to direct-modelling

In sections 6.2.2 and 6.2.3 we described two methods to estimate spectral magnitude estimate. Both methods model the envelope of the spectrum and not the excitation and its harmonics. This is a valuable property since we are only interested in the location of the formants and not the waveform that excites them. In the definition of the *LPC* spectrum, we know that it is a minimal MSE-fit on the true signal (see Appendix A). As for the direct-model, we cannot make any statements on how well it fits onto the spectrum before it is calculated. One solution to optimise the fit of a direct model is to calculate a constant, C^2 , so that $\frac{1}{C}\widehat{M}(\Omega_k)$ has the same energy as $X(\Omega_k)$. This should enable us to compare the spectral modelling of the *LPC* with that of direct-modelling. We calculate C^2 as follows:

If E is the energy in $X(\Omega_k)$,

$$E = \frac{1}{N} \sum_{k=0}^{N-1} |X(\Omega_k)|^2 \quad (6.10)$$

and the energy contained in the direct-model is

$$E_{DM} \triangleq \frac{1}{N} \sum_{k=0}^{N-1} \widehat{M}(\Omega_k)^2. \quad (6.11)$$

Then

$$C^2 = \frac{E_{DM}}{E}. \quad (6.12)$$

Now we can state that the energy in $\frac{1}{C}\widehat{M}(\Omega_k)$ is equal to the energy in the true spectrum and we can compare a direct-model to a *LPC*-model as in Figure 6.13. As expected, the models form a similar spectral estimation.

In the scope of this thesis, the direct-model has three prominent advantages over the *LPC*-method:

- Individual pitch harmonics will not be modelled as opposed to the *LPC*-model in the case of a high pitched female voice and a rather large filter order. This property can make the *LPC*-model dangerous to use since it may prove adequate when modelling male and alto female voices, but can fail when the pitch becomes very high. Since the direct-model interpolates the prominent peaks, the valleys between harmonics will not be modelled as long there are no spurious peaks in those regions.
- The valleys between formants are modelled exceptionally well. A model obtained through *LPC*-analysis does not go down as “deep” as a direct-model. See figure 6.13.

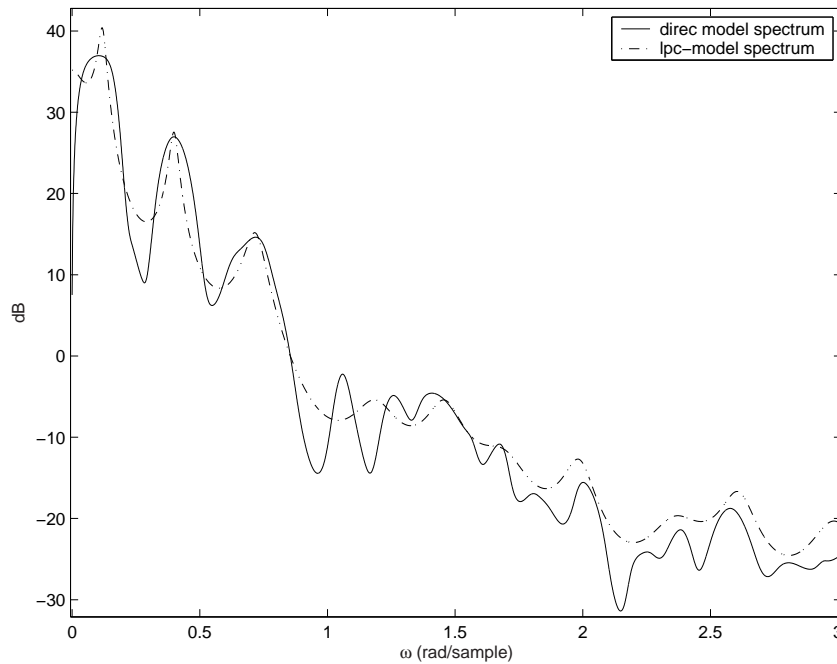


Figure 6.13: *LPC*-modelling and direct-modelling

- The direct-model can be estimated directly in the frequency domain, which is an attractive property when working with phase vocoders because the signal is presented by short-time spectra. The *LPC*-method requires a time domain representation of a signal in order to model it.

The direct-model has two significant disadvantages:

- It is computationally expensive to calculate the splining curves because of the polynomial calculations. We express it as a continuous spectrum $\widehat{M}(\omega)$ that has to be resampled again to give $\widehat{M}(\Omega_k)$, making this type of modelling unsuitable for real-time applications.
- The direct-model does not provide phase information about the signal it models, only magnitude information.

6.3 Modifying the excitation

As described above, we can decompose a single frame under quasi-stationary assumptions into its excitation signal and the corresponding spectral model. Now we turn the attention to the excitation signal and how to modify its fundamental frequency. There are two ways to achieve this:

- Time domain: Resample the excitation signal $e(n)$ in the time-domain at intervals $\frac{1}{\beta}$. Doing this will change the spacing between the time domain pitch pulses by a factor

β . The signal's fundamental frequency will change by a factor $\frac{1}{\beta}$ (the spectrum will expand or compress at the same factor and the time duration will change by a factor of β). This should deliver satisfactory results, but if the long-term signal of interest is presented by a phase vocoder, it would prove useful to have a method operating in the frequency domain.

- Frequency domain: Expand or compress the excitation spectrum $E(\Omega_k)$ by a factor β . If we resample the spectrum at intervals β , the result will be equivalent to that of the above time domain method. It is important to note that the time domain signal will expand/compress with the inverse factor of the frequency domain's compression/expansion. This method is more useful than the previous because of its frequency domain application and therefore the rest of this section will be devoted to its details.

Suppose we have a complex spectrum $E(\Omega_k)$ and we want to expand or compress it in the frequency domain. This means we need to evaluate the spectrum at frequencies that lie between the samples we can reach through the index k . One solution is a simple linear interpolation between the closest two values of the original spectrum. Let $E'(\Omega_k)$ denote the new spectrum, then:

$$E'(\Omega_k) = (1 - \rho(k))E(\Omega_{\bar{k}}) + \rho(k)E(\Omega_{\bar{k}+1}), \quad (6.13)$$

where

$$\bar{k} = \lfloor k\beta \rfloor$$

and

$$\rho(k) = k\beta - \bar{k},$$

where $\lfloor k\beta \rfloor$ indicates the largest integer smaller than $k\beta$. Note that the interpolation is complex, meaning we resample the magnitude and the phase. The same results can be achieved by resampling the magnitude and the unwrapped phase independently. The above expressions cause the fundamental frequency to change with a factor $\frac{1}{\beta}$ and the time duration with a factor β . We expand on this technique as described in [12].

Two important observations regarding spectral manipulation can be made on the basis of Figure 6.14:

- If the spectrum expands ($\beta < 1$), the spectral content spreads beyond $\frac{1}{\beta\pi}$, meaning some frequencies are moving into a range that are above the *Nyquist frequency*³ and

³Half the sampling rate

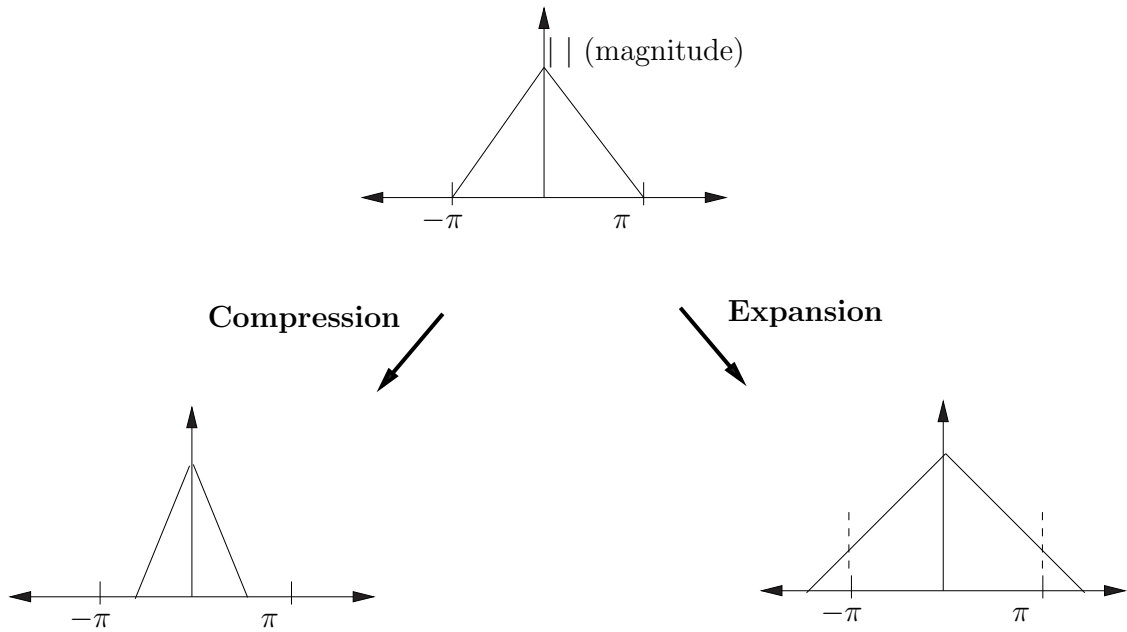


Figure 6.14: Spectral manipulation

cannot be supported by the sampling rate. We deal with this by simply discarding the spectral information in these areas. The dashed lines in Figure 6.14 indicate this cutoff. The motivation behind this is simple. If we assume that the modified spectrum can be obtained by sampling an original source, these are the frequencies that are too high to be sampled and would not appear in the spectrum.

- If the spectrum is compressed ($\beta > 1$), a dead band occurs between the highest frequency in the new spectrum and the *Nyquist frequency*. This is a quite significant artifact at low sampling rates, e.g. 8 kHz. Suppose we have a voiced spectrum with $F_0 = 400Hz$, and we want to modify F_0 so that $F_0 = 200Hz$. This calls for $\beta = 2$. If the described techniques are applied to modify the pitch, we will end up with a signal that occupies a bandwidth of only $2kHz$, which is extremely low. The result is less audible when for example $F_s = 44.1kHz$, but is still significant. This problem can be solved by a method called *high frequency regeneration*. A certain form of *high frequency regeneration* is called *spectral copying*. Values from lower parts of the spectrum are used to fill the missing band. This method works well for higher sampling rates since the higher parts of the spectrum has complex noise⁴ characteristics and can be copied to other high parts without introducing audible artifacts.

⁴Modelling magnitude and phase scattering.

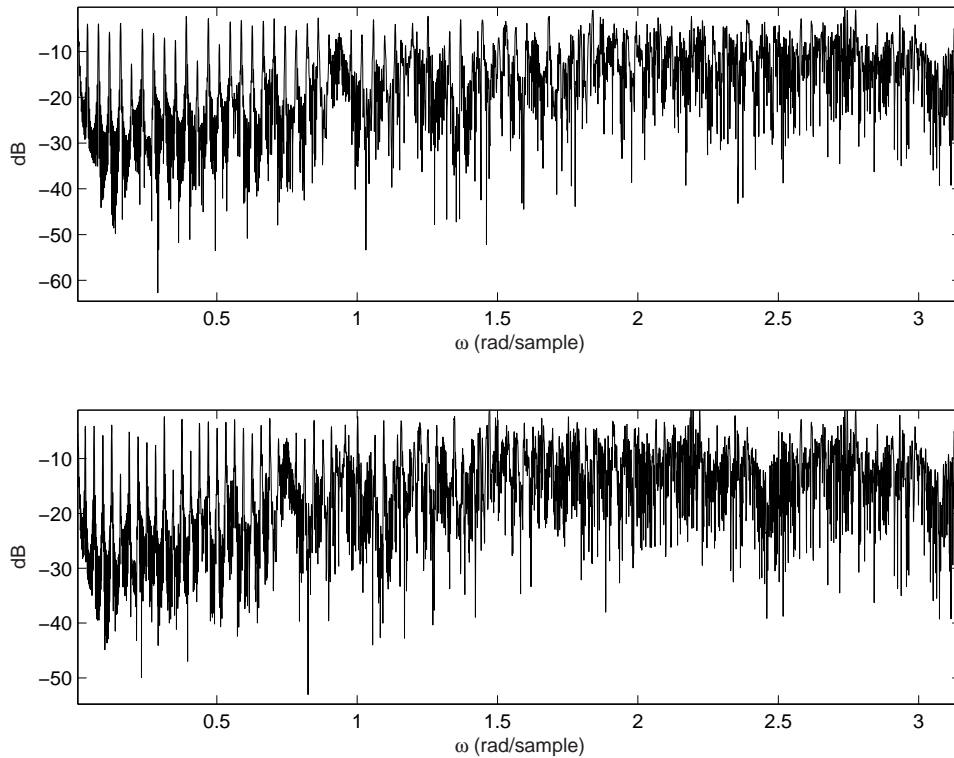


Figure 6.15: White spectrum, before (top) and after (bottom) pitch modifications

As an example of pitch-shifting we choose $\beta = 1.25$ and $r = 8$, a zero-padding factor, and apply it to the same frame used for Figure 6.11. The frame is windowed by a *Hamming* window. The top spectrum in Figure 6.15 is the original excitation magnitude spectrum and the bottom one is a resampled version. A closer look should make it clear that the top spectrum is compressed into the lower three quarters of the bottom spectrum. The missing band is filled in by spectral copying, therefore the last part of the artificial spectrum repeats itself. We take a section from the lower part of both spectra to investigate the excitation harmonics. It is clear from Figure 6.16 that the harmonic spacing changed, meaning the fundamental frequency changed (by a factor $\frac{1}{\beta}$).

6.4 Breathing life into the excitation

6.4.1 More than pitch matters

We discussed a technique to change the fundamental frequency of the excitation of a voiced frame of singing. This new signal would sound terrible, to say the least, since we've flattened its spectrum. To restore the human character of the sound, we need to re-apply any of the models described in section 6.2. This will force the formants of the original singer onto an artificial excitation, $e'(n)$, resulting in a signal that should

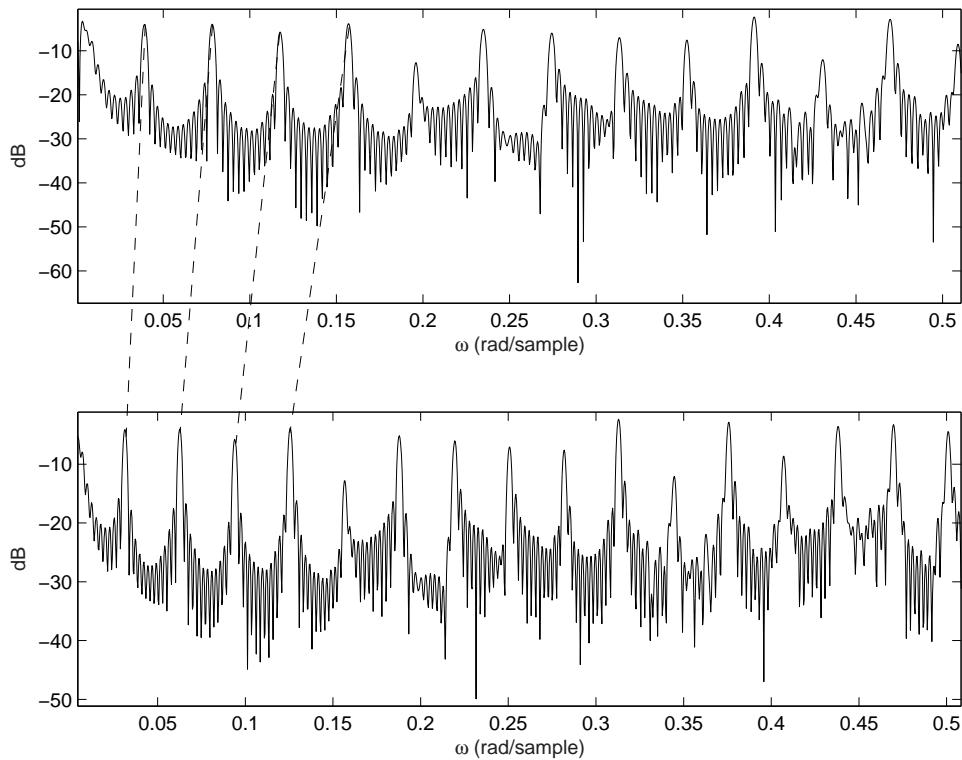


Figure 6.16: Enlarged extractions from Figure 6.15

sound like the same vocalist, singing the same vowel but at a fundamental frequency of our choice (within bounds).



6.4.2 Re-applying a *LPC*-model

Equation 6.1 (p49) is an expression for a *LPC*-model derived from a short-time signal $x(n)$. Once the coefficients are calculated through the autocorrelation method, we can implement the filter in the discrete-time domain. Let $y(n)$ be the desired signal, that is a pitch-shifted version of $x(n)$, and $e'(n)$ the artificial excitation. Using equation 6.1 and the \mathcal{Z} -transform we can state:

$$Y(Z) = E'(Z)H(Z). \quad (6.14)$$

Taking the inverse z -transform and rearranging terms we get a time domain expression for $y(n)$:⁵

$$\begin{aligned} y(n) = & G[e'(n) + a_1e'(n-1) + a_2e'(n-2) \dots \\ & + a_{p-1}e'(n-p+1) + a_pe'(n-p)]. \end{aligned} \quad (6.15)$$

⁵When we use equation 6.15 with the gain term G included, $e'(n)$ must have unity power so that $x(n)$ and $y(n)$ have equal power. If equation 6.15 is used with $G = 1$, $e'(n)$ alone accounts for the power in $y(n)$.

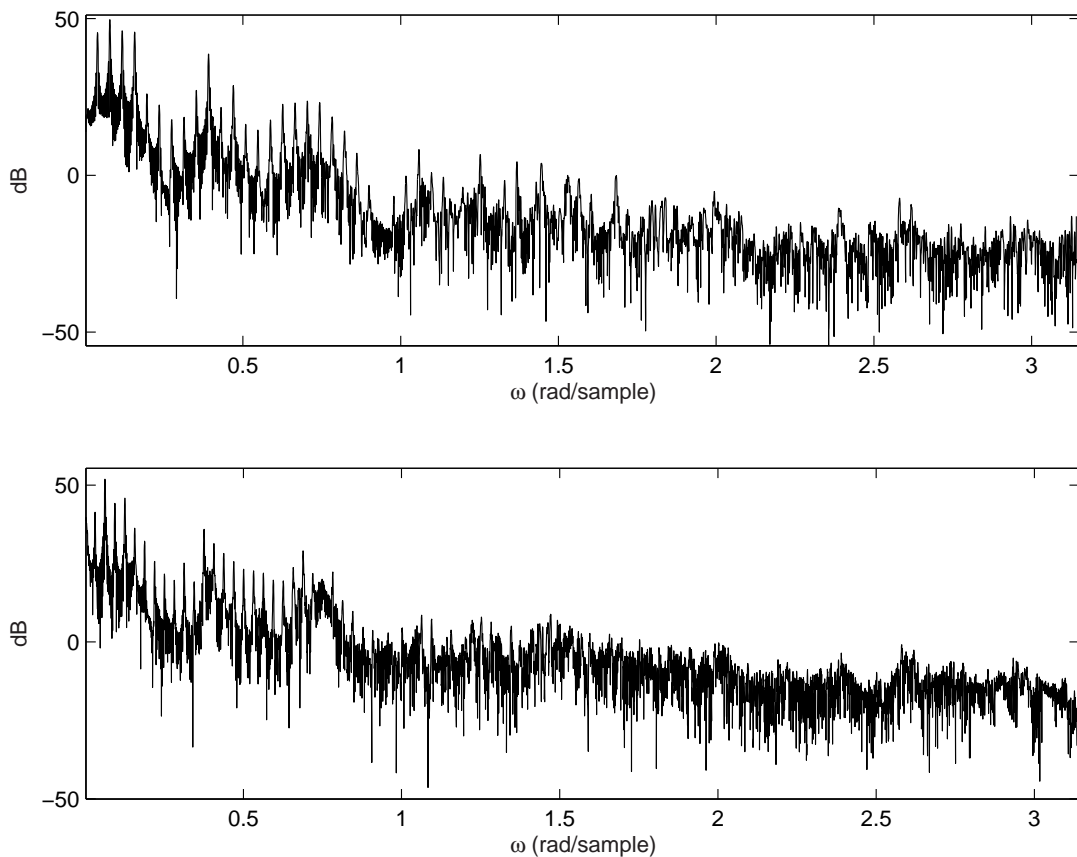


Figure 6.17: Pitch-shifting: LPC model used to restore formants
original spectrum (top) and artificial spectrum (bottom)

To demonstrate, we take the artificial excitation spectrum in Figure 6.15, transform it to the time domain to get $e'(n)$, filter $e'(n)$ as in equation 6.15, transform back to the frequency domain and compare it to that of the original in Figure 6.17. It is evident from Figure 6.17 that the formants are restored.

The path we followed up to here can be said to be a true pitch-shifting technique since we changed the fundamental frequency without moving the formants.

6.4.3 Re-applying a direct-model

The restoration of the formants by means of a direct model is simply a multiplication of two spectra. Let $E'(\Omega_k)$ be the spectrum of the pitch-shifted excitation, i.e. the Fourier transform of $e'(n)$, and let $\widehat{M}(\Omega_k)$ be the direct-model. The artificial magnitude spectrum $|Y(\Omega_k)|$ can be calculated as:

$$|Y(\Omega_k)| = |E'(\Omega_k)|\widehat{M}(\Omega_k). \quad (6.16)$$

Figure 6.18 shows the same excitation used in Figure 6.17 to calculate $|Y(\Omega_k)|$, only this time we employed the direct-model. Note how the formants are restored, very much like

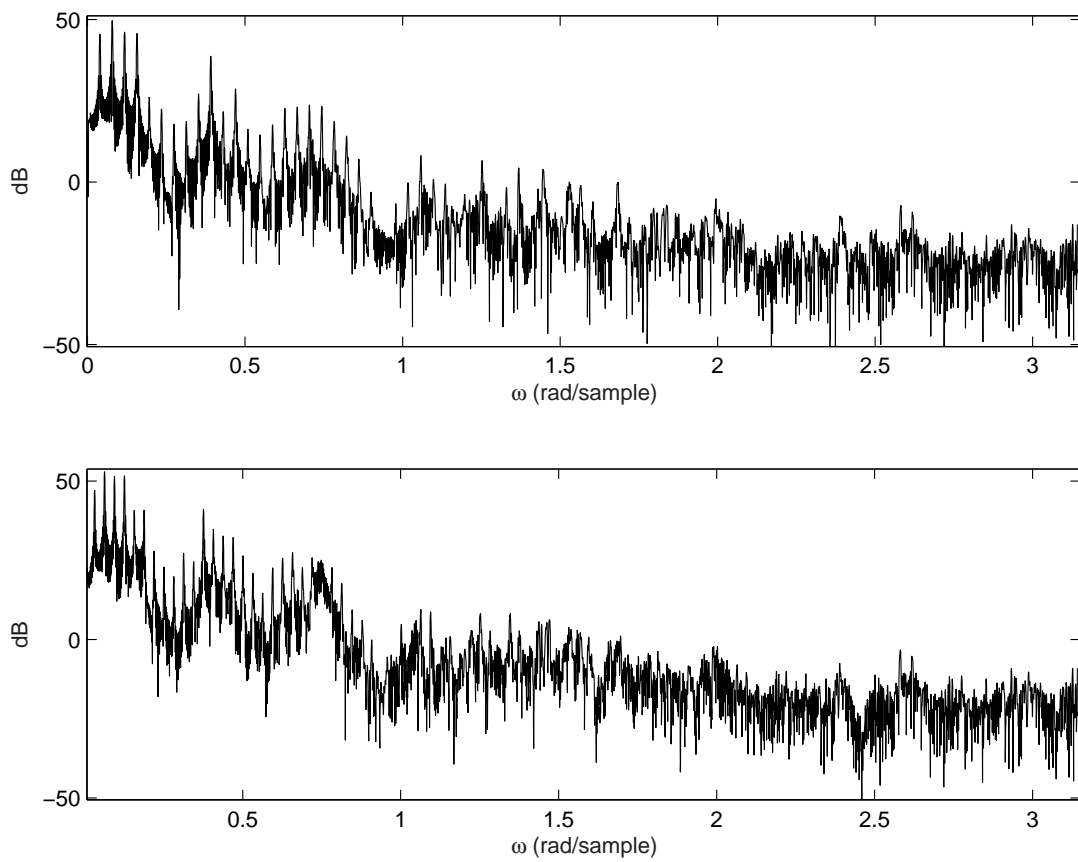


Figure 6.18: Pitch-shifting: direct-model used to restore formants
original spectrum (top) and the artificial spectrum (bottom)

in the *LPC* case.

6.5 A note on phase

Throughout the chapter we mentioned nothing about the phase of individual components of the frame. This becomes an important issue when a signal is studied as a series of frames, and each frame needs to be pitch-shifted. The phase needs to run free from one artificial frame to the next to avoid phase distortion. In Chapter 7 we will use the properties of the phase vocoder to ensure continuous phase as described in Chapter 4.

6.6 Summary

We started this chapter with an example of the serious artifacts caused by an elementary algorithm that tries to change the pitch of a signal. The problem we face when changing the playback sampling rate is the formants that move to unwanted locations. We described an algorithm, operating on a single frame, that aims to change the pitch while introducing as little as possible other modifications to the signal - which is the definition of a true pitch-shifter.

The design led us through source-filter decomposition where we discussed two different methods: *LPC*-modelling and direct-modelling. After the separation we described how to change the pitch of the excitation signal, and then how to restore the original spectral envelope.

The design is able to shift the pitch of a steady state vowel in a very natural way, however there are limitations on how far we can shift the pitch before it sounds unnatural. The amount of pitch shift that can be applied is rather dependent on the vowel and the singer, but in most cases a shift of up to 4 semitones succeeds.

In the next chapter we will use this design to design a bigger system that can shift the pitch of a long-term signal. By dividing the long-term signal into overlapping frames and by applying the pitch-shifting algorithm to each frame, we have control over the pitch of a long-term signal.

Chapter 7

Artificial singing voices

Heard melodies are sweet, but those unheard are sweeter.

- John Keats

7.1 Introduction

In this chapter we combine techniques described in earlier chapters to create synthetic vocals in the hope of doing justice to their human counterparts. We will consider a sampled audio signal $x(n)$, the source vocal, and create a target vocal signal $y(n)$, standing in some music-theoretical relationship to $x(n)$. The relationship could be one of three main options:

1. $y(n)$ could be a vocal signal with a harmonious melody, governed by music theory, that complements $x(n)$.
2. $y(n)$ could be an improved version of $x(n)$, meaning that $y(n)$ contains less unpleasant deviations from the applicable musical scale than $x(n)$. This process is also referred to as pitch correction.
3. $y(n)$ could be a constant pitch-shifted version of $x(n)$.

The purpose of this chapter is to describe a non-causal technique through which these ideals could be achieved. This method assumes we have complete information on the signal before the processing starts, i.e. past, present and future values are known. This calls for a pre-recorded audio track. Figure 7.1 shows a diagram of the system we are about to discuss. Each functional block will be dealt with individually and the reader may want to refer back to the diagram as we advance through the chapter.

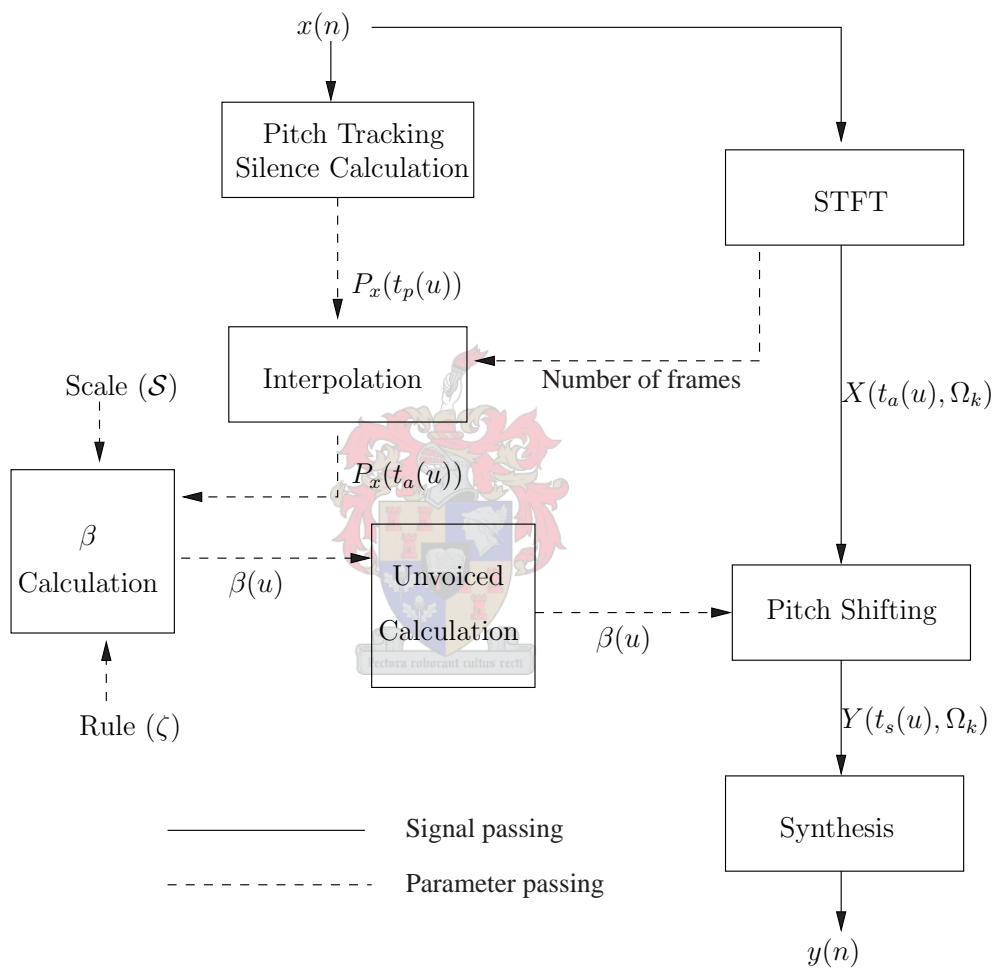


Figure 7.1: High level diagram of the non-causal pitch-shifting system

7.2 Analysis

The signal $x(n)$ is transformed into a series of STFT instances, $X(\Omega_k, t_a(u))$, as in the *analysis* stage of the phase vocoder (as explained in Chapter 4).

7.3 Pitch-tracking

For the sake of clarity we repeat some of the notation introduced in Chapter 5. $P_x(t_p(u))$ is a pitch-track of $x(n)$, calculated from windows starting at time instants $t_p(u) = uR_p$, where u is an integer called the pitch frame index and R_p is the pitch hop-size. We calculate the pitch-track through the method described in Chapter 5.

Since the pitch-track $P_x(t_p(u))$ does not necessarily have the same time resolution as $X(t_a(u), \Omega_k)$, i.e. $t_p(u) \neq t_a(u)$, a simple linear interpolation must be performed on the pitch-track so that a pitch value is associated with each *analysis* instant. After interpolation we have $P_x(t_a(u))$.

7.4 Pitch-shifting

Pitch-shifting can be performed by employing any of the modelling methods described in Chapter 6. It is convenient to whiten the frames through inverse *LPC*-filters before Fourier transforming it, and to use direct-modelling to restore the formants after spectral resampling.

Each *analysis* instant, $X(\Omega_k, t_a(u))$, may be pitch-shifted individually in order to create the *synthesis* instants $Y(\Omega_k, t_s(u))$, which can be recombined by an overlap-and-add procedure to form $y(n)$. This is a phase vocoder process and allows for dynamic pitch modifications, since each frame can be pitch-shifted by an independent factor β . If β is constant, $y(n)$ will be the same melody as $x(n)$, only in a different musical key, i.e. higher or lower. If β is variable, then $y(n)$ can be a different melody, depending on the control of β . The dynamic behaviour of β is governed by two parameters: a scale and a rule - the topic of the next section. Since β is not a constant anymore we may express it as a function of the frame index u , thus we have $\beta(u)$.

To summarise: We shift $X(\Omega_k, t_a(u))$ by a factor $\beta(u)$ to get $Y(\Omega_k, t_s(u))$.

7.5 β calculation

7.5.1 Scale notation

Before we show how to calculate β for each frame, we need to introduce our notation of scales and a new type of pitch-track.

A scale is a series of musical frequencies and each element denotes a certain note. We denote a scale, associated with $x(n)$, that spans the range of a piano, by \mathcal{S} . Each element in \mathcal{S} can be addressed by using an integer subscript. \mathcal{S} is defined so that $\mathcal{S}_j < \mathcal{S}_{j+1}$. The scale specifies the allowable frequencies for a melody played in that scale. All fundamental frequencies that occur in $x(n)$ should be contained in a scale, or should approximate a value in the scale. If this condition is not met, $x(n)$ will be “off-key” and unpleasant when listened to.

Since $P_x(t_a(u))$ is a pitch-track derived from human created audio signals, the frequencies will seldom be exact as prescribed by the scale, therefore we introduce a new type of pitch-track: $P_x^{\mathcal{S}}(t_a(u))$, which contains frequencies from the scale \mathcal{S} that are closest to the values in $P_x(t_a(u))$. We can summarise its calculation in two equations:

$$P_x^{\mathcal{S}}(t_a(u)) = \mathcal{S}_{m(u)}, \quad (7.1)$$

where

$$m(u) \triangleq \underset{j}{\operatorname{argmin}} \left| \mathcal{S}_j - P_x(t_a(u)) \right| \quad (7.2)$$

and where \mathcal{S} is the scale in which $x(n)$ is performed. Figure 7.2 illustrates the above discussion. More information on scales and music theory appear in Appendix B.

7.5.2 Harmony

The rule specifies what β should be for any given pitch value of $x(n)$. One class of rules is what we call *constant step* rules¹: β is calculated so that if

$$P_x^{\mathcal{S}}(t_a(u)) = \mathcal{S}_{m(u)} \quad (7.3)$$

and a rule of ζ (an integer) is applied, then

$$P_y(t_s(u)) = \mathcal{S}_{m(u)+\zeta}. \quad (7.4)$$

In general, if the scale and rule are known, we can express β as a function of the frame index u :

$$\beta(u) = \frac{\mathcal{S}_{m(u)+\zeta}}{P_x^{\mathcal{S}}(t_a(u))}. \quad (7.5)$$

¹We call it *constant step* because the harmony $y(n)$ will contain notes from the scale that are a constant number of notes higher or lower than the notes in $x(n)$.

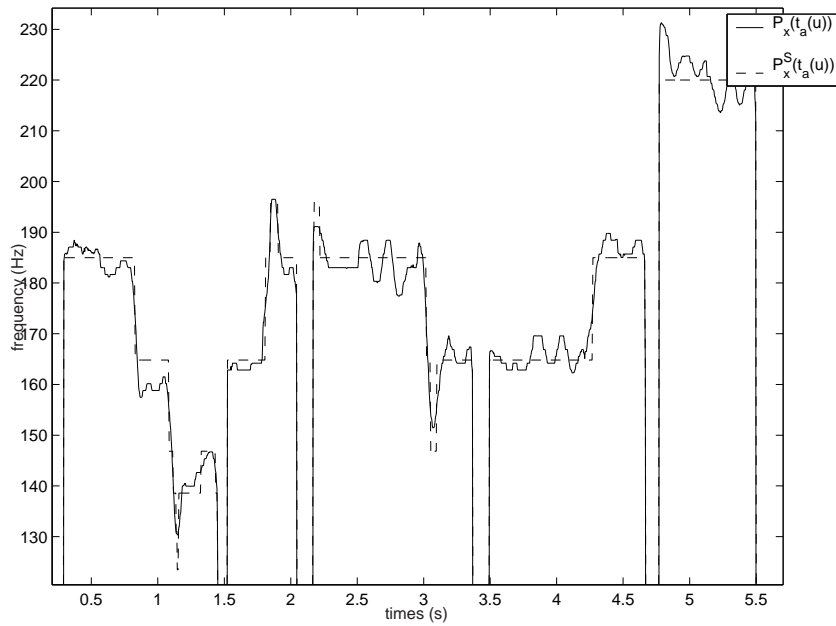


Figure 7.2: Example of $P_x(t_a(u))$, an original pitch-track, and $P_x^S(t_a(u))$, a scale corrected pitch-track

Equation 7.5 is just a specialised form of equation 1.1 (p5), which we repeat below:

$$\beta = \frac{\text{source pitch}}{\text{target pitch}}. \quad (7.6)$$

It is important to note that although the rule ζ stays constant, β can be a varying quantity, and is dependent on the nature of the scale. Table 7.1 includes some of the possible *constant step* rules.

By using the formulae in the likes of those in Table 7.1, we assume that the vocalist sings true to the scale, and subtle pitch deviations like vibrato and “scoops” will be passed on to the artificial signal $y(n)$. If we do not want these subtleties we can correct the pitch of the signal in another stage.

7.5.3 Pitch correction

$$\beta(u) = \frac{P_x(t_a(u))}{P_x^S(t_a(u))} \quad (7.7)$$

The effect we get from applying equation 7.7, is a frequency clamping of the source vocal and all off-key frequencies are changed into their closest counterpart in the specified scale. This is pitch correction.

7.5.4 Constant pitch-shift

The preceding discussion was on cases where the value β was a dynamic quantity. A very simple case of pitch-shifting is where β is a constant value. The results of such a process,

Table 7.1: β calculation rules

| $Rule(\zeta)$ | $\beta(u)$ | Effect |
|---------------|--|-----------------------------------|
| 2 | $\frac{\mathcal{S}_{m(u)+2}}{P_x^{\mathcal{S}}(t_a(u))}$ | Third harmony above source vocal |
| -2 | $\frac{\mathcal{S}_{m(u)-2}}{P_x^{\mathcal{S}}(t_a(u))}$ | Third harmony below source vocal |
| 3 | $\frac{\mathcal{S}_{m(u)+3}}{P_x^{\mathcal{S}}(t_a(u))}$ | Fourth harmony above source vocal |
| -3 | $\frac{\mathcal{S}_{m(u)-3}}{P_x^{\mathcal{S}}(t_a(u))}$ | Fourth harmony below source vocal |
| 7 | $\frac{\mathcal{S}_{m(u)+7}}{P_x^{\mathcal{S}}(t_a(u))}$ | Octave harmony above source vocal |
| -7 | $\frac{\mathcal{S}_{m(u)-7}}{P_x^{\mathcal{S}}(t_a(u))}$ | Octave harmony below source vocal |

$y(n)$, is an audio signal with a pitch-track that can be expressed as:

$$P_y(t_p(u)) = \beta P_x(t_p(u)). \quad (7.8)$$

7.6 Unvoiced and silence detection

We implement a very simple but effective way of detecting unvoiced parts. Singing has very little unvoiced sounds and if they appear, it is for a short duration only. Therefore this part of the system is not absolutely necessary, and is a safety feature.

Since unvoiced sounds do not have significant fundamental frequencies, we would like them to be copied unchanged from $x(n)$ into the synthetic signal $y(n)$, instead of being

pitch shifted.² Because of their noise characteristics, the pitch detection stage will be very unreliable and typically gives pitch values many times higher than the surrounding voiced areas. We use this typical mistake to label a region as unvoiced: if β is calculated from equations in the likes of those in Table 7.1, it will seldom be much larger than 2 or smaller than 0.5, such as in the octave harmony cases. Thresholds of 3 and 0.3 proved to be adequate. In other words, if β is found to be outside the area bound by these thresholds, the unvoiced detection stage changes β to a value of 1; thus the frame passes unaltered to the synthesis stage.

Silence does not really need detection because a frame that lacks audible volume cannot introduce audible artifacts. However, to save computation time we determine the areas where silence occurs through the method explained in section 5.6 (p43).

7.7 Synthesis

After each *analysis* instant is pitch shifted by a factor β , we have the *synthesis* instants $Y(\Omega_k, t_s(u))$. Each instant must be phase corrected as explained in Section 4.3.2 in order to avoid phase distortion. After the adjustments have been made, $Y(\Omega_k, t_s(u))$ may be inverse Fourier transformed, to get the short-time signals $y_w(n)$, used for *synthesis*. We use equation 4.8 (p28) to calculate $y(n)$. (Of course $y(n)$ must have the same length as $x(n)$ after *synthesis*.)

7.8 Windowing issues

The different window types used on the frames in the different algorithms are important for the success of the total system. Here we give a short summary of which type of window is used where:

- For the pitch detection stage we need to use a *square window* because of its narrow main-lobe. The narrow lobe width gives good frequency resolution which is desirable during pitch detection. Although the side-lobes are noisy, they will be suppressed successfully by a “LULU”-smoother of minimum order r , the zero-padding factor.
- For spectral whitening we use an *LPC*-filter that is calculated from a *Hamming*-windowed frame, since we want to smooth the edges of the frame.
- We restore the formants using a direct-model which is determined from a *square*-

²“S” sounds are produced the same, no matter how high or low the singer goes and it is good singing practice to reduce the use of it.

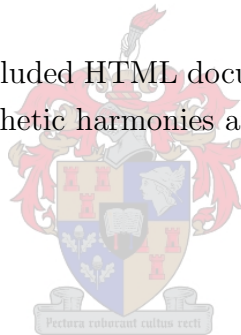
windowed frame of speech, since the peak extraction stage prefers a narrow main-lobe. Again, the relative high side-lobes will be suppressed by a “*LULU*”-smoother.

7.9 Summary

We used the techniques discussed in Chapters 3-6 to design a system that can create vocal signals from an original recording. The new, or synthetic, voices can be one of three types:

1. Harmonies for the original can be created by using *constant step* rules. These types of pitch-shifts create a note in the synthetic signal that is a constant number of scale steps away from the original note.
2. The synthetic signal can be an improved version of the original by changing off-key notes into ones allowed by the scale.
3. A constant pitch-shift, in other words the new signal’s pitch is higher or lower by a constant factor.

(The reader may refer to the included HTML document. The link to Chapter 7 examples provides audio examples of synthetic harmonies and pitch correction.)



Chapter 8

Further Applications

8.1 Introduction

This chapter is a case study of two rather esoteric but interesting applications of the theory discussed in the previous chapters and we go beyond the normal effects of pure pitch-shifting and time-stretching. The two applications are:

1. Based on a few stationary frames of a sung vowel, we design a wave synthesis technique in section 8.2. The frequency of the excitation to a vowel of choice can be controlled from a MIDI-keyboard, creating a continuous sung vowel, reminiscent of the vocal techniques used by members of a choir.
2. Transforming a voice from an original vocalist to a voice of the opposite sex.

8.2 Wave synthesis

We transform a high quality recording of a sung vowel $x(n)$, with arbitrary but constant pitch, into a series of short-term Fourier transforms as described in Chapter 4. This gives us a series of *analysis* instants, $X(t_a(u), \Omega_k)$. The frames in the centre region are isolated and the rest discarded in order to reject the transients expected at the beginning and end of the vowel. The remaining frames are then used in the synthesis process.¹

We use the pitch-shifting algorithm developed in Chapter 6 to create a variety of *synthesis* instants of the vowel at different pitches. Since the technique relies on prior knowledge of the vowel's original pitch, we need a short pitch-track of the vowel. We use the pitch-tracking algorithm in Chapter 5 to get the pitch-track² $P_x(t_p(u))$. We found

¹For the rest of the chapter, $X(t_a(u), \Omega_k)$ will denote the instants from the centre region.

²Calculated from the centre region.

that the average of the pitch-track is a good measure of pitch for the time span of the vowel, and we simply denote it by P_x , where

$$P_x \triangleq \frac{1}{K} \sum_{u=1}^K P(t_p(u)) \quad (8.1)$$

and K denotes the number of frames.

A MIDI-stream from a keyboard gives information on how long a note was held and what the frequency of the note was. For the sake of simplicity we only investigate the monophonic case, meaning the next note cannot start before the previous one ends. Suppose a note of frequency P_y was held for t seconds. This means a pitch-shifted version of $x(n)$ should be synthesised for a duration of t seconds. We achieve this in three steps:

1. Copies of $X(t_a(u), \Omega_k)$ are appended in series, giving more short-term Fourier transforms so that a signal with a duration of t may be synthesised.
2. After $X(t_a(u), \Omega_k)$ is “stretched”, each instant is pitch-shifted by a factor β to form the *synthesis* instants, $Y(t_s(u), \Omega_k)$. We calculate β as:

$$\beta = \frac{P_x}{P_y}. \quad (8.2)$$

3. The resulting signal is synthesised by combining the *synthesis* instants $Y(t_s(u), \Omega_k)$ through the overlap-and-add procedure.

Return to step 1 for the next note. The next note will join the previous one seamlessly because of the phase vocoder’s ability allows us to do phase adjustments.

Figure 8.1 shows a diagram of the above process. The result is long sustained vowels with a pitch that is up to the choice of a keyboard player. Since the pitch-shifting algorithm has limitations on how far β may deviate from unity, we found that simple tunes such as nursery rhymes with notes having frequencies in the same octave as the vowel’s original pitch, are best suited for demonstration. If a larger number of notes needs to be covered, we would have to store the same vowel recorded at a few different pitches.

8.3 Gender transformer

In Chapter 2 we noted with reference to Rossing [18], the difference between a female and a male voice is that the formants of a female voice are about 25% higher and the pitch about twice as high. This translates into a pitch-shifting factor of $\beta \cong 0.5$ accompanied

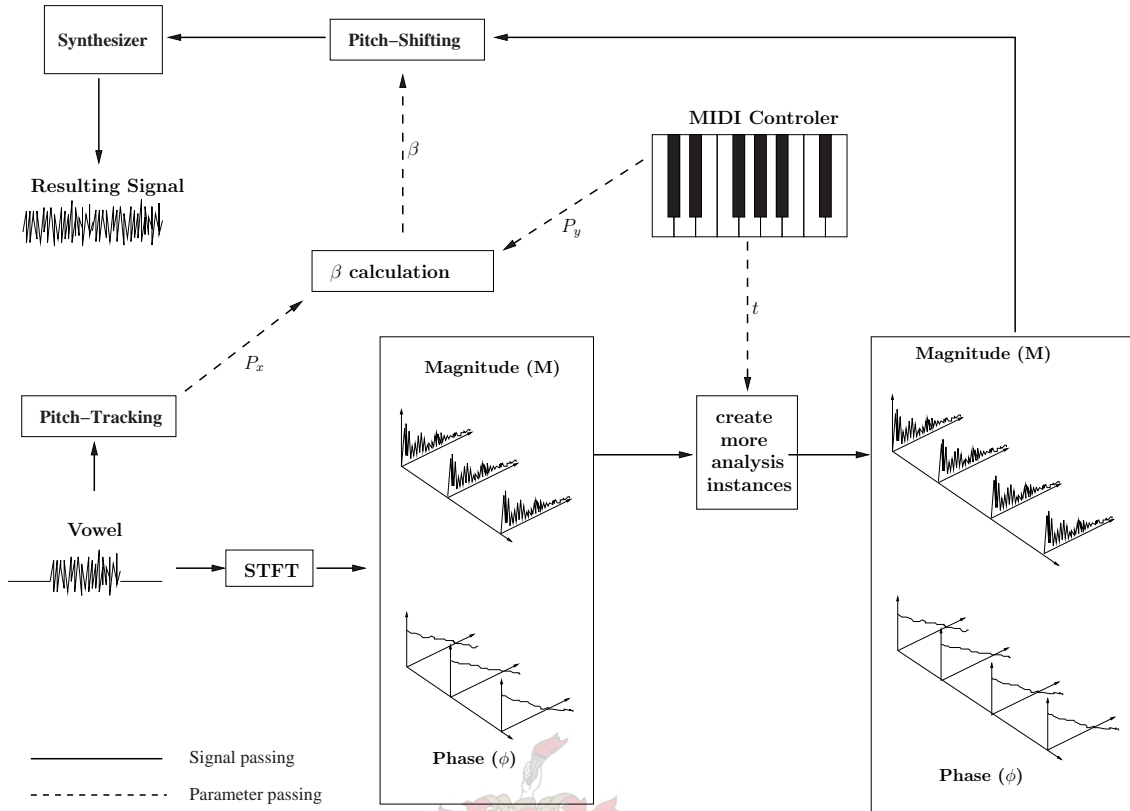


Figure 8.1: High level diagram of vowel wave synthesis

by a formant shift to transform the pitch of a male voice to a female voice. See Table 2.2 (p14) for the formant frequencies of male and female singers.

We combine pitch-shifting with formant shifting, both described in Chapter 6, to achieve the formant and pitch changes needed to change the gender of a voice. As before, we express the original signal as a phase vocoder, presenting us with short-time spectra, $X(t_a(u), \Omega_k)$. Each of the short-time spectra, or *analysis* instant, is pitch shifted by 100% and its direct-model $\widehat{M}(t_a(u), \Omega_k)$ is formant shifted by 25%. By formant- and pitch-shifting each short-time spectrum we have a series of *synthesis* instants $Y(t_a(u), \Omega_k)$ which is recombined through the overlap-and-add procedure after we made the necessary phase adjustments.

The technique is not perfect and its success is dependent on “tweaking” the algorithm for a specific voice. The naturalness of the result is hard to measure and is subjective, but is nonetheless an indication of how a recording would have sounded if done by someone of the opposite sex.

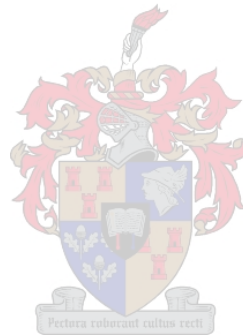
This application is particularly useful for creating a duet. Usually a duet is formed by a male and a female vocalist, and if we have one of the voices, the other one can be synthesised.

8.4 Summary

We designed and implemented two techniques:

1. A technique through which we can create a continuously sung vowel, of which the pitch is controlled by a MIDI-keyboard.
2. A technique for changing the gender of a pre-recorded singing voice.

These techniques employ the pitch-shifting, time-stretching and synthesis algorithms discussed in the previous chapters. (The reader may refer to the included HTML document. The link to Chapter 8 examples provides audio examples of wave synthesis and gender transformation.)



Chapter 9

Results

The thermometer of success is merely the jealousy of the malcontents.

-Salvador Dali

9.1 What to measure

Measuring results of the developed and implemented techniques in this thesis must be preceded by the question: What do we measure? Since the results are musical by nature, the only true judge of their success is the ears of people not bothered with the mathematics behind the results - and ultimately the ear of a musician. However, the “pleasantness” of the results is very hard to measure, not forgetting that non-scientific measurements, such as pleasantness, are intrinsically subjective.

For these reasons we turn to parameters that are absolutely measurable. These parameters occur in the functional building blocks of the described systems and are also measurements of success. Although not the ultimate measurements, they are at least deterministic:

1. Accuracy and resolution of the pitch-tracking algorithm.
2. Accuracy of the pitch-shifting algorithm and the extent by which it complies with the characteristics of a true pitch-shifter. After this discussion we show some results of applications of the pitch-shifting algorithm.
3. Seamlessness of the synthesis process.

We end the chapter by giving a high level evaluation of our designs.

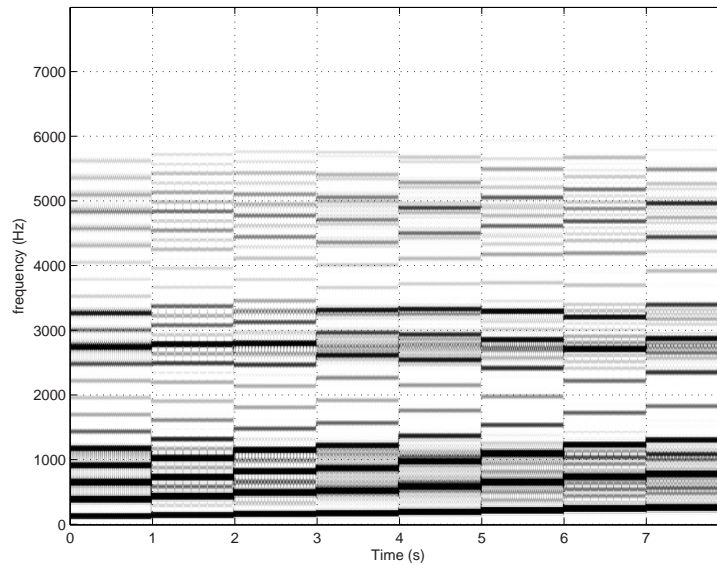


Figure 9.1: Spectrogram of a synthetic signal

9.2 Accuracy and resolution of the pitch-tracking algorithm

Throughout the thesis we used the same pitch-tracking algorithm as described in Chapter 5. To test the algorithm, we create a synthetic signal with known fundamental frequencies and compare this known pitch-track to the one determined by the algorithm.

We create a sinusoidal FM-signal, modulated so that it is a C-major scale; discrete frequency steps that lead from note C_3 to note C_4 on the piano (see Figure 2.6 p15). See Appendix B for more information on scales. To create higher harmonics, such as found in a singing voice, the FM-signal is put through a hard limiter¹, which turns the input into a square wave. To give the signal a human quality we correct the amplitudes of the harmonics by filtering it through an *LPC*-filter derived from a vowel. We denote this signal, as before, by $x(n)$. Figure 9.1 is a spectrogram of this synthetic signal. Note the harmonics and the formants.

Figure 9.2 shows the true pitch-track overlaid by the one determined by our algorithm. From a distance it appears to be nearly perfect, but a closer look shows a certain noise around the true pitch value. Figure 9.3 shows the noise around the true notes in $x(n)$. We list the following parameters for each note in Table 9.1:

- The mean of the pitch-track for the duration of the note [Hz].
- The true note [Hz]. We list this quantity to be sure that the mean is a very close

¹A transfer function that gives 1 as output for positive input and -1 for negative input.

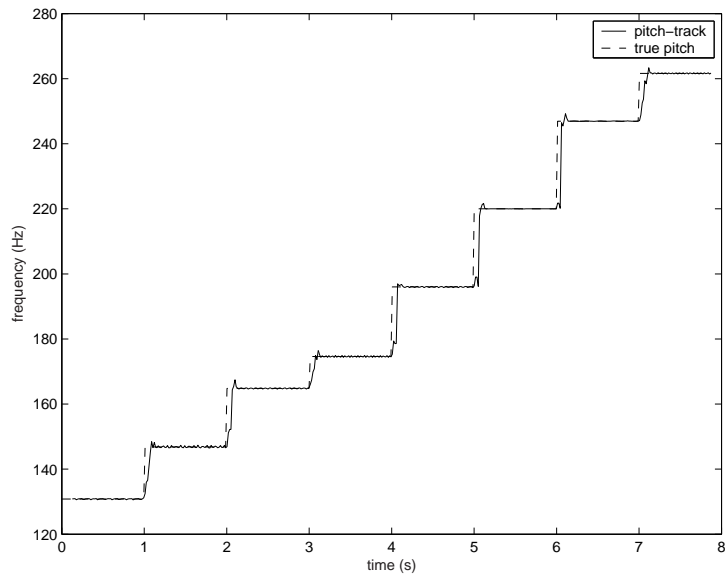


Figure 9.2: Pitch-track of a synthetic signal

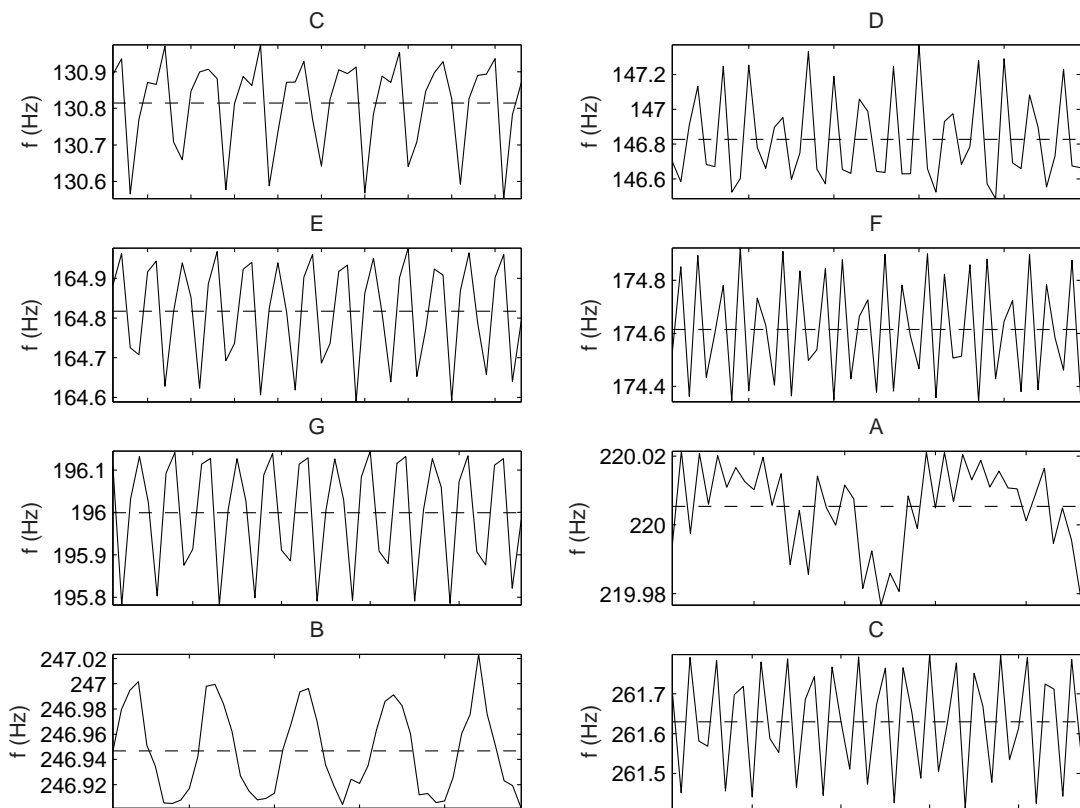


Figure 9.3: Pitch-track noise around the true pitch

Table 9.1: Pitch-tracking parameters

| mean | true note | σ |
|----------|------------|----------|
| 130.8142 | C 130.8150 | 0.1217 |
| 146.8276 | D 146.8349 | 0.2608 |
| 164.8173 | E 164.8166 | 0.1269 |
| 174.6145 | F 174.6171 | 0.2083 |
| 195.9992 | G 196.0010 | 0.1304 |
| 220.0054 | A 220.0037 | 0.0141 |
| 246.9468 | B 246.9458 | 0.0346 |
| 261.6295 | C 261.6300 | 0.1338 |

approximation of the true note.

- The standard deviation σ of the pitch-track for the duration of the note. [Hz]

The maximum standard deviation is $\sigma_{max} = 0.2608$ Hz. This means that we can expect the pitch-track to deviate a typical value of σ_{max} up or down from the true pitch. To answer the question on the accuracy of the pitch detection algorithm, we can state that the algorithm has an expected tolerance of σ_{max} . Since the tolerance can cause the pitch-track to be σ_{max} higher or lower than the true pitch, we can expect the pitch detection algorithm to detect deviations of $2\sigma_{max}$ (and higher) in the true pitch. Although the tests were performed on a synthesised signal, it still gives a good indication of its reliability when using real data.

9.3 Accuracy of the pitch-shifting algorithm

9.3.1 Evaluation of the algorithm

We use a recording of a two-tone vowel $x(n)$ sung by a male vocalist to test the pitch-shifting algorithm. We apply the algorithm for different values of β , where β is constant for the duration of $x(n)$. As before, the pitch-shifted signal is denoted by $y(n)$. After pitch-shifting we calculate pitch-tracks for $x(n)$ and $y(n)$ and denote them by $P_x(t_p(u))$ and $P_y(t_p(u))$ respectively. If the pitch-shifting algorithm is successful, the following equation should be true for all values of u :

$$P_y(t_p(u)) = \beta P_x(t_p(u)). \quad (9.1)$$

Since the pitch-shifting algorithm is not perfect, equation 9.1 is only an approximation of the real-life situation. We present an example to explain the test that follows it.

Table 9.2: Pitch-shifting parameters

| β | $\overline{\beta'}$ | average accuracy |
|---------|---------------------|------------------|
| 0.6250 | 0.6255 | 99.2014 % |
| 0.7700 | 0.7727 | 99.1830 % |
| 1.3000 | 1.3076 | 98.7839 % |
| 1.6000 | 1.6032 | 99.2348 % |
| 1.9000 | 1.8929 | 99.5791 % |

If, for example, we applied a pitch-shift of $\beta = 0.5$, we expect that

$$P_y(t_p(u)) = 0.5P_x(t_p(u)). \quad (9.2)$$

Say the pitch-shifting technique did not quite achieve this, but came close, so that the following equation is true for certain u :

$$P_y(t_p(u)) = 0.493P_x(t_p(u)). \quad (9.3)$$

We denote this approximation of β by β' , and use it as an accuracy measure of the pitch-shifting algorithm. For the above example β' has a value of 0.493. In general we calculate β' as:

$$\beta' = \frac{P_y(t_p(u))}{P_x(t_p(u))}. \quad (9.4)$$

A measure of accuracy of the pitch-shifting algorithm is the measure by which β' approximates β . We express it as a percentage:

$$\text{accuracy} = \left\{ 1 - \frac{|\beta - \beta'|}{\beta} \right\} \times 100. \quad (9.5)$$

We continue and perform several pitch-shifts on the test signal. Figure 9.4 shows the pitch-track of the original two-tone signal (dotted line), with pitch-tracks of five pitch-shifted copies. We chose β so that the pitch is shifted by a third at a time: three shifts up and two shifts down, thus $\beta \in [0.0526; 0.6250; 0.7700; 1.3000; 1.6000]$.

Using these pitch-tracks we calculate a series of β' s for the duration of each pitch-track. We plot β' versus u in Figure 9.5, overlaid by the theoretical value β . (Note the transient regions where the pitch jumps a semitone. Remember that the pitch-tracking algorithm does not work well during non-steady state times and that pitch-shifting depends on it.)

Omitting the transient areas, we average the values of β' to get $\overline{\beta'}$. Now, using β and $\overline{\beta'}$ in equation 9.5, we can calculate an average accuracy for each pitch-shift and list the results in Table 9.2.

As we mentioned before, the actual pitch-shifting is not the only matter concerning a pitch-shifting algorithm. The formants need to stay in their original locations to retain

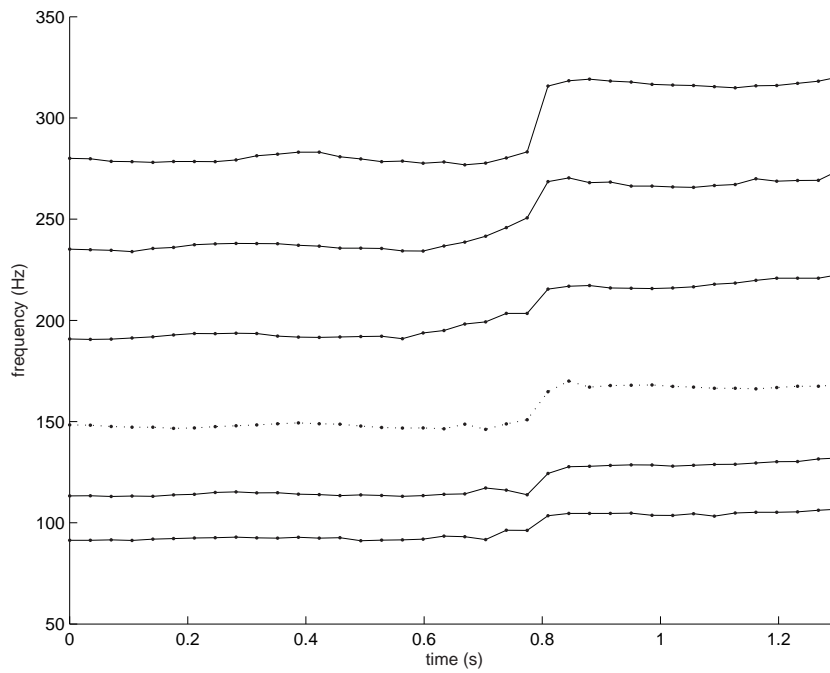


Figure 9.4: Pitch-tracks of an original signal (dotted line) and of five pitch-shifted copies

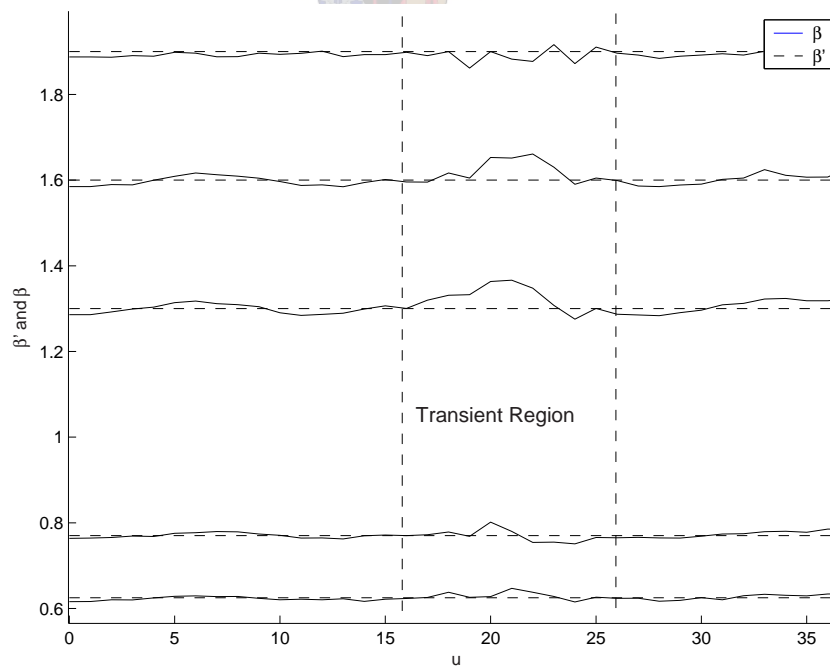


Figure 9.5: Plots of the actual pitch shifting factor β' , overlaid by the theoretic pitch-shifting factor β

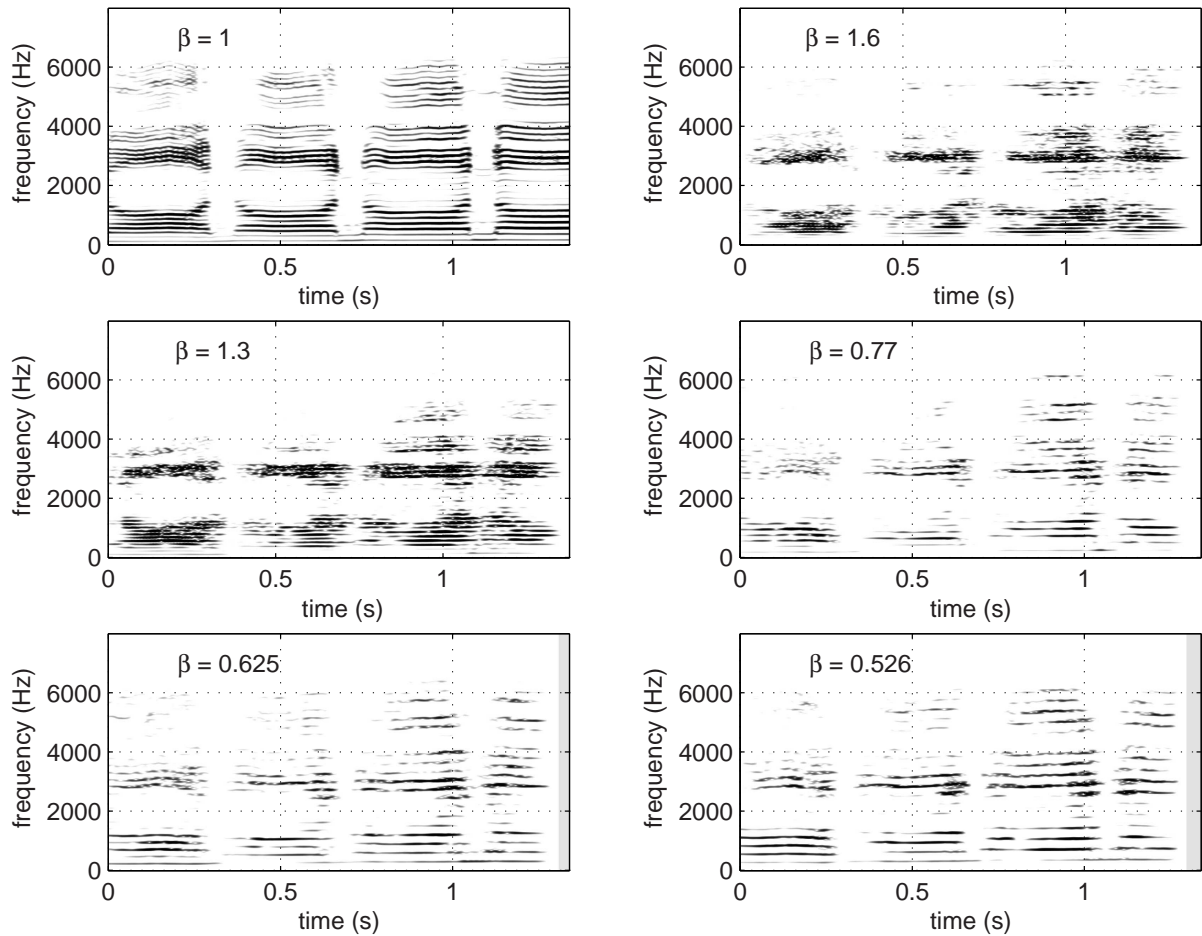


Figure 9.6: Spectrograms of pitch-shifted signals

the naturalness of the sound. We performed a series of pitch-shifts on a vocal recording of four notes sung by a male vocalist, using a constant vowel. Figure 9.6 shows spectrograms of the results, with $\beta = 1$; the original signal. These spectrograms are presented so that the pitch and the formant locations could be observed. From the figures we see that although the shape of the formants change a bit, they stay in the correct locations despite the change in pitch, which is a necessity for a true pitch-shifting algorithm.

9.3.2 Results from pitch-shifting applications

In Chapter 7 we introduced several classes of pitch-shifting. Some formulae for β , the shifting factor, are given in Table 7.1 (p72). These types of rules specify that β need not remain constant for the duration of the signal. We show the results from two such examples:

1. For the first example we perform pitch correction as described in section 7.5. Figure 9.7 shows pitch-tracks of $x(n)$, the original signal, and $y(n)$, the pitch-shifted

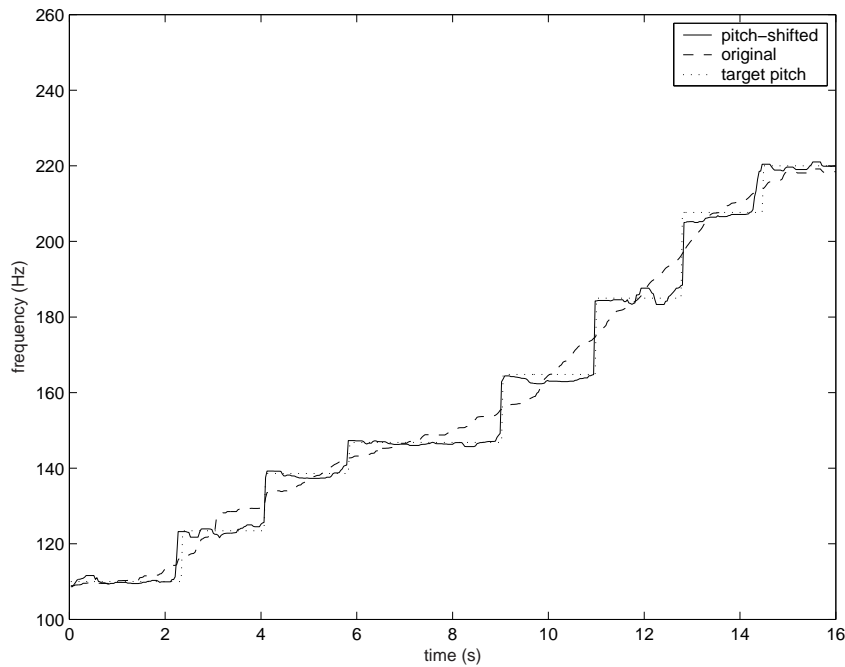


Figure 9.7: Pitch correction

version. They are overlaid by the target pitch frequencies, which are the values the original pitch-tracks are “drawn” to. The result is a frequency quantisation, or in music technological terms: pitch correction. A closer look at Figure 9.7 will inform the reader that the result is a C-major scale, created from a random vocal “scoop”.

2. For the second example we choose $\zeta = 2$ in Table 7.1, delivering a third harmony above the source vocal. Observe the pitch-tracks in Figure 9.8. The harmonies run parallel for most of the signal except for the note occurring after about nine seconds from the start, where it is obvious that β has a different value. The artificial signal is a complementing melody to the original that cannot be obtained by a constant value for β .

9.4 Synthesis

The success of the synthesis procedure lies in the continuity of the result. The time domain waveform may not have discontinuities since it will manifest as clicking sounds. The phase correction and the windowed overlap-and-adding process, described in Chapters 4 and 7, give us a smooth result with very little notion of frames being added together. The seamlessness of the synthesis is hard to perceive without listening to the results, but in Figure 9.9 we show an excerpt from a synthetic waveform, overlaid by lines indicating the locations of $t_s(u)$, the *synthesis* time instants. These markings are the locations where

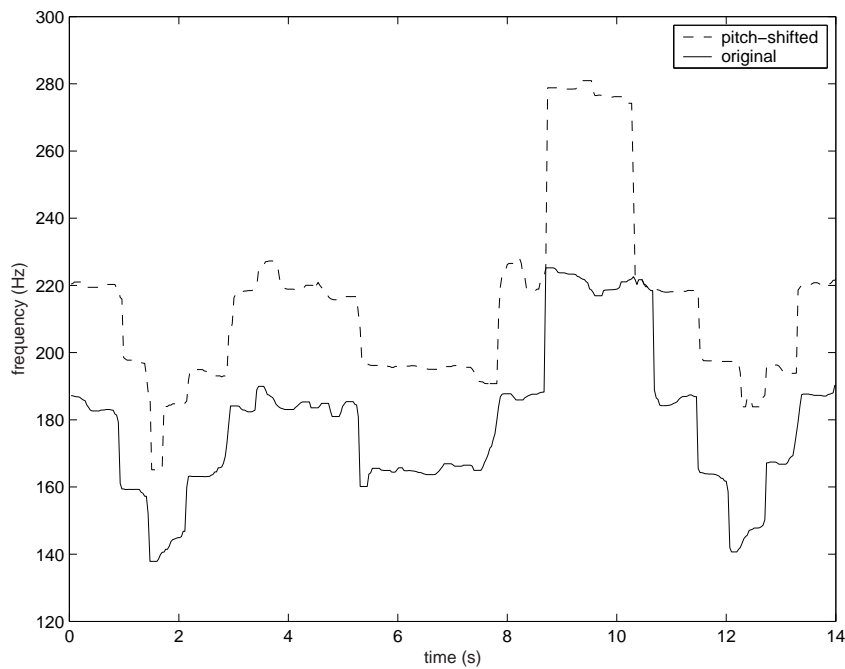


Figure 9.8: Harmonisation

a new window starts. If the synthesis process had discontinuities, these would be the locations where they occur. Figure 9.9 proves that the result is a continuous waveform.

9.5 High level evaluation

We said in section 9.1 that the ultimate measurement is the pleasantness and the natural character of the synthetic signal. We found that these are achieved to a certain extent. The results are not good enough for solo performances, but work very well for backing vocals, as long as they are not competing in volume with the main voice.

The wave synthesis procedure in section 8.2 works quite well and one gets the impression that the result is human-like. One flaw however, is that the transients between notes do not sound human-like. When a human voice advances from one note to the next, it entails a complex transient process, beyond our scope of modelling.

Generally, the gender transformation does not sound very natural. We found that it does succeed when a person sings in the range of the opposite sex. Among other trials, we tested the algorithm on a recording done by a female vocalist singing in the range of a tenor. When gender transformed, we achieved a nearly natural sound, reminiscent of a bass vocalist.

(The reader may refer to the included HTML document which provides a variety of audio examples)

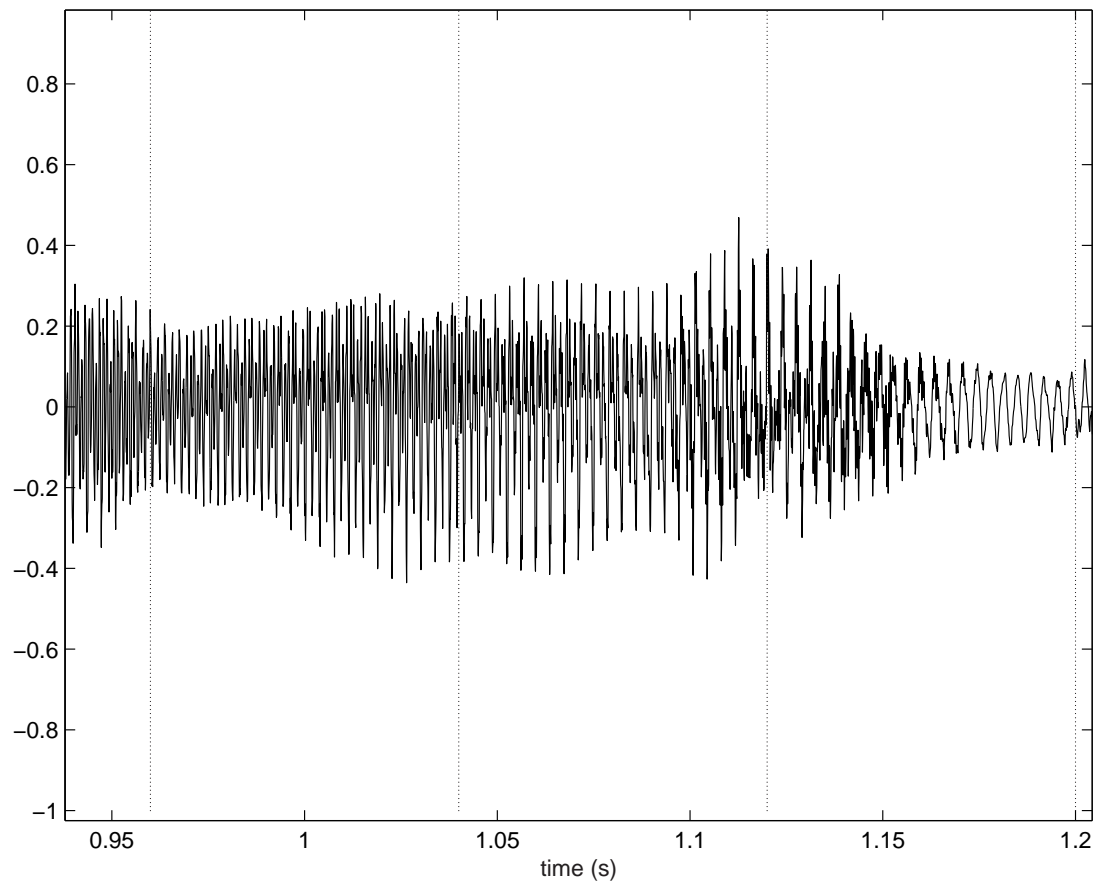


Figure 9.9: Synthesised waveform

Chapter 10

Final comments

10.1 Conclusions

In the foregoing chapters we set out to design an algorithm that controls the pitch of a singing voice while introducing as little as possible artifacts. This problem not only concerns the fundamental frequency control, but also control of the formants. This led to an investigation of spectral modelling. We implemented algorithms that aim to solve these problems and evaluated them with a measure of success.

Throughout the development it became apparent that the scope of the problem is immense and that the solution is not a trivial one. The techniques we suggested give satisfactory results although the artifacts are audible in some cases. The fragility of the voice becomes apparent as soon as we make rather large changes, like:

- Shifting the pitch too far from the original. Pitch-shifts of more than ± 4 semitones begin to sound unnatural.
- Shifting the formants too far. In the section on gender transformations we implemented a 25% shift in the formants as suggested in [18]. Our algorithm is not quite up to a shift of that calibre and the results sound a bit “tampered with”.

As side topics to the main problem, we solved another problem successfully: we designed a robust pitch detection technique of high accuracy that specialises in singing voice. A powerful smoother, usually found in graphical applications, was used to gain insight in the spectral behaviour of the singing voice.

Employing the above techniques, we designed a successful system that creates waveforms that imitate a person singing a constant vowel at different pitches. The pitch can be controlled from a MIDI-keyboard while the actual vowel is based on a short recording. Another application was a suggestion on how to transform a voice to that of the opposite sex.

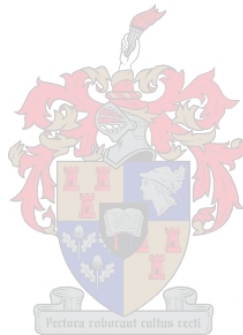
10.2 Future work

By no means have we designed a complete pitch and formant control system. There are many shortcomings that could be investigated by subsequent research efforts:

- Currently the system needs a scale as input when we want to synthesise a harmonious signal. It would be very useful if the system could detect the scale from the input signal.
- The pitch-tracking algorithm is specialised for singing voices and does not work well for normal speech. An investigation on how to modify the algorithm for speech would prove useful in the field of speech processing.
- The pitch-shifting technique “locks” the formants in their original locations and they do not move, no matter how much the pitch is shifted. This is correct in theory, but in practice certain vowel substitutions occur (as mentioned in Chapter 2). A pitch-shifting algorithm that includes these considerations may generate a more natural sounding effect.
- The system currently works off-line. A real-time implementation would be very useful. If such a system could be devised, a performer could have backup vocals on stage - done by a computer.
- For a pitch-shifted version of an original signal, the system uses a numerical representation of the excitation. This representation contains errors caused by the resampling process we introduced in Chapter 6. An investigation of a parametric excitation signal (used in conjunction with the formant modelling techniques we described) may lead to more natural sounding results.
- During the current gender transformation process the formants are moved by a constant shift. From Table 2.2 (p14) it is evident that the formants locations of a female singing voice do not merely constitute a constant shift of the male locations. We approximated these differences by a constant shift of 25%. An algorithm that shifts formants independently may give the resulting voice a more natural feel.
- A representative corpus of singing voices should make the results more meaningful. As it is, the system may specialise on the few voices it was tested on. A research effort may be launched to gather samples of:
 - Different vowels
 - Different sexes
 - Different voice types like bass, tenor, soprano, etc.

10.3 Last thoughts

The digital manipulation of a singing voice is a process that needs careful thought. The human voice is a beautiful instrument and any uncalculated digital alterations to its sound can do a lot of harm. When an unsuspecting listener cannot hear that the voice was altered we believe that the algorithm is successful. The reader may reference to included HTML document with the audio examples and challenge each example with the question: “Does this sound truly natural?”. We feel that the harmony examples passes the test as well as the phase vocoder examples. The fact that some complex algorithms which received a lot of thought and input fail this test, made us aware of how complex the waveform of the human voice is and we learned that the voice, especially singing, should not be taken for granted. Finally, we conclude that the digital manipulation of a singing voice is like adding another story to a card house: Improvement is possible, but digital gravity is strong!



Bibliography

- [1] “Music Scales.” <http://tyala.freeyellow.com/t4scales.htm>.
- [2] DE GOTZEN, A., BERNARDINI, N., and ARFIB, D., “Traditional Implementations of Phase Vocoder: The Tricks of the Trade.” *Proceedings of the COST G-6 Conference on Digital Audio Effects*, 2000.
- [3] DELLER, J., PROAKIS, G., and HANSEN, H., *Discrete-Time Processing of Speech Signals*. New York: Macmillan, 1993.
- [4] FLANAGAN, J. and GOLDEN, R., “Phase Vocoder.” *Bell System Technical Journal*, 1966, pp. 1493–1509.
- [5] GONZALEZ, R. and WOODS, R., *Digital Image Processing*. Second edition. New Jersey: Prentice-Hall International, 2002.
- [6] KOOPMAN, J., *A Brief History of Singing*. 1995.
- [7] LAROCHE, J. and DOLSON, D., “Phase-vocoder: About this phasiness business.” *Proc. IEEE Workshop on Applications of Signal Processing to audio and Acoustics*, 1997.
- [8] LAROCHE, J. and DOLSON, D., “Improved phase-vocoder time-scale of audio.” *IEEE Trans. Speech and Audio Processing*, May 1999.
- [9] LAROCHE, J. and DOLSON, D., “New Phase-Vocoder Techniques for Pitch-Shifting, Harmonizing and Other Exotic Effects.” *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1999.
- [10] MAKHOUL, J., “Linear prediction: A tutorial review.” *Proc. IEEE*, 1975, Vol. 63, pp. 561–580.
- [11] MARQUARDT, A., TOERIEN, L., and TERBLANCHE, E., “Applying Nonlinear Smoothers to Remove Impulsive Noise from Experimentally Sampled Data.” *University of Stellenbosch Applied Mathematics Research and Development Journal*, 1991, pp. 15–18.

- [12] MOULINES, E. and LAROCHE, J., “Non-Parametric Techniques for Pitch-Scale Modification of Speech.” *Speech Communication*, 1995, Vol. 16, pp. 175–205.
- [13] PORTNOFF, M., “Implementation of the Digital Phase Vocoder Using the Fast Fourier Transform.” *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1976, Vol. 24, pp. 243–248.
- [14] PROAKIS, J. and MANOLAKIS, D., *Digital Signal Processing: Principles, Algorithms, and Applications*. Third edition. New Jersey: Prentice-Hall International, 1996.
- [15] PUCKETTE, M., “Phase-locked Vocoder.” *IEEE ASSP Conference on Applications of Signal Processing to Audio and Acoustics*, 1995.
- [16] RHOWER, C., “Idempotent One-Sided Approximation of Median Smoothers.” *Journal of Approximation Theory*, 1989, Vol. 58.
- [17] RHOWER, C., “Variations and LULU-smoothing.” 1991.
- [18] ROSSING, T., *The Science of Sound*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1982.
- [19] SCHAFER, R. and RABINER, L., “Design and simulation of a speech analysis-synthesis system based on short-time Fourier analysis.” *IEEE trans. Audio Electroacoustics (Special issue on 1972 conference on Speech Communication Processing)*, 1973, Vol. AU-21, pp. 165–174.
- [20] SEUL, M., O’GORMAN, L., and M.J., S., *Practical Algorithms for Image Analysis: Description, Examples, and Code*. Cambridge: Cambridge University Press, 2000.

Appendix A

Linear Prediction Analysis

Linear prediction is a technique for modelling discrete time signals. We assume that each sample can be predicted from a weighted sum of p preceding samples.

Let $\hat{s}[k]$ bet an estimate of $s[k]$, the actual sample. Now linear prediction can be expressed as:

$$\hat{s}[k] = \sum_{i=1}^p a_i s[k - i] \quad (\text{A.1})$$

. The error between the actual signal and the predicted signal is :

$$\begin{aligned} e[k] &= s[k] - \hat{s}[k] \\ \Rightarrow s[k] &= \sum_{i=1}^p a_i s[k - i] + e[k]. \end{aligned} \quad (\text{A.2})$$

Now we take the \mathcal{Z} -transform to calculate the transfer function:

$$\begin{aligned} \mathcal{Z}(s[k]) &= \sum_{i=1}^p a_i z^{-i} S[z] + E[z] \\ \Rightarrow H(z) &= \frac{S(z)}{E(z)} \\ &= \frac{1}{1 - \sum_{i=1}^p a_i z^{-i}}. \end{aligned} \quad (\text{A.3})$$

The transfer function, $H(z)$, may be viewed as an all-pole LTI system, with the prediction error, $E(z)$, as input.

Suppose we have the coefficients $a_1 \cdots a_p$, we may measure how well the model describes the actual signal by calculating the mean-square-error between the two signals.

$$e[k] = s[k] - \hat{s}[k] \quad (\text{A.4})$$

The total mean-square-error over the segment $s[0] \cdots s[N-1]$ is:

$$E = \sum_{n=0}^{N-1} e^2[n] = \sum_{n=0}^{N-1} \left(s[n] - \sum_{i=1}^p a_i s[n-i] \right)^2. \quad (\text{A.5})$$

This a quadratic equation in terms of $a_1 \cdots a_p$. By minimising this function, we can calculate a formula that will give the optimal values of $a_1 \cdots a_p$. To get this, we take the partial derivative of E with respect to a_j , set it equal to zero and solve.

$$\begin{aligned} \frac{\partial E}{\partial a_j} &= 0 \\ &= \sum_{n=0}^{N-1} \left[2 \left(s[n] - \sum_{i=1}^p a_i s[n-i] \right) \left(-s[n-j] \right) \right] \\ &= 2 \sum_{n=0}^{N-1} s[n] s[n-j] + 2 \sum_{n=0}^{N-1} \sum_{i=1}^p a_i s[n-i] s[n-j] \end{aligned} \quad (\text{A.6})$$

$$\Rightarrow \sum_{n=0}^{N-1} s[n] s[n-j] = \sum_{n=0}^{N-1} \sum_{i=1}^p a_i s[n-i] s[n-j]. \quad (\text{A.7})$$

Now we have p simultaneous equations, called the normal equations.

Let:

$$\phi_{i,j} = \sum_n s[n-i] s[n-j]. \quad (\text{A.8})$$

Now we can write the normal equations as:

$$\phi_{i,j} = \sum_{j=1}^p \phi_{i,j} a_j \quad i = 1, 2, \cdots p. \quad (\text{A.9})$$

In matrix form:

$$\begin{bmatrix} \phi_{1,0} \\ \phi_{2,0} \\ \vdots \\ \phi_{p,0} \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,p} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{p,1} & \phi_{p,2} & \cdots & \phi_{p,p} \end{bmatrix}}_{\mathbf{\Phi}}$$

$\mathbf{\Phi}$ is symmetric since $\phi_{i,j} = \phi_{j,i}$.

We may now solve the equations by means of the Autocorrelation Method. This method assumes that the signal is zero outside the bounds of the analysis frame:

$$\phi_{i,j} = \sum_{n=0}^{N-1-(i-j)} s[n] s[n+(i-j)] \quad i \geq j. \quad (\text{A.10})$$

The unbiased autocorrelation estimate for a finite data window is:

$$r_k = \sum_{n=0}^{N-1-k} s[n]s[n+k] \quad k \geq 0 \quad (\text{A.11})$$

$$\Rightarrow \phi_{i,j} = r_{i-j}. \quad (\text{A.12})$$

The normal equations may now be written as:

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{bmatrix} = \underbrace{\begin{bmatrix} r_0 & r_1 & \dots & r_{p-1} \\ r_1 & r_0 & \dots & r_{p-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & \dots & r_0 \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}$$

\mathbf{R} is a highly symmetrical form known as *Toeplitz*.

The above set of linear equations can be solved recursively by a method called *Levinson Durbin recursion*. It is referred to as a recursive-in-model-order solution for the autocorrelation equations. This means the solution for the desired order, say M, is successively build up from lower order models, beginning with an order 0 predictor. We can express the *Levinson Durbin recursion* as follows:

Initialisation: $E_0 = r_0$

Recursion for $i = 1, 2, 3 \dots p$:

$$r_i - \sum_{j=1}^{i-1} a_{i-1}(j)r(i-j)$$

- $k_i = \frac{\quad}{E_{i-1}}$
- $E_i = (1 - k_i^2) E_{i-1}$
- $a_i(i) = k_i$
- Cycle through $j = 1, 2, 3 \dots i - 1$:
 $a_i(j) = a_{i-1}(j) - k_i a_{i-1}(i - j)$

End of current recursion.

The linear prediction coefficients are: $a_p(1), a_p(2), \dots, a_p(p)$, and may be used in equation A.3. This filter will approximate the behaviour of $s[k]$, the signal portion it was derived from.

Appendix B

Basic Perspective on Musical Scales

The most basic concept behind music theory is a note. A note is a frequency kept constant for a certain duration. The notes on a piano is a geometric series, starting at $27.5Hz$ and increasing with a factor $2^{\frac{1}{12}}$ ending at $4186Hz$. We denote the lowest note on the piano A_0 . Equation B.1 is the sequence of frequencies of the piano notes.

$$A_0, A_0(2^{\frac{1}{12}}), A_0(2^{\frac{1}{12}})^2, \dots, A_0(2^{\frac{1}{12}})^{85}, A_0(2^{\frac{1}{12}})^{86}, A_0(2^{\frac{1}{12}})^{87} \quad (\text{B.1})$$

This type of tuning, which nowadays is standardised, is called *equal tempered* tuning. The frequency will double every 12 notes, irrespective of what note is started on. We refer to $A_0(2^{\frac{1}{12}})^{12}$ as A_1 , indicating the same note, but one octave higher, i.e. double the frequency. We use the letters A to G to reference notes and they can be followed by an integer to indicate the octave, e.g. A_0, A_4 , etc. The symbols \sharp and \flat are used in combination with a letter to make up the eleven symbols needed to denote an octave. They indicate one note higher or lower respectively and are pronounced *sharp* and *flat*. We expressed an octave as:

$$A, B\flat, B, C\sharp, C, D, E\flat, E, F, F\sharp, G, G\sharp$$

On its own, a note doesn't do much - we need a sequence of notes to form a melody which is a sequence of notes, usually from the same scale. A scale is a certain path through the notes on a piano, increasing in frequency from any given note to its octave. These notes are defined as the "valid" notes for a certain melody. We usually define them for one octave, but it applies to any other octave used in the melody.

A scale is referenced by its *root note* and by the type of scale. The most important scale types are *major*- and *minor*-scales but there exists a vast variety of other scales not mentioned here.

Another important point is that melodies do not necessary belong to single scale and can be theoretically complex. See [1] for more information on scales.

The scale type is a certain path through an octave. The *major*-scale uses the following note jumps on a piano:

$$\text{root } 2 \ 2 \ 1 \ 2 \ 2 \ 2 \ 1,$$

where 1 denotes the next note and 2 a skipped note. A *minor*-scale's sequence is:

$$\text{root } 2 \ 1 \ 2 \ 2 \ 1 \ 2 \ 2.$$

Figure B.1 illustrates the notes of a *C major* scale. It is convenient to work with this scale since it has no *sharps* or *flats*, which are the black notes on the piano.

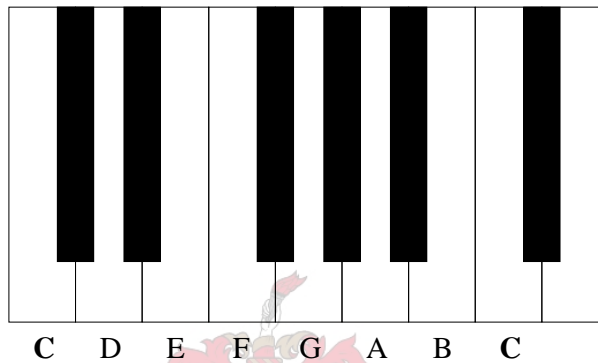


Figure B.1: A piano octave

To play a tune in a certain key, means to play it in a scale with a specified root. We can therefore play the same tune in different keys. The tune determines the scale type while the musician may choose the key.

If two melodies are played simultaneously, they have to be at least in the same key to sound correct to a trained ear. There exists endless types of harmonies to a single tune, but there are a few well known and well used ones. The most basic harmony is called a *third*-harmony and is a parallel harmony running two steps higher or lower than the original melody, in the a major scale. The term *third* refers to the fact that the third note in the scale is used to harmonise with the *root* note. It is important to note that a single step leads to the next note in the scale and not necessary the next note on the piano. The *third*-harmony is used very often in rock, pop and other modern day music styles and gives a “warm” feeling to either *major* or *minor* based melodies. Other harmonies are *fifths* and *octaves*, where the name also indicates which note in the scale is used together with the root.

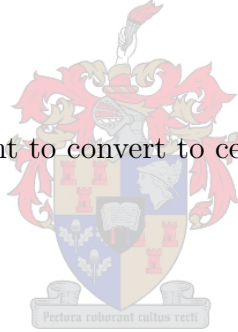
Appendix C

Pitch Cents

Cents is a logarithmic representation of musical pitch and is used to present absolute pitch in terms of a hundred cents per semitone with respect to middle C. Middle C is designated zero cents. Hence, A4 is 900 cents and A3 -300 cents. Since notes increase exponentially on the piano, a logarithmic representation enables us to view any note as a linear increase from the previous one. The formula for calculating cents from Hertz is:

$$c = 1200 \log_2 \frac{f}{f_C}, \tag{C.1}$$

where f is the frequency we want to convert to cents, and f_C is the frequency of middle C on the piano.



Appendix D

Autocorrelation function pitch detection methods

D.1 Autocorrelation Basics

In the literature there are many well documented techniques for pitch detection. Many of these are based on a mathematical operation known as autocorrelation.

The autocorrelation function is defined by

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n-l),$$

where $x(n)$ is a discrete time signal of length N and l is a discrete time lag. This operation corresponds to “sliding” one copy of the signal over another, multiplying matching elements and adding results to previous ones for each lag instant. The result is a signal of length $2N + 1$. In the case where $x(n)$ has strong periodicity, $r_{xx}(n)$ contains impulses with a spacing equal to $\frac{1}{F_0}$, where F_0 is the fundamental frequency of $x(n)$ - illustrated by Figures D.1 and D.2.

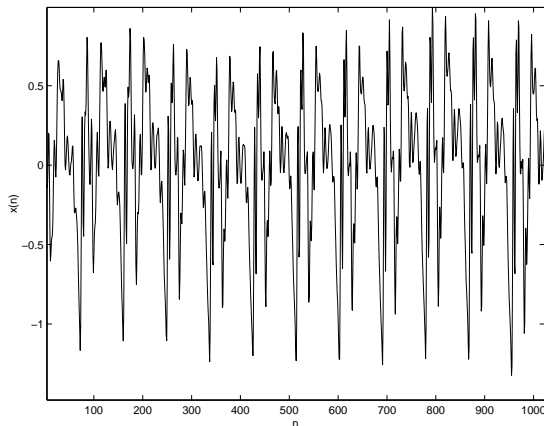


Figure D.1: Periodic time signal

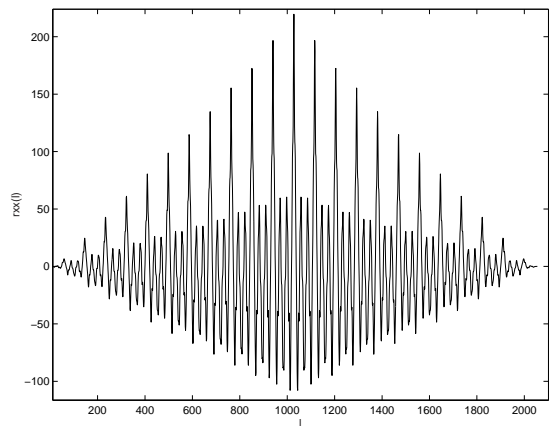


Figure D.2: Autocorrelation of time signal

The simplest pitch detection AC-method works as follows:

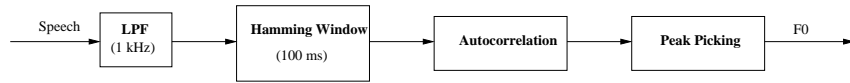


Figure D.3: Autocorrelation-based pitch detecting system

The idea would work well for the signal in Figure D.1 but would fail in cases where F_0 is weak, where the first formant has a peak close to F_0 or where the periodicity is not strongly maintained throughout the segment. The latter makes it very hard to find the peaks.

D.2 Improving on Basic Autocorrelation Method

There are several remedies that can be applied to enhance pitch detection with the autocorrelation function:

- By raising the signal to a higher power, while preserving the sign, the high amplitude portions of the signal will be stronger emphasised. This is useful since these regions occur at the start of a pitch period. See Figure D.4 below.

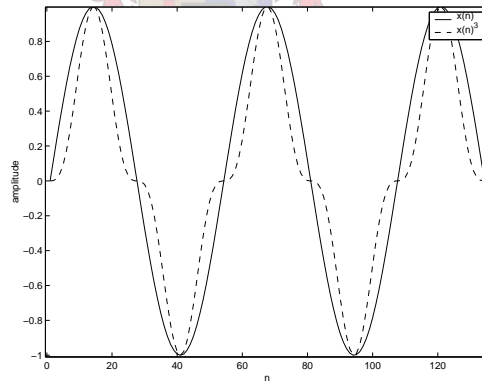


Figure D.4: A signal and its power-raised version

- The high amplitude portions of the signal $x(n)$ can also be emphasised by centre clipping the signal, i.e. the portions of $x(n)$ below a certain threshold becomes zero.

$$x'(n) = \begin{cases} x(n) - T & \text{if } x(n) > T \\ 0 & \text{if } -T < x(n) < T \\ x(n) + T & \text{if } x(n) < -T \end{cases}$$

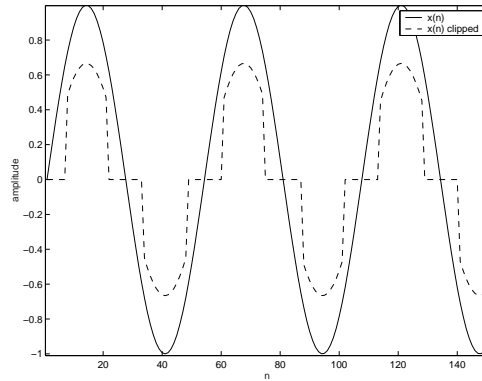


Figure D.5: A signal and its centre-clipped version

- Another useful clipping scheme is called “Tertiary Clipping”. This reduces the signal to three levels: -1 , 0 and 1 . This greatly reduces the autocorrelation complexity, since floating point multiplication is not needed.

$$x'(n) = \begin{cases} 1 & \text{if } x(n) > T \\ 0 & \text{if } -T < x(n) < T \\ -1 & \text{if } x(n) < -T \end{cases}$$

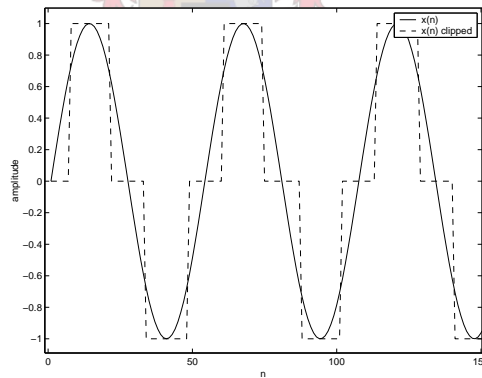


Figure D.6: A signal and its tertiary-clipped version

D.3 Average magnitude difference function (AMDF)

This is a modified autocorrelation function and has no multiplications, making it computationally very efficient. The AMDF is defined by:

$$d[k] = \sum_{n=0}^{N-1} |x_w[n] - x_w[n+k]|$$

where $x_w[k]$ is a windowed speech frame. It is clear that when $k \rightarrow 1/F_0$, $d[k]$ will exhibit a minimum. Pitch detection will consist of finding the valley in the AMDF. Figure D.7 illustrates the AMDF of the signal in Figure D.1. One can clearly see the minimums are spaced equally at the pitch period, delivering results that are similar to the autocorrelation function in Figure D.2.

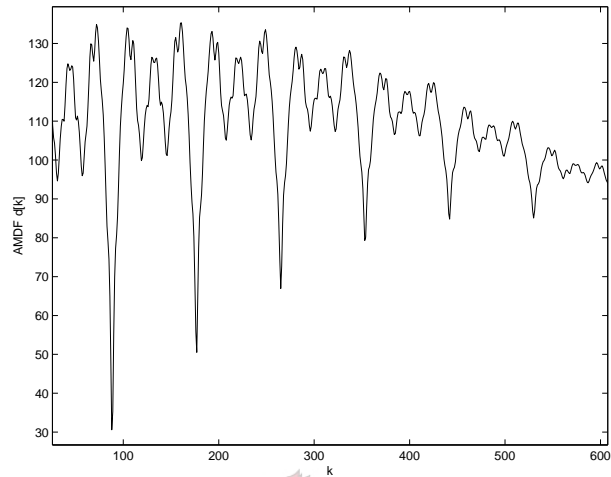


Figure D.7: AMDF

