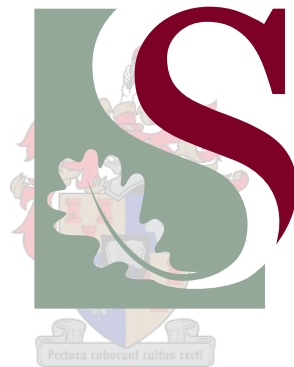


Speech Generation in a Spoken Dialogue System

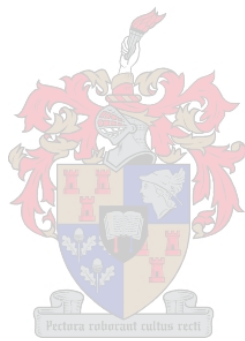
ALBERT S. VISAGIE



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

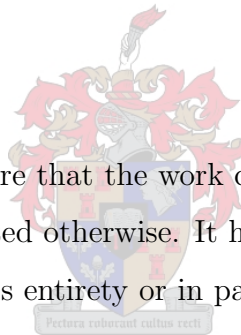
SUPERVISOR: Prof J.A. du Preez

December 2004



Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, except where stated otherwise. It has not been previously submitted to any University for a degree in its entirety or in part.



SIGNATURE

DATE

Abstract

Spoken dialogue systems accessed over the telephone network are rapidly becoming more popular as a means to reduce call-centre costs and improve customer experience. It is now technologically feasible to delegate repetitive and relatively simple tasks conducted in most telephone calls to automatic systems. Such a system uses speech recognition to take input from users. This work focuses on the speech generation component that a specific prototype system uses to convey audible speech output back to the user.

Many commercial systems contain general text-to-speech synthesisers. Text-to-speech synthesis is a very active branch of speech processing. It aims to build machines that read text aloud. In some languages this has been a reality for almost two decades. While these synthesisers are often very understandable, they almost never sound natural. The output quality of synthetic speech is considered to be a very important factor in the user's perception of the quality and usability of spoken dialogue systems.

The static nature of the spoken dialogue system is exploited to produce a custom speech synthesis component that provides very high quality output speech for the particular application. To this end the current state of the art in speech synthesis is surveyed and summarised. A unit-selection synthesiser is produced that functions in Afrikaans, English and Xhosa.

The unit-selection synthesiser selects short waveforms from a recorded speech corpus, and concatenates them to produce the required utterances. Techniques are developed for designing a compact corpus and processing it to produce a unit-selection database. Speech modification methods were researched to build a framework for natural-sounding speech concatenation. This framework also provides pitch and duration modification capabilities that will enable research in languages such as Afrikaans and Xhosa where text-to-speech capabilities are relatively immature.

Opsomming

Telefoniese, spraakgebaseerde dialoogstelsels word steeds meer algemeen, en is 'n doeltreffende metode om oproepsentrumkoste te verlaag. Dit is tans tegnologie moontlik om 'n groot aantal eenvoudige transaksies met automatiese stelsels te hanteer. Sulke stelsels gebruik spraakherkenning om intree van die gebruiker te ontvang. Hierdie werk fokus op die spraakgenerasiekomponent wat 'n spesifieke prototipestelsel gebruik om afvoer aan die gebruiker terug te speel.

Vele kommersiële stelsels gebruik generiese teks-na-spraak sintetiseerders. Sulke teks-na-spraak sintetiseerders is steeds 'n baie aktiewe veld in spraaknavorsing. In die algemeen poog navorsing om teks te kan lees en om te sit in verstaanbare spraak. Sulke stelsels bestaan nou al vir ten minste twee dekades. Alhoewel heeltemal verstaanbaar, klink hierdie stelsels onnatuurlik. In telefoniese spraakgebaseerde dialoogstelsels is kwaliteit van die sintetiese spraak belangrik vir die gebruiker se persepsie van die stelsel se kwaliteit en bruikbaarheid.

Die dialoog is meestal staties van aard en hierdie eienskap word benut om hoë kwaliteit spraak in 'n bepaalde toepassing te sintetiseer. Om dit reg te kry is die huidige stand van sake in hierdie veld bestudeer en opgesom. 'n Knip-en-plak sintetiseerder is gebou wat werk in Afrikaans, Engels en Xhosa.

Die sintetiseerder selekteer kort stukkies spraakgolfvorms vanuit 'n spraakkorpus, en las dit aanmekaar om die vereiste spraak te produseer. Outomatiese tegnieke is ontwikkel om 'n kompakte korpus te ontwerp wat steeds alles bevat wat die sintetiseerder sal nodig hê om sy taak te verrig. Verdere tegnieke prosesseer die korpus tot 'n bruikbare vorm vir sintese.

Metodes van spraakmodifikasie is ondersoek ten einde die aanmekaargelaste stukkies spraak meer natuurlik te laat klink en die intonasie en tempo daarvan te korrigeer. Dit verskaf infrastruktuur vir navorsing in tale soos Afrikaans en Xhosa waar teks-na-spraak vermoëns nog onvolwasse is.

Acknowledgements

I would like to thank these wonderful people for their direct and indirect inputs to this work:

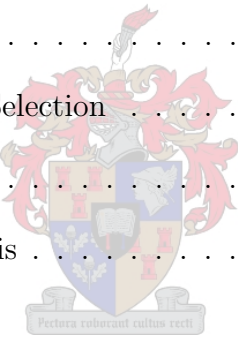
- My parents, for their lovingly given (and seemingly endless) love, financial and moral support.
- Elana, for her love and my sanity.
- My supervisor, for his patience and many stimulating conversations.
- DSP Lab members, present and past, for making coffee, keeping the network running, being productive and merciless sounding boards and for providing all the distractions that helped make the experience very worthwhile.
- Many friends who shared many good times.



Contents

1	Introduction	1
1.1	Artificial Speech	1
1.2	Motivation for this Study	2
1.3	Background	4
1.3.1	Verbal Communication	4
1.3.2	Speech Synthesis	7
1.4	Objectives	11
1.5	Contributions	11
1.6	Overview and Organisation of this Work	12
2	Waveform Synthesis by Unit Selection	14
2.1	Introduction	14
2.2	Indexing the Corpus	17
2.3	Corpus Design	20
2.3.1	Text Selection	21
2.4	Unit Selection	26
2.4.1	Search	26
2.4.2	Candidates	27
2.4.3	Concatenation and Target Costs	32
2.4.4	Optimisations	34
2.5	Unit Concatenation	35

2.6	Implementation	36
2.6.1	Text Processing	37
2.6.2	Corpus Design	37
2.6.3	Processing the Synthesis Database	41
2.6.4	Specialisation of Unit-Selection Synthesis	48
2.6.5	Results	52
2.7	Conclusions & Suggestions	57
3	Speech Modification	60
3.1	Introduction	60
3.2	Literature Overview	61
3.2.1	Signal Components	62
3.2.2	Speech Modification	64
3.2.3	Spectral Smoothing for Concatenative Synthesis	73
3.3	Implementation	76
3.3.1	Analysis	76
3.3.2	Synthesis	77
3.4	Evaluation	83
3.4.1	Isolated Spectral Smoothing Experiments	83
3.4.2	Making Pitch Targets	83
3.4.3	Pitch-mark Location	87
3.5	Conclusions & Suggestions	89
4	Pitch Determination	92
4.1	Introduction	92
4.2	Defining Pitch	93
4.3	Overview of Pitch Tracking Methods	94
4.3.1	Pitch Feature Extraction	95

4.3.2	Post-processing of Pitch Features	101
4.4	Implementation	103
4.4.1	Pitch Feature Extraction	104
4.4.2	Post-processing of Pitch Features	107
4.5	Experiments	112
4.5.1	Test Methodology	112
4.5.2	Results	115
4.6	Conclusions & Suggestions	115
5	Conclusion & Suggestions	118
5.1	Future Work	119
5.1.1	Intonation	119
5.1.2	Improved Unit Selection	120
5.1.3	Novel words	120
5.1.4	General Synthesis	121
		
	Bibliography	122
A	Hidden Markov Models	134
A.1	Introduction	134
A.2	Speech Recognition	136
A.3	Features	137
A.4	Training	137
A.4.1	Simplified MAP	140
A.4.2	Embedded Re-estimation	141
A.5	Forced Alignment	142
A.6	Symbols	144

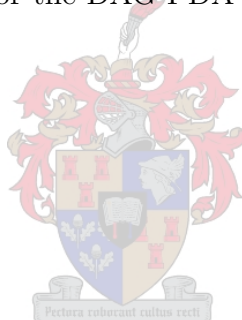
List of Figures

1.1	General architecture of spoken dialogue systems	2
1.2	Data-flow in speech synthesis	8
2.1	The unit-selection process	16
2.2	The general greedy algorithm for set-covering	23
2.3	The distribution of the diphones in the Project Gutenberg set of sentences	25
2.4	The distribution of the triphones in the Project Gutenberg set of sentences	26
2.5	An example of the phonological structure in PSM	28
2.6	Querying a decision tree for unit clusters	31
2.7	The introduction of an artificial <i>glue word</i>	40
2.8	The distribution of word bigrams for the HRS dialogue	41
2.9	Phonemic alignment of recorded speech using a speech synthesiser and DTW	43
2.10	Merging vowel models	45
2.11	Alignment results for the English HRS corpus	46
2.12	Alignment results for the Xhosa HRS corpus	47
2.13	Synthesis Example 1	54
2.14	Synthesis Example 1: Detail of “February”	55
2.15	Synthesis Example 2	56
2.16	Synthesis Example 3	56
2.17	Synthesis Example 4: Time-domain signal	57
3.1	Views of a speech signal	63

3.2	Taxonomy of speech modification methods	64
3.3	Mapping analysis time to synthesis time	68
3.4	The Fourier transform of a short-time signal	71
3.5	The linear prediction residual	73
3.6	PSOLA on the LP residual	74
3.7	Smoothing concatenations	81
3.8	Formant movements in smoothed concatenations	81
3.9	Spectrogram of a synthesised nonsense word with smoothing	84
3.10	Smoothed pitch contour	85
3.11	Manually corrected pitch contour	86
3.12	Synthesis Example 3: Spectrogram	87
3.13	The effect of pitch-mark location on LP synthesis	88
3.14	The effect of pitch-mark location on the LP residual	89
4.1	Segment of speech showing a voiced and an unvoiced phone	93
4.2	General diagram of PDAs	95
4.3	Autocorrelation of voiced and unvoiced segments of speech	97
4.4	The Directed Acyclic Graph Pitch-tracker	103
4.5	The band-pass filtered energy contour	104
4.6	Computing the Harmonic Product Spectrum	106
4.7	Pitch information cast as a weighted graph	107
4.8	Schematic representation of weight assignment to edges	108
4.9	Connecting the vertices	109
4.10	Best-path search through the DAG	111
A.1	A three-state phoneme model.	135
A.2	A simple compound HMM to recognise connected speech.	136
A.3	An utterance model for embedded re-estimation or forced alignment.	142

List of Tables

2.1	Selection statistics for limited-domain fillers	40
2.2	The hierarchical naming scheme for unit names	50
3.1	Smoothing time constants for voiced phonemes	80
4.1	Pitch-tracking results for the DAG PDA	116



List of Acronyms

AMDF	Amplitude Magnitude Difference Function
ASR	Automatic Speech Recognition
CART	Classification and Regression Trees
DAG	Directed Acyclic Graph
DFT	Discrete Fourier Transform
DP	Dynamic Programming
DSP	Digital Signal Processing
DTFT	Discrete-Time Fourier Transform
DTMF	Dual Tone Multi-Frequency
DTW	Dynamic Time-Warping
DWT	Discrete Wavelet Transform
EM	Expectation Maximisation
eSRPD	Enhanced Super-Resolution Pitch Determinator
FD	Frequency-domain
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HNM	Harmonics plus Noise Model
HRS	Hotel Reservation System
GCI	Glottal Closure Instances
HPS	Harmonic Product Spectrum
HSS	Harmonic Sum Spectrum
IPA	International Phonetics Association

LAR	Log-Area Ratios
LDOM	Limited Domain Synthesis
LNRE	Large Number of Rare Events
LP	Linear Prediction
LPCs	Linear Prediction Coefficients
LTS	Letter-to-Sound
MAP	Maximum <i>a Posteriori</i>
ML	Maximum Likelihood
MSE	Mean Squared Error
NLG	Natural Language Generation
OLA	Overlap-Add
PDA	Pitch Determination Algorithm
PSOLA	Pitch-Synchronous Overlap-Add
PDF	Probability Density Function
PSM	Phonological Structure Matching
SDS	Spoken Dialogue Systems
SNR	Signal-to-Noise ratio
SRPD	Super-Resolution Pitch Determinator
STFT	Short-time Fourier Transform
TD	Time-domain
TTS	Text-to-Speech Synthesis
V/U	Voiced/Unvoiced
WFST	Weighted Finite-State Transducers

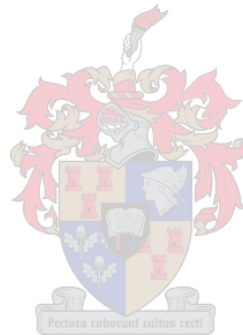
List of Symbols

$\lceil x \rceil$	The smallest integer that is greater than x . (<code>ceil(double)</code> in the C programming language)
$\lfloor x \rfloor$	The largest integer that is smaller than x . (<code>floor(double)</code> in the C programming language)
a_i	The i 'th filter coefficient of a linear prediction filter $H(z)$.
A_i	The surface area of the openings at the ends of a tube in the simplified tube model of speech production.
α	The coefficient used in the pre-emphasis filter.
β	The pitch modification factor.
c_q	The q 'th Fourier coefficient.
C^t	Target cost.
C^c	Concatenation or continuity cost.
C_j^t	The j 'th target sub-cost.
C_j^c	The j 'th concatenation sub-cost.
$d(n)$	The array that stores the cost of the path for reaching the current vertex v_n from $v_{p(n)}$.
$D(V, U)$	Acoustic distance between two units, V and U .
$D(\eta_k)$	The mapping of synthesis time-instants η_k to analysis time-instants η_m .
E_{gross}	Percentage of time containing gross pitch measurement errors.
$E_{v \rightarrow u}, E_{u \rightarrow v}$	Percentage of time misclassified as voiced or unvoiced.
f_{nm}	The frequency implied by the difference between two time-instants.
f_s	The sampling frequency of a digitised signal.
f_{high}, f_{low}	High and low frequency boundaries of the range to search for pitch hypotheses.

F_0	The pitch frequency of a periodic signal in Hertz.
$h_t(t)$	A continuous-time DSP window.
$h(n)$	A discrete-time DSP window.
$h_m(n)$	The analysis DSP window that defines the analysis frame.
$H_m(z)$	The Z-transform of the linear prediction filter computed from $x_m(n)$.
h_n	The amplitude in the time-domain pitch features at t_n .
k	Subscript for synthesis time-instants and short-time frames.
κ_i	The i 'th reflection coefficient associated with a linear prediction filter $H(z)$.
$\boldsymbol{\kappa}$	A vector of reflection coefficients.
$\tilde{\boldsymbol{\kappa}}_k$	A vector of reflection coefficients after smoothing at synthesis time-instant η_k .
L_P	The number of samples in the DFT.
L_{win}	The number of samples in the DSP window.
λ_k	The k 'th LAR associated with a linear prediction filter $H(z)$.
m	Subscript for analysis time-instants and analysis frames.
$m(k)$	The mapping that associates analysis time-instants η_m with synthesis time-instants η_k .
M_n	The set of indices m to vertices that are connected to v_n . In other words, w_{nm} , $m \in M_n$ are non-zero.
$N_0(n)$	The pitch period in samples at discrete time n .
N_0	The pitch period in samples.
N^t, N^c	The number of target and concatenation costs, respectively.
N_H	The harmonic compression factor in HPS.
N_j	The number of occurrences of $s_i \in \Sigma_s$ in S^o .
N_V, N_U	The number of feature vectors that constitute a unit.
N_{Σ_s}	The number of symbols in Σ_s .
N_v	The number of vertices v_n in the Directed Acyclic Graph (DAG).
η_m	A pitch-synchronous analysis discrete time-instant.
η_k	A pitch-synchronous synthesis discrete time-instant.
η_L, η_J, η_R	The synthesis times of the left-most, designated target and right-most

	times in the smoothing region, respectively.
$p(n)$	The array such that $v_{p(n)}$ is the vertex preceding v_n in the best path through a DAG.
$P(l)$	The discrete power spectrum, where l is an integer.
$P(f)$	The continuous power spectrum.
P_{hps}	The Harmonic Product Spectrum.
P_{exc}	A high-pass filtered <i>log</i> power spectrum, containing mostly excitation information.
$\psi(n)$	The AMDF sequence.
$r(n)$	The autocorrelation sequence.
$s(n)$	The LP PSOLA synthesised signal.
S	The set of small subsets S_i^s of the set of symbols, Σ_s .
S_i^s	The i 'th small subset in S .
S^o	The smallest subset of S such that all the symbols in Σ_s are covered by it.
Σ_s	The set of all unique symbols.
t_n	The time in the signal corresponding to v_n .
T_0	The pitch period in seconds.
U_n^i	The n 'th candidate unit for the i 'th target unit.
μ	The analysis frame length is μN_0 .
v_n	The n 'th vertex in a DAG.
\mathcal{V}_{ac}	A threshold on the tallest peak in the normalised autocorrelation sequence.
\mathcal{V}_p	A threshold for the power for a frame to be considered voiced.
\mathcal{V}_s	A threshold for the pitch continuity used in the voicing decision.
V^i	The i 'th target unit in the sequence of target units.
w_{nm}	The weight of the directed edge from v_n to v_m .
w_k	The smoothing target weight at synthesis time η_k , used for weighting LARs when concatenating units.
W_{dur}	The weight of duration variation in the acoustic distance calculation.
W_j^c	The weight of the j 'th concatenation sub-cost.

W_j^t	The weight of the j 'th target sub-cost.
ω	A continuous frequency (in radians) used with the DTFT.
$x(n)$	A sample sequence or discrete-time signal. In Chapter 3 it refers to the signal begin analysed for PSOLA processing.
$x_i(n)$	A short-time speech signal, centred at η_i in $x(n)$.
$X(n)$	The DFT of $x(n)$.
$X(\omega)$	The DTFT of $x(n)$.
$y(n)$	A PSOLA synthesised signal.
z	Frequency in the Z -transform domain.



Chapter 1

Introduction

1.1 Artificial Speech

Interest in artificially reproducing human speech dates back to as long ago as 1779, when C.G. Kratzenstein built a system of acoustic resonators to explain the production of vowels. Von Kempelen later demonstrated a more successful machine. His used bellows which supplied air to a reed which excited a resonator in the form of a deformable leather tube. The operator manipulated the tube to produce the different resonances needed to produce vowels. Fricative consonants were produced by different whistles. A famous mechanical system was Dudley's Voder. More sophisticated methods had to wait for electronics; first analogue, later digital. Dunn built the first all-electrical machine that produced intelligible speech. A skilled human could play it like a musical instrument. [1]

Digital signal processing made better modelling of the signal possible and more powerful generic computers enabled better "reading" of text. One ground-breaking system still serves as a lesson in the art of speech synthesis: Klatt's MITalk [2]. It is a fully general text-to-speech system, and uses extensive rule-based modelling of language and speech sounds to attain its goal.

Speech synthesis, and later general text-to-speech conversion, remained mostly a research toy, a means to test and refine models of speech production, pathology and perception. Recently, along with improvements in speech recognition, the capabilities emerged to build commercially viable speech interfaces. Many potential applications exist where the dialogue with a human agent is sufficiently repetitive for automatic systems to provide

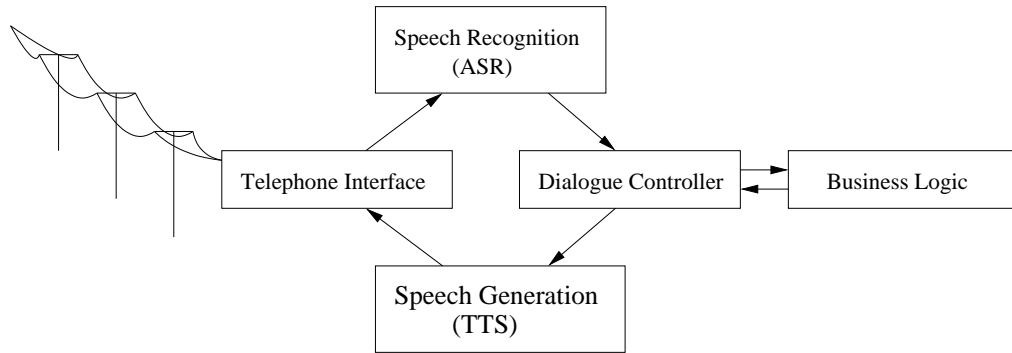


Figure 1.1: General architecture of telephone-based spoken dialogue systems. The input speech is decoded by the Automatic Speech Recognition (ASR) component, the dialogue controller determines the flow of the dialogue based on the input and the business logic, and the TTS component turns the response information into an audible utterance.

a cost-effective alternative to employing people. Marketing of such systems claim that they also improve customer experience.

Such commercial applications drive the need for more natural and pleasant synthetic speech, as this is the component cited as having the greatest influence on the user's perception of quality of a complete speech interface system. Synthesising natural-sounding speech is still a major research issue involving wide-ranging fields of study.

1.2 Motivation for this Study

Spoken Dialogue Systems (SDS) are rapidly becoming mainstream as the quality of automatic speech recognition, speech synthesis and natural language understanding improves. The ultimate goal of SDS is to allow humans to interact with computers using normal dialogue. Current commercial applications focus on simpler and more repetitive tasks.

Figure 1.1 shows the basic structure of such systems. The user enters requests via the speech recognition components. Requests are interpreted by the natural language understanding component and then a dialogue manager decides what action to perform next. It contains a dialogue model to keep track of the state of the discourse, consults external resources, such as databases, and then uses that information to generate a response. The speech generation component turns the response into speech which conveys it back to the user.

The dialogue model considered here is turn-based and can be represented as a finite-state network, with each state representing a turn in the dialogue. The system starts by eliciting some user input and then changes state based on its interpretation thereof. Upon entering a new state, an appropriate prompt is played back to the user. This is also the basis of the industry-standard VoiceXML specification for specifying dialogues for speech interaction with machines [3].

The user might for example request more information at some point in the dialogue. Based on the input, the dialogue model progresses to a state where it gives the user the desired information, and presents her with choices again. This type of system takes specific actions at each state. The contents of the reply may be generated automatically, but most commercial systems today script the reply explicitly using so-called “slot-and-filler” prompt generation. In this paradigm, a sentence that contains variable information contains a slot where information phrases such dates, amounts or strings of digits are filled in at run-time. The point to note is that the variability of the system’s responses is limited.

Errors in the dialogue may arise in several ways. To understand how, consider the following levels on which the interaction takes place [4]:

Speech: The user and the SDS both use audible speech to communicate.

Words: The speech carried between the participants contains words. It is important that the wording of both parties convey the information clearly. Humans use words naturally, but may not always phrase their intentions clearly, and the purpose of the dialogue designer is to minimise the variability in the user’s responses by minimising the number of potential ambiguities. This certainly affects the speech recognition and natural language understanding components of the system. A primary concern in this regard is that the speech generation component generates understandable speech.

Intention: The speech uttered by each participant is intended to convey information—conversion into words loses some information and causes some “noise”. Also, humans put much more than mere words into their speech. The speech synthesis component has to render speech clearly and with the proper characteristics, so that it is not

misunderstood by the user. For example, the intonation of the utterance plays a significant role in distinguishing between questions and statements in spoken language. Prosody indicates as much of the syntactic phrasing and dialogue turns as the verbal content of the reply. Such discourse effects in speech play a significant role in determining turns. For example, unclear renditions of questions could be confusing and disrupt the efficient flow of the dialogue as intended by the dialogue designer.

Even if one considers the SDS to be stupid or hard of hearing, the judgement of the quality of an SDS depends in no small part on appropriateness and clarity of the system's replies. It is therefore important to strive toward more natural speech in terms of proper intonation, clear pronunciation of words and high voice quality. This is especially difficult in languages in which speech synthesis has not been extensively researched.

1.3 Background



This section first gives some background on speech synthesis in general. Section 1.3.1 provides a brief and qualitative breakdown of verbal communication which sets the stage for the rest of this work. Section 1.3.2 follows with an overview of speech synthesis.

1.3.1 Verbal Communication

Verbal communication as studied by linguists can be divided into various levels of abstraction. We start with the most concrete field of study, the speech signal itself, and progress to the most abstract constructs in language.¹

Speech Signal: The study of characteristics and digital processing of the sound pressure waves of speech.

Phonetics and Phonology: The study of the natural production and perception of speech sounds.

¹Adapted from Jurafski et al. [5]. We add the lowest element in the hierarchy: the speech signal.

Morphology: The study of the formation of words from the root morphemes through changes the morphemes themselves and the addition of prefixes and suffixes.

Syntax: The study of the structural relationships between words.

Semantics: The study of meaning of words and utterances.

Pragmatics: The study of how people use language to accomplish goals.

Discourse: The study of the composition of units larger than a single utterance and their relationships.

The **speech signal** is generally considered to consist of several components. The components can be traced to various physiological features of the vocal tract. An engineering description of the vocal equipment is presented by Deller et al. [1, Chapter 3]. A brief summary of required supporting concepts follows. Their relationship to various properties of the signal in the time and frequency domains is detailed in Chapter 3.

Phonemically², speech sounds are classified into two main classes. *Consonants* are produced by forcing air through partial or complete constrictions in the vocal tract, possibly accompanied by vocal chord activity. Partial constriction while air is being forced out of the lungs, produces sustained sounds. These are named according to where the constriction occurs, such as palatal: [s,z] and [ʃ,ʒ], velar: [x], dental: [θ,ð] or labiodental: [f,v].³ Total constriction is usually in the form of a short build-up of pressure followed by a burst, such as [t], [k] or [p]. Sustained constriction with the velum open to the nasal cavity results in the nasal sounds, such as the bilabial: [m], palatal: [n] or uvular [ŋ]. *Vowels* are produced when the vocal tract is open and the vocal chords active.

Phonemes can be rendered in a massive range of ways through changing pitch, speaking rate, loudness (intensity) and the co-articulation effects from surrounding phonemes. The higher-level content of speech dictates the precise use of this variation.

²The terminology used here follows Deller et al. [1]. A phoneme is the smallest theoretical unit of speech whose change results in a change in meaning, where its audible incarnation is the phone. Phones are much more varied than phonemes.

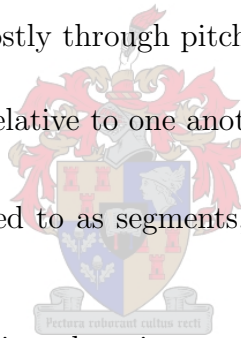
³International Phonetics Association (IPA) symbols are written between []. Where two IPA symbols are used in this section, separated by a comma, the second one is voiced.

At the **morphological** and **lexical** levels, sequences of phonemes are grouped into words. Their rendition by a speech synthesiser requires awareness of how the vocal system transitions between various configurations to render the phoneme sequence. Depending on the language, some syllables may be more prominent than others and are the basic units that carry specific patterns of loudness and pitch modulation.

Relationships between words contribute as much to the meaning of a utterance as the meanings of the words themselves. The **syntax** of a language specifies the allowed relationships between words, grouping them into phrases and sentences. Correct indication of syntax through subtle effects on the way the phonemes are produced and placement of breathing pauses allow listeners to perceive the correct syntax. The grouping of inflections used to indicate the higher-level features of the phonemes is known as prosody. It has two main components:

1. **intonation**, expressed mostly through pitch and loudness, and
2. the **duration** of phones relative to one another.

Phonemes are commonly referred to as segments, and prosody as a collection of supra-segmental effects.



Eventually there comes a point where it must be asked what the intent of producing the utterance was. Human speech is all about this intent. The linguistic schools of **semantics** and **pragmatics** deal with this level.

Semantics, pragmatics, syntax, words and phonemes themselves influence how phonemes are rendered in the speech act. Perception of speech relies on the proper production of phonemes under all these influences.

Text is another form of verbal communication, and it also embodies structure. In fact, syntax is an integral part of writing expressed in the words themselves, punctuation and formatting. Semantic information is also expressed in text, but the mechanism differs from spoken language. It is usually in more subtle aspects like choice of words, formatting and paragraph breaks. Transforming text to spoken speech is the aim of TTS. It has been argued that this goal may be inappropriate as much written material was never written to be spoken [6]; the two forms are not equivalent. This transformation between forms of

verbal communication has many practical applications however, and the challenge is to infer enough of the missing information from text in order to usefully synthesise speech.

The interplay between levels of the hierarchy of information is tremendously complex and theoretically it has to be understood to render text into speech. The challenge in speech synthesis has always been to find a workable compromise. Early TTS systems modelled everything, in keeping with their purpose as research tools. Detailed modelling is also a good means to compact representations which enabled research with the limited computing power available. The inaccurate modelling of the speech production apparatus is to blame for their robot-like quality. As voice quality improved, poor intonation modelling and text interpretation was blamed for the synthesiser's bland speaking style. The focus of current research in TTS is on these aspects in order to produce more natural-sounding speech output.

In this work, the text is written to be spoken, and the dialogue has been mapped out beforehand. This gives us the opportunity to *encode* a human's interpretations up to the semantic level rather than *model* all the levels of meaning in the text and speech automatically. The implication is that we record utterances in the context of the dialogue, and index them correctly. Indeed, as Chapter 2 will expound, the correspondence between the speaker's interpretation and rendition of the prompts, and the sufficient indexing of the recordings, is the main challenge and art.

1.3.2 Speech Synthesis

TTS is the art of creating audible speech from text. In almost all approaches to the problem, the input text is processed into a symbolic form. This form describes the linguistic aspects of the utterance in the hierarchy of Section 1.3.1 in enough detail that a model of speech production can produce speech.

Speech synthesisers are in a sense an expression of our understanding of speech. In the early days especially, every aspect of producing speech from text was built using detailed models. Naturally the reach of the models was finite, and the highest level on the hierarchy (Section 1.3.1) reached, was the syntactic level. The most basic influence of syntax on the speech signal was modelled in the form of intonation and duration models.

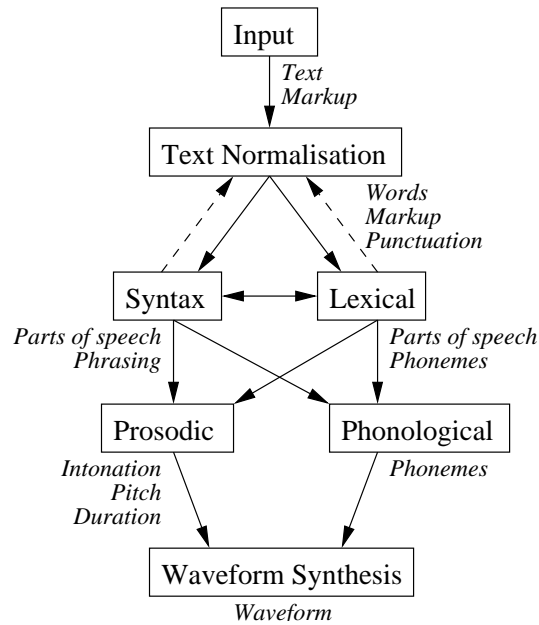


Figure 1.2: Data-flow in speech synthesis, showing the major processes or components, and their products.

Figure 1.2 shows the typical steps performed when synthesising speech: from input text, to the string of units that describes the sound, to finally producing the waveform. Variations abound, but the same basic information is always needed to build an intelligible utterance. The following sections explain how each step adds the information needed to eventually arrive at audible speech.

1.3.2.1 Text Normalisation

Text normalisation is the initial step in the synthesis process, forming the synthesiser’s text interpretation capability. It is important that the text to be analysed linguistically in the next step, is of a standard form and does not contain ambiguities about the meaning of words and symbols. The classic example is that of saying a number as a date or a string of isolated digits. Another is abbreviations, for example “dr.” can be read as “doctor” or “drive”, depending on the context. Where syntax and parts of speech help to distinguish between possible renditions of an abbreviation or other token, the text normalisation component may be integrated with downstream components.

Approaches are typically rule-based, whether they are hand-written or deduced from large amounts of text by automatic means. The intended purpose of the synthesiser determines the scope of what will be needed here. A system that has to read news must

be able to determine what to say from anything one might find in a typical news item. When a synthesiser will only need to be capable of “reading” specially written utterances, one could build a much simpler text normalisation component.

1.3.2.2 Linguistic Analysis

The next step is to derive all the information that is needed to determine how an utterance should be pronounced.

Closest to the signal level, speech is described from a phonological point of view, starting with the first writing systems that employed symbols to represent syllables. This implies that knowledge about the phonemes enables one to determine the pronunciation. Klatt successfully used this approach in MITalk [2], and it is still the basis of modern TTS. MITalk performs extensive morphological analysis, with a lexicon of morphs, to produce a phonetic transcription that describes the utterance to be synthesised. For novel words, a set of hand-written Letter-to-Sound (LTS) rules were employed. Modern synthesisers use a variety of techniques, but most involve a large lexicon for word pronunciation, and LTS rules for unknown words. There is a tendency toward automatically deriving LTS rules from large lexicons [7].

Once the text is in a standard form and word pronunciations are available, higher-level information about the utterance must be derived to produce natural prosody. The parts of speech of the words are first derived, usually using a mix of tags from a lexicon, automatic methods based on Weighted Finite-State Transducers (WFST) or Classification and Regression Trees (CART), and morphological parsing. The parts of speech may be used to group the words into prosodic phrases, assigning phrase and utterance breaks. The features of each word thus derived can be used to assign a high level description of prosody [8], which is used to compute pitch and duration.

Semantic effects on the spoken utterance need to come from the dialogue level or higher, and cannot be derived from the text. For example, the final phrase of an utterance carries special significance in signalling to the other party in the conversation that a response is expected, that more information will follow or to emphasise new information. Attempts have been made to introduce tag-sets into speech synthesis input that can

provide this information and more [9, 10, 11]. It is used to render prosodic aspects correctly.

1.3.2.3 Prosody: Intonation and Duration

There exists a massive body of research into inferring prosody from syntactic [12, 13] and semantic information, and desired speaking style [10, 14]. Much of this research is language specific, and only those languages that have been studied extensively, and for which mature text analysis systems exist, have had success with this level of modelling. It remains the difficult problem in speech synthesis.

1.3.2.4 Waveform Synthesis

The waveform synthesis component is charged with rendering a speech signal with the characteristics specified by the upstream modules in the text-to-speech synthesiser. Put simply, it must produce a correct sequence of phonemes with correct prosody.

Waveform synthesis has been done in a variety of ways. The earliest came to be to test and demonstrate understanding of the speech production mechanism and of speech perception. The analogue approaches mentioned above amounted to physical models of speech production. Indeed, this modelling interest continues in speech research. Articulatory synthesis is still studied, and some consider it the most viable long-term option for truly flexible speech synthesis [15].

Voice-coding (as practised in the telecommunication industry) provides a number of encoding schemes for speech. When coupled with a level of phonological encoding, such as a diphone or unit-selection synthesiser, some of these proved very useful for producing high quality speech [16, Chapters 6,7 and 16][17, 18].

Linear prediction coding is the most prominent of these. It enables decomposition of various components of the speech signal into compact representations. It does this in a way that holds some intuitive agreement with our understanding of speech production. This enables manipulation of the separate components in intuitive ways. Sinusoidal models arose in recent years as useful encoding parameterisations. Chapter 3 deals with this topic.

1.4 Objectives

This thesis has as its main objective the implementation of a speech generation component for an SDS. Toward that goal we

1. survey current methods in speech synthesis, and
2. distill from that survey all that is necessary to allow a relatively simple implementation.
3. We also demonstrate the application of these techniques to obtain:
 - quick to implement and workable synthesis of previously unsynthesised languages, and
 - *very high quality*

in the setting of an SDS employing a “slot-and-filler” Natural Language Generation (NLG) component.



1.5 Contributions

This thesis makes these contributions:

1. An extensive literature study is presented on the unit-selection methodology of speech synthesis (Chapter 2) and useful signal processing techniques (Chapters 3 and 4) in this context.
2. A limited-domain synthesiser was implemented and the principle shown to work well for languages where several linguistic components do not yet exist. (Section 2.6)
3. An automatic method for deciding the contents of a compact and complete speech corpus is presented. (Section 2.3)
4. An automatic method for performing phonetic alignment on small corpora is developed. (Section 2.6.3.2)

5. The first incarnation of the system was evaluated, and several shortcomings identified. (Section 2.6.5).
6. A synthesis engine based on Linear Prediction Pitch-Synchronous Overlap-Add was implemented to help alleviate some of the shortcomings of the limited-domain methodology. (Chapter 3)
7. The aforementioned synthesis engine requires pitch-tracking; a robust pitch tracking algorithm tailored to the signal synthesis component was developed to that end. (Chapter 4)

1.6 Overview and Organisation of this Work

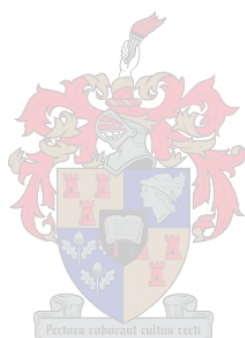
The trade-off between encoding and modelling, or moving knowledge from the synthesiser into the data [6] is the underlying theme for this work.

Chapter 2 discusses unit-selection synthesis and sketches a picture of the movement of speech synthesis from modelling to ever more intricate encoding and indexing of the encoded speech. Constraining the output of the synthesiser in this way to incorporate all the prior knowledge about the task at hand allows one to avoid much of the intricate modelling, and to encode the knowledge in carefully constructed data instead. This study works toward that end in the context of a specific spoken dialogue system application. It is found that in a limited domain virtually everything can be delegated to the process of making a small set of recordings, and indexing them appropriately.

Finally, a partial return to knowledge expressed in rules and models is made to improve the results and to make the synthesiser more robust. Chapter 3 discusses methods to modify the encoded speech to make it conform to a model in specific situations where the encoding can be shown to fail. The modification component requires very good pitch-tracking, the topic of Chapter 4.

To summarise, we aim to use the opportunity given by cheaper computing power and the constrained environment of an SDS to build appropriate speech synthesis up to the semantic level. First, as much information as possible is encoded by making appropriate

recordings. Finally a brief return to modelling is made to modify the speech signal in cases where the encoding philosophy is too restrictive.



Chapter 2

Waveform Synthesis by Unit Selection

2.1 Introduction

The first attempts to synthesise speech on digital computers involved detailed models of speech production, as well as complex sets of rules to determine the parameters of these models. The classic examples include the Voder (where the parameters come from the human “player”) and formant-based systems like MITalk [2]. Later, the idea of setting up templates for parameter progressions came to lessen the role of the rules in specifying the parameters. Realisations of speech waveforms based on the templates are constructed by concatenating the short waveforms synthesised from each template. The classic example of this level of concatenative speech synthesis is the diphone synthesiser, where the inventory of templates attempts to cover all the transitions between different phones or allophones. Classic diphone synthesisers make clear decisions about the inventory of speech sounds that can be produced, and then rely on signal processing techniques to extend the set to cover pitch and duration variations, for example. These templates of parameter progressions are captured by analysing speech recorded for that purpose, usually a series of nonsense words.

Modern speech synthesisers make use of cheaper, faster computers by extending the inventories. While some follow phonological arguments to define a much larger set of predefined unit templates, others go on to the point where they use large corpora of fluent

speech directly. The synthesiser selects short pieces of speech waveform from a corpus and concatenates them to produce the required utterance. As such, the inventory is defined implicitly by the data that the synthesiser has access to. Such a speech synthesiser is called a unit-selection, or concatenative synthesiser. Two common observations support this technique [19, 20]:

1. speech spoken in one context may be used in another, and
2. sections of a speech waveform may be concatenated without audible distortion.

Figure 2.1 depicts the unit-selection synthesis process in general. The synthesiser starts by extracting a number of candidates for each “unit” from its database of fluent speech, and then selects one from each set. The selected units are concatenated to produce the final utterance.

Unit-selection synthesis forms the basis for the current drive of making speech synthesisers trainable. The fact that a large amount of single speaker data is available, allows the exploitation of the same data to train models for predicting the pitch and duration from the linguistic specification. The predominant view here is that our knowledge of speech production is too limited and that the knowledge may be shifted from the synthesis engine to the data. Unit-selection synthesisers give up some of the flexibility and control that more detailed models and even diphone synthesis have, making it more difficult to build reliable unit-selection synthesisers.

Since it uses the recordings directly or with very minor modification and attempts to mimic the prosody seen in the database, a *unit-selection* synthesiser can usually imitate the person whose voice was recorded for its construction quite convincingly. So much so in fact, that it becomes viable to build a synthesiser that produces speech in the style that would suit the application.

This discussion immediately raises several issues.

- The waveforms must be indexed so that the synthesiser can find appropriate units. How can this be done efficiently, and what is meant by a “unit”? (Section 2.2)
- Which units are needed by the synthesiser to perform well on its intended task? (Section 2.3)

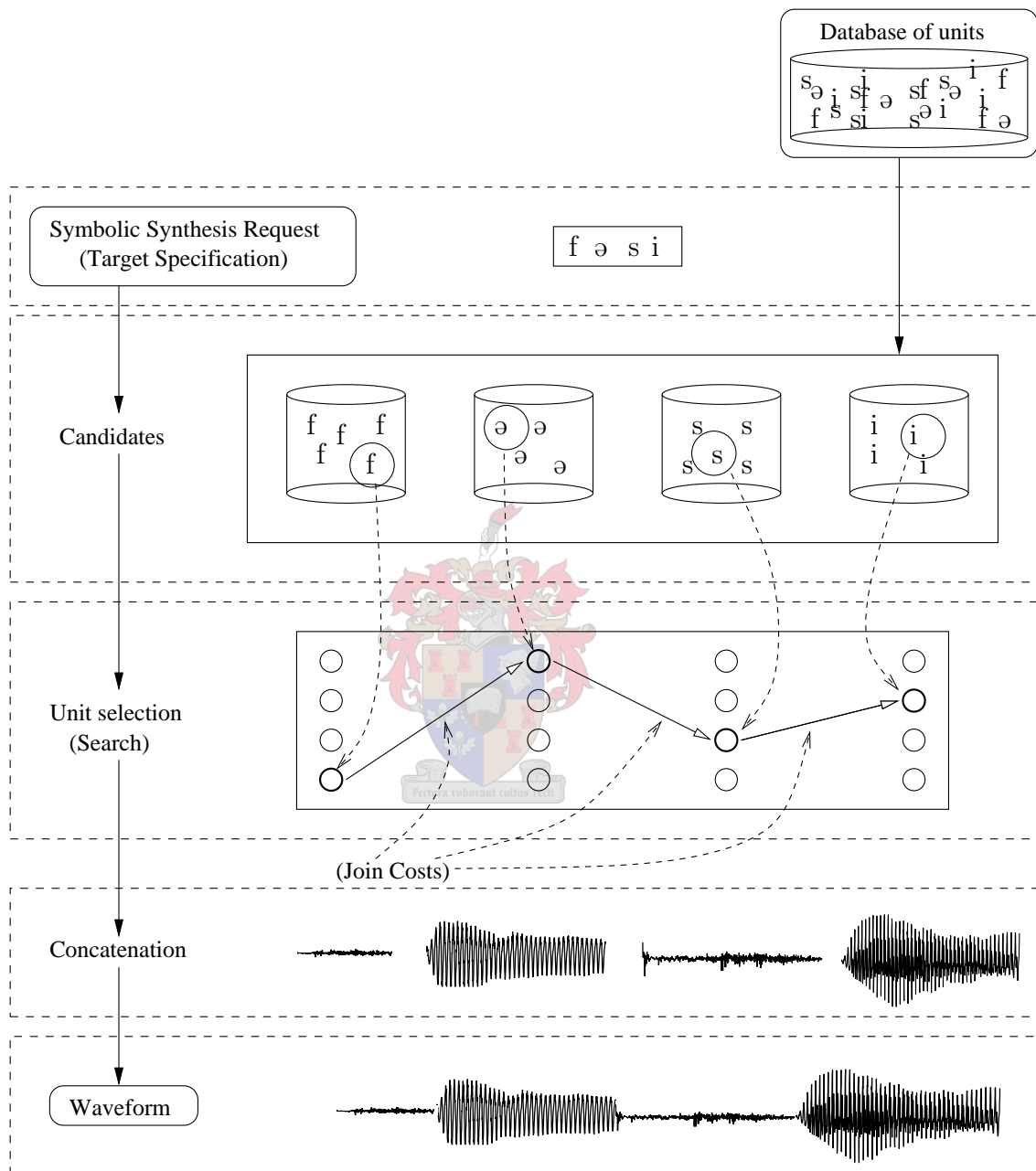


Figure 2.1: The unit-selection synthesiser receives a synthesis request in the form of a symbolic target specification, and from a set of candidates for each “unit” in the target, finds an optimal example. Next the optimal examples are concatenated, and sometimes modified.

- How do we select units from the corpus to best synthesise the required utterance? (Section 2.4)
- How do we splice the selected units in order to minimise audible artifacts? (Section 2.5)

This chapter explores these questions from literature on the topic, and then distills an answer to the problem stated in Chapter 1.

Chapter 3 discusses signal processing of the waveforms for smooth concatenation and possibly improving the fulfilment of the target specification by modifying pitch and duration. Chapter 1 touched on issues of determining the specification of the target utterance.

2.2 Indexing the Corpus

A unit-selection synthesiser takes the place of the waveform synthesiser in the TTS functional block diagram in Figure 1.2. Its input is linguistically augmented phonemic targets. Since unit-selection synthesisers make use of fluent speech databases, their indexing of the data may allow advantageous use of higher-level linguistic features, including stress, accent, syllable boundaries and word boundaries. There exists a close synergy between the capability of the text analyser, the annotation used for the database and the actual speech in the database.

The first widely known unit-selection synthesiser was a system called CHATR. It was developed at ATR Interpreting Telecommunication Research Laboratories in Japan [21]. The first versions of CHATR used phonemes¹ as the basic unit for concatenation, so that concatenation occurs at the phoneme boundaries.

The suitability of concatenating at phoneme boundaries is questionable. Yi conducted a detailed study of English phonology with concatenative synthesis in mind, and constructed a large number of common phoneme sequences from a large lexicon [22, 20]. The sequences were constructed manually and in such a way that the boundaries of the

¹A note on terminology: we use the term *phoneme* to be a linguistically distinct unit, i.e. substituting a different phoneme would result in a change of meaning. A *phone* is a realisation of the more abstract idea of the *phoneme*.

phoneme sequence are conducive to simple and distortion-free concatenation. Concatenation in areas where the source changes from voiced to unvoiced or vice versa, were virtually inaudible. Further good splice points are indicated by the observation that stop sounds bear very little co-articulation effects, making either side of the stop sound a good join spot. “Variable length” units are then defined to be phonemes or parts of phonemes between good join spots, and the units were recorded in various prosodic contexts.

A Japanese synthesiser has been built on similar ideas of predefining a large set of units based on commonly occurring strings of phonemes [23], and designing the speech corpus to provide good examples.

In almost all of these approaches to segmenting a corpus of speech into units, the eventual unit selection is performed through optimising, among other things, the total sum of concatenation costs (see Section 2.4). Most modern systems borrow this idea from CHATR. In this scheme, the concatenation cost of concatenating two units that happen to be consecutive in the original recordings, can be hard-wired to zero. This encourages the system to select longer sets of consecutive units, resulting in *variable length* or *non-uniform* units.

This flexible, data-dependent means of selecting longer synthesis units can be used to exploit the increase in concatenation points that smaller units give, without necessarily causing more concatenation points. Both the creators of CHATR and researchers at AT&T experimented with half-phones as the fundamental unit of concatenation [24]. Using half-phones allows the concatenation of sections from the database in the stable regions of vowels. The system is then still free to concatenate speech sections at the boundaries of stop sounds too. An advantage this scheme has, is that it can mimic the concatenation of diphones where appropriate. Note that this idea could break down very badly if the join costs are ignorant of the underlying phonetics (See Section 2.4.3).

Another fine-grained approach that allows more flexibility in where to join the units, is an exploitation of properties of the speech recognition technology used to automatically segment the corpus. Hidden Markov Model (HMM) techniques have been used for automatic phonemic labelling of speech synthesis corpora [21, 25, 26]. Most use simple three-state left-to-right acoustic models, one per phonemic label. Instead of drawing phoneme boundaries on the transitions between different phoneme models, one can label

each *state transition*, resulting in three labels per phoneme. The synthesiser allows splices to be made at the boundaries of each of these so-called *senones* [27, 26].

Much of the variation in human speech comes from higher-level considerations: the effect of syntax, semantics, discourse and their realisation in prosody. It follows that finding the desired units in the database based only on their phonetic context will very likely produce inappropriate candidates. Including higher-level features in the candidate search results in much more natural-sounding speech [28, 29, 30].

The choice of higher-level features depends on the quality of the text analysis component of the synthesiser and how much variation is available in the data. Also critical is the quality of the labelling [31, 32, 33, 27]. Even expert human labellers often disagree on how to assign higher-level labels like ToBI tags [12] to a segment of speech. Solutions are a combination of introducing automatic techniques to hopefully improve consistency, and simplifying the labelling scheme [33, 28]. The latter aids both human and automatic labellers.

The same ambiguity is found in the choice of the set of phone or phoneme labels. Only sparse mention is made of this subject in the literature on unit selection. Synthesisers that define the set of units beforehand need to make explicit mention of all the distinct sounds the synthesiser can produce. Unit-selection synthesisers move the finer distinctions from the synthesiser to the data and so they can actually perform better when a phoneme label describes a somewhat wider range of phenomena. That it can even be considered to build synthesisers based on the orthographic spelling of words, without converting to a phonemic representation, is testimony to this [34].

Another topic which is rarely addressed in the literature, is how well the text analyser's interpretation of the input utterances and the annotation of the database matches the interpretation of the speaker who delivered the speech for the database. This affects the quality of the synthesiser in a fundamental way. Steps in the direction of explicitly capturing and replicating speaker behaviour have been taken [35, 36], and automatic labelling of corpora incorporating multiple possible pronunciations have been reported [37, 38]. Methods exist to train models of duration and intonation of units on single speaker corpora [33, 37], further adopting speaker specificities.

To conclude, unit-selection synthesisers require time-aligned symbolic information to

select candidate units. Phonemic labelling forms the basis of most indexing schemes. Higher-level labelling is often added to give clues about the context of a phone, which enables a synthesiser to find an appropriate set of candidates. At first sight it might appear that the more detail the annotation contains, the better. However, the consideration of wider than phonetic constraints that result from the global optimisation process, allow unit-selection synthesisers to get the finer details from the data. The annotation should better enable this, rather than being more specific. This phenomenon is another question about the roles of rules and data in speech synthesis [7].

2.3 Corpus Design

The speech quality that results from unit-selection speech synthesis is the result of a synergy between the contents of the corpus, the way it is indexed and the text-analyser. The text-analyser gives targets to the unit-selection synthesiser, which has to be able to find relevant waveforms from that, and the corpus must contain a sufficient number of relevant examples. The first unit-selection synthesisers simply used as much data as they could, but the cost of synthesis quality data is high and the results not quite predictable. The success of synthesisers that explicitly define a set of units in the style of diphone synthesisers, alludes to the benefits of carefully giving the synthesiser what it needs [22, 23, 39]. In this section we briefly report on techniques to design compact corpora that provide coverage of the intended language or domain.

The first question to answer would be what it means to cover the language or domain. Donovan [25] simply selected many random sentences and recorded them. Van Santen et al. [40, 41] and Möbius [42] argue that most phonetic events in speech are rare, and that because of the overwhelming number of rare events, the synthesiser is likely to come across many of them. This is the so-called Large Number of Rare Events (LNRE) phenomenon. A random corpus of the order of a thousand spoken sentences will certainly not contain that many different events. The domain or language is *covered by the corpus* if it contains examples of everything the synthesiser must be able to produce. Note that since the text-analyser gives the requests to the waveform synthesiser, it is usually sufficient if the corpus covers the output domain of the text-analyser.

Secondly, what events must the corpus cover? The combinatorial growth in the number of units as wider phonetic and higher-level contexts are taken into account, forces a compromise between more and more detailed features and corpus size. The Next-Gen system at AT&T simply uses a “large enough” corpus, following CHATR. At IBM [37] the corpus is designed to contain more than a minimum number of all the diphones produced by the text-processing front-end on a large text corpus. The IBM synthesiser’s corpus is built by using a greedy algorithm that selects the sentence with the largest number of new diphones at each iteration. Others still cover triphones [43]. Others go further and try to cover the most common morphemes, syllables and phoneme strings, in various prosodic contexts [44]. It is also possible to see which acoustic variations of phonemes are needed when synthesising a large text corpus [45]. The selection statistics are then used to design a more compact corpus, or to compact the existing one. Its application for database design is limited to allowing better datasets to be built once a very comprehensive one has already been used in unit-selection synthesis. To ensure smooth transitions between words, a strictly limited-domain synthesiser should exhaustively cover all the word pairs possible in its target domain. It is an accepted fact that unit-selection corpora that try to cover everything in the language would probably never have enough data [46, 42].

Once it has been decided what the goal of the covering is, a very large corpus of text is used to get many sentences and the text-analyser of the synthesiser is used to find all the symbols that each sentence could contribute to the corpus, if it were selected. We take the frequencies of phonemic events in the large phonetised text corpus to represent that of the language, or at least the representation of it in the domain which the synthesiser targets. The product of text selection is a corpus whose frequency of occurrence of the chosen type of events is much more dense in rare events than the original large database; its histogram of event frequencies will be much flatter.

2.3.1 Text Selection

Analogous to speech recognition systems, data-driven speech synthesisers are only as good as the underlying data. As discussed above, the synthesiser must be able to produce a wide range of rare events and as many as possible of these must be present in the database. Since the waveform synthesiser produces waveforms from the output of the text-processing

front-end, exercising the front-end using sentences from a large corpus of text is a good way to build training data. Each sentence produces certain symbols. Symbols are most often comprised of a phoneme, indications of its phonetic context and perhaps selected higher-level features such as lexical stress or accent. The goal of text selection is then to choose a subset of these sentences, such that as many of the symbols as possible are present at least a certain number of times, in the smallest subset of the original sentences.

Selecting the smallest subset can be formulated as the set covering problem. Given a global set $\Sigma_s = \{s_1, s_2, s_3, \dots, s_j, \dots, s_{M-1}\}$ of all possible symbols, and a set S of N small subsets of Σ_s , $S = \{S_0^s, S_1^s, S_2^s, \dots, S_{N-1}^s\}$ with $S_i^s \subset \Sigma_s$ $i \in [0; N - 1]$, select the smallest subset $S^o \subset S$ that contains all or a sufficient number of the symbols in Σ_s . The definition of the symbols s_j depends on the problem at hand, as described in Section 2.2.

This problem has an optimal solution, but it is NP-complete [47]. Approximate solutions employing greedy selection algorithms have been employed with very usable results. Several variations exist [40, 43, 45]. The basic form of the algorithm is depicted in Figure 2.2.

The particular forms of the *done()* and *score()* functions determine the performance of the algorithm. To cover diphones for an entire language, one would include a certain minimum number of examples of each diphone. A consideration for a database for training speech recognition models or duration prediction systems could also be to cover the data sufficiently for estimation of parameters of a model [40]. The designer of a limited-domain synthesiser would be most interested in recording as few utterances as possible, but needs at least one example of each word or word pair in the relevant contexts.

In the classic version of the algorithm, the *score()* function would simply return the number of symbols in S_i^s not yet covered sufficiently in S^o , and the algorithm terminates when a minimum number of each symbol has been selected. A refinement suggested by van Santen [40] is to score each symbol s_j in a set S_i^s by the inverse of its frequency over the entire set S . The score of the subset S_i^s is the sum of the individual symbol's scores. In that way, the algorithm prefers sentences containing the rarest symbols, and selects them first while common symbols are picked up in passing. Once one or more examples of a symbol have been selected, up to a threshold, it no longer contributes to the score of a subset S_i^s , and the algorithm terminates when no utterances score above 0. In this form,


```

Initialise: Start with two arrays,  $S \leftarrow \{S_0^s, S_1^s, S_2^s, \dots, S_{N-1}^s\}$  and
 $S^o \leftarrow \emptyset$ 

Iterate:

while not done( $S^o$ ),

    let  $i^* = \arg \max_i \text{score}(S_i^s)$ 

    add  $S_{i^*}^s$  to  $S^o$ 

    remove  $S_{i^*}^s$  from  $S$ 

```

Figure 2.2: The greedy algorithm for set-covering.

the greedy algorithm has to compute $\text{score}()$ for each sentence in the corpus, for as many times as the algorithm is iterated. For large sets, it becomes expensive very quickly.

The last refinement we make, is to compute $\text{score}()$ only on the sentences that will contribute the rarest, as yet uncovered, symbol. Once the rarest symbol has been covered sufficiently, the algorithm moves on to cover the second rarest, and so on. In this way, the number of computations of $\text{score}()$ per iteration is reduced from the size of the set S , to the frequency of the next rarest symbol at each iteration. An additional initialisation step is required to collect, for each symbol, a list of the sentences that contain it. The cost of this collection step is small compared to running through the full set at each iteration. The more common units are picked up in passing, while the algorithm ensures the presence of examples of all the symbols.

Experiments on the diphone and triphone sets described below, as well as limited-domain tasks described in Section 2.6 show that the refinement to $\text{score}()$ causes about 5% of the selected sentences to differ. There was no significant difference in the symbol coverage statistics and the resulting set size. The difference in computational cost is two to three orders of magnitude on a practically sized set.

2.3.1.1 Experiments

An experiment was conducted using roughly 280 000 English sentences harvested from electronic books from Project Gutenberg. The sentences were selected using a simple regular expression that disallowed a large number of sentences since they contained illegal characters, such as quotation marks. Each sentence was transcribed into a phonemic string using the English text-analysis facilities of Festival [48]. The sentences contain many proper nouns and some phrases in French, German and other languages. These and other out-of-lexicon words were transcribed by Festival using letter-to-sound rules that were trained on the included CMU lexicon of about 80 000 English words. As a result, there are mistakes in the pronunciations. When using the selected data to build general synthesisers, the issue of pronunciations must be addressed for all these words.

The set of phonemic sentences were converted to a set of diphone sentences, containing 1 540 diphones. A theoretical total of 2 116 diphones is possible, but not all possible combinations of two phonemes occur in a language. Only 270 sentences were needed to obtain at least one occurrence of each diphone. If each diphone has to be in the corpus at least five times, the selected set is enlarged to 1 046 sentences. Requiring 10 examples enlarged the selected set to 1 977, which is still a viable number to record and process.

Figure 2.3 shows the fraction N_j/N_S , where N_j is the number of occurrences of the particular diphone s_j and N_S is the sum of all the N_j in the set of sentences under consideration:

$$N_S = \sum_{j=0}^{M-1} N_j. \quad (2.1)$$

The diphones are indexed by j in order of increasing cardinality in the original large set of sentences, S . Since the graph remains monotonically increasing, the order of units are the same in the selected sets. Plots are shown for minimum values of $N_j \in \{1, 5, 10\}$ examples of each diphone in the database. The more common units are still much more common than rare ones, but the percentage of rare units has increased. The resulting database is much more dense a representation of the diphones that occur in the initial large set.

Figure 2.4 shows the fraction N_j/N_S for the total number of triphones for the complete set of sentences, as well as the selected sets for a minimum of 1, 5 and 10 examples of each triphone in the database. The total number of triphones is 30 571, many more than the

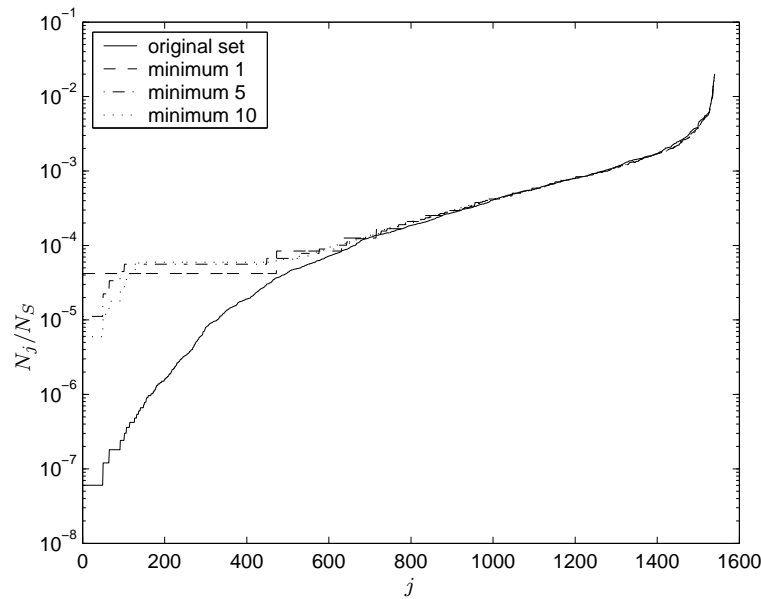


Figure 2.3: The distribution of the diphones in the Project Gutenberg set of sentences and the selected sets. The plots show the fraction N_j/N_S , where N_j is the number of occurrences of a particular diphone s_j and N_S is the total number of diphones in the selected set. Plots are shown for minimum values of $N_j \in \{1, 5, 10\}$ examples of each diphone in the database.

diphones and far fewer than the total number of possible combinations of three phonemes: 97 336. Covering each triphone at least once required 7 166 sentences, an approximately 40 \times reduction in size of the text corpus. The reason that triphones require so many more sentences to cover, is that the sentence length stayed the same, and thus the probability of finding a sentence that yields more than one rare symbol is much lower than in the diphone case. One option is to throw away these very rare cases and rely on diphone coverage to synthesise them when they do occur [43].

Covering triphones using an algorithm that forces the rarest symbol to be present in the selected subset, is perhaps not the best way to ensure good coverage. It is however desired when selecting text for limited-domain synthesisers. Another option is to select sentences according to a score derived using phonological knowledge, and without the last optimisation mentioned in the previous section. The selection algorithm can then focus on what is important to the synthesiser, at the cost of not covering all the tokens exhaustively [40]. It could also be argued that this is a better way to cover symbols in light of the errors that most certainly occurred during the text to phoneme conversion.

This section discussed selecting text for speech synthesisers. Further applications

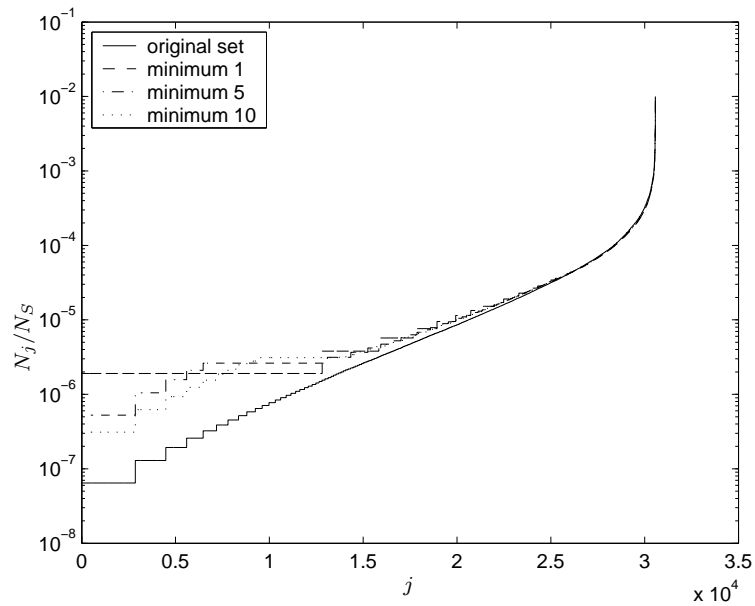


Figure 2.4: The distribution of the triphones in the Project Gutenberg set of sentences and the selected sets with minimum counts of 1, 5 and 10. The triphones s_j are sorted in order of increasing cardinality.

include the design of compact training and testing corpora for speech recognition. Since this set-covering method is very efficient, it can be used to design an optimal set of sentences given constraints on the final database size and the choice of symbols using a rapid prototyping approach to see how various factors influence the size of the selected set.

2.4 Unit Selection

2.4.1 Search

A crucial insight from the development of CHATR was that the search for the optimal unit in each bin could be cast as a search through a finite-state network, analogous to that in HMM-based speech decoding [19].

Synthesis starts by building a set of candidates for each unit in the target (see Section 2.4.2). Each candidate in each set is fully connected to all the candidates in the preceding set to form a weighted finite-state network, from which an optimal path can be found using a Viterbi search. Section 2.4.3 discusses the weights in more detail. Figure 2.1

shows a hypothetical network constructed from a set of candidates, and one possible search path.

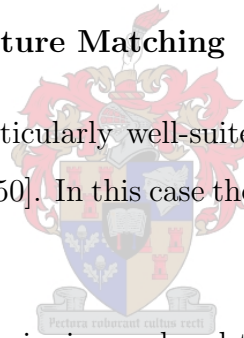
The size of the database could quite easily grow to the order of hundreds of thousands of units, and as Section 2.4.3 will show, this means that the search could become very computationally expensive.

2.4.2 Candidates

When the synthesiser receives its synthesis request, it starts by finding all the units in its database that could fit into the utterance it has to produce. CHATR worked by simply taking all realisations of a phoneme from the required context in the database and considering them to be viable candidates.

2.4.2.1 Phonological Structure Matching

Another approach which is particularly well-suited to limited domains, is Phonological Structure Matching (PSM) [49, 50]. In this case the database is annotated in a hierarchical fashion.



Phrase: Each phrase has its beginning and end times marked.

Word: Phrases contain words, and these could have associated features such as phrase position (initial, internal, final), sentence position, boundary tone type (rising eg. at the end of a question, falling eg. at the end of a statement).

Syllable: The words contain syllables. The syllables can be marked according to lexical stress or tone accent.

Phonemes: The syllables in turn contain phonemes. Features of phonemes include syllabic position: onset or coda.

Figure 2.5 shows an example adapted from Schweitzer et al. [50]. They assigned primary and secondary features to each unit. Primary features include phonetic transcription, stress, word and phrase boundaries. The secondary features are not necessary for a unit in the database to be a candidate for selection, but the unit is scored later on how well it

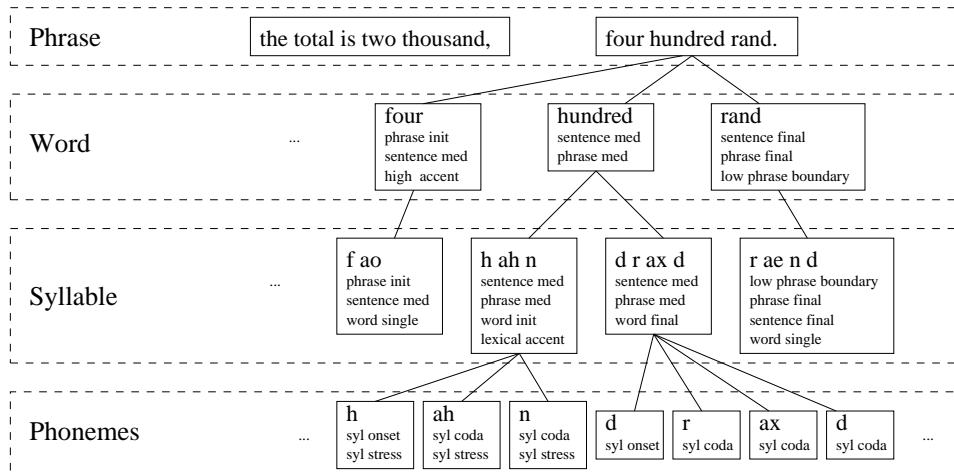


Figure 2.5: An example of the phonological structure in PSM. The search for candidate units start at the phrase level, and proceeds through the word, syllable and phoneme levels until suitable candidates are found. The example is adapted from Schweitzer et al. [50], and depicts their notion of primary and secondary features.

matches the secondary features of the target specification. Secondary features are different at each level of the phonological tree. At the word level they include boundary tone, pitch accent type and phrase position. More detailed levels add features, such as word position for syllables and syllabic position for phonemes.

The synthesis algorithm proceeds by finding the best matching candidates at each level, starting at the top. If it does not find any units, or perhaps less than a minimum number, at the phrase level, it proceeds to search for candidates at the word level. It continues downward to the phoneme level. In limited domains, or at least when the database has been augmented with domain specific phrases, this approach could yield very good results with minimal computational load in cases where only a few units are needed to synthesise the utterance.

Alternatively, one could make the units smaller, for instance phones or half-phones. All these units are binned together, one bin of candidates for each unit in the target specification. Instead performing a beam search to optimise the search, the units in the database can be indexed so that more appropriate candidates are selected. That is the topic of Section 2.4.2.3.

2.4.2.2 WFST-based Methods

Two new and related methods are those by Yi [20] and Bulyko [27]. Both use Weighted Finite-State Transducers (WFSTs) to jointly predict prosody, find good candidate units and find the optimal subset of units to concatenate. Yi goes further and devises a method for training weights for a general phonemically aware join cost. Jointly predicting prosody and performing optimal unit selection allows the prosody available in the data to influence the selection process, making the procedure more robust to missing phenomena in the data. The challenge of indexing the data appropriately remains.

2.4.2.3 Acoustic Clustering

Another idea, also found in speech recognition, is to automatically cluster units of the same type, as they may have acoustically different realisations. A clustering method would have to allow the synthesiser to reach the correct cluster, based on symbolic features of units at synthesis time. The Classification and Regression Tree (CART) method has been used in speech recognition [51], and has also been explored for use in synthesis since the early 1990's [25, 37, 52]. It continues to find application in unit-selection synthesis. At synthesis time, the features of a symbolic target unit is used to find the leaf node of the tree which indicates a smaller and more appropriate set of candidates.

There are two prominent ways to use the trees. As an example of one, Donovan built the decision trees as part of HMM training on the single speaker database. The HMMs are used for automatic phonetic alignment of the data, with units being labelled as the waveforms that align to each state of the phoneme models [25, 37]. The units that belong to each state are clustered by maximising the log-likelihood of Gaussian mixture models over a set of questions about phonetic contexts.

The second approach is that of Black and Taylor [52], as found in Festival [48]. One major difference from Donovan's incarnation is that it uses an acoustic distance measure $D(V, U)$ between two units V and U of the same type. Equation 2.2 describes the distance measure. The units V and U are represented acoustically as two sequences of M -dimensional vectors. $V(i, j)$ is the j 'th component of the i 'th vector of unit $V(i, j)$,

and likewise for $U(i, j)$.

$$D(V, U) = \frac{N_V W_{dur}}{N_U} + \frac{1}{N_V} \sum_{i=0}^{N_V-1} \sum_{j=0}^{M-1} W_j |U(\lfloor iN_U/N_V \rfloor, j) - V(i, j)| \quad (2.2)$$

with $N_V \geq N_U$.

W_{dur} is a weight that determines the importance of the difference in duration on the total distance and W_j is the weight of the difference between the j 'th components of the vectors that make up the units. The shorter unit is matched linearly in time to the longer one. Thus, $D(U, V)$ is a weighted Mahalanobis distance-based measure of acoustic similarity. The *impurity* of the cluster is defined as the mean of the distances of each unit to all the others of the cluster of units.

Building the Decision Trees

In both the acoustic distance and log-likelihood incarnations of the clustering algorithm, trees are built using a greedy algorithm. The algorithm starts with all the units in the set to be split. In a synthesiser where phonemes are used, all the units will be of the same phoneme type. Methods that use HMMs and log-likelihoods to build decision trees naturally cluster units corresponding to HMM states. At each iteration, the algorithm splits the cluster under examination according to questions about symbolic features that describe the unit. The question which results in the best gain in log-likelihood, or the two clusters with the lowest impurity, is chosen. Each resulting cluster is examined in turn, and in the same way. The algorithm terminates when

- no more clusters with more units than a threshold exist,
- no more splits would result in a bigger drop in impurity than a threshold or
- when a maximum number of clusters have been reached.

Typical examples of questions include questions about

- surrounding phonemes,
- types of the surrounding phonemes (vowel, consonant, voiced, unvoiced, dental, palatal, rounded, etc.),

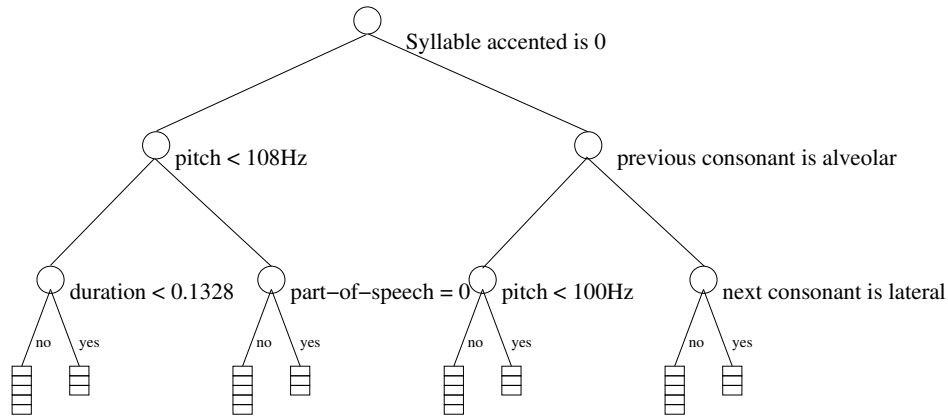


Figure 2.6: The target unit [oj] (a diphthong) is dropped down the tree associated with it and questions about its context used to lead to the correct cluster. This cluster of units is used in the Viterbi search for this unit.

- syllable position,
- stress and
- *predicted* pitch.

The intended use of the CART is to reach the correct cluster with only the symbolic synthesis request on hand. Therefore only symbolic information obtained by text processing or mark-up may be used to build the decision trees.

Figure 2.6 depicts a decision tree with questions about the features of a target unit. Note that the pitch and duration questions refer to the *predicted* pitch and duration. The pitch and duration targets are predicted using the same parts of speech, stress, accent and other linguistic information as is available to the trees. They may, however, still provide useful summary information according to which to find appropriate units. In this example the CART building algorithm seems to have had too many questions to choose from, as it is difficult to see the relevance of the question about the parts of speech being undefined.

The tree clustering method deals with data scarcity implicitly, in that clusters will only be split if there is a sufficient number of examples and there is significant gain in splitting a cluster. Another, less fortunate property of clustering data using a greedy algorithm to approximate the optimal split, is its tendency toward data fragmentation. The data may be split such that equally appropriate candidate units may end up in different clusters [37]. Merging small clusters again after splitting may yield better decision trees.

Phonemic Context in Building CARTs

Donovan [37] performed extensive experiments to determine how much effect questions about phonemic context had on the resulting synthesis quality. They compared systems built with questions derived by considering

- one phoneme on either side,
- two phonemes on either side,
- three phonemes on either side,
- one phoneme on either side and a tag which gave the position in the syllable and
- one phoneme on either side and a tag which described three possibilities for lexical stress on the central phoneme.

In each case word boundary markers were inserted as an additional “phoneme”. In speech recognition word error rate experiments, the extended contexts had a worthwhile effect. Their synthesis system also used these questions, indicating that they resulted in splits with worthwhile log-likelihood gains. During listening tests however, the improvement in modelling had not been carried over to the final product. One hypothesis offered was that the subsequent processing after candidates had been obtained from the tree was the source of much of the synthesis artifacts listeners described as unpleasant. Specifically, discontinuities at the join locations and signal processing artifacts were to blame, and the improved modelling is powerless further down the synthesis chain.

Another possibility is that the required transitions simply were not present in the database, causing join artifacts. Again, the synthesiser is only as good as the underlying data and, the range of questions from the Festival system may be too wide to really be supported by any amount of data.

2.4.3 Concatenation and Target Costs

Inspired by CHATR, modern unit-selection synthesisers weigh transitions in the state transition network according to target and join costs. Let V^i be the i 'th target unit

in the sequence of units to be synthesised. For each target unit V^i , the unit-selection algorithm assembles a set of candidate units U_n^i .

The target cost $C^t(V^i, U_n^i)$ expresses how well candidate unit U_n^i matches the target specification V^i from the synthesis request. The join or concatenation cost $C^c(U_m^{i-1}, U_n^i)$ indicates how well candidate units U_n^i and U_m^{i-1} would fit together if they were selected and concatenated. Hunt and Black [19] define the two costs as the weighted sum of various sub-costs:

$$C^t(V^i, U_n^i) = \sum_{j=0}^{N^t-1} W_j^t C_j^t(V^i, U_n^i), \quad (2.3)$$

$$C^c(U_m^{i-1}, U_n^i) = \sum_{j=0}^{N^c-1} W_j^c C_j^c(U_m^{i-1}, U_n^i). \quad (2.4)$$

The target sub-costs are derived by assigning numerical values to differences in various symbolic features of the units. The most common concatenation sub-cost is a distance between acoustic features at the end of U_m^{i-1} and at the beginning of U_n^i .

Sub-costs that have been used in the realisation of $C^t(V^i, U_n^i)$ include differences in predicted duration, pitch and energy with the values of the candidate units, phonetic context differences (in the absence of acoustic clustering) and considerations based on the amount of signal modification a unit would have to undergo [37, 19].

The join cost tries to infer how smoothly two units would join if selected. This naturally leads to distance measures inspired by human speech perception. Klabbers et al. [53] investigated the use of several distance measures based on comparison of the spectral envelope of speech frames in the context of diphone-based synthesis. Listeners were asked to rate discontinuities and the correlation of the distance measures with respect to the listener ratings were compared. According to their study Mel-scaled cepstra did not perform very well at all, while a Kullback-Leibler distance-based measure correlated quite well with the perceptual measurements.

In a subsequent study at IBM [54], a similar set of tests were performed. The supporting observation was that the perception of the join cost was highly dependent on phonetic context. They again used a decision tree to cluster the difference of vectors at both sides of a join. A ‘‘typical’’ difference could be derived for each phonemic context in this way. When the join cost is to be computed for two units at synthesis time, the decision tree

is consulted, and the distance is computed in a way that takes the typical difference for that context into account. Since speech is highly dynamic, one would expect a distance measure that “knows” what the difference should be rather than one that simply measures similarity to perform much better. The results correlate better with listeners’ perception than other measures tested by a large margin.

While the tree-based distance measure provides very good results in computing join costs, it must still be balanced with the target costs, and target sub-costs must be balanced with each other. Several attempts at training the weights W_j^t to work well with the join cost computation have been made in other contexts [19, 20, 31]. Some claim that the results are usually better than what can be obtained by manual tuning. Coorman et al. claim that manual tuning based on linguistic expertise gives better results [55]. They go one step further and base join costs on additional features such as pitch and devise non-linear functions for the cost based on the absolute value of the pitch and the pitch difference to indicate acceptable ranges for feature differences. It should be noted, however, that the unit-selection search is quite robust to surprisingly large weight variations. This type of manual tuning is labour intensive and runs the risk of the resulting weights being specific to the training set, speaker or database.

Yi [20] proposes a formulation where weights are derived off-line during the build process. The information-theoretic distances that he derives from decision tree clustered acoustic data is used to automatically train join and target weights for different clusters. The results are shown to correlate very well with that of a detailed phonological study. The distances amount to an extension of the context-aware distance measure built at IBM [54], to include target costs and provide a phonological basis for the observed improvement in listener correlation.

2.4.4 Optimisations

Hunt and Black [19] reduced the computational cost of the search significantly by performing a beam search with a beam width of 20-40, and report that it did not affect synthesis quality dramatically.

Off-line pre-selection and pre-computation of costs have been reported to make an order of magnitude difference in computational cost with virtually no difference in output

quality. The idea stems from the observation that a large number of units in the database are never joined and that a relatively small number of join costs need to be pre-computed and stored [56]. The idea of pre-computing join costs becomes even more pertinent when used in conjunction with acoustic clustering, as join costs can be computed per cluster. While the pre-computation of join costs makes a significant difference, it does not by itself reduce the size of the database that needs to be shipped with the synthesiser. Donovan presents a comparison between several ideas that made it into the literature [37]. The best algorithms obtain speed increases of around a factor of two, and a resulting database size of 58% of the original database's non-silence units.

2.5 Unit Concatenation

In cases where the units selected were consecutive in the database, the two waveforms can simply be concatenated, but the usefulness of a unit-selection synthesiser lies in its ability to re-sequence the speech units. There are three domains in which the unit concatenation results in discontinuity.

- **Time domain discontinuity** is marked by impulses at the join location where the last sample of the left-hand unit and the first of the right hand unit differs by a large amount. Time domain discontinuities are most easily smoothed by aligning the units according to pitch-marks and then simply cross-fading between the left and right units. Another reason could be that the difference in loudness at the join location may not be natural in the phonetic context.
- **Spectral discontinuities** occur when the vocal tract is in a different position on the right-hand side of the join than on the left. This is a thorny issue, since the configuration of the vocal tract changes all the time during speaking, but suffice it to say here that the difference must not be unnatural.²

Solutions include better distance measures for unit-selection join costs and signal processing to smooth the spectrum of the signal (see Chapter 3).

²See the discussion on acoustic distance measures and the idea of capturing the typical difference between cepstra given the phonetic context in Section 2.4.3.

- **Prosodic discontinuity** is difficult to quantify. General TTS synthesisers rely on the global specification of prosody, the selection of appropriate units and modification of the units to fit the target prosody. Examples include pitch discontinuity, loudness discontinuity and unnatural differences in duration of concatenated units.

Chapter 3 addresses the signal processing side of these issues.

2.6 Implementation

The effect of encoding to side-step the difficult problem of modelling speech is that the synthesiser sounds more similar to the speaker who delivered the database than diphone synthesisers do. Many modern synthesisers not only use the phonetic examples in the database, but also train prosodic models from it. This trend can be pushed even further by using the data as is, and explicitly use the prosody encapsulated in the database. Of course the synthesiser would need significantly more context for each unit to get correctly realised units from the database. Wider contexts require many more units, except if the synthesiser's output domain is restricted.

The type of dialogue system targeted in this study uses a finite-state network to define turns. At each turn the machine responds with a scripted slot-and-filler prompt. There are filler types of which the vocabulary is closed, like digit-strings (telephone numbers, credit card numbers, reference numbers, etc.), dates, times, amounts and numbers. These form the bulk of the required utterances. There are also open types, consisting mostly of proper nouns as found in addresses. The closed types can be completely specified before building the synthesiser, while the synthesiser would need to infer much more detail about the novel words in the open filler types.

Focusing on closed types, and theoretically open types but with restricted vocabulary, allows one to remove all the modelling except for a phonemic specification from the synthesis engine. The limitations placed on the abilities of the synthesiser still allows it to meet the requirements of the SDS, while allowing very high quality, natural-sounding synthesis without the need for good text analysis or prosodic modelling. Although minority languages like Afrikaans and Xhosa have been studied for some time, computer usable lexica, morphological parsers and prosodic models have not been built yet. Deferring

this task to the data makes it possible to build synthesisers that can function in dialogue systems with higher flexibility than simple prompt playback systems and much higher quality than immature general synthesisers.

The success of a limited-domain synthesiser lies in the data it is given (Section 2.6.2), and its ability to find the correct units (Section 2.6.4). Phonemic alignment and other processing of the recorded speech into a unit-selection synthesis database is discussed in Section 2.6.3. A prototype Hotel Reservation System (HRS) dialogue is used for experiments throughout [57].

2.6.1 Text Processing

The limited-domain synthesiser requires a bare minimum of text processing. Phrasing is indicated explicitly by punctuation, with the intention that it influences phrase final prosody directly and predictably. A comma indicates continuation of the utterance, while a full stop, exclamation mark, question mark and colon should be interpreted by the speaker as being utterance final, and he or she will probably realise a low boundary tone, for example.

Orthographic to phonetic conversion is accomplished using the freely available CMU American English lexicon distributed with Festival [48]. A lexicon is available for Afrikaans, but its utility is limited by the Afrikaans practise of forming compound words. Xhosa was transcribed using letter-to-sound rewrite rules [58]. The HRS dialogue in all three languages contained many domain specific words. In Afrikaans and Xhosa, all loan words and proper nouns had to be manually transcribed and entered into an exception lexicon. The exception lexicon is consulted before the language lexicon or letter-to-sound rules.

2.6.2 Corpus Design

The dialogue of the SDS is written as a set of prompts for each state of the dialogue. The bulk of the prompts are static, i.e. they never change. The dynamic prompts contain information items, or slots, to be filled from information provided by the business logic of a voice application. The primary interest for coverage lies with the dynamic prompts. All the prompts are carefully written to have phrasing indicated explicitly by the punctuation.

At first sight, making sure that the database contains all the words it might need seems sufficient. This does not ensure smooth joins between words, however. Since the database provides the correct prosody and does not cover for diphones, drawing units smaller than words from the data to aid in splicing words is not viable either. The solution is to cover for word bigrams.

The natural language generation component of the dialogue system is used to obtain a very large number of sentences intended to at least contain all the word bigrams the synthesiser might encounter. Each sentence is processed into a set of tokens or symbols for the greedy set-covering algorithm. The symbols are constructed so that they encode phenomena relevant to the limited-domain synthesiser. For instance, phrase final words, sentence final words and words inside phrases differ in their prosody, so that the database needs examples of all three. Words immediately following a pause can be concatenated to any preceding pause, and thus pauses could be considered a cost-free transition between different words. Consequently, word bigrams are constructed according to these rules:

- any two words next to one another is a bigram and
- a punctuation mark after a word is counted as a “word”, so that “**rand_**,” is a valid bigram and “**,_and**” is omitted.

Note for example that “**rand_.**” is a different bigram, and will also be present in the selected set. Also, pauses are not considered for construction of bigrams. They are incorporated through their implication by punctuation.

The principle was tested by first generating sentences for selection from the information items we expected: dates, digit-strings, amounts and numbers. The format of each information item is fixed:

dates: “<day of the week>, the <day> of <month>, <year>.”,

digit-strings: “<digit> <digit> <digit>, <digit> <digit> <digit>.”,

numbers: “<number>.”, and

amounts: “<number> rand, and <number> cents.”

Examples of all types except digit-strings in English and Xhosa were generated by simply enumerating all the possibilities, resulting in 372 date sentences, 10 000 numbers (up to 10 000) and 10 000 amounts (up to 10 000 rand). The format for digit-strings was chosen to allow one sentence to contain a number of different events. Enumerating the digit-string sentences would lead to a massive set and so 20 000 random examples were generated.

The digit-strings show the concept of explicit phrasing in the punctuation well: the comma indicates a phrase final word rather than an utterance final one. Indeed, the speakers all pronounced the digit before the comma with a rising boundary tone, indicating that more is to follow. The utterance final word preceding the full-stop carried a falling boundary tone.

A prototype synthesiser built on this set indicated that the numbers seemed to be over-specified. In English, we can insert commas at places where a speaker would typically pause, for example: “four thousand, five hundred and eighty rand.”. This reduces the set of eleven bigrams that would be needed if all transitions from “thousand” and all words following needed to be covered, to three. Note that the word “and” after “hundred” and the “of” in the dates have a similar effect of breaking bigrams.

A final observation that helps to select fewer sentences to record, is that in English numbers, amounts and dates there are eight words that end in “-ty” and seven that end in “-teen”. The suffix could have a similar effect of breaking bigrams as the pause and “and” or “of” had if we had split it from the words. Another way to view this is that these *glue words* or *syllables* take the number of possible transitions from $M \times N$ to $M + N$. Figure 2.7 presents this graphically. Xhosa has a conjunctive writing style which attaches suffixes and prefixes to words in number phrases that indicate noun classes, so that the solution of manually identifying common prefixes and suffixes is even more worthwhile.

When selecting prompts for recording for a particular dialogue, it is desirable that all the word bigrams that link the sentence with the fillers are covered. We do this by generating many random fillers, and putting them into the slots in carrier phrases to construct a large set of random sentences. The greedy algorithm is blinded to all the filler-internal bigrams by removing them from the list of symbols to cover. This ensures that only bigrams at filler boundaries are covered. Many filler-internal bigrams will be

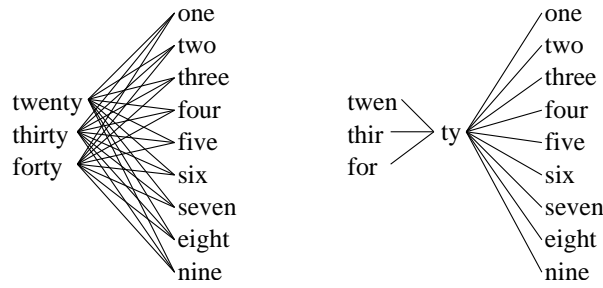


Figure 2.7: The introduction of an artificial *glue word* takes the number of word-word transitions from $M \times N$ to $M \times 1 + 1 \times N = M + N$.

Phrase type	English		Xhosa	
	Split	Not Split	Split	Not Split
Carriers with fillers	123	183	108	172
Digit-strings	17	16	24	23
Dates	8	13	11	321
Numbers and Amounts	4	34	17	156

Table 2.1: Selection statistics for limited-domain fillers. Splitting off common prefixes and suffixes introduces a number of bigram breaks, so that fewer tokens need to be covered and fewer sentences recorded. The splits make a much bigger difference in Xhosa because of its conjunctive writing style.

picked up in passing.

Next, all the bigrams from the sentences selected are again used to seed a selection of “sentences” that contain only one filler per sentence, and nothing of the carriers. This final step mops up the remaining filler-internal word bigrams. The resulting set of sentences is guaranteed to contain at least one example of each word bigram that can occur in the domain. This scheme ensures that the recording set contains a minimum number of full sentences, while still providing all the needed word bigrams.

The number of sentences selected for each filler-only sentence, and the carrier sentences with filled slots are shown in Table 2.1. Figure 2.8 shows the frequencies of occurrence of tokens in the manner of Figures 2.3 and 2.4.

A small change was made in the splitting algorithm for the Afrikaans and English HRS system dialogues: the phrase final words were not split. This is because of the importance of the prosody of phrase final words in indicating the end of a phrase. This change only required about 15 additional prompts to be recorded, and provided high quality phrase

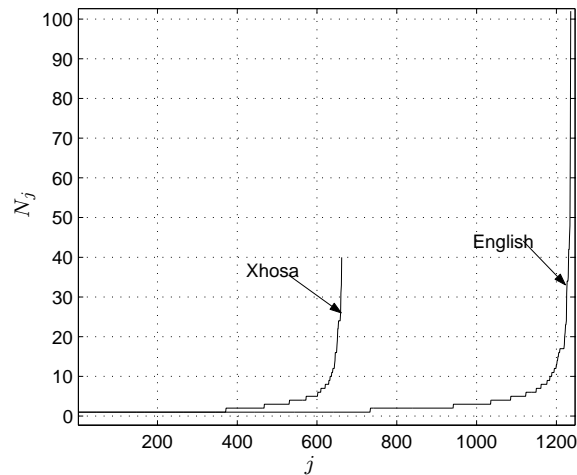


Figure 2.8: The distribution of the word bigrams selected for English and Xhosa HRS dialogues. The bigrams were enumerated from the rarest to the most common. The plots show N_j , which is the number of occurrences of bigram number j . The set is very dense in that most of the bigrams occur only once.

final versions of filler words to the database. In the Xhosa prompts this required no more prompts to be recorded, since in the HRS dialogue no splittable words occurred in a phrase final position.

As has been mentioned before, the dialogue is carefully written to indicate phrasing and pauses explicitly by punctuation. It removes the need for phrasing algorithms in minority languages, and helps the prompt selection process to cover the domain in fewer utterances.

The next step is to record the prompts. Delivering the prompts with consistency in pronunciation and intonation, as well as closely following the pauses indicated by the punctuation, is quite an art. The voice artists must be reminded of the context of each prompt to ensure that they encode the proper semantics into the utterances. A further consideration is that examples of a filler word will be reused in different sentences; consistency in pronunciations is a requirement.

2.6.3 Processing the Synthesis Database

The unit-selection synthesiser needs access to indexing information and several pre-computed features of the speech waveforms in order to find the correct units, compute join costs and concatenate units properly.

Pitch-marks: Pitch-marks are assigned to the waveform, ideally a pitch-mark to each of the pitch-pulses, and uniformly spaced markers of sufficient resolution in the unvoiced regions. The pitch-marks are used in concatenation, and acoustic features are computed pitch-synchronously. Chapter 4 deals with pitch extraction in more detail.

Feature extraction: Pitch-synchronous cepstra for acoustic distance computation are computed for speech frames of 4 times the pitch-period, centred on each pitch-mark. In the Linear Prediction (LP) Pitch-Synchronous Overlap-Add (PSOLA) (Chapter 3) case, the LP coefficients and residual are computed as well.

Phonemic alignment: The synthesiser indexes the data by phoneme labels. The phoneme sequence is generated by the same text processing, lexical look-up and letter-to-sound rules as will be used during synthesis. The phoneme sequence can be aligned with the recorded prompt manually, using a speech synthesiser to provide a reference utterance for Dynamic Time-Warping (DTW) or HMMs. We have experimented with both and given enough data, the HMM method is preferred.

2.6.3.1 Phonemic Alignment using a Speech Synthesiser

After recording, the database consists of sentences, one per wave-file, and a phonemic transcription of each. The problem now is to find the time alignment of the phoneme labels with the waveform. This problem is related to continuous speech recognition, and as such, DTW [1, Chapter 11] has been applied to it. DTW requires a reference waveform, or a set of features of the same phoneme string, against which to align the utterance. The reference waveform can be constructed if a general speech synthesiser is available. Figure 2.9 outlines the process.

The quality of the existing speech synthesiser is irrelevant since only the speech recognition features of the output is used. Prosody and join artifacts have very little effect on the ability of the DTW to align. Experiments were conducted using an American English diphone synthesiser, and 12 dimensional cepstra including C_0 . For some speakers this worked quite well, although the alignment of labels in at least 20% of the English HRS dialogue's prompts had to be corrected manually.

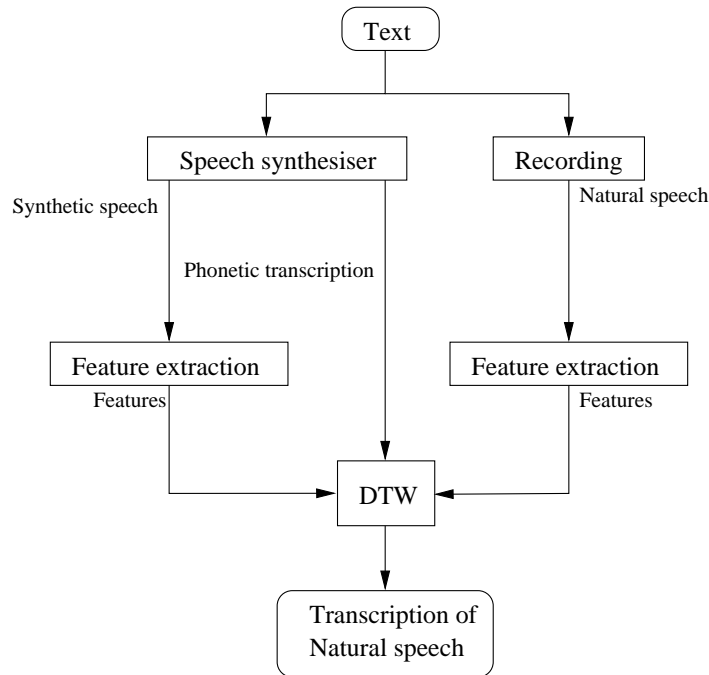


Figure 2.9: Phonemic alignment of recorded speech using a speech synthesiser and DTW. (Adapted from [38].)

The same synthesiser, but with Xhosa letter-to-sound rules, was used to align the Xhosa prompts.³ The results were not usable at all. We expected the DTW to be more robust but the speaker reduced many unstressed syllables to mere whispers, and a number of the vowels were not close to any in the American English synthesiser’s repertoire.

The experience with Afrikaans was similar. The lexicon contained many mistakes, and the female speaker’s speech was very different from the diphone synthesiser’s output; she spoke much faster, and articulated much less clearly during de-emphasised parts. It seems normal to do so in Afrikaans and better care should be taken during recording to ensure better agreement with the test-processing subsystem’s output.

Another approach which seemed feasible, was to repeat the sentence selection task using the recorded prompts as the large set, and then selecting words instead of word bigrams. The selected subset of the corpus can be aligned manually, and used to construct

³The subjective impression of the Xhosa letter-to-sound rules were that, although phonetically strictly accurate, they seemed to over-generate compared with what the speaker actually pronounced. The proper way to improve the orthographic to phonetic conversion would be to use post-lexical processing with parts of speech and stress in mind.

a limited-domain synthesiser that has all the correct words. This synthesiser could then be used to align the remainder. This would work much better, since the same speaker is used for the reference utterances. It still requires manual alignment of 89 out of 359 prompts for English, and 129 out of 351 prompts for Xhosa. In some cases the only really important words that require accurate alignment are the filler words. Only labelling these would reduce the number of words needed, and thus the number of prompts to be segmented manually. However, this reduction would render the synthesiser completely unable to cope with minor changes in the dialogue, and to exploit good join spots in the carrier sentence between filler phrases reliably.

In light of these troubles, HMM alignment provides an alternative if suitable initialisation for the phoneme models can be found and the data scarcity problem dealt with. That is the topic of the next section.

2.6.3.2 Phonemic Alignment using HMMs

The problem of automatic alignment is to align a known phoneme string and a matching waveform; the desired output is a time alignment of the phoneme string. This problem can be solved using HMMs from a speech recogniser with a Viterbi search to perform the segmentation. This section briefly describes the design and training of the models from the TIMIT corpus [59] first, and presents some results of the alignment compared to manual alignment. Embedded re-estimation with much simplified Maximum *a Posteriori* (MAP) adaptation is suggested to improve the alignment results. It concludes with a brief description of forced alignment using the adapted HMMs and a discussion of the results. Some theoretical background is given in Appendix A to clarify the simplified MAP training.

Classical mono-phone continuous density HMMs are structured as depicted in Figure A.1. The HMMs have no skip transitions, and model the observations at each state with a multivariate Gaussian Mixture Model (GMM) with eight components. The models are trained using the forward-backward algorithm on the TIMIT training set (with the closure phonemes merged with the associated stop sounds). This provides a three-state model for each phoneme label in the TIMIT phoneme set. Silence is modelled as a phoneme, but using an eight-state ergodic HMM with a single multivariate Gaussian

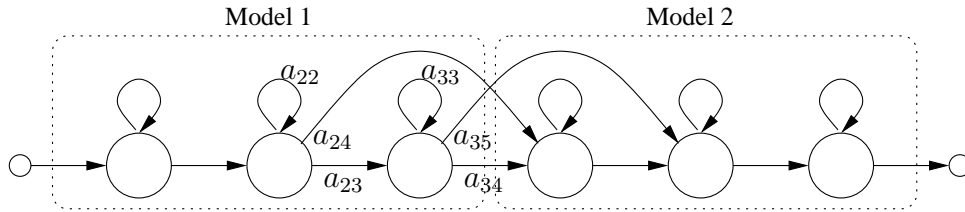


Figure 2.10: Merging vowel models to build diphthong and triphthong models.

Probability Density Function (PDF) to model the observations at each state.

The English HRS synthesiser uses the standard US English phone-set shipped with Festival [48], which is quite close to the TIMIT phone-set for American English. The multi-lingual phone-set devised for Xhosa and Afrikaans differs significantly, so that the alignment system needs new models for the different phonemes. Since the models have already been initialised, and many of the phonemes in the TIMIT phone-set have close counterparts in the multi-lingual phone-set, it makes sense to devise a mapping from the TIMIT phone-set to the multi-lingual phone-set. The mapping is complicated by the fact that the multi-lingual phone-set has more consonants, diphthongs, triphthongs and finer distinctions between vowels than the TIMIT set. The solution is to merge vowel models into diphthongs or triphthongs as shown in Figure 2.10, and map phonemes to their closest counterparts. The skip states are introduced to allow some flexibility in duration in light of the fact that the diphthongs are often somewhat shorter than the two vowels constituting it. The skip weight is made equal to the forward weight, and the weights are then scaled to sum to one. If b represents the weights of Model 1 in the merge, then we compute the weights a of the new model as follows:

$$a_{22} = \frac{b_{22}}{b_{22} + 2b_{23}}, \quad (2.5)$$

$$a_{23} = a_{24} = \frac{b_{23}}{b_{22} + 2b_{23}}, \quad (2.6)$$

$$a_{33} = \frac{b_{33}}{b_{33} + 2b_{34}}, \quad (2.7)$$

$$a_{34} = a_{35} = \frac{b_{34}}{b_{33} + 2b_{34}}. \quad (2.8)$$

These models were used in a forced alignment configuration to determine the locations of the phoneme boundaries. The alignment results are shown in Figures 2.11 and 2.12.

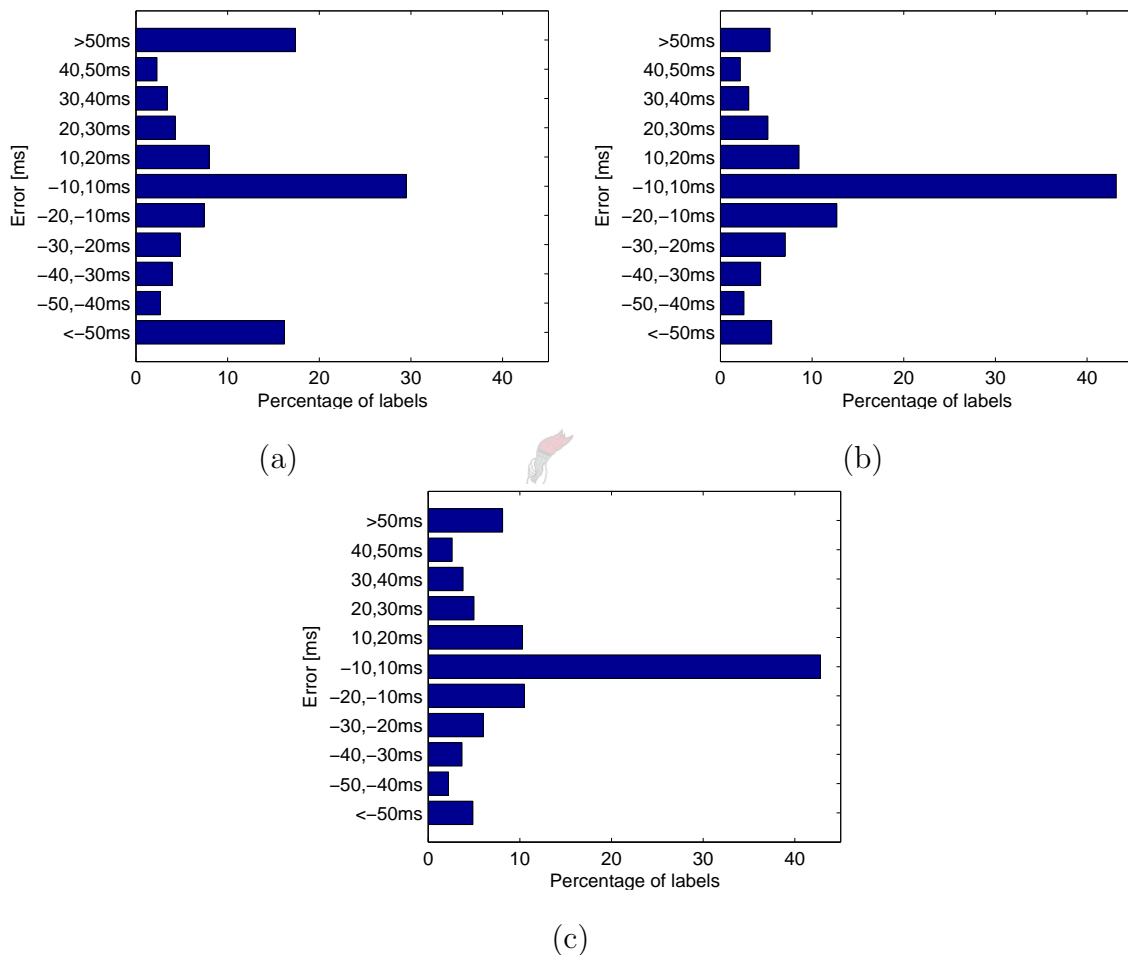


Figure 2.11: Alignment results on the English HRS corpus using (a) the TIMIT models mapped to the US English phone-set, (b) the TIMIT models trained using the simplified MAP procedure and (c) the TIMIT models trained using the simplified MAP procedure on both the English and Afrikaans HRS corpora in the multi-lingual phone-set. The vertical axis shows the range of alignment errors in milliseconds, with the horizontal axis showing the percentage of labels which was within this range from the reference data.

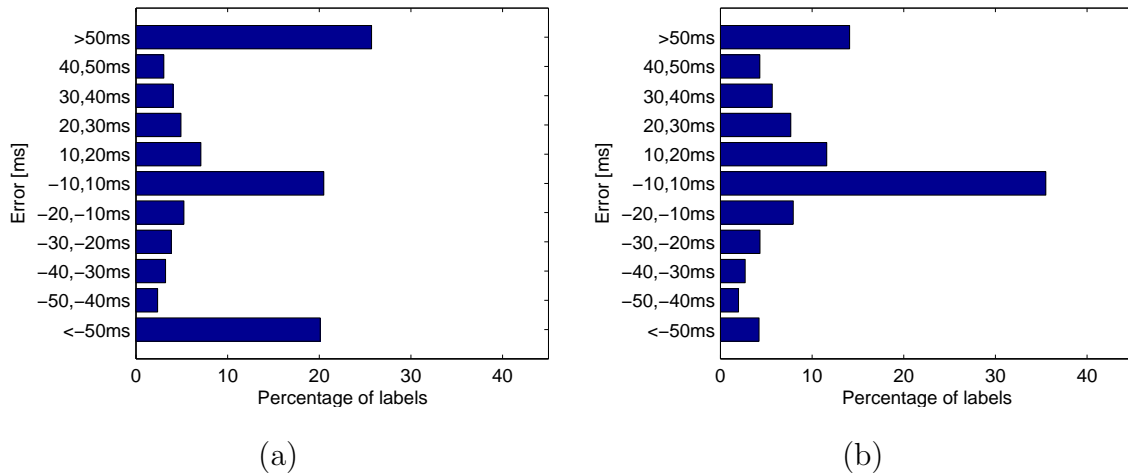


Figure 2.12: Forced alignment results for the Xhosa HRS corpus using (a) TIMIT models directly, and (b) using MAP adapted models.

The first alignment of the three HRS corpora using the TIMIT models was not very useful for unit-selection synthesis: it made too many mistakes that were far wider than the phonemes it was supposed to find. In the English HRS corpus the very different accent of the speaker, the different channel characteristics and inaccuracies in the orthographic to phonetic conversion with respect to the natural speech were to blame. These factors were amplified dramatically in the case of the Xhosa and Afrikaans HRS corpora. In light of this finding, new phoneme models were constructed by mapping a small number US English models to a large number of multi-lingual phoneme models. Diphthongs and triphthongs were constructed by concatenating phoneme models as previously detailed.

Clearly the models need to be adapted to the data we intend to align. Simply training using embedded re-estimation did not work, as the deliberately compact HRS corpus is too small for many of the Gaussian PDFs to be estimated properly. MAP adaptation of the models is a more sensible choice than the Maximum Likelihood (ML) approach [60]. The simplified MAP training algorithm uses the initialised models' parameters as estimates of the prior probabilities, and then adds a constant number of artificial data points to the data used for estimating the parameters of the HMM. In this way, if enough data is available, the estimate is biased to favour the data, and if too little data is available, it favours the prior. This solves the data scarcity problem in an elegant way, while allowing the data that is in the corpus to have a big say in the final models. The adapted models were used for forced alignment, and the results also shown in Figures 2.11 and 2.12. The

results are clearly much better, and tests with the resulting limited-domain synthesiser were very encouraging.

Both the Afrikaans and English HRS corpora were recorded by the same speaker. Although the corpora use different subsets of the multi-lingual phone-set, the overlap was sufficient to encourage joining the corpora. The results are very similar, although subjectively the result produces a slightly better synthesiser.

Although the MAP procedure allows the model parameters to at least be stable in the absence of sufficient data, the data scarcity problem remains however; the model accuracy cannot be good where it has no data.

In keeping with the fact that the bulk of the variation in phonemic context of phonemes in the limited-domain HRS corpora occur at word boundaries, it seems that the boundaries in these locations are the most accurate. However, the phoneme boundaries inside words are typically consistent across different examples of the word, thus not diminishing their utility.

The synthesisers built from the three HRS corpora were tested using a set of test sentences designed using the greedy selection algorithm. The test-set contains all the word bigrams, and thus provides an extensive functional test. In the Afrikaans corpus only about 15 mistakes had to be corrected manually. Subjectively, the resulting synthesisers in Xhosa and English sounded better than the manually aligned versions. Inspection of the labels showed that where it succeeds, the adapted HMM aligner draws its boundaries very consistently, i.e. it always finds the same location in terms of the features for the boundary of a phoneme. The variation introduced by the admittedly hurried, but still time-consuming manual alignment resulted in inconsistently drawn boundaries. These inconsistencies cause distances to be computed between sounds that do not correspond to their labels, making the naive cepstral distance measure work in a somewhat haphazard fashion.

2.6.4 Specialisation of Unit-Selection Synthesis

The limited-domain restrictions imposed on the synthesiser enable some short-cuts that make it much simpler and faster, as long as it can be assured that the synthesiser has the correct units from appropriate contexts.

2.6.4.1 Candidate Selection

The system works with phonemes as units and requires the candidates to come from the same word as is requested in the target specification. The database design phase (Section 2.6.2) ensures that words that occur at the end of a phrase, in the middle of phrases and those at the end of a sentence are all present.

Units in the database are named according to a hierarchical scheme that takes into account which word the unit is from, the prefix or suffix (if any) and the position in the phrase it is from. The prefix or suffix comes from the manually constructed list of words to split as discussed in Section 2.6.2. The hierarchy has three levels. At synthesis time, a level 1 name is constructed for each unit in the target specification, and the database searched for all the examples of that unit. If no level 1 name is found, the database is searched for the less specific level 2 name. If the level 3 name is not found, synthesis fails, since that implies that the word is not present in the database.

The names for split words are built up as shown in Table 2.2. The level 1 name indicates a very specific context, namely the word with its prefix or suffix, and the phrase position as a simple indication of prosody. As the search proceeds through levels 1 to 3, the unit name becomes less specific and it is constructed in such a manner that more examples of a unit name will be available at level 3 than at level 1. The increase in number of examples comes at the cost of looser requirements on the context of the word that the unit comes from. In Table 2.2 <ph> represents the phoneme label and <word> is the “word” part of a word, eg. “six” in “sixteen” or “thoba” in “lesithoba”⁴. Note that “thoba” is not the root morpheme, but it was a convenient choice of identifier given the form of the other Xhosa number words. The <pp> indicates phrase position: *comma*, *mid* or *stop* for phrase final, phrase internal or sentence final, respectively.

In principle, the database should contain level 1 or 2 names for all units that could result from the dialogue that the system was built with. In practice however, changes had to be made to the dialogue. The third level in the hierarchical naming scheme made synthesis more robust in these cases, while restricting the candidates to the smallest and most appropriate set.

⁴“Lesithoba” is the form of the Xhosa word for nine, when referring to a noun of the “ama-” class.

Level	Prefix	Suffix
Word		
1	<ph>_<word>_<prefix>_<pp>	<ph>_<word>_<suffix>_<pp>
2	<ph>_<word>_<prefix>	<ph>_<word>_<suffix>
3	<ph>_<word>	<ph>_<word>
Example (Word)		
1	t_thoba_lesi:_mid	s_six_:teen_stop
2	t_thoba_lesi:	s_six_:teen
3	t_thoba	s_six
Suffix/Prefix		
1	<ph>_<prefix>	<ph>_<suffix>_<pp>
2	no name	<ph>_<suffix>
3	no name	no name
Example (Suffix/Prefix)		
1	l_lesi:	t_:teen_mid
2	no name	t_:teen
3	no name	no name

Table 2.2: The hierarchical naming scheme for unit names. If units named in level 1 in a synthesis request can not be found, the unit type name is relaxed to level 2 to include more units.

Units from all other words, i.e. those not in the split word list, are simply named $\langle \text{ph} \rangle_{\langle \text{word} \rangle}_{\langle \text{pp} \rangle}$ in level 1. In level 2 they do not have a name, and in level 3 they are $\langle \text{ph} \rangle_{\langle \text{word} \rangle}$. This is done so that split words in which the word part coincides with another word may be joined in the candidates they give. The split word “sixteen” is one such example.

While crude from a computational linguistics point of view, this scheme allows small sets of appropriate candidates to be extracted from the corpus, and reduces its size. It makes this possible without relying upon morphological parsing and detailed language specific knowledge, at the cost of being language and domain specific.

2.6.4.2 Join Costs

In the notation of Equation 2.3, the concatenation cost is computed from three weighted sub-costs.

- A fundamental frequency cost defined as the difference between the pitch period T_0 in seconds at the end of unit U_n^{i-1} and at the start of unit U_n^i : $C_{T_0}^c(U_n^{i-1}, U_n^i) = T_{0,U_n^{i-1}} - T_{0,U_n^i}$.
- A weighted Mahalanobis distance between the final cepstral vector of candidate U_n^{i-1} and the first vector of a candidate U_n^i for the next unit. The cepstral vector excludes the zeroth cepstral component.
- The difference between a normalised zeroth cepstral component, or C_0 . The normalisation makes the variance of the zeroth component comparable to the other components.

The weights for the cepstral distance was set to 1. In this strictly limited-domain synthesiser we regard continuity in energy and pitch to be very important. The weights of the fundamental frequency sub-cost and the C_0 difference were set through inspection in order to emphasise them over the cepstral distance in cases where the pitch and power discontinuity is high.

The cost of concatenating units that happen to be consecutive in the corpus is set to 0, encouraging the Viterbi algorithm to select contiguous sequences of units. Another

interpretation is that the selection algorithm becomes “greedy” for longer, non-uniform units from the database.

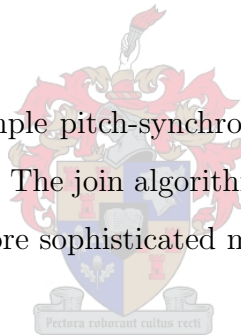
2.6.4.3 Unit Selection

Candidates are selected by a Viterbi search. The hierarchical naming of candidates in the database restricts the search already, and in most cases there are fewer than five candidates to consider.

The only exception lies in selecting candidates for pauses. The simplest solution is to make one artificial pause unit and not consider the ones in the database to be candidates at all. In that way, different phrases are completely decoupled, and many pause units are removed from consideration to make the search faster.

2.6.4.4 Concatenation

Units are concatenated in a simple pitch-synchronous way. The pitch-marks nearest to the unit boundaries are aligned. The join algorithm cross-fades from one unit to the next between two pitch-marks. A more sophisticated method is described in Chapter 3.



2.6.5 Results

This section presents four examples. In the first example synthesis worked very well, and the result reflects that. All the units were realised by level 1 representatives. The second example comes from a sentence in the dialogue that needed to be modified. Thus a level 2 unit was used, since the word in the proper position in the phrase was not available. The third example illustrates a situation that results from the fact that we take prosody straight from the database. The speaker delivered some of the sentences in a lower pitch, and the best path found by the Viterbi search contained one large prosodic discontinuity. The utterance is phonetically correct, but the prosodic effect makes the sentence less understandable. In Example 4 a label was misplaced and part of an incorrect phoneme included in the synthesis. The accompanying CD-ROM contains the audio for these examples and more.

2.6.5.1 Example 1: Perfection

The following sentence was synthesised:

“So you’ll be arriving on the sixteenth of February, two thousand.”

The unit-selection synthesiser was able to produce the sentence using only units in their level 1 specifications. The labelling for this database was manually corrected, and therefore the phoneme labels describe the waveform correctly. Figure 2.13 shows the resulting waveform. The concatenation points indicate phoneme boundaries where concatenation between units that were not consecutive in the database occurred.

The concatenation between [ə] and [ɪ] in “February” occurred in a voiced region. Figure 2.14 shows the concatenation spot in more detail. Note that the pitch cycles are synchronised perfectly, and the rise in amplitude is smooth.

Incidentally, this example shows the potential for mismatch between the interpretation of the linguistic processing and the speaker during recording. The linguistic front-end used the American English pronunciation lexicon shipped with Festival, and the speaker spoke in a South-African English accent, potentially resulting in different pronunciations of “February”. She pronounced it in the American way however. The word context would have allowed the correct waveform to be selected, even though the label is technically incorrect.

2.6.5.2 Example 2: A good recovery

The following randomly generated test sentence contained a request for the word “thousand”, followed by a comma.

“Sorry, the departure date the twelfth of April, two thousand, is before the arrival date, the thirteenth of July, twenty fifteen.”

At level 1, the hierarchical naming scheme will search for units named <ph>_thousand_comma. Since that failed, it looked for <ph>_thousand, the level 3 name. Figure 2.15 shows the result for the phrase “two thousand”. The prosody is acceptable, and although the concatenation between [u] and [θ] may seem rather abrupt, it occurs at a point where the ex-

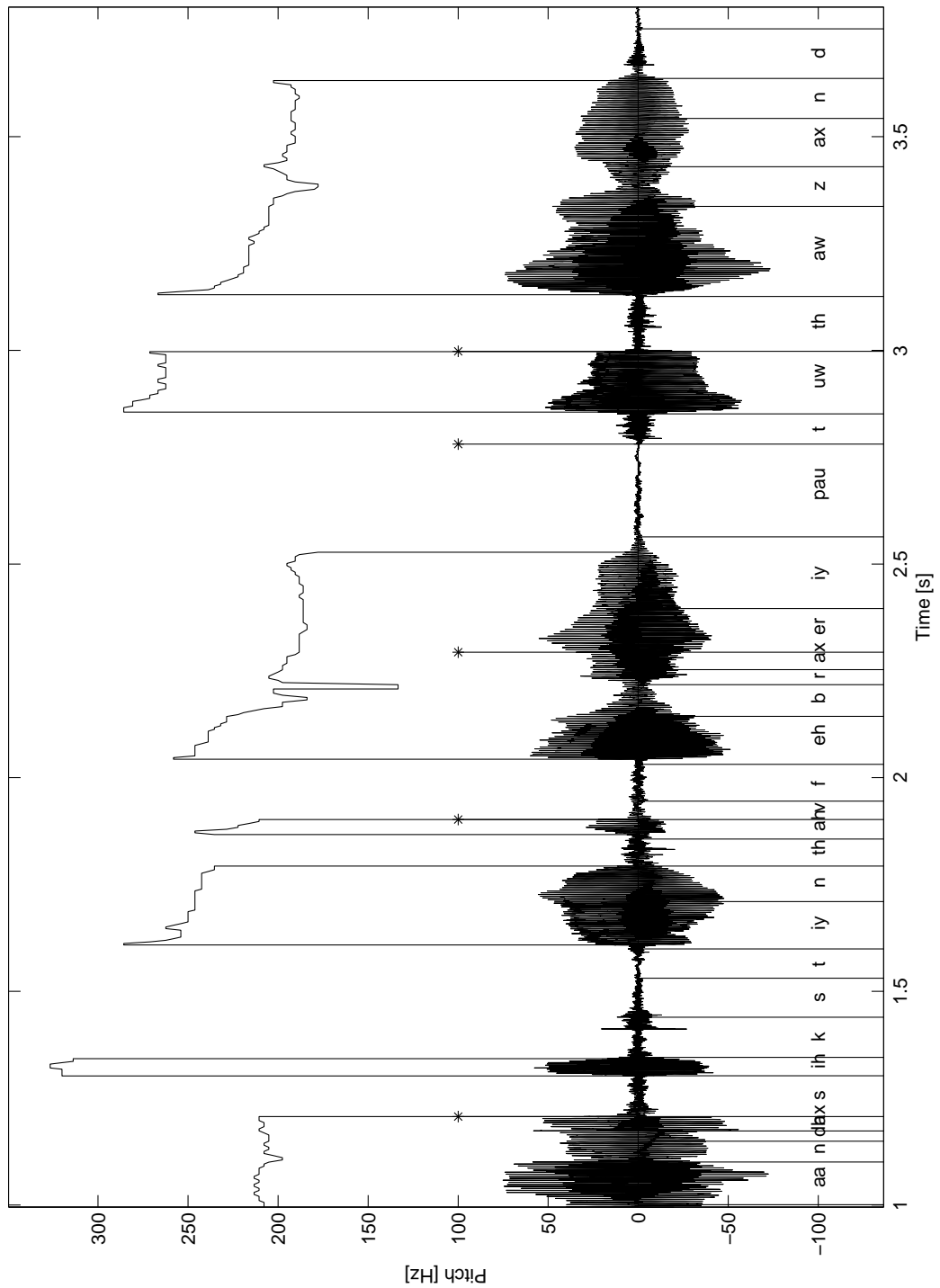


Figure 2.13: Synthesis Example 1. The plot shows the waveform, phoneme labels and boundaries, as well as the concatenation points, indicated by asterisks.

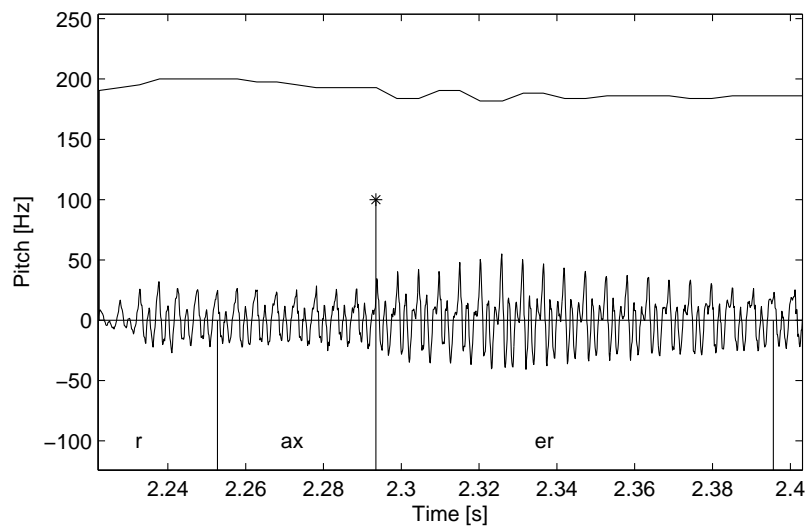


Figure 2.14: Detail of “February”. The amplitude also matches up well on either side of the concatenation spot.

citation signal changes from harmonic to noise and the spectral envelope changes rapidly. Thus it is not noticeable.

2.6.5.3 Example 3: Prosodic Discontinuity

Although somewhat contrived, this example illustrates the problems of prosodic and spectral discontinuity. The original utterance is

“Now for the number of rooms.”

For purposes of dialogue flow, this was changed to

“Now for the rooms.”

The synthesised utterance is shown in Figure 2.16. There are join locations on either side of the [ə] of the word “the”. (The [ə] is shown by /ax/ in Figure 2.16.) The unit for [ə] was extracted from a context where a [b] followed it. Figure 2.16 also details the spectrogram of the region around the concatenations. The run-up from the [ə] to the [b] can be seen in the spectrogram. This spectral discontinuity is clearly audible.

The pitch contour varies rapidly through a large range. Such a wide range from start to finish of such a short utterance is unnatural.

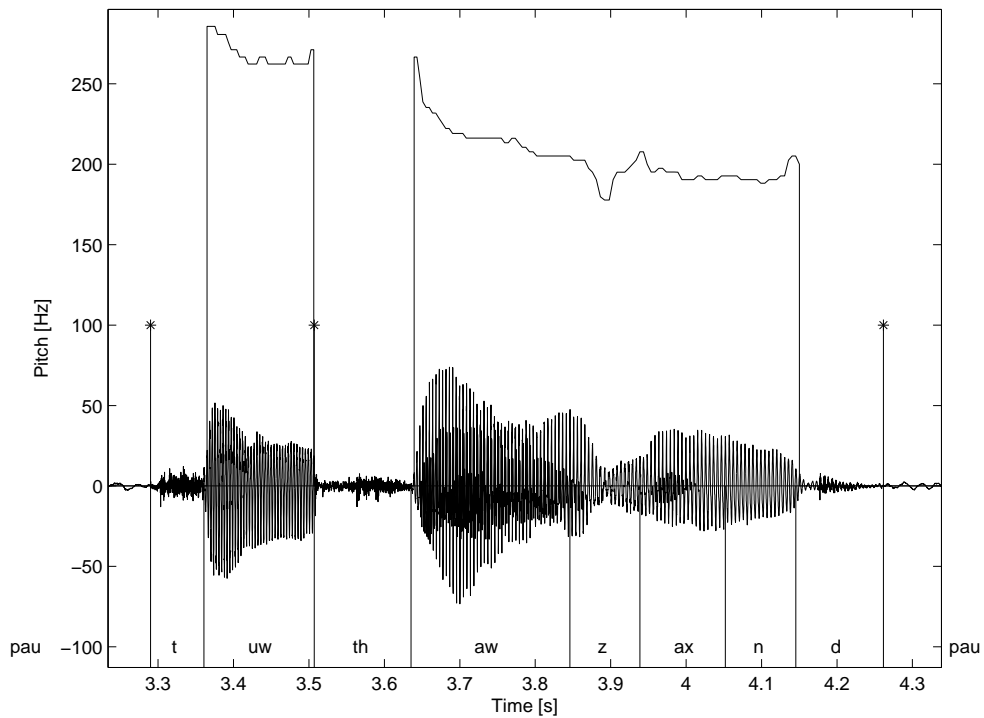


Figure 2.15: The result of the synthesis of “... April, two thousand, is ...”. The word “thousand” was recorded in a different context.

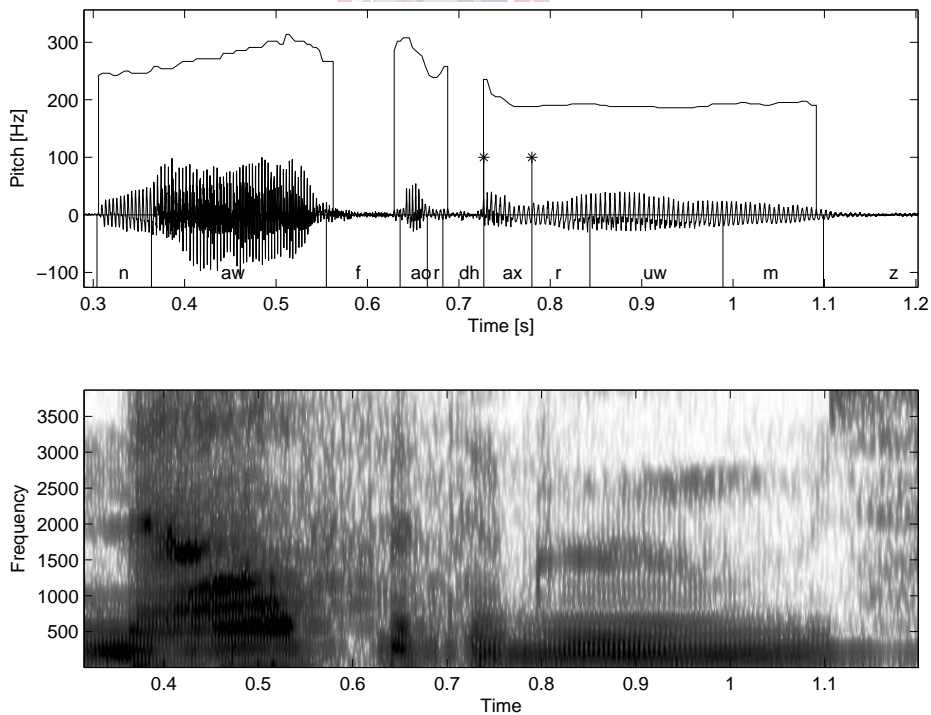


Figure 2.16: The result of synthesis where a sharp pitch and spectral discontinuities occur. The effect can clearly be seen around 0.8s in both the time domain plot and the spectrogram.

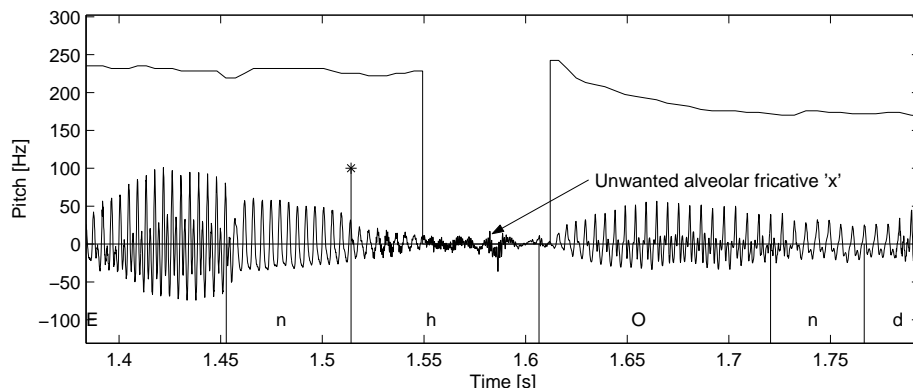


Figure 2.17: Detail of the synthesised signal for “eenhonderd”. The unwanted [x] sound appears because of a labelling error.

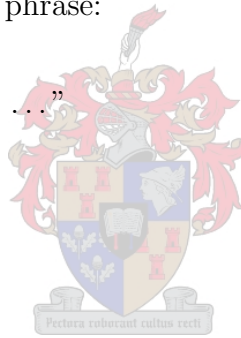
2.6.5.4 Example 4: Labelling Errors

By far the most common reason for failure of the synthesiser is labelling errors. Figure 2.17 illustrates part of the Afrikaans phrase:

“... eenhonderd en vyftig ...”

or phonetically:

[ɛnhənərt en fɔiftəx].



The recording that delivered the phonemes after the first [n], contained a labelling error that include the preceding [xə] from the word “nege”. As a result, it was synthesised as

[ɛnxəhənərt en fɔiftəx].

2.7 Conclusions & Suggestions

The HRS was demonstrated in English, Afrikaans and Xhosa. The system worked very well. In most cases the prompts were very clear, and the reactions of first-time users led to the belief that the semantics encoded in the prompts were appropriate.

The approach followed here makes it possible to rapidly build custom voices that are capable of more variability in prompts than the standard concatenation-based systems

used in Dual Tone Multi-Frequency (DTMF)-based interactive voice response systems. Computational efficiency is also such that 50 channels could easily be serviced using an entry-level personal computer.

That said, the system still has its problems. The Xhosa voice worked very well, and virtually no examples of synthesis were found that had clear synthesis artifacts. The English and especially the Afrikaans systems suffered from many more problems. The most notable was prosodic discontinuity in the forms of large pitch discontinuities and even changes in speaking rate and articulation quality. Utterances that were recorded in the same session were generally more similar.

The delivery of the recordings is certainly to blame. Over-dramatisation of the Afrikaans and English prompts made consistency much more difficult. The Xhosa speaker did not exhibit this tendency. The Xhosa prompts were also delivered by a male. It appears that men not only have lower pitch, but also modulate their pitch in a much smaller range.

Recording the prompts, selecting and instructing the speaker, and designing the dialogue all remain in the realm of art. It is only with some experience that good results can be obtained. This is a natural consequence of encoding rather than modelling. Chapter 3 takes a partial turn toward modelling again, with encouraging results.

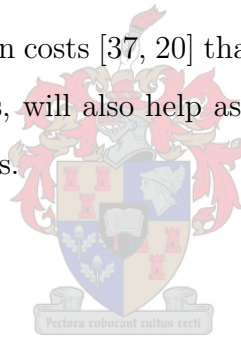
There are several areas in which to extend and improve on the groundwork laid here.

- The major effort in building any new voice (assuming the existence of the phoneme set and lexica) remains the alignment of phoneme labels. Embedded re-estimation followed by forced alignment is the most automatic way to do it, and the data scarcity problem can be solved in two ways.
 1. If more than one system is to be built, it makes sense to use previously recorded prompts by the same speaker in the embedded re-estimation step.
 2. Adding more data by including a small set of prompts designed to cover diphones in emphasised and de-emphasised syllables, will dramatically improve labelling. At the same time it will provide data for rendering novel words, such as proper nouns.

- Labelling errors and especially fine errors will continue to plague data-driven systems. Optimal coupling is a good angle of attack on the problem of labels not exactly corresponding to the data or not being totally consistent. Its major shortcoming, however, is its lack of awareness of the nature of the underlying signal.

In principle it should be possible to treat voiced and unvoiced (rapidly changing or sustained) speech in different ways, identifying good join spots in the signal. This would be expensive computationally but could also be run off-line to identify join locations as part of the build process.

- Alternatively, and especially in light of the abilities developed in Chapter 3, it would make sense to explicitly consider join locations in the middle of vowels and nasals. As mentioned before, concatenating units on phoneme boundaries is not the best way in all cases.
- The phonetically aware join costs [37, 20] that recognise good directions and rates of change in various contexts, will also help as development proceeds in the direction of synthesising novel words.



Chapter 3

Speech Modification

3.1 Introduction

The limited-domain unit-selection synthesiser developed in Chapter 2 has its failings. Most notable are spectral and power discontinuities and unnatural pitch transitions. This chapter concerns itself with the modification of the speech signal to smooth the concatenations and otherwise improve the quality and naturalness of the concatenated speech.

In the past, model-driven speech synthesisers used models of speech signal production to generate the speech signal. Among these were formant synthesisers and articulatory synthesisers. In the spirit of the modern trend of encoding rather than modelling subtle signal properties, we would rather modify the speech by a small amount. This amounts to a return to modelling of simpler, better-understood aspects to repair damage to the signal caused by concatenation. A requirement is that it leaves the encoded higher-level properties intact.

The bulk of the speech synthesisers described in the literature use speech modification methods to make the prosody, as expressed in duration and pitch, conform to targets set by the linguistic components of the synthesiser. Setting the prosodic targets requires detailed linguistic information to be deduced from the input text, approaching general TTS and thus being beyond the scope of this work. This chapter focuses on means to reach pitch and duration targets and to only set pitch targets for smoothing concatenations. The techniques presented here are important, as they form the basis for signal synthesis in TTS. This approach provides the infrastructure needed for future research on intonation

and duration modelling.

3.2 Literature Overview

The literature on speech modification and speech coding contain much overlapping material, with the drive to find more compact parameterisations for the speech signal as the common goal. Speech coding applications want more compact information to transmit, and those wishing to modify the speech signal, hope to find more relevant parameters to modify. The literature seems to focus on the distinction between parametric and non-parametric approaches [61, 18, 62], but it is sensible to present the taxonomy of speech modification methods in terms of how they separate various aspects of the signal.

A study of voiced speech leads, via the lossless tube model of the vocal tract [1, Chapter 3], to a model of speech production known as the source-filter approach. This model views the speech production process as a source signal passed through a filter whose properties vary through time. Voiced speech contains periodic source waveforms originating in the larynx, and the filter consists of the entire vocal tract, sometimes including the nasal cavity. The vocal tract has three main resonant cavities, and constrictions sometimes shorten the vocal tract (consider the [p], where the vocal tract length is effectively zero), and move the place of excitation.

Of particular interest to speech synthesis is co-articulation effects. In producing an utterance, the articulators have to change through a sequence of configurations. It is well known that to produce intelligible speech, the synthesiser has to mimic the transitions. The diphone synthesiser mentioned in Chapter 2 explicitly captures transitions between a small number of idealised sounds, usually phonemes. In contrast, the unit-selection synthesiser lifts the restrictions on the number of sound types and therefore vastly increases the number of transitions. By its data-driven nature, the unit-selection synthesiser is subject to failure due to missing phenomena in the data. Relatively simple signal models can provide some of the missing information, enabling a graceful recovery.

Section 3.2.1 gives a qualitative view of various aspects of the speech signal, based on the physiology of speech production. Section 3.2.2 discusses the most prominent directions in speech modification.

3.2.1 Signal Components

We now turn to manifestations of the speech production mechanism on the speech signal. Figure 3.1 illustrates a speech signal of the spoken word “flattery”, [flætə.ɪ], and the power spectrum in the middle of each phoneme. The unvoiced consonants, [f] and [t], look like coloured noise, albeit with different temporal properties. Indeed, the unvoiced consonants have been successfully produced in vocoders using Gaussian noise modulated in time and frequency. The voiced sounds, [l], [æ], [ə], [ɹ] and [i] are all clearly periodic when viewed on a short enough time-scale. The periodicities show up clearly as harmonics in the frequency domain. The voiced signal also contains a significant time-modulated noise component, due to turbulence noise from air forced through the vocal tract. Generally speaking, the two components overlap in frequency, with the harmonic structure clearly dominant at lower frequencies.¹

The power spectra of voiced speech sounds have similar fine structure, while the main difference lies in the envelope of the spectra. The same holds for the unvoiced sounds. This property has long been exploited in speech recognition and coding, where the envelope is encoded by a small number of parameters, discarding the phonemically redundant information about the fine spectral structure.

The source-filter model of speech production holds that the envelope is due to the filter realised by the vocal tract, and that the rapidly varying excitation, containing various mixtures of harmonic and time-modulated noise signals, affects the fine structure. The filter (spectral envelope) expresses certain resonances, called formants. The relationship between the first and second formants has been strongly linked to specific vowels.

Speech modification methods that aim to modify the pitch have to separate the phonemic information contained in the vocal tract configuration as expressed in the filter parameters, the spectral envelope and formants, from the excitation signal. Simply resampling the signal will raise or lower the pitch, but it will also shrink or stretch the envelope, changing the formant frequencies. While the vowel identity is typically retained, the perceived vocal tract length changes, resulting in the well-known cartoon chipmunk effect.

¹This section gives an intuitive account based on [1, Chapter 2].

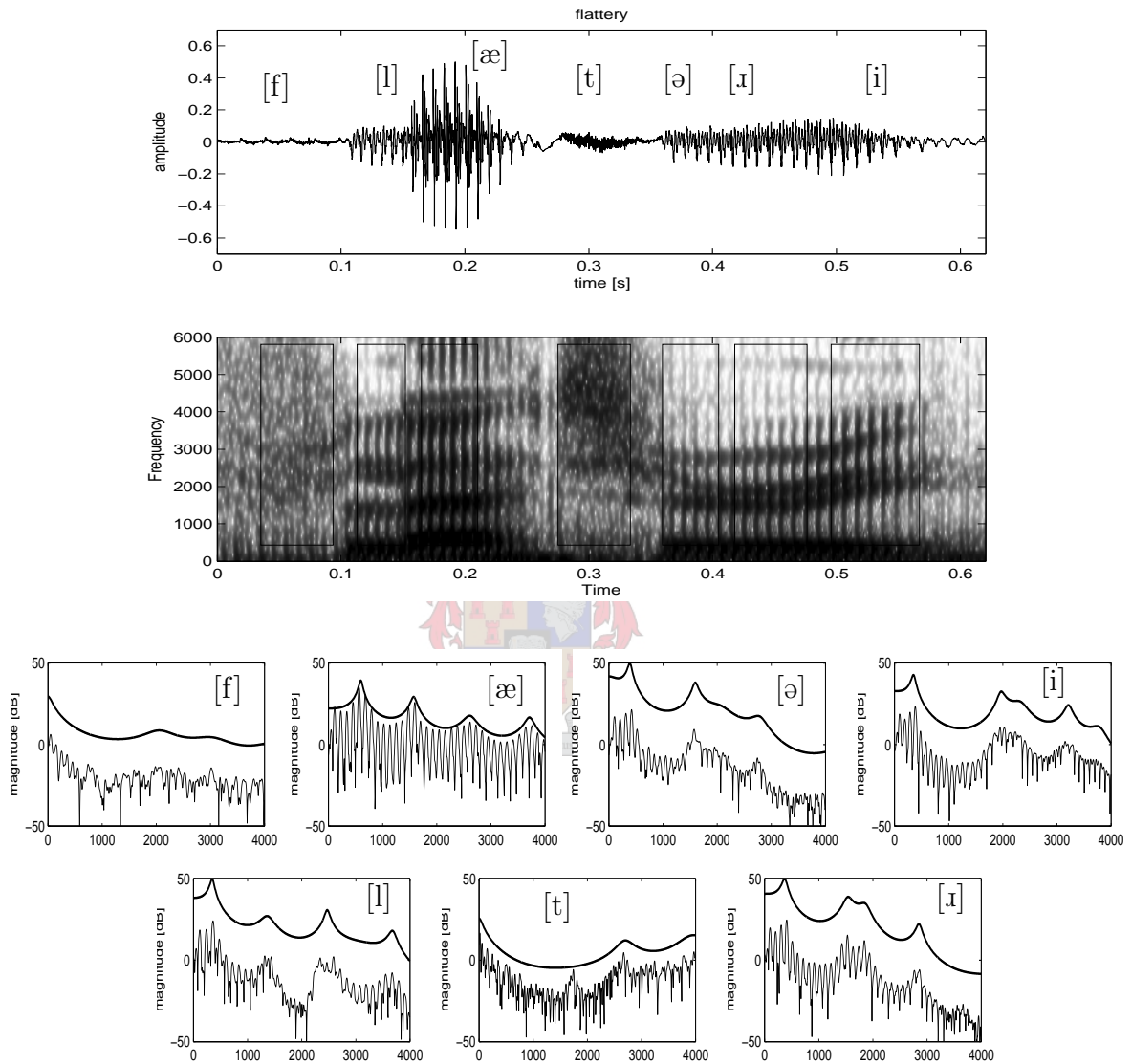


Figure 3.1: The speech signal of the word “flattery”, [flætəɪ], illustrating a fricative, vowels, a lateral, an approximant and a stop sound. The power spectrum in the middle of each speech sound is also shown. The smoothed envelope obtained by LP analysis is shown by a thick, smooth line.

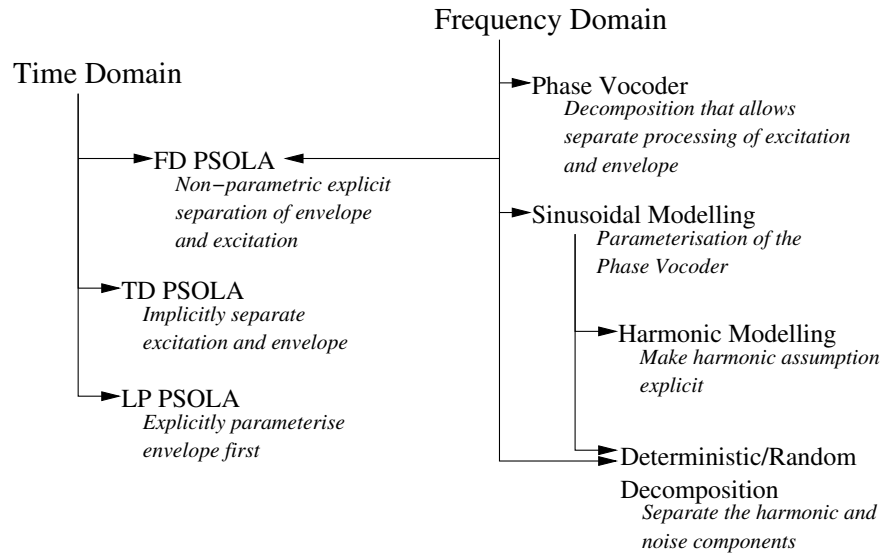


Figure 3.2: A taxonomy of speech modification methods. All these separate the vocal tract effect from the excitation. Some go further and decompose the excitation into harmonic and noise components. The extent to which the three different components are parameterised varies.

3.2.2 Speech Modification

This section describes methods of separating the vocal tract effect from the excitation signal under assumption of the validity of the source-filter model, and possibly even further separating the noise and deterministic components of the excitation signal. A taxonomy of speech modification methods is depicted diagrammatically in Figure 3.2. The frequency-domain approaches have many parametric and non-parametric ways to use the information in the signal.

Section 3.2.2.1 glosses over the plethora of frequency-domain methods for separating speech components. Section 3.2.2.2 describes a simple time-domain method that has been the baseline for pitch and duration modification for the past decade. Section 3.2.2.3 describes a hybrid time and frequency-domain method and is emphasised as it forms the basis for the implementation discussed in Section 3.3. Section 3.2.3 considers options for smoothing the spectrum around concatenations.

3.2.2.1 Frequency-domain Methods

The phase vocoder as first published by Flanagan [63] revolutionised the field of digital signal processing methods for modifying speech. It relies on explicit decomposition of

the signal into time-frequency components through a sub-sampled Short-time Fourier Transform (STFT). The sub-sampled STFT is realised by a bank of filters, equally spaced in frequency. Later refinements by Portnoff, Seneff and Crochiere [61] established the links between the filterbanks proposed by Flanagan and the STFT. Modern techniques use the Fast Fourier Transform (FFT) and its inverse in various configurations, performing manipulations directly in the frequency domain. Frequency-domain techniques often use constant-length frames with constant frame spacing. The frames are typically spaced so that they overlap. Frequency-domain methods operate under narrow-band conditions to resolve harmonics in the spectrum. The longer window imposed by the narrow-band requirement also explains why frequency-domain techniques are often reported to have trouble with rapid transitions in the speech signal. Pitch-synchronous frames with and without pitch-related frame lengths have been used to increase the time-resolution of the parameter estimation [18, 64, 65].

Modification of the harmonic content of signals, while keeping formant information intact, requires separation of these two components. As previously illustrated in Figure 3.1, this amounts to modifying the harmonic frequencies of the signal spectrum, leaving the envelope intact. Extraction of the excitation part of the spectrum can be done in a number of ways, including LP or all-pole modelling [66, Chapter 11][1, Chapter 5], homomorphic filtering (the cepstrum) [67][66, Section 4.6][1, Chapter 6] and heuristics that use the peaks in the spectrum [18], effectively flattening the spectrum. Resampling and padding the spectrally flattened Discrete Fourier Transform (DFT), and re-applying the previously separated envelope achieves the required pitch modification. The significant problems that all frequency-domain methods face, are

- maintaining phase coherence among frequency components and continuity from frame to frame, and
- modelling and modification of unvoiced speech.

Further developments, related partly to increases in computing power, exploit the fact that a periodic signal can be described by a line-spectrum. This description is effectively a sum of sinusoids, and encodes only the peaks in the spectrum as sinusoids. Each sinusoid is fully described by its phase, amplitude and frequency. Many different methods of extracting the sinusoids from the signal have been proposed:

- spectral peak picking and tracking schemes [17, 67],
- instantaneous frequency methods [68, 69],
- analysis by synthesis [70, 71],
- matching pursuits [65], and matching pursuits extended to include human auditory modelling [72], and
- optimisation of the amplitude and phase parameters of a linear combination of harmonically related sinusoids [18].

Sinusoidal methods amount to a parameterisation of the DFT. Its proponents claim not only improved ability to modify pitch by larger factors, but also the flexibility to model more subtle aspects of the sound. The pitch can be modified by moving the harmonic sinusoids around under the envelope, effectively resampling it. Some go one step further and enforce the harmonic relationship between the sinusoids. This simplifies the phase coherence problem, and makes a more compact database and more computationally efficient synthesis possible [73, 18].

Sinusoidal decomposition methods share the difficulty of modifying unvoiced speech with other pitch modification methods—the synthesised sounds gain a tonal quality at even moderate pitch raising or time-stretching factors. Solutions to this problem require isolating the sinusoidal or deterministic part of the signal from the noise or random part. The possibility of separating the components along the frequency axis is an often-cited advantage of sinusoidal decompositions.

The Harmonics plus Noise Model (HNM) assigns a signal adaptive maximum voiced frequency [18], above which no more sinusoids are extracted. The harmonic spacing between sinusoids and their amplitudes are manipulated to modify pitch. The noise component is modelled using LP in the frequency-domain, and the time-modulation of the noise is modelled by computing the average power from non-overlapping 2ms windows. Synthesis realigns the time-domain peaks of the noise signal with the modified harmonic signal, maintaining the reportedly important coherence between the harmonic and noise components.

Others assign the deterministic or random label to specific sinusoidal tracks, obtained by associating peaks in successive frames and grouping them into longer-lived sinusoids.

During synthesis they could randomise the phases [67] or even the frequencies [73] of those sinusoids deemed to be noise. It is also possible to subtract the sinusoids from the original signal to obtain the sinusoidal modelling residual and then consider that to be noise [71].

The most sophisticated methods seem to be iterative procedures which decompose the periodic and aperiodic components on a frame-by-frame basis in the frequency domain [74, 75].

3.2.2.2 Time-domain Methods: Time-domain PSOLA

PSOLA has become the de facto standard method for pitch modification. Its conceptual simplicity, low computational complexity and high-quality for moderate pitch and duration modification factors (0.5 to 2.0), have made it ubiquitous. The three original flavours are Time-domain (TD) PSOLA, Linear Prediction (LP) PSOLA and Frequency-domain (FD) PSOLA. The latter two are hybrids since they perform explicit separation of components in the frequency domain before further time-domain processing. FD PSOLA is not described further.

TD PSOLA analyses the signal $x(n)$ at pitch-synchronous time-instants, referred to as analysis time-instants or pitch-marks. Let η_m be the m 'th discrete analysis time. The length L_{win} of the analysis window $h_m(n)$ is related to the local pitch period at the analysis time-instant, $N_0(\eta_m)$, by a factor which is typically in the range $\mu \in [2, 4]$. We choose the window length L_{win} to be the smallest odd integer such that $L_{win} \geq \mu N_0(\eta_m)$. If a continuous window function $h_t(t)$ that is non-zero for $-\frac{1}{2} \leq t \leq +\frac{1}{2}$ and zero elsewhere, is sampled at discrete intervals, then

$$h_m(n) = \begin{cases} h_t\left(\frac{n}{L_{win}}\right), & -\frac{L_{win}+1}{2} \leq n \leq \frac{L_{win}-1}{2} \\ 0, & \text{elsewhere.} \end{cases} \quad (3.1)$$

The analysis short-time signals $x_m(n)$ are then

$$x_m(n) = h_m(n)x(n + \eta_m). \quad (3.2)$$

Note that the analysis frames are centred around $n = 0$.

TD PSOLA performs synthesis by first computing the sequence of synthesis time-instants η_k and associating each of them with zero or more analysis time-instants. A

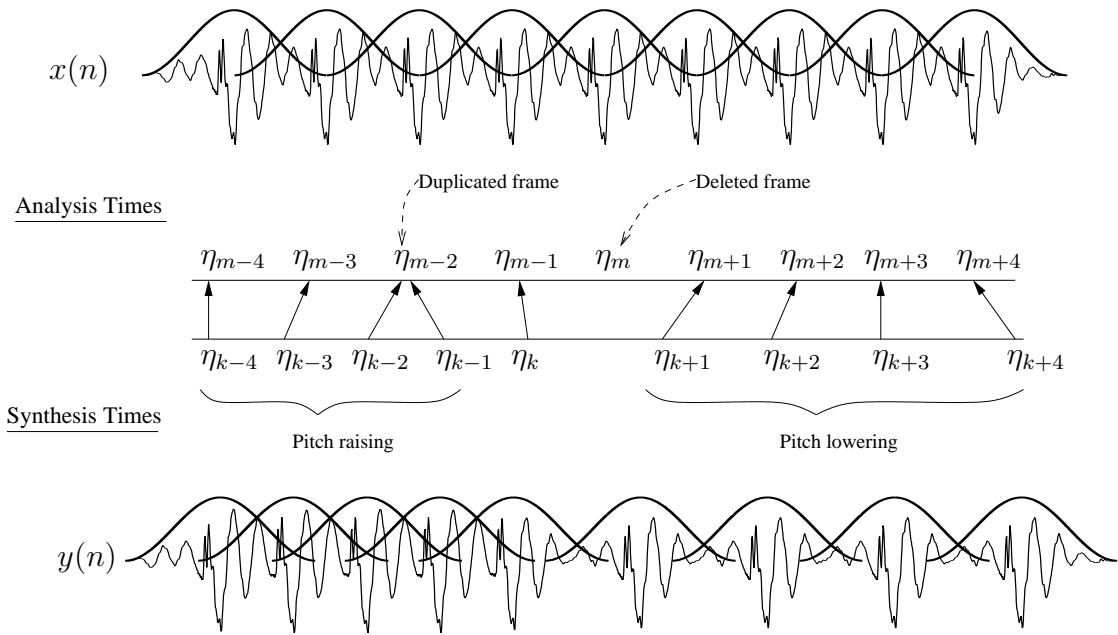


Figure 3.3: A diagrammatic representation of the mapping of synthesis time-instants to analysis time-instants, $m(k) : k \rightarrow m$. The mapping of time-instants constitute the temporal translation of short-time signals that achieves time-scaling and pitch modification.

simple and convenient algorithm to compute the synthesis time-instants is described in Section 3.3.2.2. The synthesis time-instants are placed according to the desired local pitch and are associated with analysis time-instants according to the time-warping to be applied, expressed by writing m as a function of k . The mapping of synthesis parameters to analysis parameters, $m(k)$, may in general be many-to-many, i.e. some k may be associated with zero or more consecutive m , and vice versa. This implies that some short-time signals may be repeated or deleted in the synthesis process. Figure 3.3 illustrates the mapping between the two time axes diagrammatically.

The second step in synthesis is to add the short-time signal $x_{m(k)}(n)$ at its computed location η_k in the output signal. The simplest form of overlap-add to obtain the output signal $y(n)$ can be written as

$$y(n) = \sum_k x_{m(k)}(n - \eta_k). \quad (3.3)$$

More sophisticated methods of performing the overlap-add are available [61]. The simple approach in Equation 3.3 leads conveniently to an expression for the frequency-domain interpretation of TD PSOLA. This synthesis method is also particularly well suited to LP PSOLA, as it is efficient and the particulars of the LP synthesis algorithm solves the

inter-frame continuity problem (see Section 3.3). The analysis and synthesis signals are also illustrated in Figure 3.3.

Although “astonishingly simple” [61], TD PSOLA results in very high quality synthesis for moderate pitch and time modification factors. A frequency-domain interpretation shows how the time-domain procedure results in much the same operation as frequency-domain approaches: under certain conditions the spectral envelope is effectively resampled at the harmonic frequencies of the synthetic signal. In this way it accomplishes the same as the simplest forms of sinusoidal methods.

To see the resampling property of TD PSOLA, we follow the analysis of Moulines and Charpentier [64] for the deterministic, or periodic, component of the signal. It is adapted slightly to better correspond to the implementation in Section 3.3. Assume for simplicity that

- the analysis signal is periodic and has a constant pitch period of N_0 samples,
- that a constant pitch modification factor β is applied, and
- that β is also applied to time scale modification so that the mapping $m(k) : k \rightarrow m$ is one-to-one.
- Furthermore, the analysis window $h_0(n) = h_t(n/L_{win})$, is assumed to be a square window such that the length of the short-time signals are exactly $L_{win} = \beta N_0$ and odd, and therefore do not overlap.

By these assumptions the analysis time-instants are

$$\eta_m = mN_0, \tag{3.4}$$

and the synthesis time-instants are

$$\eta_k = k\beta N_0, \tag{3.5}$$

and $m(k) = k$. The synthesis equation becomes

$$\begin{aligned} y(n) &= \sum_k x_0(n - k\beta N_0), \\ &= \sum_k h_0(n - k\beta N_0)x(n - k\beta N_0). \end{aligned} \tag{3.6}$$

Since $x(n)$ is periodic, $y(n)$ consists of translated copies of the same prototype waveform, which we denote $x_0(n) = h_0(n)x(n)$. As a consequence, $y(n)$ is also periodic. The Fourier series expansion for the discrete periodic signal $y(n)$ is defined as [66]

$$y(n) = \frac{1}{\beta N_0} \sum_{q=0}^{\beta N_0-1} c_q e^{j2\pi qn/\beta N_0}, \quad (3.7)$$

with the Fourier coefficients given by²

$$c_q = \sum_{n=0}^{\beta N_0-1} y(n) e^{-j2\pi qn/\beta N_0}. \quad (3.8)$$

The continuous and periodic Discrete-Time Fourier Transform (DTFT) of a single period of $y(n)$, namely the DTFT of $x_0(n)$ is defined as

$$X_0(\omega) = \sum_{n=-\infty}^{\infty} x_0(n) e^{j\omega n}. \quad (3.9)$$

Since $x_0(n)$ has a limited time-span due to the square window $h_0(n)$ of length βN_0 , the DTFT becomes

$$X_0(\omega) = \sum_{n=0}^{\beta N_0-1} x_0(n) e^{j\omega n}. \quad (3.10)$$

Comparing this spectrum of a single prototype analysis short-time signal to the Fourier coefficients of the synthesised signal $y(n)$ as expressed in Equation 3.8, leads to the final result:

$$c_q = X_0\left(\frac{2\pi q}{\beta N_0}\right). \quad (3.11)$$

The Fourier coefficients are complex numbers representing the magnitude and phase of sinusoids that make up the line spectrum of the periodic signal. The spectral envelope of these sinusoids is the *DTFT of a single prototype short-time frame*.

Since the analysis window length varies as the inverse of the pitch period modification factor β , raising the pitch shortens the window and increases its bandwidth, while lowering it lengthens the window and decreases its bandwidth. Figure 3.4 shows the Fourier transforms for values of $\beta \in \{0.5, 1.0, 2.0\}$. The window bandwidth can be seen to smooth

²The normalisation factor $\frac{1}{\beta N_0}$ has been added to the synthesis equation (Equation 3.7) of the Fourier series expansion to match [64].

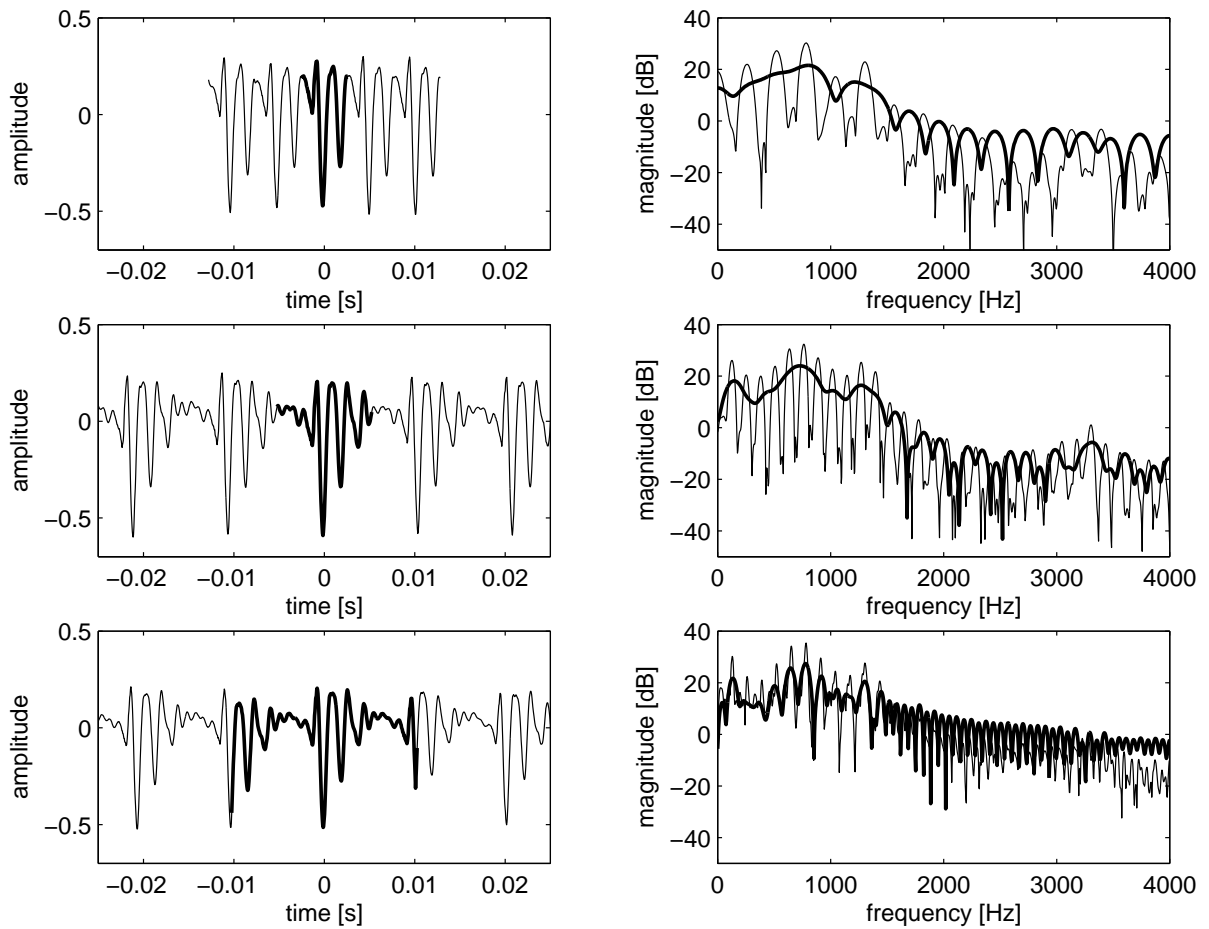


Figure 3.4: The Fourier transform of a single short-time analysis signal (thick line) and those of synthesised signals with $\beta \in \{0.5, 1.0, 2.0\}$. The window bandwidth can be seen to smooth out formants when $\beta = 0.5$ and resolve pitch harmonics when $\beta = 2.0$. The thick line shows a single short-time frame in both frequency and time domains.

out formants when $\beta = 0.5$ and resolve pitch harmonics when $\beta = 2.0$, resulting in the attenuation of every second harmonic. A further distortion follows from the high side-lobes of the square window, which masks the lower-magnitude high-frequency components.

PSOLA (as cast here) resamples the spectrum uniformly, while the most successful sinusoidal methods selectively resample only the region of the spectrum deemed to be harmonic. They can then apply more appropriate modelling to the noise component of the signal. Higher modification factors and more stringent quality requirements necessitate this decomposition between noise and harmonics.

Note that the square window used in the synthesis equation results in a buzzy quality as it effectively introduces impulses of various amplitudes at frame boundaries. In practice,

TD PSOLA requires an overlapping window that tapers to zero at the edges to be applied, as shown in Figure 3.3. Also note that at values of $\beta > 1.5$, copies of the waveform will start to resemble the original waveform. This is seen in the frequency domain as the resampling of a harmonic spectrum for $\beta = 2.0$ in Figure 3.4, where every second harmonic is attenuated. Therefore $\mu = 4$ is typically the maximum allowable value if a tapered window like a Hamming window is used.

3.2.2.3 Hybrid Methods: Linear Prediction PSOLA

The other two variants of PSOLA, LP PSOLA and FD PSOLA, explicitly separate envelope modelling (in the frequency domain) and modifying the spacing between the pitch harmonics (in the time domain). Advantages of LP PSOLA is that the LP analysis inherently smooths the spectrum to obtain the envelope and models it parametrically. This allows different frame lengths to be used for estimation of the envelope parameters and PSOLA analysis and synthesis. The separation also allows deliberate modification of the spectral envelope for

- smoothing the spectral envelope over join locations in concatenative synthesis, and
- adapting the envelope to the pitch and amplitude modification [76].

The analysis frames are centred on the pitch-synchronous analysis time-instants η_m used for PSOLA analysis. The frame length is set to a multiple of the local pitch period, $\mu N_0(n)$, and $\mu > 2$, so that it incorporates a small number of pitch periods. In this work, LP analysis of order d constructs the Yule-Walker equations using the biased short-time autocorrelation estimate and solves them using the Levinson-Durbin recursion. The result is the $(d + 1)$ -dimensional vector \mathbf{a}_m of filter coefficients a_i , $i \in \{0, 1, 2, \dots, d\}$, of a minimum-phase all-pole model of the vocal tract, with $a_0 = 1$ [66, Chapter 11]. The sequence of LP filters represents the progression of the vocal tract through the utterance. Examples of the spectral envelopes encoded by the LP coefficients are depicted in Figure 3.1.

In the z domain the transfer function of the filter can be written as

$$H_m(z) = \frac{1}{\sum_{i=0}^d a_i z^{-i}}. \quad (3.12)$$

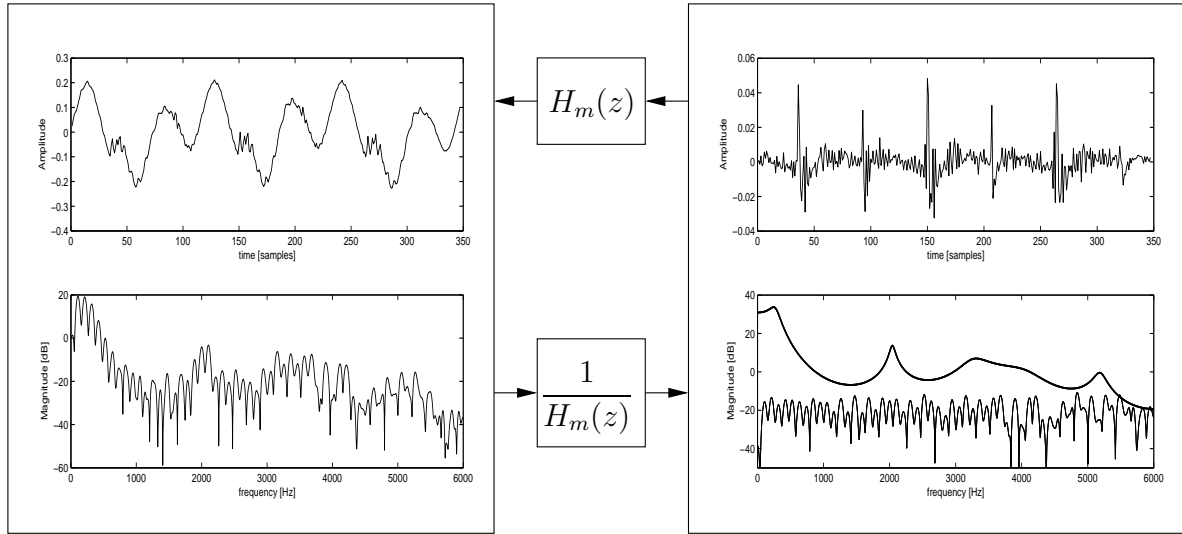


Figure 3.5: The linear prediction residual. The left-hand-side panel shows the original signal in the time and frequency domains, and the right-hand side the residual signal. The frequency-domain plot on the right hand side also shows the magnitude response of the LP filter as a thick smooth line.

Figure 3.5 shows a short-time analysis frame in the time and the frequency domains, before and after inverse filtering with $1/H_m(z)$. The spectrum of the *residual* or *linear prediction error* can be seen to be approximately flat, and harmonic. In this way the vocal tract and excitation information have been effectively separated. The envelope may be re-applied by filtering the residual through $H_m(z)$.

Figure 3.6 depicts the result of the PSOLA process on the residual of the signal in Figure 3.4. The residual short-time signals were also windowed using a Hamming window of length μN_0 , with $\mu = 2$. This smooths the spectrum at modification factors of $\beta \geq 2$, and so removes the distortion of the spectrum caused by a window that resolves the pitch harmonics. Because of the flatness of the spectrum compared to the steep spectral tilt in the spectrum of the original short-time signals (Figure 3.4), the window's side-lobes do not distort higher frequencies either.

3.2.3 Spectral Smoothing for Concatenative Synthesis

Concatenation of speech segments can lead to unnatural discontinuities in the spectral envelope. The sudden jumps in formant frequencies degrade the perceived quality and understandability of the synthetic speech, even though the spectral envelope sometimes

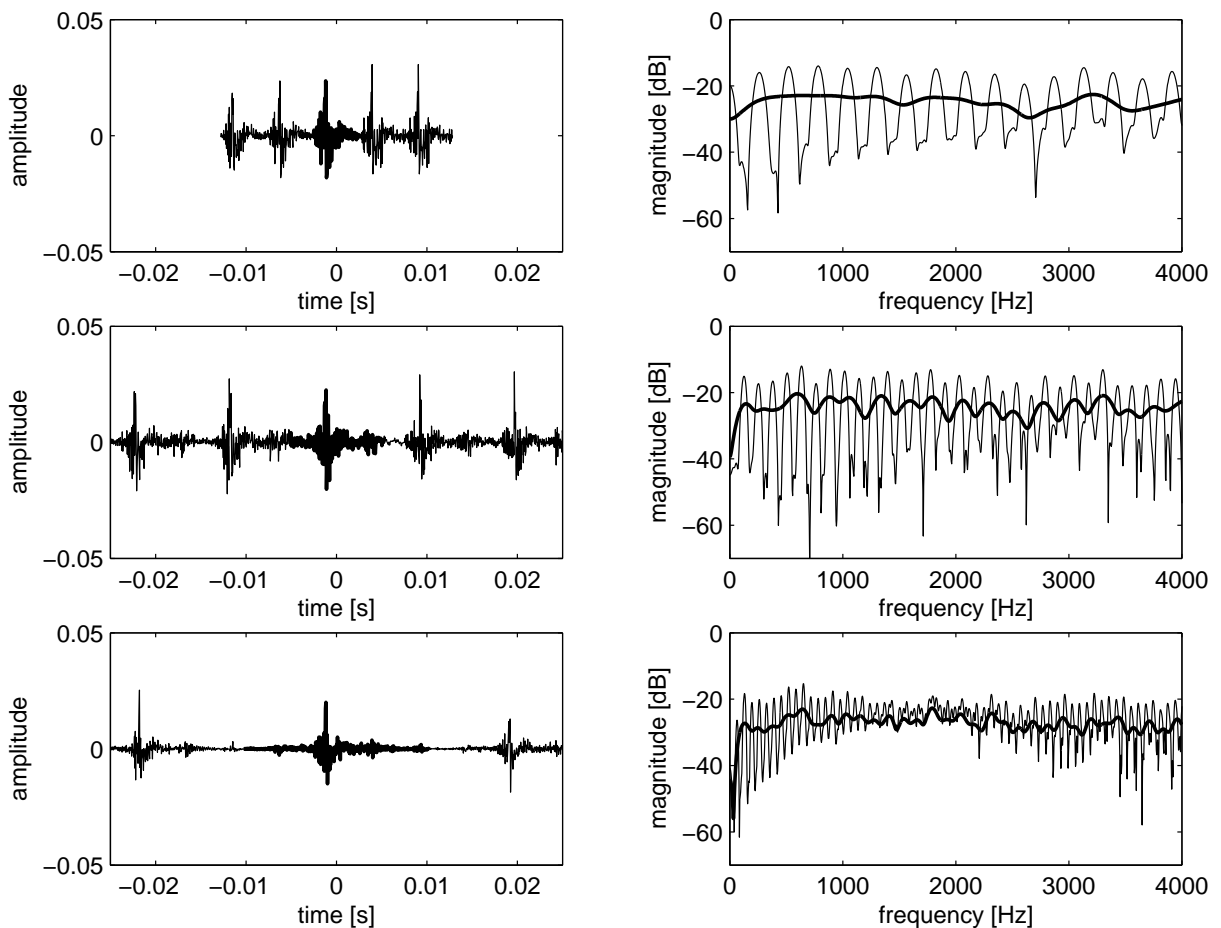


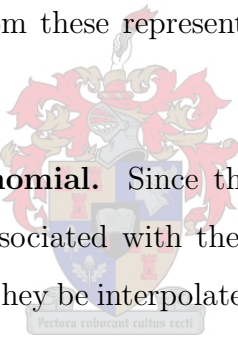
Figure 3.6: PSOLA on the LP residual, again for $\beta \in \{0.5, 1.0, 2.0\}$. In this case $\mu = 4$ during LP analysis and $\mu = 2$ during PSOLA analysis of the residual. Note that the DSP window's sidelobes have no influence on this signal at higher frequencies due to its lower dynamic range, and the formants are not distorted since they have been separated from the time-domain signal by LP inverse filtering. The Hamming window lowers the frequency resolution of the analysis so that pitch harmonics are not resolved when $\beta = 2.0$.

changes rapidly during natural speech. In some contexts, rapid changes are natural, while in others, the rapid change in the synthesised speech cannot be realised by the human speech apparatus, and thus sounds unnatural.

Some work has been done to alleviate the discontinuities by devising smarter join costs (see Section 2.4.3), extending the corpus [77, 53] or modifying the signal to smooth the discontinuities.

The envelope can also be processed heuristically to smooth spectral discontinuities that arise from concatenative synthesis [78, 79]. The need for envelope smoothing first arose from voice coding, where spacing consecutive frames further apart reduces the bit-rate.

Chappell and Hansen [78] compared interpolation of various alternative representations of LP parameters, as the polynomial coefficients themselves cannot be interpolated without a good chance that the LP filter may become unstable. The filter coefficients \mathbf{a}_m can be converted to and from these representations without loss of information [1, Chapter 5].

- 
- **Roots of the LP polynomial.** Since the complex conjugate roots of the LP polynomial are directly associated with the resonances that the filter encodes, it makes intuitive sense that they be interpolated over frames in concatenation regions. The difficulty with using the roots is that the roots from frames on the left must be associated with roots on the right of the concatenation region. Especially in filters of higher orders (12–24), this becomes more error-prone.
 - **Line-spectral Frequencies.** Another direct representation of the resonances, line-spectral frequencies also do not suffer from the association problem.
 - **Interpolation of the Cepstrum.** While interpolation of the cepstrum will yield stable envelopes, it is very “brute force” as there is no physical interpretation; the formants simply fade out and in at different locations.
 - **Interpolation of Log-Area Ratios (LAR) [79].** Log-area ratios are derived from the reflection coefficients computed during Levinson-Durbin solution of the Yule-Walker equations. LARs are traditionally used where the reflection coefficients must be quantised for transmission. Reflection coefficients may also be used. Although

the trajectory of the interpolated filter coefficients will differ, it has been found experimentally to make no perceptible difference.

Section 3.3 elaborates on an implementation and experiments performed on interpolating LARs.

3.3 Implementation

A waveform synthesiser was implemented for the limited-domain concatenative synthesiser described in Chapter 2. It uses the limited-domain corpus (with parameters for LP PSOLA synthesis computed and stored beforehand) and concatenates the waveforms into the utterance required by a synthesis specification.

3.3.1 Analysis

LP PSOLA requires the sequence of analysis time-instants, the associated short-time signal and the LP filter coefficients for each short-time signal. Each utterance in the database is automatically pitch-marked and segmented into voiced and unvoiced signals in a way that gives priority to finding periodic regions in the waveform. The time-instants are positioned in locations that work well with the time-domain modification technique. Chapter 4 is devoted to the topic of finding good analysis time-instants.

The input waveform is filtered using a pre-emphasis filter with transfer function

$$H(z) = 1 - \alpha z^{-1}. \quad (3.13)$$

A value of $\alpha = 0.98$ was used throughout. This filter introduces a zero near the frequency $z = 1$, which lessens the severe spectral tilt in voiced speech. This has been shown to improve the stability of the LP analysis. It can also be argued that the zero cancels a pole in the vocal tract system that is due to the glottis and not part of the vocal tract configuration determined by phonemes [1]. LP analysis is performed on Hamming-windowed frames centred on the analysis time-instants. The frame length is set to four times the pitch period. The biased autocorrelation is used to compute the Levinson-Durbin recursion. The LP order is determined experimentally to be as low as possible

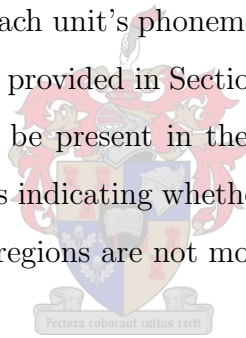
without causing audible distortion at pitch modification factors of $\beta = 0.5$ and $\beta = 2.0$. The waveforms were sampled at 16kHz, and an LP order of 20 was used throughout these experiments.

Each short-time analysis frame is inverse filtered using its LP filter coefficients and stored.

3.3.2 Synthesis

3.3.2.1 Inputs

The synthesiser takes as input the corpus of phonemically labelled utterances, the extracted features as described above and a synthesis specification. The synthesis specification provides an index of the recorded utterance, as well as the start and end times of each unit to be concatenated. Each unit's phoneme type is also indicated. More detail on the recognised phoneme types is provided in Section 3.3.2.3. If duration and pitch targets were available, they would also be present in the synthesis specification. Furthermore, the source data contains markers indicating whether the frame is voiced or unvoiced. The pitch and duration of unvoiced regions are not modified.



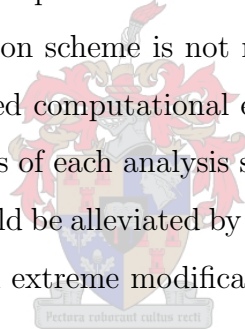
3.3.2.2 Computing the Synthesis Time-Instances

The synthesis time-instants define the pitch in the voiced regions of the synthetic waveform, and their association with analysis time-instants through $m(k)$ determine the time-warping function. They can be computed from

- the analysis time-instants,
- the desired pitch $N_0(\eta_m)$ or pitch modification factor $\beta(\eta_m)$ at each analysis time-instant, and
- the desired time-stretching factor at each analysis time-instant. The time-stretching factor could also be given indirectly by specifying the source and target times of beacons. This implies a piece-wise linear, monotonically rising time-warping function.

The synthesis time-instants are produced by keeping two time variables, η_m , the *source time*, and η_k , the *target time*. They are initialised to the same utterance starting value η_0 . They are related through $\eta_m = D(\eta_k)$. The algorithm “walks” forward in target time. Each new synthesis time-instant is obtained by adding the target pitch period: $\eta_{k+1} = \eta_k + N_0(D(\eta_k))$. If $\eta_k > \eta_m$, then m is incremented until $\eta_m > \eta_k$, keeping the analysis and synthesis at the same phonemic point in the utterance. At each iteration the new values for k and m are added to the mapping of synthesis time η_k to analysis time η_m , $m(k)$. The resulting lock-step between m and k takes care of the repetition and deletion of frames.

This simple algorithm associates the LP parameters and short-time residual signal with the closest values to $D(\eta_k)$ in the analysis set of short-time parameters. More sophisticated versions could interpolate the LP coefficients and residual signals between neighbouring sets of values. The parameter variation from frame to frame seems sufficiently slow that the interpolation scheme is not needed. It makes no audible difference and incurs considerably increased computational expense. Large time-scale modification factors will cause multiple copies of each analysis short-time signal, resulting in a characteristic droning sound which could be alleviated by interpolation. Typical speech synthesis applications do not require such extreme modifications.



3.3.2.3 Spectral Smoothing of Concatenations

The limited-domain concatenative synthesiser often spliced waveforms at places where both sides of the join are voiced, but have different formant frequencies. The resulting spectral discontinuities are quite prominent and are often reinforced by accompanying pitch discontinuities. The discontinuous joins usually occur where a data shortage exists in the corpus, or where the indexing is incorrect or inconsistent.

Similar joins occurred between voiced and unvoiced waveforms, but these were much less noticeable, as the spectral envelope and the excitation characteristics change rapidly in these regions anyway. Furthermore, the pitch can vary widely across unvoiced regions in natural speech. Therefore we focus on joins between voiced phonemes.

After computing the synthesis time-instants and copying the synthesis LP parameters from their analysis counterparts, the transitions are smoothed by interpolating the spec-

trum as represented by the reflection coefficients at each synthesis time-instant. Smoothing is applied to a region of pre-determined size to the left and to the right of the join location η_J . That is, the indices of the synthesis times at the left-hand side boundary of the smoothing region, at the join location and at the right-hand side boundary of the smoothing region are N_L , N_J and N_R , respectively.

Transitions between different classes of phonemes can be observed to progress over different lengths of time. The transition from the [l] to the [æ] in the word “flattery”, for example, (see Figure 3.1) is very fast compared to the transitions around the approximant [ɹ]. Similarly, short time constants hold for transitions from vowels to nasals. Transitions from vowels to glides ([y], [w]), approximants ([r]) and other vowels are generally slower. Another effect is that nasal and lateral phonemes cause co-articulation effects in the surrounding vowels, rather than bear the effects themselves. This is because the semi-constricted configuration of the vocal tract allows less variation than does the open configuration that produces the vowel.

Bearing this in mind, Table 3.1 is used to determine the time span of the regions left and right of the join location to be smoothed. The times were determined heuristically. Slightly longer time constants worked well for a large number of observed cases. In some cases, however, the units to be concatenated are very short due to labelling variation or inclusion in de-emphasised syllables. Longer smoothing time constants have been found to cause the smoothing operation to destroy very short phonemes. Instead of making the times longer, the region from η_L to η_R is stretched by a factor of 1.2 by inserting some new synthesis frames into the concatenation region. The stretching factor was introduced because vowels, nasals, laterals and approximants are sometimes much shorter than expected, and it allows shortening the amount of time taken up by the smoothing regions. Some additional awareness of phoneme type and length would be useful to make this algorithm more robust.

Figure 3.7 depicts the smoothing process for concatenation between the two vowels [ɛ] and [a]. A target value for the d -dimensional vector of smoothed reflection coefficients at the join location η_J , $\tilde{\kappa}_J$ ³, is computed by taking the arithmetic average of values of the

³The reflection coefficients κ_m are calculated from the LP coefficients \mathbf{a}_m . It is an equivalent representation, and can be converted back easily [1].

Phoneme Class	Time (seconds)
Vowel	0.04/0.03
Nasal	0.01
Lateral	0.01
Approximant	0.05
Voiced Stop	0
Voiced Fricative	0

Table 3.1: Smoothing time values for various voiced phoneme classes. There are two times for vowels, the shorter is used when smoothing it into a nasal or a lateral.

reflection coefficients at η_L and η_R :

$$\tilde{\kappa}_J = \frac{\kappa_L + \kappa_R}{2}. \quad (3.14)$$

The target weight w_k , $k \in [L, R]$, for interpolation of the spectrum progresses linearly from 0 to 1 from η_L to the join location η_J . From the join location it progresses linearly from 1 to 0 at the right-hand edge of the smoothing region, η_R . The smoothed reflection coefficient values are computed by:

$$\tilde{\kappa}_k = w_k \tilde{\kappa}_J + (1 - w_k) \kappa_k. \quad (3.15)$$

This scheme morphs the spectral envelope to the target spectral envelope in $J - L$ pitch cycles, and fades it from the target envelope to the envelope on the right-hand side of the join in $R - J$ pitch cycles. The asymmetric weighting around the join spot allows the tempo of the interpolation to be varied according to the phonetic class. This simple scheme avoids the association problems of interpolating the roots of the LP filter polynomial and still results in smooth formant movements, as can be seen by the example in Figures 3.8 and 3.9.

The waveform interpolation on the short-time LP residual frames operates using the same weights [78]. The two short-time residual frames to be added are aligned on their centres, and values outside the time range of each frame is taken to be zero.

Informal experiments showed that if only spectral smoothing or waveform interpolation was applied, the discontinuity was still clearly audible. This can be attributed to the fact

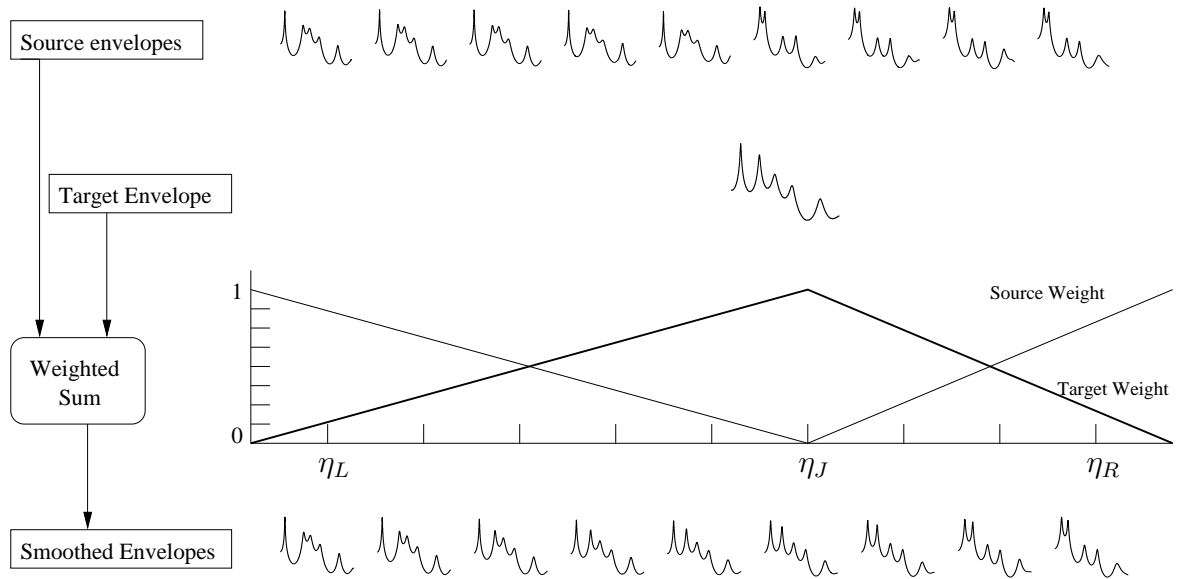


Figure 3.7: Smoothing concatenations by interpolating the reflection coefficients. The example spectra were obtained by interpolation of a splice of $[\varepsilon]$ to $[a]$. Figure 3.8 shows the detail of the formant movements

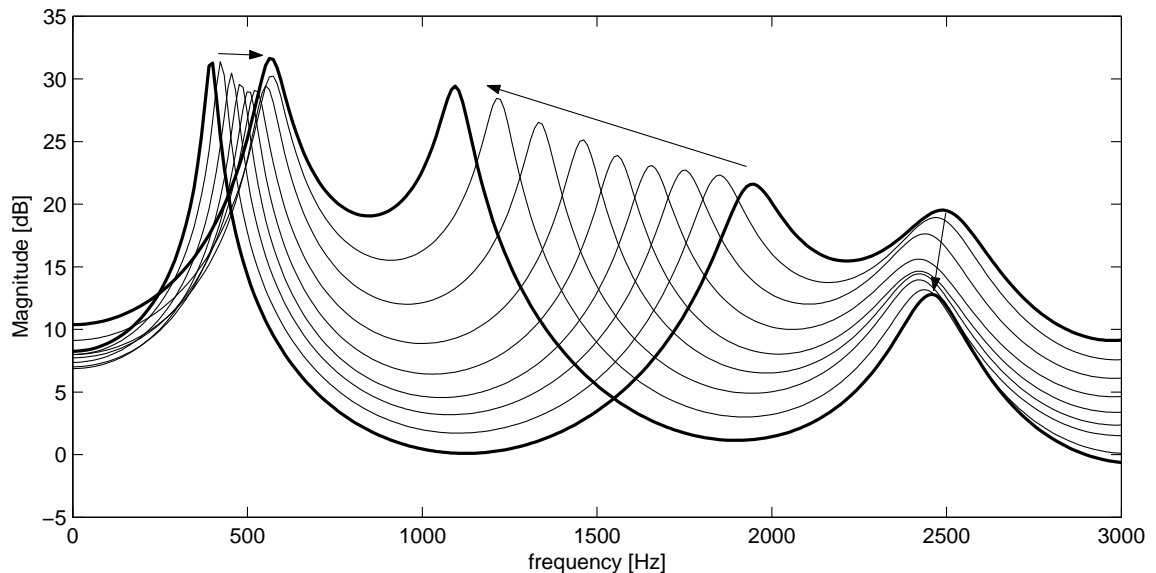


Figure 3.8: Formant movements in smoothed concatenations. The example spectra were obtained by interpolation of a splice of $[\varepsilon]$ to $[a]$. The spectra are the same as in Figure 3.7. The thick lines show the spectra of the two endpoints of the interpolation.

that the LP modelling by itself cannot account for the difference in phase scattering and amplitude differences in the residual on either side of the join. Section 3.4.1 presents some toy experiments on nonsense words synthesised from phonemes recorded in isolation.

3.3.2.4 Waveform Synthesis

At this stage of synthesis, each synthesis time-instant has its own LP parameters and residual, which are either simple copies of those at the associated analysis time or interpolated versions. The residual $y(n)$ is produced using non-overlapping frames as given by Equation 3.3, with the synthesis window $h_k(n)$ a square window centred on zero such that

$$-\lfloor(\eta_k - \eta_{k+1})/2\rfloor + 1 \leq n \leq \lfloor(\eta_{k+1} - \eta_k)/2\rfloor, \quad (3.16)$$

where η_k is the index in $y(n)$ of the centre of the synthesis frame.

Filtering the residual signal with a time-varying all-pole filter with coefficients \mathbf{a}_k results in the desired synthesised waveform $s(n)$. Where the frame is part of a smoothed region, a short-time speech waveform with the modified spectral envelope is produced. Care must be taken with filter transient responses. The filter input signal is $y(n)$, the PSOLA processed residual signal. It is filtered through the all-pole filter $H_k(z)$ using a simple implementation of the difference equation

$$s(n) = y(n) - \sum_{p=1}^d a_p s(n-p) \quad (3.17)$$

When the index n crosses the boundary of a frame, the filter parameters \mathbf{a}_k are updated with the associated filter parameters for that frame, leaving the filter memory intact. The progression through time of the envelope given by the filter parameters has been smoothed by the large overlap between analysis windows. Thus the disturbance caused by this additional filter input is not audible. (See Section 3.4.3.)

The simple splicing of LP residual signals might seem naive at first, with the potential to cause buzziness. Note however that the LP inverse filtering whitens the spectrum, decorrelating the samples. Therefore the frame-level concatenation simply splices sequences of decorrelated samples with similar variance. The join is therefore perfectly natural for the type of signal.

The effect of pre-emphasis is finally removed by filtering the waveform through

$$H(z) = \frac{1}{1 - \alpha z^{-1}}. \quad (3.18)$$

3.4 Evaluation

Attempts to test the ability of the speech modification system are frustrated by the lack of good targets to modify concatenated speech to. The result of setting any target inevitably only allows the result to be as good as the target that has been set.

To show the speech modification component in isolation, three experiments were constructed. The first (Section 3.4.1) concatenates vowels recorded in isolation and shows the operation of the spectral smoothing method. The next two (Section 3.4.2) sets the pitch target by simply smoothing the pitch contour, and by manually flattening some unnatural pitch excursions. The accompanying CD-ROM contains the audio of these examples, and more.

3.4.1 Isolated Spectral Smoothing Experiments

Figure 3.9 depicts the concatenation with and without smoothing of three phones of the nonsense word [maɛ], as well as a naturally spoken version. Clear discontinuities can be seen in the simply concatenated version. The formants in the recorded utterance clearly move at the vowel boundary, which is mimicked closely in the smoothed version of the nonsense word. Perceptually, the smoothing also results in a convincing transition. Concatenating disparate phonemes is an admittedly artificial situation, but it shows the ability to smooth formant movements well. The result is supported subjectively. This type of smoothing is an enabling factor when extending the unit-selection synthesiser to general TTS.

3.4.2 Making Pitch Targets

Two experiments show that the pitch modification produces very good results in the limited-domain synthesiser, improving naturalness in most cases. It also shows the need

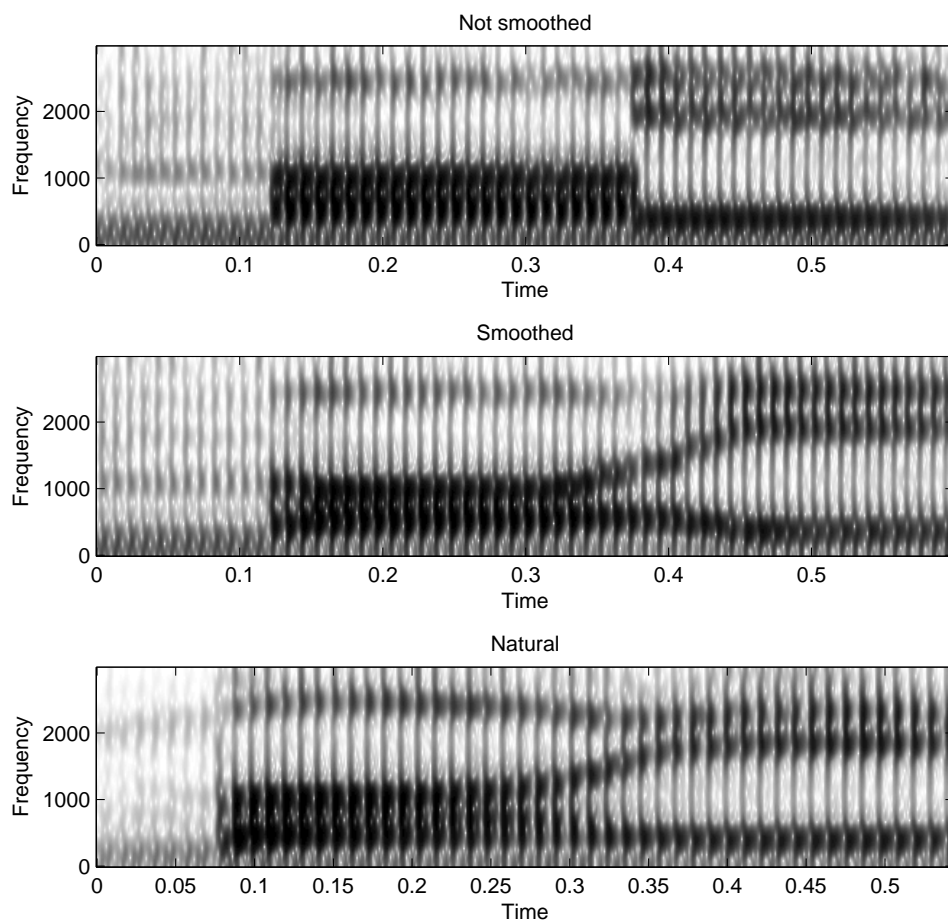


Figure 3.9: Spectrograms of the nonsense word [maε]: unsmoothed, smoothed and natural versions. The smoothed version (middle panel) resembles the natural version (bottom panel).

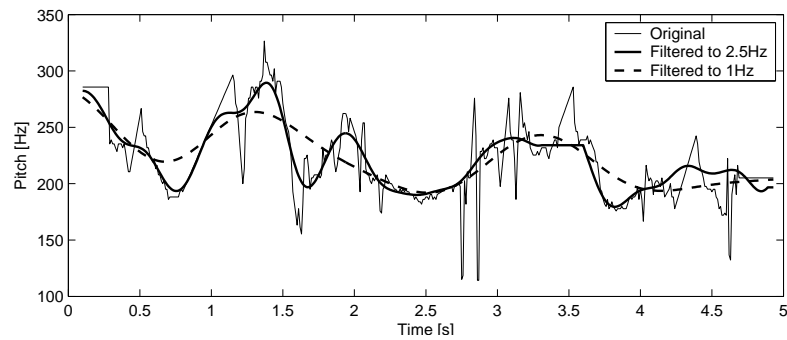
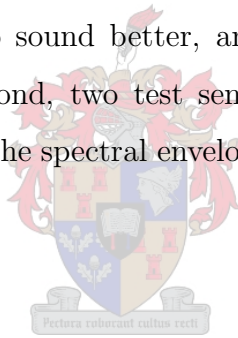


Figure 3.10: The pitch contour of the utterance: “I’m sorry, we only have eleven double rooms, and seventy-nine single rooms in that period.” Two smoothed target pitch contours are shown. One was low-pass filtered to 2.5Hz, and the high pitch excursion, at about 3.5s, in the word “nine” has been flattened manually. The other was low-pass filtered to 1Hz in order to produce a more exaggerated effect.

for prosody modelling if the synthesiser is to be extended in the direction of general TTS. The first experiment adds a simple pitch smoothing operation to the concatenative synthesiser. Some examples do sound better, and on a whole the operation does not lower voice quality. In the second, two test sentences were identified in which severe discontinuity occurred, both in the spectral envelope and in pitch. The pitch contour was edited manually.



3.4.2.1 Pitch Smoothing

Figure 3.10 shows the pitch contour for the utterance: “I’m sorry, we only have eleven double rooms, and seventy-nine single rooms in that period.” The pitch contour was linearly interpolated through unvoiced regions, connecting the ends of voiced pitch sections. The resulting contour was resampled at 100Hz, and low-pass, zero-phase filtered to 2.5Hz. Many of the sharp high-pith or low-pitch sections occur at the edges of voiced regions, and thus the pitch modification does not operate there.

Generally speaking, this reduced the over-dramatisation in the English HRS recordings. The result from the pitch modification sounded very convincing; informal listening tests involving four non-expert listeners and one speech recognition expert listener showed that listeners did not have a consistent preference for unsmoothed or smoothed utterances. This is so in spite of clearly audible differences in the intonation of the utterances. Listeners also said that the voice sounds more mature and professional in the smoothed versions;

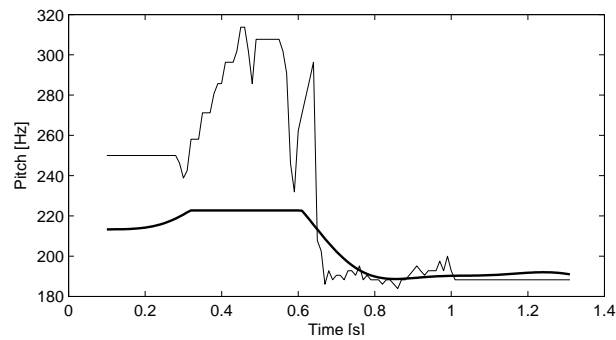


Figure 3.11: A smoothed pitch contour of the utterance: “Now for the rooms.” from Section 2.6.5.3. The very high pitch excursion at the start of the utterance has been flattened manually, and the pitch range has been compressed.

critique on voice quality was never given.

Smoothing the pitch contour even more by setting the low-pass filter’s cut-off frequency to less than 2Hz resulted in the utterances sounding sleepy. There are also concerns about deformed pitch accents in Xhosa.⁴

3.4.2.2 Manual Pitch Correction

Figure 3.11 show the pitch contour of the utterance: “Now for the rooms.” from Section 2.6.5.3. The abrupt rise in pitch at the start has been clipped. Following that, the pitch contour range for the entire utterance was compressed by subtracting the mean value, dividing the result by a factor of 1.5, and adding the mean again. The waveforms are available on the accompanying CD-ROM.

Figure 3.12 shows the spectrogram of the utterance around the concatenation region between “the” and “rooms”. Note the gently smeared formants and smoother amplitude transition around the join location.

The utterance now sounds more subdued, and the concatenation is less disturbing. The word “Now” was selected from a different utterance whose entire pitch contour is higher. The word “Now” still sounds shrill however. That may be attributed to differences in loudness, the effect of the loudness on the spectral envelope and the possibility that

⁴Here we are taking the view of Roux et al. [80] that a strong case may be made for Xhosa to be a pitch accent language.

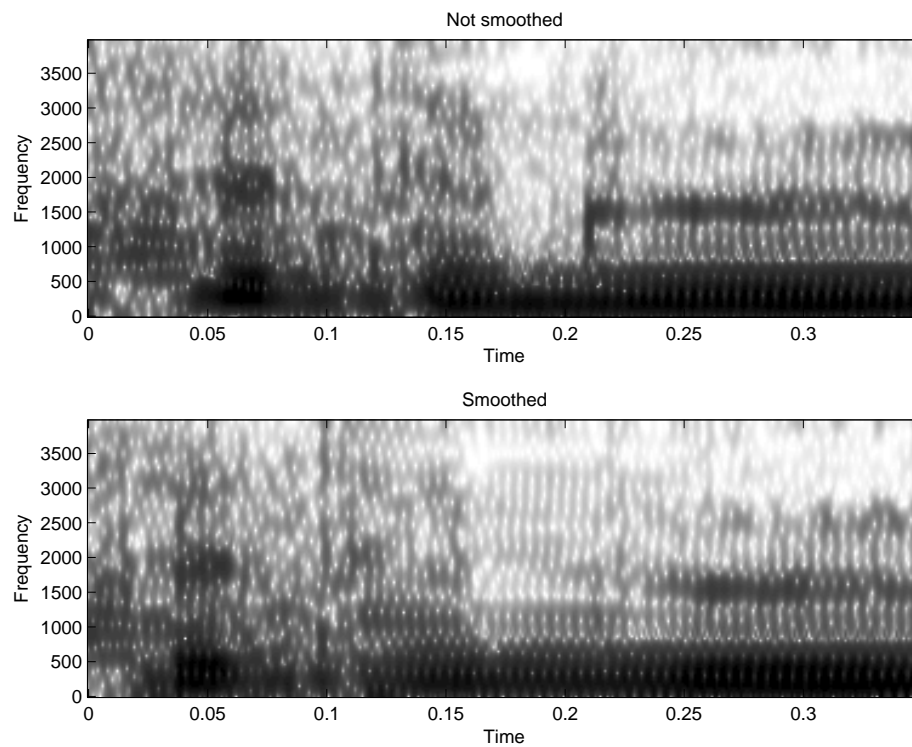


Figure 3.12: Detail of the spectral envelope around the concatenation point between “the” and “rooms” (at 0.21s): the top panel is the spectrogram from Section 2.6.5.3 and the bottom panel shows the smoothed version.

the speaker was closer to the microphone than when recording the utterance from which “rooms” was selected.

3.4.3 Pitch-mark Location

Figure 3.13 illustrates the effect of the location of the analysis pitch-marks on the resulting synthesis. The pitch tracking algorithm described in Chapter 4 favours synchronising on the highest peaks in the pitch cycle. The two sets of pitch-marks were obtained by running the pitch tracker on the original waveform, and on the inverted waveform obtained by multiplying by -1 . Since the peaks that seem to correspond with the location of impulsive excitation of the vocal tract extend downward, the best pitch-marks for this sample are extracted from the inverted waveform. This is not always the case for this particular speaker. Note that the same pitch periods are extracted, but they are out of phase.

The distortion of the waveform in the lower panel in Figure 3.13 is clearly audible. Examination of the PSOLA-processed LP residual signal in Figure 3.14 reveals an atten-

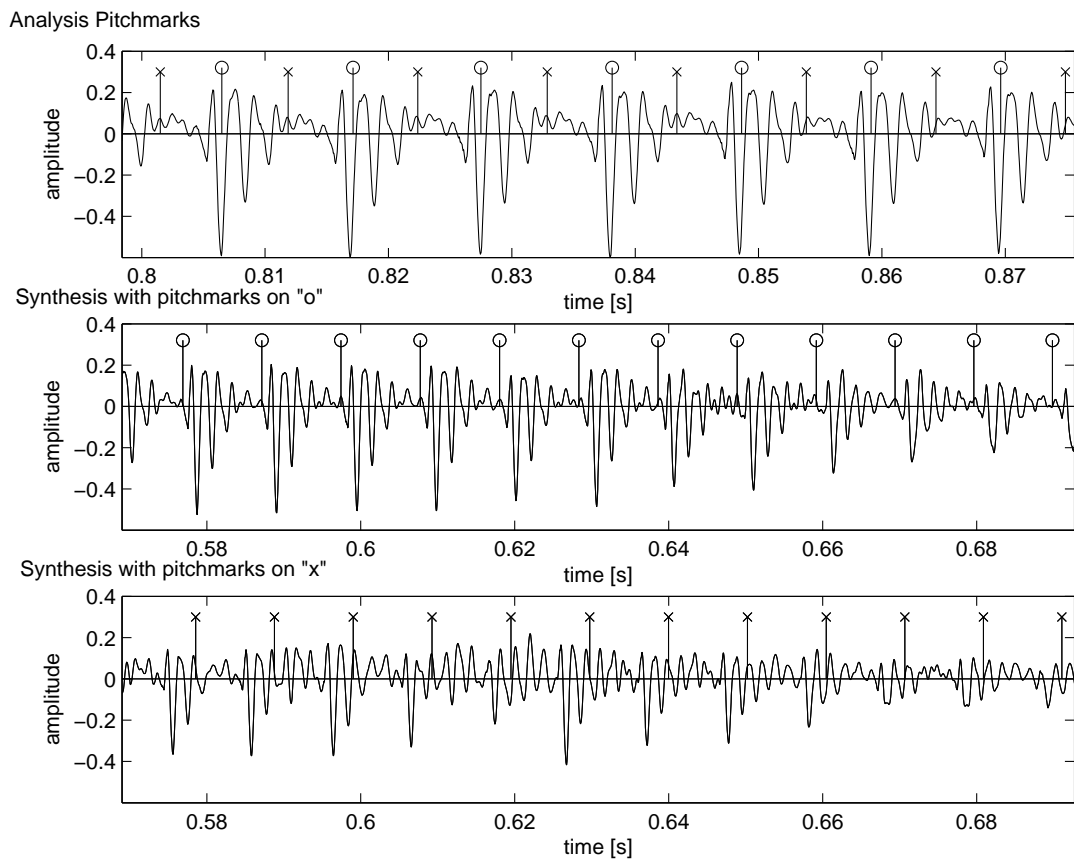


Figure 3.13: The effect of pitch-mark location on synthesis. The top panel shows the original waveform with the two sets of pitch-marks. The two bottom panels show synthesised signals with each set of pitch-marks. The bottom panel clearly does not resemble the original waveform as well as the middle panel. It exhibits clear time-varying distortion of the pitch pulses.

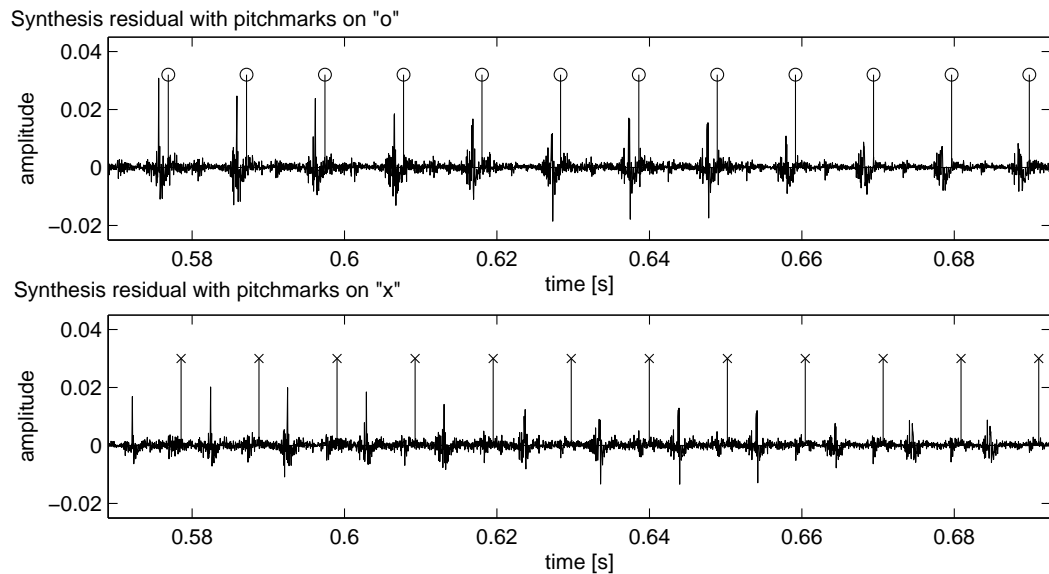


Figure 3.14: The effect of pitch-mark location on the LP residual. The bottom panel shows the residual signal that resulted in the distorted version from Figure 3.13. Note the deformed peaks. The filter parameters are updated halfway between synthesis time-instants, in the region of the higher amplitude portions of the pitch-cycle.

uation in the peaks of the residual signal. This is due to the Hamming windowing of the residual frames during analysis. The peak in the pitch cycle in the residual signal should fall in the centre of the window. The square windowing for the non-overlapping synthesis operation also affects some peaks.

Another hypothesis is that updating the filter parameters during a high amplitude portion of the pitch cycle is not desirable. This point is also raised by Rank [79].

Possible solutions and enhancements to the pitch determination algorithm are discussed at the end of Chapter 4.

3.5 Conclusions & Suggestions

Many comparisons exist between novel speech modification methods and TD PSOLA. It should be noted that they mostly compare two methods on a diphone synthesis problem. Few specify the windows they used to obtain the analysis short-time signals, or the interpolation they used to obtain the new synthesis pitch-marks or synthesis short-time signals. It is therefore difficult to judge the objectivity of such reports. Pitch transplanta-

tion experiments with varying requirements on the range of pitch or duration adjustment, would be more appropriate.

Much criticism has been levelled at LP-based approaches: they often give a quality of vocoded speech. The major cause for this is that the residual is often processed to reduce its storage requirements. In this work, the residual is stored directly, resulting in perfect resynthesis if no modification is performed. Apart from increased storage requirements, this approach is computationally very efficient.⁵ A general synthesiser's database might contain on the order of thousands of prompts, suggesting that the storage problem should be addressed.

The LP PSOLA approach (with no overlap) we have taken here, has been shown to provide very high quality speech modification in the context of limited-domain concatenative synthesis. None of the excellent voice quality has been lost, and it has shown the ability to overcome the limited-domain synthesiser's most pertinent data shortage problems: the prosodic and spectral discontinuities. Accurate global pitch specification will complete this capability.

Listeners commented that the voice sounds more mature and professional in the smoothed versions. No critique on voice quality was ever given, suggesting that the LP PSOLA approach delivers excellent results for the requirement of adapting pitch in this setting.

This work is only the beginning of building such a system, however. Some work needs to be added to build a fully automatic system.

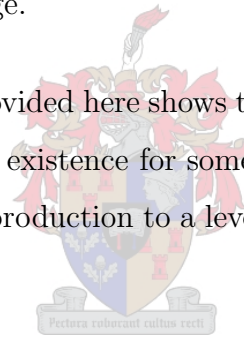
- In some cases the phoneme labels were spaced very close together. It occurred when speech was de-emphasised, and was thus spoken very fast and not articulated clearly. Sometimes phonemes were smoothed away completely because they fall inside the smoothing region. As remedy, the smoothing system needs more phonemic awareness.
- Knowledge about average durations of phonemes in particular words will aid in

⁵The synthesiser performs synthesis from synthesis specification to a waveform at about 20 times real-time on a 400MHz Pentium II-based personal computer. The experiments were all done at a sampling rate of 16kHz.

setting adaptive phonetically aware smoothing time-constants. Setting good duration targets for phonemes, followed by time-scale modification in the LP PSOLA framework, will further improve the robustness to labelling variation.

- Modification of loudness is easy in this framework. The next step is to obtain targets for modification of loudness.
- The spectral envelope should also be modified in conjunction with the pitch, loudness and speaking rate changes.
- An interesting approach to pursue is fusion units [77]. Fusion units explicitly encode the trajectories that the envelope parameters should take in a particular phonetic context. The difficulty here is that the fusion units must also be in the database. This makes it an ideal approach to attempt if the limited-domain corpora are extended to diphone coverage.

The modification framework provided here shows that speech modification capable of very high quality output has been in existence for some time. The major shortcoming of TTS systems is modelling of speech production to a level where good targets can be set for the waveform synthesis step.

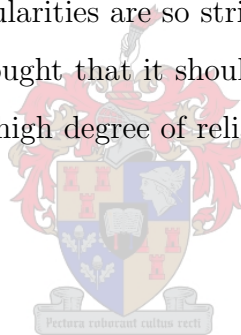


Chapter 4

Pitch Determination

“One of the most conspicuous features of the normal speech signal is the regularly occurring periodicity at its so-called voiced parts. When first encountered, these clear regularities are so striking that many a young scientist in speech research has thought that it should not be too difficult to measure these periodicities with a high degree of reliability.” — Hermes [81]

4.1 Introduction



Over the years, a vast number of solutions to the problem of pitch determination in speech signals have been proposed. It appears that every body of speech signal processing research includes its own work on the subject. It is almost a right of passage for speech researchers. The age of easy availability of digital computing and digital signal processing made this an even more prolific field.

We present an overview of work on Pitch Determination Algorithms (PDAs). The presentation draws on overviews from [62, 81, 82] to set the stage for a brief discussion on more recent developments. Sections 4.4 and 4.5 explain and evaluate a PDA that simultaneously optimises a selection of pitch-mark locations (time-domain peaks), as well as the selection of pitch frequency candidates from a simple frequency-domain pitch determination method.

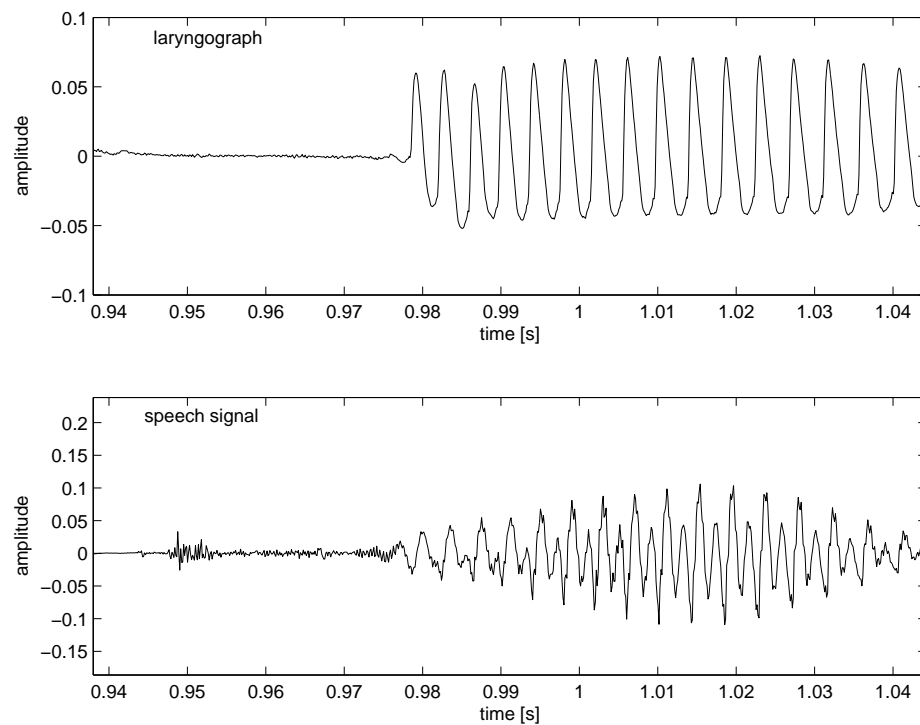


Figure 4.1: A short segment of a speech waveform showing a voiced and an unvoiced phone. The bottom panel shows the speech signal; the top panel shows the accompanying laryngograph measurement. The pitch-pulses caused by the vibration in the vocal folds can be seen very clearly in the voiced part.

4.2 Defining Pitch

Pitch is a quality of a sound that humans perceive. While many people may have trouble discerning quantitatively exactly what the pitch of a voice or instrument is, they generally agree on whether a sound has pitch, and on whether the pitch of two different sources are similar or not. Intonation is almost completely embodied in pitch, hence our interest in measuring and manipulating it.

Pitch is associated with voiced speech, which is produced by regular vibrations in the vocal tract. The effect of these vibrations can be seen very clearly in the speech waveform in Figure 4.1. The sharp peaks follow shortly after the Glottal Closure Instances (GCI). The GCI are those moments in a speech waveform where the vocal folds close, just before they open to release the impulse that excites the vocal tract system. Also note how suddenly the vibration starts and then stops again. This supports the notion that speech can be split into voiced and unvoiced parts.

Pitch has been linked to various properties of signals. The most conspicuous is the

similarity between sections of the waveform, spaced at regular intervals in time. We know from Fourier theory that such a regular spacing results in harmonically spaced peaks in the magnitude spectrum. That these properties are strongly linked to pitch is further supported by research into the working of the human auditory system.

The main consumer of pitch information in this work is the waveform synthesiser. Many such algorithms are pitch-synchronous [61, 18], and even those that are not, also require pitch-pulse locations to ensure phase continuity [71, 67]. These all work on the principle that pitch is contained in the finer harmonic structure of the spectrum or in periodic regions in the voiced waveform.

The properties of voiced speech signals and requirements of downstream signal processing modules indicate that the aim of the PDA should be to

- determine where pitch is present in the signal (the voiced/unvoiced decision),
- measure the pitch where it is present (pitch determination in voiced speech), and
- determine pitch-related time-instants useful to the waveform synthesiser.

A secondary aim is to make PDAs as automatic as possible. Many PDAs have a number of parameters that allow the user to tune the PDA for a specific situation. This makes the PDA unreliable and much more difficult to evaluate and compare performance to others.

4.3 Overview of Pitch Tracking Methods

Most PDAs involve three steps, depicted in Figure 4.2. First is a feature extraction or enhancement step. Feature extraction can be broken down further into pre-processing and transforming to a domain where pitch may be better represented. Pre-processing steps like pre-whitening and pre-emphasis are performed to enhance features that the pitch candidate extraction step can use, and suppress features that might confuse it. In some cases the pre-processor attempts to tag voiced and unvoiced segments in the waveform as well. More recent algorithms employ statistical methods, and from that point of view, pitch candidate extraction could also be thought of as *pitch hypothesis extraction*. Finally, a post-processing step classifies each time-instant as voiced or unvoiced, and in the voiced

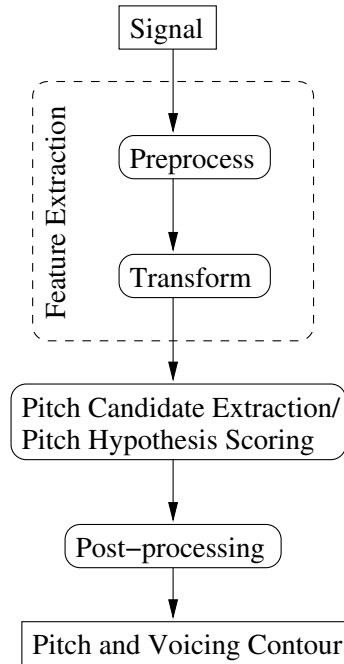


Figure 4.2: General flow diagram of PDAs. Speech signals are sometimes pre-processed to highlight desired features. The feature extraction step represents the signal in a way that is more conducive to finding the pitch. The post-processing step removes some types of errors that stem from the feature extraction step. Finally a pitch contour is obtained.

case selects an optimal pitch. The optimality criterion may be as simple as selecting the highest scoring pitch for each frame, or it may attempt to optimise the pitch candidate selection over multiple frames.

4.3.1 Pitch Feature Extraction

Pitch in a signal manifests itself as similar waveforms repeating in the time domain. The burst in the pitch-pulse bursts are generally wide-band phenomena, due to the discontinuity of the airflow through the glottis. In the frequency domain the quasi-periodic nature of the signal produces clear harmonics in a spectrogram, with the pitch generally considered to correspond with the fundamental frequency. The feature extraction step in the pitch determination process attempts to exploit some or all of these manifestations of voiced excitation.

Pitch feature extraction usually produces a vector of pitch hypotheses for a frame of speech. Each element of the vector represents the likelihood of a corresponding frequency

to be the correct pitch. Other methods extract only a limited number of hypotheses at each analysis instant [83].

4.3.1.1 Overview

The earliest PDAs were implemented in analogue electronics. They mostly consisted of filtering out the first harmonic. For relatively simple speech, without wide variation in pitch, band-pass filters could be tuned to a band which always contained only the fundamental. With the advent of digital electronics, PDAs that pick successive peaks, count zero crossings, calculate autocorrelation and perform a myriad of heuristics became more popular. The increase in computing power of digital devices led to the ability to integrate pitch information across the entire frequency-domain bandwidth.

Time-domain Pitch Feature Extraction

Autocorrelation is still a very popular basis for implementations of pitch trackers [84]. The idea that pitch in speech gives rise to a waveform that is similar to itself at regular intervals in time, leads one to believe that the autocorrelation function should show a peak at the pitch-period. Similarly, since unvoiced sounds resemble stationary coloured noise, the autocorrelation should not show any distinct peaks except at the origin. An estimate of the autocorrelation sequence, $r(n)$, of a speech frame $x(n)$ of length N is given by

$$r(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)x(k+n) \quad (4.1)$$

where $x(n)$ is assumed to be zero outside the range $n \in [0, N-1]$. Figure 4.3 illustrates the result. One of the main drawbacks of autocorrelation is that the first formant resonance may boost higher harmonics. In such cases, the autocorrelation function will have higher peaks at half the correct delay, giving rise to pitch doubling errors.

Autocorrelation is commonly used with spectral whitening by linear prediction to remove the effect of formants before autocorrelation is computed. This constitutes the well-known SIFT algorithm by Markel as described in [1, Section 5.6]. The idea is that the time-varying formants and loudness change the waveform so that periods are less similar. Therefore removing such effects through inverse filtering should improve the results.

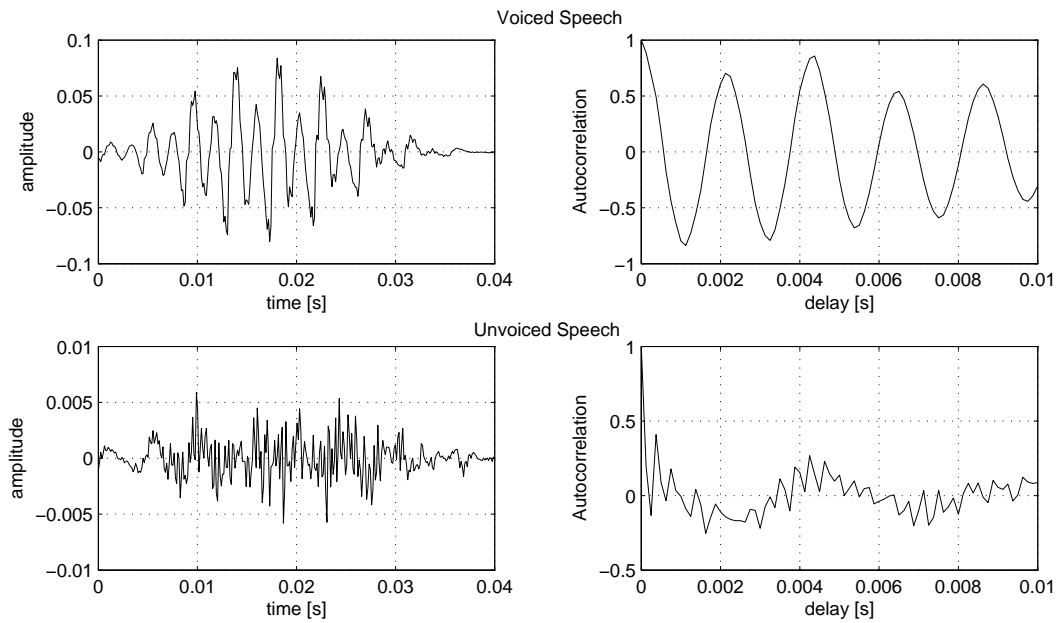


Figure 4.3: Autocorrelation of a voiced and an unvoiced segment of speech. The top panel shows a voiced segment of speech and its autocorrelation, The bottom panel shows that of an unvoiced segment of speech. Both plots have been normalised by dividing by the autocorrelation at zero delay. Note that the value at the maximum of the normalised autocorrelation gives a measure of similarity at a delay, and thus a voiced/unvoiced threshold can be set on this value.

Other popular pre-processing approaches for autocorrelation analysis include taking the exponent of the signal while preserving sign, and centre clipping. Both these non-linear operations introduce or reinforce harmonics of the fundamental and highlights the higher amplitude parts of the pitch cycle.

Autocorrelation allows for a consistent normalisation through normalising its value at zero lag, or equivalently, normalising the frame power. This property simplifies the setting of thresholds on the autocorrelation value to build a voiced/unvoiced classifier.

A similar idea is the Amplitude Magnitude Difference Function (AMDF), defined by

$$\psi(n) = \frac{1}{N} \sum_{k=0}^{N-1} |x(k) - x(k+n)|. \quad (4.2)$$

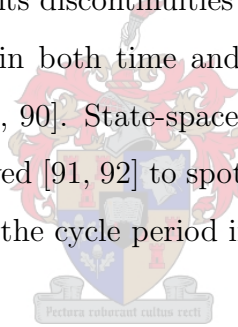
It can be seen that at values for n where the signal is similar to itself, $\psi(n)$ has small values [1, 62]. The AMDF can be shown to be similar to the autocorrelation while being cheaper to compute, since it does not involve multiplication.

Many variations on the basic theme of similarity in the time domain exist. The best known is probably the Super-Resolution Pitch Determinator (SRPD) [85] and the

Enhanced Super-Resolution Pitch Determinator (eSRPD) of Bagshaw [86]. It works by maximising the correlation between adjacent, non-overlapping regions in the signal, and setting a multitude of interdependent thresholds to simultaneously determine voicing and pitch.

A significant problem in pitch determination is coping with various types of noise. Shimamura and Kobayashi [87] deduced that autocorrelation and the AMDF respond independently to additive noise. They exploited this by weighting the value of the autocorrelation function at each lag value with the value of the AMDF at the same lag. This is claimed to yield a more robust measure with the efficiency of autocorrelation and the AMDF.

As new signal processing techniques proliferate into the mainstream of signal processing, they too are applied to pitch determination. The Discrete Wavelet Transform (DWT) has the property that it highlights discontinuities in signals and their derivatives, and allows the localisation of events in both time and frequency. A number of attempts to exploit this has surfaced [88, 89, 90]. State-space embedding techniques from non-linear dynamics have also been employed [91, 92] to spot deterministic and stochastic regions in the signal, as well as determine the cycle period in the deterministic regions.



Frequency-domain Pitch Feature Extraction

The frequency domain provides a rich space from which to distill features. An old and very direct method to locate the pitch in the frequency domain, is to pick the fundamental peak in the short-time discrete Fourier transform. The frame must be long enough for the harmonics to be discriminated. The frames could also be zero-padded to interpolate the spectrum for more accurate peak-picking.

In voiced speech signals, especially those with low pitch, the fundamental frequency often has a lower amplitude than higher harmonics. This makes such a naive method prone to miss the fundamental altogether and make pitch doubling errors. This illustrates the necessity of integrating information across the spectrum when performing pitch feature extraction in the frequency domain.

A more refined method is either the Harmonic Sum Spectrum (HSS) or the Harmonic Product Spectrum (HPS), where regions of a processed FFT are combined in a

way that reinforces harmonics [81]. The HSS and the HPS are forerunners of the harmonicogram [93]. Incorporating information from higher harmonics allows such a pitch feature extractor to find the correct pitch even though the fundamental may be completely missing. Frequency-domain PDAs gain this advantage from the fact that they can use information about harmonics over the entire spectrum.

Seneff [94] developed one of the first published methods that explicitly use peaks in the magnitude spectra of speech frames to derive pitch. The method relies on the observation that voiced regions of speech show clear harmonics in frequencies under about 1kHz. The fact that harmonics are all F_0 Hz apart in the spectrum is exploited by a heuristic that first throws out peaks it judges to be spurious, and then accumulates counts of distances between peaks. The peak in the resulting histogram of distances is then identified as the pitch. Much of the robustness of the method stems from its ability to discern between spurious and true peaks in the spectrogram. It suffers from the fundamental shortcoming of having many parameters that customise performance for a specific speaker.

The best-known method of measuring pitch in the frequency domain is based on the cepstrum. The method is originally by Noll [1, Chapter 6]. It exploits the ability of the real cepstrum to separate the periodic excitation and the effect of the vocal tract on the periodic excitation. The development by Deller et al. [1, Chapter 6] shows that one can expect the effect of the vocal tract on the cepstrum to decay quickly along the cepstral “time”-axis, and the excitation to show up in periodic pulses at the pitch delay. Pitch feature extraction using the cepstrum relies on the assumption that the window used to compute the cepstrum is at least as long as a small number of pitch-periods¹.

Using Features from Sinusoidal Analysis

The investigation of sinusoidal methods for speech coding and synthesis leads to a new derived set of features to describe signals. Analysis of sinusoidal parameters under the assumption that they are harmonically related, is a popular and effective way to encode and modify speech. The harmonic assumption gives an accurate requirement to test pitch and voicing hypotheses for a particular frame of speech. This section highlights some of

¹In practise implementations use windows that are three to four times the length of the period of the lowest pitch that they expect to see.

the efforts in the literature to exploit these facts in the context of sinusoidal coding.

McAulay and Quatieri [95] used a method to minimise the mean squared error that a re-synthesised signal would have with respect to F_0 if the sinusoidal components of the signal were assumed to be harmonic. The criterion that has to be optimised can be shown to be free of spurious peaks at sub-harmonics of the true pitch of the speech frame. To perform a Voiced/Unvoiced (V/U) decision, they set a threshold on the ratio of the energy of the original signal, and the energy of the difference between the harmonically reconstructed signal and the original.

Another way of computing the fit of an observed spectrum to a harmonic hypothesis of fundamental frequency F_0 is the spectral comb. It amounts to nothing more than maximising over F_0 the correlation between the magnitude spectrum of a frame of speech and a frequency-domain comb function with components at $\{F_0, 2F_0, 3F_0, \dots\}$ convolved with the frequency response of the window function.

An efficient algorithm to derive the pitch based on the spectral comb was proposed by Chazan [96]. First, peaks are picked out from the magnitude spectrum. The algorithm heuristically throws out spurious ones. It proceeds to add values to a histogram of pitch hypotheses. A strong sinusoid observed at frequency f could be a component of a periodic signal with fundamental f/n for any integral $n > 0$. To paraphrase, all pitch hypotheses that could give rise to a sinusoid at the observed frequency are reinforced. Again, clever heuristics allow the algorithm to ignore most of the peaks, speeding it up. This method suffers from the same propensity to pick sub-harmonics of the pitch as does HPS and HSS.

Several methods find sinusoidal components in the signal using the notion of instantaneous frequency, computed from the phase spectrum [68, 97, 98]. These have also been employed more directly in finding the pitch.

Auditory-Based Pitch Tracking

Biologically inspired PDAs have been improving steadily over the past two decades, following improving understanding of the human auditory system and the increase in available computing power. The HSS, HPS and spectral comb type methods agree with some of auditory theory, a reason often cited for their strength [81].

Newer methods explicitly operate on the hypothesis that the human auditory system perceives pitch by first decomposing the incoming waveform into narrow-band spectral components. Each spectral component is further processed essentially in the time domain, after which the information across the spectral components is integrated to split multiple overlapping pitches. AMPEX [99] is a famous PDA that operates in this way, although it only tracks a single pitch, and incorporates a speech-specific decision on whether the signal contains pitch. Other examples include [100] and [101]. Coupled with sophisticated post-processing methods, these time-frequency methods show promise in tracking multiple additive periodicities [83]. Such systems offer the hope of being the fourth generation of PDAs in the classification scheme of Hermes [81].

4.3.2 Post-processing of Pitch Features

The next step in a PDA is post-processing or pitch optimisation. Pitch estimators almost always work with short frames of speech. Post-processing in PDAs imposes constraints on consecutive estimates from the pitch estimator, and thus attempts to remove common local errors that the pitch estimator might make.

The most common constraint imposed follows from the observation that pitch normally does not change much over the space of time between frames. Realisations of this constraint range from simple median filtering of the pitch contour to constructing sophisticated statistical models to enable tracking multiple tones.

The eSRPD algorithm performs post-processing by extensive heuristic thresholding and the non-linear smoothing technique by Rabiner et al. [102].

Dynamic Programming (DP) is a useful idea in speech processing in general, as well as in pitch post-processing [103, 104, 105, 106, 62, 88]. The algorithm can be implemented to run on pitch hypotheses in the frequency domain [62], or in the time domain [88, 103]. Some algorithms, like that of Entropic as implemented in Snack [107], enforce pitch continuity through only evaluating the preceding five frames through the Dynamic Programming (DP) procedure. DP is appropriate and highly effective because it yields a globally optimised result.

Bagshaw [108] stylises the pitch contour through linear and non-linear operations to

make it more understandable when presented visually and to make it more readily usable by automatic prosodic transcription tools.

4.3.2.1 Statistical Post-processing

As pitch feature extraction techniques improve and applications move to more difficult problems like multi-pitch extraction, more sophisticated post-processing methods become important.

Wu, Wang and Brown [83] propose a very promising scheme capable of tracking pitches of two periodic signals added together. An HMM is set up without training. This model derives a set of pitch hypotheses from frames of speech using an auditory motivated technique to form the observation nodes in the HMM. These are then combined probabilistically to form the observation probabilities given each state. Each of the hidden states represent a pitch hypothesis. The state transition probabilities represent the pitch dynamics from each speech frame to the next. Wu, Wang and Brown have found that the transition between pitch states on continuous pitch tracks can be accurately described by a sampled Laplacian distribution. The parameters of the Laplacian can easily be estimated from existing pitch tracks. To form the decision between zero, one or two periodic phenomena being present in the signal, they introduce further states and variability to how the observation probabilities are computed. The final voicing decision and pitch tracks are computed using the Viterbi algorithm's optimal path.

An older approach is that of Gu and Van Bokhoven [100]. They constructed a much simpler three-state HMM for enforcing the continuity constraint of pitch-tracks. Their HMM had one state to describe the case where the quantised pitch stays constant, one for increasing and one for decreasing pitch. The output PDFs describe the pitch value and the pitch change from the previous frame. Results from this type of modelling was only tested on waveforms constructed from one utterance by a male, and one by a female. Two HMMs were then trained on male and female speech and used to evaluate the dual pitch extractor.

These HMM approaches exploit the continuity property of pitch and optimise globally through the Viterbi search.

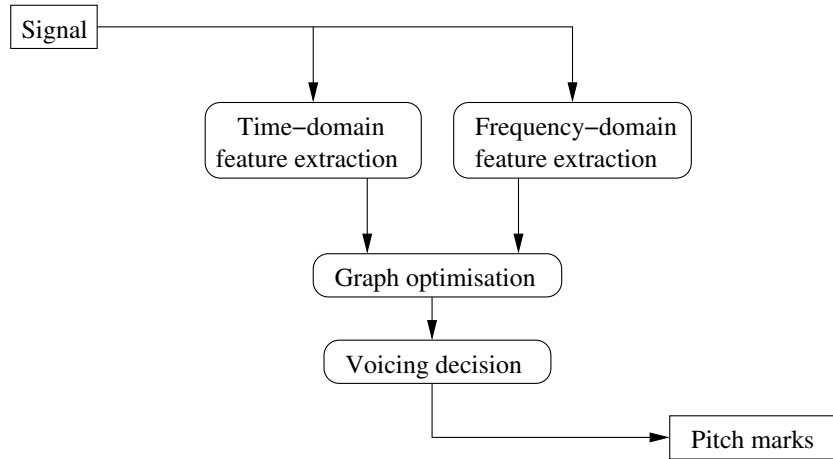


Figure 4.4: The Directed Acyclic Graph Pitch-tracking algorithm.

4.4 Implementation

Several pitch modification methods operate pitch-synchronously, and they need to know the locations of the pitch-pulses. It is sufficient to know the location of the maximum energy in each pitch cycle. More importantly, they need precise locations for pitch-pulses at the onset and end of voicing. The estimates that frequency-domain methods yield are imprecise here because they operate on an entire frame of speech, and the onset of voicing is a very dynamic region in the speech signal.

Inspired by the ability of dynamic programming algorithms to find good pitch candidates in the frequency domain, and applications where good time-domain locations for pitch-pulses are found [103, 88], a pitch tracker was developed that gives good locations of pitch-pulses according to time-domain energy fluctuations and frequency-domain information.

Figure 4.4 outlines the proposed algorithm. The feature extraction step produces two sets of features: in the time domain it band-pass filters the signal and finds energy peaks; in the frequency domain it computes the HPS. The post-processing algorithm incorporates these two sources of information into a Directed Acyclic Graph (DAG). The best path through the graph selects good peaks in the time-domain waveform. A further heuristic step (using autocorrelation) removes spurious peaks that slipped through the optimisation process to constitute a voicing decision.

The algorithm's input consists of a pitch search range (given by the lowest (f_{low}) and

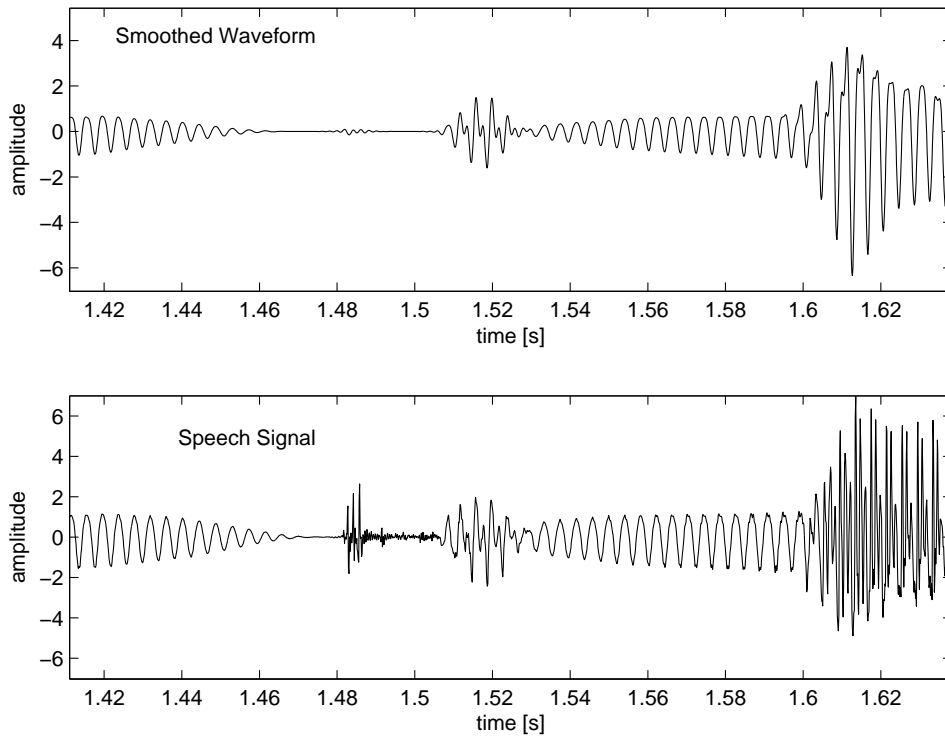


Figure 4.5: The band-pass filtered energy contour from which the time-domain peaks are extracted, and the speech signal.

the highest (f_{high}) expected pitch frequencies in Hz) and an input signal sampled at f_s Hz.

4.4.1 Pitch Feature Extraction

The time-domain feature extraction has to provide good locations for placing pitch-marks. Peaks in the energy contour of the signal provide good candidates for processing algorithms like PSOLA [61].

To this end, the waveform is band-pass filtered to between 40Hz and 1kHz to remove high-frequency noise components from the signal, squared and then band-pass filtered again to the pitch search range to obtain a smooth energy contour. The original sign of the sample values are restored after squaring. All filtering is performed in a zero-phase manner to preserve the location of the peaks in time. Figure 4.5 shows the original speech waveform along with its smoothed energy contour. The positive peaks are selected and their locations in time and the square roots of their heights are stored. The square root compression was introduced to alleviate a problem with some valid peaks being much lower than their neighbours during the post-processing procedure.

Other possibilities exist for extracting the time-domain peaks from the waveform. Specifically, since LP PSOLA as described in Chapter 3 is the principal user of pitch-marks, it could be argued that the LP residual is a better waveform from which to extract the time-domain peaks. For some speakers this gives very consistent results, but for the speaker who delivered the prompts for the Afrikaans and English HRS systems, it failed all too often.

The frequency-domain feature extraction computes the HPS based on the description by Bagshaw [108]. Figure 4.6 illustrates the process for a frame of voiced speech. The logarithm of the discrete power spectrum, $P(l)$, is computed from the DFT of a 30ms, Hamming-windowed speech frame. The frame is zero-padded to yield a DFT with a length of L_P , chosen so that the frequency resolution is at least 2Hz. Next, $P(l)$ is smoothed by convolving it with a Hamming window of length L_{win} , $h(l)$, centred around $l = 0$. The power spectrum is mirrored around zero to alleviate edge effects: $P(l) = P(-l)$, $l < 0$.

$$P_{smooth}(l) = P(l) * h(l). \quad (4.3)$$

The length of the window L_{win} is about 2.5 times the widest expected separation between harmonics, $(L_P f_{high}/f_s)$, and its length is odd.

$$L_{win} = 2 \left\lfloor \frac{2.5 \times (L_P f_{high}/f_s)}{2} \right\rfloor + 1. \quad (4.4)$$

The smoothed log spectrum $P_{smooth}(l)$ is subtracted from the log spectrum. This yields the log-spectrum equivalent of a high-time filtered cepstrum $P_{exc}(l)$ that contains only excitation information:

$$P_{exc}(l) = P(l) - P_{smooth}(l). \quad (4.5)$$

It should also be clear that the harmonics in harmonic signal frames will be higher than 0, while the noise and leakage energy in between will be lower than zero. This means that the negative logarithms of sub-harmonics will subtract from the HPS, while harmonics of a pitch hypothesis will add to it. The HPS $P_{hps}(l)$ can now be computed using

$$P_{hps}(l) = \exp \left(\sum_{n=1}^{N_H} P_{exc}(nl) \right). \quad (4.6)$$

The compression harmonic factor N_H determines how many harmonics of a pitch hypothesis are multiplied together to form the HPS. In practice a value of 5 is often reported.

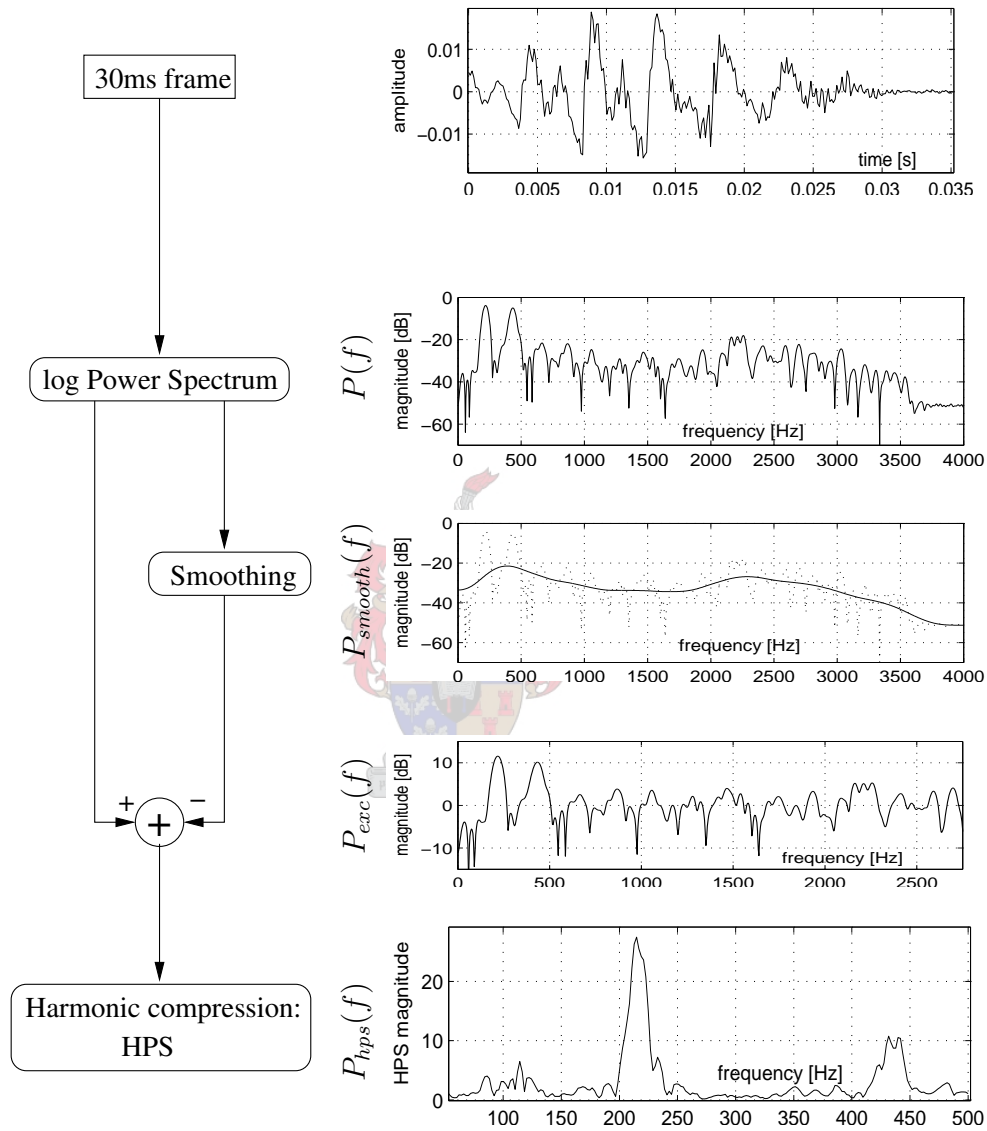


Figure 4.6: Computing the HPS. The logarithm of the power spectrum is computed. Then it is smoothed according to the highest fundamental frequency expected, and the smoothed spectrum is subtracted. The multiplicative harmonic compression is finally applied.

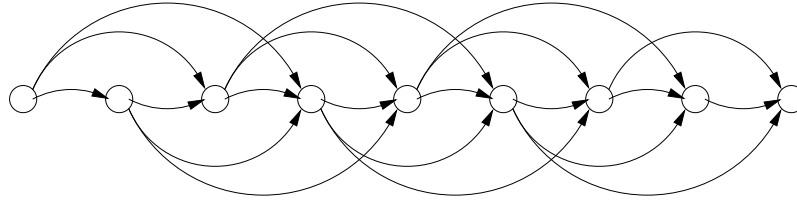


Figure 4.7: Pitch information cast as a weighted graph. Each vertex represents a time-domain peak and weights are assigned to the edges according to how well their spacing agrees with a frequency-domain pitch extraction method.

4.4.2 Post-processing of Pitch Features

The aim of post-processing of the time-domain and frequency-domain features is to select a good subset of time-domain peaks with large amplitudes that are also well spaced according to the frequency-domain information in the HPS in the region. The peaks and how they are related can be viewed as a graph, as depicted in Figure 4.7.² Each vertex represents a peak in the time-domain feature-set. If the weights on the edges are correctly constructed, the optimal path through the graph will go through vertices that represent tall peaks with good temporal spacing, considering the frequency-domain pitch information.



4.4.2.1 Building the DAG

The vertices represent the time-domain peaks. The graph building algorithm takes each vertex in order of increasing time and constructs edges from it to each following peak, up to 50ms forward. The time difference between the peak under consideration and a following peak gives the pitch hypothesis supported by the edge between the vertices that represent them.

The algorithm is illustrated in Figure 4.8, and described in Figure 4.9. Let the N_v vertices be $v_n, n \in 0, 1, 2, \dots, N_v - 1$. The time of v_n is t_n , and its height is h_n . The weight of the edge from v_n to v_m is w_{nm} . The height of the HPS at time t_n for pitch frequency f_{nm} is $P_{hps}(\lfloor L_P f_{nm} / f_s \rfloor)$, where $f_{nm} = 1/[t_m - t_n]$. Note that $P_{hps}(l)$ is the

²The graph terminology follows the conventions in computer science [47]: A vertex is a node in the graph and edges connect vertices. Edges may be directional, i.e. they connect vertices only in one direction, and they may have weights associated with them.

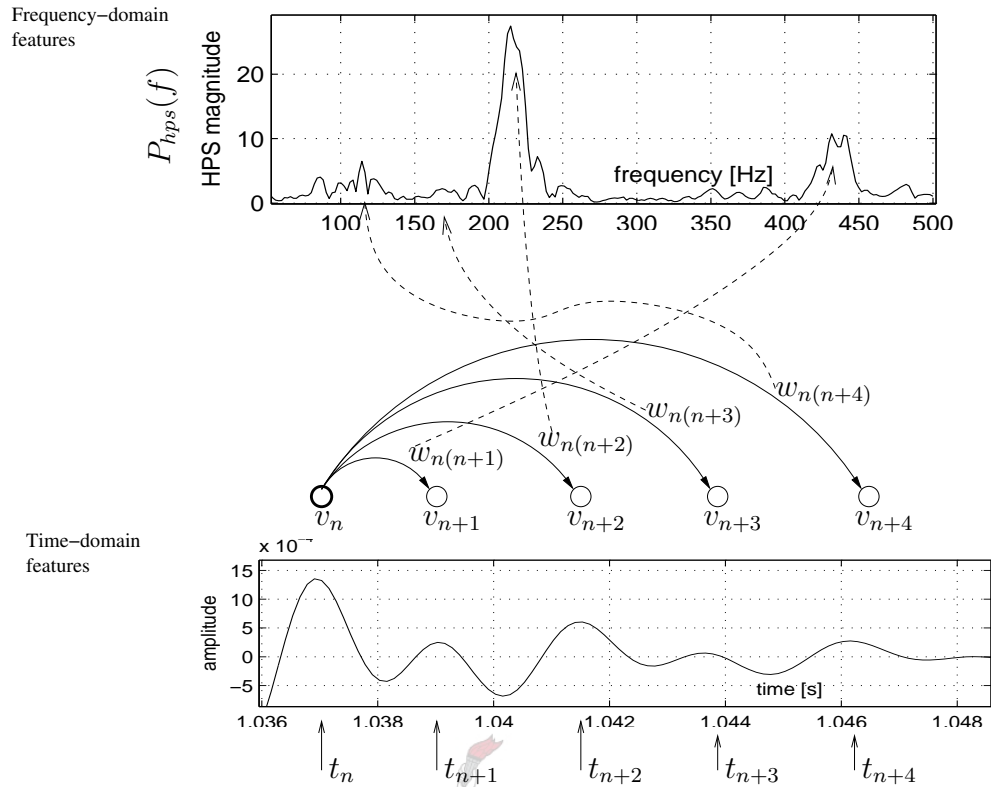


Figure 4.8: Schematic representation of the weight assignment to edges.

highest value of the harmonic product spectra computed from the *three* frames nearest to t_n . This ensures that a good pitch value is found at the onset and stopping of voicing.

Let M_n be the set of integers that index destination vertices v_m from v_n . The weight w_{nm} , $m \in M_n$, is then calculated by

$$w_{nm} = (h_n + h_m) \cdot P_{hps}(\lfloor LPf_{nm}/f_s \rfloor). \quad (4.7)$$

The graph is directed and acyclic since all the departing edges from a vertex go forward in time; none go back. It is therefore impossible that cycles exist. If vertices are in topologically sorted order, the following holds true: for any edge w_{nm} , $n < m$. This holds for the pitch graph as the vertices are sorted in order of increasing time and edges are constructed to point forward in time.

4.4.2.2 Best Path Search

The path through the DAG with the highest cumulative additive weights, gives the desired pitch-mark locations. An efficient greedy algorithm that searches the DAG for the best

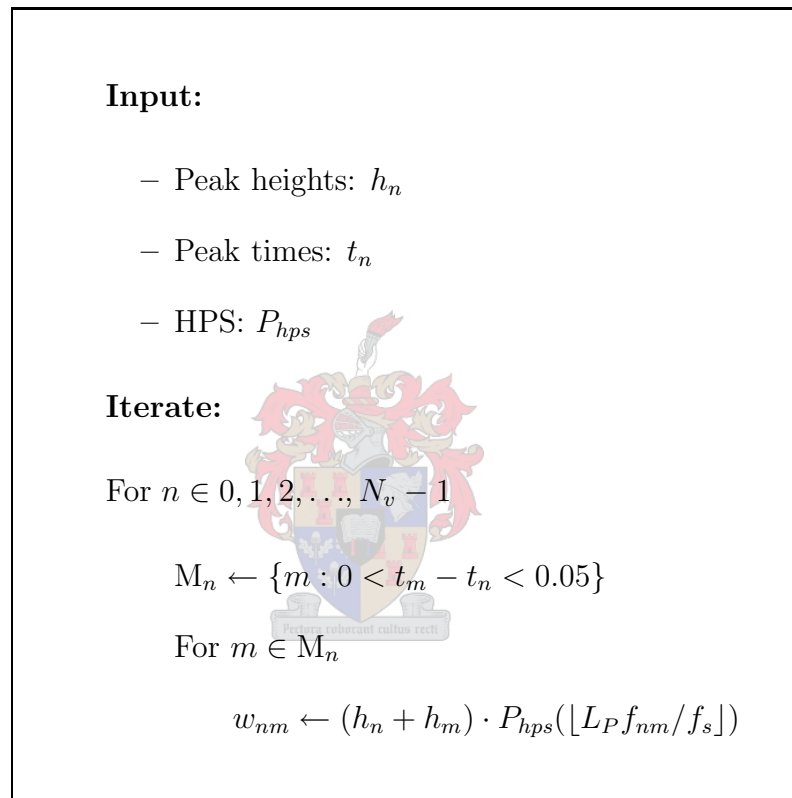


Figure 4.9: The algorithm for connecting the vertices into a DAG. The temporal separation between peaks is used to look up the “goodness” of that time difference in the HPS. The frequency-domain score is then multiplied by the sum of the peak heights connected by the vertex in question.

path is shown in Figure 4.10 [47]. It requires that the graph be directed and acyclic, and that the vertices be visited in topologically sorted order, to yield a globally optimal solution.

The algorithm stores two additional values per vertex.

- The cost of the current best path to each vertex is stored in the array $d(n)$. All $d(n)$ are initialised to $-\infty$.
- The vertex from which the best path was obtained for each vertex v_n is stored in the array $p(n)$.

The algorithm starts with the first vertex v_0 , and applies the following procedure to each vertex in the topological order. At each vertex v_n , it visits all the vertices that follow it through edges. The score of the path through v_n to a following vertex v_m is $d(n) + w_{nm}$. If $d(n) + w_{nm} > d(m)$, then a path going through the current vertex v_n is better than what previous visits to v_m could produce, and the new best path replaces the previous one: $v_m: d(m) \leftarrow d(n) + w_{nm}$ and $p(m) \leftarrow n$.

Finally, the vertices extracted by the best path search can be found by starting at the last vertex v_n , $n = N_v - 1$, and pushing $p(n)$ onto a queue. The process is repeated for each previous vertex $v_{p(n)}$, until the first vertex is reached.

4.4.2.3 Voicing Decision

In regions where the HPS contains strong peaks that agree with the spacing of tall time-domain peaks, the optimal path yields the subset of “good” pitch-mark candidates. In unvoiced and silence regions there will still be peaks, but their spacing will not be as regular and will not correlate as well with the frequency-domain features as in the voiced regions. The selected peaks are sufficiently well-behaved for three simple decision criteria to make adequate voicing decisions.

1. The autocorrelation of short signal frames around two adjacent pitch-marks must be higher than the threshold \mathcal{V}_{ac} , which defaults to 0.4 in the implementation.
2. The energy in the signal at the pitch-mark must also be higher than a threshold, \mathcal{V}_p . The threshold \mathcal{V}_p is set quite low; its purpose is only to exclude silence.

Input:

- Destination index sets for each vertex: $M_n, n \in \{0, 1, 2, \dots, N_v - 1\}$.
- Edge weights $w_{nm}, n \in \{0, 1, 2, \dots, N_v - 1\}, m \in M_n$.

Output:

- The sequence of vertices through which the optimal path goes, Q .

Initialise: $d(0) \leftarrow 0 ; d(n) \leftarrow -\infty, n \in \{1, 2, 3 \dots N_v - 1\}$

Iterate:

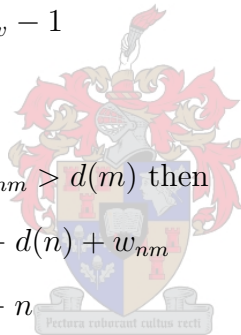
For $n \in 0, 1, 2, \dots, N_v - 1$

 For $m \in M_n$

 if $d(n) + w_{nm} > d(m)$ then

$d(m) \leftarrow d(n) + w_{nm}$

$p(m) \leftarrow n$



Backtrack:

$Q \leftarrow \emptyset$

$n \leftarrow N_v - 1$

while $n \neq 0$

 add $p(n)$ to Q

$n \leftarrow p(n)$

Figure 4.10: The algorithm for searching the DAG. It iterates over the vertices in topological order. At each vertex the score at the destination node of each edge, is updated.

3. Pitch-marks must occur in groups of at least four to be considered voiced speech, and their time spacing may not differ by more than a factor \mathcal{V}_s . The default is $\mathcal{V}_s = 1.25$ in the implementation.

4.5 Experiments

4.5.1 Test Methodology

The profusion of PDAs described in the literature indicates the difficulty of the problem of pitch determination. The large number of options may well be due to the difficulty of comparing performance of PDAs, making the choice a haphazard affair. This section first discusses the efforts of Rabiner [109], where methods of evaluating and comparing PDAs were first reported. Next, three more modern publications are briefly discussed to highlight what they have in common. Lastly, a simplified evaluation framework is defined.

Rabiner et al. [109] published the first and most cited work on evaluation of PDAs. They defined several scores by which to compare PDAs, and also compared seven standard techniques that are still used today. They first derived a ground truth by a semi-automatic procedure³, and then ran all their implementations on the data using the same framing of the signals. The first important score they define is the percentage of gross pitch errors, E_{gross} . This is defined as the percentage of the total number of voiced frames where the PDA measured the pitch incorrectly by more than a set percentage or threshold. According to Stylianou [110], one should aim for a $E_{gross} < 3\%$ error for harmonic re-synthesis to remain acceptable. They also defined E_{fine} , which is the mean and standard deviation of the measurements in all frames that are voiced and where no gross pitch errors were made. Another important measurement is the voiced/unvoiced error, quantified in $E_{u \rightarrow v}$ and $E_{v \rightarrow u}$, defined as the ratio of the number of frames that were misclassified as either voiced or unvoiced and the number of *voiced* frames in the reference data.

Kawahara and De Cheveingé [111] discuss the evaluation of PDAs in general. They also evaluated ten common PDAs, including eSRPD, on data that is readily available.

³Dated 1976, the reference data set was quite small (only about 20 sentences), since the semi-automatic method they used already required 60 hours of computer time.

The goal of their testing was to compare PDAs in terms of how accurately they could find pitch at a specific time in the utterance. They limited the measurements to regions in the waveforms where pitch is unambiguous and did not consider the errors made by PDAs in the regions where pitch is poorly defined. Referring back to Figure 4.1, only regions where the pitch-pulses are regularly spaced in both the speech and laryngograph signals are labelled as voiced for their purposes. This labelling was performed manually, and regions corresponding to regular vibration in the laryngograph were automatically pitch-marked using simple heuristics. The PDAs were scored on the number of frames where they measured the pitch incorrectly. Only frames that are voiced in the ground truth data were taken into account, and where possible, the voiced/unvoiced decision feature of the PDAs were disabled, forcing them to provide a pitch estimate at every time-instant. In this paradigm, PDAs are compared on E_{gross} only. Error rates were often higher than what the tested PDAs' authors claimed, due to sub-optimal parameter variations. This shows the difficulty of using PDAs that have many tunable parameters.

The first problem is therefore to obtain the *ground truth*, i.e. a set of data that is believed to be correct and according to which PDAs can be compared. Rabiner et al. [109] used a semi-automatic labelling procedure. Bagshaw [108] used recordings of speech utterances for which time-synchronised laryngograph data is also freely available. The laryngograph measures the conductance between two electrodes stuck to either side of the speaker's throat. This allows one to deduce a measure of surface area of contact of the vocal folds, and thus the pitch-period. (See Figure 4.1 for an example of a speech waveform with the accompanying laryngograph data.) The relative simplicity of the laryngograph data makes it easy to use heuristics to locate pitch-pulses exactly, and then compute the pitch in a given frame.

Two corpora with laryngograph data were used in the evaluation of the DAG-based pitch tracker. The first data set was released by Bagshaw [86]. It contains 50 sentences spoken by one male and one female speaker, totalling 260 seconds of clean speech.

The Keele Speech Database is also freely available [112]. It includes the laryngograph waveform with ten utterances varying from 25 to 40 seconds each, 5 by male and 5 by female speakers. The data also includes information about whether pitch is present in the speech waveform, the laryngograph, both or none. We use this information to leave the

doubtful regions out of consideration in the calculation of E_{gross} . Some of the speakers croak excessively, and the laryngograph data is quite noisy, making this a difficult set to work with. Comparisons with other PDA implementations are not available for this data set.

The ground truth was derived by running the DAG PDA on the much simpler laryngograph data, as suggested by Plante [112]. This ensured that only regions of the waveform that contained well-behaved pitch-pulses are considered as voiced, due to the second criterion of the voiced/unvoiced decision. The pitch-marks were manually inspected, and the pitch search range was restricted for each speaker to give the best results. The reference thus amounts to data that conforms to the principles of Kawahara and De Cheveingé [111]. The manual inspection of the laryngograph signals and the derived pitch-marks confirmed this.

4.5.1.1 Scoring the DAG PDA

Much of the work on pitch tracking concerns robustness to noise and tracking of perceived pitch. Since the goal in this work is modification and good concatenation locations, the aim is to find regions in the signals that agree with the signal models used for modification. Noise robustness and perceived pitch where the waveform is not vaguely periodic are ignored.

The graph-based PDA will be scored on two different categories:

- **Pitch accuracy** as reflected by E_{gross} . This metric is the same as was used by De Cheveigné and Kawahara [111], and indicates the percentage of voiced frames where the pitch tracker made a gross error. Note that in their study, the PDAs gave a pitch estimate even on frames that contain no tonality. However, only frames that are voiced according to the reference data are taken into account for the scoring process. In this test, the PDA performs a voiced/unvoiced decision, and only regions where the reference and PDA output agree on voicing are taken into account.
- **Voiced/Unvoiced accuracy** as reflected by $E_{v \rightarrow u}$ and $E_{u \rightarrow v}$. The PDA provides a voiced or unvoiced decision. Silence counts as unvoiced speech.

Fine pitch errors, E_{fine} , are disregarded in this work. The PDA being tested puts pitch-marks on the high amplitude part of the pitch cycle, and therefore the pitch can be considered correct if no gross pitch errors exist. Significant errors will be shown by E_{gross} , $E_{v \rightarrow u}$ and $E_{u \rightarrow v}$.

4.5.2 Results

Table 4.1 show the values of E_{gross} , $E_{v \rightarrow u}$ and $E_{u \rightarrow v}$ for the Bagshaw and Keele data sets. The DAG PDA compares favourably on E_{gross} with the best PDAs in the study by Kawahara and De Cheveingé [111] on the Bagshaw set. Note that the DAG PDA was run with a general search range of 50–500Hz, and defaults on its remaining parameters for all the speakers.

The results for $E_{v \rightarrow u}$ and $E_{u \rightarrow v}$ seem less encouraging, but still compares favourably with published results [83, 86], especially considering that in these publications, the ground truth data were manually corrected. Manual inspection further explains the results—many voiced regions are simply not visible in the speech signal, while they are clear in the laryngograph. These were found to make up a large proportion of $E_{v \rightarrow u}$. Unvoiced speech was most often misclassified at the edges of voiced regions where impulsive unvoiced sound occurred. These gave rise to peaks in the time-domain features, and were close enough to harmonic regions to fool the frequency-domain feature extraction.

The results on the Keele set is less encouraging. Manual inspection revealed numerous cases where the voice is very croaky and quite irregular, or the laryngograph is corrupted beyond salvage. The provided labelling of dubious regions in the speech and laryngograph brings the voiced/unvoiced error results in line with those on the Bagshaw database. Further manual inspection reveals a result that matches very well with “naked eye” pitch tracking. It must also be said that the Keele database is not as clean as the Bagshaw one, as its purpose was a wider field of research than was attempted here.

4.6 Conclusions & Suggestions

A PDA has been presented that successfully integrates time-domain and frequency-domain data to yield pitch-marks that are useful for pitch-synchronous pitch and duration mo-

	Bagshaw	Keele
E_{gross}	0.6%	2%
$E_{v \rightarrow u}$	10%	12%
$E_{u \rightarrow v}$	4%	4%

Table 4.1: Pitch-tracking results for the DAG PDA.

dification schemes. As far as perceived pitch is concerned, this work hardly scratches the surface. The ability of pitch modification schemes is limited when no clear harmonic behaviour is present anyway. For clean speech from speakers whose voices will work well with speech synthesis and voice modification schemes, the DAG PDA achieves good results with very few parameters to fine-tune.

The biggest failing of this incarnation of the pitch tracker is its enforcement of two assumptions regarding the nature of pitch in speech:

1. voiced speech contains regularly spaced, easy to discern pitch-pulses, and
2. pitch shows up as clear harmonics in the magnitude spectrum that can be extracted using the HPS.

This makes it prone to miss voiced regions in croaky voices where clear peaks can be found but they are not regularly spaced. Voiced speech (where pitch is clearly present and regular) may not have clear peaks, often causing the graph pitch tracker to lose synchronisation. This results in jumps in the pitch contour where it was not overly detrimental to the DAG optimisation score—the new synchronisation could make up for the loss incurred by one low scoring transition. Such “re-synchronisation” events cause breaks in voiced regions.

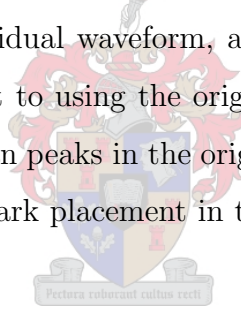
The test methodology and results confirm the difficulty of evaluating PDAs using reference data. It has been suggested that speech synthesisers or vocoders be used, but this is complicated by the myriad of other factors and interactions in a complex system such as a speech synthesiser. It would also not be representative of the problem of finding analysis time-instants in natural speech for LP PSOLA.

The work of Wang, Wu and Brown [83] shows another way of how auditory-based time-domain and frequency-domain features can be joined to yield good pitch estimates. The

graph pitch post-processing scheme has the advantage that it does not require parameters to be trained, and provides another way of integrating pitch information from both time and frequency domains into one pitch estimate. More sophisticated pre-processing as proposed by Wang et al. could improve the DAG PDA.

The DAG PDA synchronises pitch-marks by positioning them on peaks in the band-pass filtered intermediate waveform. Synchronisation on peaks in the LP residual is perhaps more relevant to LP PSOLA synthesis. This proved difficult in practice since the phase spectrum is often smeared, and thus a discernible peak is not always available.

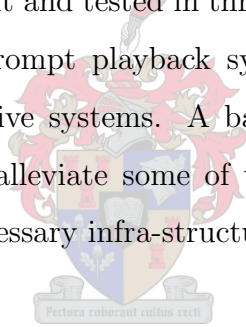
A technique that enabled building a high-quality vocoder at ≈ 2000 bits per second computes the phase-equalised pitch waveform [113]. The residual is put through a phase equalisation filter. The phase smearing in the LP residual was found to evolve slowly enough that processing each frame in the LP residual in this way does not result in audible phase modulation artifacts. The phase-equalised residual exhibits much clearer peaks than the unprocessed residual waveform, and synthesis using the phase-equalised residual is reportedly equivalent to using the original residual. Synchronising the pitch-marks on these peaks rather than peaks in the original waveform, would yield much more robust and appropriate pitch-mark placement in the LP PSOLA context.



Chapter 5

Conclusion & Suggestions

The state-of-the-art in unit-selection waveform synthesis techniques in speech synthesis has been surveyed and summarised. A limited-domain synthesiser for a prototype hotel reservation system has been built and tested in three different languages. The synthesiser is more flexible than simple prompt playback systems and gives much better output quality than simple concatenative systems. A baseline system for high quality speech modification was necessary to alleviate some of the shortcomings of the unit-selection system. It also provides the necessary infra-structure for future work on different aspects in speech synthesis.



Corpus design and post-recording processing procedures were developed for building the voices. These automate much of the work involved in producing high quality voices for specific applications, enabling a short turn-around time for new voices.

This work also laid the groundwork for more general synthesis. The waveform synthesis principles explored here are staples of the field of TTS. The LP PSOLA waveform synthesiser repairs some of the damage done by the concatenation procedure, making it more robust. More significantly, it provides the ability to realise high quality speech with prosody as predicted by automatic systems. This ability is crucial to further research in prosody prediction in languages such as Xhosa and Afrikaans.

The limited-domain speech synthesiser's major shortcoming is its limited vocabulary. The fact that so much more than words have been encoded into the recorded utterances also means that the words contained in the corpus are not usable in different contexts. The speech modification component alleviates this to some extent, but natural intonation

in all cases will require more sophisticated methods to determine target pitch.

5.1 Future Work

Speech synthesis is far from being a solved problem. Similar to speech recognition however, good results may be obtained from methods that exploit the specific simplifications that a particular application allows. This work showed the viability of exploiting the specific form of the dialogue in an SDS using limited-domain concatenative synthesis in Afrikaans, English and Xhosa. The next step is to start lifting some of the restrictions.

The unit-selection and waveform synthesis components developed here also provide a basis for testing the higher-level TTS components, such as orthographic to phonetic conversion (lexicon, rule-based orthographic to phonetic conversion and cross-word effects), language modelling (for determining parts of speech for homograph disambiguation and prosody) and prosodic modelling. Indeed, if the number of pages devoted to waveform synthesis compared to the number of pages describing the text-processing modules in Klatt's [2] original work is an indication, the bulk of the work remains to be done.

The rest of this section discusses some directions for future work to extend the synthesis methodology in this work in various directions. Section 5.1.1 mentions a simple data-driven technique that has the potential to improve the robustness of the system to inconsistent intonation patterns in the recordings. Section 5.1.3 highlights steps toward lifting some constraints on the vocabulary of the synthesiser. Section 5.1.4 glosses over the first stumbling blocks toward general synthesis.

5.1.1 Intonation

In the slot-and-filler reply generation context, it is possible to decompose intonation components due to the utterance time-frame, and the lexical and syllable time-frames in terms of Fujisaka's superpositional model of intonation [8]. This could provide a data-driven method of intonation prediction in the limited-domain synthesiser and improve pitch continuity by providing an utterance global, coherent pitch contour.

Such a more sophisticated intonation system will make the limited-domain synthesiser much more robust against the prosodic inconsistencies that are so difficult to control

during recording. It will also provide a way to predict intonation for novel words, given syllabic stress or pitch accents.

5.1.2 Improved Unit Selection

If intonation prediction and modification is to be included in such a system, the unit-selection methodology must be made prosody-aware. It should be biased toward preferring units that are a bit longer than desired, since shortening units loses less quality than lengthening them. It should also prefer units with roughly the correct pitch.

The unit selection may be further improved by including heuristics to make it more aware of the underlying phonetics. The current system splices units on phoneme boundaries. Vowels, diphthongs and other sustained voiced sounds are better spliced in the middle. Slight spectral or pitch mismatches caused by this are easily removed using the LP PSOLA speech modification engine.

5.1.3 Novel words

Synthesis of novel words in the limited-domain synthesiser requires specification of the phonemes of the novel word, its intonation, the relative loudness of phonemes and their duration. The phonemes can be provided by a lexicon that is extended as the new words are introduced.¹ The lexicon should also provide syllabic stress (Afrikaans and English) and pitch accents or tonal patterns (Xhosa).

Syllabic prominence is reportedly quite regular in Xhosa, and could be usefully predicted by rules. In English and Afrikaans it can be predicted reasonably well from syllabic stress, parts of speech or additional mark-up in the synthesis specification, also using rules. This obviously requires that the database contains more than just the units needed for the limited-domain task. As previously mentioned, the addition of prompts that cover diphones in carefully selected contexts and are recorded in a fairly bland and almost hyper-articulated style, will not only provide the needed additional units, but also make automatic alignment using HMMs much more reliable.

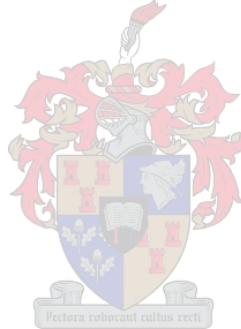
¹The most useful novel words would be proper nouns, on which LTS rules are of little use.

The intonation and duration targets can be realised with high quality using the speech modification engine described in Chapter 3.

5.1.4 General Synthesis

English and Afrikaans are stress languages, which absolutely necessitates a lexicon that gives syllabic stress. Xhosa is traditionally viewed as a tone language, although it contains strong elements of a pitch accent language [80]. Xhosa synthesis requires a morphological parser and accompanying lexicon that can predict placement of various tones and pitch accents. All three lexical systems must also be able to produce parts of speech.

Once the unit-selection synthesiser is capable of synthesising novel words, the ability to derive these higher level features from text is the enabling factor in building fully automatic methods of producing natural prosody.



Bibliography

- [1] DELLER, J., HANSEN, J., and PROAKIS, J., *Discrete-Time Processing of Speech Signals*. New York: MacMillan, 1993.
- [2] ALLEN, J., HUNNICUTT, M. S., and KLATT, D., *From Text to Speech: The MITalk System*. Cambridge University Press, 1987.
- [3] “Voice Extensible Markup Language (VoiceXML) Version 2.0.”
<http://www.w3.org/TR/voicexml20/>. May 2004.
- [4] SCHEFFLER, K. and YOUNG, S., “Probabilistic Simulation of Human Machine Dialogues.” in *Proceedings of the ICASSP*, June 2000.
- [5] JURAFSKI, D. and MARTIN, J. H., *Speech and Language Processing*. First edition. Upper Saddle River, New Jersey, USA.: Prentice Hall, 2000.
- [6] CAMPBELL, N., “Where is the Information in Speech? (and to what extent can it be modelled in synthesis?).” in *The Proceedings of the Third ESCA/COCOSDA Workshop on Speech Synthesis*, (Jenolan Caves, Australia), 1998. available at <http://www.slt.atr.co.jp/cocosda/jenolan/sessions.htm> (23 January 2002).
- [7] VAN SANTEN, J., MACON, M., CRONK, A., HOSOM, P., KAIN, A., PAGEL, V., and WOUTERS, J., “When will Synthetic Speech Sound Human: Role of Rules and Data.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [8] BOTINIS, A., GRANSTRÖM, B., and MÖBIUS, B., “Developments and Paradigms in Intonation Research.” *Speech Communication*, 2001, Vol. 33, pp. 262–296.

- [9] SPROAT, R., HUNT, A., OSTENDORF, M., TAYLOR, A., P. BLACK, LENZO, K., and EDINGTON, A., “SABLE: A Standard for TTS Markup.” in *Proceedings of the ICSLP*, (Sydney, Australia), 1998.
- [10] KOCHANSKI, G. and SHIH, C., “Prosody Modelling with Soft Templates.” *Speech Communication*, 2003, Vol. 39, pp. 311–352.
- [11] WOUTERS, J., RUNDLE, B., and MACON, M. W., “Authoring Tools for Speech Synthesis using the SABLE Markup Standard.” in *Proceedings of Eurospeech*, (Budapest Hungary), 1999.
- [12] SILVERMAN, K., BECMAN, M., PITRELLI, J., OSTENDORF, M., *et al.*, “ToBI: A Standard for Labelling English Prosody.” in *Proceedings of the ICSLP*, vol. 2, pp. 867–870, 1992.
- [13] TAYLOR, P., “Analysis and Synthesis of Intonation using the Tilt Model.” *Journal of the Acoustical Society of America*, 2000, Vol. 107 3, pp. 1697–1714.
- [14] SHIH, C. and KOCHANSKI, G. P., “Synthesis of Prosodic Styles.” in *The Proceedings of the Fourth ISCA Tutorial and Research Workshop on Speech Synthesis*, (Atholl Palace Hotel, Scotland), 2001. available at <http://www.ssw4.org/>.
- [15] SHADLE, C. and DAMPER, R., “Prospects for articulatory synthesis: A position paper.” in *The Proceedings of the Fourth ISCA Tutorial and Research Workshop on Speech Synthesis*, (Atholl Palace Hotel, Scotland), 2001. available at <http://www.ssw4.org/>.
- [16] XUEDONG HUANG, H.-W. H., ALEX ACERO, *Spoken Language Processing*. Upper Saddle River, New Jersey: Prentice Hall PTR, 2001.
- [17] MCAULAY, R. J. and QUATIERI, T. F., “Speech Analysis/Synthesis based on a Sinusoidal Representation.” *IEEE Transactions on Acoustics, Speech and Signal Processing*, August 1986, Vol. 34, No. 4, pp. 744–754.

- [18] STYLIANOU, Y., “Applying the Harmonics plus Noise Model in Concatenative Speech Synthesis.” *IEEE Transactions on Speech and Audio Processing*, January 2001, Vol. 9, No. 1, pp. 21–29.
- [19] HUNT, A. and BLACK, A., “Unit Selection in a Speech Synthesis System using a Large Speech Corpus.” in *Proceedings of the ICASSP*, (Atlanta, Georgia), May 1996.
- [20] YI, J., *Corpus-based Unit Selection for Natural-Sounding Speech Synthesis..* Ph.D. dissertation, Massachusetts Institute of Technology, 2003.
- [21] “CHATR (Generic Speech Synthesis System).”
<http://www.itl.atr.co.jp/chatr/>, November 2000.
- [22] YI, J., “Natural Sounding Speech Synthesis using Variable Length Units.” Master’s thesis, Massachusetts Institute of Technology, May 1998.
- [23] TAKANO, S., TANAKA, K., MIZUNO, H., ABE, M., and NAKAJIMA, S., “A Japanese TTS System Based on Multiform Units and a Speech Modification Algorithm with Harmonics Reconstruction.” *IEEE Transactions on Speech and Audio Processing*, January 2001, Vol. 9, No. 1, pp. 3–10.
- [24] CONKIE, A., “Robust Unit Selection System for Speech Synthesis.” in *Joint Meeting of the ASA, EAA, and DAGA*, (Berlin, Germany), May 1999.
- [25] DONOVAN, R. E., *Trainable Speech Synthesis.* Ph.D. dissertation, Cambridge University Engineering Department, 1996.
- [26] HUANG, X., ACERO, A., ADCOCK, J., HON, H., and GOLDSMITH, J., “Whistler: A trainable Text-to-Speech System.” in *Proceedings of the ICSLP*, (Philadelphia), October 1996.
- [27] BULYKO, I., *Flexible Speech Synthesis Using Weighted Finite State Transducers.* Ph.D. dissertation, University of Washington, 2002.
- [28] WIGHTMAN, C. W., K.SYRDAL, A., STEMMER, G., CONKIE, A., and BEUTNAGEL, M., “Perceptually Based Automatic Prosody Labeling and

- Prosodically Enriched Unit Selection to Improve Concatenative Text-to-Speech Synthesis.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [29] CAMPBELL, N. and BLACK, A. W., “Prosody and the Selection of Source Units for Concatenative Synthesis.” in van Santen *et al.* [115], Ch. 22.
- [30] BREEN, A. P. and JACKSON, P., “Non-uniform Unit Selection and the Similarity Metric within BT’s Laureate TTS System.” in *The Proceedings of the Third ESCA/COCOSDA Workshop on Speech Synthesis*, (Jenolan Caves, Australia), 1998. Available at <http://www.slt.atr.co.jp/cocosda/jenolan/sessions.htm> (23 January 2002).
- [31] SYRDAL, A. K., WIGHTMAN, C. W., CONKIE, A., STYLIANOU, Y., SCHROETER, M. B. J., STROM, V., LEE, K.-S., and MAKASHAY, M. J., “Corpus-based Techniques in the AT&T NextGen Synthesis System.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [32] SYRDAL, A. K. and MCGORY, J., “Inter-transcriber Reliability of ToBI Prodosic Labeling.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [33] BLACK, A. W. and DUSTERHOFF, K., “Generating F0 Contours for Speech Synthesis using the Tilt Intonation Theory.” in *Proceedings of ESCA Workshop of Intonation*, (Athens, Greece), IEEE, 1997.
- [34] BLACK, A. W. and FONT LLITJÓ, A., “Unit Selection Without a Phoneme Set.” in *IEEE TTS Workshop*, (Santa Monica, California), IEEE, 2002.
- [35] BLACK, A. W. and BENNETT, C. L., “Using Acoustic Models to Choose Pronunciation Variations for Synthetic Voices..” in *Proceedings of Eurospeech*, (Geneva, Belgium), 2003.
- [36] KIM, Y.-J., SYRDAL, A., and JILKA, M., “Improving TTS by Higher Agreement Between Predicted versus Observed Pronunciations.” in *The Proceedings of the Fifth ISCA Tutorial and Research Workshop on Speech Synthesis*, (Carnegie Mellon University, Pittsburgh), 2001. available at <http://www.ssw5.org/>.

- [37] DONOVAN, R. E., “Topics in Decision Tree Based Speech Synthesis.” *Computer Speech and Language*, 2003, No. 17, pp. 40–48.
- [38] MALFRÈRE, F., DU TOIT, T., DEROO, O., and RIS, C., “Phonetic Alignment: Speech Synthesis-based vs. Viterbi-based.” *Speech Communication*, 2003, Vol. 40, pp. 503–515.
- [39] WU, C.-H. and CHEN, J.-H., “Automatic Generation of Synthesis Units and Prosodic Information for Chinese Concatenative Synthesis.” *Speech Communication*, October 2001, Vol. 35, No. 3-4, pp. 219–237.
- [40] VAN SANTEN, J. P. H. and BUCHSBAUM, A. L., “Methods for Optimal Text Selection.” in *Proceedings of Eurospeech*, (Rhodes, Greece), pp. 553–556, Sep 1997.
- [41] VAN SANTEN, J. P. H., “Combinatorial Issues in Text-to-Speech Synthesis.” in *Proceedings of Eurospeech*, (Rhodes, Greece), pp. 2511–2514, Sep 1997.
- [42] MÖBIUS, B., “Rare events and Closed Domains: Two Delicate Topics in Speech Synthesis.” in *The 4th ESCA Workshop on Speech Synthesis*, (Scotland), ESCA, 2001.
- [43] FRANÇOIS, H. and BOËFFARD, O., “Design of an optimal continuous speech database for text-to-speech synthesis considered as a set-covering problem..” in *Proceedings of Eurospeech*, (Scandinavia), 2001.
- [44] BELLEGARDA, J. R., LENZO, K., SILVERMAN, K. E. A., and ANDERSON, V., “Statistical Prosodic Modelling: From Corpus Design to Parameter Estimation.” *IEEE Transactions on Speech and Audio Processing*, January 2001, Vol. 9, pp. 52–66.
- [45] BLACK, A. W. and LENZO, K. A., “Optimal Data Selection for Unit Selection Synthesis.” in *Proceedings of Eurospeech*, (Scandinavia), 2001.
- [46] BLACK, A. W., “Perfect Synthesis for All of the People All of the Time.” in *IEEE TTS Workshop*, (Santa Monica, California), IEEE, 2002.

- [47] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., and STEIN, C. (Eds), *Introduction to Algorithms*. Second edition. Cambridge, Massachusetts, USA: MIT Press, 2001.
- [48] BLACK, A. W., LENZO, K. A., and CALEY, R., “The Festival Speech Synthesis System, System Documentation.”
<http://www.cstr.ed.ac.uk/projects/festival.html>. August 2004.
- [49] TAYLOR, P. and BLACK, A. W., “Speech Synthesis by Phonological Structure Matching.” in *Proceedings of Eurospeech*, (Budapest Hungary), 1999.
- [50] SCHWEITZER, A., BRAUNSCHWEILER, N., KLANKERT, T., MÖBIUS, B., and SÄUBERLICH, B., “Restricted Unlimited Domain Synthesis.” in *Proceedings of Eurospeech*, (Rhodes, Greece), Sep 2003.
- [51] ENGELBRECHT, H. A., “Automatic Phoneme Recognition of South African English.” Master’s thesis, University of Stellenbosch, April 2004.
- [52] BLACK, A. W. and TAYLOR, P., “Automatically Clustering Similar Units for Unit Selection in Speech Synthesis.” in *Proceedings of Eurospeech*, (Rhodes, Greece), Sep 1997.
- [53] KLABBERS, E. and VELDHUIS, K., “Reducing Audible Spectral Discontinuities.” *IEEE Transactions on Speech and Audio Processing*, January 2001, Vol. 9, pp. 39–51.
- [54] DONOVAN, R. E., “A New Distance Measure for Costing Spectral Discontinuities in Concatenative Speech Synthesizers.” in *The Proceedings of the Fourth ISCA Tutorial and Research Workshop on Speech Synthesis*, (Atholl Palace Hotel, Scotland), 2001. available at <http://www.ssw4.org/>.
- [55] COORMAN, G., FACKRELL, J., RUTTEN, P., and VAN COILLE, B., “Segment Selection in the L&H Realspeak Laboratory TTS System.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.

- [56] CONKIE, A., BEUTNAGEL, M., K.SYRDAL, A., and BROWN, P., “Preselection of Candidate Units in a Unit Selection-based Text-To-Speech Synthesis System.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [57] VAN DEN HEUVEL, M., PRETORIUS, L., WIESE, B., and GEGGUS, K., “Speech Interface Dialogue Specification for a Hotel Reservation System v1.0.” tech. rep., African Speech Technology, 2001.
- [58] SWART, P. H., “Prosodic Features of Imperatives in Xhosa: Implications for a Text-to-Speech System.” Master’s thesis, University of Stellenbosch, 2000.
- [59] “Linguistic Data Consortium.” <http://www.ldc.upenn.edu/>. April 2004.
- [60] GAUVAIN, J.-L. and LEE, C.-H., “Maximum *a Posteriori* Estimation for Multivariate Gaussian Mixture Observations of Markov Chains.” *IEEE Transactions on Speech and Audio Processing*, April 1994, Vol. 2, pp. 291–298.
- [61] LAROCHE, J. and MOULINES, E., “Non-parametric techniques for pitch-scale and time-scale modifications of speech.” *Speech Communication*, 1995, Vol. 16, pp. 175–205.
- [62] SCHWARDT, L., “Voice Conversion: An Investigation.” Master’s thesis, University of Stellenbosch, 1998.
- [63] FLANAGAN, J. and GOLDEN, R., “Phase Vocoder.” *Bell Systems Technical Journal*, 1966, Vol. 45, pp. 1493–1509.
- [64] MOULINES, E. and CHARPENTIER, F., “Pitch-Synchronous Waveform Processing Techniques for text-to-speech Synthesis Using Diphones.” *Speech Communication*, 1990, Vol. 9, No. 5/6, pp. 453–467.
- [65] GOODWIN, M. M., *Adaptive Signal Models: Theory, Algorithms and Audio Applications*. Ph.D. dissertation, University of California, Berkeley, 1997.
- [66] PROAKIS, J. G. and MANOLAKIS, D. G., *Digital Signal Processing*. Third edition. New Jersey, USA: Prentice Hall, 1996.

- [67] MCAULAY, R. J. and QUATIERI, T. F., “Shape Invariant Time-Scale and Pitch Modification of Speech.” *IEEE Transactions on Acoustics, Speech and Signal Processing*, March 1992, Vol. 40, No. 3, pp. 497–510.
- [68] KAWAHARA, H., MASUDA-KATSUSE, I., and DE CHEVEIGNÉ, A., “Restructuring Speech using a Pitch-Adaptive Time-Frequency Smoothing and an Instantaneous-Frequency-based F0 extraction: Possible Role of a Repetitive Structure in Sounds.” *Speech Communication*, 1999, No. 27, pp. 187–207.
- [69] ZOLFAGHARI, P. and KAWAHARA, H., “A Sinusoidal Model Based on Frequency-to-Instantaneous Frequency Mapping.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [70] GEORGE, E. and M.J.T., S., “Analysis-by-synthesis Overlap-add Sinusoidal Modelling Applied to the Analysis and Synthesis of Musical Tones.” *Journal of the Audio Engineering Society*, 1992, Vol. ASSP-32, No. 6, pp. 497–516.
- [71] MACON, M. W., *Speech Synthesis Based on Sinusoidal Modeling*. Ph.D. dissertation, Georgia Institute of Technology, 1996.
- [72] VERMA, T. S. and MENG, T. H. Y., “Sinusoidal Modelling using Frame-based Perceptually Weighted Matching Pursuits.” in *Proceedings of the ICASSP*, 1999.
- [73] O’BRIEN, D. and MONAGHAN, A. I. C., “Concatenative Synthesis Based on a Harmonic Model.” *IEEE Transactions on Speech and Audio Processing*, January 2001, Vol. 9, pp. 11–20.
- [74] RICHARD, G. and D’ALESSANDRO C.R., “Modification of the Unvoiced Component of Speech Signals for Synthesis.” in van Santen *et al.* [115], Ch. 4.
- [75] JACKSON, P. J. B. and SHADLE, C. H., “Pitch-Scaled Estimation of Simultaneous Voiced and Turbulence-Noise Components in Speech.” *IEEE Transactions on Speech and Audio Processing*, October 2001, Vol. 9, No. 7, pp. 713–725.

- [76] KAIN, A. and STYLIANOU, Y., “Stochastic Modeling of Spectral Adjustment for High Quality Pitch Modification.” in *Proceedings of the ICASSP*, vol. 2, pp. 949–952, June 2000.
- [77] WOUTERS, J. and MACON, M. W., “Unit Fusion for Concatenative Synthesis.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [78] CHAPPELL, D. T. and HANSEN, J. H. L., “A Comparison of Smoothing Methods for Segment Concatenation Based Speech Synthesis.” *Speech Communication*, 2002, Vol. 36, pp. 343–374.
- [79] RANK, E., “Exploiting Improved Parameter Smoothing Within a Hybrid Concatenative Speech Synthesiser.” in *Proceedings of Eurospeech*, (Budapest Hungary), 1999.
- [80] ROUX, J., “Xhosa: A Tone or Pitch Accent Language?.”
- [81] HERMES, D. J., “Pitch Analysis.” in *Visual Representations of Speech Signals* (COOK, M., BEET, S., and CRAWFORD, M. (Eds)), Ch. 1, England: John Wiley & Sons, 1993.
- [82] HESS, W., *Pitch Determination of Speech Signals: Algorithms and Devices*. Berlin: Springer, 1983.
- [83] WU, M., WANG, D., and J. BROWN, G., “A Multi-pitch Tracking Algorithm for Noisy Speech.” in *Proceedings of the ICASSP*, pp. 369–372, 2002.
- [84] RABINER, L., “On the use of Autocorrelation Analysis for Pitch Detection.” *IEEE Trans. on Acoustics and Speech Signal Processing*, 1977, Vol. 25, pp. 24–33.
- [85] MEDAN, Y., YAIR, E., and CHAZAN, D., “Super-resolution Pitch Determination.” *IEEE Transactions on Signal Processing*, 1991, pp. 40–48.
- [86] BAGSHAW, P. C., HILLER, S. M., and JACK, M. A., “Enhanced Pitch Tracking and the Processing of F0 Contours for Computer Aided Intonation Teaching.” in *Proceedings of Eurospeech*, (Berlin), 1993.

- [87] SHIMAMURA, T. and KOBAYASHI, H., “Weighted Autocorrelation for Pitch Extraction of Noisy Speech.” *IEEE Transactions on Speech and Audio Processing*, 2001, Vol. 9, pp. 727–740.
- [88] SAKAMOTO, M. and SAITOH, T., “An Automatic Pitch-Marking Method using Wavelet Transform.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [89] SHABANA, F. and FITCH, J., “A Wavelet-Based Pitch Detector For Musical Signals.” in *Proceedings of the 2nd COSTG6 Workshop on Digital Audio Effects*, 1999.
- [90] KADAMBE, S. and BOUDREAUX-BARTELS, G., “Application of the wavelet transform for pitch detection of speech signals..” *IEEE Transactions on Information Theory*, 1992, Vol. 38, No. 2.
- [91] TEREZ, D. E., “Robust Pitch Determination using Nonlinear State-Space Embedding.” in *Proceedings of the ICASSP*, 2002.
- [92] MANN, I. and MCLAUGHLIN, S., “A Nonlinear Algorithm for Epoch Marking in Speech Signals using Poincaré Maps.” in *Proceedings of the 9th European Signal Processing Conference*, vol. 2, pp. 701 – 704, September 1998.
- [93] HINICH, M., “Detecting a Hidden Periodic Signal when its Period is Unknown.” *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1982, Vol. 5, pp. 747–750.
- [94] SENEFF, S., “Real-time Harmonic Pitch Detector..” *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1978, Vol. 26, No. 4, pp. 358–365.
- [95] MCAULAY, R. J. and QUATIERI, T. F., “Pitch Estimation and Voicing Decision based on a Sinusoidal Model.” in *ICASSP*, (Albuquerque), pp. 249–252, 1990.
- [96] CHAZAN, D., TZUR (ZIBULSKI), M., HOORY, R., and COHEN, G., “Efficient Periodicity Extraction Based on Sine-wave Representation and its Application to Pitch Determination of Speech Signals.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.

- [97] ATAKE, Y., IRINO, T., KAWAHARA, H., LU, J., NAKAMURA, S., and SHIKANO, K., “Robust Fundamental Frequency Estimation Using Instantaneous Frequencies of Harmonic Components.” in *Proceedings of the ICSLP*, (Beijing, China), 2000.
- [98] CHARPENTIER, F., “Pitch Detection using the Short-term Phase Spectrum.” in *Proceedings of the IEEE ICASSP*, pp. 113–116, 1986.
- [99] IMMERSEEL, L. M. V. and MARTENS, J.-P., “Pitch and Voiced/Unvoiced Determination with an Auditory Model.” *Journal of the Acoustical Society of America*, June 1992, Vol. 91, No. 6, pp. 3511–3526.
- [100] GU, Y. and VAN BOKHOVEN, W., “Co-channel Speaker Separation Using a Frequency Bin Non-linear Adaptive Filter..” in *Proceedings of the ICASSP*, pp. 949–952, 1991.
- [101] ROUAT, J., LIU, Y. C., and MORISETTE, D., “A Pitch Determination and Voiced/Unvoiced Decision Algorithm for Noisy Speech..” *Speech Communication*, 1997, Vol. 21, pp. 191–207.
- [102] RABINER, L., SAMBUR, M., and SCHMIDT, C., “Applications of Non-linear Smoothing Algorithms to Speech Processing.” *IEEE Transactions of Acoustics, Speech and Signal Processing*, 1975, Vol. 23, No. 6, pp. 552–557.
- [103] GONCHAROFF, V. and GRIES, P., “An algorithm for accurately marking pitch pulses in speech signals.” in *Proceedings of the IASTED International Conference, Signal and Image Processing*, (Las Vegas, USA), 1998.
- [104] HARBECK, S., KIESSLING, A., KOMPE, R., NIEMANN, H., and NÖTH, E., “Robust Pitch Period Detection Using Dynamic Programming With An ANN Cost Function.” in *Proceedings of Eurospeech*, (Madrid), 1995.
- [105] “DP-based Determination of F0 Contour from Speech Signals..”
- [106] NEY, H., “Dynamic Programming Algorithm for Optimal Estimation of Speech Parameter Contours.” *IEEE Transactions on Systems, Man and Cybernetics*, 1983, Vol. 13, pp. 208–214.

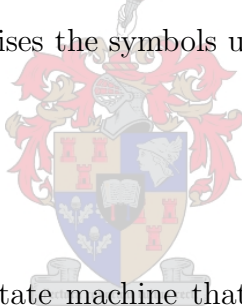
- [107] SJÖLANDER, K., “Snack v2.7 - A Sound Toolkit for Scripting Languages.”
<http://www.speech.kth.se/snack/>. July 2004.
- [108] BAGSHAW, P. C., *Automatic Prosodic Analysis for Computer Aided Pronunciation Teaching*. Ph.D. dissertation, University of Edinburgh, 1994.
- [109] RABINER, L., CHENG, M., ROSENBERG, A., and MCGONEGAL, C., “A Comparative Performance Study of Several Pitch Detection Algorithms.” *IEEE Trans. on Acoustics and Speech and Signal Processing*, 1976, Vol. 24, pp. 399–417.
- [110] ZELJKOVIC, I. and STYLIANOU, Y., “Single Complex Sinusoid and ARHE Model based Pitch Extractors.” in *Proceedings of Eurospeech*, (Budapest Hungary), 1999.
- [111] DE CHEVEIGNÉ, A. and KAWAHARA, H., “Comparative evaluation of F_0 estimation algorithms.” in *Proceedings of Eurospeech*, (Scandinavia), 2001.
- [112] MEYER, G., PLANTE, F., and AINSWORTH, W., “A Pitch Extraction Reference Database.” in *Proceedings of Eurospeech*, (Madrid), 1995.
- [113] HIWASAKI, Y., MANO, K., and KANEKO, T., “An LPC Vocoder based on Phase-Equalised Pitch Waveform.” *Speech Communication*, 2003, Vol. 40, pp. 277–290.
- [114] DE VILLIERS, E., “Automatic Alignment of Speech Corpora (unpublished).” Master’s thesis, University of Stellenbosch, 2004.
- [115] VAN SANTEN, J. P. H., SPROAT, R. W., OLIVE, J. P., and HIRSCHBERG, J. (Eds), *Progress in Speech Synthesis*. Springer-Verlag, 1996.

Appendix A

Hidden Markov Models

This Appendix describes the HMM. It starts by glossing over the basic theory to define terminology and notation, and then describes the embedded re-estimation, forced alignment and simplified Maximum *a Posteriori* (MAP) adaptation techniques used in Chapter 2. Section A.6 summarises the symbols used in this Appendix.

A.1 Introduction



An HMM is a weighted finite-state machine that generates a random vector called an *observation* upon entering a state [1, Chapter 12].¹ In this formulation, and as is common in speech recognition, the HMM is formulated to operate in discrete time. Figure A.1 depicts an HMM with a simple three-state structure. States 0 and 4 are null states. They do not emit observations, and they do not swallow up time steps. We only consider first-order HMMs, which implies that the state transition made at any one time only depends on the current state.

The HMM models two random processes. One is the “true” underlying process that is the state sequence of length T , $X = \{x(1), x(2), \dots, x(T)\}$, where $x(t) = i, 1 \leq i \leq S$ in an HMM with S states. The parameter a_{ij} is the probability of making the transition from state i to state j , and can also be written as a component of the $S \times S$ matrix \mathbf{A} .

¹In the Moore form, the HMM generates a random vector upon entering a state. The Mealy form generates the random vector at each state transition. The Moore form is used here.

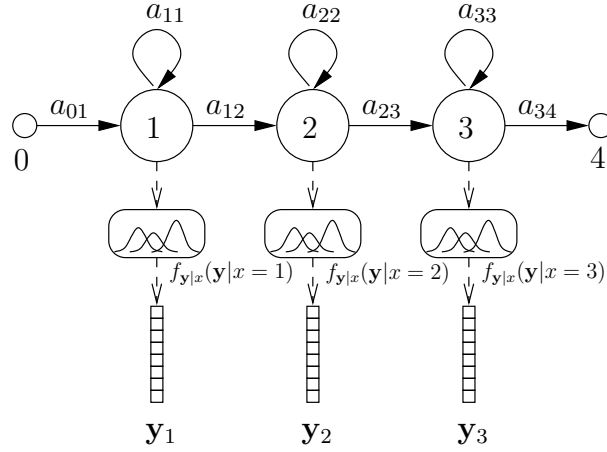


Figure A.1: A three-state phoneme model, with no forward skip.

Let $\boldsymbol{\pi}(t)$ be the vector containing the probabilities of being in each state at time t , and let $\boldsymbol{\pi}(1)$ be known. Then $\boldsymbol{\pi}(t) = \mathbf{A}^{t-1}\boldsymbol{\pi}(1)$, since we make the first-order assumption. The matrix \mathbf{A} and the initial probability distribution $\boldsymbol{\pi}(1)$ define the underlying random process completely.

The other random process is the observed or measured process, $Y = \{\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(T)\}$. The observed process is a series of random vectors of dimension D . For each state $x(t)$ in the state sequence an observed value $\mathbf{y}(t)$ is emitted. The HMM models the production of these observation vectors with the conditional PDF $f_{\mathbf{y}|x}(\mathbf{y}|x(t))$. We model it using a weighted mixture of M Gaussian PDFs.

$$f_{\mathbf{y}|x}(\mathbf{y}|x(t) = i) = \sum_{m=1}^M w_{im} \mathcal{N}(\mathbf{y}, \boldsymbol{\mu}_{im}, \mathbf{C}_{im}) \quad (\text{A.1})$$

where

$$\mathcal{N}(\mathbf{y}, \boldsymbol{\mu}, \mathbf{C}) = \frac{1}{(2\pi)^{D/2} |\mathbf{C}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})\mathbf{C}^{-1}(\mathbf{y} - \boldsymbol{\mu})^T\right\} \quad (\text{A.2})$$

and

$$\sum_{m=1}^M w_{im} = 1. \quad (\text{A.3})$$

The determinant of \mathbf{C} is written as $|\mathbf{C}|$.

From this discussion, it should now be clear that the HMM \mathcal{M} is completely specified by the parameters

$$\mathcal{M} = \{S, \mathbf{A}, \boldsymbol{\pi}(1), \{w_{im}, \boldsymbol{\mu}_{im}, \mathbf{C}_{im}, 1 \leq i \leq S, 1 \leq m \leq M\}\} \quad (\text{A.4})$$

where we assume that the Gaussian mixtures at all states $1 \leq i \leq S$ have M components.

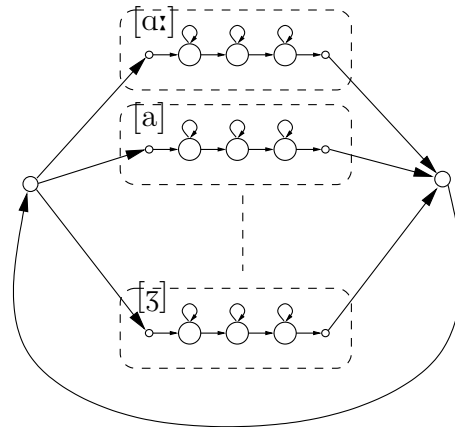


Figure A.2: A simple compound HMM to recognise connected speech.

A.2 Speech Recognition

The simple three-state HMM in Figure A.1 can be employed to model phonemes. A feature extraction step computes regularly spaced D -dimensional feature vectors from the input waveform (see Section A.3) which the HMM can model as its observations. A separate set of parameters is usually associated with each phoneme in the language, resulting in a set of phoneme models.

The parameters \mathcal{M} of each phoneme model are estimated from many examples of feature vector sequences associated with the particular phoneme during the training process (see Section A.4). A set of trained models can now be used to *identify* a given series of features as a specific phoneme by computing the likelihood that each phoneme model “generated” the data, and then selecting the model whose likelihood was the highest.

A sequence of feature vectors computed from a speech waveform can be modelled using a special *compound* HMM as depicted in Figure A.2 to *recognise* speech. The trained phoneme models are connected through null states with transitions embodying phoneme statistics. A Viterbi search then finds the most likely sequence of states X , given the observation sequence Y . The most likely sequence of states implies the most likely sequence of phonemes.

A third application is to find the boundaries of phonemes when the phoneme sequence is known; this is the application of interest for automatic alignment of speech synthesis corpora. Section A.5 treats it in more detail.

A.3 Features

Ideally, the feature vectors should contain information that is pertinent to the job at hand. In the case of phoneme modelling, the features should allow distinguishing between phonemes. In speech recognition it has been realised early on that the spectral envelope contains the necessary information, while spurious data such as pitch may be ignored for many languages. Compact representations of it has been found in linear prediction and its derived forms. A favourite is Mel-frequency cepstral coefficients (MFCCs) [1, Chapter 6]. Further work to enable the HMM to represent more perceptually valid features led to perceptual linear prediction (PLP), and its cousin that aims to remove non-stationary line effects from the features, RASTA.

The general feeling is that the cepstral vectors still contain too much unnecessary information, so linear discriminant analysis and principle component analysis are often used to reduce the dimensionality of the feature vectors. This leads to savings in computational cost while maintaining or even improving recognition results. Since cepstra do not contain any dynamic information, the dimensionality of the data is often raised before dimension reduction by adding difference or delta (Δ) vectors to the features. These Δ 's give an indication of velocity of change of the features. Acceleration, or $\Delta\Delta$'s, are often used as well.

In this work we chose a simple option: 12-dimensional MFCCs computed from speech sampled at 16kHz. The so-called zeroth cepstral coefficient was included as an indication of the energy in the signal. The vectors were computed from fixed-length 32ms frames, with the frame centres spaced 5ms apart. The relatively low dimensionality of the data leads to fewer parameters to estimate, which is better in light of the small size of the training sets.

A.4 Training

The task of training an HMM is that of estimating the parameters of the model \mathcal{M} . This section shows how the Expectation Maximisation (EM) re-estimation equations as given by Deller et al. [1, Chapter 12], can be viewed as the accumulation of sufficient statistics

from *weighted* examples. This interpretation then leads to the simplified MAP procedure, used to stabilise the HMMs given too few examples of each phoneme for estimating the parameters during embedded re-estimation.

The likelihood that an observation sequence was generated by a model can be estimated using the Baum-Welch estimation procedure. It provides the forward, α , and backward, β , joint probabilities that the model was in state i at time t , and generated the *forward* sequence $Y_1^t = \{\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(t)\}$ and will generate the *backward* sequence $Y_{t+1}^T = \{\mathbf{y}(t+1), \mathbf{y}(t+2), \dots, \mathbf{y}(T)\}$:

$$\alpha(Y_1^t, i) = P(Y_1^t, x(t) = i | \mathcal{M}) \quad (\text{A.5})$$

$$\beta(Y_{t+1}^T, i) = P(Y_{t+1}^T, x(t) = i | \mathcal{M}). \quad (\text{A.6})$$

This allows us to write down the probability of generating the observation sequence and being in state i at time t ,

$$P(Y, x(t) = i | \mathcal{M}) = \alpha(Y_1^t, i) \beta(Y_{t+1}^T, i), \quad (\text{A.7})$$

and the total probability of the observation sequence,

$$P(Y | \mathcal{M}) = \sum_{i=1}^S \alpha(Y_1^t, i) \beta(Y_{t+1}^T, i), \quad \text{for any } 1 \leq t \leq T. \quad (\text{A.8})$$

Now we can write the conditional probability of $x(t) = i$ given the observation sequence Y :

$$P(x(t) = i | Y, \mathcal{M}) = \frac{P(Y, x(t) = i | \mathcal{M})}{P(Y | \mathcal{M})} \quad (\text{A.9})$$

$$= \frac{\alpha(Y_1^t, i) \beta(Y_{t+1}^T, i)}{\sum_{j=1}^S \alpha(Y_1^t, j) \beta(Y_{t+1}^T, j)} \quad (\text{A.10})$$

$$\triangleq \nu_{hmm}(i; t). \quad (\text{A.11})$$

For later use, let $\nu_{hmm}(i; t)$ denote this quantity. It is the weight or importance that the HMM gives to the training vector for re-estimating parameters of state i at time t . A qualitative way to look at it is that $\nu_{hmm}(i; t)$ states how much a training sample at time t “belongs” to state i .

Similarly, the importance of the training sample to the component m of the Gaussian mixture at state i is expressed by the probability of the component given the observation

and the state:

$$P(m|\mathbf{y}(t), x(t) = i) = \frac{w_{im}\mathcal{N}(\mathbf{y}(t), \boldsymbol{\mu}_{im}, \mathbf{C}_{im})}{\sum_{l=1}^M w_{il}\mathcal{N}(\mathbf{y}(t), \boldsymbol{\mu}_{il}, \mathbf{C}_{il})} \quad (\text{A.12})$$

$$\triangleq \nu_{gmm}(i; t, m). \quad (\text{A.13})$$

The equations for computing new estimates $\{\bar{w}_{im}, \bar{\boldsymbol{\mu}}_{im}, \bar{\mathbf{C}}_{im}, 1 \leq m \leq M\}$ of the Gaussian mixture parameters $\{w_{im}, \boldsymbol{\mu}_{im}, \mathbf{C}_{im}, 1 \leq m \leq M\}$ for the next iteration of the EM algorithm are given in [1, Chapter 12] in terms of $\nu(i; t, m)$. Using Equations A.11 and A.13 it can be written as

$$\nu(i; t, m) = \frac{\alpha(Y_1^t, i) \beta(Y_{t+1}^T, i)}{\sum_{j=1}^S \alpha(Y_1^t, j) \beta(Y_{t+1}^T, j)} \cdot \frac{w_{im}\mathcal{N}(\mathbf{y}(t), \boldsymbol{\mu}_{im}, \mathbf{C}_{im})}{\sum_{l=1}^M w_{il}\mathcal{N}(\mathbf{y}(t), \boldsymbol{\mu}_{il}, \mathbf{C}_{il})} \quad (\text{A.14})$$

$$= \nu_{hmm}(i; t) \cdot \nu_{gmm}(i; t, m). \quad (\text{A.15})$$

This separation of the weights ν as being a result of the HMM and of the Gaussian mixture enables a flexible software architecture that allows a hierarchical structure in which weighted training vectors can be passed from the HMM to the correct Gaussian mixture. This simplifies tying of parameters and sharing of Gaussian mixtures among different states and even different phoneme models.

The equations for re-estimating the parameters of the HMM after the weights of each training vector for each state and Gaussian component has been gathered, can now be written as:

$$\bar{w}_{il} = \frac{\sum_{t=1}^T \nu(i; t, m)}{\sum_{m=1}^M \sum_{t=1}^T \nu(i; t, m)}, \quad (\text{A.16})$$

$$\bar{\boldsymbol{\mu}}_{il} = \frac{\sum_{t=1}^T \nu(i; t, m) \mathbf{y}(t)}{\sum_{t=1}^T \nu(i; t, m)}, \quad \text{and} \quad (\text{A.17})$$

$$\bar{\mathbf{C}}_{il} = \frac{\sum_{t=1}^T \nu(i; t, m) [\mathbf{y}(t) - \boldsymbol{\mu}_{il}] [\mathbf{y}(t) - \boldsymbol{\mu}_{il}]^T}{\sum_{t=1}^T \nu(i; t, m)}. \quad (\text{A.18})$$

These equations are easily extended for multiple training sequences—as is always the case in phoneme modelling. Let $\mathbf{y}(n, t)$ denote the sample at time t of the n 'th out of N

training observation sequences. Then

$$\bar{w}_{il} = \frac{\sum_{n=1}^N \sum_{t=1}^{T_n} \nu'}{\sum_{m=1}^M \sum_{n=1}^N \sum_{t=1}^{T_n} \nu'}, \quad (\text{A.19})$$

$$\bar{\boldsymbol{\mu}}_{il} = \frac{\sum_{n=1}^N \sum_{t=1}^{T_n} \nu' \mathbf{y}'}{\sum_{n=1}^N \sum_{t=1}^{T_n} \nu'}, \quad \text{and} \quad (\text{A.20})$$

$$\bar{\mathbf{C}}_{il} = \frac{\sum_{n=1}^N \sum_{t=1}^{T_n} \nu' [\mathbf{y}' - \boldsymbol{\mu}_{il}] [\mathbf{y}' - \boldsymbol{\mu}_{il}]^T}{\sum_{n=1}^N \sum_{t=1}^{T_n} \nu'} \quad (\text{A.21})$$

where $\nu' = \nu(i; n, t, m)$ and $\mathbf{y}' = \mathbf{y}(n, t)$.

A.4.1 Simplified MAP

Under ideal circumstances, the estimation of full covariance Gaussian mixture components require at least $D+1$ training vectors to yield invertible covariance matrices. This assumes that the vectors do not all happen to lie in a hyper-plane that is a subspace of the D -dimensional space. Even if they are only close to the hyper-plane, the covariance matrix will be ill-conditioned. Often there are simply not enough examples available to estimate the mixture properly. Remember, however, that in this application the models can be initialised using a large amount of data, while what little data is available during the retraining for the particular speaker who delivered the synthesis database, must have an influence on the parameters. More data is available for some phonemes than others, and where more data is available, the influence of the new data on the parameters should grow. This type of problem has been studied in speaker adaptation. MAP is one such approach [60].

A simplified implementation was available for experiments on adaptation of well initialised models for use in alignment of the HRS corpora (see Section 2.6.3.2). In principle, the method introduces the notion of phantom observations with weight $\nu = 1$. Each Gaussian component of the mixtures in the HMM is well initialised by training on different data, yielding the priors $\{w_{im}^{\mathcal{P}}, \boldsymbol{\mu}_{im}^{\mathcal{P}}, \mathbf{C}_{im}^{\mathcal{P}}\}$. The $N^{\mathcal{P}}$ phantom observations are chosen so that if only they were available, the parameters would remain the same. This amounts to

a modification of Equations A.20 and A.21:

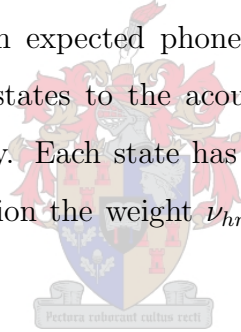
$$\bar{\boldsymbol{\mu}}_{il} = \frac{N^{\mathcal{P}} \boldsymbol{\mu}_{im}^{\mathcal{P}} + \sum_{n=1}^N \sum_{t=1}^{T_n} \nu' \mathbf{y}'}{N^{\mathcal{P}} + \sum_{n=1}^N \sum_{t=1}^{T_n} \nu'} \quad \text{and} \quad (\text{A.22})$$

$$\bar{\mathbf{C}}_{il} = \frac{N^{\mathcal{P}} \mathbf{C}_{im}^{\mathcal{P}} + \sum_{n=1}^N \sum_{t=1}^{T_n} \nu' [\mathbf{y}' - \boldsymbol{\mu}_{il}] [\mathbf{y}' - \boldsymbol{\mu}_{il}]^T}{N^{\mathcal{P}} + \sum_{n=1}^N \sum_{t=1}^{T_n} \nu'}. \quad (\text{A.23})$$

The equation for estimating the weights of the Gaussian mixture (Equation A.19) remain the same, as they require fewer examples to estimate reliably than the other parameters of the Gaussian mixture.

A.4.2 Embedded Re-estimation

Many authors refer to the seeming ability of the HMM to organise itself according to underlying data [1, Chapter 12]. The typical example is where a word model is constructed with roughly one state for each expected phoneme. In such cases, the HMM has an uncanny ability to match the states to the acoustic phenomena in the word. This is easy to understand qualitatively. Each state has a PDF that begins to “like” a certain distribution, and at each iteration the weight $\nu_{hmm} \nu_{gmm}$ matches the state’s PDF with training vectors that



1. are similar and
2. fits into the temporal structure of the model.

The latter suggests that limiting the temporal structure of the HMM may help to have states “adopt” the desired acoustic phenomena.

This self-organising property can be further exploited to train phoneme models when explicit phoneme boundaries are not available, but the phoneme sequence is known. Figure A.3 depicts the construction of an utterance model from phoneme models and the phoneme string. During training, the model assigns certain training vectors to appropriate states, and accumulates sufficient statistics for estimating the parameters \mathcal{M} . The accumulation is performed over all the utterance/phoneme string pairs in the training set in the expectation step of the EM algorithm.

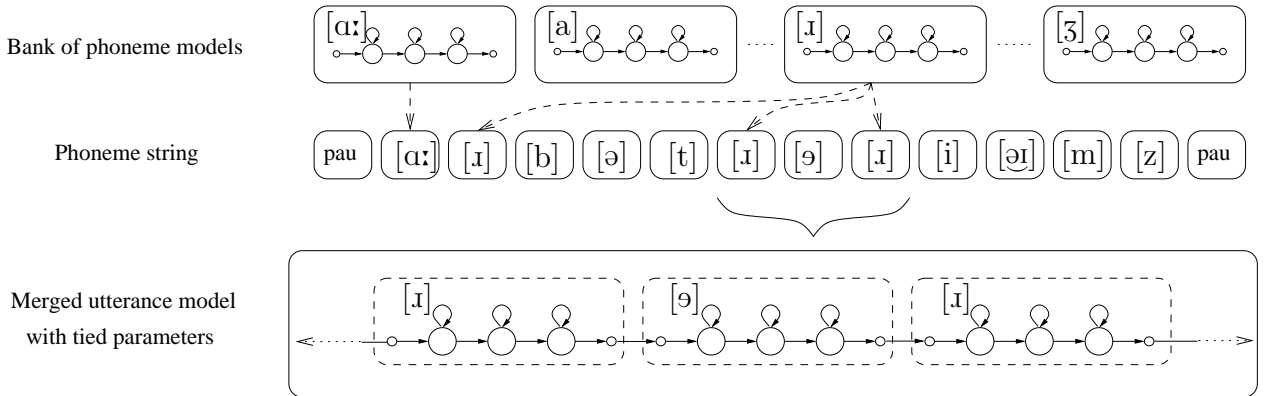


Figure A.3: An utterance model for embedded re-estimation or forced alignment. The phoneme models are strung up into one big model by connecting their initial and final null-states. The parameters for each phoneme model is still estimated *globally* so that all the examples of each phoneme contribute to one model.

Since the temporal structure of the utterance model is very well constrained, the phoneme models that fit their phonemes well, form boundaries within which the less well-defined models can “find their place”. If the training set has examples of each phoneme in many different contexts, the phoneme models should, in principle, adopt the similar parts it is shown at each iteration, and therefore the training should converge. If this happens, the phoneme models will represent what the designer thinks they do.

One of the strengths of the HMM is the ability to entertain multiple hypotheses as weighted branches of the finite-state network. Given a lexicon with a number of different pronunciations for lexical items, they can all be incorporated by introducing parallel sets of states in the utterance model [38]. The HMM should assign greater importance to the branch that fits the utterance better, and thus still train the correct phoneme models for the underlying acoustics.

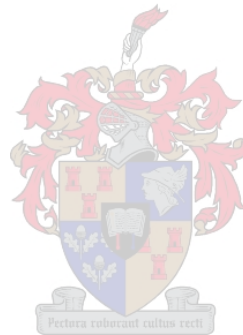
Good initialisation of the models is very important when performing embedded re-estimation. At least a small amount of labelled data is necessary to bootstrap the process [114].

A.5 Forced Alignment

Given a single speaker database, and well-trained speaker-independent phoneme models, embedded re-estimation with the above-mentioned simplified MAP procedure provides a

way to adapt the phoneme models to the speaker. We can now construct an utterance model for each utterance that must be aligned with its known phoneme string, and find the optimal state sequence using a Viterbi search. The optimal state sequence gives the time alignment of the phoneme string. An added advantage is that the procedure can find the correct one of several hypothesised pronunciations.

Forced alignment can be very robust. In cases where the phoneme string does not match the pronunciation exactly, the ill-matching phonemes can still be bounded by neighbours that match the acoustics better. In the HRS speech synthesis corpora for instance, the speakers spoke very fast in de-emphasised parts of the utterances. Clearly pronounced parts were aligned perfectly in spite of this. As the goal of the alignment is limited-domain synthesis in a slot-and-filler NLG system, the clearly pronounced bits are the important ones.



A.6 Symbols

This section summarises the symbols used in this Appendix. The symbols have been chosen to conform as much as possible with those of Deller et al. [1], and therefore conflict with the symbols used in the rest of this work.

α, β	The forward and backward probabilities in the Baum-Welch algorithm.
a_{ij}	The probability of the HMM to transition from state i to state j .
A	The transition probability matrix that contains the a_{ij} 's.
\mathbf{C}	Covariance matrix.
D	Dimensionality of the observation vectors.
$f_{\mathbf{y} x}(\mathbf{y} x = 1)$	PDF of \mathbf{y} given that the present state is $x = 1$.
M	The number of components in the Gaussian mixtures used to model the observation probabilities.
N	The number of states in the HMM.
$\mathcal{N}(\mathbf{y}, \boldsymbol{\mu}, \mathbf{C})$	Gaussian PDF of the random vector \mathbf{y} .
$\boldsymbol{\pi}(t)$	The vector of state occupancy probabilities at time t .
\mathcal{P}	Indicates a quantity associated with the prior estimates in the simplified MAP scheme.
S	The number of states in the HMM.
t	The discrete time index.
T	The length of the observation Y and state X sequences.
$\boldsymbol{\mu}_{im}$	The mean vector of the Gaussian component m of state i .
w_{im}	The weight of mixture component m at state i .
x, x_i	The state number of the HMM, and the state number at time-step i .
\mathbf{y}, \mathbf{y}_i	The emitted observation vector, and the observation at time-step i .
Y	The sequence of observed output vectors \mathbf{y}_i of length T .
Y_n^m	The sequence of observed output vectors \mathbf{y}_i for $\{n, \leq i \leq m\}$.