Time-Domain Concatenative Text-to-Speech Synthesis

Daniel Samuel Gordon Vine

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

September 1998

Bournemouth University
in collaboration with DevR

Abstract

A concatenation framework for time-domain concatenative speech synthesis (TDCSS) is presented and evaluated. In this framework, speech segments are extracted from CV, VC, CVC and CC waveforms, and abutted. Speech rhythm is controlled via a single duration parameter, which specifies the *initial portion* of each stored waveform to be output. An appropriate choice of segmental durations reduces spectral discontinuity problems at points of concatenation, thus reducing reliance upon smoothing procedures.

For text-to-speech considerations, a segmental timing system is described, which predicts segmental durations at the word level, using a timing database and a pattern-matching look-up algorithm. The timing database contains segmented words with associated duration values, and is specific to an actual inventory of concatenative units. Segmental duration prediction accuracy improves as the timing database size increases. The problem of incomplete timing data has been addressed by using 'default duration' entries in the database, which are created by re-categorising existing timing data according to articulation manner. If segmental duration data are incomplete, a default duration procedure automatically categorises the missing speech segments according to segment class. The look-up algorithm then searches the timing database for duration data corresponding to these re-categorised segments.

The timing database is constructed using an *iterative synthesis/adjustment* technique, in which a 'judge' listens to synthetic speech and adjusts segmental durations to improve naturalness. This manual technique for constructing the timing database has been evaluated. Since the timing data is linked to an *expert judge's* perception, an investigation examined whether the expert judge's perception of speech naturalness is representative of people in general. Listening experiments revealed marked similarities between an expert judge's perception of naturalness and that of the experimental subjects. It was also found that the expert judge's perception remains stable over time. A synthesis/adjustment experiment found a positive linear correlation between segmental durations chosen by an experienced expert judge and duration values chosen by subjects acting as expert judges.

A listening test confirmed that between 70% and 100% intelligibility can be achieved with words synthesised using TDCSS. In a further test, a TDCSS synthesiser was compared with five well-known text-to-speech synthesisers, and was ranked fifth most natural out of six. An alternative concatenation framework (TDCSS2) was also evaluated, in which duration parameters specify both the start point and the end point of the speech to be extracted from a stored waveform and concatenated. In a similar listening experiment, TDCSS2 stimuli were compared with five well-known text-to-speech synthesisers, and were ranked fifth most natural out of six.

Contents

1. Introduction 10

1.1 This thesis 10
1.2 Background Information 13
1.2.1 Speech Production 14
1.2.2 Speech Synthesis methods 17
1.2.3 Text-to-Speech 31
1.3 Limitations of This Research 44

2. Speech Synthesis System Components 46

2.1 Introduction 46
2.2 The Text-to-Segment Conversion Module 47
2.3 The Timing Module 48
2.3.1 The Timing Database 48
2.4 The Synthesis Module 49
2.5 Summary 50

3. A Concatenation Framework for Speech Synthesis 51

3.1 Introduction 51
3.2 Time-Domain Waveform Concatenation Framework 53
3.2.1 Choice of concatenative units 54
3.2.2 Obtaining the concatenative units 55
3.2.3 Segment boundaries 56
3.2.4 Formation of CVC syllables 58
3.2.5 Concatenation nodes 59
3.2.6 Phonemic category shifts 60
3.3 The Text-to-Segment Conversion Module 60
3.4 An alternative concatenation framework 62
3.4.1 Advantages of the two-parameter concatenation framework 64
3.5 Summary 65

4. Segmental Duration Prediction for Time-Domain Concatenative Speech Synthesis 66

4.1 Introduction 66
4.2 Segmental Duration Prediction 68
4.3 Duration Look-up Rules 69
4.3.1 Rule 1: Largest possible sub-string 69
4.3.2 Rule 2: Position in word 70
4.3.3 Rule 3: Length of database entry 70
4.4 Default Durations 71
4.5 Segmental duration prediction for TDCSS2 72
4.6 Summary 73

5. Timing Databases 74

5.1 Introduction 74
5.2 Timing Database Creation 77
5.3 Timing Database Content 80

5.3.1 Timing Database Size Issues 80
5.3.2 Using Default Duration Entries in the Timing Database 82
5.4 Evaluation of naturalness in temporally modified words 85
5.4.1 Experiment Series Overview 85
5.4.2 Experiment 1 (randomised for each word) 89
5.4.3 Experiment 2 (fully randomised) 91
5.4.4 Experiment 3 (expert judge as subject) 92
5.4.5 Results 93
5.4.6 Conclusion 97
5.5 A comparison of segmental durations produced by different expert judges .. 99
5.5.1 Subjects 100
5.5.2 Method 100
5.5.3 Instructions 101
5.5.4 Results 102
5.5.5 Discussion 105
5.6 Support tools for timing database development 106
5.7 Summary 107

6. Evaluation of Time-Domain Concatenation Speech Synthesis 109

6.1 Introduction 109
6.1.1 Segmental Intelligibility Tests 110
6.1.2 Semantically Unpredictable Sentences test 111
6.1.3 Paired Comparison tests 111
6.2 Chapter Overview 112
6.3 Evaluating Intelligibility 112
6.3.1 Stimuli 113
6.3.2 Subjects 113
6.3.3 Method 113
6.3.4 Results 113
6.4 Evaluating Naturalness: Investigating the One-Parameter Concatenation Framework (TDCSS) 114
6.4.1 Stimuli 114
6.4.2 Subjects 116
6.4.3 Method 116
6.4.4 Results 117
6.4.5 Discussion 118
6.5 An Evaluation of the Two-Parameter Concatenation Framework (TDCSS2) 119
6.5.1 Stimuli 119
6.5.2 Subjects 120
6.5.3 Method 120
6.5.4 Results 120
6.6 Summary 121

7. Conclusions 123

7.1 Future Work 126

8. Appendix A - Naming Conventions 128

9. Appendix B - Duration Look-Up Algorithm 130

10. Appendix C - Default Duration Procedure 132

11. Appendix D - Listening Experiment Results 133

12. Appendix E - Timing Database Contents 142

13. Appendix F - Timing Module Code 149

14. Appendix G - Letter-To-Sound Rules 175

15. References 190

Accompanying Material

During the course of this research, several papers were presented at various refereed conferences, and one paper was submitted to a journal. These are shown below, and a selection of these is included at the back of the thesis.

Sahandi, M.R. & Vine, D.S.G. (1996). A Genetic Algorithm Approach for obtaining Diphone Durations for English Speech Synthesis. *In: DSP Scandinavia 96, Copenhagen, Denmark, 18-19 June 1996.*

Vine, D.S.G. & Sahandi, M.R. (1996). Optimisation of segmental time durations in concatenative speech synthesis. *In: KT96 International Conference on Knowledge Transfer, London, 22-26 July 1996.*

Sahandi, R. & Vine, D.S.G. (1997a). A Concatenation Framework for Text-to-Speech Synthesis using CVC Syllables. *In: SALT Workshop on Evaluation in Speech and Language Technology, University of Sheffield, UK, 17-18 June 1997.* 172-179.

Sahandi, R. & Vine, D.S.G. (1997b). Text-to-Speech Animation for the Hearing Impaired. *In: KT97 Int. Conf. on Knowledge Transfer, London, 15-16 July 1997.*

Vine, D.S.G. & Sahandi, M.R. (1997). Timing Databases for Concatenative Speech Synthesis. *In: DSP Scandinavia 97, Stockholm, Sweden, 3-4 June 1997.* 21-24.

The following papers have also been accepted for publication:

Vine, D.S.G. & Sahandi, M.R. (accepted). An Evaluation of Naturalness in Time-Domain Concatenative Speech Synthesis. *In: Proc. Inst. of Acoustics.*

Sahandi, R., Vine, D.S.G. & Longster, J. (accepted). Text-to-Visual Speech Synthesis. *Informatica.*

Acknowledgements

This research was funded by Development Research (DevR) and Bournemouth University. Except where stated, the work was carried out unaided.

I would like to thank my supervisor, Dr. Reza Sahandi, for his constant support and guidance throughout this study, and for giving me the freedom to follow my research interests. I would also like to thank my secondary supervisors, Professor Peter Comninos and Ron Burns, for their advice. Thanks also to Professor Martin Shepperd for objective advice and encouragement.

I received some invaluable guidance regarding experimental design and statistical methods from Dr. Martin Lefley, Dr. David Heathcote and Dr. Gerry Griffin of Bournemouth University. Prof. Dan Diaper helped considerably with the design of the 2AFC listening experiments in Section 5.4. I was privileged to discuss my work with the late Christian Benoît, and also with Andy Breen, Denis Johnston and Mike Edgington from BT Laboratories. I would also wish to thank Paul Taylor of Edinburgh University and Andy Breen for reviewing a research report. I am also grateful to Mark Huckvale of University College London for the use of his Speech Filing System software.

I would like to thank all those who have worked with me in my office and in the SPECTA (SPEech & Character Animation) research laboratory at Bournemouth University, including Paul Rogers, Peter Barker, Chris Cowley, Gelerah Roushan, Colin Kirsopp, Jenny Longster, Jon Macey, Tom Kinsella and Abdelati Mohammed. Thanks also to fellow researchers Austen Rainer, Glyn Hadley, Emma Nesbitt, Tom Teng, Min Park and Simon Crowle. I am indebted to those who volunteered to participate in the various experiments. I am also grateful to Ali Adams for compiling the code for the text-to-phoneme conversion program and to Ron Burns for getting DECtalk working. Many thanks to my sister Anna for double-checking some of the statistics. Thanks also to Phil Keaggy and Ron Kneusel.

I must thank my friends and family for their support, including my parents, grandparents, Robert, Viv, Douglas, Lynette, Lucy, Anna, Tamsin, Louise, The Hills, Mike and Micky. Finally, I would also like to thank Jesus Christ for making it all possible.

Chapter One

1. Introduction

1.1 This thesis

In speech synthesis research, considerable attention is paid to improving models of prosody. However, other aspects of speech synthesis are often given less emphasis. For instance, in concatenative speech synthesis, problems still arise due to unnatural discontinuities when conjoining speech segments (Hess, 1992). These discontinuities are detrimental to segmental intelligibility and cannot be hidden by a good prosodic model (Carlson, 1991). This research presents and evaluates a concatenation framework for text-to-speech synthesis, which is potentially a means of improving the naturalness and intelligibility of synthetic speech; however the framework is at present constrained to the synthesis of isolated words from a text input.

In this concatenation framework, speech segments are concatenated in the time domain to produce synthetic speech (see Chapter 3). In the concatenation process, speech segments are abutted without the use of explicit smoothing procedures; a key feature of this method is that the point of concatenation itself can be varied from segment to segment. By choosing an optimal concatenation point, the naturalness of the synthesised speech is greatly improved. Concatenation points are directly related to a segmental duration parameter, which specifies the *initial portion* of a speech unit that is to be abutted. There is a direct relationship between the extracted portion of a speech unit and the segmental duration parameter, because the selected portion has an intrinsic duration. This time-domain concatenation technique has not been explored extensively in the literature, and so this research seeks to evaluate its potential.

The concatenation framework is based primarily on CVC (consonant-vowel-consonant) syllables, formed by joining together CV (consonant-vowel) and VC (vowel-consonant) syllables. Some CC (consonant-consonant) diphones are also used. The development of the inventory of speech waveforms is described. The concatenation framework is not limited to the above types of concatenative unit, and additional types of segment can subsequently be added to the waveform inventory. It

is shown that under this framework, intelligible synthetic speech is attainable for whole words.

For text-to-speech applications, a segmental duration prediction algorithm has been developed (see Chapter 4), which uses a timing database and a pattern-matching look-up algorithm. The timing database consists of segmented words with their corresponding duration values and is constructed using an *iterative synthesis/adjustment* approach, in which a human listens to the synthetic speech and adjusts segmental durations to improve naturalness. This manual approach was chosen because it is difficult to judge one component of a text-to-speech synthesiser independently of the others (Carlson, Granström & Klatt, 1979). By using a human listener, it is possible to make a more reliable assessment of the final synthetic speech signal, and the effect of altering a particular parameter, such as segmental duration. In this way, segmental duration data are produced that are directly linked to a specific waveform inventory; it is also a perceptual approach, since a human listener is involved in creating the timing database. Throughout this thesis, this person is referred to as an *expert judge*.

In this study, segmental duration data are initially obtained on a manual basis, using an expert judge. The timing data acquired are thus based on a particular expert's judgement of synthetic speech naturalness, and differ in nature to conventional timing data, which are based on labelled recordings of natural speech (e.g., Carlson & Granström, 1986). This raises several questions as to whether the expert's judgement is stable over a period of time, and whether it is representative of other people's perceptions. These issues are examined through a set of listening experiments and a synthesis/adjustment experiment, which are described fully in Chapter 5.

In the above listening experiments, subjects listened to several versions of a synthetic word, and ranked them in order of perceived naturalness (the different versions of the word contained small segmental duration modifications). The subjects' preferences were then compared with those of an expert judge.

In the synthesis/adjustment experiment, subjects synthesised a given set of words, manipulating the segmental durations according to their personal criteria for

naturalness. A positive linear correlation was found between these segmental durations and those pre-chosen by an expert judge.

In a listening test, a time-domain concatenative speech synthesiser (TDCSS) was compared with five well-known text-to-speech synthesisers, and was ranked fifth most natural out of six. An alternative concatenation framework (TDCSS2) was also evaluated, in which duration parameters specify both the start point and the end point of the speech to be extracted from a stored waveform and concatenated (see Section 3.4). In a similar listening experiment, TDCSS2 stimuli were compared with five well-known text-to-speech synthesisers, and were also ranked fifth most natural out of six. The implications of these results are discussed in Chapter 6.

In studying TTS synthesis, researchers continue to use annotated speech recordings as a starting point for building segmental timing and pitch prediction systems. However, until both timing and pitch prediction systems are perfected, there will remain the problem of interaction between the various components of the TTS system. For example, listeners may still perceive correctly predicted segmental durations as unnatural if an inappropriate pitch contour is used.

A clear implication is that annotated speech recordings may not be the most appropriate prosodic databases from which to formulate prediction systems for segmental durations or pitch. It may be that prosodic databases (for both pitch and segmental duration) should somehow account for human perception of synthetic speech based on that data. For this reason, the iterative synthesis/adjustment method has been investigated in this research. It is anticipated that similar practices may become more popular in an attempt to overcome the problems of interactions between TTS system components. For instance, the IPO research into pitch prediction ('t Hart, Collier & Cohen, 1990) makes use of 'perceptual stylisation', in which subjects synthesise pitch contours which they judge to be perceptually equivalent to a given pitch contour. An additional trend may be that *for each new speaker* added to concatenative speech synthesisers, a new prosodic database may have to be created, specifically for that speaker. Unfortunately for TTS research, this is a time-intensive process.

It is concluded that time-domain concatenative speech synthesis methods are viable for producing isolated words, and can be incorporated into a text-to-speech framework on the word level (for sentence-level speech synthesis, a mechanism for pitch manipulation would need to be introduced). Evidence is also shown that the iterative synthesis/adjustment technique is a valid means of obtaining the necessary timing data for segmental duration prediction.

1.2 Background Information

It has long been a goal of humans to produce talking machines. Speech is a fundamental means of communication between humans, and is even linked to one's personal identity (Crystal, 1987). Synthetic speech is viewed as a way of making the interaction between humans and machines more natural, and so has been the subject of much research. Many years before the advent of electronic computers, speech synthesis was attempted using mechanical means. The first talking machine was invented by Wolfgang von Kempelen (1734-1804). This used a set of bellows to produce a flow of air which vibrated a wooden reed, thus providing a voicing source (Dudley & Tarnoczy, 1950). The air passed through a leather tube, which acted as the vocal tract. A skilled operator operated the bellows with the left hand, whilst altering the vocal tract's shape with the other (Crystal, 1987). This machine could produce approximately twenty speech sounds. Alexander Graham Bell (1847-1922) also constructed a synthetic 'talking head' that could produce a variety of speech sounds (Crystal, 1987).

The first electronic speech synthesiser was demonstrated at the New York world fair in 1939; Dudley's 'Voder' (VOice DEmonstratoR) aroused considerable public interest at the time (Breen, 1992a). The Voder's speech output was controlled via a keyboard and pitch-control pedal, and required a human operator (Dudley, Riesz & Watkins, 1939). The Voder produced the speech signal electronically by modifying an excitation source with a bank of ten filters, which resonate at fixed frequencies. (N.B. A *filter* is a device used to separate the frequency components of a sound wave (Crystal, 1987)). In the above technique, the excitation signal (which is analogous to sound produced by the vocal cords) is treated separately to the action of the vocal

tract. This is known as the *source-filter* theory (see Section 1.2.1.1), and has been incorporated widely into speech synthesis research (Fant, 1960).

After Dudley's Voder, there were several milestones in the development of electronic speech synthesis (e.g., Kelly & Gerstman, 1961). One of the first speech synthesis systems was due to Holmes, Mattingly and Shearme (1964). This required the user to input the pronunciation for each word, and a set of stress markers. The system was later improved to automatically assign a pitch contour to utterances. Coker, Umeda and Browman (1973) developed a comprehensive text-to-speech synthesiser, which was able to predict a plausible timing and pitch structure for the output speech. Dennis Klatt developed the MITtalk speech synthesiser (Allen, Hunnicutt & Klatt, 1987), which later became DECtalk. This synthesiser produces highly intelligible speech, and still remains a benchmark against which other synthesisers are compared.

Subsequent speech synthesis techniques (Moulines & Charpentier, 1990) have made a greater level of naturalness possible. However, an ongoing challenge in synthetic speech research is to achieve such naturalness using sets of rules. This is particularly necessary in the case where speech is synthesised with only unrestricted text as an input (see Section 1.2.3). The various speech synthesis techniques are described later, in Section 1.2.2. Much of the research into speech synthesis is influenced by the human mechanism for speech production, and so this is described in the following section.

1.2.1 Speech Production

Humans produce sounds by expelling air from the lungs. This pulmonic egressive airstream (Breen, 1992a) passes through the larynx, which consists of interlocking sets of cartilage (see Figure 1.1). Between the thyroid and cricoid cartilage are the vocal folds; these are fleshy membranes which are held apart during normal breathing. Voiced speech (phonation) is produced by a rapid vibration of the vocal folds. The tension of the vocal folds is altered by tilting the thyroid cartilage; this alters the fundamental period of vocal fold oscillation, and hence, the fundamental frequency of the speech. The fundamental frequency (f0) of voiced speech is

influenced by the pressure difference across the vocal cords; this consequently affects the air flow rate through them (Rowden, 1992). A faster rate of air flow results in a rise in f0.

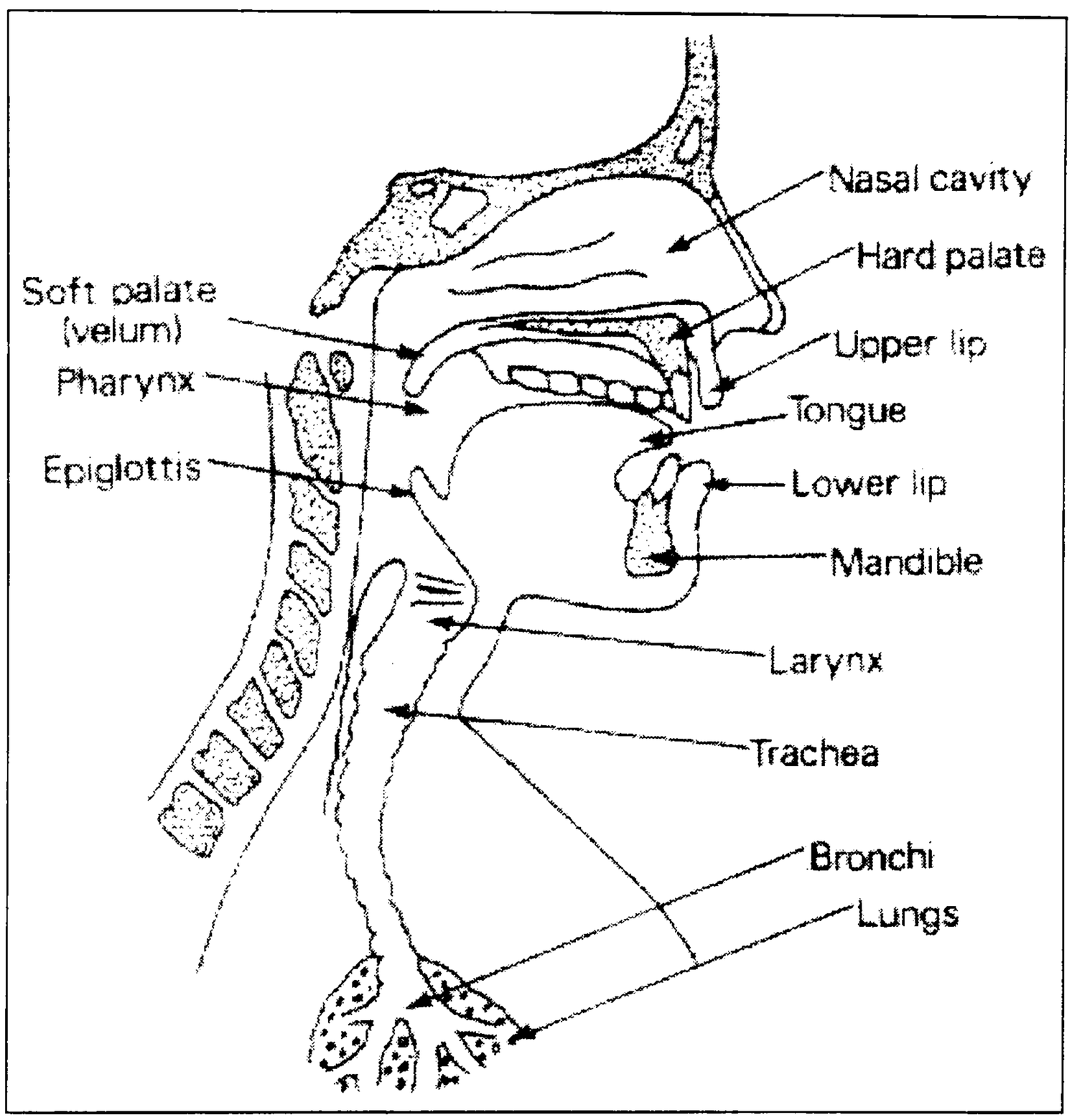


Figure 1.1: Cross-section through the head - based on Holmes (1988)

The field of *articulatory phonetics* examines the linguistically important sounds of a language in terms of movements of the articulators (i.e., the lips, jaw, tongue and velum). In articulatory phonetics, several parameters are considered to be important. For consonants, the chief variables are *voice, manner* and *place* (Breen, 1992a). The 'voice' variable refers to whether a sound is *voiced* or *unvoiced.* Voiced sounds are produced by the larynx, whilst unvoiced sounds are influenced by constrictions in the vocal tract. *Place* of articulation refers to the points of constriction in the vocal tract, such as the glottis, palate, gums, teeth and lips (Yost, 1994). *Manner* of articulation describes the way in which the sound is produced, i.e., how the air pressure from the voice mechanisms is altered to produce sounds. The main manners of articulation

include *plosive*, *nasal* and *fricative*. Plosives are created by a sudden release of air pressure built up as a result of blocking the vocal tract. Fricatives result from a turbulent flow of air from the mouth. Nasals are produced by using the nasal cavity as a place of articulation (Yost, 1994). The main parameters for producing vowels are *front/back*, *tongue shape* and *lip rounding* (i.e., whether the lips are spread or rounded).

Some of the above aspects of human speech production influence speech synthesis techniques directly. The most widespread example of this is the source-filter model of speech.

1.2.1.1 Source-Filter model of speech

Many speech synthesisers are based upon the source-filter model of speech production (Fant, 1960). The source-filter model (see Figure 1.2) views the speech as being produced via an excitation source (or *glottal source*) $g(t)$, representing the airflow passing the vocal cords (Veldhuis, 1998; Witten, 1986). The vocal tract response is represented by a filter with a transfer function $H(jw)$. The excitation source consists of white noise, a periodic pulse or a mixture of both. A radiation characteristic R models the conversion of the airflow to a pressure wave $s(t)$ at the lips (Veldhuis, 1998).

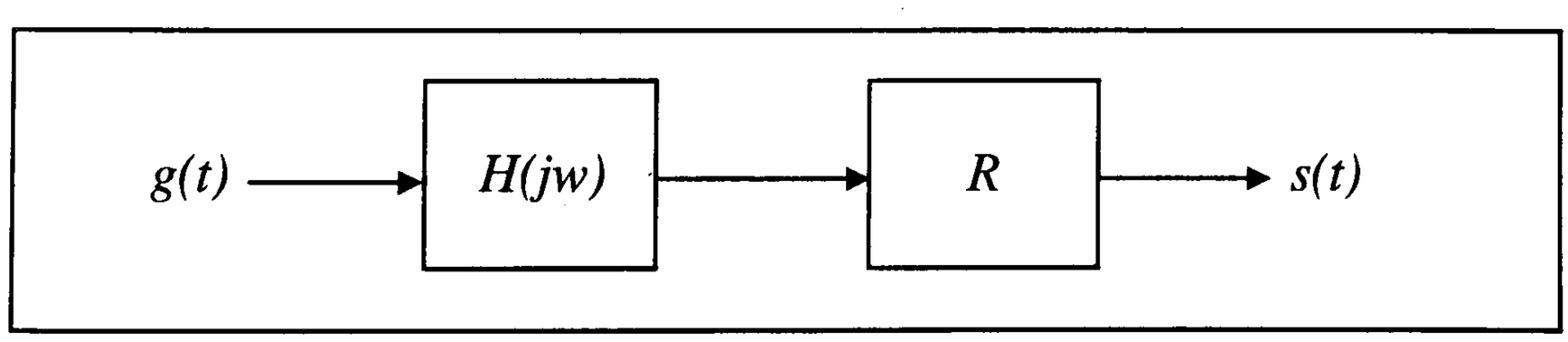


Figure 1.2: Source-filter model showing glottal source, vocal tract filter and radiation characteristic respectively

The assumption that the source and filter are independent is only an approximation of reality. In practice, the human vocal tract can produce a combination of voiced source and noise source. However, in many speech synthesisers, this mixture is often not implemented (Breen, 1992a). The reliability of the source-filter independence

assumption is further diminished by the fact that in female voices, the vocal cords remain open for longer during each pitch period, and, in some cases, this results in an interaction between the trachea and the vocal tract.

The choice of glottal source is related to the quality of synthetic speech (Veldhuis, 1998). Indeed, the use of simplified glottal sources has been responsible for a lack of naturalness in synthetic speech (Carlson, 1991), often giving rise to the 'buzziness' that characterises some speech synthesisers (Dutoit, 1997). This has led to the creation of complicated voicing models, which enable many of these features to be varied (Klatt, 1987; Klatt & Klatt, 1990). However, substantial improvements are necessary in the rules that control these models (Klatt, 1987).

1.2.2 Speech Synthesis methods

Synthetic speech can be produced using either parametric or concatenative methods. The most common parametric method is known as *formant synthesis*, in which the speech signal is described in terms of its formant frequencies, antiformants, bandwidths, etc. (Dutoit, 1997). Detailed discussion of this method is given in Section 1.2.2.2. Another parametric method is *articulatory synthesis*, in which speech is described in terms of movements of the articulators (see Section 1.2.2.1). At present, this approach is generally considered to be less successful than formant synthesis.

In *concatenative synthesis*, speech is produced by joining together segments of pre-recorded speech. In this approach, it is important that the acoustic speech segments are conjoined in such a way that they will be perceived as natural; they must also be timed so that the listener will perceive a correct rhythm. Detailed discussion of this method may be found in Section 1.2.2.3.

1.2.2.1 Articulatory Synthesis

In articulatory speech synthesis, speech is described in terms of movements of the speech articulators. This method also requires a means of representing the shape of the vocal tract. The vocal tract is often modelled in terms of a series of shapes of

varying cross-sectional area. Since the shape of the vocal tract changes slowly with time, the area function only requires updating at intervals of between 20 and 50 ms (Linggard, 1985).

The vocal tract can also be viewed as a system of two tubes, which represent the oral and nasal cavities. One of the tubes corresponds to the nasal side branch at the top of the pharynx. The nasal tract is of fixed cross-sectional area and shape, whilst the oral tract can vary in shape, as the articulators are moved. The most important variable that must be predicted is, therefore, the oral tract's cross-sectional area (which is assumed to be fixed during a single analysis frame).

In vocal tract modelling, the analogy of an electrical transmission line is sometimes used (Linggard, 1985), with variations occurring in inductance and capacitance per unit length. Air flow is modelled as electric current, whilst the storage of air in the lungs is represented as capacitance. Acoustic energy losses via the lips and nostrils (the radiation characteristic) can be modelled as a resistive load (Linggard, 1985). Unfortunately, it is difficult to vary real inductors and capacitors with sufficient speed to simulate articulatory movements.

The appeal of articulatory synthesis is that, in theory, only a few control parameters are necessary to produce human-quality speech. Since movements of the articulators are relatively slow, articulatory parameters potentially allow low bit-rate speech coding (Rahim *et al.*, 1993). Furthermore, *coarticulation* effects should automatically be produced as the articulators are moved (coarticulation refers to changes in the articulation of a segment of speech due to preceding and forthcoming segments) (Cohen & Massaro, 1993). This is attractive because coarticulation is difficult to control as a set of rules in formant synthesis (Allen, Hunnicutt & Klatt, 1987).

At this point, it is necessary to define some commonly used terms. A *phoneme* is the smallest unit in the sound system of a language where substitution of one unit for another might make a distinction in meaning (e.g., *do* versus *to*). An *allophone* is a variation in the form of a phoneme that does not alter its basic identity (Crystal, 1987).

One problem that must be considered is that there is not a simple correspondence between articulatory positions and individual speech sounds such as allophones. In other words, no one-to-one mapping exists (Rahim *et al.*, 1993). However, this need not be a problem if one starts by studying articulatory gestures and their resulting sounds (Tatham, 1970).

Accurate metrical data for the articulators are rather difficult to obtain (Tatham, Morton & Mansell, 1971). The traditional method is to use X-ray photography (O'Shaughnessy, 1987). However, it is easier to analyse speech sounds than to study movements of the vocal organs (Witten, 1982). There is therefore a need for ways of automatically extracting articulatory parameters from a speech signal (Rahim *et al.*, 1993). This was attempted by Levinson, Coker and Schmidt (1981), who derived area functions from real speech by matching the parameters of synthetic speech to those of a recorded utterance.

Whilst articulatory synthesis has not yet achieved a great level of commercial success, it remains a valuable tool for studying articulatory dynamics. It also has the potential to incorporate the interaction between the source and filter (Rahim *et al.*, 1993), which is discarded by the source-filter model's assumption that they are independent.

1.2.2.2 Formant Speech Synthesis

In formant speech synthesis, the speech signal is described in terms of a set of parameters that vary with time. During natural speech, resonances are produced in the vocal tract; these resonances produce peaks in the energy spectrum of the speech wave, known as *formants* (Witten, 1982). The use of rules to generate formant synthesiser control parameters is sometimes referred to as 'speech synthesis by rule'; however, this term is somewhat misleading, since waveform concatenation speech synthesisers can also be controlled by rule. In formant synthesis, speech is synthesised by attempting to mimic the formant frequencies of natural speech as they vary with time. Up to three formants are generally required to synthesise intelligible speech, with four or five being sufficient to produce high quality speech. Formants can be physically produced using a set of filters, which resonate at variable frequencies.

There are two types of formant synthesiser - *parallel* and *cascade* (see Figure 1.3). With the parallel architecture, an excitation signal is applied to all the formants in parallel and their outputs are summed, enabling individual gains to be specified for each formant. In cascade formant synthesisers, the output of one formant is applied to the input of the next. Whilst both kinds have their drawbacks, it is generally found that cascade synthesisers produce better non-nasal voiced sounds, whilst the parallel type produces superior nasals, fricatives and stops (Linggard, 1985). Excitation sources have been proposed in which two possible source functions are used: a pulse train for producing voiced speech, and a random noise source for fricatives (Klatt, 1972). The excitation source is often implemented using a -12dB/octave filter. An additional +6dB/octave filter is often placed after the vocal tract filter to approximate the radiation characteristic.

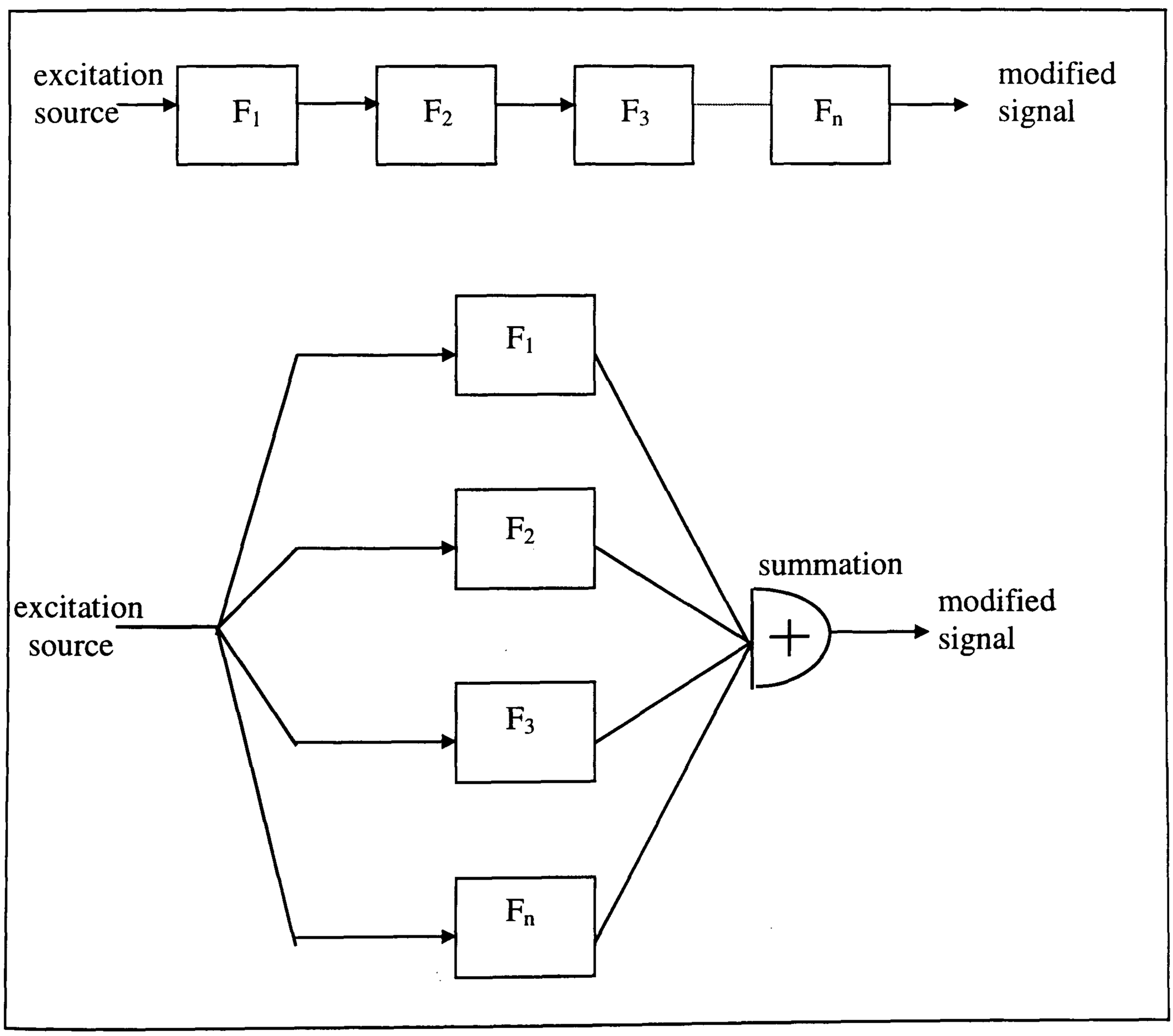


Figure 1.3: Speech synthesis architectures: cascade (top) and parallel (bottom)

The control signals that drive a formant synthesiser can be obtained either by analysing speech or using sets of rules. In synthesis-by-analysis, often called 'copy synthesis', sets of synthesiser control signal values are derived through an intensive analysis of the original speech (Breen, 1992a). Synthetic speech produced in this manner can be of a very high quality, largely due to the fact that very few assumptions are made about the relative importance of particular acoustic events to the naturalness of speech. In effect the only assumptions used are those made during the design of the synthesiser.

An important demonstration of the capabilities of formant synthesisers was reported by Holmes (1973). In a listening experiment, people unaccustomed to synthetic speech were unable to distinguish between natural and synthetic versions of the same utterance, even when using earphones. The synthetic utterance was generated by manually tuning the control parameters of a parallel formant synthesiser to reproduce the natural utterance. Whilst Holmes' results were encouraging, a major challenge still remained in producing natural-sounding speech by rule. In fact, the success of rule-based formant synthesis still largely depends on how well the influence of phonemes on one another is described (Dutoit, 1997).

Efforts to improve formant synthesisers have been made by combining both the parallel and cascade synthesiser arrangements. Klatt's (1980) synthesis-by-rule system was one of the most sophisticated, using a complex excitation model and relying on by 39 parameters, which were updated every 5 ms. Additional parameters such as resonances and anti-resonances assisted in the synthesis of nasals. A sixth formant enabled the production of very high frequency noise, whilst a further filter modelled the radiation characteristic. The synthesiser produced high quality speech, and has since been incorporated into several commercial systems, and used by many researchers in their work.

In formant synthesis, a general approach is to use a look-up table containing the parameters necessary to specify each phoneme. An interpolation method is also required, to calculate the transitions between phonemes, in order to improve the naturalness of the resulting speech. Various interpolation methods have been considered for speech synthesis, including piecewise linear (e.g., Holmes, Mattingly

& Shearme, 1964), decaying exponentials (Rabiner, 1969), and increasing exponentials (Lawrence, 1974). Holmes, Mattingly & Shearme (1964) used phoneme look-up tables, containing information concerning formant frequencies, formant amplitudes, fixed and transitional data and a 'transition-rank'. The transition rank determined which of two adjacent phonemes would dominate a mutual transition, and the extent to which they would overlap. Stops (such as /p/ and /t/ phonemes) were given the highest rank, and vowels the lowest. At the boundary between two phonemes, a 'boundary value' was calculated for each control parameter, using specified fractions of the parameter values on both sides of the transition, and taking into account their transition-ranks. Linear interpolation was used to join this boundary value across the transition regions of the two phonemes.

Klatt's (1980) approach was to use target values of control parameters and an interpolation algorithm. However, the target values for phonemes were not all obtained from tables; some were calculated by rules from data concerning place, context and articulation manner. Also, the time constants of the parameter transitions were not stored, but derived from rules on contextual information.

Formant synthesis can be improved by the painstaking examination of continuous speech, parameterised using speech analysers. When speech is parameterised, analysis biases can occur if the analysis algorithm does not yield the best possible parameters in the first place (Dutoit, 1997). These biases are known as *extrinsic errors. Intrinsic errors* also arise due to inaccuracies in the mathematical equations used to describe speech in the synthesis algorithm. This 'trial and error' method of producing sets of rules can be a labour-intensive process, because formant frequencies and bandwidths are inherently difficult to estimate from speech data (Dutoit, 1997).

1.2.2.3 Concatenative Synthesis

In concatenative synthesis, speech segments are joined together (*concatenated*) to form the required acoustic signal. The segments are speech samples taken from an inventory of pre-recorded natural speech data (Hess, 1992). Segments of speech can then be concatenated to form arbitrary words and sentences, given the correct

pronunciation. A major advantage of this method is that it is easier to produce realistic coarticulation effects, if suitable speech segments are chosen. It is also appealing in terms of its simplicity, in that all knowledge concerning the synthetic message is inherent to the speech segments to be concatenated (Dutoit, 1997). Thus, little attention needs to be paid to the modelling of articulatory movements.

Various digital signal processing techniques are employed for smoothing boundaries between speech segments and producing a pitch contour. Two main categories of signal processing are used in concatenative systems: Linear Prediction and Pitch-Synchronous Overlap-Add (PSOLA). These are described in Sections 1.2.2.6 and 1.2.2.7 respectively.

The selection of speech segments for concatenation is in itself an important consideration which affects both the quality of the synthetic speech and the storage requirements. Such decisions often involve trade-offs between storage requirements, the extent to which signal processing is required, and so on. The selection criteria are discussed in the following section.

1.2.2.4 Concatenative Unit Selection

Various speech segments, of differing sizes, have been tried in concatenative speech synthesis. Initially, *phonemes* were favoured by researchers (Witten, 1982), because the small number of units required (approximately forty for the English language) kept storage requirements to a minimum. However, this approach requires a great deal of attention to be paid to coarticulation at the boundaries between phonemes (Witten, 1982). Consequently, synthesis using phonemes requires the formulation of complex coarticulation rules (Linggard, 1985).

Coarticulation problems can be minimised by a careful choice of concatenative unit. A popular unit is the *diphone*, which consists of the transition from the centre of one phoneme to the centre of the following one (Dixon & Maxey, 1968). This helps to capture transitional information between phonemes. For instance, the word 'cat' may be represented using the diphones /silence_k/, /k{/, /{t/ and /t_silence/ (here, the

/t_silence/ segment is necessary to represent the egression of air as the /t/ stop is released). A complete set of diphones would number approximately 1600, since there are $(40)^2$ possible combinations of phoneme pairs. Diphone speech synthesis thus requires only a moderate amount of storage. The main disadvantage of diphones is that they lead to a high density of concatenation points (one per phoneme), so that heavy reliance is placed upon an efficient smoothing algorithm.

Syllables have the advantage that most coarticulation occurs within syllable boundaries (Fujimura & Lovins, 1978). Syllables thus produce good-quality speech when joined together. The disadvantage of using syllables is that there are approximately 10,000 of them (for the English language), and a complete syllable inventory would require significant amounts of storage space. In response to the problem of minimising storage requirements whilst accounting for syllables, the use of *demisyllables* was introduced. Demisyllables are extracted from syllables. A demisyllable includes all the consonants before the vowel and the beginning of the vowel in an initial demisyllable; in final demisyllables, the end of the vowel and the following consonants are included (Dettweiler & Hess, 1985).

Subphonemic wavelets have been used as concatenative units (Bhaskararao, 1994; Rodet, 1980; Sharman, 1994). These wavelets consist of a single pitch period of recorded speech, and are put together to synthesise speech in the time domain. One problem with this method is that placing wavelets next to each other often results in signal discontinuities, which are detrimental to the overall speech quality. This problem can be alleviated somewhat, by attenuating the ends of each wavelet to zero (Rodet, 1980). However, some speech degradation is inevitable due to fluctuating amplitudes in the signal.

The problem of coarticulation can be greatly reduced by using *word-sized* units, recorded in isolation with a neutral intonation (Linggard, 1985). The words are then concatenated to form sentences. With this technique, it is important that the pitch and stress patterns of each word can be altered in order to give a natural sentence. Word concatenation was successfully used by Fallside and Young (1978), using Linear Predictive Coding as a storage means (see Section 1.2.2.6).

Obtaining units for concatenation can be time-consuming. Segments are extracted from a corpus of speech recorded by a human subject. The speech corpus may consist of several minutes' worth of continuous speech or a set of nonsense words (known as *logatomes* (Dutoit, 1997)). The speech should be recorded in a sound-proof booth, preferably in a single session. It is desirable that the speaker should utter the speech with a constant speaking rate and in a neutral manner (e.g., without exhibiting attitudes such as anger or boredom). However, it is very unlikely that a speaker will be successful in recording the speech in this manner, and so it is often necessary for some speech segments to be re-recorded. Once the speech segment inventory has been obtained, it is usually normalised for amplitude and fundamental frequency (Dutoit, 1997; Tatham & Lewis, 1992).

Concatenative units can be automatically extracted from speech recordings (Donavon, 1996), using procedures based upon Hidden Markov Models. Whilst such approaches give a consistent approach to segmentation, they are less accurate than manual labelling (Vonwiller *et al.*, 1997).

1.2.2.5 Concatenation & Speech Quality

The process of concatenation, in which consecutive speech segments are conjoined, usually involves smoothing algorithms. Every junction between segments is a possible source of degradation in quality, particularly when acoustic units are conjoined. It is thus important to minimise signal discontinuities at each junction (Hess, 1992). This can be achieved in several ways. Firstly, by using appropriate segments in the inventory, degradation due to vowel-vowel concatenation points can be avoided (the density of concatenation points can be decreased overall by reducing the number of segments that make up utterances). Secondly, with careful recording of speech segments, mismatches between the segments in the inventory can be minimised (Dutoit, 1997). Thirdly, the speech segments can be stored in parametric form, so that a simple linear interpolation of the parameter tracks can be carried out from one segment to the next. Other techniques can be employed to reduce clicks at segment boundaries, by reducing amplitude discontinuities. These include the interpolation of amplitudes at segment boundaries and the attenuation of the ends of

waveforms to zero (Yazu & Yamada, 1986). Signals can also be slightly overlapped (e.g., with an overlap width of 25 ms) and weighted by a raised cosine window (Huckvale, 1996).

A significant improvement to concatenative speech synthesis techniques came with the *Pitch-Synchronous Overlap-Add* (PSOLA) method (Moulines & Charpentier, 1990). This technique allows the point of concatenation to be distributed over an entire pitch period (see Section 1.2.2.7) and modifies the speech signal directly, without using a source-filter model.

Although the production of realistic prosody is important (see Section 1.4), it has been shown (Kraft, 1994) that more attention is needed for the concatenation process itself in speech synthesis research.

1.2.2.6 Linear Prediction

Linear Prediction (LP) follows the source-filter speech synthesis method (explained in Section 1.2.1), and is used extensively in concatenation synthesisers (Witten, 1982). LP has a simpler structure than formant synthesis, and can be used for both speech synthesis and speech storage. For instance, LP was used as a storage means in the commercially successful 'Speak'n'Spell' toy (Witten, 1986).

A linear predictive analysis of speech yields all the spectral properties of speech (except its periodicity). LP exploits the redundancy inherent in speech, so that only the difference between the current speech sample $x(n)$ and the previous sample $x(n-1)$ needs to be transmitted (Linggard, 1985). The current speech sample $x(n)$ must be multiplied by a coefficient a_1 to minimise this difference for each time step.

The error between speech samples is expressed in Equation 1-1.

$$e(n) = x(n) - a_1 x(n-1) - a_2 x(n-2) - \ldots - a_p x(n-p)$$

$$= x(n) - \sum_{k=1}^{p} a_k x(n-k) \qquad \textbf{(1-1)}$$

In Equation 1-1, p refers to the number of speech samples preceding the current sample. The LP coefficients, a_k are obtained by minimising the sum of the squared errors over the current analysis frame and must be recalculated every 10-25 ms. This calculation is commonly performed using either the *covariance method* or the *autocorrelation method*, which differ in the range of n over which the error is minimised (Witten, 1982). Coefficients calculated using the autocorrelation method have the advantage that the filter they define is guaranteed to be stable (Markel & Gray, 1976). It can be shown that LP analysis is equivalent to matching the power spectrum of the all-pole filter defined by the LP coefficients to the spectrum of the speech signal. This matching is weighted to achieve the most accuracy in the vicinity of the formant peaks (Markel & Gray, 1976). The digital filter thus models the spectral envelope of the speech signal, and the error signal $e(n)$ (ideally) contains only the harmonic structure of the speech and/or white noise. The speech can therefore be re-synthesised at a different fundamental frequency by exciting the filter.

It should be noted that LP is not well suited to rule-based synthesis, since rules are most easily specified in terms of formants, and the relationship between the coefficients used to define the LP filter and formants is not a simple one (Donavon, 1996).

1.2.2.7 Pitch-Synchronous Overlap-Add

Pitch-Synchronous Overlap-Add (PSOLA) is a concatenation technique that allows prosodic modifications to be performed directly on the speech signal (Charpentier & Stella, 1986; Moulines & Charpentier, 1990). PSOLA is not a source-filter model. Rather, it simply works by windowing and recombining the waveforms of existing concatenative units (Donovan, 1996). PSOLA allows prosodic attributes such as pitch

(see Section 1.2.3.5) to be manipulated by directly modifying continuous waveforms, without using any parametric model.

PSOLA involves three main processes: analysis, modification and synthesis. Firstly, speech waveforms are analysed to give an intermediate representation. The speech waveform $x(n)$ is converted to a sequence of short-term signals $x_m(n)$, by multiplying it by a sequence of pitch-synchronous Hanning windows $h_m(n)$:

$$x_m(n) = h_m(t_m - n)x(n) \tag{1-2}$$

The pitch-synchronous Hanning windows are so-called because each one is centred on one of several pitch markers t_m (which are placed at a pitch-synchronous rate on voiced portions of the signal, and at a constant rate on unvoiced parts). Each Hanning window is set so that adjacent short-term signals overlap somewhat. This involves setting each window's length to be proportional to the local pitch period.

Next, prosodic modifications are applied to this intermediate representation. Pitch modifications are produced by changing the time-shift between frames from the original pitch period T_0 to a new period T. This is achieved through a combination of possible operations: modifying the number of short-term signals, modifying the time delay between short-term signals or modifying the short-term signals themselves. This constitutes a simple and efficient way of changing the pitch of a periodic signal (Dutoit, 1997).

Finally, the modified intermediate representation is recombined to produce the final synthetic signal. This is achieved using the overlap-add (OLA) operation, of which several versions are available (Moulines & Charpentier 1990).

Since PSOLA is a family of methods for manipulating the prosody of speech waveforms, some of the different versions of PSOLA are outlined below.

1.2.2.8 Time-Domain Pitch-Synchronous Overlap-Add

In PSOLA, a signal is decomposed into separate, overlapping segments and then recombined via overlap-adding. In Time-Domain Pitch-Synchronous Overlap-Add (TD-PSOLA), the speech segment inventory consists of parameter lists consisting of sequences of pitch markers t_m, denoting the centres of overlap-add frames. Pitch marking is applied on voiced parts of segments, with the assistance of automatic pitch marking algorithms (Di Francesco & Moulines, 1989), although such algorithms suffer from a lack of precision and often require manual correction. On unvoiced portions, pitch marks are regularly spaced. Apart from pitch marking, no parameter estimation stage is necessary. Before segment recombination takes place, a new pitch-marker sequence is defined, in order to manipulate the fundamental frequency. This is carried out by adjusting the time intervals between pitch markers. For example, increasing the time intervals results in a lowering of pitch. Duration modifications are made by repeating or omitting segments (Kortekaas & Kohlrausch, 1997). TD-PSOLA is an efficient technique which is generally preferred for real-time concatenative synthesis, due to its low computational requirements (Dutoit, 1997).

1.2.2.9 Other versions of PSOLA

One limitation of TD-PSOLA is that no spectral smoothing is possible, and with simple diphone inventories, some audible discontinuities are produced during concatenation. This problem was alleviated somewhat by Dutoit and Leich (1993) with the Multi-Band Re-synthesis PSOLA (or MBR-PSOLA) technique, which is a means of improving the segment inventory. This involves applying a (computationally expensive) Multi-Band-Excited analysis-synthesis procedure to the synthesis unit inventory, in order to make it more suitable for the TD-PSOLA method. The voiced parts of segments are re-synthesised with a constant pitch, eliminating the need for pitch-markers in the segment inventory. This method goes some way towards reducing the concatenation discontinuities associated with simple diphone inventories.

A Linear Predictive Pitch-Synchronous Overlap Add (LP-PSOLA) method has also been described (Moulines & Charpentier, 1990). This involves performing an LP

analysis of speech to obtain the signal's source and filter components. TD-PSOLA modifications can then be applied directly to the LP filter's excitation signal. This method allows an improved spectral modelling, since 'the TD-PSOLA scheme produces less distortion when applied to the residual waveform than on the speech signal itself' (Moulines & Charpentier, 1990).

Frequency-domain versions of PSOLA also exist (Moulines & Charpentier, 1990). In FD-PSOLA, the synthesis short-term signals are obtained by a frequency-domain transformation of the translated signal. FD-PSOLA consists of analysis, modification and synthesis stages. In the analysis stage, the signal is separated into source and filter components by computing the complex short-term spectrum of the signal (where the time origin coincides with the analysis pitch-mark). In the modification stage, the spacing between the pitch harmonics is altered, giving the required fundamental frequency. In the synthesis stage, the modified representation of the signal is reconverted back to a synthetic complex spectrum, via an inverse Fourier Transform. FD-PSOLA has the benefit of supporting modifications of voice quality (Valbret *et al.*, 1992), since the speech signal's spectral characteristics can be altered.

1.2.2.10 Drawbacks of PSOLA

The PSOLA method has the disadvantage that small Hanning windows (Linggard, 1985) must be used, otherwise a mismatch occurs between the new synthesis frequency and the frequency inherent to each short-term signal during synthesis. This results in selective alteration of the pitch harmonic amplitudes for voiced speech, and causes a reverberation effect to be heard. Secondly, PSOLA can introduce annoying artefacts such as 'hoarseness and roughness' of the synthesised signal (Kortekaas & Kohlrausch, 1997), which are similar to comb filtering. Such artefacts are hard to predict beforehand.

Little is known yet of the perceptual effects of the PSOLA operation. Kortekaas and Kohlrausch point out that 'even if PSOLA manipulation of a speech signal does not lead to the perception of either of the artefacts mentioned above, the manipulation does affect its spectral content'. Problems can also occur when increasing the

durations of unvoiced sounds. Repeating the unvoiced short-term signals can result in tonal noise, due to the local periodicity. Voiced sounds can also suffer similar problems. This can be solved for unvoiced sounds by reversing the time axis for repeated short-term signals. However, since PSOLA has been so successful, it is possible that these spectral modifications are perceptually subliminal.

1.2.2.11 Hybrid Concatenative/Formant Synthesis Systems

There are clear advantages to both the formant and concatenative synthesis approaches. Formant synthesis has the benefit of providing almost unlimited flexibility, allowing any property of speech to be modified easily (Witten, 1982). Concatenative synthesis easily captures the *microprosodic* attributes of speech (Posmyk, 1989), which include natural voice characteristics on the segmental level. It is therefore not surprising that some research seeks to combine the best aspects of both of the above approaches. For instance, Pearson, Holm and Hata (1997) combined a library of sampled consonant waveforms with a sampled natural glottal source. It is expected that in the future, hybrid approaches will become more widespread.

1.2.3 Text-to-Speech

Most speech synthesisers can produce a speech signal, given a stream of phoneme symbols to specify the pronunciation (as defined in Section 1.2.2.1, a phoneme is a minimal, linguistic unit). A logical extension of speech synthesis is to produce natural-sounding synthetic speech from unrestricted text, either input from a keyboard or from a text file. This is known as *text-to-speech synthesis*, and can be carried out using either formant or concatenative speech synthesisers. Text-to-speech synthesis involves applying many conversion processes to the input text before the speech can be synthesised (see Figure 1.4). Firstly, the text is transformed into symbols such as phonemes (Robinson, 1994; Yost, 1994). The text analysis stage also produces symbols (known as *prosodic* symbols) which define aspects of the speech such as pitch, rhythm and amplitude (see Section 1.2.3.4). By correctly manipulating prosodic factors, the synthetic utterance can sound more natural, whilst its meaning is clarified. In practical terms, text-to-speech systems must be able to derive these prosodic labels,

given only the input text. These must be predicted accurately if natural-sounding speech is to be synthesised.

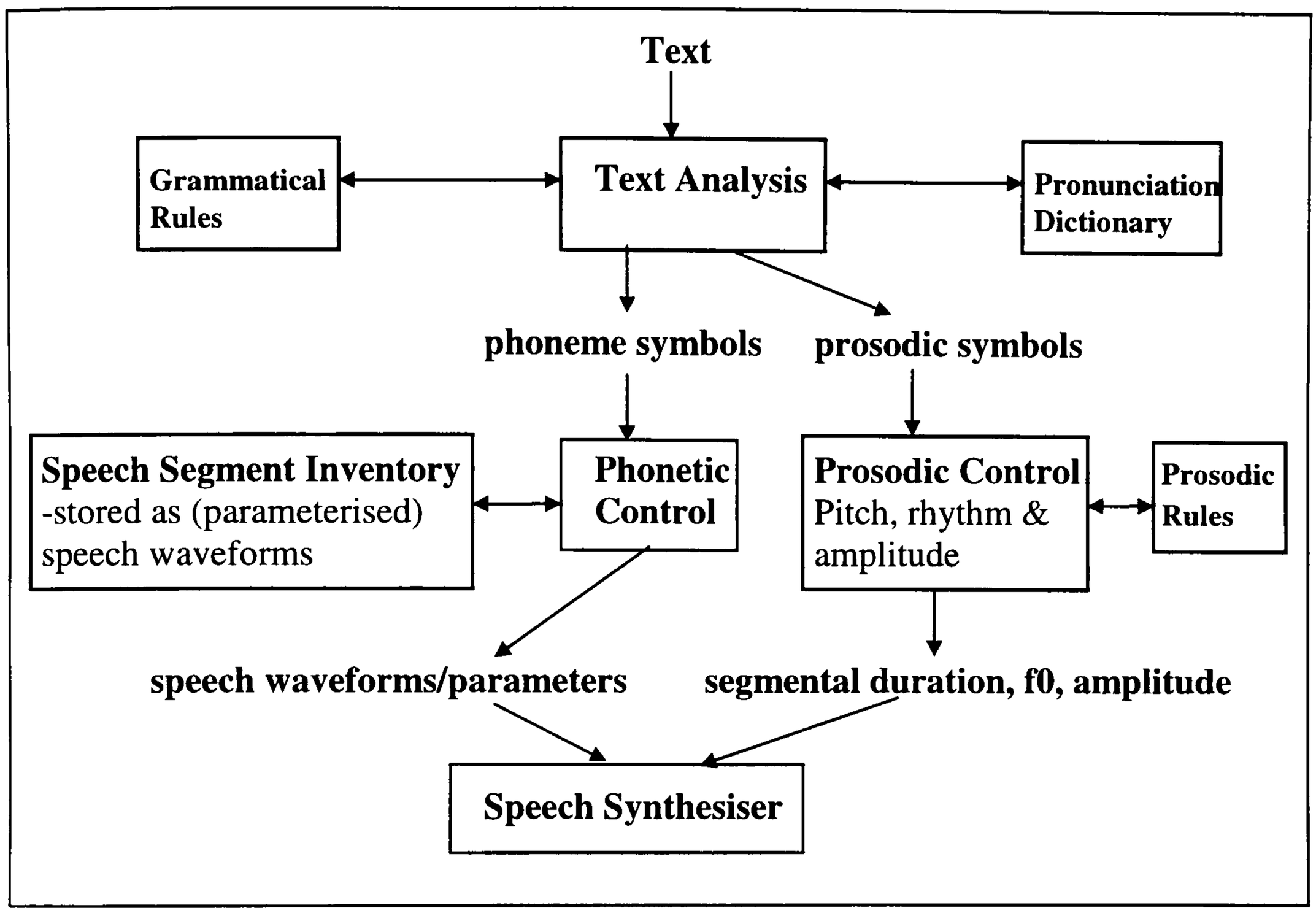


Figure 1.4: Overview of text-to-speech synthesis

1.2.3.1 Text Analysis

For realistic text-to-speech synthesis, one must predict, from a given text, sufficient information to allow a speech signal to be constructed. Text-to-speech systems must therefore extract as much information from the input text as is feasible, given the constraint of producing speech in real time.

Text-to-speech systems contain a text analysis module, which generates both a pronunciation (see Section 1.2.3.3) and a linguistic description from the input text (Edgington *et al.,* 1996a). The linguistic description assists in deciding the meaning of the utterance, and this in turn affects the pitch, rhythm and amplitude that must be applied to the synthetic speech (see Section 1.2.3.4). The generation of a linguistic

description is carried out via *tagging* and *parsing* operations, which are described below.

Before these operations take place, the text is *normalised,* involving the expansion of abbreviations and the interpretation of non-word characters such as paragraph markers, punctuation and numbers (Edgington *et al.,* 1996a). Unfortunately, of all the punctuation marks, only the role of the question mark in a sentence is unambiguous (McAllister, 1989). Often, contextual and real-world information have to be used to disambiguate abbreviations such as 'St.', which could refer to either 'Saint' or 'Street'. In text-to-speech systems, text normalisation is usually carried out by a separate module (Hallahan & Vitale, 1997).

Whilst text analysis is mainly carried out for shorter phrases and sentences, it is becoming more ambitious in scope, extending to longer sentences (Edgington *et al.,* 1996a).

1.2.3.2 Tagging and parsing operations

Tagging involves the assignment of *part-of-speech* information to each word, in an unambiguous manner (Edgington *et al.,* 1996a). Some researchers refer to part-of-speech attributes such as *function words* and *content words* (Dutoit, 1997; Sorin *et al.,* 1987), although this terminology is becoming somewhat outdated. Function words express grammatical relationships, whilst content words have an independent, 'dictionary' meaning (Crystal, 1987). Examples of function words are articles, conjunctions and prepositions; content words include nouns, verbs and adjectives. Such information is necessary for the assignment of pitch context.

Two main groups of part-of-speech category are usually identified: *lexical categories,* such as nouns, verbs, adjectives, adverbs, and *grammatical categories* such as pronouns and articles, prepositions, conjunctions, interjections.

It is important at this stage that potential ambiguities are identified. Consider adjectives and nouns that might be orthographically identical (e.g., the word

'content'), but require different stress patterns; in adjective form, this word is rendered as [con+***tent***], whilst the noun form has the pattern [***con***+tent] (Tatham & Lewis, 1992). These differences in category should be signalled by information from a dictionary (see Section 1.2.3.3) and passed to the *parser*.

Parsing or *syntactic analysis* produces a structural representation of the input text. This assists in the addition of *prosodic markers* such as prominence, juncture and sentence type (Allen, 1995). Prosodic markers are used in the prediction of segmental durations and pitch contours (see Section 1.2.3.4). The phrase markers produced are useful in determining the pronunciation of a word, since this can vary with part-of-speech. Phrase boundaries can also correspond to pauses for breath (Donavon, 1996), and can be signalled by punctuation marks. Accurate placing of phrase boundaries requires a syntactic parse of each sentence. This is a non-trivial task since phrase boundaries are not always indicated by the punctuation. Consequently, phrase boundary placement is often carried out using data-driven methods to automatically construct a *decision tree* from a labelled speech database. For instance, Breiman *et al.* (1984) used Classification and Regression Trees (CARTs) to cluster such a database according to features such as speaker identity, sentence type, time- and word-based distance information, part-of-speech, syntax and pitch accent.

In text-to-speech systems, parsing is usually only carried out to a limited extent. Whilst full clause-level parsing remains elusive, phrase-level parsing is rapid, reliable and less complex. Measures can also be taken to ensure that a valid parse is obtained for each sentence. For instance, in the SPRUCE text-to-speech system, if a parse fails, then a statistically based choice is made between default solutions (Tatham & Lewis, 1992).

1.2.3.3 Text-to-phoneme conversion

Text-to-phoneme conversion aims to produce an accurate pronunciation for any input text, represented by a stream of phonemes. Text-to-phoneme conversion is an essential part of the text-to-speech process, in that no synthetic speech is acceptable unless a correct pronunciation is used. This process involves several stages.

Words and symbols produced by the text normalisation stage are used to determine *word pronunciation.* Determining word pronunciation is very difficult for the English language, since there are many pronunciation exceptions which make it difficult to find comprehensive sets of pronunciation rules. For this reason, it is common practice to maintain a dictionary of phonological information (N.B. *Phonology* is the study linking high-level linguistics and phonetics).

The MITalk system (Allen *et al.,* 1987) used a dictionary containing 12,000 morphemes and their respective phonetic representations. *Morphemes* are the smallest contrastive units of grammar (Crystal, 1987); these basic, meaningful elements make up words (O'Shaughnessy, 1987) and contribute to a word's meaning. Using morphemes is a more feasible alternative to storing a comprehensive word lexicon. With this method, each word input to the text-to-speech system must be reduced to its constituent 'morphs' (a *morph* is the physical form of a morpheme), in a morphemic analysis stage (Hallahan & Vitale, 1997). So, the word 'bidding' might be split into the morphs 'bid' and 'ing' (Witten, 1982). Morphemic analysis can be problematic, since there can be several possible decompositions for a single word. For instance, 'resting' could either be split into the morphs 'rest + ing', or 're + sting'. Selection rules must therefore be incorporated into the morph decomposition process, in order to choose the correct decomposition (Allen, 1976). Once the morphs are obtained, a lexicon or dictionary is searched for the phonetic pronunciation of each morpheme. However, it is not always possible to directly concatenate a sequence of morphs, since a morph's pronunciation may be affected by its context. Allen (1976) addressed the problem of morph concatenation using morphophonemic rules.

After the system's lexicon/dictionary has been searched, any remaining words that were not found are passed to a set of letter-to-sound rules, in order to produce a pronunciation (Williams, 1994). Letter-to-sound rules treat a word as an unstructured sequence of letters. A scanning window is then passed across each word to identify consonant and vowel clusters that can be converted to phonemes. This approach can, however, produce pronunciation errors if the rules are applied across morph boundaries.

Alternatives to rule-based phonetization systems have appeared in the form of connectionist systems, such as NETtalk (Sejnowski & Rosenberg, 1987). The NETtalk neural network accepted a 7-letter sliding window of text as an input, and used three layers of neurons. The input layer was arranged as 7 groups of units, corresponding to a single letter, punctuation marker or word boundary. The middle layer contained 120 units, with approximately 25,000 connections. For the central letter in the input window, an output was produced in terms of articulatory information, syllable boundaries and stress markers. The system predicted 90% of phonemes correctly after being trained on a dictionary of 20,000 words. Various improvements have been made to the NETtalk system (Adamson & Damper, 1996; McCulloch *et al.*, 1987), such as the use of a recurrent (temporal) network. However, results produced by neural network approaches are relatively poor, compared to dictionary look-up methods (Dutoit, 1997).

1.2.3.4 Prosody

Listeners often complain that text-to-speech synthesisers sound unnatural. Common objections are that 'the rhythm sounds wrong' or that 'the pitch sounds unnatural'. These aspects of speech are covered by the term *prosody*, which is analogous to 'the music of speech', and refers to properties such as pitch contour, loudness, and segmental duration. As well as making synthetic speech more natural-sounding, prosody can be used to clarify the meaning of a sentence, and by itself can convey linguistic concepts (Sonntag & Portele, 1996).

The following sections examine the major aspects of prosody: fundamental frequency and segmental duration, and their application to text-to-speech synthesis.

1.2.3.5 Fundamental Frequency

Fundamental frequency (f0) is an important cue for sentence type (e.g., questioning, declarative, etc.). Fundamental frequency is linked (but not identical) to *pitch*, which is a psychological construct related to the *perception* of frequency (Levitin, 1998). When a suitable pitch contour is applied to a synthetic utterance, naturalness is

improved and intended meaning is clarified. The intonational 'melody' of the speech conveys syntactic information, and hence semantic information ('t Hart, Collier & Cohen, 1990; Selkirk, 1984). This includes markers of questioning, and discourse information such as topic or theme. Fundamental frequency also conveys the speaker's attitude, which may include politeness or irony (Parke & Waters, 1996).

Whilst fundamental frequency conveys semantic information to the listener, deducing it from unrestricted text is particularly difficult. For instance, pitch is affected by the 'importance' or 'information value' of a word. 'New' items in an utterance tend to be accented, whilst 'given' information is usually deaccented (Hirschberg, 1993). In determining pitch accent, heuristics such as *previous mention* are used, in which the number of previous occurrences of a word (or its root) is recorded, giving an indication of the importance of a word (Pelachaud & Prevost, 1995). However, determining the 'givenness' of items remains troublesome. For instance, in the sentence 'The SLEEPY SOLDIER SLEPT for an HOUR', the mention of *sleepy* does not necessarily imply that *slept* is deaccentable (Hirschberg, 1993).

The above discussion highlights the importance of adequate prosodic labelling of text. Once these prosodic labels are available, actual pitch contours must be predicted. Fundamental frequency algorithms for text-to-speech separate into distinct categories: those where an actual pitch contour is applied (possibly from an inventory of tunes), and those in which a sequence of relative, elementary tones is specified (Dutoit, 1997). These different approaches are described in the following sections.

Generating f0 as sequences of stylised contours

Fundamental frequency can be generated as a sequence of stylised contours, usually taken from an inventory of tunes, or melodic elements (Kohler, 1991). This approach exploits the fact that certain types of phrases exhibit characteristic patterns. For instance, most phrases exhibit a slight initial rise in f0, followed by a steady decline towards the end of the phrase. It is also usual to raise the f0 pattern at the end of interrogative sentences (Sagisaka, 1990). Once a sentence has been categorised, an overall tune can then be applied.

Pitch contours can be generated with several possible approaches. Basic pitch patterns can be specified in terms of two falling declination lines: a high 'topline' and a lower 'bottomline', which gradually converge towards each other as the sentence progresses (O'Shaughnessy, 1987; Tatham & Lewis, 1992). Another reasonably simple way of assigning f0 is to divide the utterance up into intonational phrase groups, which can each be characterised by a single 'tune'. A pitch contour is assigned to each tone group by identifying the tonic syllables (i.e., the major stress point of each tone group). For instance, Halliday (1970) identified five primary intonation contours, each hinging on the tonic syllable (i.e., the most prominent one). Under Halliday's scheme, a tone group boundary is placed by each punctuation mark. The tonic syllable is identified as the first syllable of the last foot in a tone group (N.B. a *foot* is the interval between two stressed syllables in a sentence). A pitch contour is then chosen depending on the tone group's position within a paragraph, and whether the question is a *yes-no question* or not.

A data-driven alternative to the above is to select and hierarchically organise meaningful segmental features from a prosodic database, producing a decision tree (Dutoit, 1997). Each leaf of this tree has an associated melodic contour, which is defined by averaging the melodic information provided by each of its representatives in a prosodic database. Connectionist approaches have also been used to map syntactic-prosodic information to the observed stylised curves. For example, Traber (1992) controlled f0 using a recurrent neural network, which used a sliding window technique to associate letter sequences with pitch contours. The system produced a series of stylised melodic patterns, which were then concatenated (Dutoit, 1997).

Generating f0 as sequences of tones

In this technique, f0 is viewed as a series of relative, elementary tonal units (Pierrehumbert, 1981), which are converted later to numerical f0 values. Pierrehumbert used a target and transition notation to describe pitch contours. This remains a popular annotation method, and has been adapted to form other labelling systems, such as ToBI (Silverman *et al.,* 1992). In this system, the pitch contour is initially described as abstract intonational events, which are later converted to actual phonetic values (Delin & Zacharski, 1997).

Intonational melodies are described by pitch accents, phrasal tones and boundary tones (Pelachaud & Prevost, 1995). High or low frequency phrasal tones are represented by the symbols **H** and **L**, whilst stressed syllables are marked with an asterisk. Stressed syllables are generally higher in amplitude than unstressed syllables (Crystal, 1987), although syllables can also be perceived as stressed if they are acoustically longer in duration. Pitch accents specify the relative pitch for various parts of a word. In English, the set of pitch accents is as follows:

H*, L*, L+H*, H*+L, L*+H and **H+L***

Using this notation, the pitch accent contained in the word *computer* would be annotated as **L+H***; this translates to a pitch contour made up of a low pitch, followed by a marked pitch increase on the second syllable. Intonational phrase boundaries are represented by the markers **H%** and **L%**. An example of prosodic labelling using ToBI is shown below for the sentence 'Marianna made the marmalade':

```
Marianna made the marmalade.
     H*            H*   L-L%
1    1  1            4
```

In the example, the numerical values in the 'break-index tier' label the end of each word for the subjective strength of its association with the next word, on a scale from 0 (for the strongest perceived association) to 4 (for the most disjoint). This relative notation is used in order to describe pitch contours in cognitive terms, signalling the speaker's intent. However, since this notation only describes pitch in relative terms, a procedure is necessary for producing an actual intonation melody.

For this reason, tone sequence intonation models for text-to-speech are often separated into high-level and low-level systems. An example is the O'Shaughnessy fundamental frequency algorithm (1977), which was used in the MITalk system (Allen *et al.*, 1987). At the high-level, syntactic information such as sentence type, clauses and phrases were used to produce an approximate f0 contour. The low-level system refined this contour using information such as phoneme type, number of syllables per word and lexical stress.

1.2.3.6 Segmental Duration

An essential aspect of prosody, is segmental duration (Carlson *et al.,* 1979; Ostendorf *et al.,* 1990). Segmental duration is an important perceptual cue, which facilitates discrimination between different segments of speech. For instance, duration aids the judgement of whether a vowel is long or short (Nooteboom, 1973), or whether a fricative is voiced or unvoiced (Cole & Cooper, 1975; Idson & Massaro, 1981). Segmental durations can be used to produce contrastive stress, or to alter the speaking rate (Klatt, 1979). On the sentence level, segmental duration can also signal whether a word is phrase-final or not (Klatt, 1975).

Segmental duration directly affects the overall rhythm of speech, and it is important to consider its effect on timing at the sentence and paragraph level. One theory of speech timing that has been widely discussed is the idea that continuous speech is isochronous, i.e., stressed syllables seem to occur at regular intervals (Abercrombie, 1964). This is also referred to as *stress timing.* Intervals between stressed syllables are known as 'feet', and are approximately comparable to musical 'bars'. In the isochronous foot theory, the duration of syllables is adjusted in such a way as to make the total duration of each foot approximately constant (Dutoit, 1997). However, the isochrony theory has not been strongly supported by empirical evidence (Lehiste, 1977). Despite this lack of evidence, English is often viewed as being isochronous. This suggests that isochrony is mainly a perceptual phenomenon in which heavy syllables tend to dominate (Fant *et al.,* 1991). The belief that English is stress timed is due to the human tendency to impose a rhythm upon any sequence of intervals (Allen, 1975).

The perception of duration can be studied using a range of different stimulus types. These include non-speech stimuli such as white noise, filtered noise or pure tones (Fujisaki *et al.*, 1975). In the *acoustic marker* technique, sensitivity to duration is measured by presenting subjects with sinusoidal tones, separated by pauses. An experimental stimulus would be a sinusoidal tone lasting 170 ms, followed 10 ms later by an identical 170 ms tone. Other stimuli would differ only by the length of the silence between the two acoustic markers (Yost, 1994).

For both formant and concatenative speech synthesis, a duration parameter must be predicted for each segment that is output. Careful chosen segmental durations result in a marked improvement in the naturalness of synthetic speech, whilst inappropriate segmental timing can cause the speech to sound 'unnatural and laboured' (Breen & Easton, 1994). This makes the prediction of segmental durations an important task. Unfortunately, it is also a difficult task, since segmental timing is influenced by many factors (Port, 1981).

Segmental timing for text-to-speech is influenced by syntactic, lexical and semantic factors (Klatt, 1976). Syntactic factors include the locations of phrase and clause boundaries in the input text. These boundaries may be determined manually (Black & Taylor, 1994), predicted by rule or identified from the input text using a stochastic approach (Edgington *et al.,* 1996a).

Lexical stress markers define the stress level for a speech segment. It is common for three stress levels to be used: high, normal and unstressed. All of these levels have an effect on segmental duration. A substantial part of the variation in segmental timing involves the actual combination of speech segments that is used to synthesise speech. The segments produced by the speech synthesiser's lexical module all possess inherent durations, which by themselves account for most of the variation in segmental duration (Takeda *et al.*, 1989).

Semantic factors contribute to the meaning of an utterance. These include contrastive stress (Pelachaud & Prevost, 1995) and variations in speaking rate. For example, speakers will often slow their speaking rate when words with a high information content (i.e., unusual words) are encountered (although Eefting (1991) showed that for the Dutch language, duration is only affected if an unusual word is stressed). However, non-verbal communication, such as 'information content', is not usually available to text-to-speech synthesisers (Hawkins & Warren, 1991).

Duration models have been proposed for various languages (Bartkova & Sorin, 1987; Carlson & Granström, 1986; Coker, Umeda & Browman, 1973; Huber, 1990; Möbius & van Santen, 1996; Pitrelli & Zue, 1989). Segmental duration models are usually produced with the assistance of a timing database (van Santen, 1994) consisting of a

recording of continuous human speech, labelled with segmental durations (Campbell, 1992). Duration rules can then be formulated by trial and error, attempting to fit the predicted durations to those in the timing database (e.g., Klatt, 1979). It is desirable for duration rules to incorporate articulatory or phonological principles from the phonetics literature (van Santen, 1993). Alternatively, a statistical model can be constructed, using a tree approach (e.g., Breen, 1992b) to exploit statistical regularities in speech data (van Santen, 1993).

Rule-based segmental duration models usually incorporate several 'rules of thumb'. For instance, Morton (1987) states that all sounds have intrinsic durations and amplitudes, and the intrinsic duration of vowels may be shortened in certain contexts (such as when they occur before /p, t or k/). Thus, in most rule-based segmental duration models, an intrinsic duration is first assigned to segments (including pauses), which is further modified by successively applying rules combining co-intrinsic and linguistic factors into additive or multiplicative factors (Bartkova & Sorin, 1987; Klatt, 1976; O'Shaughnessy, 1984).

Klatt's (1979) duration model is probably the most well known of its kind, forming the basis for many other models. In this model, an 'inherent duration' is assigned to each phoneme. The inherent duration is 'roughly the duration to be expected in nonsense CVCs spoken in the carrier phrase 'Say bVb again' or 'Say CaC again''. Here, *CVC* refers to a consonant-vowel-consonant syllable. Klatt specified a set of 11 durational rules which cause a percentage change in the segment's inherent duration, taking into account factors like syntactic information, position in a word, stress markers, etc. A 'minimum duration' value was also defined, which limits the extent to which a phoneme can be shortened. This corresponds to the minimum duration for that phoneme if stressed. Equation 1-3 is used to calculate segmental durations:

$$\mathbf{DUR=[(INHDUR-MINDUR)\ x\ PRCNT]/100 + MINDUR} \qquad \textbf{(1-3)}$$

where INHDUR refers to a phoneme's inherent duration, MINDUR is the minimum duration that a phoneme must take when stressed and PRCNT is a percentage shortening value determined by Equation 1-4:

$$\mathbf{PRCNT=(PRCNT \times PRCNT1)/100} \qquad \textbf{(1-4)}$$

Here, PRCNT1 is a value specified in each of the 11 duration rules.

Klatt's rules predicted reasonably adequate segmental durations. When segmental durations predicted by Klatt's rules were compared with continuous speech, read by himself, a standard deviation value of 17 ms was obtained.

One problem with such models is that the inherent durations for phonemes are difficult to estimate (Huber, 1990), and vary somewhat between the different studies (Carlson & Granström, 1986; Klatt, 1979; Umeda, 1977). This is no doubt a result of variations in speaking rate by individual speakers. Furthermore, there is some variation in inherent and minimum durations between British and American English (for instance, Breen (1992b) had to re-map Klatt's inherent durations to English ones).

Many duration models (such as Klatt's) are speaker-specific. Attempts have been made to create models that are independent of the speaker. Bartkova and Sorin (1987) used several corpora in studying speaker-independent intrinsic durations and their modifications in different contexts. They considered factors such as type of word (i.e., content versus function word), the location of a segment within a word and a segment's distance to major and minor phrase boundaries. This study led to the following rules:

$$\textit{Vowel Duration} = ID \;\; V_i \;\; m_c \qquad \textbf{(1-5)}$$

$$\textit{Consonant duration} = ID \; C_{ij} \qquad \textbf{(1-6)}$$

In equations 1-5 and 1-6, ID refers to intrinsic durations; V_i and C_{ij} are shortening or lengthening coefficients dependent on the position i of a segment within a word and in a sentence; j refers to the phonetic class membership of consonants and m_c denotes, for certain closed syllables, the influence of a consonant or semivowel on the preceding vowel.

More recently, the availability of large speech corpora has been exploited to allow general, statistically-derived models to be produced, in which a large number of possible control factors are simultaneously varied. A general additive model has been proposed by Kaiki *et al.* (1990); Classification and Regression Trees (CARTs) have also been used by Riley (1992) to make continuous segmental duration predictions; CARTS were used to cluster allophone durations according to their context. Campbell (1992) used a connectionist approach. Whichever method is used, it has been shown (Breen, 1992b) that statistical duration models tend to produce slightly more accurate durations than rule-based ones.

Most of the segmental duration models and systems described are designed to predict phoneme durations. However, most concatenative speech synthesisers are based upon other segments of speech (such as diphones). This raises the interesting problem of deciding how to distribute phoneme durations amongst diphones or syllables (O'Shaughnessy *et al.*, 1988). For example, one could assign 50% of the current phoneme duration and 50% of the following phoneme duration to a diphone. However, this problem remains unsolved, and has prompted the creation of segmental duration models based on diphones (Huber, 1990).

It can be seen from the above discussion that segmental timing is affected by many factors, which interact in a complex manner (Port, 1981). Whilst these timing factors have been identified, the extent to which they affect each other is not yet fully understood (Pitrelli & Zue, 1989). It should also be noted that in natural sentence utterances, segmental durations fluctuate more widely than in isolated-word speech (Takeda *et al.*, 1989). On the sentence level, it is probable that a certain amount of variability should be introduced into the predicted segmental durations, particularly in cases where the sentence may be repeated several times within a dialogue (Tatham, 1996). Thus, predicting sentence-level durations will continue to be an important factor in producing natural-sounding synthetic speech.

1.3 Limitations of This Research

This research has several given limitations that should be noted. For instance, the

speech synthesis framework is monolingual, being tailored towards British English. If the developed system were to be used for other languages, additional waveforms would have to be recorded and added to the inventory, to account for additional speech sounds. A new timing database would also be necessary, based on words from the desired language. Furthermore, for each new speaker, a new segmental duration database must be created using the iterative synthesis/adjustment method. Unfortunately, this is requires a lot of time and manual effort. However, it is possible that all concatenative TTS synthesisers require such handcrafted databases in order to sound more natural.

Although this thesis is entitled 'Time-Domain Concatenative Text-to-Speech Synthesis', it is acknowledged that at present, speech can only be synthesised for isolated words. However, the concatenation framework is designed in such a way that text-to-speech synthesis is achievable if some additional components are added. For instance, mechanisms for pitch manipulation and prediction are required. It is intended that this will be achieved using TD-PSOLA. Segmental duration prediction also needs to be implemented on the sentence level, rather than at the word level only.

Although the time-domain concatenative speech synthesis method is evaluated in this thesis, the evaluation is not exhaustive. Further evaluation, such as segmental intelligibility tests, could be carried out.

Chapter Two

2. Speech Synthesis System Components

2.1 Introduction

A time-domain concatenative speech synthesis (TDCSS) method has been devised, which enables speech to be synthesised for whole words, given a text input. Speech is synthesised by concatenating pre-recorded segments of speech, taken from an inventory of speech waveforms (see Chapter 3). The speech segments used are CV, VC and CVC syllables, supplemented with CC diphones. This chapter summarises the components of a prototype text-to-speech synthesiser which has been implemented (see Figure 2.1).

The text-to-speech system contains a Text-to-Segment Conversion module which converts the input text into a segmental representation. A Timing Module then predicts a duration for each segment (this is described in Chapter 4). The Timing Module contains a duration look-up algorithm and a timing database. Finally, the stream of segments and their segmental durations (supplied by the Timing Module) are passed to the Synthesis Module, to be converted into a speech signal. The Synthesis Module contains an inventory of pre-recorded speech waveforms. In order to synthesise a word, an appropriate sequence of waveforms is firstly selected from the inventory, using the given segmental representation. According to the given segmental duration values, a portion of speech is extracted from each waveform. These temporary portions are then concatenated to form a single waveform, which is converted to an audio signal via a Digital-to-Audio Converter.

These individual modules are described in more detail in the following sections.

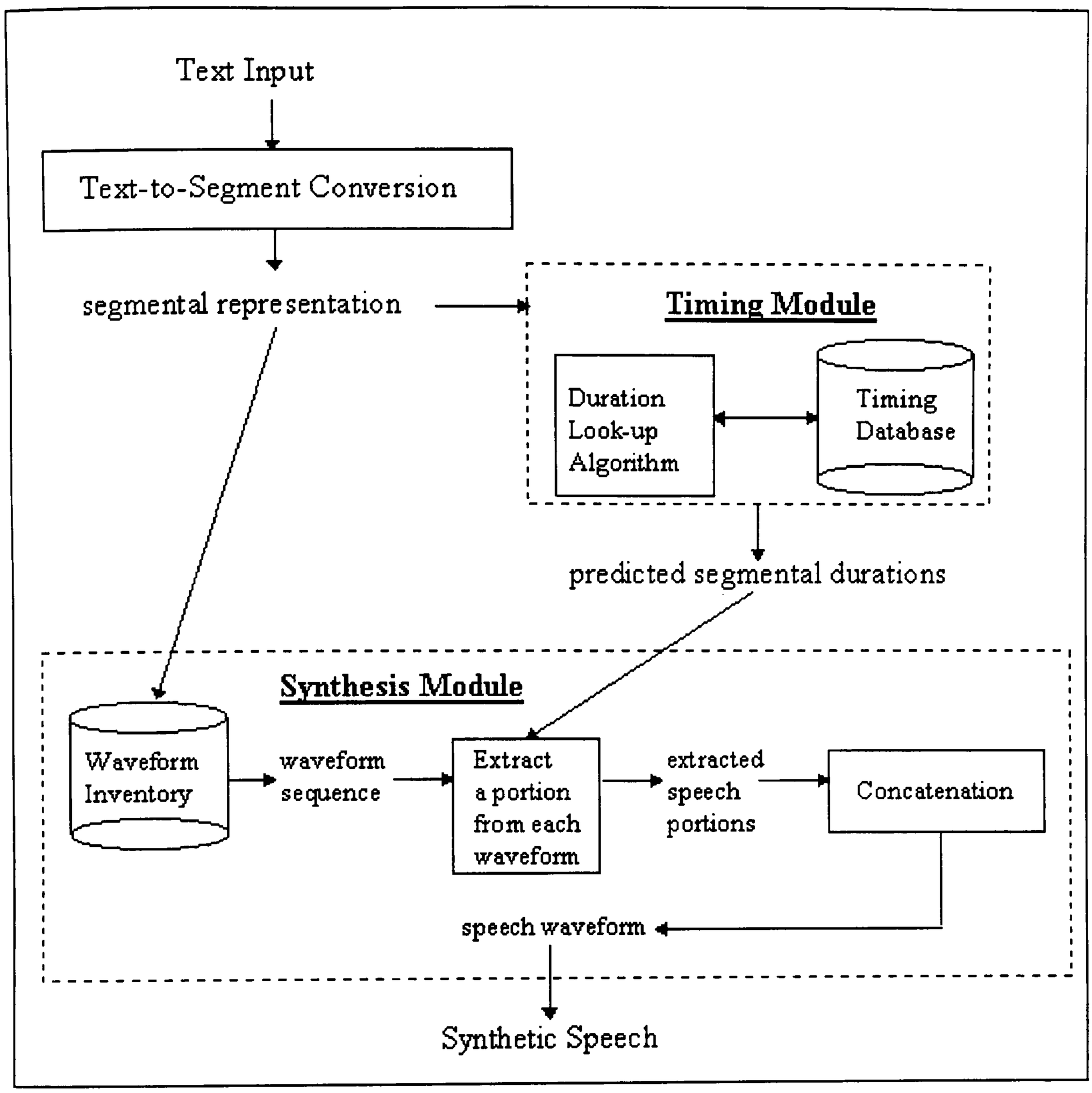


Figure 2.1: Overview of Text-to-Speech System

2.2 The Text-to-Segment Conversion Module

The Text-to-Segment Conversion module converts the input text to a stream of symbols, representing the actual speech segments to be concatenated (see Chapter 3, Section 3.3). In this thesis, these are referred to as *speech segment symbols*, because they represent speech segments in an actual waveform inventory; on the other hand, the term *speech segment* refers to a section of recorded speech. This module contains a text-to-phoneme conversion system, which breaks up the input text into a stream of phoneme symbols, using a set of letter-to-sound rules, which are an adaptation of a set of letter-to-phoneme rules for American English published by the U.S. Naval Research Laboratories (Naval Research Laboratory, 1976).

The phoneme symbols produced are then converted to speech segment symbols such as CC diphones, CV and VC syllables, by grouping them into pairs. The Text-to-Segment Conversion module ensures that for any given text, the largest possible speech segments are selected from the waveform inventory. Where possible, CV-VC syllable pairs are replaced by a single CVC syllable, when available in the speech segment inventory. This ensures that fewer speech segments are used to make up a word, and consequently, the number of concatenation points per word is minimised. Reducing the number of concatenation points per word tends to produce more fluent-sounding speech with fewer abrupt 'clicks' (Kraft, 1993).

2.3 The Timing Module

The Timing Module predicts segmental durations for any input word using a pattern-matching, duration look-up algorithm and a timing database. The duration look-up algorithm is supplemented with a default duration function, which is activated whenever the look-up algorithm fails to return a duration value for a segment. The duration look-up algorithm is described more fully in Chapter 4, and the timing database is explained in Chapter 5. With this approach, the intelligibility and naturalness of the output speech are closely related to the amount of timing data available.

2.3.1 The Timing Database

The timing database consists of sets of segmental durations for frequently used English words. The timing database is created using an iterative synthesis/adjustment technique, involving a cycle of synthesising speech and adjustment by an expert judge. The segmental durations are adjusted purely on the basis of the expert judge's perception of the synthetic speech's naturalness. The creation and nature of the timing database are described in more detail in Chapter 5.

Several important issues are raised, concerning the expert judge approach, used in the creation of timing databases. These are discussed in Chapter 5. These include:

- the question of whether the expert's judgement of naturalness is representative of other people's perception
- the stability of the expert judge's perception over a period of time
- the issue of whether non-expert listeners can carry out an iterative synthesis/adjustment task in the same manner as the expert judge

These points have been examined via a series of listening experiments and an iterative synthesis/adjustment experiment. The findings are discussed in Sections 5.4.6 and 5.5.6.

2.4 The Synthesis Module

At the heart of the Synthesis Module is an inventory of speech segments, recorded by a single, male speaker. The methods used to record the CC diphones and the CV, VC and CVC syllables are described in Section 3.2.2. The Synthesis Module produces the synthetic speech (in the form of a single speech file) by concatenating portions of speech, extracted from pre-recorded waveforms according to a set of segmental duration values.

The Synthesis Module has been implemented using a combination of the C language and the Speech Filing System (SFS) software, version 1.1 (Huckvale, 1996). SFS is a suite of software tools, file and data formats, standards and special programming languages which facilitates speech research. It was supplied freely by the University College of London.

The synthesis program is written in C, and is used to obtain parameters concerning the word to be synthesised, such as an appropriate segmental representation and a set of segmental durations. For example, in order to synthesise the word 'Tom', the symbols /to/ and /om/ would be required as an input. A set of segmental durations (in milliseconds) is also necessary, e.g., [37.5, 140.0]. The segmental durations are produced by the Timing Module, and have a high degree of accuracy in order to minimise discontinuities when concatenating speech segments. Using the above

information, the synthesis program then activates the relevant SFS functions to concatenate the speech segments according to the given segmental durations.

The segmental durations supplied by the Timing Module are used to construct temporary speech files, which are then concatenated to produce the synthetic word. Each temporary speech file is created using the SFS function *slink*, which allows a segment of speech to be extracted from a stored waveform, according to a specified segmental duration. The temporary speech files are concatenated into a single speech signal using the SFS function *spaned*. Although the *spaned* function supplies a smoothing facility, this is not used by the Synthesis Module - a zero overlap is specified, in order to remain true to the concatenation framework described in Chapter 3. Alternative versions of the TDCSS synthesiser that have also been implemented include a 'smoothed' Synthesis Module and a TDCSS2 Synthesis Module, which follows the two-parameter concatenation method described in Chapter 3.

The concatenation framework underlying the Synthesis Module is explained in Chapter 3. The synthetic speech output is evaluated in Chapter 6, in terms of its intelligibility and naturalness. These aspects of the synthetic speech were both investigated by means of listening experiments. One experiment established that intelligible speech is attainable using TDCSS. In a further experiment, the synthetic speech was ranked in terms of naturalness, alongside five well-known text-to-speech synthesisers.

2.5 Summary

A time-domain concatenative speech synthesis method is described, which enables speech to be synthesised for whole words, from a text input. This chapter has outlined the various components of a prototype text-to-speech synthesiser for whole words. The concatenation framework and speech synthesis method underlying this prototype are described in Chapter 3. A Timing Module is also described (see Chapter 4) which is capable of predicting segmental durations for whole words, using a duration look-up algorithm and a timing database.

Chapter Three

3. A Concatenation Framework for Speech Synthesis

3.1 Introduction

Speech can be synthesised by concatenating speech segments, taken from an inventory, to form a continuous signal (Rudnicky *et al.,* 1994). In text-to-speech applications, it must be possible to synthesise an unlimited vocabulary, even though the speech segment inventory is of fixed size (Dixon & Maxey, 1968). The size and content of speech segment inventories is thus an important issue. Although inventories should not be prohibitively large, they must contain a sufficient number of segments to cover a wide range of possible speech sounds. Furthermore, an individual speech segment may have to be used in many different phonetic contexts. This is a difficult problem, because a speech segment, extracted from recorded human speech, consists of one or more *allophones* (Tatham, 1969), rather than phonemes. The transitions between segments may not sound natural, even though a logical order of segments may be used. Dixon and Maxey (1968) succinctly summed up the above problem with this question:

> 'Can a library of synthetic segments be devised which will be acceptable to listeners if the segments are forced to fulfill multiallophonic roles in continuous speech?'

We have seen in Chapter 1 that the concatenation of allophones (Ozum & Bulut, 1994) has an immediate appeal due to the minimal storage requirements. However, the main challenge is that of producing realistic coarticulation - describing the influence of one phoneme on another. The need to model transitional information between phonemes led to the introduction of diphones, which were first proposed by Peterson, Wang & Silvertsen (1958). Requiring only moderate storage space, diphones, which are sometimes referred to as *dyads* (Olive, 1977), are the transition from the middle of one phoneme to the middle of the next. The approach is successful because the centres of allophones are relatively stable, acoustically, and are

reasonably ideal concatenation points. In fact, if a diphone inventory is recorded adequately, it is possible to synthesise intelligible speech by simply appending the relevant diphones (Elsendoorn & 't Hart, 1982) with no smoothing. However, one must exercise caution, since large differences in either fundamental frequency or amplitude can be manifested as clicks, particularly when voiced diphones are appended. One solution is to adjust the initial pitch of the current diphone to match the pitch period of the preceding diphone, thus creating a 'pitch ramp' between the diphones (Yarrington & Foulds, 1993).

The drawback of diphones is that they require a rather large number of concatenation points (one for each allophone). Each concatenation point is a potential source of degradation in the speech output, and so heavy reliance is placed upon an effective smoothing algorithm (Breen, 1992a). A further disadvantage is that diphones can only account for some of the coarticulatory effects exhibited by a language (Dutoit, 1997). It is found that to be really effective, they should ideally be taken from a carrier word similar to the phrase to be synthesised (Breen, 1992a).

A linguistically appealing concatenative unit is the syllable. Syllables are larger in size than diphones, and require fewer concatenation points. Syllables are thought to be more directly linked to the timing of speech, to the extent that some languages are referred to as being *syllable-timed* (Lehiste, 1977). Syllables produce good-quality speech when joined together because most coarticulation occurs within syllable boundaries (Fujimura & Lovins, 1978). As mentioned in Chapter 1 the disadvantage of syllables is that there is a large number of them (approximately 10,000) in the English language.

The introduction of demisyllables (or *half-syllables*) reduced these storage requirements, whilst maintaining the concept of syllable timing. Demisyllables are defined by Fujimura and Lovins (1978) as being half of a syllable, either the set of initial consonants plus half of the vowel, or the second half of the vowel plus any postvocalic consonants (Allen *et al.*, 1987).

It is found that different languages can be synthesised more effectively using particular concatenative unit types. For Japanese speech synthesis, CV syllables have

commonly been used, since a small number of combinations of CV segments can cover all the possible utterances (Iwahashi & Sagisaka, 1995).

The naturalness of concatenative speech can be improved by using several stored versions of a particular phoneme, taken from different contexts, such as triphones (Hirokawa, Itoh & Sato, 1993). When a word is to be synthesised, an optimal set of segments (e.g., diphones) is selected from the segment inventory on the basis of minimising a concatenation cost function (Iwahashi & Sagisaka, 1995). The cost function accounts for factors such as potential waveform distortion between consecutive segments. Since each type of concatenative unit has its advantages, an optimal combination is often used. For instance, Dettweiler and Hess (1985), included demisyllables, diphones and suffixes in the speech segment inventory.

A further alternative is to use concatenative units which are of non-uniform size (Takeda, Katsuo & Sagisaka, 1992). In this technique, speech segments are selected from a continuous speech recording in such a way that they match the context of the message to be synthesised as closely as possible.

This chapter describes a framework for waveform concatenation speech synthesis, that allows intelligible speech to be synthesised at the word level (this is verified in Chapter 6). The concatenation framework utilises a mixture of waveform types, including CV, VC and CVC syllables, extracted from digitised human speech (see Appendix A). Some CC diphones are also used. The framework allows the speech segments to be concatenated with minimal discontinuities whilst avoiding smoothing at segment boundaries. This is achieved by outputting the initial fraction of each segment, specified by a segmental duration parameter. An alternative concatenative framework is also described, in which two segmental duration values are used, specifying a variable 'start point' and 'end point' for each speech segment to be concatenated.

3.2 Time-Domain Waveform Concatenation Framework

One goal of this research was to examine whether intelligible and natural-sounding

speech could be synthesised using a concatenation of speech waveforms in the time domain. The literature available on this subject is not extensive, often dismissing such techniques without a full evaluation. For instance, Witten (1986) warns against a simple abutting of time waveforms. Indeed, such an approach is unlikely to succeed unless both the waveforms and the concatenation points are chosen carefully. Furthermore, Linggard (1985) states that 'simply concatenating words, or syllable-type units will not result in natural-sounding or even intelligible speech.' However, this research demonstrates that certain time-domain concatenation techniques should not be dismissed so readily, and investigates the extent to which TDCSS is a viable method for text-to-speech synthesis.

Concatenating speech waveforms in the time domain has the advantage that the microprosodic information carried in the speech segments is preserved. Processing requirements can also be considerably reduced. This is advantageous because excessive processing of speech segments tends to degrade the overall speech quality. The chief problems associated with this method are waveform discontinuities at segment boundaries (Yarrington & Foulds, 1993) and the incorporation of prosodic information, such as fundamental frequency, into the synthesised speech. These problems are discussed in Sections 3.2.3 and 3.2.5.

3.2.1 Choice of concatenative units

The units chosen for concatenation are CV, VC and CVC syllables, supplemented with CC diphones. Using CVC segments has the advantage that coarticulation effects rarely extend beyond syllable boundaries (Fujimura & Lovins, 1978). The CC diphones are often necessary for producing clusters of consonants, and can be used to span word boundaries in connected speech. For instance, a /ts/ diphone would be needed in the synthesis of the expression 'The ca*t* *s*aw nothing'.

3.2.2 Obtaining the concatenative units

All of the speech waveforms were recorded digitally by a single, male speaker. A PC soundcard was used to convert the analog signal to a digital representation such as a

'.wav' file. The speaker was the author, who is referred to in this thesis as *DSGV*. All speech waveforms were recorded in a quiet room, using a sampling rate of 44,100 Hz and a 16-bit representation. In an attempt to minimise fluctuations in amplitude, a constant microphone distance was used. Also, a constant pitch was used for each speech segment as far as possible. Throughout the recording sessions, the author would frequently listen to random examples of previously recorded speech segments in order to be reminded of the pitch to be used. After every recording session, the speech segments were normalised for amplitude.

The CC diphones were manually extracted from nonsense words by DSGV, using a time waveform display program. Typically, a nonsense word would be recorded several times by the speaker, with short pauses between each version. The relevant diphone would be extracted from the best version (which would be judged by ear, in terms of clarity, overall amplitude and pitch).

When recording CV or VC syllables, it was found experimentally that it was easier to utter them in isolation, rather than to extract them from nonsense words. This approach has the potential danger that the final synthetic speech could sound overly accentuated. In order to counteract this effect, the speaker would imagine the syllable to be part of a carrier sentence such as 'You must *not* ____ a*gain*', where the underscore indicates the position of the syllable. In this sentence, the word *not* is accentuated, as well as the final syllable *gain*. It was found by experimentation that embedding the syllable in this type of carrier sentence tends to have the effect of producing an unaccented syllable. In the above technique, it should be noted that only the syllable to be recorded is actually spoken. The carrier sentence is purely a means of mentally rehearsing the duration and amplitude of the utterance.

At present, only a partial CVC inventory has been recorded. However, it is shown in this chapter that CVC syllables can be constructed by concatenating CV and VC syllables.

3.2.3 Segment boundaries

If speech is synthesised solely by concatenating entire waveforms from an inventory, intelligibility can be seriously degraded. However, this approach can be substantially improved by outputting a specified portion of each selected waveform. The portion of each waveform to be output is specified by a segmental duration parameter (Vine & Sahandi, 1996). This 'portion' refers to a section of the stored segment starting from the onset of the segment t_{start}, and finishing at any point $t_{start}+s$, where s is the specified segmental duration. Thus, depending on s, a smaller or larger portion of a waveform is output. This is shown in Figure 3.1, in which portions of the syllables /DI/ and /Is/ are used to synthesise the word 'this' (the portions are indicated by arrows). The first two windows show the complete syllables, and the final window shows the synthetic word 'this' (in the final window, the waveform has been re-scaled to fit the page). As the diagram shows, only the portion of the /DI/ waveform is output before /Is/. For intelligible speech to be output, TDCSS requires a high degree of timing accuracy, since the timing data is specific to a particular inventory of speech waveforms. This is unlike conventional speech synthesis methods, in which each segment is squashed or stretched to fit a particular duration (Isard & Miller, 1986). Since such speech synthesis methods use smoothing procedures at segmental boundaries, a wider range of segmental durations can be tolerated.

With the above technique, a segment's duration can usually be specified so that concatenation occurs at an acoustically stable portion of the current waveform. In this way, the current waveform can often be made to finish with a perceived pitch or amplitude similar to that of the following waveform. This minimises the probability of waveform discontinuities at segment boundaries, which can result in audible 'clicks'. Thus, by using well-chosen segmental timing data alone, the need for smoothing algorithms at concatenation boundaries is greatly reduced. Segmental duration also plays a major part in the intelligibility of the resulting synthetic speech.

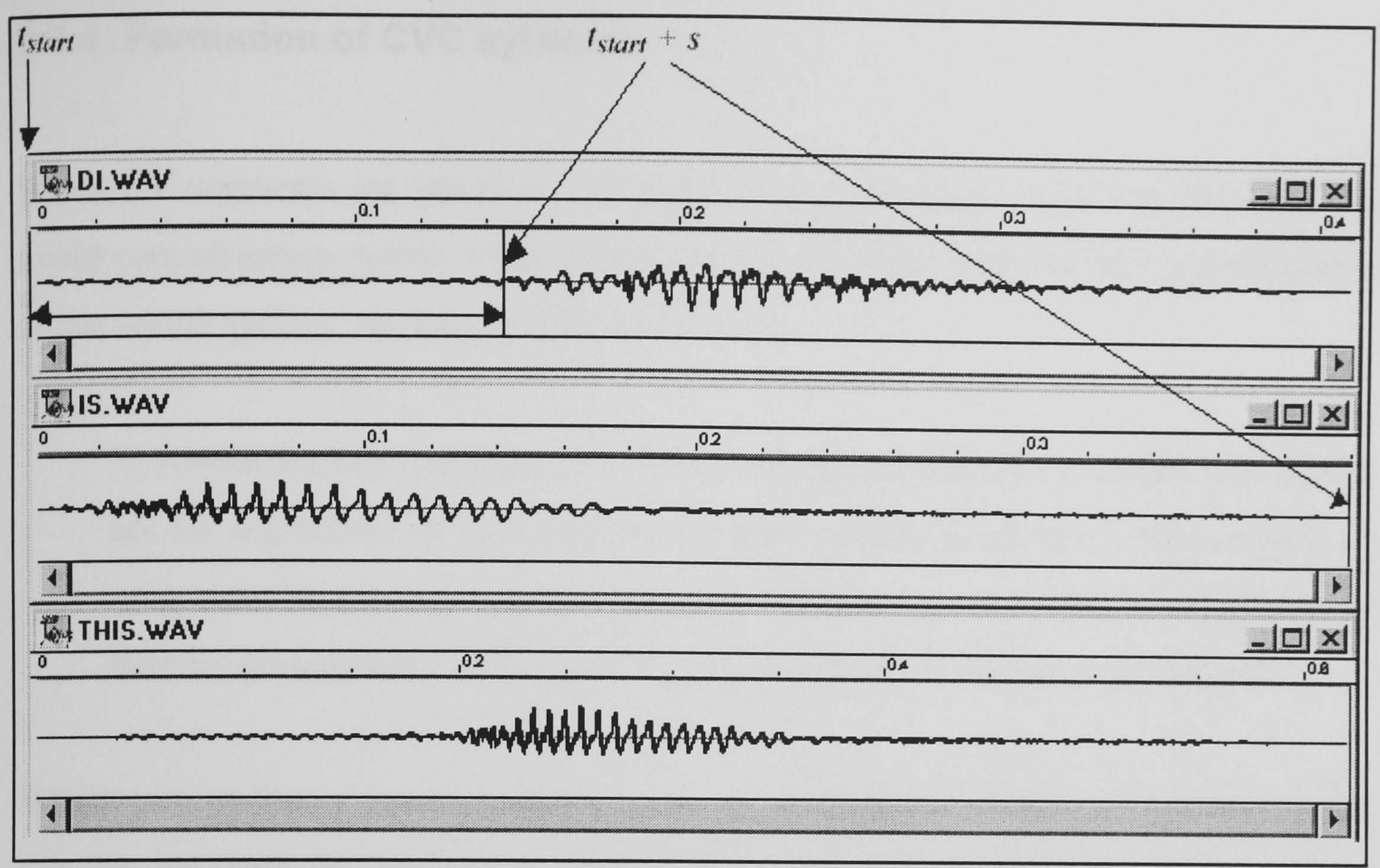


Figure 3.1: Portions of waveforms used to synthesise the word 'this'

The 'initial portion' principle described above has the interesting property that for VC syllables, one can synthesise a speech sound that corresponds to either a word-initial or a word-final position. For instance, if an /Vt/ segment is used in a word-initial position, e.g., at the start of the word 'attract', then the sound corresponding to a plosive burst of air after the /t/ would be undesirable. This plosive burst could be removed from the original /Vt/ segment by reducing the initial portion of the /Vt/ segment to be output. Alternatively, if /Vt/ is to be used in a word-final position (as in the word 'hut'), then the plosive burst is desirable, and so can be produced by using the maximum possible duration of /Vt/. N.B. In this example, the SAMPA notation standard is used. The above is an example of how a stored concatenative unit can be used to fulfil the 'multiallophonic roles' mentioned by Dixon and Maxey (1968).

In concatenative speech synthesis research, extra units such as triphones tend to be added to the waveform inventory (Shih & Sproat, 1996), so that an acoustic unit is available for every conceivable context. However, TDCSS reduces the need to keep adding extra units to the inventory.

3.2.4 Formation of CVC syllables

When CV segments are joined to VC syllables to form CVC units, it is important to avoid concatenation points within vowels where possible, because this is detrimental to the vowel quality. As Saito (1992) points out:

> '...vowels are key components determining synthesised voice quality and should not be segmented or concatenated in their middle at all and concatenation of units must be made in consonantal parts because they are small in amplitude so that the discontinuity introduced by the concatenation sounds less distinct'.

Whilst concatenation points within vowels are sometimes unavoidable, better results can be obtained by making a demi-syllabic cut where the consonant-to-vowel transition ends and the vowel portion begins (Bhaskararao, 1994). This allows for coarticulation effects and enables the listener to anticipate the following VC segment. If this does produce an audible waveform discontinuity, only the consonant portion of the CV syllable should be output.

In creating a CV-VC pair, the initial consonant can accept some degree of duration variation. For instance, a CV segment beginning with a fricative or affricate can take a wide duration range, without the likelihood of introducing waveform discontinuities. For example, the words 'shoot' or 'jig' allow considerable variation in their initial consonants. To some extent, CV-VC pairs starting with sustained sounds such as liquids can also vary in duration without compromising the speech output. Finally, even unsustained sounds such as unvoiced plosives can tolerate some duration variation. For example, experiments with TDCSS have shown that in synthesising the word 'ton', the initial /t/ can have any duration value between 45 and 348 ms.

3.2.5 Concatenation nodes

When CV and VC syllables are concatenated, there is a risk that a waveform discontinuity will result. This problem can be reduced by choosing specific concatenation points in the CV segment's waveform. These points will be referred to

as *concatenation nodes*. Figure 3.2 shows examples of concatenation nodes for the CV segment /m{/. These are labelled C_n, where n is an integer between 1 and 5. The nodes indicate the instants in the 'm' allophone where the signal crosses the time axis. These nodes usually correspond to zero-crossings in the time domain. The arrows show the relevant segmental durations.

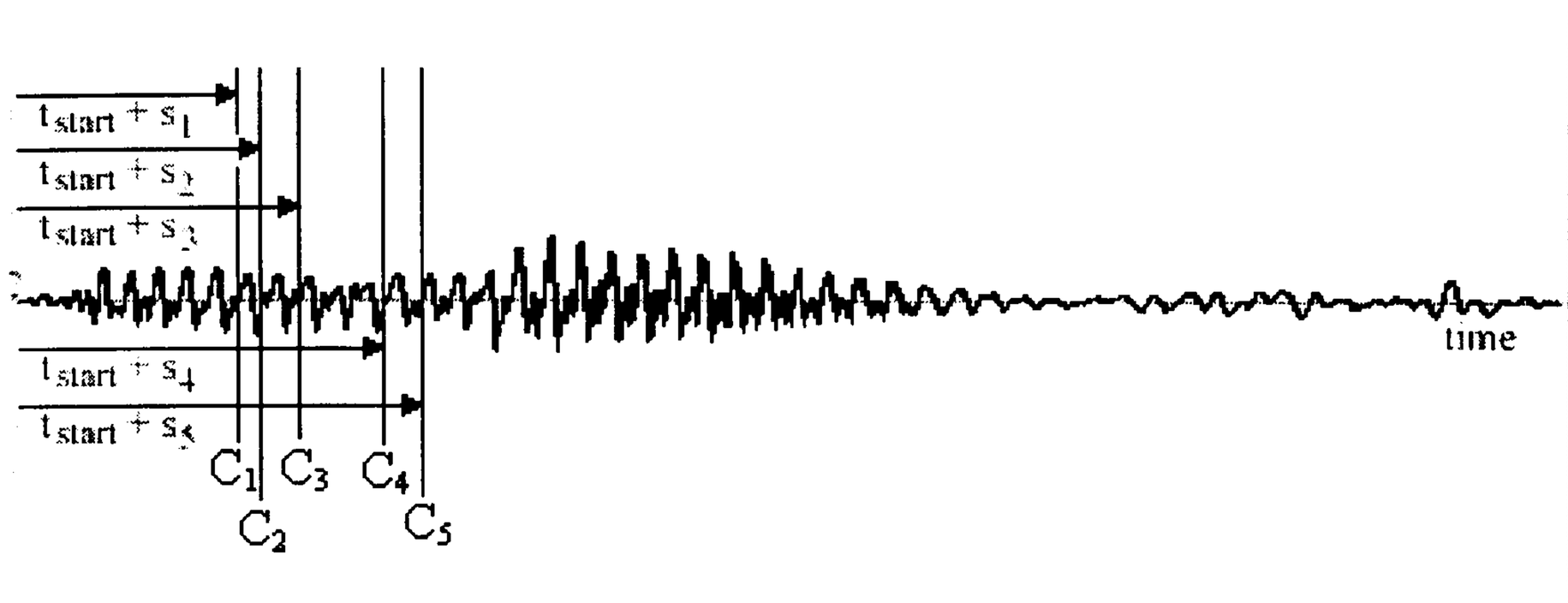


Figure 3.2: Example concatenation nodes for the syllable /m{/

An informal experiment was carried out to verify the efficacy of the concatenation nodes, by synthesising different CVC words which share the same initial CVC segment. For instance, the concatenation nodes for the /m{/ segment were established by visual inspection of the waveforms in the time domain. Initially they were tested via a synthesis of the word 'mat', using a /m{/ segment followed by an entire /{t/ segment. The same concatenation nodes for /m{/ were then tested by synthesising other CVC syllables. These consisted of an initial /m{/ segment, followed by various VC syllables starting with /{/, e.g., /{n/ or /{d/. It was found that for the concatenation nodes shown, minimal waveform discontinuities were produced, compared to CVC segments synthesised using random durations for the initial /m{/ syllable. This finding shows that concatenation points must be well-chosen.

3.2.6 Phonemic category shifts

When concatenating a CV segment followed by a VC, it is highly important to choose an appropriate duration value for the CV segment. Apart from the challenges of

creating realistic coarticulation and minimising waveform discontinuities, one must also consider the segmental intelligibility of the synthesised speech. If the duration of an initial CV segment is shortened too much, a shift is produced in the phonemic category perceived by the listener (Ferrero *et al.*, 1979). For example, if the /ʃ/ sound in the word 'ship' is shortened beyond a certain point, listeners often perceive the word 'chip'.

It is therefore evident that for the TDCSS method, a 'minimum duration' is permissible for each CV segment. Let us assume that the CV will be followed by a VC segment, where the second segment's initial vowel is the same as the final vowel of the CV. A CV segment's minimum duration is defined as *the minimum duration that a word-initial CV segment can be assigned without listeners experiencing a perceptual category shift.* This definition is related in concept to Klatt's (1976) 'minimum duration' term. However, Klatt's concepts of a minimum and maximum duration for a phoneme apply to a different synthesis method in which segments are 'squashed' or 'stretched' to fit a specified duration. This concatenation scheme does not require a maximum duration for segments, since its value depends upon the actual duration of the recorded waveform.

3.3 The Text-to-Segment Conversion Module

For text-to-speech synthesis, text must be converted to a stream of segment symbols which correspond to the stored concatenative units. A Text-to-Segment Conversion module has thus been designed, which converts the input text to a stream of symbols which represent CV, VC, CC and CVC speech segments (see Figure 3.3). This module contains a text-to-phoneme conversion system (see Appendix G), which breaks up the input text into a stream of phoneme symbols, using a set of letter-to-sound rules (Naval Research Laboratory, 1976). NB. The letter-to-sound rules are for American English, and ideally should be converted to British English. The rules consist of four parts: the text to be matched, its left context, its right context and a corresponding list of phonemes. A 'sliding window' is passed across the text, and the text within the current position of the scanning window is compared with a set of rules. If a successful match is found, the corresponding phonemes are substituted for

the windowed text. The rules also contain some text normalisation rules to convert text such as 'Dr.' to 'doctor'.

Since the phoneme symbols produced may not correlate with the stored segments in the inventory, the phonemes are thus converted to CV, VC and CC segments by grouping them into pairs. For example, the word 'cats', is converted from the phoneme stream to CV, VC and CC segments as shown:

k { t s => k{ {t ts

Speech output can be improved by using larger speech segments (Tatham, 1991). This reduces the number of segment boundaries per word and hence the likelihood of waveform discontinuities. The Text-to-Segment Conversion module therefore ensures that for any given text, the largest possible speech segments are selected from the waveform inventory. Where possible, CV-VC syllable pairs are replaced by a single CVC syllable, when available in the waveform inventory.

The Text-to-Segment Conversion module relies upon a Segment Equivalency Table, which contains equivalencies between CV-VC pairs and CVC syllables that are stored in the waveform inventory. For each consecutive pair of syllables in the stream of segments for output, the Text-to-Segment Conversion module searches the Segment Equivalency Table for an appropriate CVC syllable. For example, for the word 'cats', the /k{/, /{t/ pair could be replaced by a single segment /k{t/, if it is contained in the Segment Equivalency Table.

The Text-to-Segment Conversion module currently deals with CV, VC, CC and CVC syllables, and is potentially independent of segment type. At present, the CVC inventory is not exhaustive, but can be further developed. Whenever additional speech segments are added to the waveform inventory, the Segment Equivalency Table is also updated with appropriate equivalencies. This allows the speech waveform inventory to contain a range of segment types.

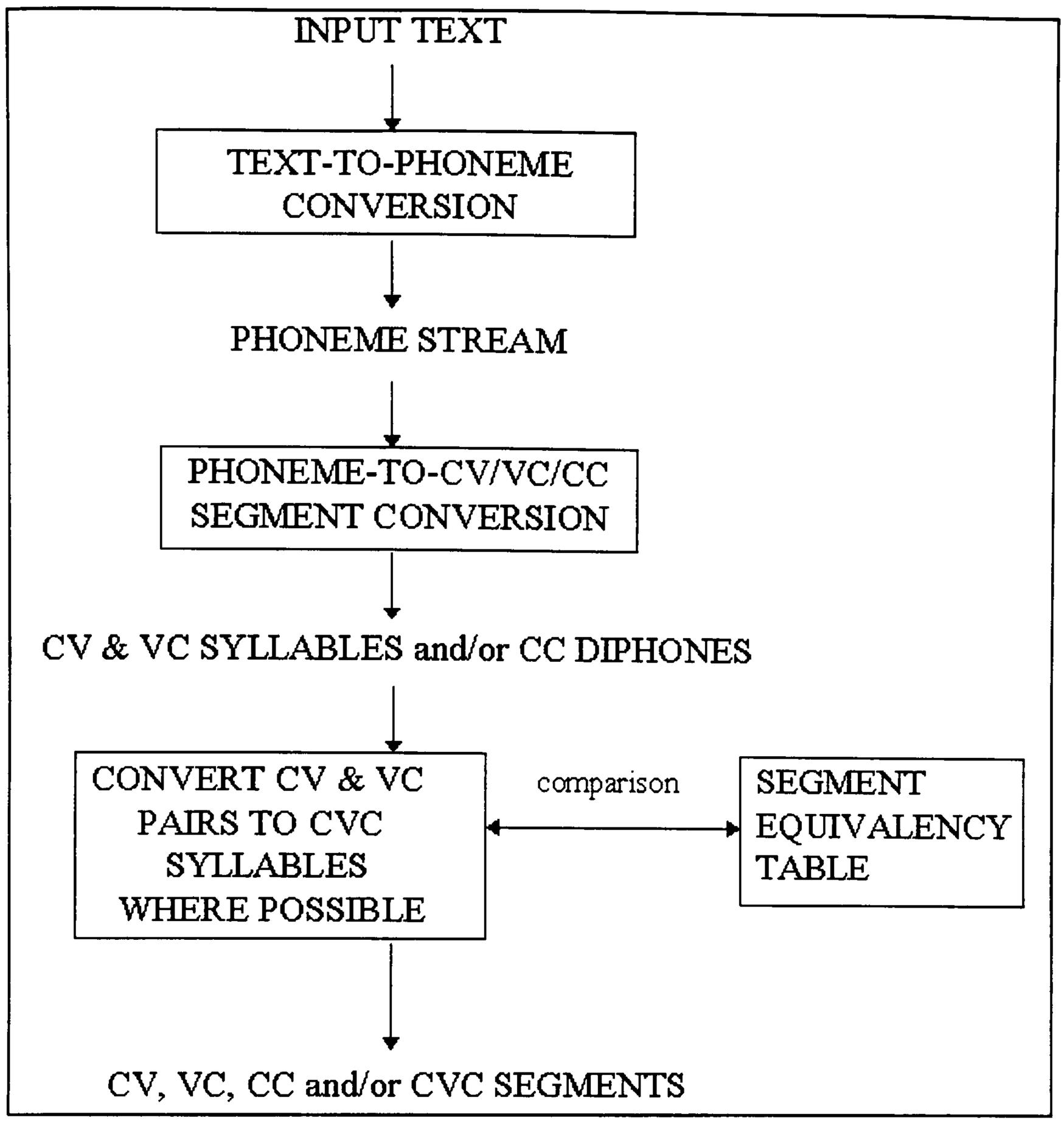


Figure 3.3: Text-to-Segment Conversion Module

3.4 An alternative concatenation framework

This research concentrates upon a concatenation framework in which an initial portion is specified for each speech segment. This framework is referred to in this thesis as TDCSS (Time-Domain Concatenative Speech Synthesis). The speech output finishes at a variable point, after the onset of the segment t_{start}, controlled by a single duration value. Thus, the segmental duration parameter simply determines the 'end point' at which concatenation is to occur.

This one-parameter method (TDCSS) is somewhat constrained in that for each speech segment for concatenation, speech output must start at the onset of the recorded segment, t_{start}. Consequently, every time a VC unit is concatenated, the onset of the

vowel is always output. This is not too detrimental to the synthetic speech's naturalness if the VC unit is situated at or near the start of a word or a foot. However, if a VC unit is not in a word-initial position, the presence of the vowel onset can sometimes make the overall speech sound over-accentuated.

Another possibility is to use a variable 'start point' at which speech output is to begin. This would require two parameters, s_1 and s_2 for each speech segment. Speech would thus be output starting from point $t_{start} + s_1$ and finishing at $t_{start} + s_2$:

$$\text{Speech output per segment} = (t_{start} + s_2) - (t_{start} + s_1) = s_2 - s_1 \qquad \textbf{(3-1)}$$

This two-parameter concatenation framework (referred to in this thesis as TDCSS2) has the potential to improve speech synthesised by waveform concatenation, and makes use of an existing inventory of CV, VC and CVC syllables, supplemented with CC diphones. Some of the key aspects of this framework are discussed below.

The TDCSS2 framework is based on joining CV to VC syllables to form CVC units. As with the one-parameter method, it is advisable to avoid concatenation points within vowels if possible, due to potential discontinuities in pitch or amplitude. Accordingly, it is likely that better results will be achieved using a demi-syllabic cut, incorporating the end of the consonant-to-vowel transition and the start of the vowel portion. If this produces poor quality speech, s_2 can be decreased for the CV segment, so that only the consonant portion is output.

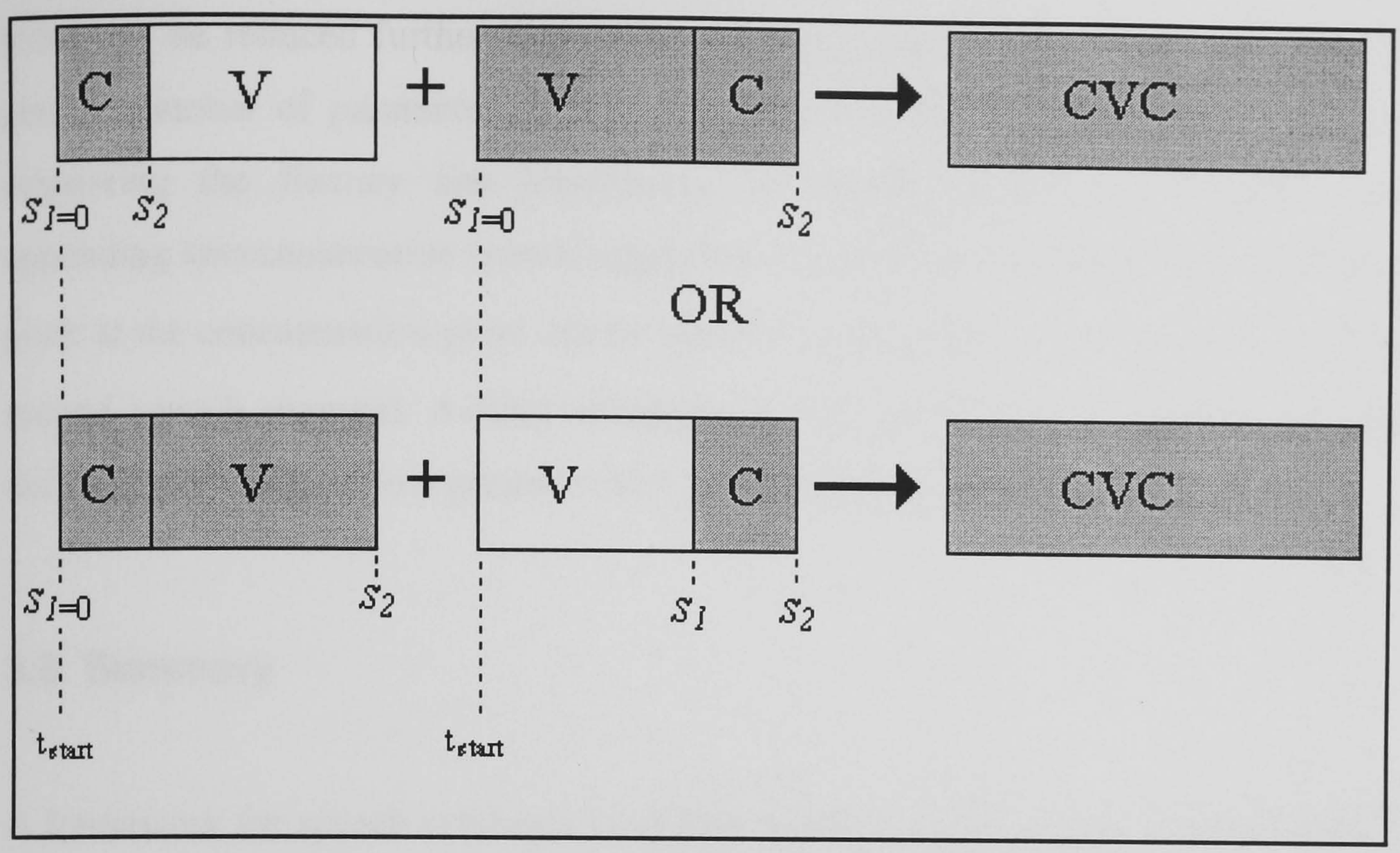


Figure 3.4: Possible strategies for avoiding the splitting of vowels

If the above strategies fail to produce natural-sounding speech, then an alternative is to output most of the CV syllable and to ignore the vowel portion of the VC segment. In other words, by manipulating the parameters s_1 and s_2, the entire vowel can be extracted from either the CV or the VC syllable (whilst ignoring the vowel part of the other syllable). This is expressed in Figure 3.4, in which the shaded areas indicate the portions of the stored CV and VC syllables that are to make up the final CVC syllable.

The TDCSS2 framework requires further investigation into the conversion of text into a segmental representation. It is likely that this would be based on the Text-to-Segment Conversion module of Section 3.3, in which the segmental representation is based on an initial grouping of phonemes into CV, VC and CC units. Chapter 6 provides a brief evaluation of this technique.

3.4.1 Advantages of the two-parameter concatenation framework

TDCSS2 has been found by experiment to have some advantages over the one-parameter framework. Firstly, extra control over prosodic segmental duration changes is available: increased speaking rates are possible, since the overall duration for a

word can be reduced further than under the single-parameter method. Secondly, the greater number of parameters available gives the expert judge a greater chance of improving the fluency and smoothness of speech output. For instance, when appending two consecutive speech segments, it is often the case that discontinuities in pitch at the concatenation point can be reduced by altering the starting point s_l for the second speech segment. A final advantage is that this framework allows consonant durations to be varied to a greater extent, particularly in liquids and fricatives.

3.5 Summary

A framework for speech synthesis via a time-domain concatenation of waveforms has been presented. The framework enables intelligible speech to be synthesised for isolated words by varying a single segmental duration parameter. This parameter specifies a variable, initial portion of each stored waveform, which is to be extracted and concatenated. The segmental duration parameter has several functions. By specifying segmental durations correctly, a natural-sounding duration pattern can be produced for each word. Also, appropriate segmental durations have the effect of minimising waveform discontinuities at the concatenation points between speech segments. This reduces the need for smoothing operations at segment boundaries. A further advantage is that greater use can be made of the existing units in the waveform inventory. By offering a choice of concatenation points for each stored waveform, it can be used in a wider variety of segmental contexts.

An alternative concatenation framework (TDCSS2) has also been discussed, in which both the start and end point of each stored speech segment can be varied. This two-parameter approach has some theoretical advantages over TDCSS, and so in Chapter 6, both frameworks are evaluated.

Chapter Four

4. Segmental Duration Prediction for Time-Domain Concatenative Speech Synthesis

4.1 Introduction

A time-domain concatenation framework for speech synthesis has been described (TDCSS), based on units of speech such as syllables and diphones. Within this concatenation framework, appropriate segmental durations are essential to the overall intelligibility and naturalness of the resulting speech. This is because the segmental duration variable effectively determines the number of speech samples that are extracted from a stored waveform, and concatenated. This enables smoother speech to be produced at segment boundaries. However, if intelligible, natural-sounding speech is to be synthesised from text, a system for predicting appropriate segmental durations is necessary. This chapter explores the prediction of segmental durations for the TDCSS method, from a given text input.

Both rule-based (e.g., Klatt, 1979) and data-driven (e.g., Riley, 1992) systems for segmental duration prediction have been described in the literature. Rule-based approaches tend to require a non-trivial level of text analysis before they can be applied. For TDCSS, it was decided to design a segmental duration system that would not require any text analysis other than a text-to-phoneme conversion. For this reason, a data-driven approach was favoured. This made necessary a database of appropriate segmental timing data.

Initial experiments suggested that it would be beneficial to design a timing database that is closely linked to both the waveform inventory and the TDCSS method itself. This method is somewhat unusual in that more conventional segmental duration prediction systems are designed independently of the intended speech synthesiser. For instance, Dennis Klatt's (1979) segmental duration rules were derived through studying a recording of his own speech, and yet these rules have been applied to a variety of other speech synthesisers, some of which use concatenative units recorded

by other speakers. The above approach also avoids the problem of trying to map predicted phoneme durations onto the speech segments in the waveform inventory, such as diphones or syllables (O'Shaughnessy *et al.*, 1988).

A timing database for TDCSS synthesis has been built, which consists of segmented words along with their corresponding duration values (see Appendix E). An iterative synthesis/adjustment approach, involving an expert judge, was adopted to assign appropriate segmental durations to segmented words. This technique implicitly enhances the speech rhythm through the listening and improving cycle, and goes some way towards incorporating a human's perception of speech rhythm. This is advantageous, because traditional timing systems have no explicit speech rhythm model, and as a result, the speech rhythm may not be correct (Breen, 1992b).

Whilst it would be possible to predict segmental durations using a straightforward look-up method for each word held in the timing database, this approach would only have a limited application. It was thus decided to use the timing database in the prediction of segmental durations for novel words. A pattern-matching technique was adopted which could account for some of the factors used in rule-based segmental duration prediction, e.g., contextual and positional effects on segmental duration.

The duration prediction algorithm operates independently of the types of concatenative unit used, and can even deal with a mixture of segment types. A range of segment sizes and types can thus be used to synthesise speech, allowing great flexibility. This is highly convenient for waveform concatenation synthesis, because its output can be significantly improved through the availability of larger speech segments (Tatham, 1991). The timing database is thus capable of simultaneously storing a mixture of segment types. For example, the word 'connect' can be represented using the segments /k@n/, /ne/, /ek/, /kt/. This word would simply be stored in the timing database, along with appropriate segmental duration data.

The following section describes a system that has been developed for the prediction of segmental durations for whole words, by means of a timing database and a look-up algorithm (see Appendix F for details).

4.2 Segmental Duration Prediction

The segmental duration module developed requires that an input word is first converted into a segmental representation (see Chapter 3). The segmented word is input to the timing system, which uses a timing database to predict appropriate segmental durations. The timing database consists of a list of segmented words, accompanied by a list of corresponding durations for each segmented word (see Appendix E). So, for example, the words 'card' and 'this' could be stored as follows, where the numerical values represent segmental duration in milliseconds:

[[kA:, A:d][106.4, 381.1], [DI, Is][189.1, 396.4], ...].

The duration prediction algorithm works by assembling sub-strings of durations from the database. In this study, a 'sub-string' refers to an unbroken sequence of speech segment symbols (along with their corresponding duration values), taken from a single database entry (this can include an entire database entry). When a list of input segments representing the input word is received, the duration prediction algorithm searches the database for sub-strings that match some or all of the input list. If several sub-strings containing the same segment are found, the prediction algorithm applies a series of three rules to determine which duration sub-string to use in the output list of durations. These rules are described in Section 4.3. If no duration value is available for a particular segment, a default value is used.

The operation of the duration prediction algorithm is illustrated in Figure 4.1, along with a simplified representation of the timing database. The database shows the words 'worms', 'hard', 'cat', 'razor', 'school', 'Peter' and 'lurks' in segmental form, accompanied by their corresponding durations (see Appendix A for details of the segmental representation). To predict durations for the input word 'works', the algorithm looks for all instances of the initial segment, /wur/. This is found in the first database entry, and so the corresponding duration, t_1, is used as the first predicted duration. The final segments (/urc/ /cs/) are found together as part of the final database entry, and so their respective durations are added to the output duration list.

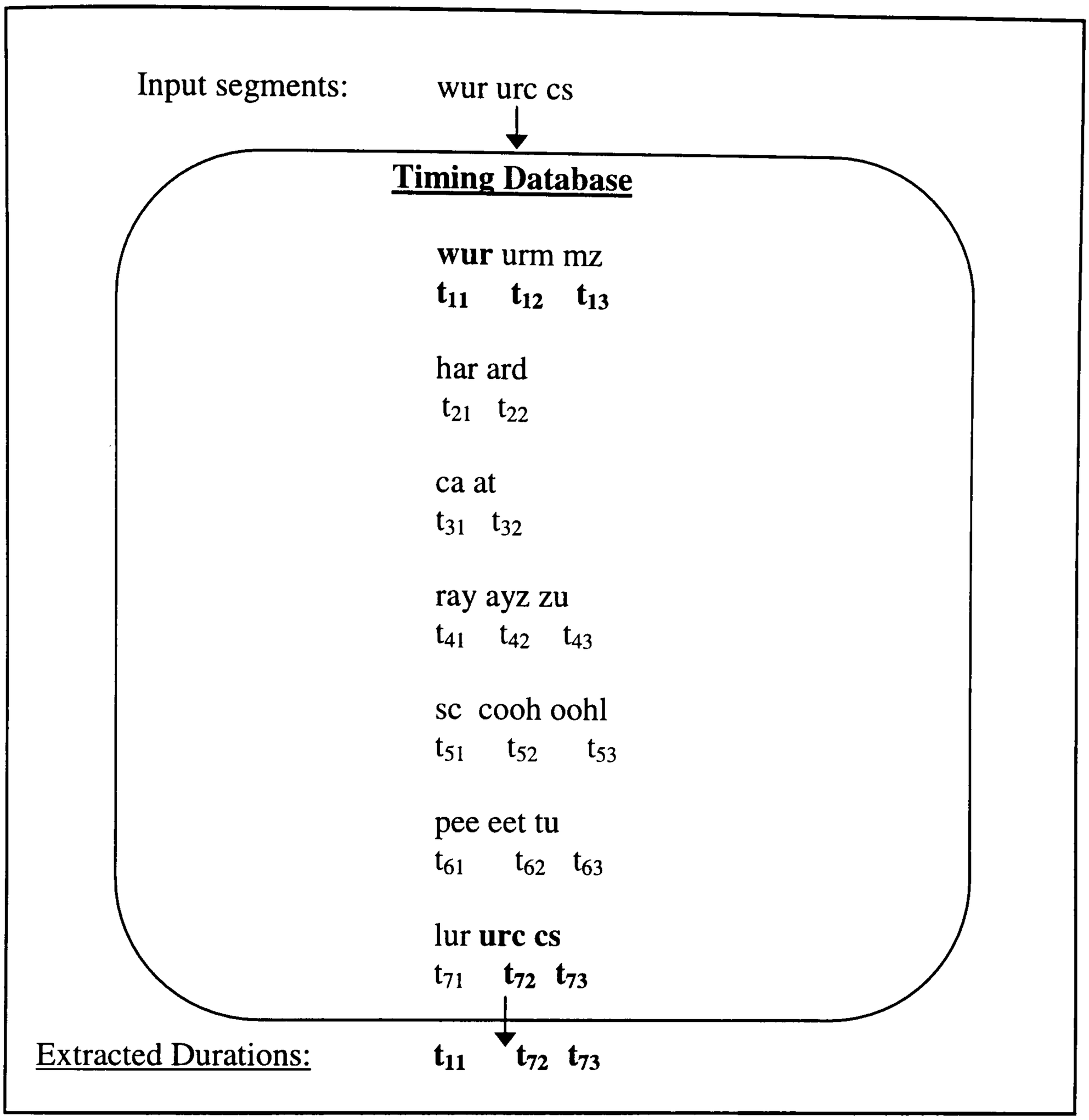


Figure 4.1: Duration look-up algorithm

4.3 Duration Look-up Rules

The following section describes a set of rules used in the duration look-up algorithm. A high-level view of the Timing Module is shown in Appendix B. The main principle is to attempt to find the largest sets of sub-strings available in the timing database. At the same time, the duration look-up algorithm also attempts to take into account the position of segments within a word (see Appendix B).

4.3.1 Rule 1: Largest possible sub-string

The duration prediction algorithm should extract the largest possible sub-strings

from the timing database (where 'largest' refers to the number of segments in the sub-string).

This rule takes into account the fact that a segment's duration is affected by its neighbouring ones (van Santen, 1992). This rule works on the basis that it is more likely that a sub-string taken from one database entry will produce more natural speech than durations taken from a variety of locations. It should be noted that the *largest possible sub-string* rule is similar to Hunt and Black's (1996) approach to the selection of concatenation units from a large speech database. Hunt and Black used a 'concatenation cost function', which encouraged the selection of multiple consecutive phonemes from the synthesis database.

4.3.2 Rule 2: Position in word

If several sub-strings of equal length are found, the sub-string closest in position to the input sub-string is selected.

When searching through large duration databases, it is likely that several sub-strings of equal length may be found by Rule 1. In such cases, the 'position in word' rule is used to differentiate between them. This rule is used because a segment's position within a word has an effect on its duration (Oller, 1973). For instance, the first occurrence of the /I/ vowel in the word 'intimate' has a longer duration than the second /I/. A further example is the well-known phenomenon of final-syllable lengthening, in which the final syllable in a word tends to exhibit a lengthening in duration.

4.3.3 Rule 3: Length of database entry

If rules 1 and 2 fail to arrive at a single sub-string, the sub-string whose database entry length most closely matches the input segment string's length is chosen.

If the first two rules fail to arrive at a single sub-string, then a 'length of database entry' rule is used, since there remain some sub-strings of identical size and position-

in-word. This rule examines the database entries for these remaining sub-strings, and compares the length of their respective database entries. The entry whose length (in terms of number of segments) most closely matches that of the input segment string is noted. The sub-string of durations corresponding to that database entry is then appended to the output duration list.

For example, consider the segmental duration prediction for the input word 'toe', represented by the segment /toh/. It is possible that the timing database may contain the words 'toe' and 'tone'. These could be represented as follows:

Segmented Word:	**Segmental Durations:**	**Word:**
[toh]	[450.0]	toe
[toh, ohn]	[43.5, 420.7]	tone

As can be seen from the above data, if the duration value was taken from the entry for 'tone', the value of 43.5 ms would mean that only the /t/ portion of the /toh/ waveform would actually be output. However, the duration of 450.0 ms (from the entry for 'toe') is very appropriate, since it is more likely that the /toh/ waveform will be output in its entirety, including the vowel portion.

4.4 Default Durations

The Default Duration Procedure is activated if the duration look-up algorithm fails to produce a complete set of segmental durations (see Appendix C). The default duration module accepts a list of the segments that were missing from the timing database as an input. The segments contained in the 'missing segments list' are then re-categorised according to the categories defined in Section 5.3.2.

For example, Table 4.1 shows the input word 'bone' represented by the segments /boh/ and /ohn/, which are input to the duration look-up algorithm. Depending on the size of the timing database, the initial duration look-up may not find a duration value for /boh/. In a default duration procedure, the missing segment is re-categorised as /plosoh/, denoting a plosive followed by /oh/. The duration lookup is then repeated,

using the new sequence of segments as an input. The subsequent look-up then finds a duration value for both of the segments. It is worth noting that it is not adequate to simply look for individual default entries (such as /plosoh/) without considering the entire sequence of segments, because information such as 'position in word' would then be inaccurate.

Input Segments	Resulting Segmental Durations
[boh, ohn]	[(not found), 420.7]
[plosoh, ohn]	[43.5, 420.7]

Table 4.1: Example inputs to the duration look-up algorithm and their possible results

If some segmental durations are still missing after this repetition of the look-up algorithm, then the missing segments are re-categorised further, with less specific segment categories. For example, a fricative-vowel segment such as /frica/ could be replaced by /*a/, where the asterisk represents a 'wild-card' category that can represent any phoneme.

In the above approach, each re-categorised segment consists of a specific part (corresponding to an allophone) and a more general part consisting of a phoneme, categorised by articulation manner (e.g., *fricative*). It is important to note that if a segment such as /at/ is re-categorised in this way, giving the segment /astp/, an assumption is being made that any duration predicted for /astp/ will be appropriate for the segment /at/. This is because durations for different categories of phoneme tend to be similar.

4.5 Segmental duration prediction for TDCSS2

A segmental duration prediction system has been implemented for the two-parameter concatenation framework mentioned in Chapter 3 (TDCSS2). In order to do this, the timing database look-up algorithm required only trivial amendments. This is because the look-up algorithm searches the timing database for speech segment symbols,

rather than segmental durations. Consequently, the time taken for a single duration look-up time remains unchanged.

Under the TDCSS2 method, the timing database only differs in that two duration values are stored for each segment, instead of one. Some example timing database entries for TDCSS2 are shown in Appendix F.

It is not yet certain how the speed of timing database development would be affected for TDCSS2 (see Section 3.4.1). However, it is not necessarily the case that database construction would be slower under such a framework. It is certainly the case that the two-parameter method merits further study, since it retains much of the simplicity of the time-domain concatenation principles evaluated in this research.

4.6 Summary

A timing system has been discussed, which predicts segmental durations on the word level, for time-domain concatenative text-to-speech synthesis. The timing system makes use of a timing database and a duration look-up algorithm which is guided by a small set of rules. The timing system is independent of segment type, and requires only a string of phonemes as its input.

In cases where timing data for a particular speech segment symbol is missing, a default duration procedure is presented, involving the re-categorisation of the segment symbol according to articulation manner. This allows for a graceful degradation in the performance of the segmental duration prediction system.

Chapter Five

5. Timing Databases

5.1 Introduction

In this chapter, the manual method used for building the timing database is evaluated. The timing database is an integral part of the Timing Module (see Appendix E), and is created using an *iterative synthesis/adjustment* technique. This method is a kind of analysis-synthesis cycle, and is based on a human's perception of speech. This method of obtaining timing data is rather different to the more usual practice of using segmental durations based on natural speech. In this way, the phonetic and linguistic knowledge of the expert is incorporated into the timing data produced. There is thus a clear need to evaluate the validity of the expert's perceptive judgement of naturalness. The iterative synthesis/adjustment concept has thus been evaluated via a set of listening experiments and a synthesis/adjustment experiment.

Normally, timing databases are used in the study of segmental duration, and consist of a recording of continuous human speech, segmented into its constituent phonemes and labelled with the corresponding durations (Campbell, 1992; van Santen, 1994). The timing database then forms the basis for producing rule-based or data-driven segmental duration prediction systems (see Chapter 1). For elementary speech synthesis applications, timing databases could be used as a simple look-up table, holding segmental duration data for pre-defined words or sentences. However, simple look-up tables have only a limited application, and so in Chapter 4, a means of using a timing database to predict segmental durations for novel words was suggested.

Since prediction accuracy improves with the growth of the timing database, this chapter includes an investigation into recommended timing database sizes in terms of the segmental duration prediction algorithm's performance. A means of alleviating the problem of incomplete timing data is also described, involving the re-categorisation of existing timing data, via 'default duration' entries. In this way, the

timing database size can be easily expanded, whilst allowing a more graceful degradation in the performance of the duration prediction algorithm.

Most timing databases are dedicated to a particular type of concatenative unit, such as phonemes or syllables (Campbell, 1992). Ideally, both timing databases and timing models should be directly related to the intended speech synthesis units. For instance, if diphones are to be concatenated, then a timing model that predicts diphone durations is most appropriate (e.g., Huber, 1990). This avoids the problem of trying to map the predicted phoneme durations to the diphones for concatenation (O'Shaughnessy *et al.*, 1988). In time-domain concatenative synthesis, it is particularly necessary for segmental duration data to relate directly to the stored concatenative units. This is because a simple abutting method is used, in which segments are not stretched or squashed to fit a target duration value (Dutoit, 1997; Isard & Miller, 1986). It is thus not practical to use labelled continuous speech for the timing database.

An additional difficulty with using segmental durations based on human speech is that even if 'correct' segmental durations are used, the synthetic speech may not be perceived as being natural unless all the prosodic correlates of duration are accurate (Carlson, Granström & Klatt, 1979; Klatt, 1979). The above problem is emphasised by van Santen (1994):

> 'The premise that we should mimic natural durations is not axiomatic because, paradoxically, flaws in other components of a text-to-speech system may require non-natural durations for the system to sound natural.'

Further problems may also arise if the 'natural' durations are based on a person other than the one whose voice was used to produce the concatenative speech segments. A possible solution to the above problems is to listen to the speech produced by the synthesiser itself, and decide whether the speech sounds natural for a particular set of segmental durations. For this reason, the iterative synthesis/adjustment technique was favoured. In effect, this technique produces timing data based on the human perception of the synthetic speech signal. Since the accuracy of segmental duration perception can depend on linguistic experience (Potapova, 1975), it is likely that the

iterative synthesis/adjustment method would benefit from using an expert judge with extensive phonetic experience.

If speech produced on the basis of such timing data is to be judged as natural by any listener, it is reasonable to expect that the expert judge's perception of naturalness should be similar to that of the general listener. Some listening experiments, which investigated this issue, are discussed in this chapter. It is possible that during the timing database development, an expert judge's perception of naturalness may change due to over-familiarity with the synthetic speech (Dixon & Maxey, 1968). Accordingly, this study also attempts to verify whether such a perceptual change occurs with the passage of time.

In judging the qualities of synthetic speech, factors such as intelligibility and naturalness are recognised as desirable, but remain ill defined. Dutoit (1997) has defined intelligibility as being linked to 'the efficiency achieved in producing phoneme-to-phoneme transitions', whilst naturalness is 'the result of the congruence of segmental and supra-segmental features'. Here, supra-segmental features refer to aspects of the speech operating above the segmental level, such as fundamental frequency, amplitude and rhythm.

In this investigation, it was impossible to measure intelligibility and naturalness separately, as defined by Dutoit. This is because in time-domain concatenative synthesis, the duration value effectively determines the choice of concatenation point and, hence, the quality of the synthetic speech, in terms of both speech timing and smoothness of the join between acoustic segments. As a result, the prediction of appropriate segmental durations is more critical than in speech synthesis methods that make use of smoothing techniques.

For the purposes of this study, the term *naturalness* is therefore defined as a general, all-inclusive term, incorporating factors such as smoothness, intelligibility and durational pattern, and is rather like the term *acceptability*, which was defined by Pavlovic, Rossi and Espesser (1990) as 'the user's overall satisfaction with the communication situation'. This definition is reasonably similar to the 'naturalness' concept investigated in this thesis.

The use of an expert judge in gathering timing data relies heavily upon the expert's perceptive judgement of the synthetic speech, and so two complementary experiments have been carried out which investigate the stability and reliability of such judgements. The first investigation consisted of a set of listening experiments, whilst in the second experiment, subjects synthesised words in an adjustment task. In both experiments, the perceptive judgement of a panel of naive listeners was compared with an expert's evaluation.

The listening experiments followed a two-alternative forced-choice paradigm (2AFC). An overview of these experiments is given in Section 5.4.1. Pairs of temporally modified CVC words were presented to subjects, who were asked to decide which version of the word was 'the more natural-sounding'. These choices were then compared with those predefined by an expert judge as the most natural.

The second experiment (see Section 5.5) investigated the perception of naturalness of synthetic speech from a synthesis/adjustment point of view. Subjects were asked to synthesise ten words, given the necessary speech segments. Each subject altered the segmental durations until each word sounded as natural as possible. The objective was to ascertain whether there is a significant difference between the subjects' chosen segmental durations and those of an experienced expert judge. This would then help to establish whether it is feasible to use teams of expert judges, in order to accelerate the acquisition of timing data.

5.2 Timing Database Creation

A timing database was developed to support segmental duration prediction for text-to-speech synthesis. The database consists of segmented words with their corresponding duration values, and is constructed using an iterative synthesis/adjustment approach (see Figure 5.1). When a new word is to be added to the timing database, the word is firstly synthesised using a set of initial segmental durations, chosen according to the expert judge's experience. The segmental durations are then repeatedly altered by an expert judge until the speech subjectively sounds the most natural possible. The segmental durations are subsequently stored in the timing database.

The notion of using an expert judge is linked to Mattingly's (1971) research into the use of speech synthesis-by-rule as a tool for phonological research. Before speech synthesisers were sufficiently well developed, phonological rules had to be tested by a phonologist, who would utter a set of experimental stimuli by natural means. An 'informant' would then judge the acceptability of these utterances, and suggest corrections which would be used to improve the original set of rules. The difficulty with this approach is that the phonologist can not be sure that the speech stimuli were uttered in exact accordance with the rules. Mattingly argued for the use of speech synthesised by rule instead:

> 'Imagine an automatic system, the inputs to which are proposed phonological rules of a language, and a phonemic transcription of an utterance of the language, and the output from which is a synthetic acoustic representation of the utterance. Such a system simulates the phonologist in his generative phase. But it does not make accidental errors, and it applies only rules which have been explicitly stated. A native informant can propose utterances to be generated - or even, if he has learned the transcription, generate the utterances himself - and report to the phonologist in what respects the synthetic versions are incorrect. In difficult cases, the informant can be invited to compare stimuli produced by alternative versions of the rules differing only in respect to the variable of interest. In the light of the informant's responses, the rules can be revised easily and quickly, and the informant can then be confronted with the output of the revised rules.'

A similar technique for 'hand crafting' speech has been mentioned by Morton (1987), who used an 'expert phonetician' to adjust duration and amplitude values in order to add variability to synthetic speech. In this way, speaker attitude and mood can be overlaid onto a synthetic utterance. This finding may well be linked to research which suggests that humans possess an internally-held criterion for perceiving segmental duration in speech (Nooteboom, 1973; 1975). However, in TDCSS, the expert judge's chosen durations can make the difference between the speech sounding 'clicky' and unintelligible, and producing fluent, natural-sounding speech. It may also be possible, that humans possess a common, internal reference for judging the naturalness of speech stimuli.

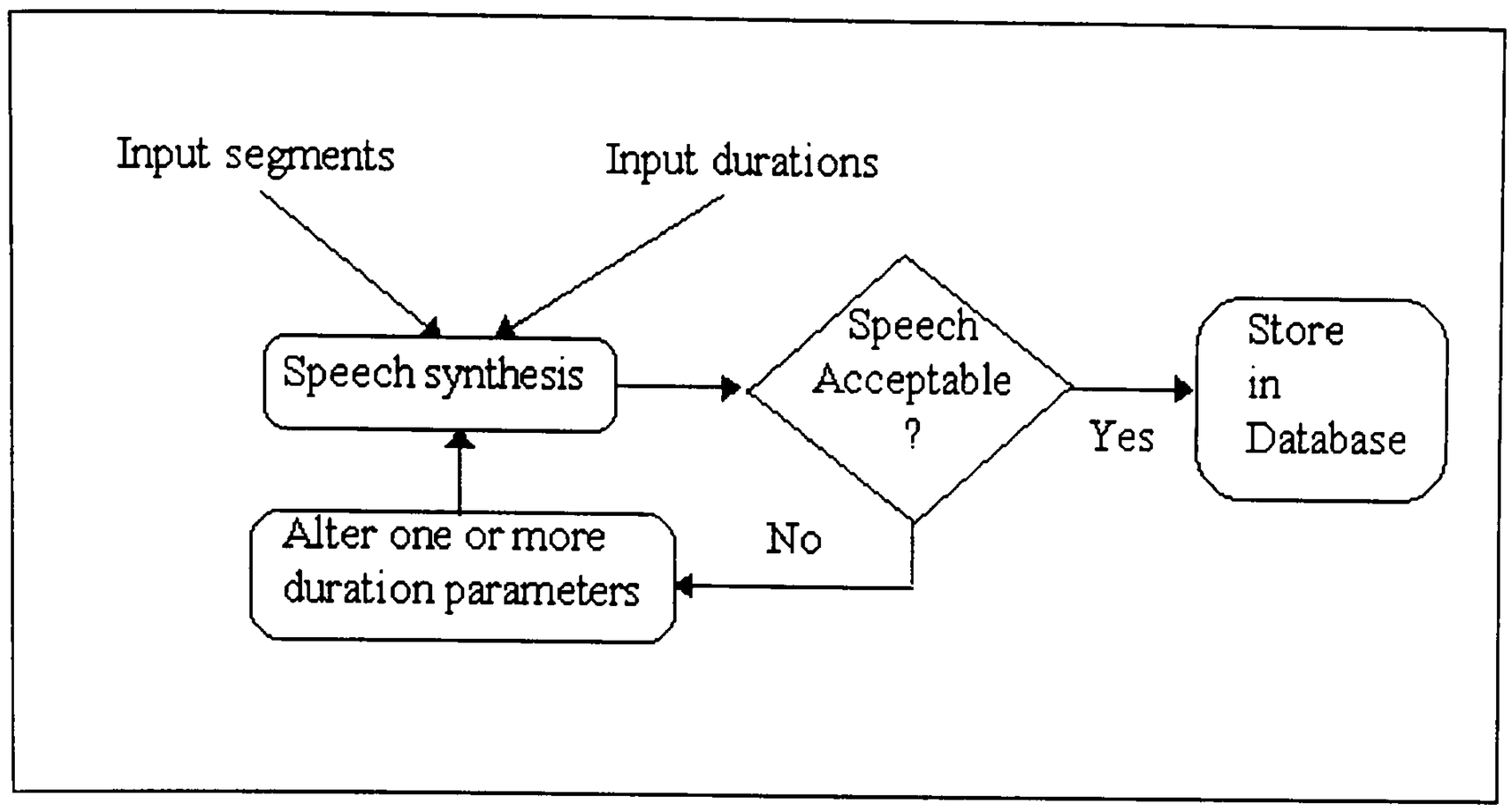


Figure 5.1: Iterative synthesis/adjustment approach

It is likely that the expert judge's knowledge of how to improve naturalness can be formulated in such a way that others can be taught to carry out a similar task and act as an expert judge. According to Morton (1987), it is possible in principle to 'externalize the knowledge that an expert phonetician applies when editing synthetic speech to enable the non-expert to do the editing.' This is particularly important for the waveform concatenation framework under discussion, because this means that teams of expert judges might be employed in timing database construction, thus speeding up the process.

In the context of the iterative synthesis/adjustment cycle, it is instructive to define what is meant by *naturalness* in negative terms, i.e., what would sound *unnatural* to a listener. A synthesised word would be perceived as unnatural if it contains abrupt discontinuities in the form of clicks, caused by abrupt changes in amplitude or fundamental frequency. Furthermore, in TDCSS, if inappropriate segmental durations are chosen, it is possible that the resulting synthetic word may contain repetitions of vowel sounds. For instance, in synthesising the word 'sit', using the stored units /si/ and /it/, the resulting word can sound like 'si-it', if too great a duration value is assigned to /si/.

5.3 Timing Database Content

It is important to choose carefully the words to be stored in the timing database. In making this choice, one must consider the intended speech synthesis application. Such applications could include the synthesis of place names, surnames, weather details, etc. Whichever application is required, a timing database can be custom-built, concentrating on words that are relevant to a topic.

In this study, an ideal aim is to produce a timing database that can be applied to text-to-speech synthesis for any isolated word. It was decided that it would be most appropriate to store the most frequently-used words in English. These were taken in descending order of frequency from the word frequency rank lists of Francis & Kucera (1982).

5.3.1 Timing Database Size Issues

When the timing system is presented with unrestricted text, it is possible that duration data for a particular segment may not be available in the desired context. Therefore, the larger the database size, the greater the likelihood of obtaining an appropriate match. If timing data are absent altogether for a particular speech segment symbol, a default duration must be used. This tends to decrease the naturalness of the output speech. It is therefore imperative that the timing database is of sufficient size.

According to Nation (1993), 89% coverage of a written academic text can be achieved with a vocabulary size of 5,000 words. A timing database of 5,000 words would thus cater for a considerable percentage of the likely text inputs and substantially reduce number of unfamiliar input words encountered.

It has been found during this research that the smaller the sub-strings of durations used when synthesising a word, the worse the naturalness of the speech. This finding can be used to gauge the naturalness of speech produced using TDCSS. This is used as a performance measure in an experiment to examine the relationship between timing database size and segmental duration prediction.

The experiment tested the effect of increasing database size on the duration prediction algorithm's output. The experiment monitored the percentage of an input set of test words for which a complete set of durations was produced by the duration prediction algorithm. Complete sets of durations were defined as a 'complete word match'. The experimental procedure was to take a set of 150 words randomly from a dictionary. The words were then input to the timing system and the number of complete word matches for this set was noted. The number of sub-strings making up the predicted durations for each word was recorded.

The experiment was repeated for every 100 words added to the timing database. Figure 5.2 shows the percentage of all input words, for which a complete word match was recorded. The results show that as more words were added to the database, this percentage increased to approximately 90%.

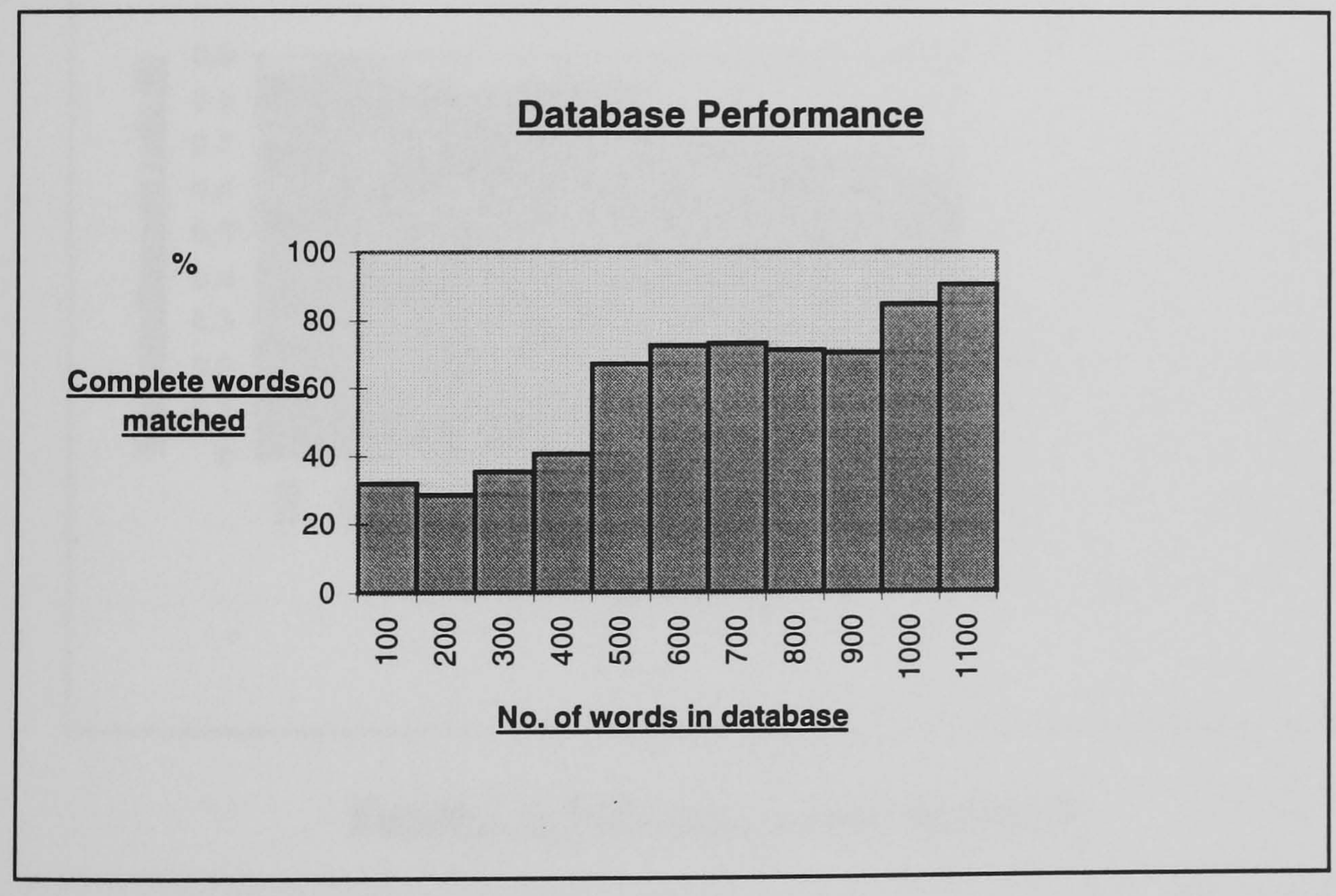


Figure 5.2: Timing Database performance

The speech output quality for a word can be judged in terms of the number of database entries from which its set of durations is taken (i.e., the number of sub-strings that make up the final set of time values). Thus, the database is of better quality if, for a set number of input words, fewer sub-strings are used, compared to

the total number of segment symbols required. Thus, for a test set size of 150 words, Equation 5-1 is defined:

$$R = \sum_{i=1}^{150} \frac{\text{Number of sub-strings required for test word } W_i}{\text{Number of segments required for test word } W_i} \qquad \textbf{(5-1)}$$

The R ratio is shown in Figure 5.3, for increasing database sizes. Figure 5.3 shows an overall decrease in the sub-string/segment ratio R, as the database size is increased. This is in accordance with the trend that a larger database allows better duration prediction.

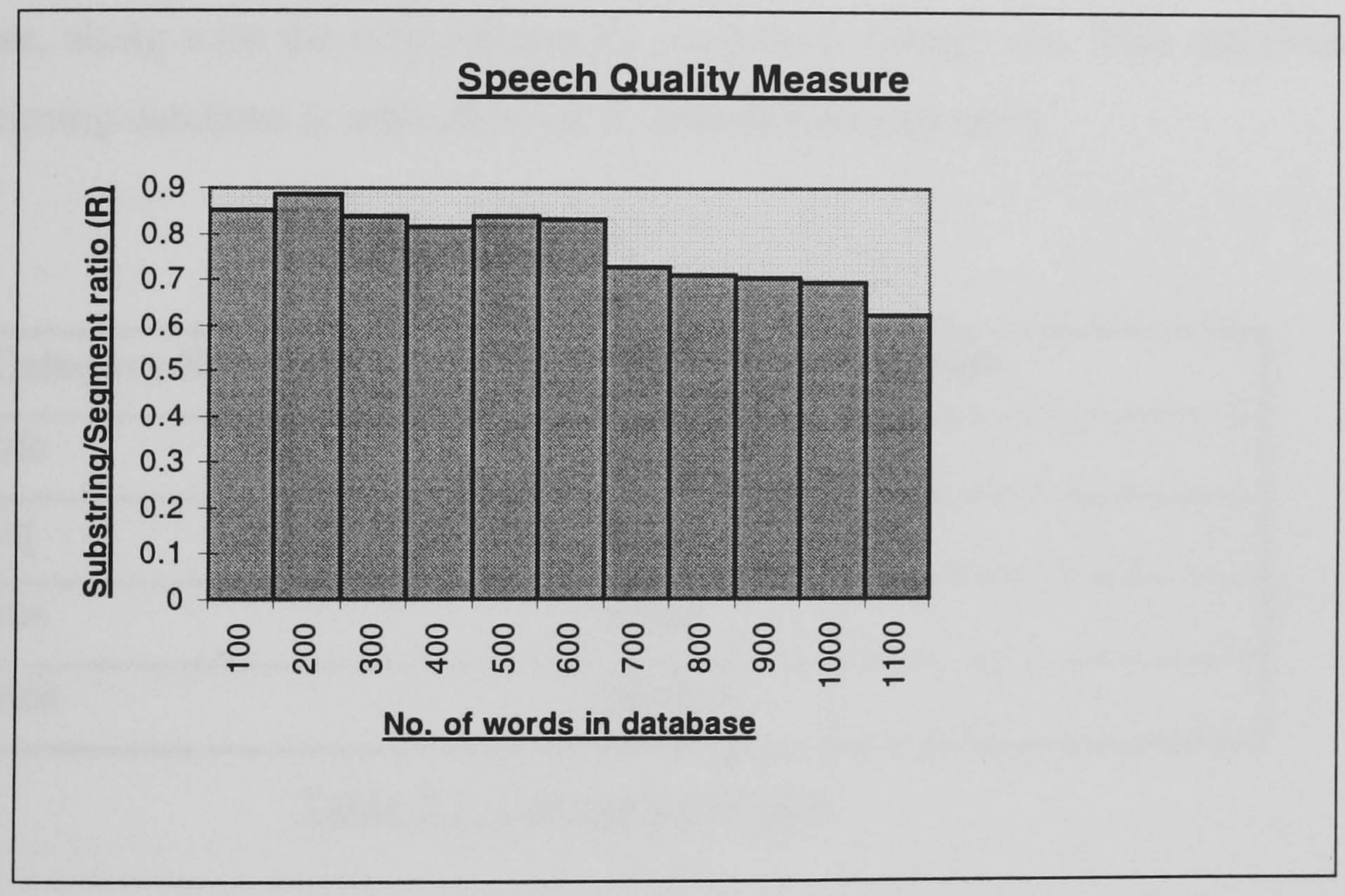


Figure 5.3: Sub-strings/segments ratio R

5.3.2 Using Default Duration Entries in the Timing Database

It is often the case that the timing database will not contain all the required segmental durations for an input word (see van Santen (1992) for an overview of 'database holes'). Whilst a single, default duration value could be used for missing segment symbols, this is hardly an ideal solution, since the default value may not be suitable

for the segment in question, its context, or its position within a word. The default duration approach taken attempts to account for the fact that certain types of speech segment exhibit similar duration properties. For instance, vowel durations tend to be longer (and more critical) than consonant durations (Klatt, 1979; van Santen, 1992). The intrinsic duration properties of the various classes of phoneme (such as stops and fricatives) have thus been exploited.

Whilst no default duration rules are used as such, the method employed takes an existing timing database entry and re-categorises some of its constituent segment symbols according to phoneme class. In particular, consonants are re-categorised according to articulation manner (e.g., liquid, fricative, etc.), whilst the original identity of any vowels is retained. One or more segment symbols in a word can be re-categorised in this way, and then the new version of the word is stored in the timing database, along with the original word's segmental timing data. This additional entry in the timing database is referred to as a 'default duration entry'.

Category Symbol	**Articulation manner**
fric	fricatives
liq	liquids
nas	nasals
plos	plosives

Table 5.1: Category symbols

Any of the symbols in Table 5.1 can be combined with any segment from Appendix A to form a default duration category. For example, the symbol /teye/ is replaced as /ploseye/, where *ploseye* refers to any syllable consisting of a plosive, followed by an /eye/ symbol (here, the naming convention from Appendix A is used). Since this procedure retains part of the original symbol, it can be thought of as a partial re-categorisation, since only the consonants are altered.

The default duration scheme allows a further re-categorisation of default duration segments (such as /ploseye/), in which the plosive category is replaced with a 'wild-

card' character, denoted by an asterisk. In this case, /*eye/ refers to any initial consonant followed by /eye/. Although such default duration entries are less specific (in that they do not distinguish between classes of consonant), they were introduced in order to allow a graceful degradation in the performance of the Timing Module.

An advantage of the 'default duration' approach is that the duration look-up algorithm (see Chapter 4) does not require major modification. The lookup algorithm treats the default duration entries in exactly the same way as any other entry, so that information such as position-in-word is taken into account.

The Default Duration Procedure is shown in Appendix C. After an initial duration look-up procedure, any missing segment is replaced with its corresponding segment category. The new string of segments is then input to the Timing Module. If this search still fails to find complete segmental durations in the timing database then a further categorisation is applied to the missing segment symbols, using the 'wild card' asterisk symbol. Once this second categorisation has taken place, the updated string of segment symbols is input to the duration look-up procedure once more.

Using default duration entries makes greater use of known timing information. In the initial stages of database creation, it potentially allows the number of database entries to be doubled or even trebled with minimal effort. This facilitates segmental duration prediction for a far wider range of input words.

The segment re-categorisation technique allows phoneme classes to exist alongside symbols that represent allophones. Whilst it would be possible to introduce separate categories for voiced and unvoiced segments, the current implementation blurs the distinction between them. For instance, it was decided to treat affricates in the same way as fricatives, working on the assumption that they would tend to require similar segmental durations. This inevitably introduces some inaccuracy into the predicted durations. However, the approach is an improvement on the practice of using a single default duration for any missing segment symbol.

5.4 Evaluation of naturalness in temporally modified words

The quality and naturalness of speech synthesised by TDCSS depend on the use of accurate segmental durations. For text-to-speech, segmental durations are predicted using a timing database and a look-up algorithm. The timing database was developed using an expert judge (DSGV) to 'hand tune' the segmental durations and optimise the perceived naturalness of the speech. To evaluate the expert judge's perception, a listening experiment was carried out, to evaluate the naturalness of speech produced using TDCSS. The motivation for the experiment was to examine the general validity of the expert judge approach, through a comparison with other people's perceptions. The results would also indicate whether it is appropriate to use teams of expert judges to speed up the process of timing database construction.

The experimental stimuli were CVC words, synthesised by concatenating CV and VC waveforms. An expert judge (DSGV) optimised the segmental durations according to his perception. Four additional variants per word were synthesised, varying the CV duration in 20 ms steps (the reason for this choice of time step is discussed in Section 5.4.1.2). In a single pair forced choice experiment, subjects ranked each word's stimuli according to perceived naturalness.

The results indicate very close similarities between the subjects' perception and that of the expert judge (see Section 5.4.5). Furthermore, the expert judge's perception remained stable over a period of time. The implications of these results are discussed in detail. The use of teams of expert judges to accelerate timing database construction, and the development of a support tool for the expert judge are also discussed.

5.4.1 Experiment Series Overview

The listening experiments evaluate the expert judge's subjective perception of naturalness of temporally-modified CVC stimuli. In total, four experiments were carried out, the first being a pilot study. To allow a systematic comparison, five word variants were created for each CVC word (instead of three in the pilot experiment). The pilot experiment required the subjects to hold three stimuli in memory

simultaneously. However, in the subsequent experiments, subjects were presented with pairs of stimuli selected from a set of five word variants. Details of the three experiments carried out are shown in Table 5.2.

Experiment 1	Stimuli presented in a partially random order
Experiment 2	Stimuli presented in a fully random order
Experiment 3	Experiment 1 was carried out, using DSGV as a subject, several weeks after the original reference stimuli (t_{ref}) were created

Table 5.2: Overview of experiments carried out

Whilst the first two experiments compared subjects' preferences with those predefined by an expert judge, Experiment 3 investigated whether the expert judge's perception remained constant over time. This was achieved by using DSGV as a subject in Experiment 1, several weeks after his original reference stimuli were chosen.

5.4.1.1 Stimuli

The speech stimuli consisted of 10 different words, synthesised using TDCSS (Sahandi & Vine, 1997). CV and VC syllables, taken from the concatenative synthesiser's inventory of speech waveforms. All speech waveforms were recorded at a sampling rate of 44,100 Hz, with a 16-bit representation. Each CVC word was synthesised by sequentially outputting a CV followed immediately by a VC segment. Whilst each VC waveform was output in its entirety, each CV segment was assigned a segmental duration value, determining the initial portion of the CV segment to be extracted and concatenated.

The stimuli were produced using an iterative synthesis/adjustment approach, in which the expert judge (DSGV) created a version of each word that subjectively sounded the most natural. Since the segmental duration parameter has a direct effect upon the salience of waveform discontinuities and the durational pattern of the synthesised word, the subjective judgement included all of these factors. The CV segment's

duration in this 'best-judged' word was termed t_{ref}. Four additional versions of the word were then synthesised, two of them having longer CV durations and two with shorter CV durations than t_{ref}. The CV durations for each stimulus (including the best-judged) were termed $t_{ref-2\propto}$, $t_{ref-\propto}$, t_{ref}, $t_{ref+\propto}$ and $t_{ref+2\propto}$ where $\propto$=20 ms. The $\propto$ value was selected on the basis of a pilot study, which is described in the following section.

5.4.1.2 Choice of $\propto$ value

The value of $\propto$ directly affects how easy it is to distinguish between the temporally-modified stimuli for a particular word. It was therefore important to choose an appropriate value for it. For instance, with too small an $\propto$ value, subjects would generally experience greater difficulty in distinguishing between a pair of stimuli; if an excessively large $\propto$ value is chosen, it would become too easy to differentiate between stimuli.

A pilot study was carried out to provide data which would assist in choosing the $\propto$ value. In this study, subjects' perceptions of temporally-modified synthetic words were compared with that of the expert judge. This listening experiment followed a three-alternative forced-choice paradigm (3AFC), and is described in more detail below.

In the interactive pilot experiment, subjects listened to temporally-modified CVC stimuli, synthesised using TDCSS. Three temporally-modified versions of each word were used, and the CVC words were synthesised using duration intervals randomly chosen between 5 and 80 ms. Three versions of each word were presented in a random sequence, and subjects were asked to choose which of these sounded 'the most natural'.

The purpose of this pilot experiment was to identify a range of values from which a suitable $\propto$ value could be chosen for the main listening experiments. Firstly, the words for which all the subjects preferred t_{ref} were examined (see Table 5.3). For these words, the subjects all agreed that t_{ref} was most natural, so it was probably too

easy to distinguish between stimuli. This could be compensated by setting $\propto$ a value less than the smallest value in Table 5.3 (i.e., $\propto$<38.0 ms). Secondly, the average duration interval for the entire pilot experiment was 34.9 ms, so it was decided to use a smaller value for $\propto$ (i.e., $\propto$<34.9 ms). This would increase the similarities between the stimuli and the difficulty level for distinguishing between them. The $\propto$ value could safely be larger than 5 ms, since shorter durations are assumed to be perceptually indistinguishable (Klatt, 1979).

Word Triplet	Average duration interval between word variants (ms)
'been'	53.5
'tin'	38.0
'rap'	72.0

Table 5.3: Words for which all subjects preferred t_{ref}.

The experiment's results can be summarised as showing that $\propto$ can take any value in the range $5 < \propto < 34.9$ ms. Taking the above factors into account, an $\propto$ value of 20 ms was chosen for the experiment series. This value is also supported by Hirsh (1959), who found that to correctly identify the order of a sequence of unrelated sounds, trained subjects require durations of about 20 ms per item. It also agrees with the just-noticeable difference value adopted by Bartkova and Sorin (1987).

5.4.1.3 Subjects

In total, twenty-two subjects participated in the experiments. None of them had extensive experience of speech synthesis. They were not made aware of the overall purpose of the experiment, but were told that the synthetic speech stimuli may contain small temporal modifications.

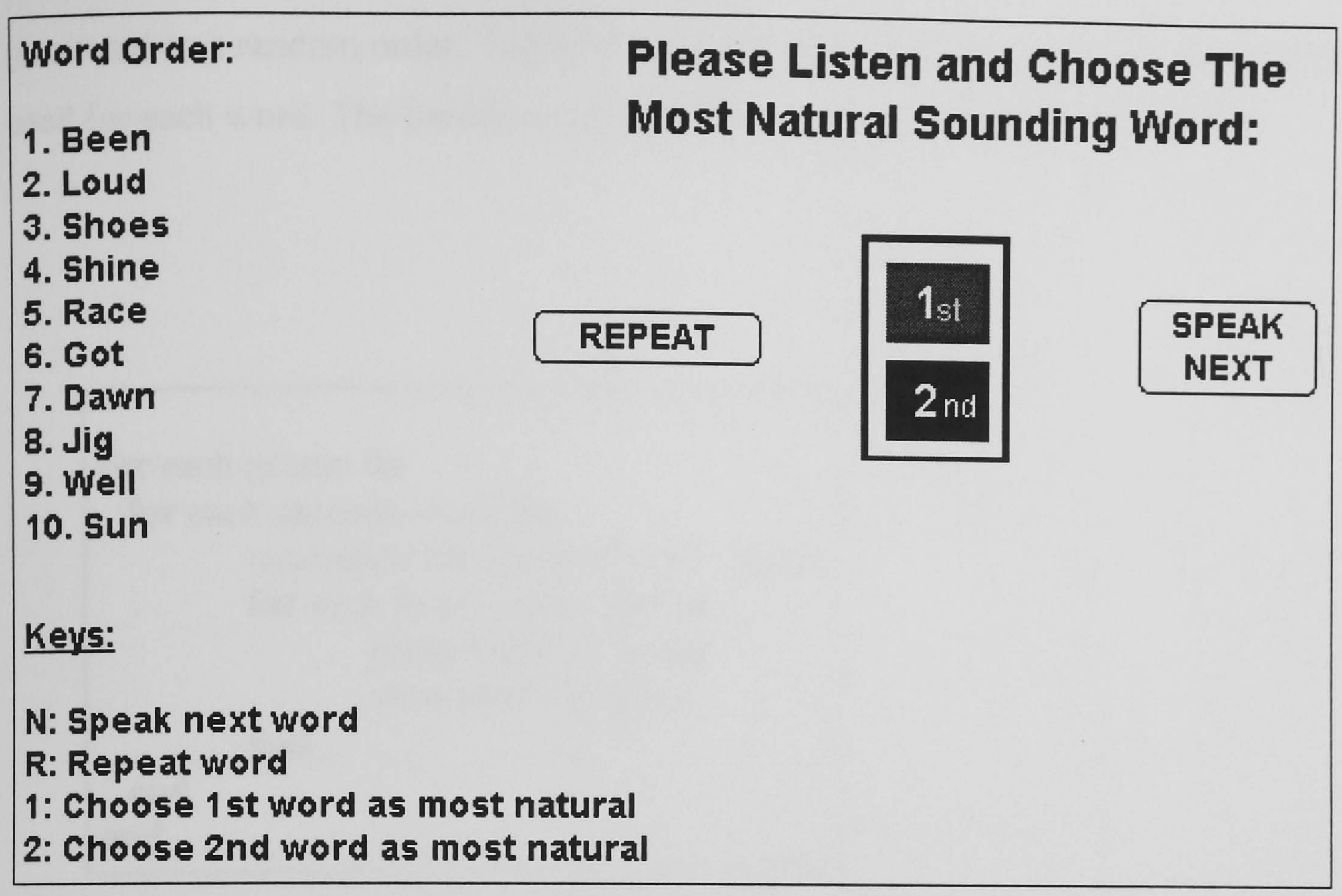


Figure 5.4: Screen layout for Experiment 1

5.4.2 Experiment 1 (randomised for each word)

The experiment followed a two-alternative forced choice paradigm (2AFC), carried out via an interactive computer program (see Figure 5.4). At the start of the experiment, each subject was told that they were to hear a series of pairs of synthetic words; they were allowed to listen to a few randomly-chosen stimuli as examples. They were told that for each pair of stimuli, they should decide which word was 'the most natural sounding'. Subjects were allowed to repeat a stimulus pair as many times as they required. In this thesis, every pair of stimuli presented is referred to as a 'trial'. For each trial, subjects noted their preferred word by clicking a button. In the event that no difference could be perceived between two stimuli, subjects were required to choose one or the other, in order to proceed to the next trial.

Each subject was presented with 14 pairs of variants for each of the 10 different words. For example, 14 trials for the word 'sun' would be presented, followed by 14 trials for the word 'dawn', and so on. For each different word, the stimulus pairs were

presented in a random order. Table 5.4 illustrates the fourteen combinations of stimuli used for each word. The pseudo-code for Experiment 1 is shown in Figure 5.5.

```
for each person do
  for each separate word do
        randomise the pairs of word variants
        for each word variant pair do
              present pair of words
              store user's response
        end
  end
end
```

Figure 5.5: Pseudo-code for Experiment 1

Fourteen English speaking people with normal hearing participated in Experiment 1. Eleven of them were native speakers of English. Details are shown below:

Male:	Female:	Native:	Non-native:
12	2	11	3

5.4.2.1 Stimulus Pairs

During the listening experiments, it would not be practical to present subjects with an exhaustive set of combinations of stimuli, since this would cause considerable boredom and listening fatigue. For this reason, only a sub-set of the possible stimulus pair combinations was used. The trials were chosen in order to cover many of the possible combinations of word variant pairs. The stimulus pairs used are shown in Table 5.4.

Stimulus pairs:			Stimulus pairs (continued):	
$t_{ref-2\propto}$	$t_{ref-\propto}$		$t_{ref-\propto}$	$t_{ref-2\propto}$
$t_{ref-2\propto}$	t_{ref}		t_{ref}	$t_{ref-2\propto}$
$t_{ref-\propto}$	t_{ref}		t_{ref}	$t_{ref-\propto}$
$t_{ref-\propto}$	$t_{ref+\propto}$		$t_{ref+\propto}$	$t_{ref-\propto}$
t_{ref}	$t_{ref+\propto}$		$t_{ref+\propto}$	t_{ref}
t_{ref}	$t_{ref+2\propto}$		$t_{ref+2\propto}$	t_{ref}
$t_{ref+\propto}$	$t_{ref+2\propto}$		$t_{ref+2\propto}$	$t_{ref+\propto}$

Table 5.4: Stimulus combinations

Whilst the above is not an exhaustive set of combinations, it does cover all possible comparisons for word variants that are 20 or 40 ms apart. This limit was imposed in order to cut the total number of trials and thus preserve subject attention.

It should be noted that the number of presentations for each duration bin is uneven (here, the term *duration bin* refers to a particular category of word variant). For instance, t_{ref} is presented 8 times for each different word, compared to 4 observations for $t_{ref\pm2\propto}$. This bias in the results is normalised by dividing the results for each duration bin by N, where N = number of data observations per bin. This is shown in Equation 5-2:

$$\text{result}_{\text{normalised}}(t_{ref+k\propto}) = \text{result}_{\text{original}}(t_{ref+k\propto})/N \qquad \textbf{(5-2)}$$

where k = -2, -1, 0, 1, or 2.

The results for Experiment 1 are discussed in Section 5.4.5.1, and are displayed graphically in Appendix D.

5.4.3 Experiment 2 (fully randomised)

It is possible that in Experiment 1, the results are influenced by subjects experiencing a 'training effect' for each different word as it is repeated for 14 trials, despite the order for each word being randomised. After several repetitions of a word, subjects can lose their ability to judge distinguish between word variants (Dixon & Maxey, 1968). It was thus decided to vary the experiment so that the trial order was completely randomised for the entire test set. The procedure for Experiment 2 is expressed in pseudo-code in Figure 5.6.

```
for each person do
    randomise entire list of possible word variant pairs
        for each word variant pair do
            output pair of words
            store user's response
        end
end
```

Figure 5.6: Pseudo-code for Experiment 2

Apart from a complete randomisation of the trial order, Experiment 2 was carried out in a similar manner to Experiment 1, with the main purpose of demonstrating whether noticeably different results would be produced. In this experiment, eight subjects were used. None of these subjects had taken part in Experiment 1. Six of them were native English speakers. Details are given below:

Male:	Female:	Native:	Non-native:
5	3	6	2

The results for Experiment 2 are discussed in Section 5.4.5.2, and are shown graphically in Appendix D.

5.4.4 Experiment 3 (expert judge as subject)

Since it may take several weeks to develop a timing database, it is important that the expert judge's perception of naturalness should remain constant throughout the development period. Experiment 3 therefore examines whether the expert judge's perception remains reasonably constant after a period of time. This was investigated by using the expert judge as a subject, participating in Experiment 1, several weeks after the t_{ref} stimuli were designated. The expert judge (DSGV) is a male, native speaker of English, with normal hearing.

5.4.5 Results

Graphical representations of the results for Experiment 1, 2 and 3 are shown in Appendix D. For each experiment, the assumption was made that whenever t_{ref} was chosen, the subject agreed with the expert judge's original preference. Choices for $t_{ref\pm\alpha}$ indicate that the subject's perception of naturalness differs to a small extent from that of the expert judge. Wherever $t_{ref\pm2\alpha}$ is preferred, a larger difference in perception is indicated.

In interpreting the results, it is assumed that the closer a duration category is to t_{ref}, the more similar the subject's perception of naturalness is to the expert judge's. This is not necessarily true in all cases, since in this study, the perception of naturalness is affected by more than one factor. For instance, in terms of smoothness of speech output (e.g., lack of 'clicks'), any stimulus may sound reasonably fluent if the CV segment's concatenation point corresponds to an appropriate zero-crossing in the time-domain. In terms of perceived similarities in overall durational pattern, the 'closeness to t_{ref}' criterion seems reasonable.

The results for each experiment were converted to a single time value, representing a time distance measure, relative to t_{ref}. This was achieved by multiplying the mean preferences for each duration bin by an appropriate constant and summing the result,

as shown in Equation 5-3:

$$\text{Time distance measure} = \sum_{k=-2}^{+2} 20k(\text{mean preference}\{t_{ref+k\alpha}\}) \qquad \textbf{(5-3)}$$

(where k= -2, -1 , 0, 1, 2)

The time distance measure is intended to indicate a general trend when averaged across all the samples, and enables comparisons between the results for the different experiments.

5.4.5.1 Results and Discussion: Experiment 1

The overall results for Experiment 1 take the form of a bell-shaped curve, with t_{ref} receiving the greatest number of choices for every single subject. Figure D.1.1 (see Appendix D) shows the average preferences for each duration bin, whilst Figure D.1.2 shows the average preferences for each CVC word. Figures D.1.3 to D.1.12 show the average preferences for the individual words. They show a normally distributed curve for 8 of the 10 words tested in the experiment. The results are less clear for the words 'Shine' (Figure D.1.6) and 'Dawn' (Figure D.1.9). In Figure D.1.6, $t_{ref+\alpha}$ is judged the most natural by the majority of subjects, for the word 'Shine'. Figure D.1.9 has less clear results overall. It is possible to interpret the results as showing a shallow normally distributed curve, centred on $t_{ref-2\alpha}$. It is also possible that no clear preferences emerge for this stimulus, since the results are fairly even for each duration bin. This result may be due to changes in the subjects' perception of naturalness as the experiment progressed. The overall results, however, are reasonably consistent, showing that each subject's perception of naturalness is largely similar to that of the expert judge.

For some words, a number of subjects favoured $t_{ref\pm2\alpha}$. These cases would seem to indicate that there are substantial differences between the expert judge's perception of

naturalness and that of the subjects. It should be emphasised, however, that some CVC syllables (such as those beginning with fricatives) can accept a considerable range of duration values for the CV segment, without affecting naturalness (see Section 3.2.4). Furthermore, due to the unpredictable nature of speech data, it is possible that occasionally, a word variant based upon $t_{ref \pm 2\propto}$ could by chance have a smooth-sounding transition.

In order to interpret the results further, a statistical test was used to examine the relationship between the results for Experiment 1 and the t_{ref} values set by the expert judge. The Wilcoxon signed ranks test (Siegel & Castellan 1988; Clarke & Cooke 1983) examines whether there is any difference between two related sets of scores and establishes whether there is a significant difference between the two sets of results. In other words, the Wilcoxon signed ranks test reveals whether the population median of the paired differences of the two samples is 0. This test has the advantage that a normal distribution of data is not assumed.

The Wilcoxon signed ranks test showed that for a T value of 26, with N=14, there is a significant difference between the results for Experiment 1 and the t_{ref} values, with a two-tailed probability of 0.05. A probability level of 0.05 was used, because according to Clarke and Cooke (1983):

> '...there is a well established working rule or convention, used by statisticians generally, which is to regard events having probability 0.05 or less as 'very unlikely''.

The interpretation of this result is that although the subjects agreed overall with the expert judge, they exhibited a wider spread of responses, i.e., the expert's results were tidier than those of the subject population.

Using Equation 5-3, the time distance measure is -1.62 ms. From this we conclude that on average, the subjects sometimes preferred stimuli with slightly shorter CV durations than t_{ref}. The time distance measure is very small when compared with the $\propto$ value of 20 ms and a possible range of 80 ms.

To summarise the results for Experiment 1, the subjects' averaged results agree with those of the expert judge, but the variance of the distribution of percepts is larger.

5.4.5.2 Results and Discussion: Experiment 2

The graphical results (fig. D.2.3 to D.2.12) show that t_{ref} was favoured over the other duration bins. Overall, figures D.2.1 and D.2.2 illustrate a normal distribution for the experiment, demonstrating a good agreement between the subjects' perception of naturalness and that of the expert judge.

A Whitney-Mann U test (Siegel & Castellan, 1988) was carried out in order to ascertain whether there is a significant difference between the results from Experiment 1 and those from Experiment 2, due to the different randomisation methods used. The Whitney-Mann U test was chosen because it reveals whether two independent groups have been drawn from the same population, and avoids the assumption that the experimental data are normally distributed. The Whitney-Mann U test produces a U value, which is a count of the total number of times each observation of one sample comes before each observation of the other sample. A U value of 29 was produced, showing that there is no significant difference between the results produced under either randomisation method (with a two-tailed probability of 0.05).

A Wilcoxon signed ranks test (Siegel & Castellan, 1988) was carried out comparing the results for Experiment 2 with the t_{ref} values set by the expert judge. This showed that for a T value of 4.5, with N=8, there is a significant difference between these values, with a two-tailed probability of 0.05. Again, this is interpreted as showing that although the subjects showed overall agreement with the expert judge, they submitted a wider spread of responses during the experiment.

For Experiment 2, the time distance measure is -0.96 ms. This shows that subjects' preferences were even closer to t_{ref} than the results for Experiment 1. A similar

conclusion is drawn for Experiment 2: the subjects' averaged results agreed with those of the expert judge, but the variance of the distribution of percepts was larger.

5.4.5.3 Results and Discussion: Experiment 3

For this experiment, the majority of the expert judge's preferences coincided with t_{ref}. The overall results are shown in figures D.3.1 and D.3.2. These figures demonstrate that the $t_{ref \pm 2\propto}$ categories were seldom preferred by the expert judge, and for the majority of trials, t_{ref} was perceived to be 'most natural'. Taken as a whole, the results demonstrate that the expert judge's judgement of naturalness for the temporally-modified words had remained largely stable as time passed. This is in keeping with the requirement that for timing database development, the expert judge's perception of naturalness should remain stable over time, since the construction process may take several months.

A Wilcoxon signed ranks test (Siegel & Castellan, 1988) was carried out, comparing the original t_{ref} values with the overall duration value for each word. This produced a T value of 12.5, with N=10; from this we conclude that there is no significant difference between the expert judge's t_{ref} choices and the results for Experiment 3, with a two-tailed probability of 0.05.

This result is in keeping with the research of Pavlovic, Rossi and Espesser (1990), who found that the 'acceptability' of synthetic speech can be accurately assessed by seasoned researchers, even after many years (see Section 5.1).

5.4.6 Conclusion

The overall trend for the set of listening experiments (Experiments 1 and 2) is that the subjects' averaged responses (in terms of perception of naturalness) agreed with those of an expert judge. However, as might be expected, the subjects gave a larger spread of responses. Furthermore, the subjects' choices only differed from the expert judge's

preferred durations by a mean value of -1.29 ms. In Experiment 3, it was shown that the expert judge's perception remained largely constant over several weeks.

No significant difference was found between the results of Experiments 1 and 2 due to the different randomisation methods used (with a probability of 0.05). However, when the subjects' responses were compared with the expert judge's chosen stimuli, a significant difference was found for both Experiments 1 and 2 (with a probability of 0.05).

Overall, one can conclude that when averaged over a large number of samples, the inexperienced listeners' perception of the most natural stimuli agreed with that of the expert judge, but the variance of the distribution of the percepts was larger.

This empirical work is important in that it validates the use of an expert judge in the assessment of synthetic speech naturalness in an iterative synthesis/adjustment process. The listening experiments' results showed clear similarities between inexperienced listeners' perceptions of waveform concatenation speech and those of an experienced, expert judge. The finding that the expert judge's perception remained stable over time is partly related to Nooteboom's (1973, 1975) study, which showed evidence that humans have an 'internal auditory representation' for syllable nucleus durations. This study, however, deals with the perception of *naturalness*, rather than segmental duration *per se* (even though the naturalness is strongly linked to segmental duration).

The experimental results are critical to the large-scale development of timing databases using an iterative synthesis/adjustment method. This approach is labour intensive. However, the similarity of the expert's judgements to those of the subjects in the experiments means that when scaling-up the approach, it will not be necessary to extensively validate an expert's judgement against naïve listeners' perceptions. This, in itself, will considerably reduce the development effort of such speech synthesis systems. At this point, however, it is important to state that these findings only apply specifically to TDCSS, used with a particular waveform inventory.

The results justify the development of a tool to support an expert judge. This could present temporally-modified word variants to the expert judge, in a version of the 2AFC paradigm. The experiments have shown the 2AFC paradigm to be very sensitive, to a point far below the perceptible difference levels (e.g., about 2 ms). For instance, statistical tests detected significant differences (at the 0.05 probability level) between the first two experiments' results and the t_{ref} values set by the expert judge. This finding shows that a 2AFC paradigm may be appropriate in such a support tool, reducing the effort required to hand tune each word, and accelerate the acquisition of timing data for waveform concatenation speech synthesis.

In the following section, the 'expert judge' concept is investigated from a different angle, in which subjects take part in an iterative synthesis/adjustment task.

5.5 A comparison of segmental durations produced by different expert judges

If timing data for TDCSS is to be produced on a large scale using a team of expert judges, one must ascertain whether different expert judges have a similar perception of naturalness, and whether they carry out the iterative synthesis/adjustment task in a similar manner. An experiment was thus carried out to investigate this issue, complementing the listening experiment described in Section 5.4. However, instead of listening to pre-synthesised stimuli and recording their preferences, subjects played a direct part in generating speech themselves, adjusting segmental durations according to their own judgement of naturalness. Subjects were initially shown how to synthesise words according to the concatenation framework in Chapter 3. Subjects were then given the task of synthesising a set of isolated words with no assistance. The resulting segmental durations were compared with durations preferred by a more experienced expert judge (DSGV) for the same set of words.

The motivation of this experiment was thus to evaluate the following hypothesis:

> *'There is no significant difference between segmental durations preferred by the subjects and those preferred by DSGV'.*

This would help to establish whether a person can be trained successfully to act as an expert judge, and generate acceptable segmental durations for synthetic speech. It also shows whether an expert judge can manually synthesise natural-sounding speech, using a waveform inventory which is based on another person's voice.

5.5.1 Subjects

Six English-speaking subjects with normal hearing were used, who had not participated in any experiments involving synthetic speech. Five were native speakers of English; the sixth was a native speaker of Arabic. Five subjects were male and one was female. The female subject had a mild Scottish accent, whilst the native English-speaking males did not have strong regional accents. Subjects were academic researchers, aged between 20 and 35.

5.5.2 Method

Subjects were given basic training by the author in the synthesis of words using an iterative synthesis/adjustment method and an existing speech segment inventory. This training included a demonstration of the rudiments of the concatenation framework described in Chapter 2. The training period lasted approximately 30 minutes. All words were synthesised by means of simple cut-and-paste techniques, using the PC-based *Autodesk Soundlab* program to display time-domain speech waveforms. A variable, initial portion of a CV or VC syllable would be copied and pasted into a new sound file. Further speech segments would then be abutted by pasting them and removing the intermediate silence. If the resulting speech was judged inadequate by the subject, the word would be re-synthesised using a modified set of segmental durations.

For the actual experiment, it was thought likely that the task of cutting and pasting speech waveforms would prove too difficult and lengthy a task for subjects. Consequently, an experienced assistant was provided, who synthesised words according to the durations specified by a subject. Subjects were only allowed to give

instructions of the form 'increase/decrease the duration of the ___ segment', which would then be carried out for them. No other conversation with the assistant was allowed during the experiment. Subjects chose the initial segmental durations for each word arbitrarily, and were given no advice on the choice of segmental durations to use.

Subjects were given the instructions shown in the following section.

5.5.3 Instructions

Please synthesise the following words using the speech segments shown in the table below (the files are situated in the directory c:\diphones). Cut and paste techniques will be necessary to assemble the word. When synthesising a word, only the *initial portion* of any segment should be varied. If a synthetic word does not sound natural, alter this portion for one or more of the speech segment files and re-assemble the word. Repeat this process until the word sounds natural.

You should attempt to synthesise the word as if it was produced in isolation, rather than in continuous speech.

In order to speed up the synthesis task, an assistant will be provided, who will carry out any adjustments to the segmental duration values that you request. Requests should be made to the assistant in the following format:

'Please increase/decrease the duration of '

When you are satisfied that the synthetic word is the most natural-sounding possible, write down the duration value for each speech segment (this is found at the bottom-right corner of a speech window).

BOURNEMOUTH
UNIVERSITY
LIBRARY

Word	Constituent Segments	Segmental durations (ms)
three	thr ree	
most	moh st	
ask	ahs sc	
without	wi ith owt	
waveform	way ayv for orm	
speak	sp eec	
must	mu us st	
single	si ing gl	
never	ne ev vu	
stop	st op	

5.5.4 Results

The results for the iterative synthesis/adjustment experiment are shown below. Figure 5.7 shows the mean segmental duration per word, chosen by the subjects and DSGV respectively. This value is directly proportional to the overall length chosen by the expert judge for a word. It can be seen that the subjects' mean durations approximately follow those of DSGV.

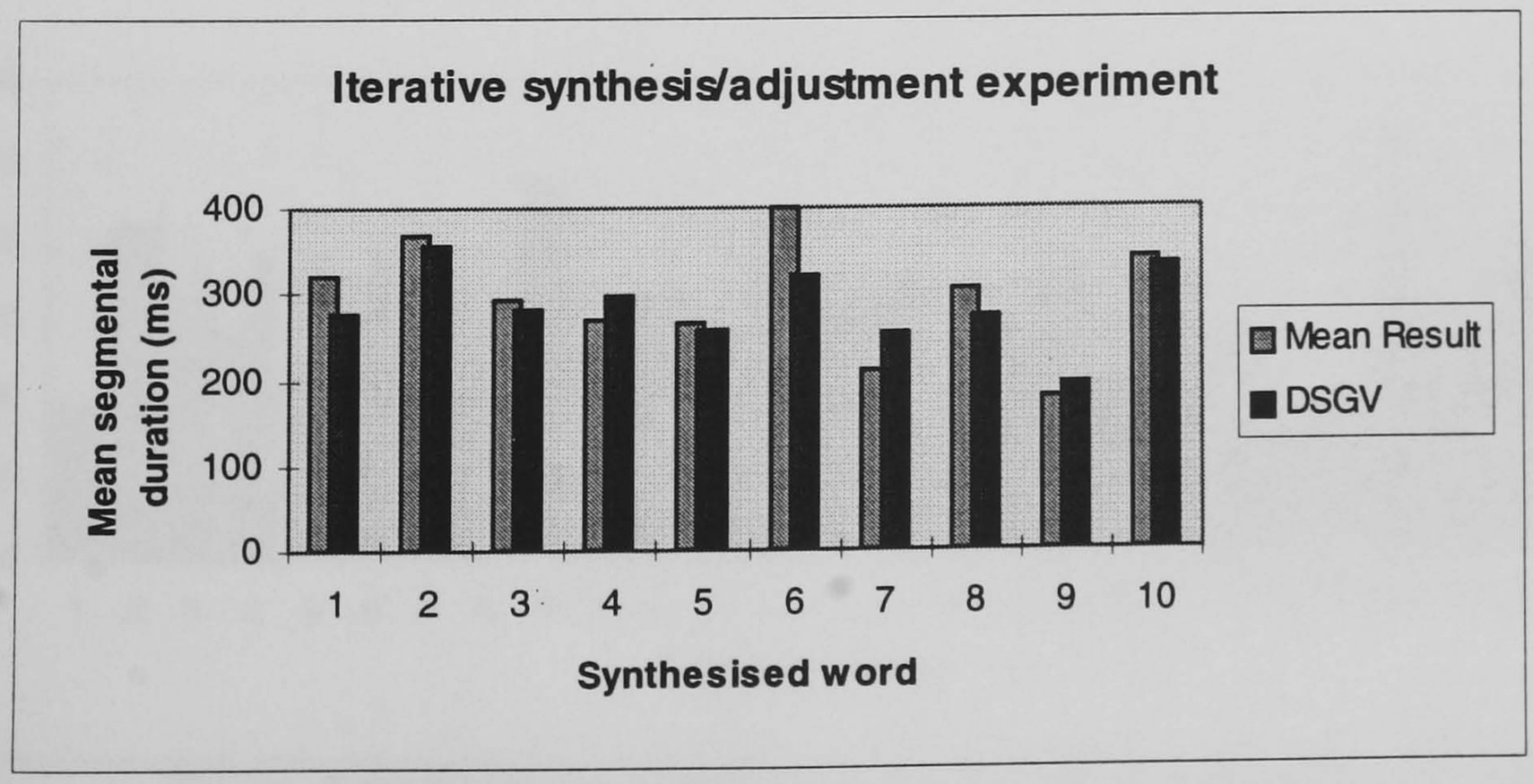


Figure 5.7: Mean duration value per word

Figure 5.8 shows that the mean segmental duration per word chosen by the author lies between the values chosen by the other subjects. This also demonstrates that TDCSS provides some flexibility in terms of overall length of word, despite the fact that this technique does not allow speech segments to be stretched or squashed (see Section 3.2.3).

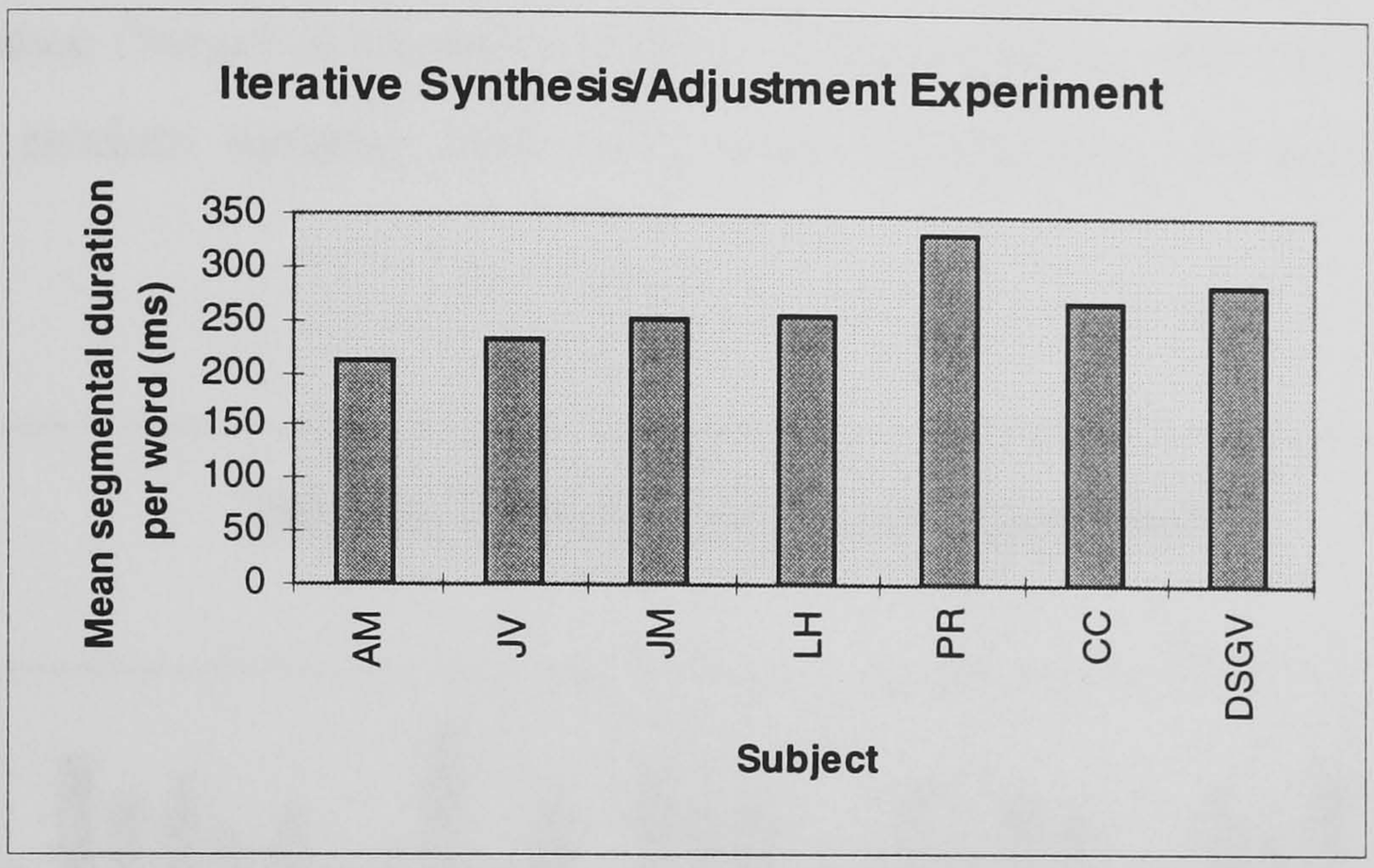


Figure 5.8: Mean segmental duration per word

Figure 5.9 shows the ordered, consecutive segmental durations for the entire set of synthesised words. The initials shown in the legend refer to the subjects' names. This is complemented by Figure 5.10, which shows the mean segmental duration chosen by the subjects compared with segmental durations preferred by DSGV.

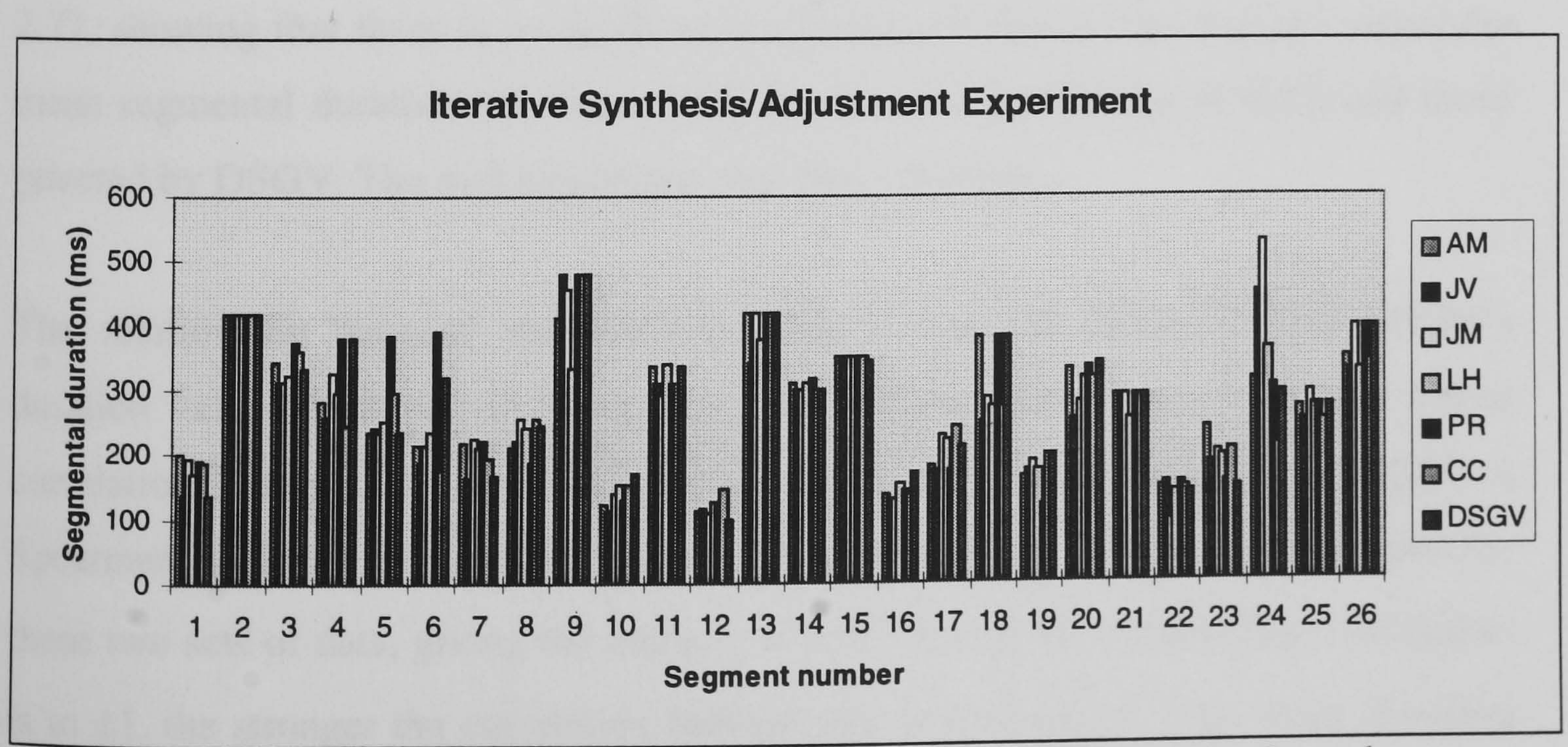


Figure 5.9: Segmental durations for the set of synthesised words

A Kruskal-Wallis test (Siegel & Castellan, 1988) was carried out in order to establish whether there is a significant difference between the groups of scores, using the mean segmental duration per word. The Kruskal-Wallis test decides whether k independent samples are from different populations, with respect to averages. Since differences between sample values usually exist, this test establishes whether the differences among the samples are genuine population differences, or whether they have merely arisen by chance (Siegel & Castellan, 1988). The test assumes that the data arise as k independent random samples from continuous distributions, all having the same shape.

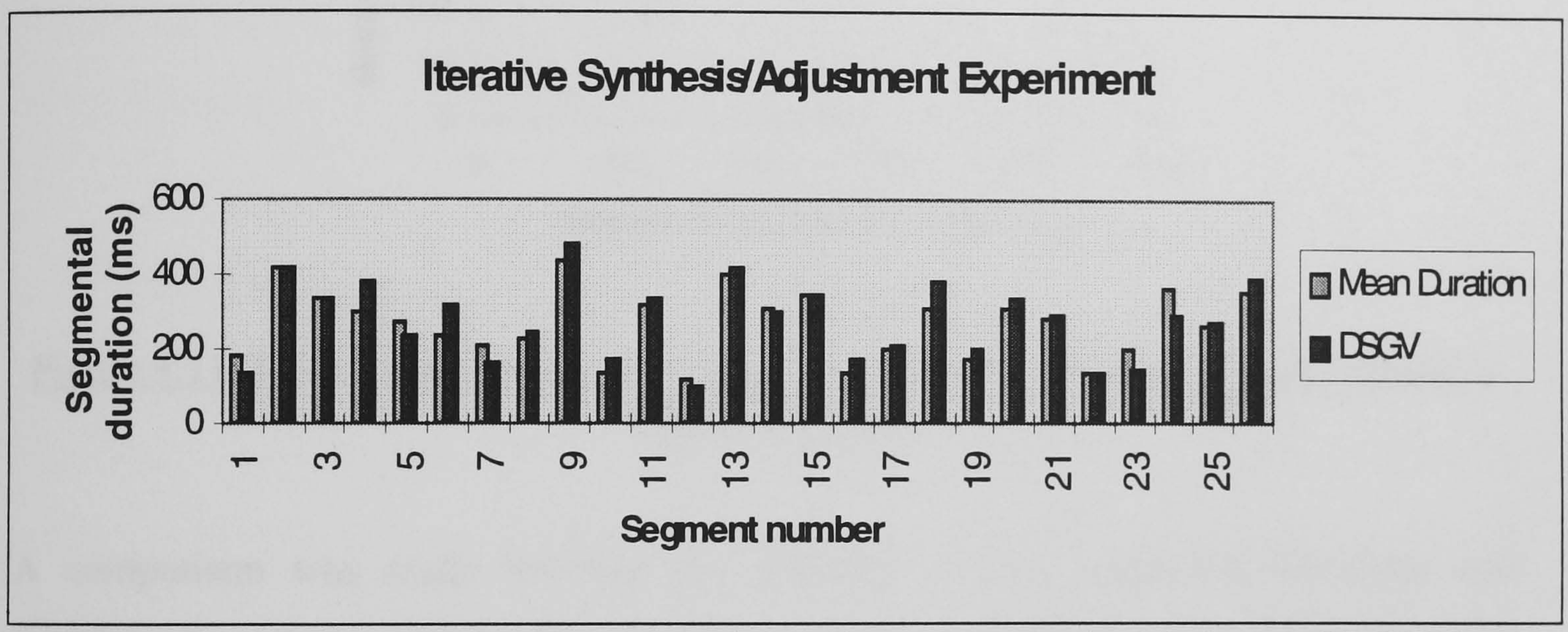


Figure 5.10: Subjects' mean segmental durations compared with those of DSGV

The null hypothesis was that there is no significant difference between the subjects' results and those of the expert judge. The Kruskal-Wallis test yielded an H value of 2.77, showing that there is no significant difference between the subjects' values for mean segmental duration per word (with a two-tailed probability of 0.05) and those selected by DSGV. The null hypothesis was thus validated.

The relationship between the subjects' mean segmental durations and DSGV's duration values is shown in Figure 5.11. This scattergram shows a positive linear correlation between the subjects' mean values and those selected by DSGV. A Spearman Rank Correlation Coefficient (Siegel & Castellan, 1988) was calculated for these two sets of data, giving the value $r_s = 0.90$. The closer a correlation coefficient is to ± 1, the stronger the correlation between the sets of values. This value therefore

confirms that there is a strong correlation between the subjects' mean segmental durations and DSGV's duration values.

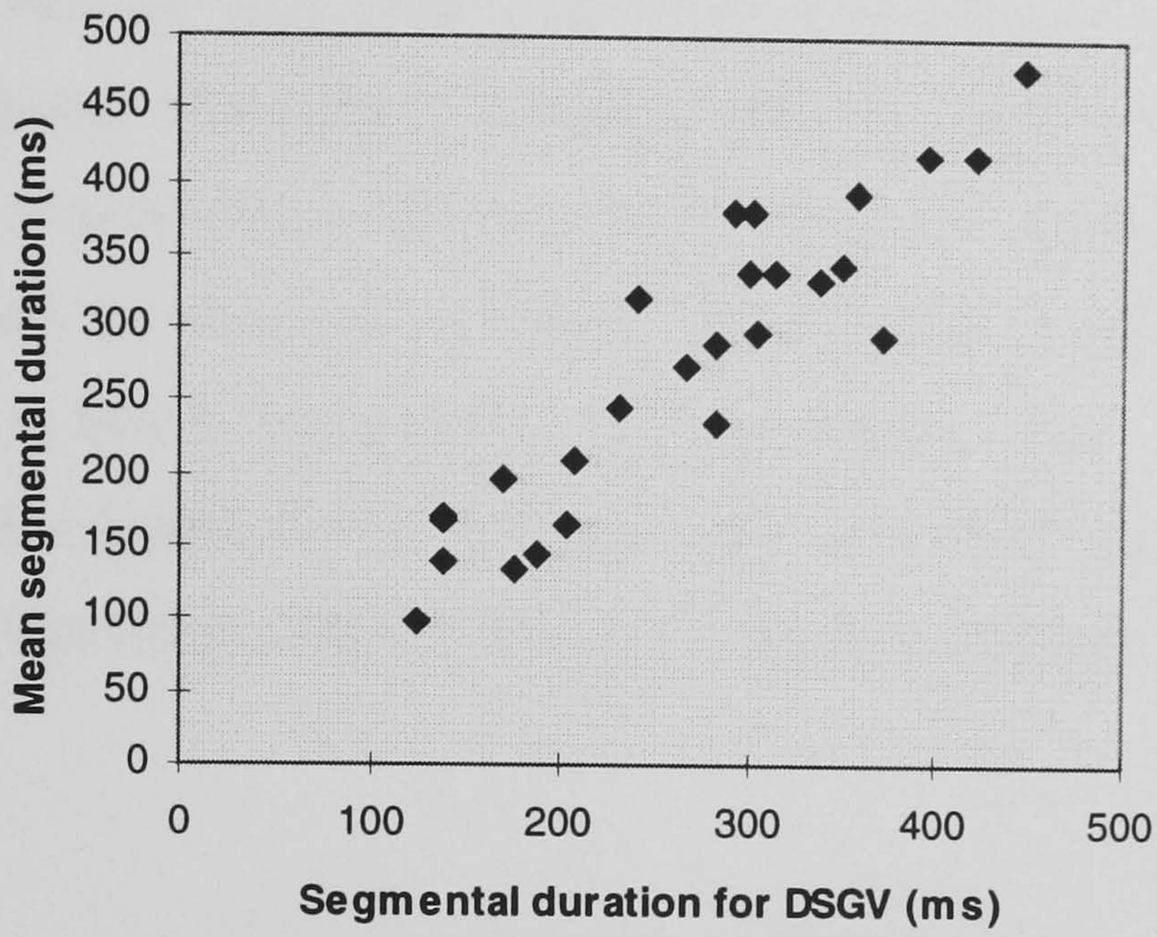


Figure 5.11: Relationship between subjects' mean segmental durations and DSGV's reference values

A comparison was made between the subjects' chosen segmental durations and DSGV's. In order to achieve this, DSGV's values were treated as a 'reference' set of results. A set of mean segmental durations was then calculated across the subjects, which was then subtracted from the reference results. The mean results produced by the subjects differed from the reference values by a standard deviation of 37.36 ms. This approach is similar to that taken by Klatt (1979), who compared segmental durations predicted by rule with spectograms of continuous speech, read by himself. Klatt obtained a standard deviation value of 17 ms.

5.5.5 Discussion

On an intuitive level, there are noticeable similarities between the subjects' chosen segmental durations (see Figure 5.9). This lends some support to the idea that inexperienced subjects could act as expert judges, and choose similar segmental duration patterns for TDCSS. Furthermore, no significant difference at the 0.05 probability level was found between the different sets of values. This is impressive,

considering that the experiment gives the subjects total freedom to differ in their responses. Also, since subjects were synthesising words in isolation, it was all the more likely that they would differ in their perception of the required word duration.

It should be noted that the experiment could have been designed in such a way as to increase the likelihood of subjects producing uniform-length words. This could be achieved using an experimental format in which the word for synthesis is embedded in a carrier sentence, recorded by the person upon whose voice the original speech segment inventory is based. An example sentence might be 'Say _____ again' (Klatt, 1979). However, this approach was avoided in this experiment, because the subjects' chosen segmental durations could be influenced by the speaking rate inherent in the carrier sentence.

5.6 Support tools for timing database development

The task of the expert judge is to listen to synthetic speech and adjust a word's segmental durations until the speech is judged to sound the most natural. This task is potentially time-intensive, so a support tool to minimise the 'hand-tuning' time per word is desirable. This could be achieved by a system that produces an 'initial guess'.

At present, the duration look-up algorithm itself is used as a support tool, providing an initial segmental duration prediction. A list of speech segment symbols is input to the look-up algorithm, and the resulting segmental durations are used as the starting point in the iterative synthesis/adjustment cycle (see Figure 5.12). This accelerates the timing data acquisition process.

A possible support tool which may be investigated would use a two-alternative forced choice (2AFC) paradigm, and would present pairs of temporally-modified stimuli to an expert judge. The expert judge would decide which of the stimuli sounded the most natural. A further pair would then be presented on the basis of the expert judge's choice. In this way, the number of segmental duration adjustments would be reduced. A current goal for such a tool could be to reduce the hand tuning time for each word to under a minute.

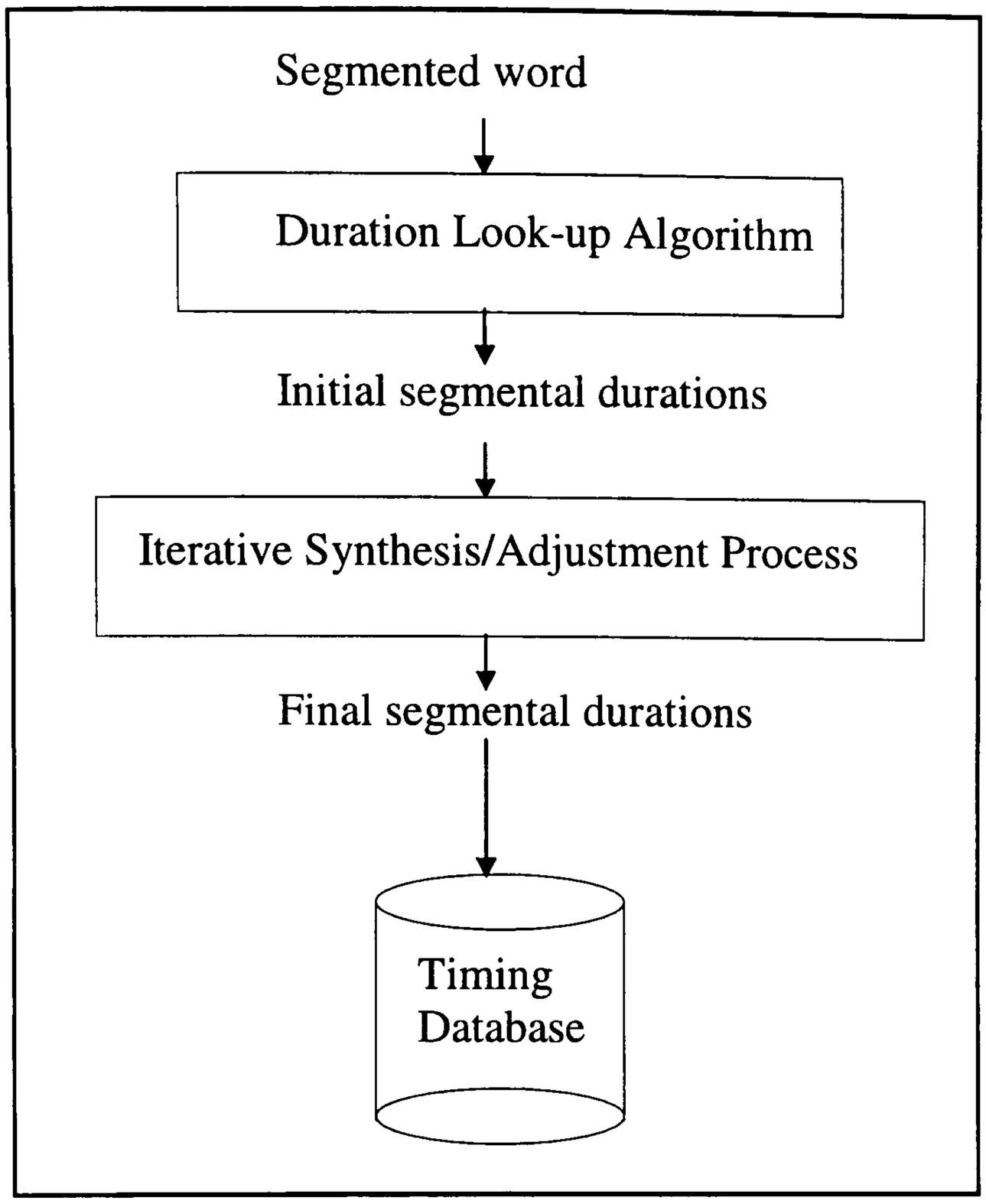


Figure 5.12: Initial segmental duration prediction

5.7 Summary

A timing database has been created for use in segmental duration prediction for text-to-speech. The database consists of segmented words with their corresponding duration values, and is constructed using an iterative synthesis/adjustment cycle. This cycle involves assigning segmental durations to a word, synthesising it and repeatedly altering the segmental durations until the speech subjectively sounds the most natural possible.

The person used to judge the synthetic speech output and adjust segmental durations is referred to as the 'expert judge'. This approach merited investigation, since several questions are likely to be raised. For instance, how do other listeners perceive the segmental durations chosen by an expert judge? Are the expert judge's preferred segmental durations representative of the general listener's perception of naturalness?

These issues were investigated using a set of listening experiments and a synthesis/adjustment experiment. The listening experiments investigated the perception of naturalness in temporally-modified CVC stimuli, and showed that on the whole, the subjects' preferred stimuli matched those preferred by an expert judge. In an adjustment experiment, subjects acted as expert judges, in synthesising a set of ten isolated words. It was found that their mean segmental durations differed from the author's chosen values by a standard deviation of 37.36 ms.

Since the task of obtaining timing data is labour-intensive, it is likely that occasionally, duration data may be unavailable for a particular segment, making it necessary to use a default duration value. This is not an ideal situation, since the default value may well be inappropriate for the segment's context. An improved default duration mechanism has been introduced which relies on the re-categorisation of existing timing data, thus creating new database entries. In this way, timing database sizes can be increased more easily. This method only requires minor modifications to the duration look-up algorithm for segmental duration prediction. The possible use of support tools for expert judges has also been discussed.

Chapter Six

6. Evaluation of Time-Domain Concatenation Speech Synthesis

6.1 Introduction

The assessment of text-to-speech systems is a necessary task, firstly because there are 'no physical measurements which result in indices that quantify perceptual attributes of synthesised speech' (Pavlovic, 1990). It is also the case that anyone who investigates speech synthesis for any great period finds it increasingly difficult to judge its naturalness effectively (Dixon & Maxey, 1968). Researchers need to be able to examine which aspects of a speech synthesis technique improve the overall speech (Morton, 1991), and yet the task of assessing synthetic speech is complicated by many interacting factors. At present, there is no universally agreed objective procedure that should be used, and so researchers largely rely upon subjective assessment by listeners to assess the speech.

Some aspects of speech can be quantitatively measured, such as intelligibility and comprehension. However, some aspects of the speech's quality can only be measured qualitatively, such as ease of listening, ease of comprehension and naturalness. *Naturalness,* whilst being a desirable quality, remains ill-defined. As mentioned in Chapter 5, Dutoit (1997) has defined intelligibility as being linked to 'the efficiency achieved in producing phoneme-to-phoneme transitions', whilst naturalness is 'the result of the congruence of segmental and supra-segmental features'.

In this investigation, it was impossible to make such a distinction and measure intelligibility and naturalness separately, following the above definition, because in TDCSS, intelligibility and naturalness are affected simultaneously by the segmental duration parameter. Therefore, in this chapter, naturalness is used in the sense of 'human-sounding', following the definition of Childers and Lee (1991). Since naturalness is influenced by factors such as smoothness, intelligibility and durational

pattern, it is used as an all-inclusive term, and is rather like the term *acceptability*, which was defined by Pavlovic, Rossi and Espesser (1990) as 'the user's overall satisfaction with the communication situation'.

Synthetic speech is often assessed on the word level. However, it also necessary to evaluate synthetic speech with reference to the prosodic (or *suprasegmental*) features of speech (Sonntag & Portele, 1997). The evaluation of prosodic features remains a difficult task, and it is unclear as to how synthetic speech should be assessed on the sentence level (Morton, 1991). This is because the various components of a text-to-speech synthesiser can interact with the prosodic component. A potential solution to this problem is to reduce the experimental stimuli to their prosodic content, such as fundamental frequency contour, intensity or rhythm (Sonntag & Portele, 1997).

Some of the most popular tests for evaluating text-to-speech synthesis are listed below:

- Segmental intelligibility tests (MRT, DRT, CLID)
- Semantically unpredictable sentences (SUS) test
- Paired Comparisons

6.1.1 Segmental Intelligibility Tests

Segmental intelligibility tests examine intelligibility at the phoneme level (Carlson *et al.*, 1992). These include the Modified Rhyme Test, Diagnostic Rhyme Test (Voiers, 1983) and Cluster Identification Test (Jekosch & Pols, 1994).

The *Modified Rhyme Test* involves the presentation of isolated, monosyllabic words to subjects (Logan, Greene & Pisoni, 1989). This has the advantage that it can be used with untrained listeners. In the *Diagnostic Rhyme Test*, listeners are presented with an isolated word taken from a set vocabulary, and must identify it in a pair of words (which differ only by their initial consonant). *Cluster Identification Tests* involve the presentation of nonsense words within short carrier phrases to subjects, who then write down what they think the word was (Kraft & Portele, 1995). The nonsense

words usually have a given syllable structure, such as C_iVC_f (where *i* and *f* indicate the number of initial and final consonants in the cluster). Each stimulus is only presented once, and subjects are provided with the overall list of words to be played. Responses are then scored according to correctness. Tests based on cluster identification are more inclusive and more reproducible than Diagnostic Rhyme Tests, requiring only a limited number of trained listeners (usually four is enough).

It should be noted that the use of nonsense words in segmental intelligibility tests may not be appropriate because listeners may perceive some words as being distorted versions of real words, whilst other words (which may be more perceptually distant from existing words) may be treated as genuine nonsense words (Morton, 1991). It is also the case that segmental intelligibility tests such as the MRT do not address issues such as listener preference, naturalness, and comprehension of fluent passages of synthetic speech (Logan, Greene & Pisoni, 1989).

6.1.2 Semantically Unpredictable Sentences test

Intelligibility is measured on the sentence level using the Semantically Unpredictable Sentences (SUS) test. Semantically unpredictable sentences aim to minimise the contextual semantic information given in sentences by using raw sentence structures filled by a random selection of words taken from predefined lists of possible candidates (Benoît, 1992). An example sentence is *'The table walked through the blue truth'* (Dutoit, 1997).

6.1.3 Paired Comparison tests

Pair comparison tests can operate on either the word or sentence level, and can be used to determine the ranking of systems (Johnston, 1996). These tests can either focus on specific attributes (e.g., loudness, distortion, etc.) or allow an open response (e.g., 'Which do you prefer?'). Such tests can be used to evaluate either intelligibility or naturalness.

6.2 Chapter Overview

This chapter describes an evaluation of the time-domain concatenative speech synthesis (TDCSS) method, using listening experiments to examine intelligibility and naturalness respectively. The first experiment was intended merely to establish whether it is possible to synthesise intelligible speech using TDCSS. This experiment was prompted by a claim by Linggard (1985) that 'simply concatenating words, or syllable-type units, will not result in natural-sounding or even intelligible speech'. Whilst the main emphasis of this quote is no doubt on the synthesis of continuous speech at the sentence level, it was considered important to test whether the concatenation of whole or partial syllables can produce intelligible speech at the word level.

In a second listening experiment, the naturalness of synthetic speech was evaluated using a single pair comparison format. Speech produced using the TDCSS method was compared with that of five well-known text-to-speech synthesisers. Subjects listened to pairs of synthetic speech stimuli and decided which stimulus sounded 'the most natural'. Using these responses, the synthetic stimuli were ranked in order of perceived naturalness.

In a final listening experiment, synthetic speech produced following the two-parameter concatenation method (TDCSS2) was ranked against five well-known text-to-speech synthesisers, in a similar manner to the previous experiment. In this way, a comparison could be made between the TDCSS and TDCSS2 methods in terms of perceived naturalness.

6.3 Evaluating Intelligibility

A small-scale listening experiment was carried out in order to evaluate the intelligibility of speech, synthesised by TDCSS. The motivation was to establish the *possibility* of synthesising intelligible speech using this technique. For this reason, only a small number of subjects was necessary.

6.3.1 Stimuli

The stimuli used were ten words, synthesised using the TDCSS method. These are showed in Table 6.1.

Speak	Single	Without	Three
Stop	Must	Ask	-
Never	Waveform	Most	-

Table 6.1: Words used in the intelligibility test

6.3.2 Subjects

Two subjects participated. One was a male, non-native speaker of English, in his mid-twenties; the other was a female, native speaker of English in her mid-twenties. Neither had extensive experience of synthetic speech.

6.3.3 Method

Each subject listened to ten synthetic words, following a random order, and was asked to write down what they heard for each stimulus. Each word was repeated 3 times, with a second's pause between repetitions.

6.3.4 Results

In this experiment, the male, non-native English speaker identified 70% of the stimuli correctly, whilst the female, native speaker of English correctly identified all the words. This demonstrates that it is indeed possible to produce intelligible synthetic speech using TDCSS.

6.4 Evaluating Naturalness: Investigating the One-Parameter Concatenation Framework (TDCSS)

In this study, the TDCSS synthesis method was evaluated in isolation, independently of the other components of the TTS system. This study assumed that appropriate segmental durations will eventually be made available through a statistical segmental timing algorithm, so in the experiment, hand-refined segmental durations were used. The investigation assessed the potential of the TDCSS method by comparing its best possible output (i.e., hand-refined) with that of some established text-to-speech synthesisers. This was accomplished using a listening experiment, following a pair comparison format. All comparisons were made in terms of perceived naturalness (see Section 6.1).

6.4.1 Stimuli

The stimuli consisted of isolated, single words of between one and five syllables (see Table 6.2). Six different speech synthesisers were used to produce the experimental stimuli: TDCSS, KTH, Laureate, Festival, Bell Labs and TruVoice. These are shown in Table 6.3. Here, KTH refers to the interactive multilingual text-to-speech service of KTH in Stockholm (KTH, 1998). Laureate refers to BT's text-to-speech synthesiser (Edgington *et al.,* 1996b; Laureate, 1998). Festival refers to Edinburgh University's Festival text-to-speech synthesiser, version 1.2.1 (Festival, 1997). Bell Labs and TruVoice refer to text-to-speech synthesisers produced by AT&T and Centigram respectively (Bell Labs, 1998; TruVoice, 1998).

Still	Computer	Hurts	Concatenation
Definitely	State	Talking	-

Table 6.2: Words used as synthetic stimuli

The non-TDCSS stimuli were obtained through on-line interactive speech synthesis demonstration pages on the internet. For each synthesiser, a normal speaking rate was

selected. In some cases, a 'word prosody' option was available, which would remove all phrasing from the text. This would have the effect of synthesising each word as if it were a separate sentence. However, the option was not used, because it was not available for all the speech synthesisers.

Speech Synthesiser	**Voice**
Bell Labs	American English, Male
Festival	British Male ('rab')
KTH	American English, Male
Laureate	British, Male
TruVoice	American English, Male
TDCSS	British, Male

Table 6.3: Speech Synthesiser Voices Used

For all the words synthesised by TDCSS, the phonetic transcription was produced by hand. The experiment was intended to test the naturalness of TDCSS, for CV and VC segments only, since this would be the most arduous test of the method. Each word was represented using CV and VC segments. It should be noted that the synthetic speech can be further improved by making use of CVC segments; this can reduce discontinuities through avoiding vowel-vowel concatenation points (see Section 1.3.3.2). Segmental timing was produced using the output of the duration look-up algorithm and hand-adjusting durations where necessary. In some cases, the segmental durations were further adjusted by hand, and so in this experiment, the TDCSS stimuli represent the 'ideal' output of the synthesiser, rather than current results using the Timing Module.

It should be noted that half of the speech synthesisers are for American English and half are for British English. The subjects generally stated that they prefer British English to American and so the results may be somewhat influenced by their like or dislike of the American accent. Male voices were used, so as to be comparable with the TDCSS stimuli.

6.4.2 Subjects

Fifteen subjects were used in the listening experiment (see Table 6.4). Nine of them were male and six were female. Nine subjects were native speakers of English whilst six were not. All of them had normal hearing, and did not have extensive experience of speech synthesis. Subjects were told that the purpose of the experiment was to compare a number of different speech synthesisers in terms of naturalness. All of the subjects were told to only judge the naturalness of the stimuli, and were instructed to ignore the meaning of the words presented.

Subject Attributes	Number of subjects
Male	9
Female	6
Native	9
Non-native	6

Table 6.4: Description of subjects

6.4.3 Method

In a ranking experiment, six speech synthesisers were compared in terms of perceived naturalness. For each synthesiser, a set of seven isolated words was synthesised. The main part of the experiment was carried out via an interactive computer program (see Figure 6.1). A pair comparison format was followed, so that each pair of stimuli presented to the subject would consist of the same word, produced by two different speech synthesisers. The pairs were arranged in such a way that every possible combination of the six speech synthesisers was used for a particular word. The pairs of stimuli were presented in a completely random order, so as to minimise any possible boredom that may arise. After each pair was presented, the subject was asked to indicate which of the stimuli they thought was 'the most natural sounding' by clicking a 'First' or 'Second' button. A 'Repeat' button allowed subjects to hear a pair of stimuli as many times as was necessary before making a choice. All of the subjects were provided with a list of the words that they would encounter, in order to reduce

any cognitive load associated with word identification, and to enable them to concentrate purely on judging naturalness.

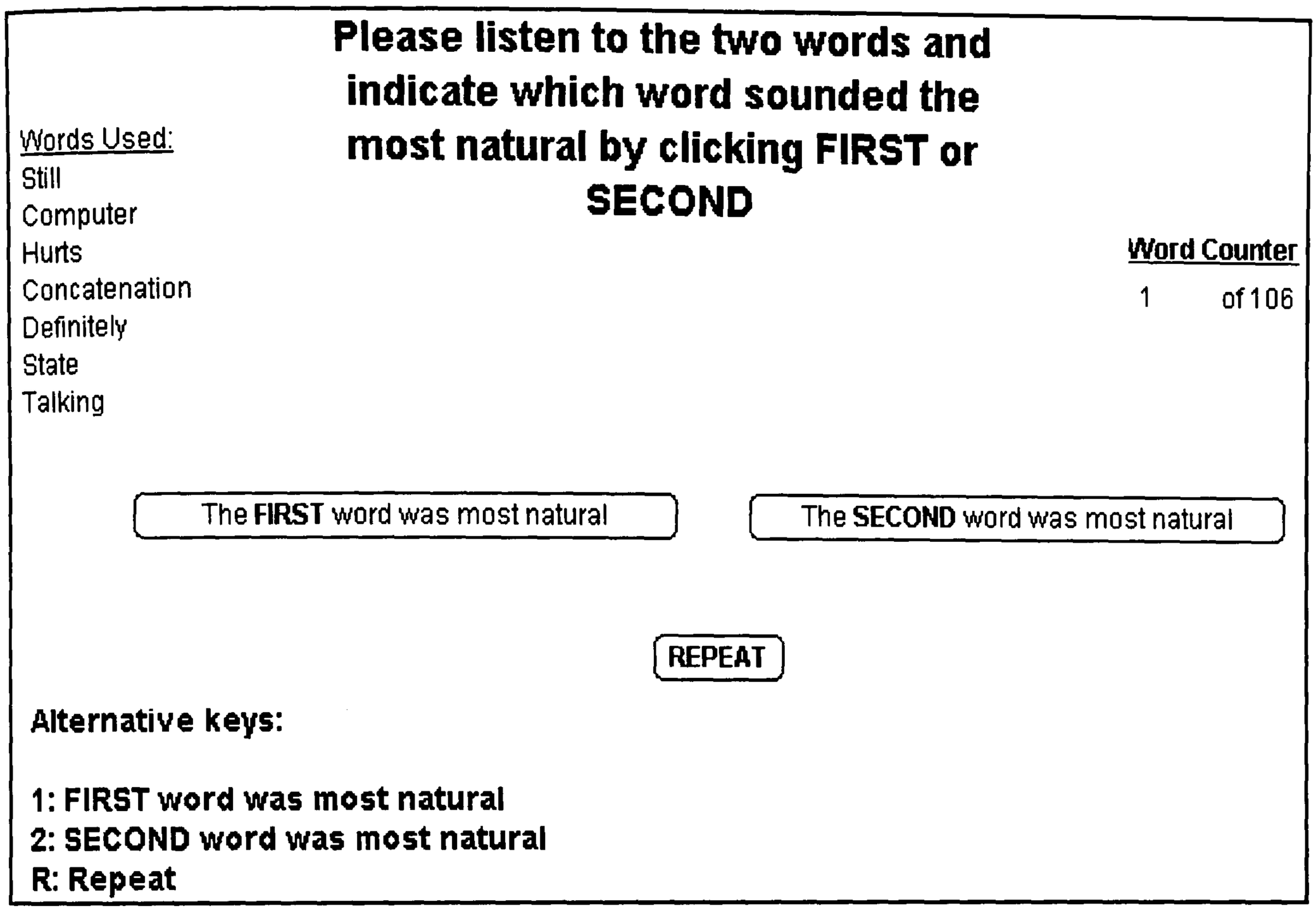


Figure 6.1: Screen layout for listening experiment

Each speech synthesiser was assigned a separate running total, initialised to zero at the start of the experiment. Every time the 'First' or 'Second' button was pressed, a total was incremented for the relevant speech synthesiser. The scores for each speech synthesiser were then used to rank them in order of perceived naturalness.

6.4.4 Results

The overall rankings for the speech synthesisers are shown in Figure 6.2. The histogram shows the average number of times that a particular speech synthesiser was preferred to another.

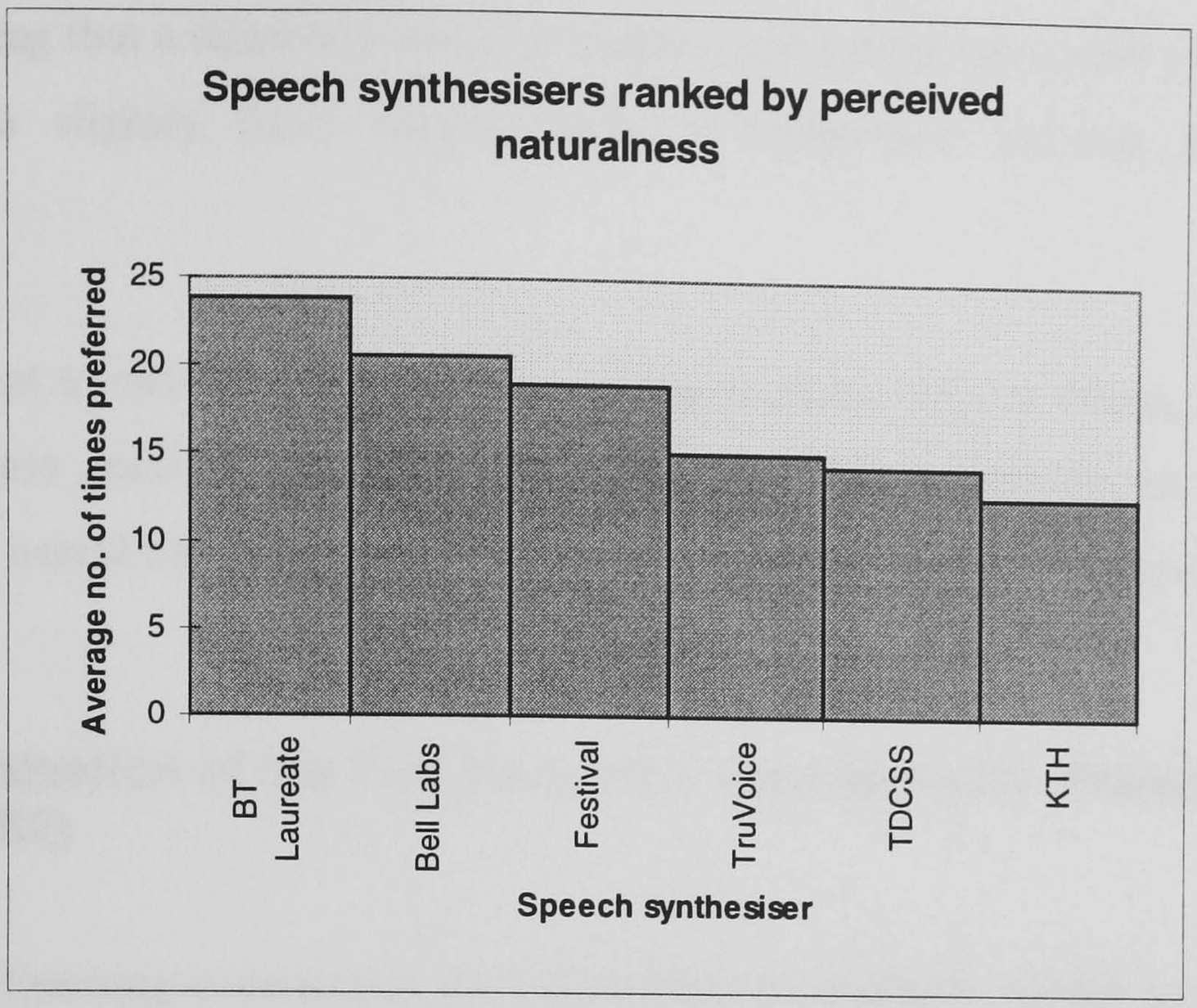


Figure 6.2: Overall rankings for six text-to-speech synthesisers

6.4.5 Discussion

When ranked against five established speech synthesisers, TDCSS was judged to be the fifth most natural out of six. This is a fairly impressive result, considering that TDCSS was compared with synthesisers that are either well-established commercially or academically. Although such experiments give an overall ranking, such data are ordinal, and so give no indication as to the actual distance between the individual ranks. It was evident that several subjects favoured a British speech accent to an American one. This is perhaps unsurprising, since all the subjects were British residents. It was often the case that shorter words, such as monosyllables and disyllables, tended to be judged more favourably. It was also found that certain subjects consistently preferred the TDCSS stimuli to the other synthesisers. It is probable that even better results would be obtained if CVC syllables were used in producing the TDCSS stimuli.

The speech synthesisers used were based on a variety of different synthesis methods, including TD-PSOLA (Laureate), LPC (Bell Labs) and formant synthesis (KTH). The overall results follow the general trend that concatenative synthesisers (particularly the TD-PSOLA method) are perceived as being more natural (Klaus *et al.*, 1997). It is

also interesting that a relatively unsophisticated synthesis method such as TDCSS was perceived as slightly more natural than an established formant text-to-speech synthesiser.

It is important to consider that in this experiment, only isolated words, of up to five syllables, were used as stimuli. It is likely that further differences between the synthesisers would emerge, in terms of predicted intonation at the sentence level.

6.5 An Evaluation of the Two-Parameter Concatenation Framework (TDCSS2)

In a second listening experiment, the naturalness of synthetic speech produced using the TDCSS2 method was compared with that of five established text-to-speech synthesisers. The experiment was carried out in essentially the same manner as the experiment described in Section 6.4, differing only in the TDCSS stimuli that were used. This had a two-fold purpose: it enabled a comparison between the one-parameter and two-parameter concatenation frameworks described in Chapter Three, whilst also providing a ranking against the non-TDCSS stimuli mentioned in Section 6.4.

6.5.1 Stimuli

The TDCSS2 stimuli were produced using only CV and VC syllables, and were the same words mentioned in Table 6.1. Since a timing database has not yet been constructed for TDCSS2, the segmental durations were entirely produced by the expert judge (DSGV), following the two-parameter concatenation framework outlined in Section 3.6. The TDCSS2 stimuli were compared with stimuli from the previous experiment, produced by the text-to-speech synthesisers KTH, Laureate, Festival, Bell Labs and TruVoice (see Section 6.4).

6.5.2 Subjects

Fifteen subjects were used in the listening experiment (see Table 6.4). None of them had been used in any of the previous listening experiments. Nine of them were male and six were female. Nine subjects were native speakers of English whilst six were not. All of them had normal hearing, and did not have extensive experience of synthetic speech. Subjects were told that the purpose of the experiment was to compare a number of different speech synthesisers in terms of perceived naturalness. All of the subjects were told to only judge the naturalness of the stimuli, and were instructed to ignore the meaning of the words presented.

Subject Attributes	Number of subjects
Male	9
Female	6
Native speaker of English	9
Non-native speaker of English	6

Table 6.5: Description of subjects

6.5.3 Method

The six speech synthesisers were compared using similar experimental procedures to those described in Section 6.4. For each synthesiser, a set of 6 isolated words was synthesised. Every possible combination of the synthetic words was presented to the subjects. The pairs of stimuli were presented in a completely random order, so as to minimise any possible boredom that may arise. For each pair, subjects indicated which stimulus was 'the most natural sounding'. All subjects were given a list of the different possible words that they would encounter.

6.5.4 Results

When ranked against five text-to-speech synthesisers, the TDCSS2 stimuli were

judged to be the fifth most natural. This is the same rank achieved by the TDCSS stimuli in the previous experiment (see Section 6.4). It would thus appear that there is little difference in naturalness between speech synthesised by either concatenation framework. However, the average number of times that the TDCSS stimuli were preferred was 14.5, compared with a value of 12.9 for TDCSS2. From this, it is possible to infer that TDCSS2 stimuli were marginally less natural than the TDCSS stimuli. This result could possibly be due to the expert judge's relative lack of experience with the TDCSS2 method.

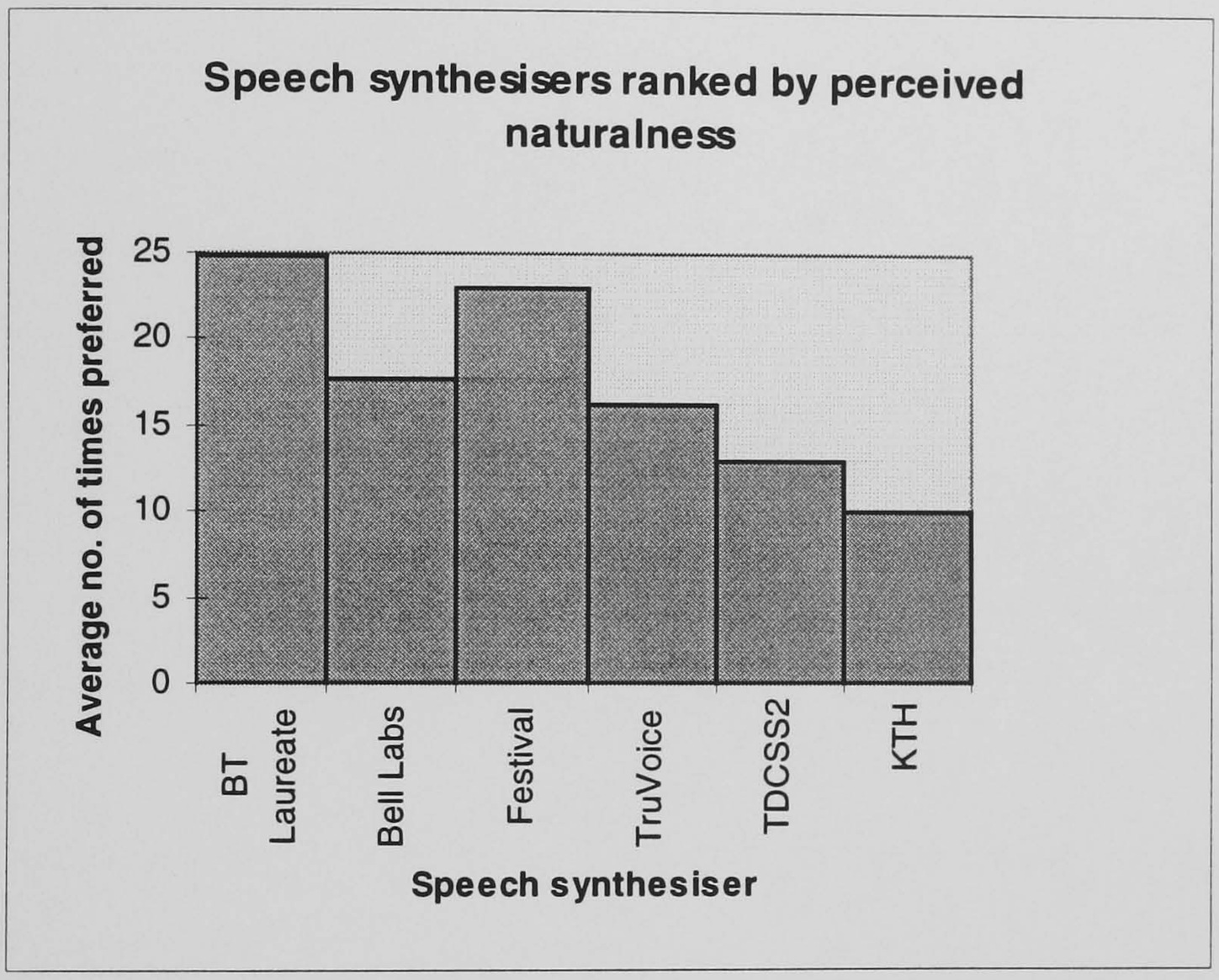


Figure 6.3: Perceived naturalness of TDCSS2 stimuli, compared with five text-to-speech synthesisers

On the whole, the results largely confirm the overall ranking of speech synthesisers found in the previous experiment, with the exception that the rankings are swapped for Bell Labs and Festival.

6.6 Summary

It has been demonstrated that intelligible speech can be synthesised using the TDCSS method. Listening experiments have also been carried out to evaluate the naturalness of speech produced by both the TDCSS (one-parameter concatenation framework) and the TDCSS2 (two-parameter concatenation framework) methods. In single pair

comparison experiments, TDCSS and TDCSS2 speech stimuli were compared with the output of five established text-to-speech synthesisers. Both TDCSS and TDCSS2 were judged to be fifth most natural out of the six synthesisers. Although the TDCSS2 method has some theoretical advantages over TDCSS (in terms of its flexibility), both sets of stimuli were judged to be equally natural in the listening experiments.

Chapter Seven

7. Conclusions

In this thesis, a time-domain waveform concatenation framework for speech synthesis has been implemented and evaluated. At present, the TDCSS framework allows the synthesis of isolated words from a text input. In the TDCSS framework, CV, VC and CVC syllables are used as concatenative units, along with CC diphones. The concatenation method has been designed in such a way as to reduce the amount of signal processing applied to each speech segment, as it retains simplicity and improves the intelligibility and naturalness of each speech segment (Saito, 1992). This investigation was partly prompted by a comment from Linggard (1985) that '...simply concatenating words, or syllable-type units, will not result in natural-sounding or even intelligible speech'. It also stemmed from the need for more attention to be paid to the concatenation process itself (Kraft, 1994). Part of this study has thus evaluated the potential of a simple concatenation method for time-domain waveforms, and shown that intelligible speech can be synthesised, for isolated words.

The concatenative framework reduces the problem of waveform discontinuities at the boundaries between adjacent speech segments. At the same time, it allows prosodic duration changes to be made, in order to produce a natural speech rhythm. Both of the above issues may be dealt with using a single segmental duration parameter that specifies the initial portion of the segment to be output. Using the segmental duration parameter, the point of concatenation itself can be varied from segment to segment. Smoother speech can be produced by choosing segmental durations that correspond to 'concatenation nodes', which often coincide with zero-crossings in the time domain. This technique helps to preserve the acoustic uniformity at concatenation points, and is somewhat similar to Hess's (1992) idea of using 'floating concatenation points', where the crossover between segments is set to the point of minimal spectral distance within a limited interval.

The appropriate segmental durations for text-to-speech synthesis can be predicted for isolated words, using a timing database and a pattern-matching look-up algorithm.

The timing database consists of segmented words with their corresponding duration values, and is constructed using an iterative synthesis/adjustment approach. In this process, a listener (known as an 'expert judge') repeatedly adjusts segmental durations according to the perceived naturalness of the speech output. In this way, segmental duration data is produced that is directly related to a specific waveform inventory. The *iterative synthesis/adjustment* approach is thus a perceptual approach to data acquisition, since an expert judge's perception of the final synthetic speech signal is involved.

Segmental duration prediction can be successfully carried out using a data-driven approach, provided that there are sufficiently large quantities of timing data available. Indeed, in the Timing Module that has been developed, the accuracy of segmental duration predictions increases with the size of the timing database. Whilst the process of creating the timing database is somewhat time-consuming, a default duration procedure has been described, in which existing timing information is re-categorised according to articulation manner. This potentially allows the size of the timing database to be doubled, or even trebled. In cases where the duration look-up algorithm fails to find sufficient timing data for a particular word, the default duration procedure re-categorises the missing speech segment symbols according to articulation manner. The timing database is then searched for the new set of speech segment symbols. This approach provides a graceful degradation in the performance of the duration look-up algorithm in cases where timing data is incomplete.

The iterative synthesis/adjustment method has been investigated via a set of listening experiments. These provided evidence that an expert judge's perception of naturalness was largely representative of that of the participating subjects. It is thus concluded that it is possible for a team of relatively inexperienced listeners to act as expert judges, participating in the acquisition of timing data via iterative synthesis/adjustment methods. An additional listening experiment showed that an expert's judgement of naturalness remained stable over several weeks. This finding is essential to the 'expert judge' method for timing data acquisition, since a single timing database may take several weeks to construct.

In a further investigation of the iterative synthesis/adjustment approach, inexperienced subjects acted as expert judges in synthesising a set of words following the TDCSS principle. It was found that there was no significant difference between a set of reference values pre-defined by DSGV (an experienced expert judge) and those favoured by the inexperienced subjects (with a two-tailed probability of 0.05). Furthermore, a strong, positive correlation was found between the subjects' chosen duration values and the reference durations. These experimental results complement the findings of the previous listening experiments, namely, that there is a commonality between the expert judge's perception of the naturalness of segmental durations and other people's perceptions.

When compared with five well-known text-to-speech synthesisers, speech synthesised by TDCSS was judged to be the fifth most natural out of the six synthesisers. This was verified using a single pair comparison experiment, with fifteen subjects. In creating the TDCSS stimuli, segmental durations were hand-adjusted, so that the stimuli represented the 'ideal' output of the speech synthesiser. The above result is encouraging, since the TDCSS stimuli were perceived to be more natural than one particular text-to-speech synthesiser, which is relatively well-established. Of course, one must be careful not to over-extrapolate this result, due to the scale of the listening experiment and the number of subjects used. The result also only applies to isolated-word stimuli.

When speech produced using the two-parameter concatenation framework (TDCSS2) was compared with the same five text-to-speech synthesisers, it was also judged to be fifth most natural. It therefore remains unclear as to whether TDCSS or TDCSS2 is a superior concatenation framework. However, TDCSS2 does have some theoretical advantages, in terms of its flexibility.

Whilst in its present form, the time-domain concatenative speech synthesis method only produces isolated words, it could be usefully applied to any system that uses shorter synthetic messages, such as enquiry or dialogue systems, or the synthesis of email subject lines. The reason for this is that in contexts in which there is not much linguistic redundancy, prosodic manipulation is less important (Hess, 1992). Of course, a system for predicting pauses would be necessary, even for the production of

short sentences. Also, at the sentence level, segmental durations predicted for isolated words would have to be re-calculated.

This thesis has thus described the implementation and evaluation of a text-to-speech synthesis method, which involves a time-domain concatenation of speech segments. It has also investigated some of the perceptual issues related to an auditory evaluation method that can be used for obtaining the necessary timing data for statistical segmental duration prediction.

7.1 Future Work

The two-parameter concatenation method (TDCSS2) merits further investigation, since it allows greater flexibility in the choice of concatenation point between speech segments. This allows the possibility of smoother transitions at speech segment boundaries, leading to a greater level of naturalness. Whilst a segmental duration look-up system has already been developed for TDCSS2, a full evaluation of this method requires the development of a much larger timing database, dedicated to two-parameter timing information.

The general time-domain concatenative speech synthesis framework is limited in that no explicit smoothing operations or manipulation of fundamental frequency is offered in its present form. This could be rectified by viewing the concatenation framework as a pre-processing stage in a conventional concatenative speech synthesiser. Since each segmental duration directly specifies a portion of a speech unit, this 'front end' is somewhat analogous to the speech segment selection process, in which an optimal set of concatenative units is constructed (e.g., Iwahashi & Sagisaka, 1995). This reduces the 'concatenation cost' incurred, in terms of spectral discontinuities.

By using optimal segmental durations in the first place, the extent to which the original acoustic samples are processed is greatly reduced, preserving the qualities of the original speech data as far as possible (for instance, speech segments are not stretched or squashed to fit a particular duration value). At this point, it would be profitable to introduce a smoothing mechanism (e.g., a form of overlap-add). At

present, a smoothed version of the Synthesis Module has already been implemented, in which adjacent speech segments are overlapped (with an overlap width of 25 ms) and weighted by a raised cosine window.

Possible future research might be to use the concatenation framework as a pre-processor to a time-domain pitch synchronous overlap-add (TD-PSOLA) synthesiser. This would allow the manipulation of fundamental frequency and provide additional smoothing. It is possible that the use of optimal segmental durations may help to solve some of the problems associated with TD-PSOLA, such as the production of artefacts similar to comb filtering (Kortekaas & Kohlrausch, 1997). Alternatively, the concatenation framework could be incorporated into a speech synthesis system that uses linear prediction techniques.

Additional work could also include an investigation into the issue of segment elision, in which some speech segments are omitted from the synthetic utterance. If suitable segment elision rules can be formulated, the omission of carefully chosen units can provide a marked improvement to the output speech. This can have the effect of reducing the density of concatenation points whilst increasing the speaking rate.

8. Appendix A - Naming Conventions

Below are details of the labelling convention used for the speech segments in the waveform inventory. Example carrier words are given for each speech sound. For the consonants, a description is given, in terms of articulation manner. Individual segments from these tables can be appended to make up CV, VC, CC or CVC segments.

For example, the word 'two' would be represented as /tooh/. This is formed using /t/ from Table A.2 and /ooh/ from Table A.1.

Vowels	Example
a	(cat)
ah	(car)
ay	(bay)
e	(bet)
ee	(bee)
i	(bit)
o	(dog)
oh	(toe)
oo	(took)
ooh	(through)
or	(lawn)
u	(bud)
ur	(burn)

Table A.1: Naming conventions for vowels

Consonants	Example	Description
b	(bat)	voiced stop
c	(cat)	plosive
ch	(chat)	unvoiced affricate
cw	(quick)	plosive followed by semi-vowel
dg	(edge)	affricate
d	(did)	voiced stop
dh	(this)	voiced fricative
f	(if)	unvoiced fricative
g	(go)	voiced stop
h	(head)	aspirate
l	(low)	liquid
m	(mow)	nasal
n	(no)	nasal
ng	(young)	nasal
p	(pat)	unvoiced stop
r	(ran)	liquid
s	(sat)	unvoiced fricative
sh	(shy)	unvoiced fricative
t	(tow)	unvoiced stop
th	(Kath)	unvoiced fricative
v	(vow)	voiced fricative
y	(yacht)	vowel
z	(zoo)	voiced fricative
zh	(Asia)	voiced fricative

Table A.2: Naming conventions for consonants

Diphthongs	Example
ew	(new)
eye	(bite)
ow	(now)
oy	(boy)
w	(we)

Table A.3: Naming conventions for diphthongs

9. Appendix B - Duration Look-Up Algorithm

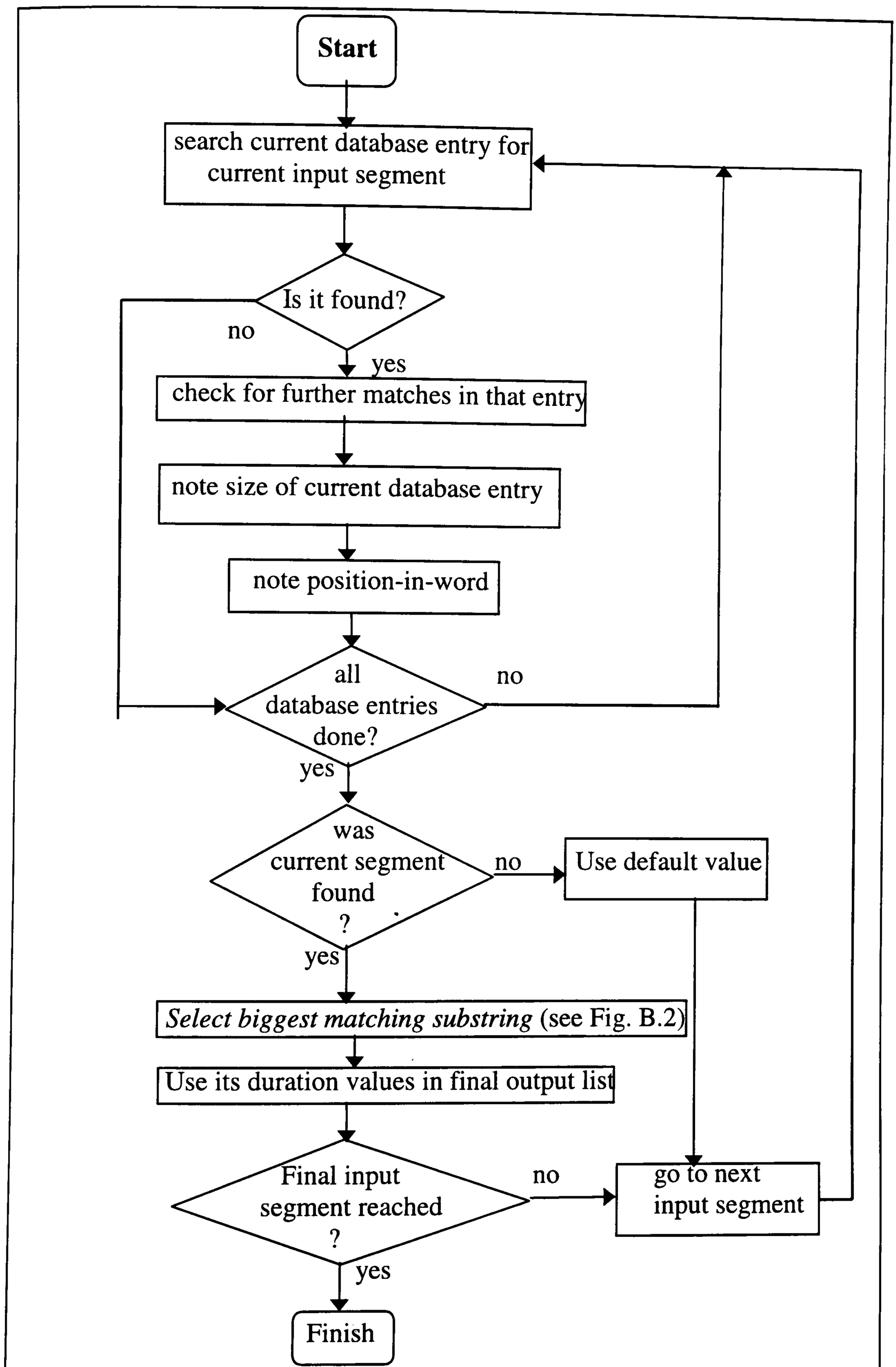


Figure B.1: Duration look-up algorithm

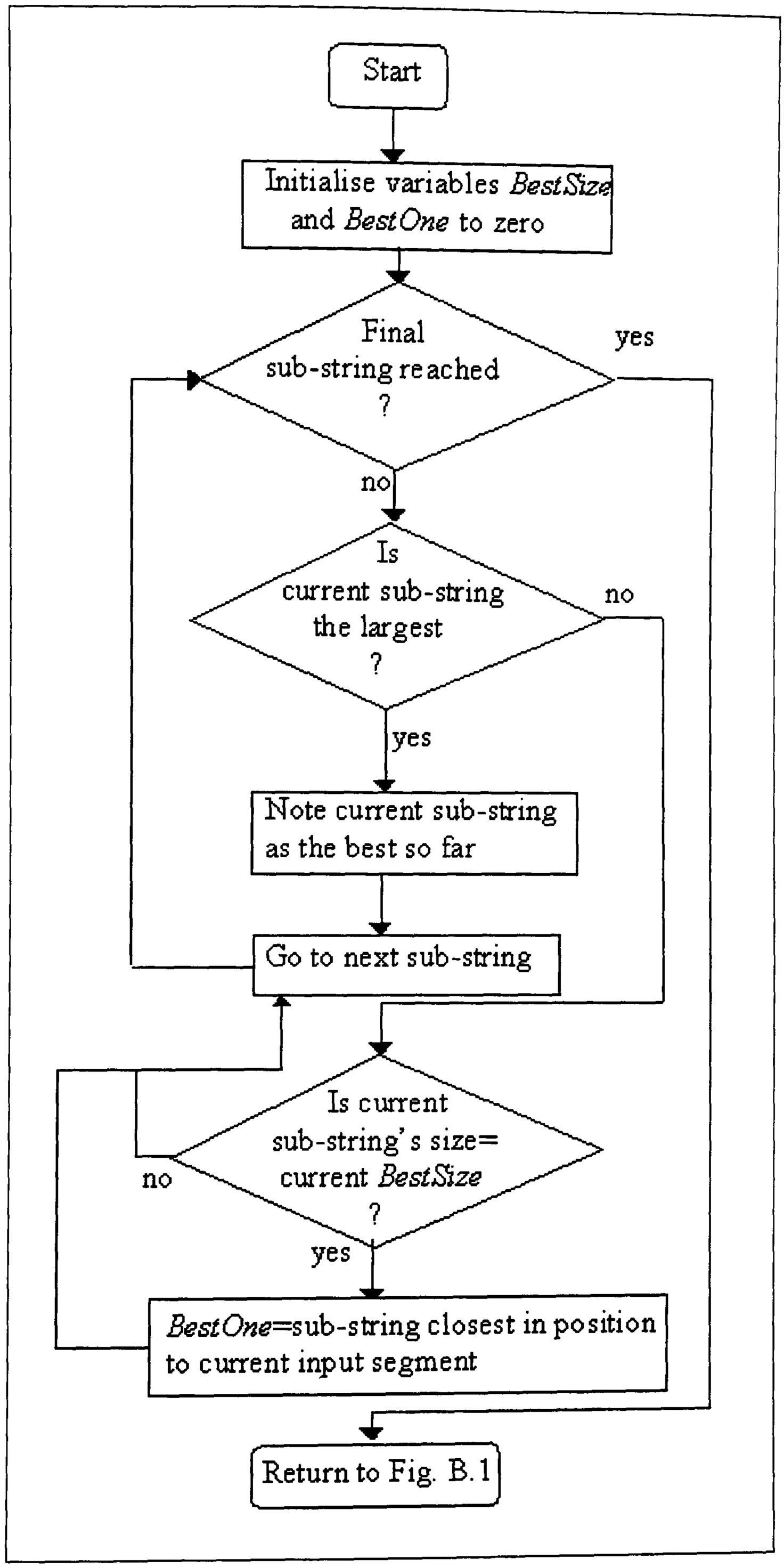


Figure B.2: *Select Biggest Matching Sub-string* function

10. Appendix C - Default Duration Procedure

Figure C.1 shows the procedure for producing default duration values when the duration look-up algorithm fails to predict a complete set of segmental durations. After the initial duration look-up, any 'missing' segments are recategorized, and a new duration look-up is carried out.

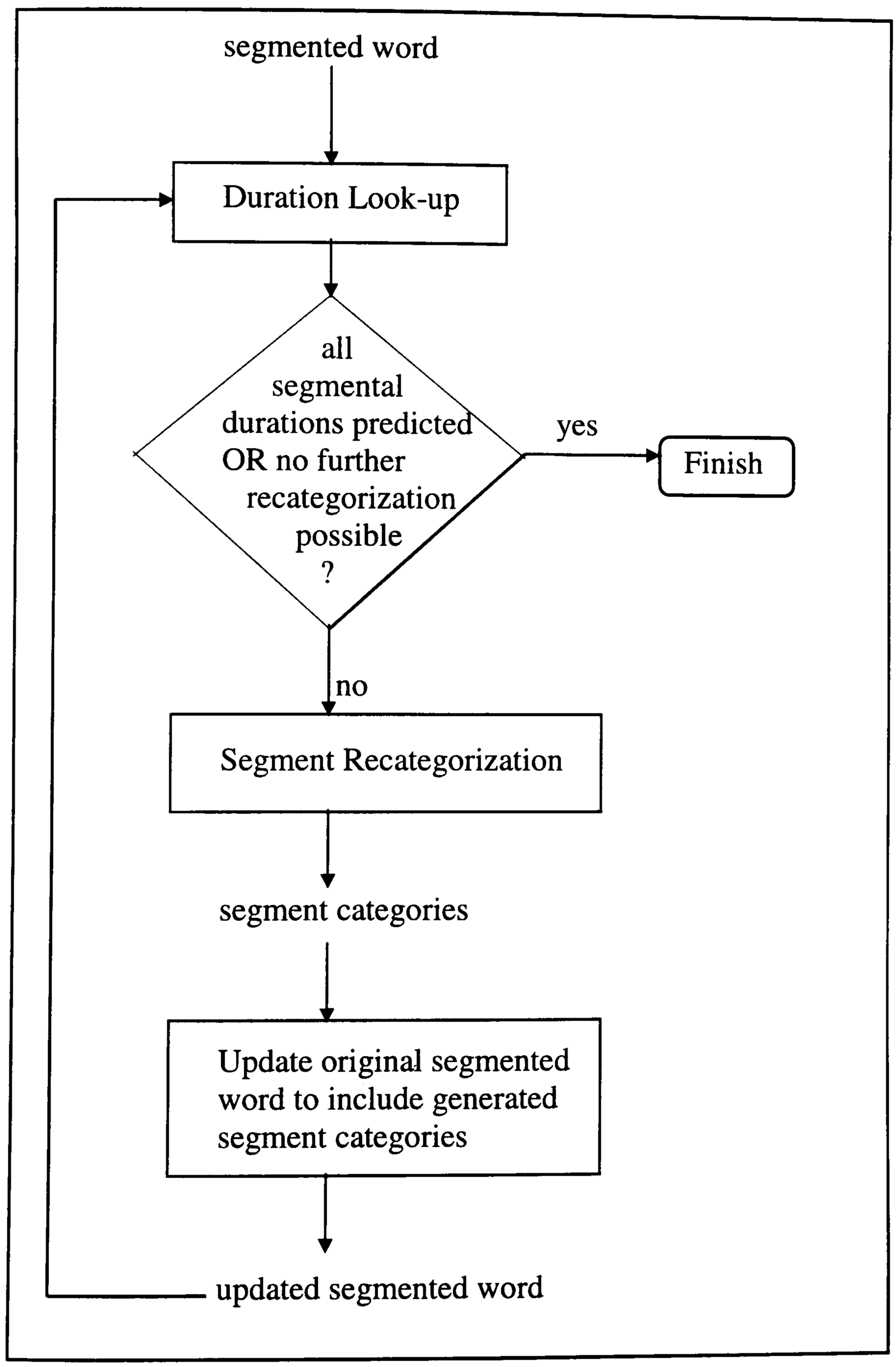


Figure C.1: Default duration scheme

11. Appendix D - Listening Experiment Results

In the following graphs, $a = \alpha = 20$ ms

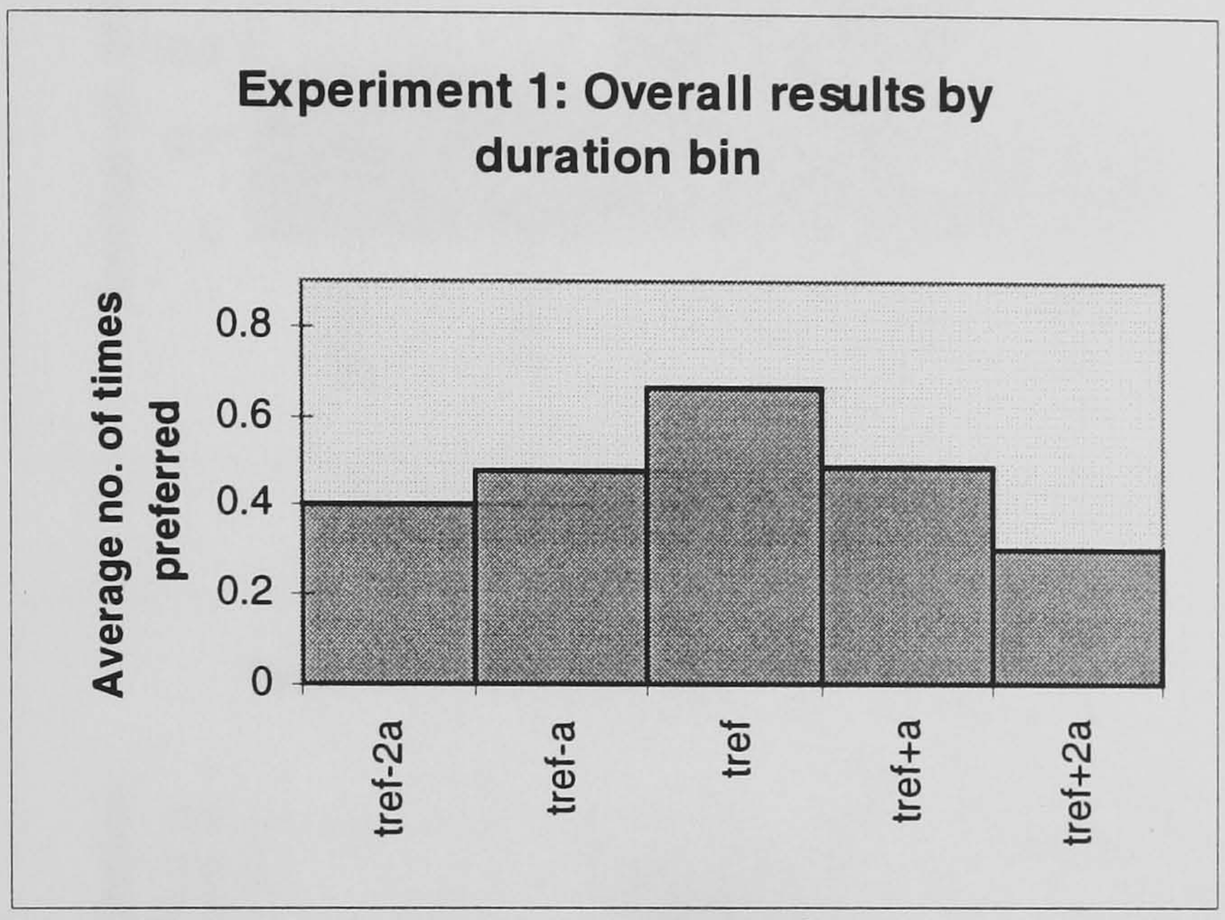


Fig. D.1.1: Experiment 1: Average preferences by duration bin

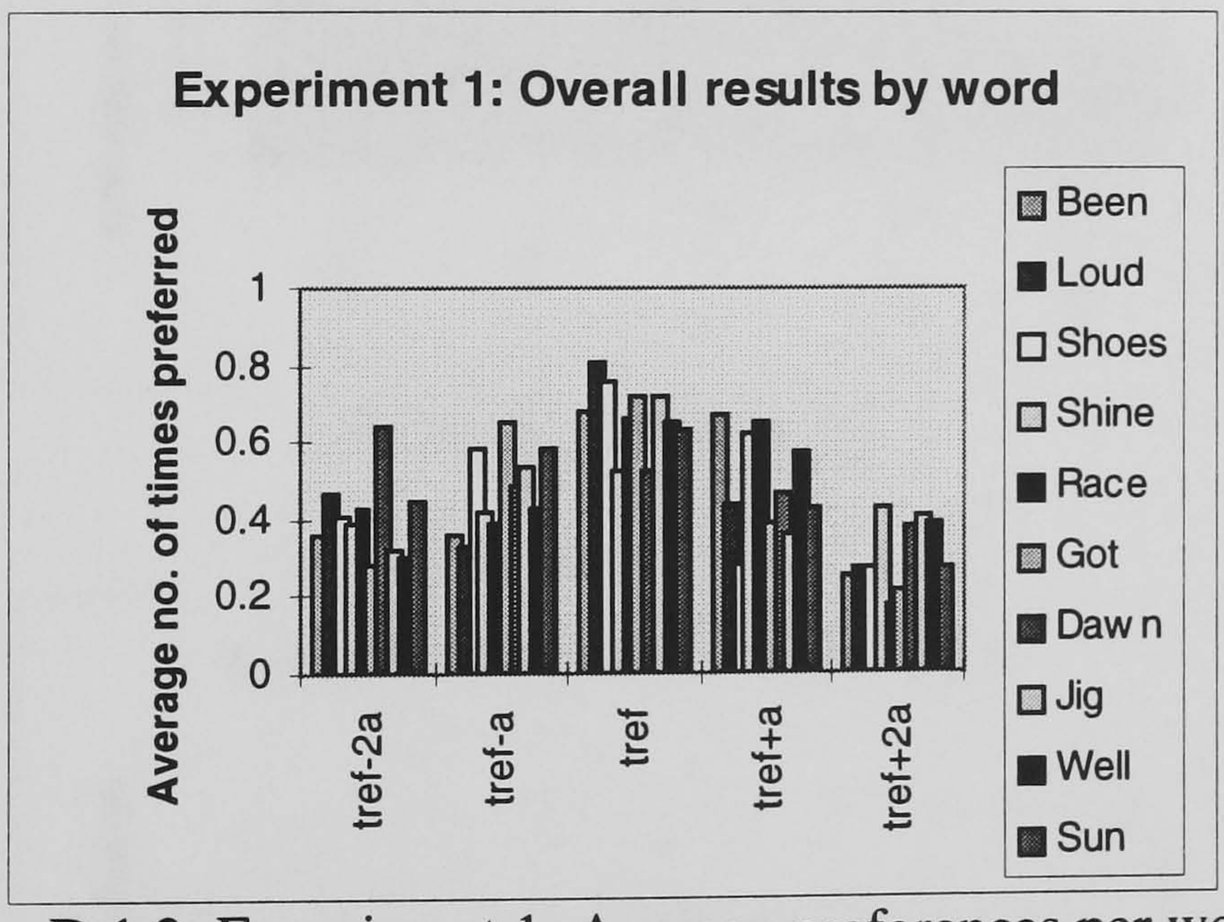
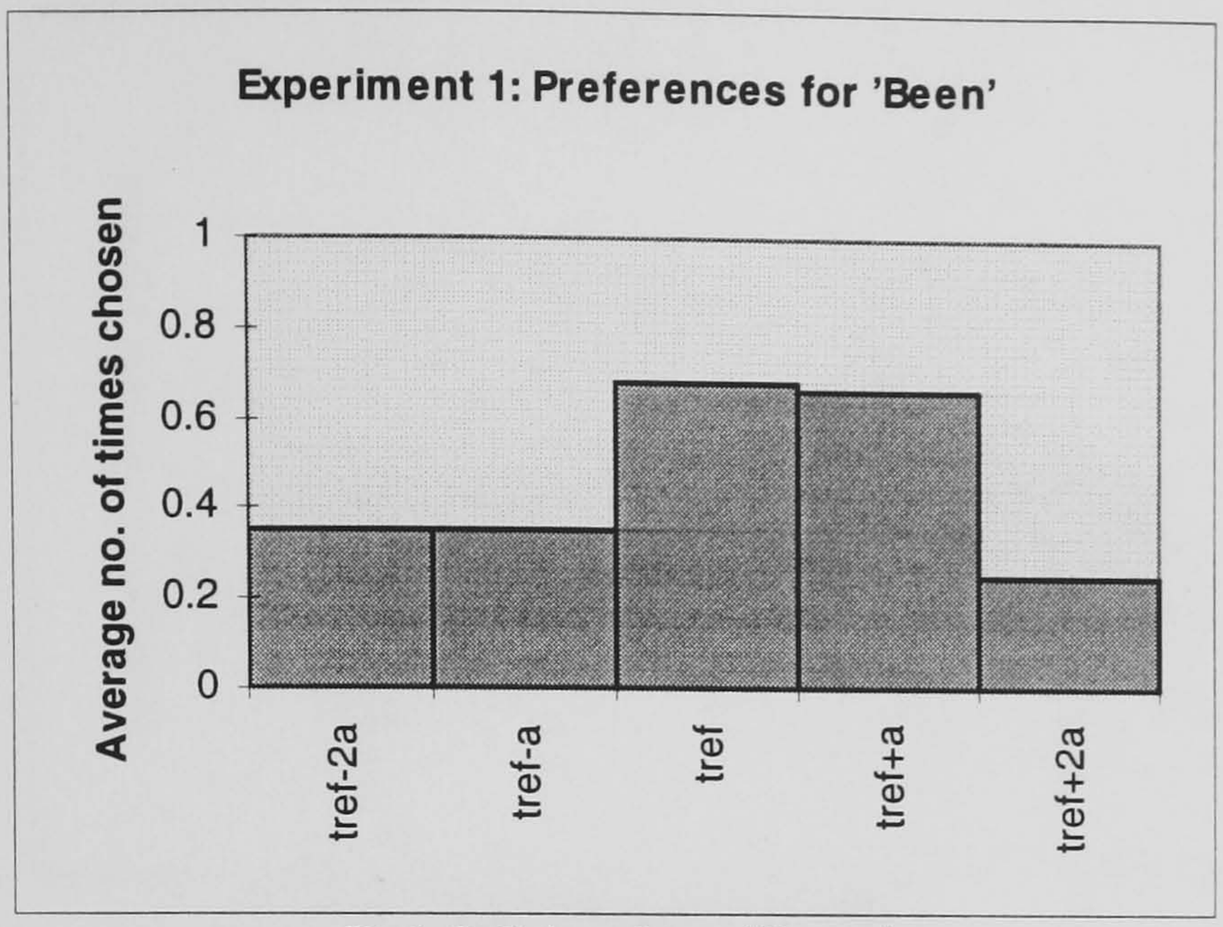
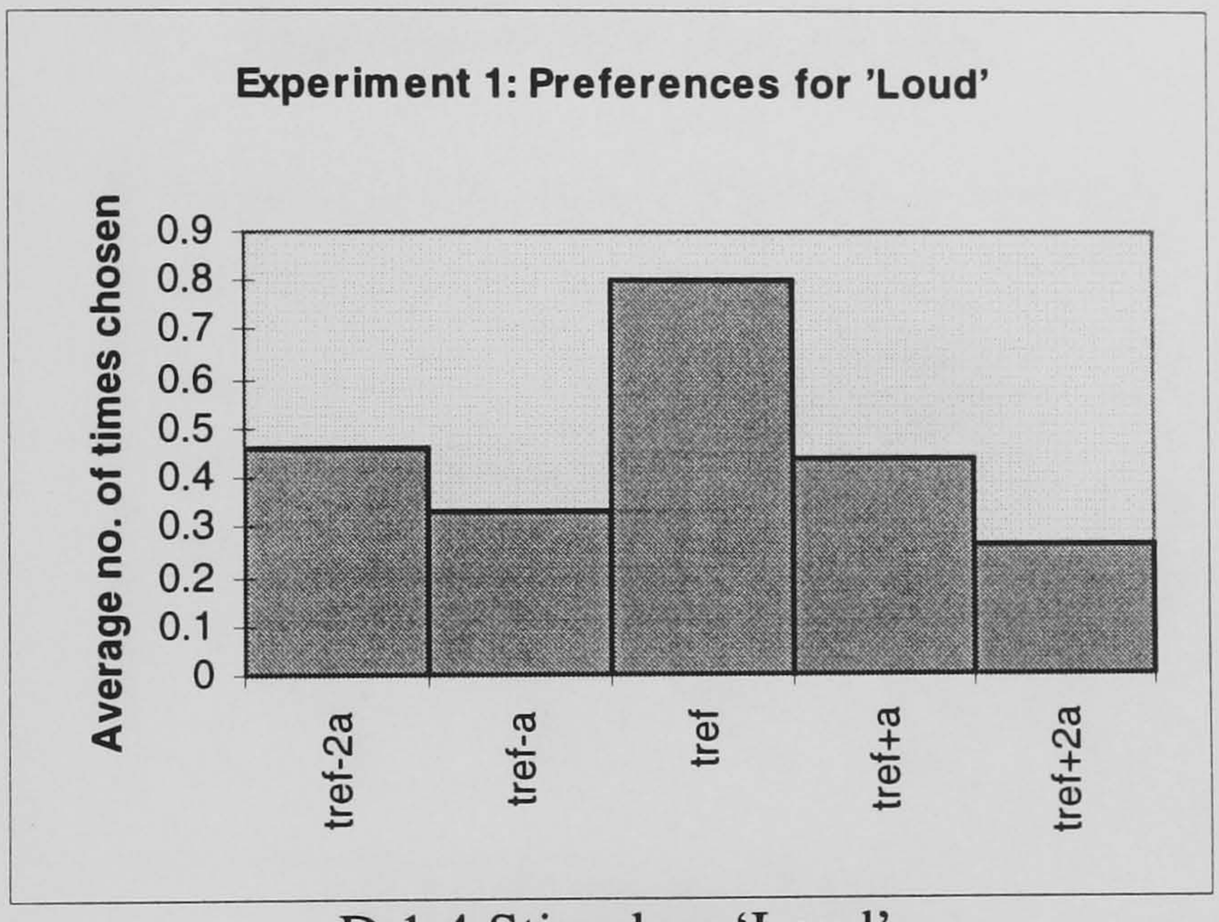


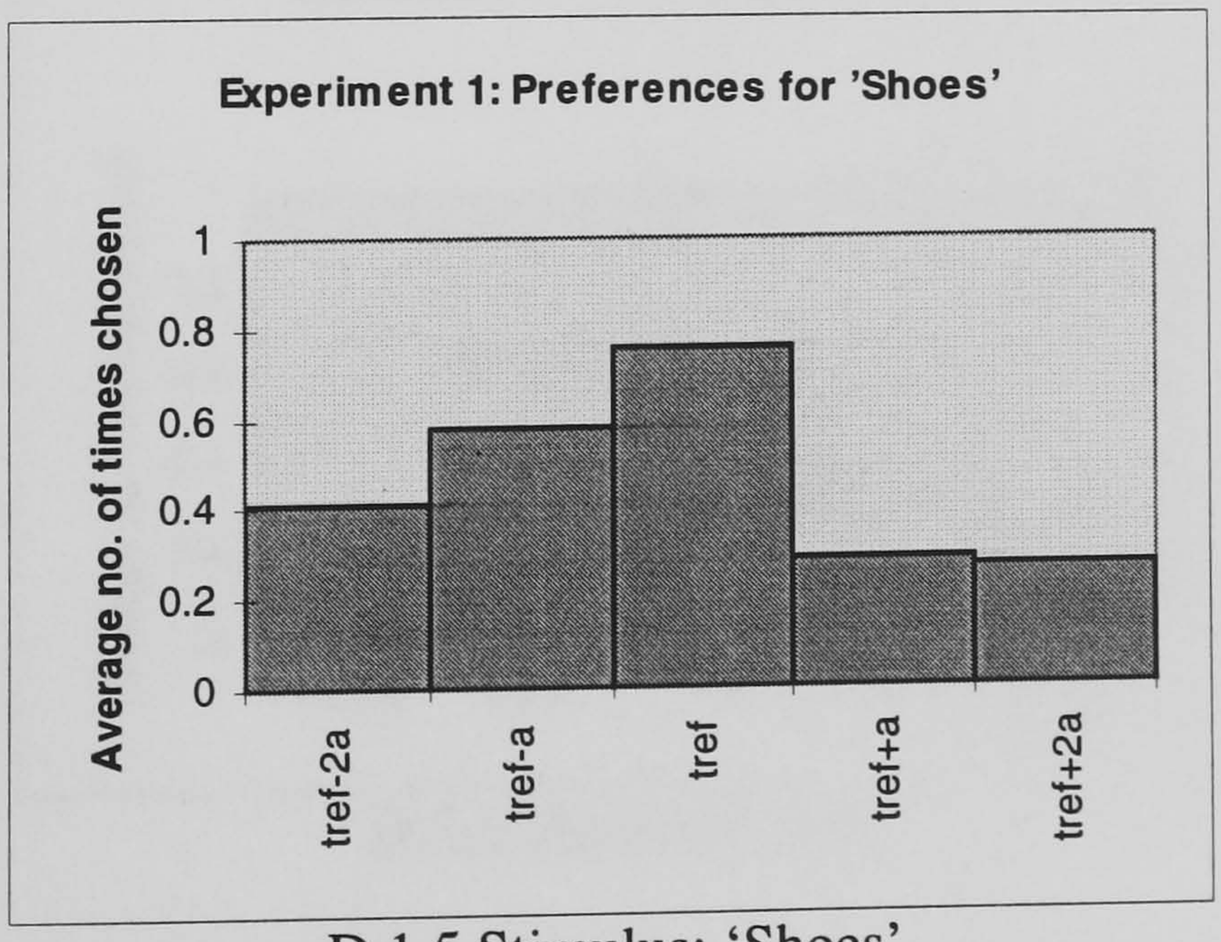
Fig. D.1.2: Experiment 1: Average preferences per word



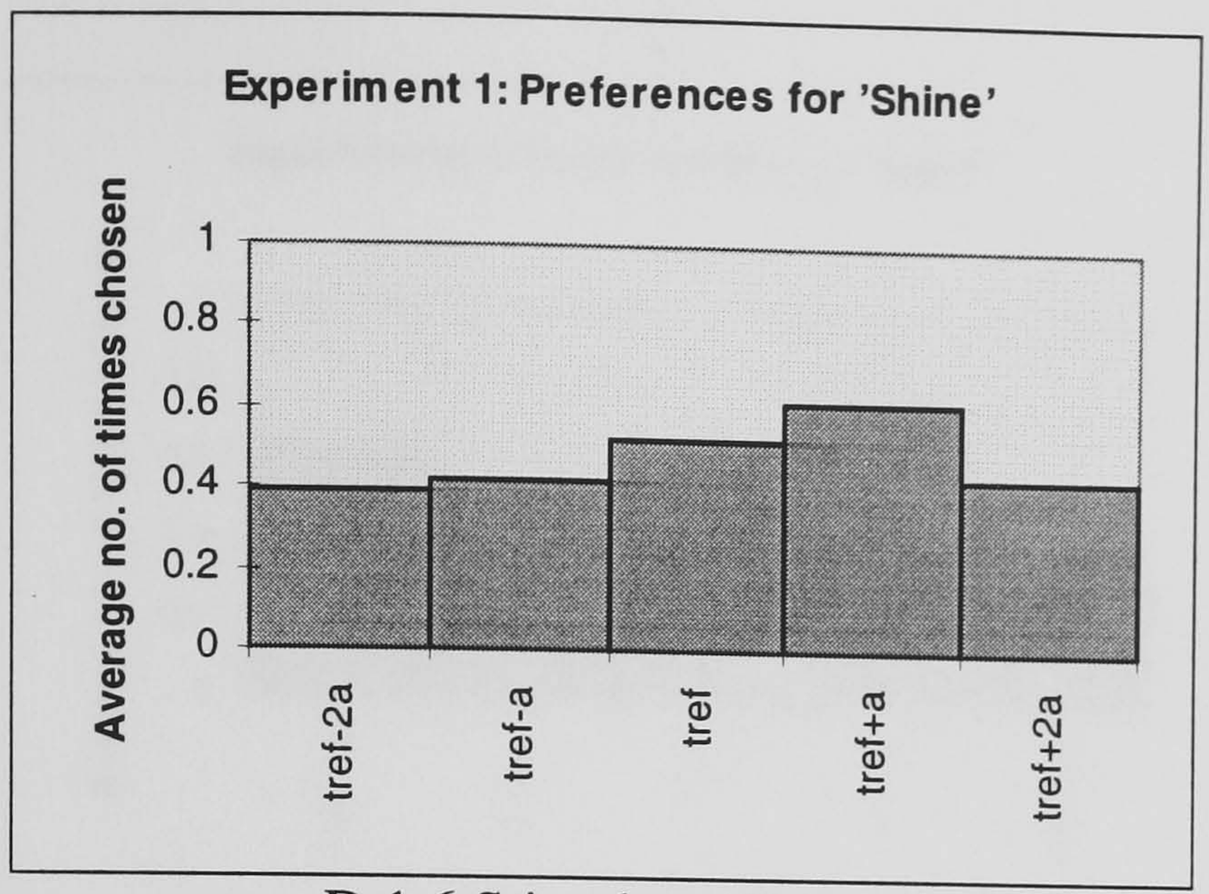
D.1.3 Stimulus: 'Been'



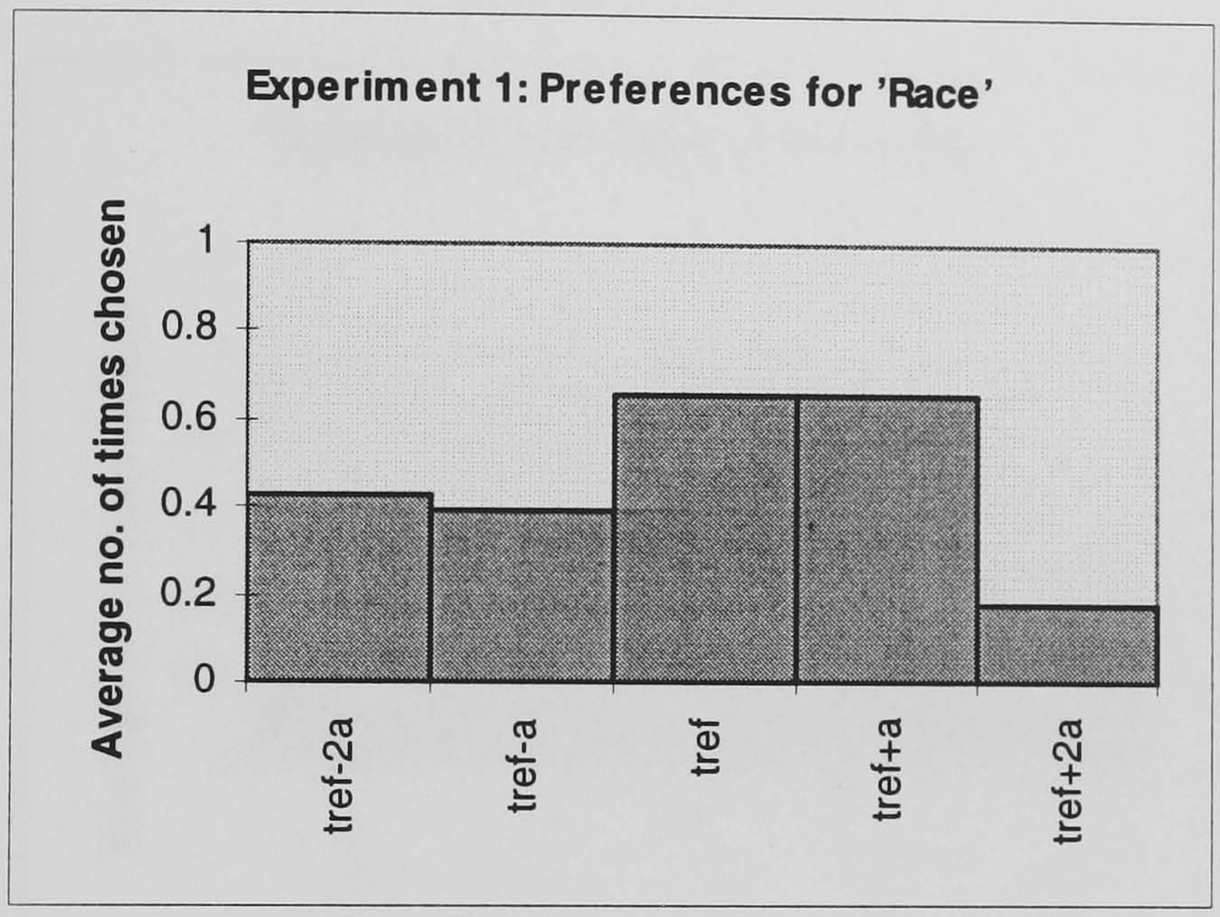
D.1.4 Stimulus: 'Loud'



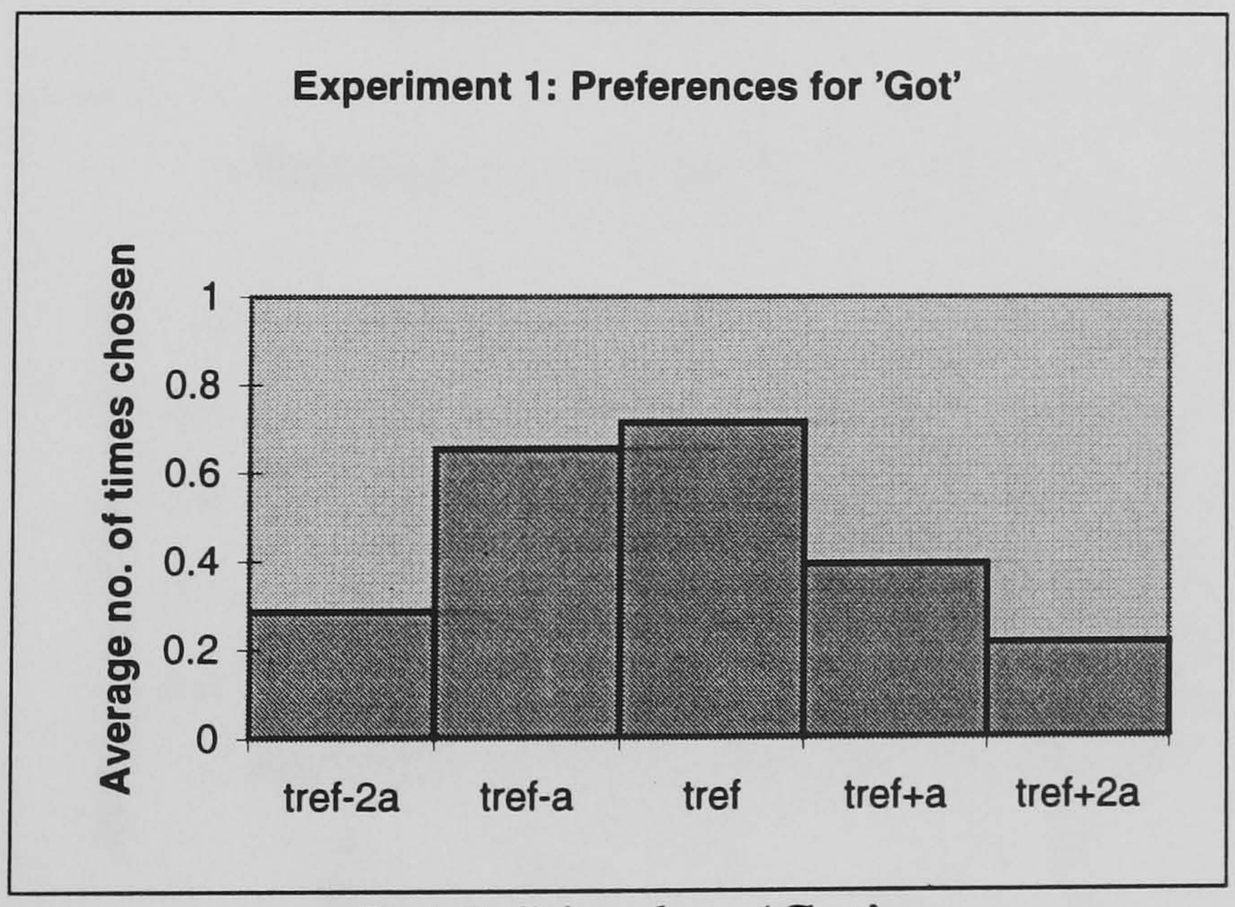
D.1.5 Stimulus: 'Shoes'



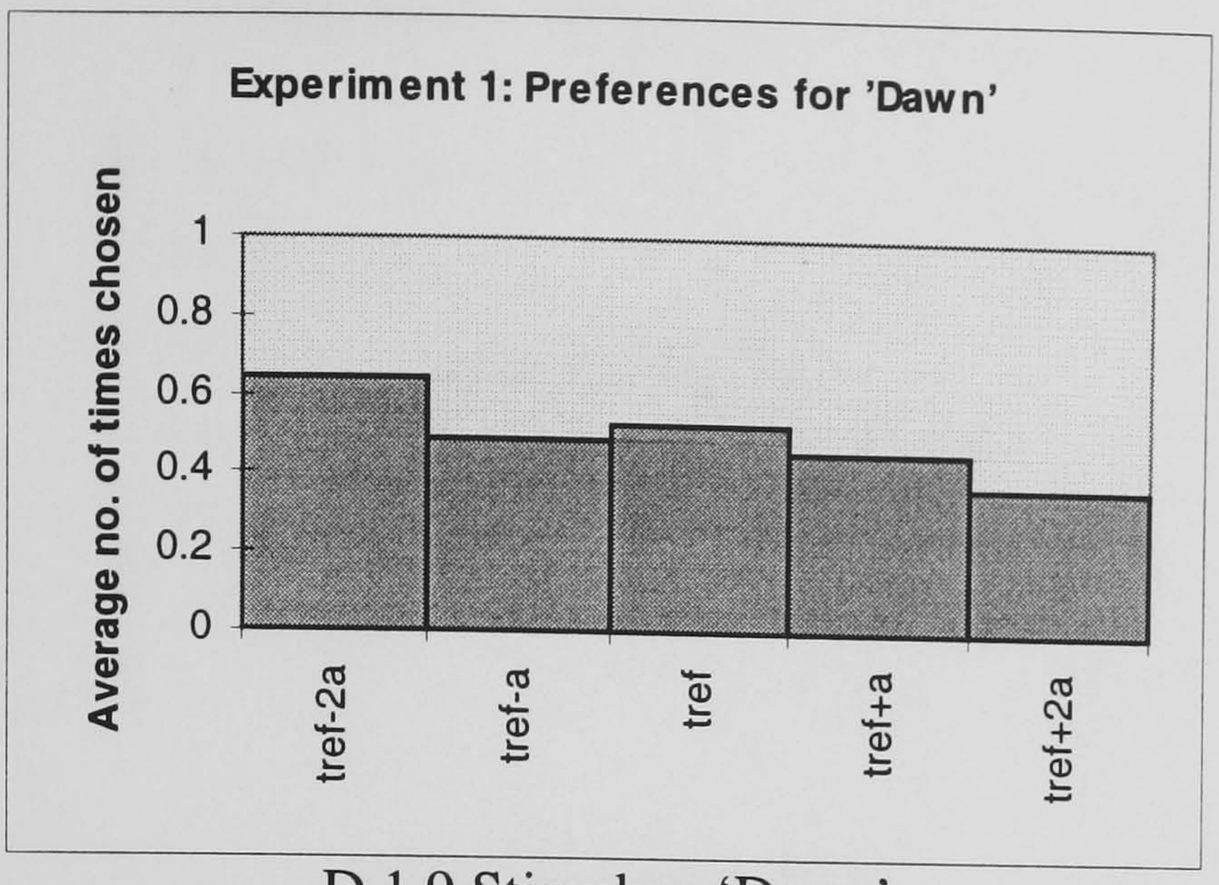
D.1.6 Stimulus: 'Shine'



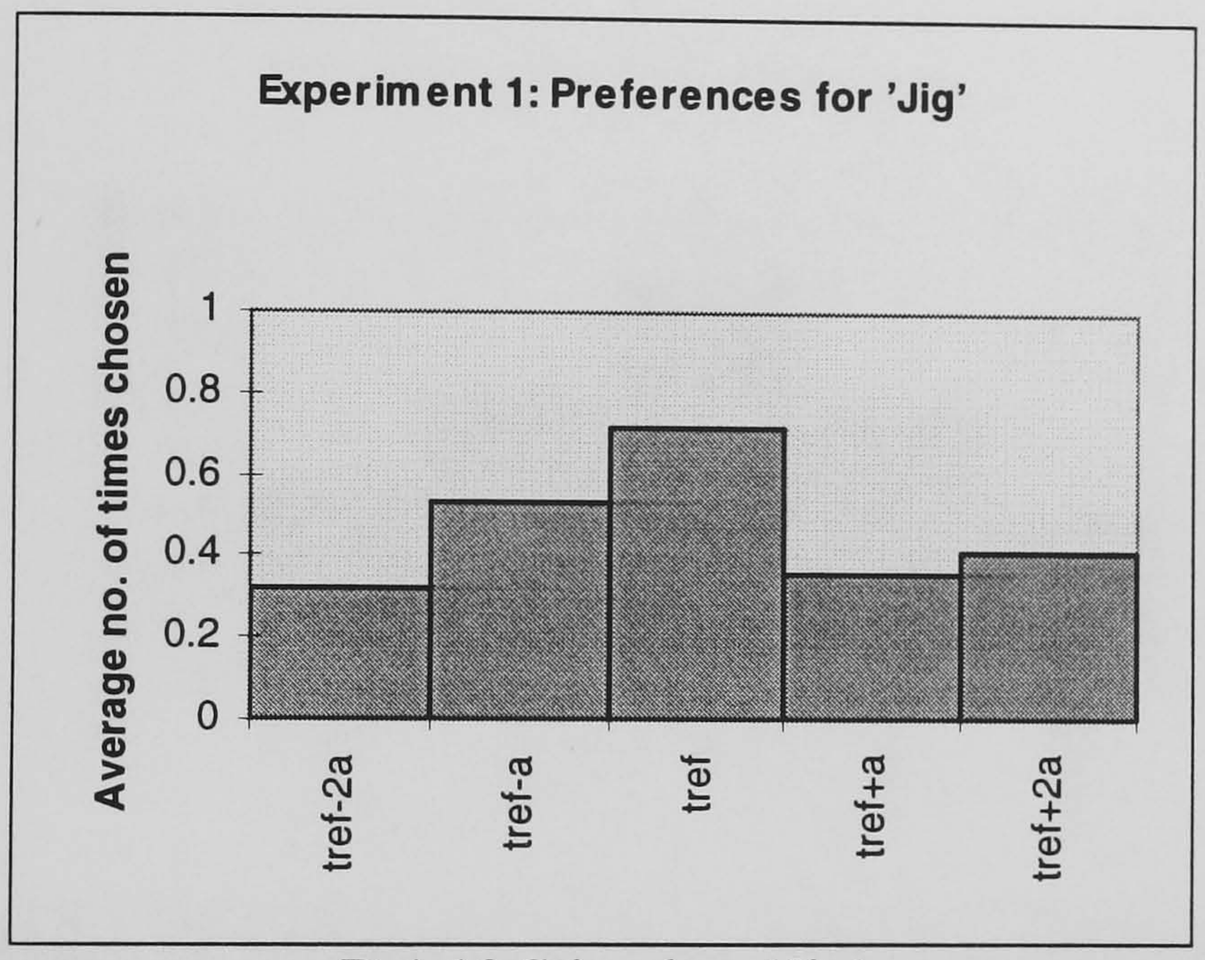
D.1.7 Stimulus: 'Race'



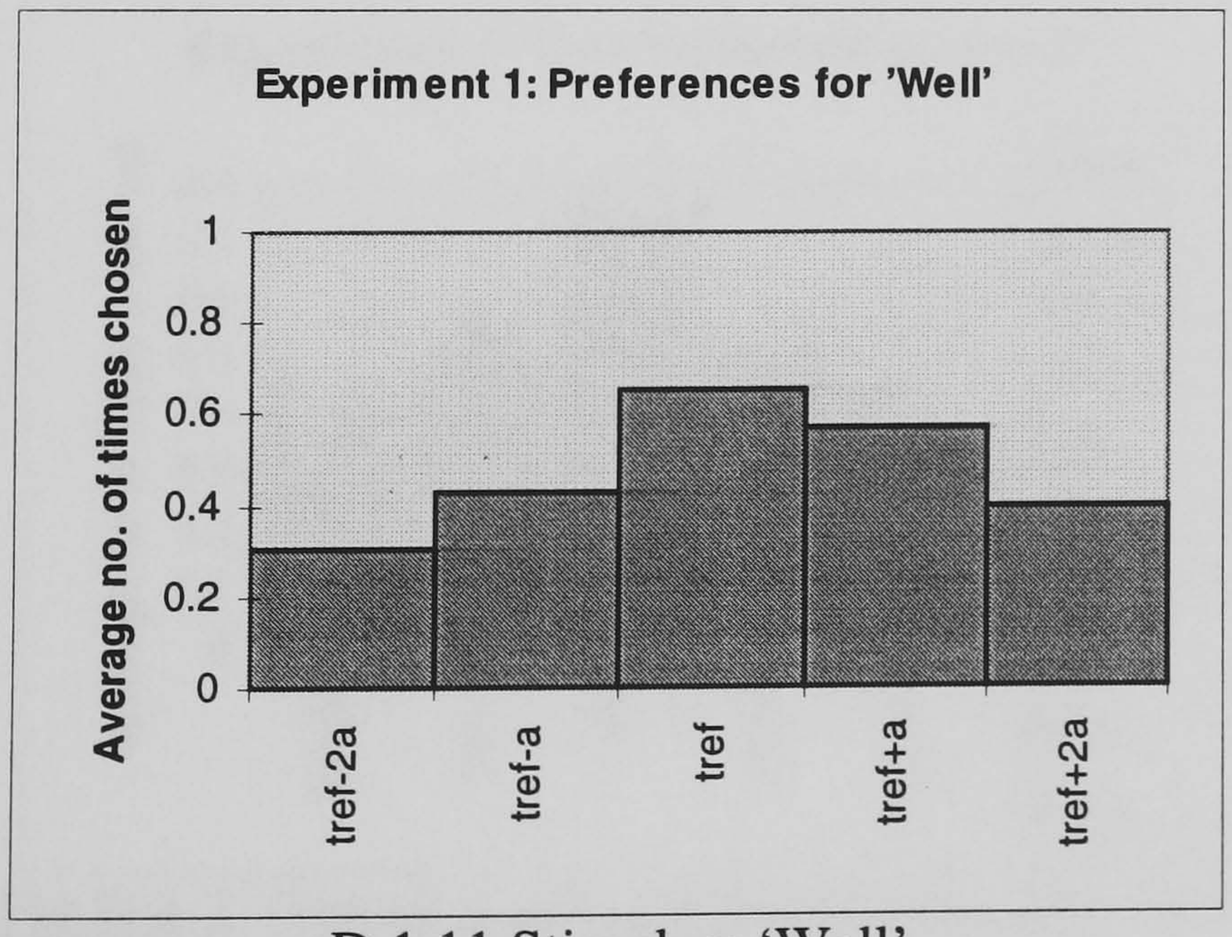
D.1.8 Stimulus: 'Got'



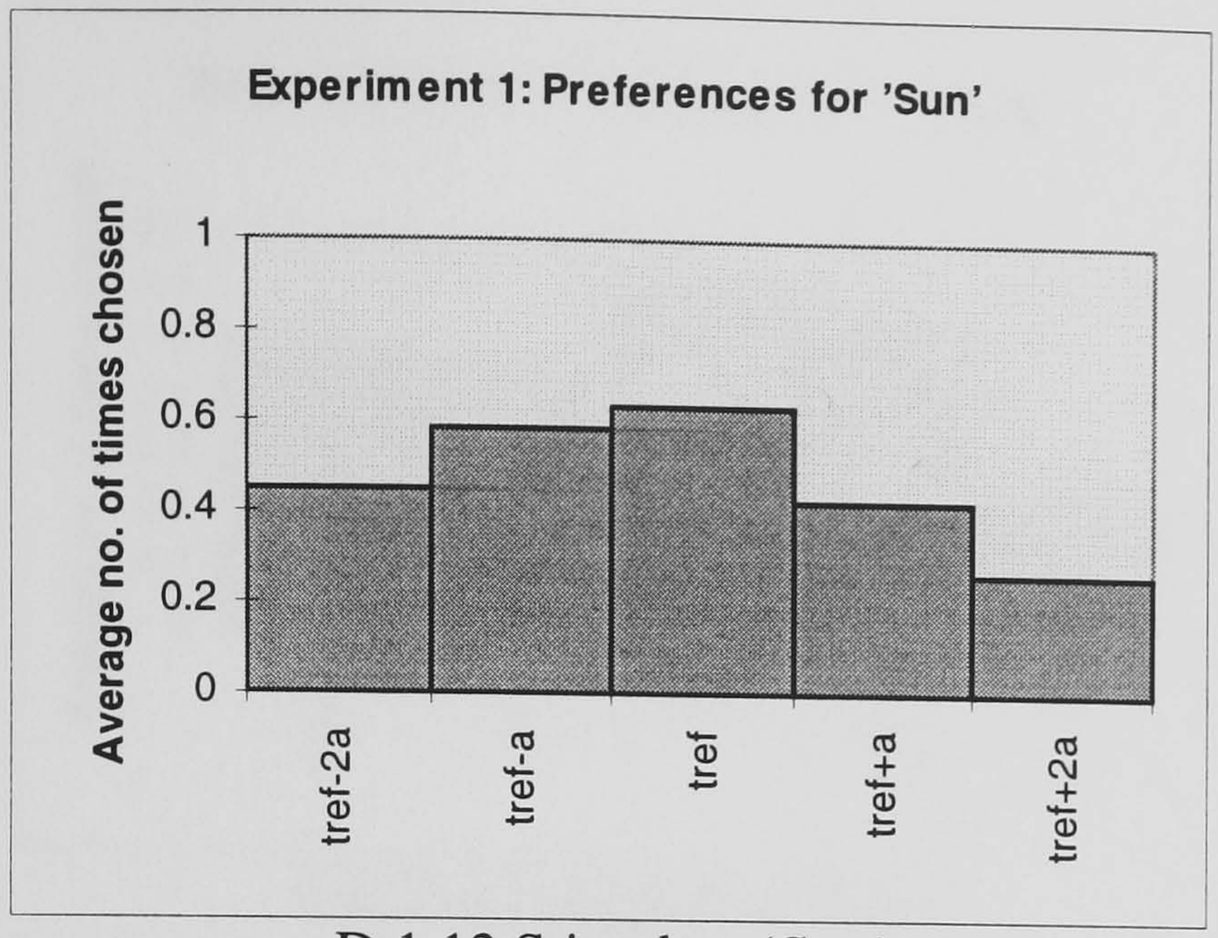
D.1.9 Stimulus: 'Dawn'



D.1.10 Stimulus: 'Jig'



D.1.11 Stimulus: 'Well'



D.1.12 Stimulus: 'Sun'

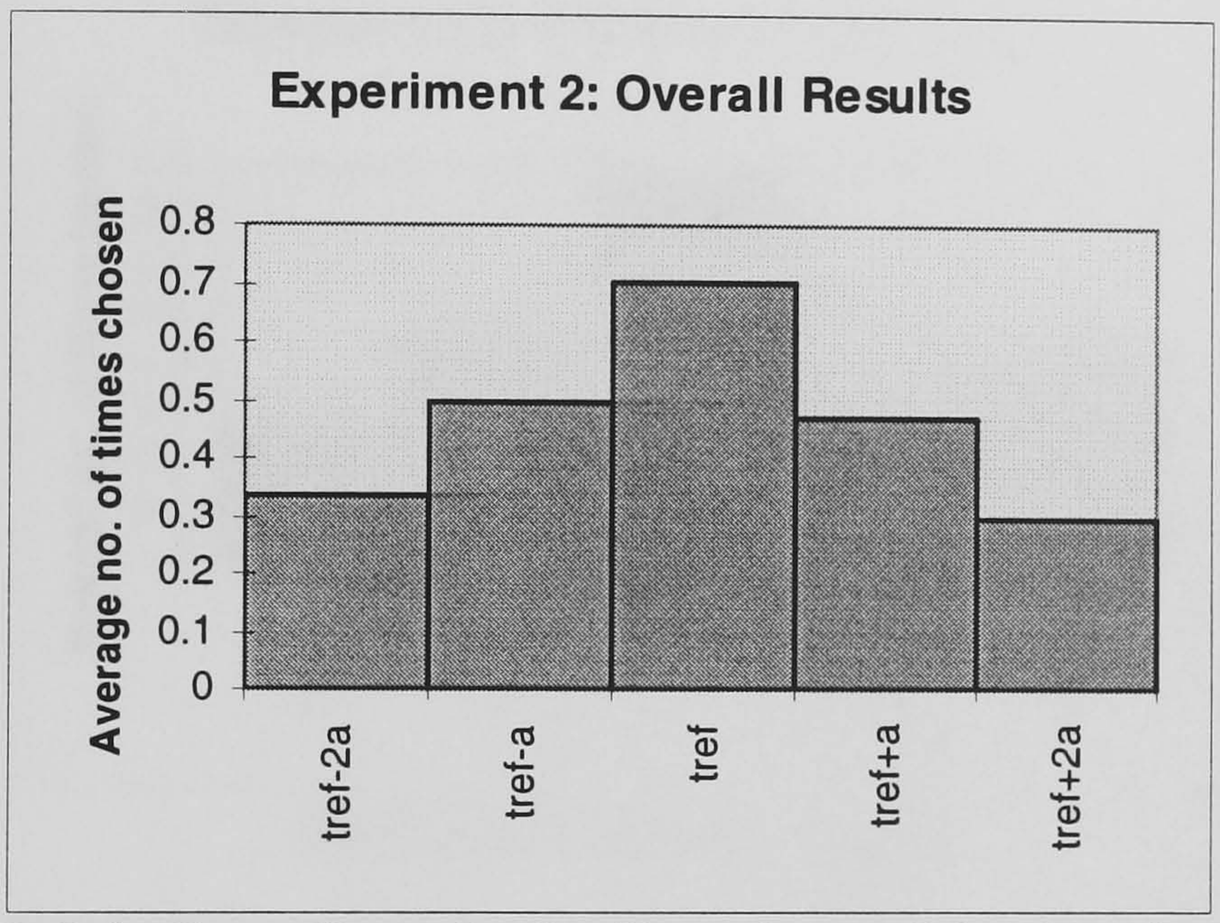


Fig.D.2.1: Overall results for Experiment 2 by duration bin

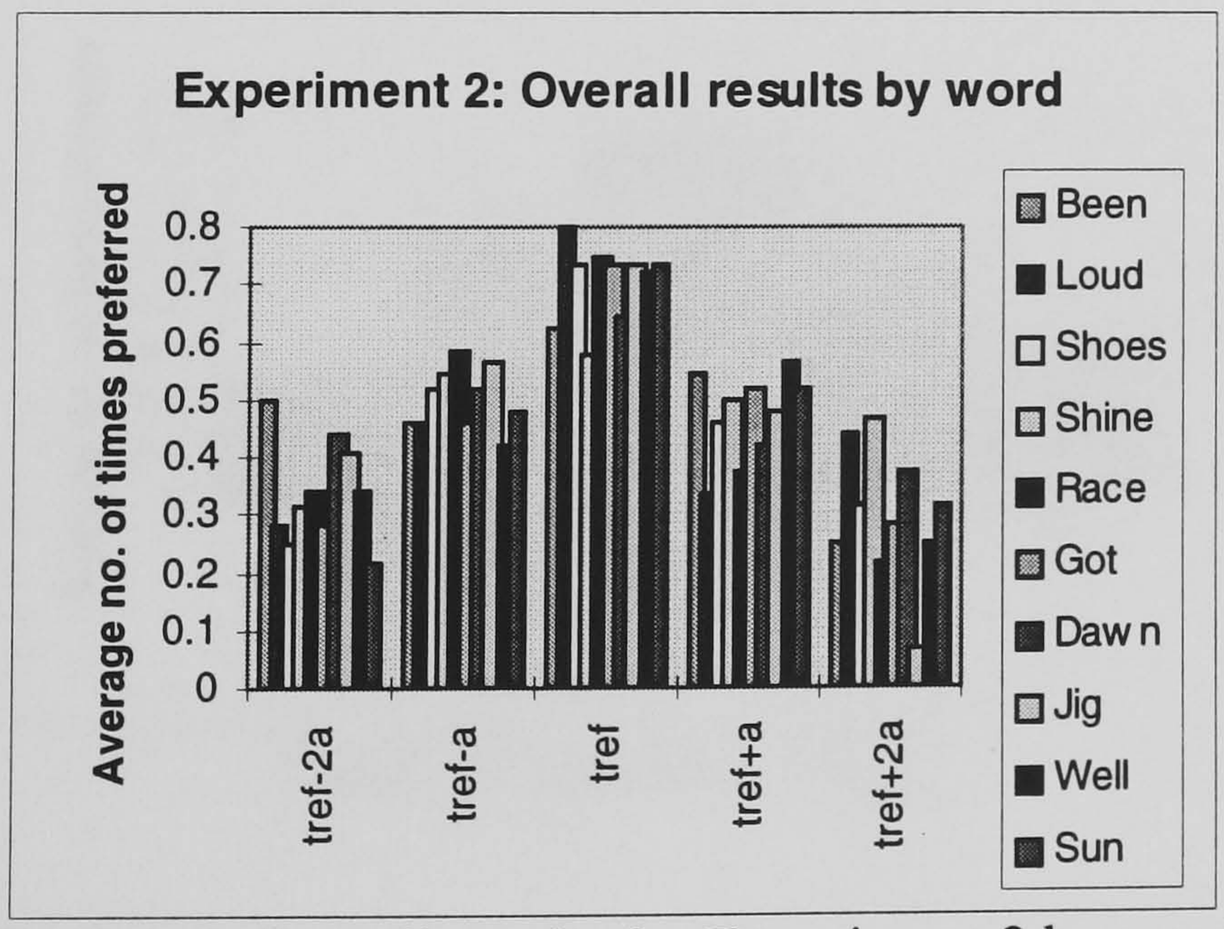


Fig.D.2.2: Overall results for Experiment 2 by word

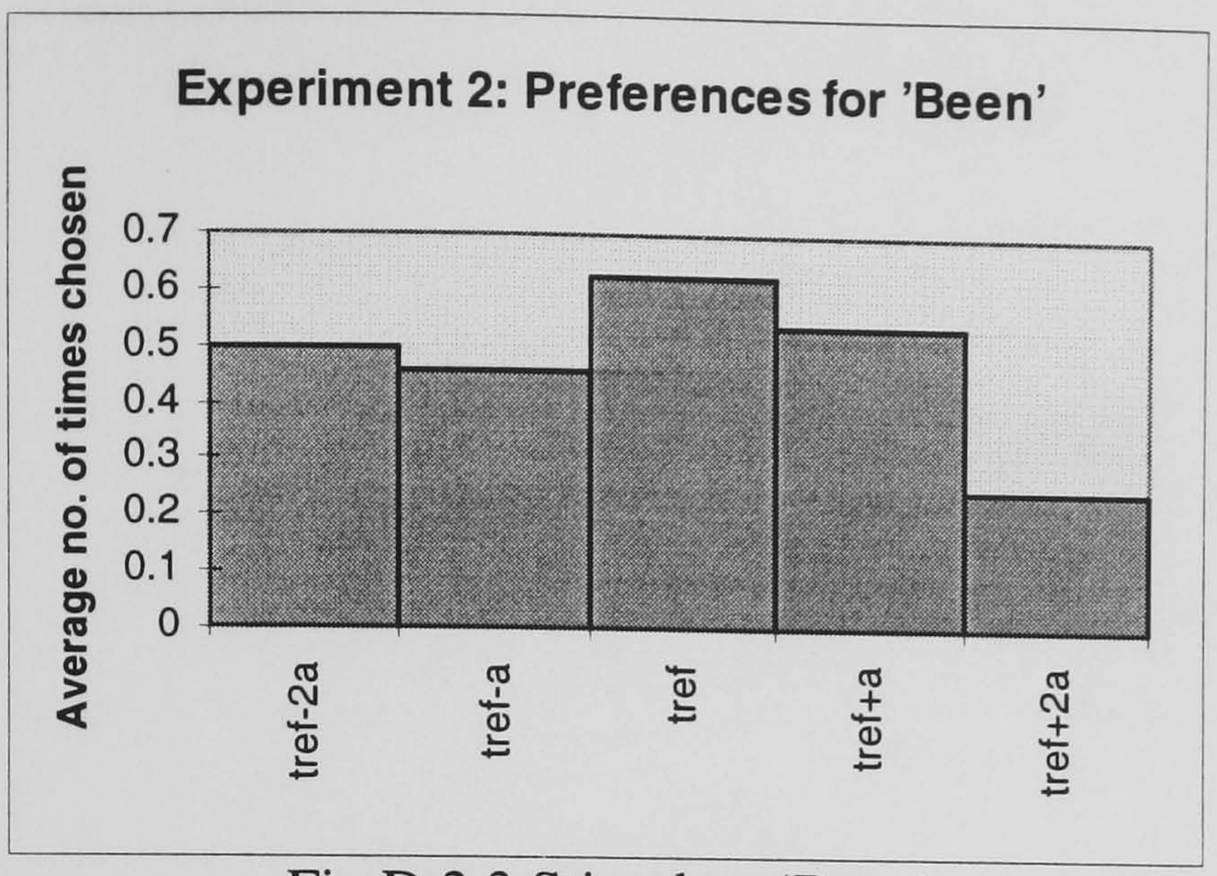


Fig.D.2.3 Stimulus: 'Been'

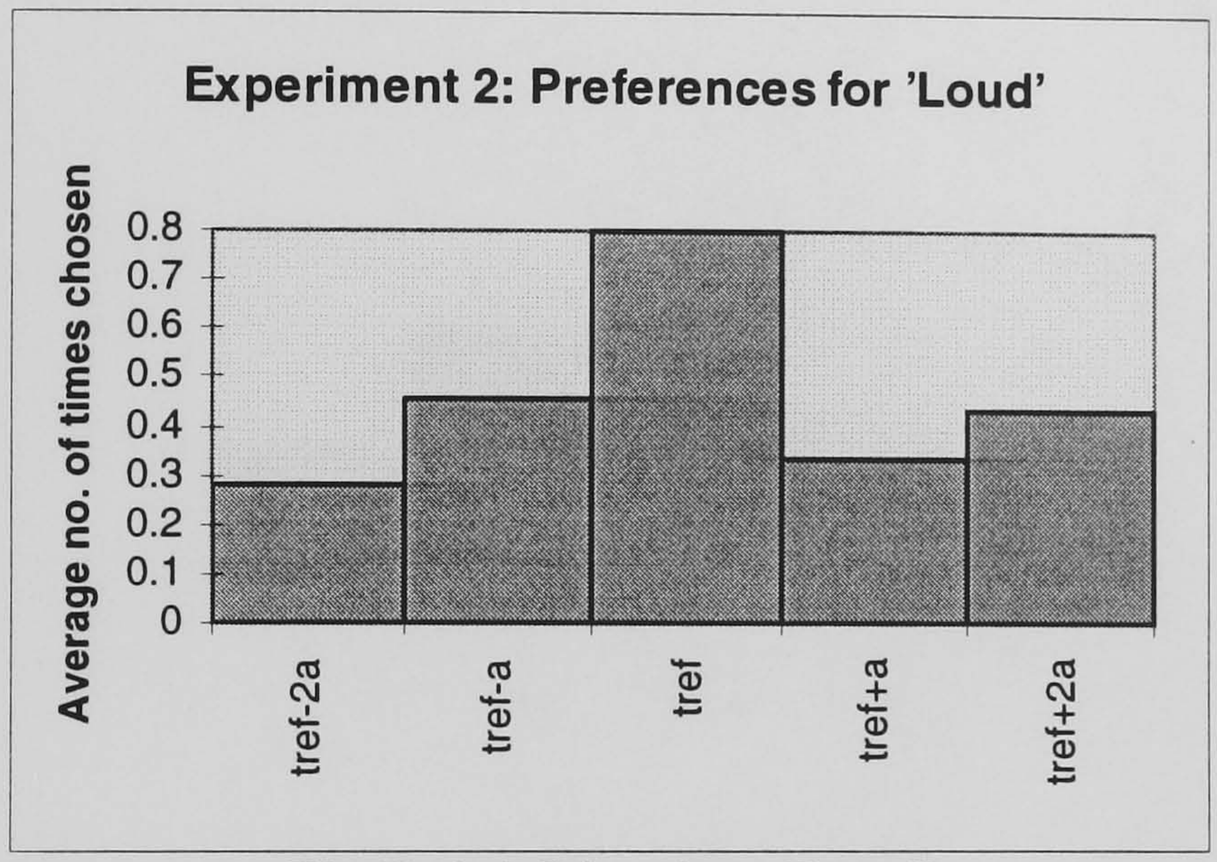


Fig.D.2.4 Stimulus: 'Loud'

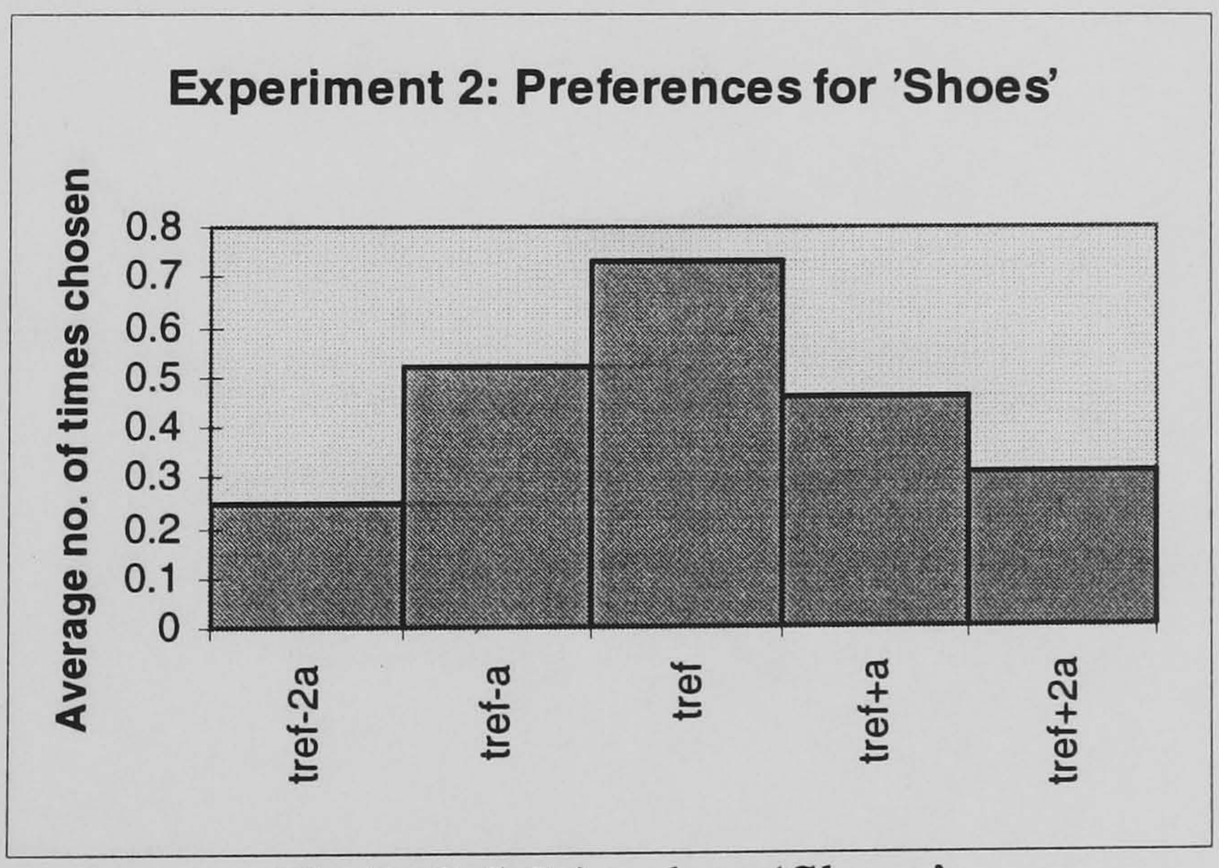


Fig.D.2.5 Stimulus: 'Shoes'

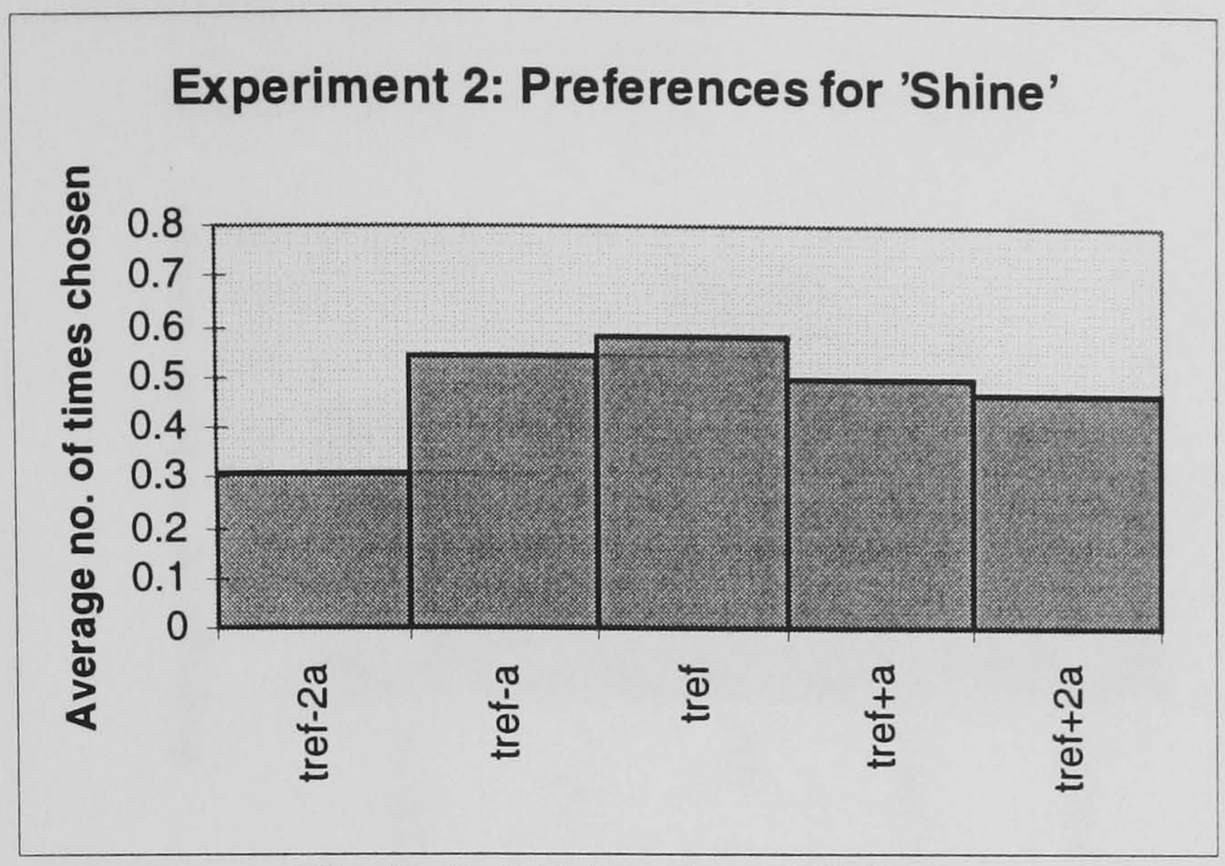


Fig.D.2.6 Stimulus: 'Shine'

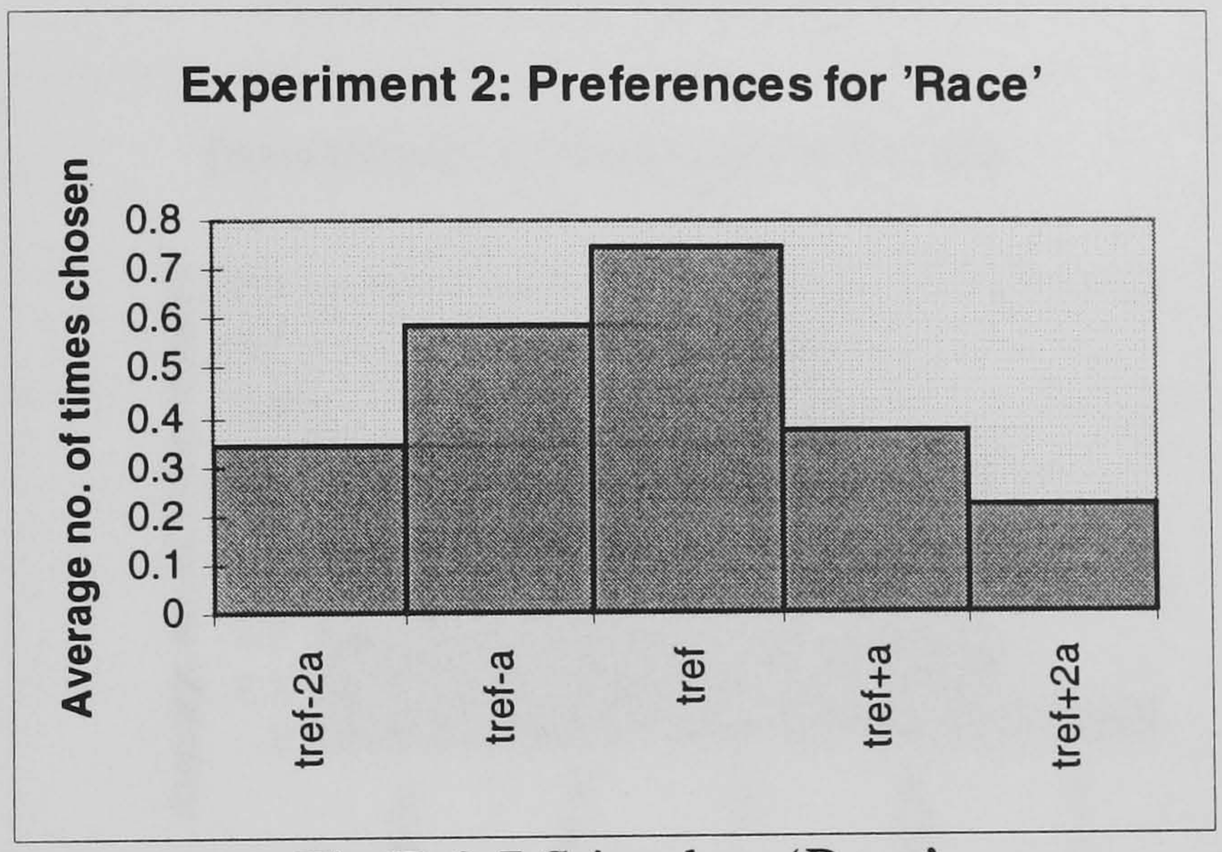


Fig.D.2.7 Stimulus: 'Race'

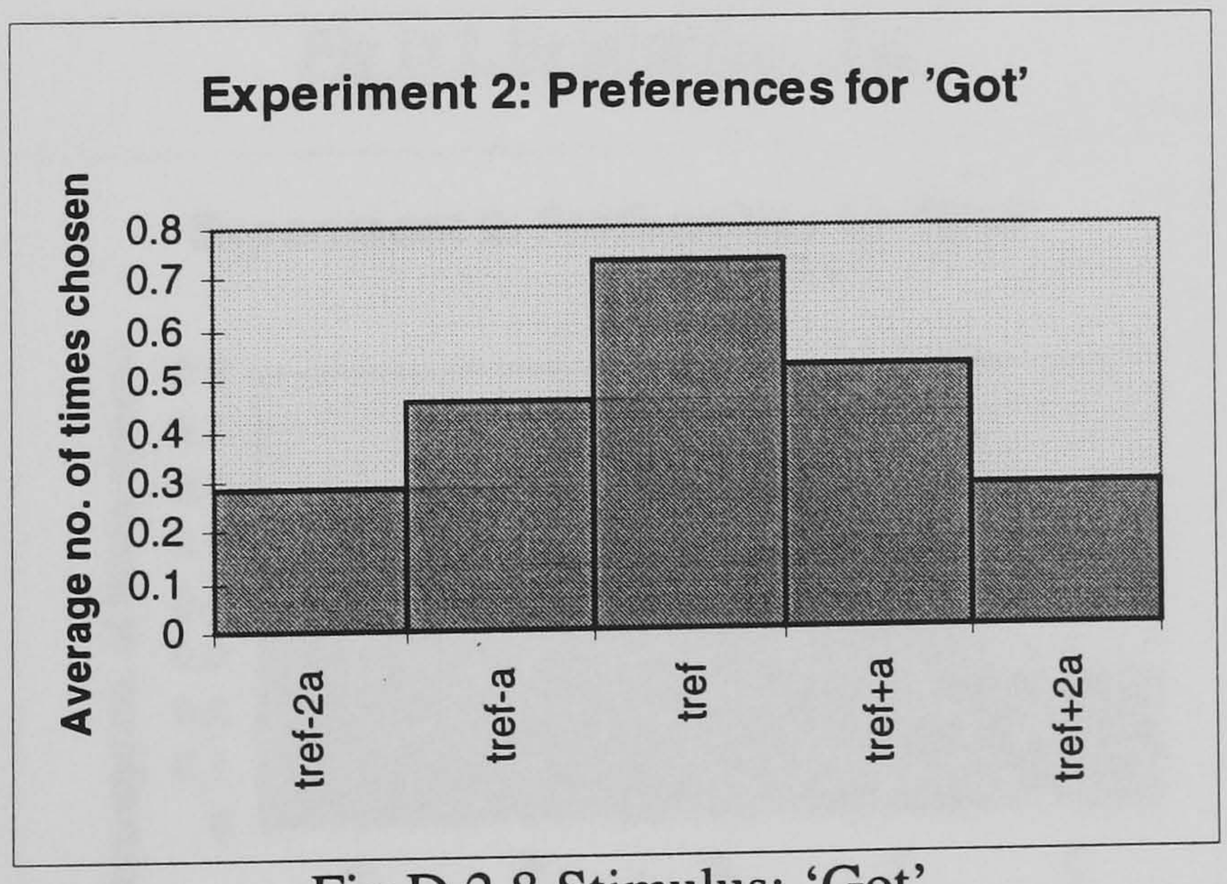


Fig.D.2.8 Stimulus: 'Got'

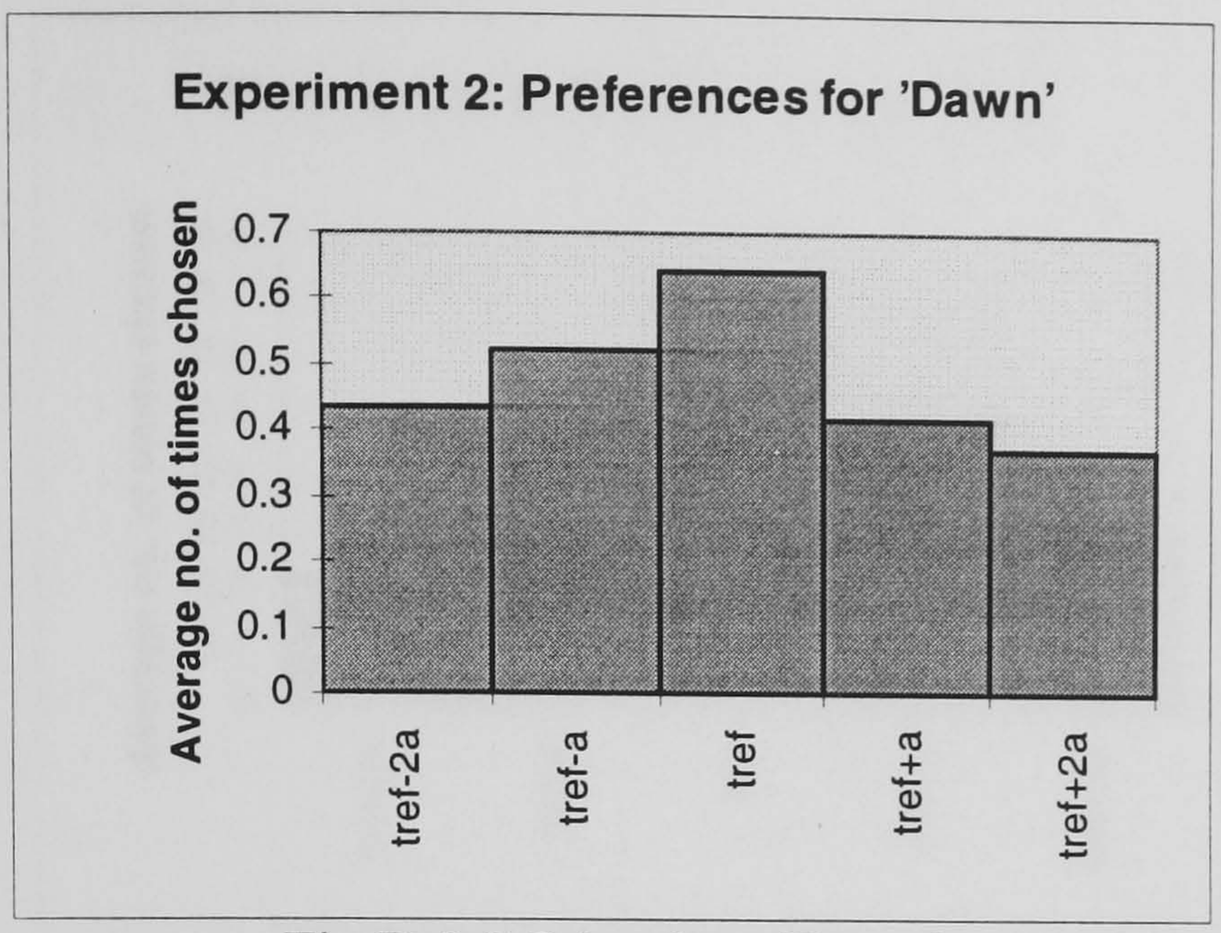


Fig.D.2.9 Stimulus: 'Dawn'

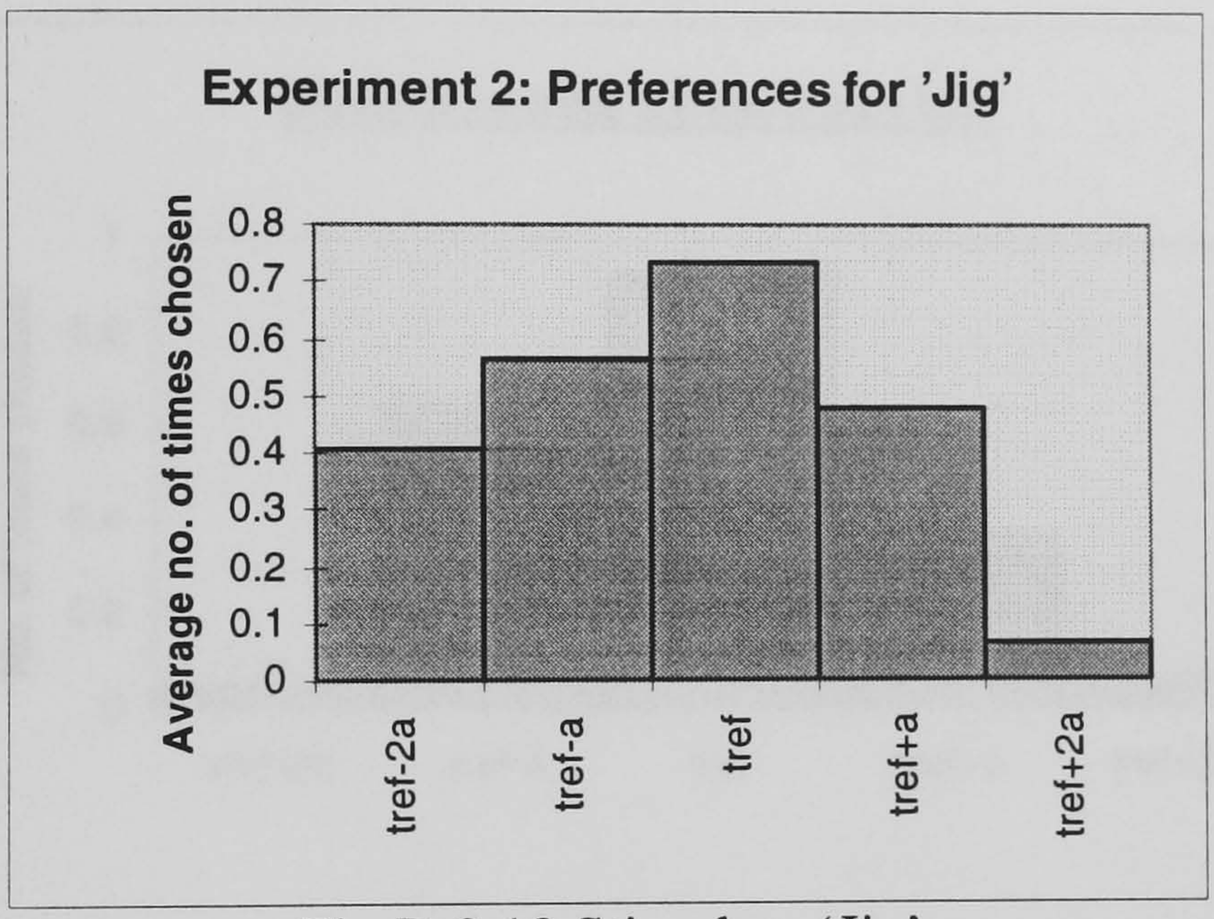


Fig.D.2.10 Stimulus: 'Jig'

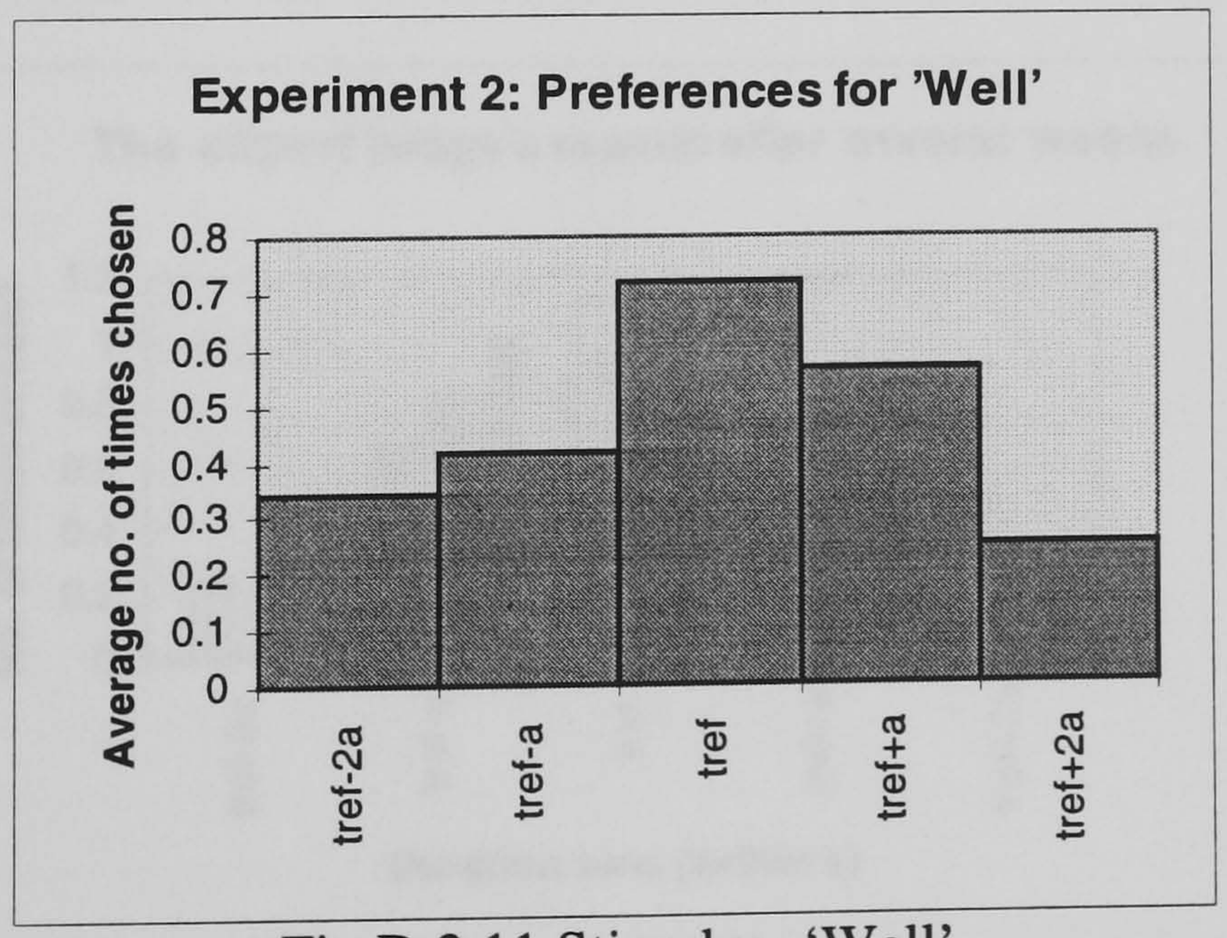


Fig.D.2.11 Stimulus: 'Well'

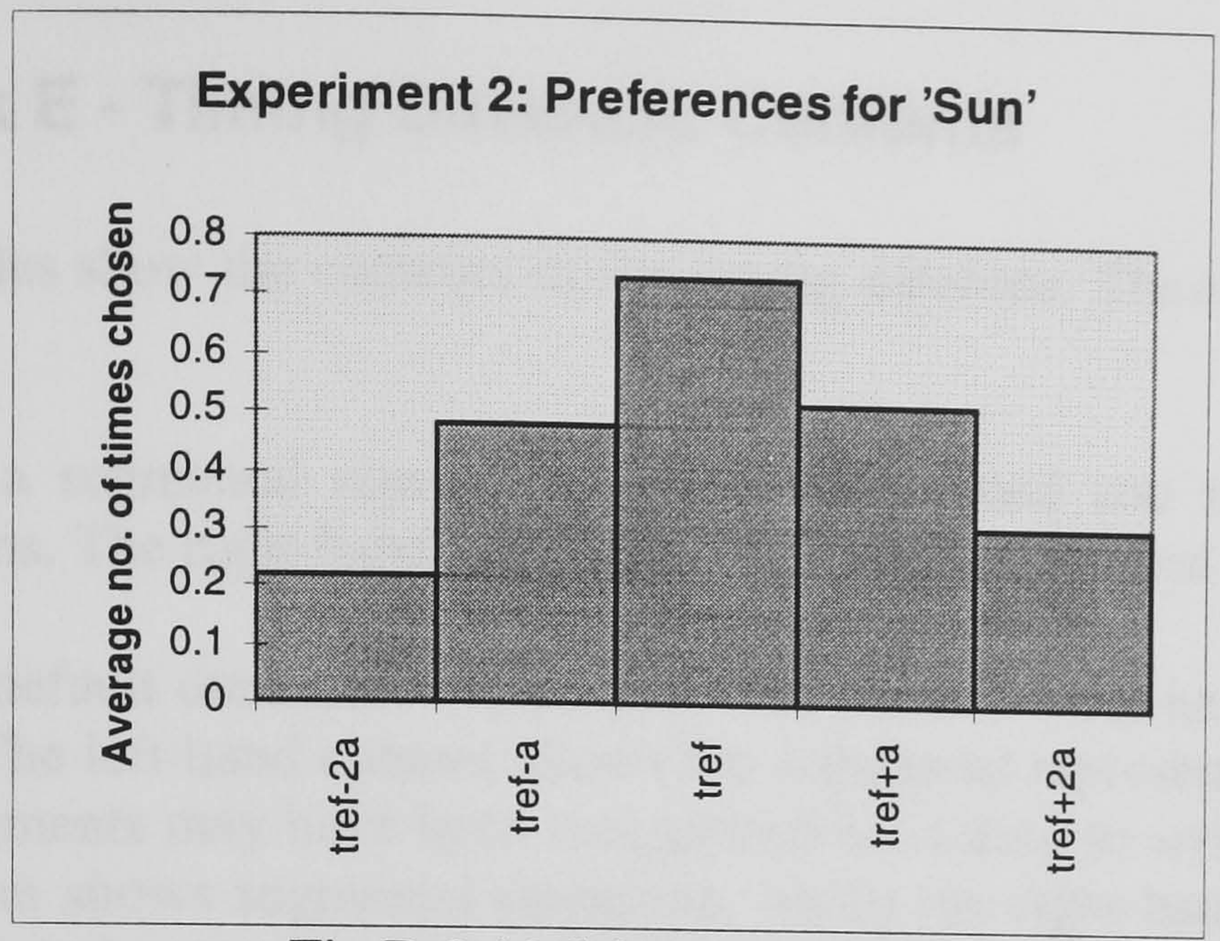


Fig.D.2.12 Stimulus: 'Sun'

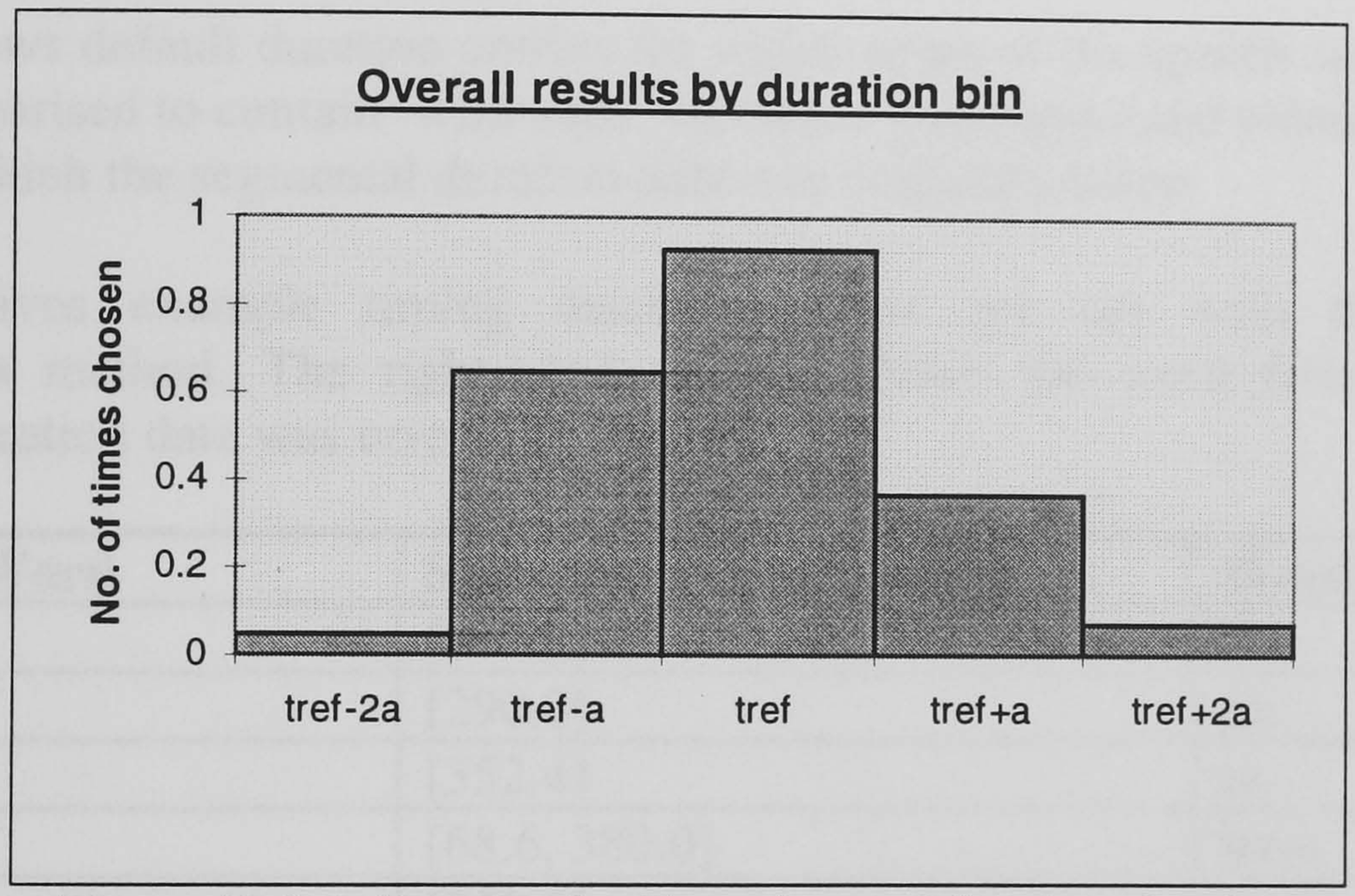


Figure D.3.1: Results for Experiment 3 by duration bin

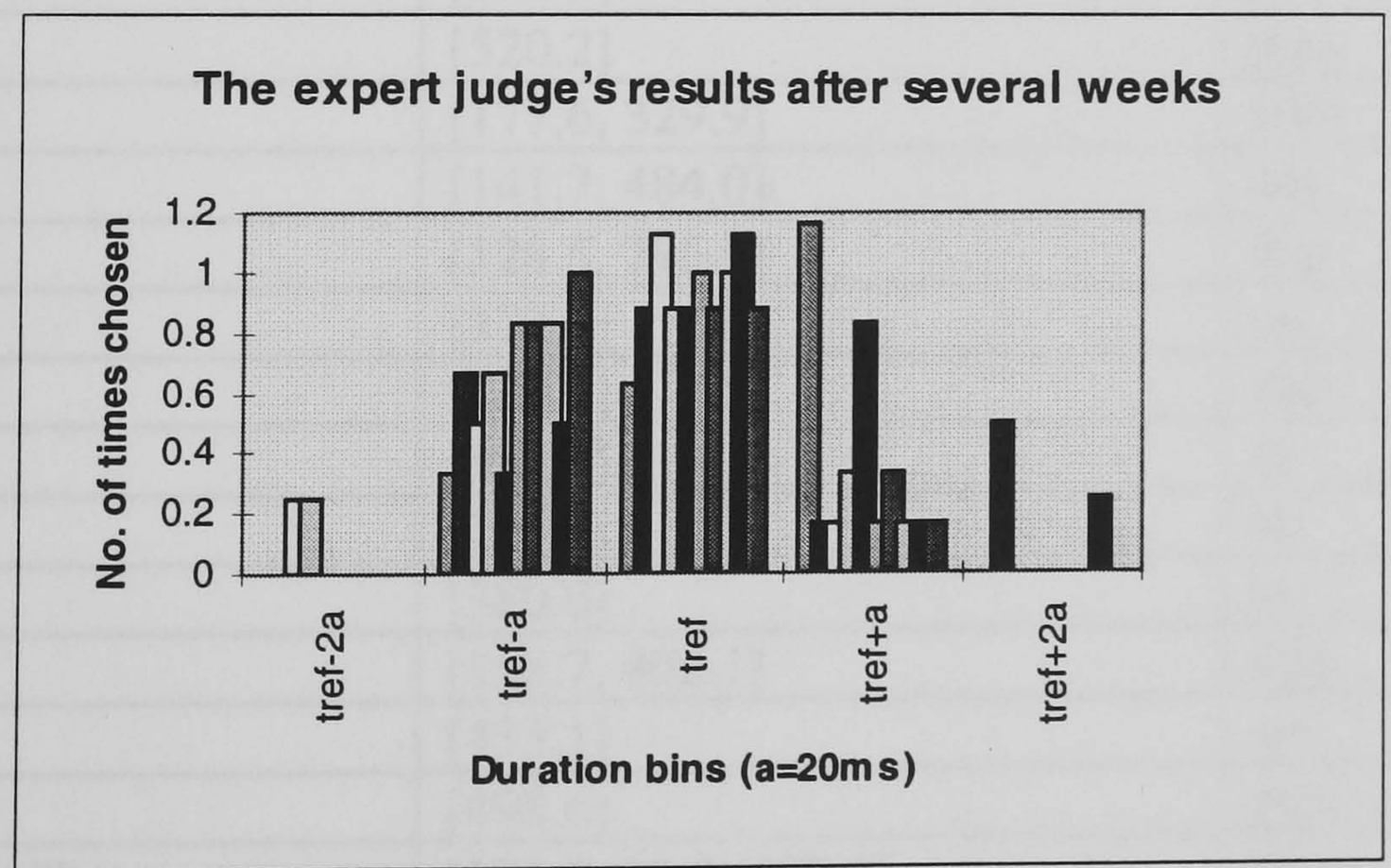


Figure D.3.2: Results for Experiment 3 by word

12. Appendix E - Timing Database Contents

The following tables show the contents of the timing database. These are described in detail below:

Table E1 shows a segmental representation for each word and its corresponding segmental durations. The right-hand column shows the original word itself.

Table E2 shows 'default duration' entries that were derived from existing data in the timing database. The left-hand column shows the segmental representation of a word. Some of these segments may have been categorised according to articulation manner. The middle column shows segmental durations, whilst the right-hand column shows the word from which the segmental duration data was originally taken.

Table E3 shows default duration entries for which some of the speech segments have been re-categorised to contain 'wild-card' elements. The right-hand column shows the word from which the segmental duration data was originally taken.

Table E4 gives example timing database entries for use with the TDCSS2 concatenation method. The right-hand column shows the word from which the segmental duration data was originally taken.

Segmented Word	Segmental Durations (ms)	Word
["in"]	[282.0]	in
["hee"]	[290.0]	he
["tooh"]	[352.4]	to
["ha","av"]	[68.6, 380.0]	have
["it"]	[350.0]	it
["eye"]	[368.3]	I
["dhay"]	[520.2]	they
["wi","idh"]	[177.6, 329.9]	with
["no","ot"]	[141.7, 484.0]	not
["dha","at"]	[128.5, 295.9]	that
["on"]	[379.9]	on
["shee"]	[420.6]	she
["az"]	[388.3]	as
["at"]	[295.9]	at
["beye"]	[380.0]	by
["dhi","is"]	[158.7, 402.1]	this
["wee"]	[435.1]	we
["yooh"]	[498.6]	you
["fr","ro","om"]	[174.2, 92.4, 400.0]	from
["dooh"]	[330.0]	do
["bu","ut"]	[53.9, 387.3]	but
["or"]	[359.0]	or
["an"]	[330.9]	an
["wi","ich"]	[170.1, 374.9]	which

["woo","ood"]	[234.3, 319.9]	would
["say"]	[430.2]	say
["orl"]	[404.6]	all
["wu","un"]	[188.0, 348.1]	one
["wi","il"]	[224.5, 249.9]	will
["hooh"]	[395.3]	who
["we","en"]	[154.7, 360.0]	when
["may","ayc"]	[143.0, 330.0]	make
["dhe","er"]	[127.0, 294.0]	there
["if"]	[351.8]	if
["ca","an"]	[90.3, 330.9]	can
["ma","an"]	[155.9, 330.9]	man
["wo","ot"]	[195.1,484.0]	what
["teye","eyem"]	[86.8, 548.7]	time
["goh"]	[339.9]	go
["noh"]	[450.0]	no
["in","nt","tooh"]	[281.5, 0, 304.2]	into
["coo","ood"]	[109.6, 319.9]	could
["up"]	[423.1]	up
["udh","dhu"]	[166.4, 451.4]	other
["dha","at"]	[137.8, 295.9]	that
["yee","eeu"]	[146.5, 410.0]	year
["owt"]	[480.0]	out
["new"]	[400.0]	new
["su","um"]	[156.3, 282.9]	sum
["tay","ayc"]	[63.3, 330.0]	take
["dhee","eez"]	[145.3, 380.0]	these
["cu","um"]	[103.2, 322.5]	come
["see"]	[439.9]	see
["ge","et"]	[56.7, 340.0]	get
["noh"]	[450.0]	know
["st","tay","ayt"]	[264.2, 0, 346.4]	state
["tooh"]	[352.4]	two
["ohn","nl","lee"]	[343.8,0.0,270.8]	only
["dhe","en"]	[173.1, 360.0]	then
["en","nee"]	[83.1, 380.0]	any
["now"]	[430.0]	now
["may"]	[367.9]	may
["dha","an"]	[168.0, 330.9]	than
["gi","iv"]	[42.2, 389.9]	give
["ub","bow","owt"]	[168.2, 49.5, 480.0]	about
["az"]	[388.3]	as
["day"]	[273.1]	day
["orl","ls","soh"]	[268.9, 0, 563.9]	also
["fur","urs","st"]	[465.5, 0, 260.1]	first
["feye","eyen","nd"]	[169.4, 345.7, 266.1]	find
["way"]	[475.1]	way

["mu","us","st"]	[168.8, 209.8, 380.8]	must
["yooh","oohz"]	[197.7, 469.9]	use
["mor"]	[439.9]	more
["leye","eyec"]	[131.1, 437.0]	like
["eev","vu","un"]	[233.8,145.9, 348.1]	even
["me","en","nee"]	[164.2, 128.2, 380.0]	many
["eech"]	[390.0]	each
["thoh","ohz"]	[224.6, 400.0]	those
["mi","is","st","tu"]	[164.4, 349.2, 0, 281.0]	mister
["ohv","vu"]	[275.6, 294.8]	over
["wur","url","ld"]	[195.5, 263.2, 283.6]	world
["see","eem"]	[187.1, 379.9]	seem
["dgu","us","st"]	[121.2, 227.7, 380.8]	just
["leye","eyef"]	[123.4, 479.5]	life
["bee","eec","cu","um"]	[163.9, 0, 100.7, 330.0]	become
["te","el"]	[117.7, 253.2]	tell
["hee","eeu"]	[73.1, 410.0]	here
["dow","own"]	[49.0, 474.7]	down
["ba","ac"]	[40.0,300.0]	back
["boh","ohth"]	[35.1, 439.9]	both
["goo","ood"]	[45.3, 319.9]	good
["bi","it","tw","wee","een"]	[45.2, 199.1, 90.4, 102.3, 342.0]	between
["st","ti","il"]	[231.7, 95.8, 249.9]	still
["ha","an","nd"]	[76.0, 169.2, 266.1]	hand
["bi","ic","co","oz"]	[37.3, 120.8, 88.5, 419.9]	because
["ve","e-r","ree"]	[150.4, 166.1, 304.0]	very
["thi","ing"]	[123.0, 339.5]	thing
["ahf","ft","tu"]	[280.6, 225.2, 296.5]	after
["dow","own"]	[42.2, 470.4]	down
["ne","ev","vu"]	[289.8, 0, 320.1]	never
["un","nu","udh","dhu"]	[189.7, 260.8, 0, 451.4]	another
["un","nd","du"]	[331.0, 0, 300.0]	under
["sc","cooh","oohl"]	[192.6, 0, 320.0]	school
["cay","aym"]	[114.0, 427.4]	came
["wur","urc"]	[188.0, 399.9]	work
["sl","loh"]	[133.8, 4]	slow
["ohl","ld"]	[165.6, 283.6]	old
["gr","ray","ayt"]	[46.1, 123.1, 346.4]	great
["ha","av"]	[75.6, 373.0]	have
["moh","ohs","st"]	[334.6, 0, 380.8]	most
["weye","eyel"]	[134.3, 511.1]	while
["pah","aht"]	[83.8, 358.8]	part
["thr","ree"]	[135.7, 419.9]	three
["ta","ad"]	[90.3, 290.0]	tad
["few"]	[479.9]	few
["cheye","eyel","ld"]	[125.2, 283.6, 283.6]	child
["wu","un"]	[163.1, 348.1]	one

["bi","ig","gi","in"]	[122.9, 0, 60.0, 287.2]	begin
["wi","ith","dhow","owt"]	[166.3, 245.8, 0, 480.0]	without
["fee","eel"]	[144.1, 279.9]	feel
["nu","um","mb","bu"]	[142.5, 299.5, 0, 292.4]	number
["pl","lay","ays"]	[47.8, 111.5, 363.7]	place
["wu","un","nt"]	[326.1, 0, 500.0]	want
["soh"]	[550.4]	so
["ub","bow","owt"]	[230.3, 0, 480.0]	about
["lah","ahs","st"]	[122.2, 268.2, 273.5]	last
["dee","ori","ing"]	[81.6, 313.5, 339.5]	during
["ahs","sc"]	[236.6, 321.8]	ask

Table E1: Original timing database entries

Segmented Word	**Segmental Durations (ms)**	**Original Word**
["liqu","unas"]	[188.0, 348.1]	one
["plosr","ray","ayt"]	[46.1, 123.1, 346.4]	great
["friceye","eyen","nd"]	[169.4, 345.7, 266.1]	find
["fricur","urs","st"]	[465.5, 0, 260.1]	first
["frica","at"]	[128.5, 295.9]	that
["nasa","an"]	[155.9, 330.9]	man
["nase","en","nee"]	[164.2, 128.2, 380.0]	many
["nasu","us","st"]	[168.8, 209.8, 380.8]	must
["plosa","an"]	[90.3, 330.9]	can
["liqo","ot"]	[195.1,484.0]	what
["liqoo","ood"]	[234.3, 319.9]	would
["liqu","un"]	[188.0, 348.1]	one
["liqeye","eyec"]	[131.1, 437.0]	like
["liqi","ich"]	[170.1, 374.9]	which
["plosu","ut"]	[53.9, 387.3]	but
["plosi","iv"]	[42.2, 389.9]	give
["plosay"]	[273.1]	day
["liqi","il"]	[224.5, 249.9]	will
["liqe","en"]	[154.7, 360.0]	when
["nasay","ayc"]	[143.0, 330.0]	make
["nasor"]	[439.9]	more
["frice","e-r"]	[127.0, 294.0]	there
["ploseye","eyem"]	[86.8, 548.7]	time
["plosoh"]	[339.9]	go
["nasoh"]	[450.0]	no
["ustp"]	[423.1]	up
["fricro","om"]	[174.2, 92.4, 400.0]	from
["frica","av"]	[68.6, 380.0]	have
["frici","ing"]	[123.0, 339.5]	thing
["frice","e-r","ree"]	[150.4, 166.1, 304.0]	very
["plosi","ic","co","oz"]	[37.3,120.8,88.5,419.9]	because

["nasah","ahc"]	[136.1, 474.9]	mark
["frica","an","nd"]	[76.0, 169.2, 266.1]	hand
["plosow","own"]	[42.2, 470.4]	down
["nase","ev","vu"]	[289.8, 0, 320.1]	never
["liqah","ahs","st"]	[122.2, 268.2, 273.5]	last
["ustp","bow","owt"]	[230.3, 0, 480.0]	about
["fricoh"]	[550.4]	so
["liqu","un","nt"]	[326.1, 0, 500.0]	want
["plosl","lay","ays"]	[47.8, 111.5, 363.7]	place
["nasu","um","mb","bu"]	[142.5, 299.5, 0, 292.4]	number
["fricee","eel"]	[144.1, 279.9]	feel
["liqi","ith","thow","owt"]	[166.3, 245.8, 0, 480.0]	without
["plosi","ig","gi","in"]	[122.9, 0, 60.0, 287.2]	begin
["plosoo","ood"]	[109.6, 319.9]	could

Table E2: Default duration entries (where some speech segments are categorised by articulation manner)

Segmented Word	**Segmental Durations (ms)**	**Original Word**
["*e","e-r","ree"]	[150.4, 166.1, 304.0]	very
["*eye","eyem"]	[86.8, 548.7]	time
["*i","ith"]	[177.6, 329.9]	with
["*a","av"]	[68.6, 380.0]	have
["*o","ot"]	[141.7, 484.0]	not
["*a","at"]	[128.5, 295.9]	that
["*a","a*"]	[128.5, 295.9]	that
["*i","is"]	[158.7, 402.1]	this
["*ro","om"]	[174.2, 92.4, 400.0]	from
["*u","ut"]	[53.9, 387.3]	but
["*i","ich"]	[170.1, 374.9]	which
["*oo","ood"]	[234.3, 319.9]	would
["*u","un"]	[188.0, 348.1]	one
["*e","en"]	[154.7, 360.0]	when
["*ay","ayc"]	[143.0, 330.0]	make
["*e","e-r"]	[127.0, 294.0]	there
["*a","an"]	[90.3, 330.9]	can
["*o","ot"]	[195.1,484.0]	what
["*i","ic","co","oz"]	[37.3, 120.8, 88.5, 419.9]	because
["*ow","own"]	[42.2, 470.4]	down
["*e","ev","vu"]	[289.8, 0, 320.1]	never
["*ah","ahs","st"]	[122.2, 268.2, 273.5]	last
["*b","bow","owt"]	[230.3, 0, 480.0]	about
["*oh"]	[550.4]	so
["*u","un","nt"]	[326.1, 0, 500.0]	want
["*l","lay","ays"]	[47.8, 111.5, 363.7]	place
["*u","um","mb","bu"]	[142.5, 299.5, 0, 292.4]	number

["*ee","eel"]	[144.1, 279.9]	feel
["*i","ith","thow","owt"]	[166.3, 245.8, 0, 480.0]	without
["*i","ig","gi","in"]	[122.9, 0, 60.0, 287.2]	begin
["*ah","ahc"]	[136.1, 474.9]	mark
["*i","il"]	[224.5, 249.9]	will

Table E3: General default duration entries (containing 'wild-card' elements)

Segmented Word	**Segmental Durations (ms)**	**Original Word**
["te","el","lu","uv", "veye","eyez"]	[[0,81.9],[0,253.2], [0,0],[68.0,284.2], [0,151.6], [81.5,579.2]]	televise
["cay","aym","mb", "br","ri","idg"]	[[0,38.4],[0,349.2], [0,0],[0,82.4],[0,110.0, [0,336.7]]	Cambridge
["sp","pee","eec"]	[[0,289.9],[49.0,296.5], [189.2,350.0]]	speak
["st","to","op"]	[[0,260.6],[0,0], [0,322.0]]	stop
["ne","ev","vu"]	[[0,299.1],[0,0], [49.9,415.4]]	never
["si","ing","gl"]	[[21.4,171.8], [0,248.2],[0,289.9]]	single
["mu","us","st"]	[[0,136.0],[0,298.6], [148.5,380.8]]	must
["way","ayv","vf","for","orm"]	[[0,184.0],[28.7, 372.8],[0,0],[0,92.0], [0,420.0]]	waveform
["wi","ith","thow","owt"]	[[0,279.7],[0,329.9], [0,86.9],[0,480.0]]	without
["ahs","sc"]	[[0,387.9], [159.5,210.8]]	ask
["moh","ohs","st"]	[[0,369.7],[0,0], [0,380.8]]	most
["st","ti","il"]	[[0,249.9],[0,0], [0,249.9]]	still
["de","ef","fi","in","nu","ut", "tl","lee"]	[[0,45.0],[21.9, 247.1],[0,0],[5.2,200.272], [22.3,310.9], [0,0],[24.2,302.2]]	definitely
["st","tay","ayt"]	[[10.7,263.3],[0,0], [22.4,345.9]]	state
["tor","orc","ci","ing"]	[[0,83.3],[12.9,269.274], [3.0,91.6],[0,339.5]]	talking
["co","on","nc","ca","at","tu", "un","nay","aysh","shu","un"]	[[0,62.8],[18.7,282.5],[0,0],[34.1,111.6],[35.0,197.5], [84.7,281.0],[0,0], [0,354.4],[0,0],	concatenation

	[24.5,161.7],[43.1,348.1]]	
["cu","um","mp","pew", "ooht","tu"]	[[0,73.8],[0,222.2], [0,0],[0,343.0],[0,0], [0,382.5]]	computer
["le","ecs","si","ic","co","on"]	[[0,214.2], [165.4,499.9], [178.5,341.9], [0,208.9], [0,37.5],[0,286.3]]	lexicon

Table E4: Example timing database entries for use with the TDCSS2 concatenation method

13. Appendix F - Timing Module Code

This section contains LINGO code for the Timing Module, which predicts segmental durations, when given an input string of speech segments. LINGO is a scripting language for Macromedia's Director software (Macromedia, 1998).

This code includes the Default Duration Procedure, which is carried out whenever the duration look-up algorithm produces incomplete timing information.

```
--Initialisation routine, executed before the main procedure
on startMovie
  global InDifs
  set InDifs=[]
  global TmpRec
  set TmpRec=[]
  setListUp                    --set up Timing Database
  setTypesUp                   --define speech segment categories
  global difcount
  global tmpstr                --a temporary string variable
  set tmpstr=EMPTY
  set the text of cast "output" to EMPTY
  set the text of cast "newBox" to EMPTY
  set the text of cast "TimeRezBox" to EMPTY
  set the text of cast "AltBox" to EMPTY
  global piws                  --contains position-in-word information
  set piws=[]
end

--This is the main procedure in the Timing Module

on enterFrame
  pause          --wait for input from keyboard
end

on exitFrame
  global phons, FinalVals, InDifs, ErrList
  set phons=[]
  set FinalVals=[]
  altSearch                    --search for segmental durations
  output
  if count(ErrList)>0 then
    getRest                    --get any missing durations
  end if
  set phons=InDifs
  set tlist=FinalVals
end
--end of main procedure
```

```
--input speech segments from keyboard
on keyUp
 global InDifs, tmpstr          --InDifs is the list of speech segments
 if the key=ENTER then          --for which durations must be predicted
  doctorString
  append InDifs, tmpstr
  continue
 else if the key=" " then
  doctorString
  append InDifs, tmpstr
  set tmpstr=EMPTY
 else if the key>="a" and the key<="z" or the key="*" then
  set tmpstr=tmpstr && the key
 end if
end

--tidy up the input characters
on doctorString
 global tmpstr
 set i=1
 repeat while i<=length(tmpstr)
  if char i of tmpstr=" " then
   delete char i of tmpstr
  end if
  set i=i+1
 end repeat
end

--default duration prediction
--involving re-categorisation of speech segments
on getRest
 global ErrList, inDifs, typeList        --ErrList contains the 'missing' segments
 set typeList=[]
 repeat with i=1 to count(ErrList)       --re-categorise 'missing' segments
  set rez=getSegType(getAt(ErrList,i))
  append typeList, rez
 end repeat
 set the text of cast "AltBox" to string(typeList)
 newSearch                               --search Timing Database again
end


--this is a search which looks for the 'missing' segments
on newSearch
 global temp, typeList, InDifs, FinalVals, ErrList
 set typeCnt=1
 repeat with i=1 to count(Indifs)
  set rez=getPos(ErrList, getAt(Indifs,i))
  if rez>0 then                          --it was a missing segment
   setAt InDifs, i, getAt(typeList, typeCnt)
```

```
    set typeCnt=typeCnt+1
  end if
end repeat
set FinalVals=[]
secondSearch                    --now call the search engine again
classifyAgain                           --recategorise any missing segments
set errCt=1
repeat with i=1 to count(finalVals)
  if getAt(FinalVals,i)=-1 then
    setAt(Indifs,i,getAt(errlist,errCt))
    set errCt=errCt+1
    end if
  end repeat
  set FinalVals=[]
  secondSearch                  --call the search engine a final time
  Output                                  --output final list of durations
end

-- segment recategorization procedures
on classifyAgain
  global errList
  --reclassify ErrList (ie replace phoneme types with wild-card segments)
  repeat with i=1 to count(ErrList)
    set str=replaceWithStar(getAt(errList,i))
    setAt(errList,i,str)                  --put updated segment types into ErrList
  end repeat
end

on replaceWithStar myStr
  if mystr starts "stp" or myStr starts "liq" or myStr starts "nas" then
    put "*" into char 1 of myStr
    delete char 2 to 3 of myStr
    return mystr
  else if mystr starts "plos" or myStr starts "fric" then
    put "*" into char 1 of myStr
    delete char 2 to 4 of myStr
    return mystr
  else if mystr contains "plos" or myStr contains "fric" then
    delete char (the number of chars in myStr-3) to the number of chars in myStr of
myStr
    put "*" after myStr
    return mystr
  else if mystr contains "stp" or myStr contains "liq" or myStr contains "nas" then
    delete char (the number of chars in myStr-2) to the number of chars in myStr of
myStr
    put "*" after myStr
    return mystr
  else
    RETURN EMPTY
  end if
```

```
end
--end of segment recategorization procedures


--the main segmental duration prediction procedure
on AltSearch
 --various initialisations of variables
 global difcount, NUMTEMPS, PosArray, InDifs, Ltemp,Temp,times,Errlist,
sizes,piws
 set the text of cast "NewBox" to "Segments not in database: "
 set ErrList=[]
 set difcount=count(InDifs)
 set posArray=[]
 global TmpRec, TmpNumRec
 set TmpRec=[]
 set TmpNumRec=[]
 set CurrentD=1
 set stop=FALSE
 --end of initialisation section
 repeat while (CurrentD<=difcount)
  set diph=getAt(InDifs,currentD)
  set piw=1
  set sizes=[]
  set found=FALSE
  repeat with i=1 to count(temp)
   set rez=getPos(getAt(temp,i),diph)
   if rez>0 then                          --found current segment,
    checkAhead(i,rez,currentD)            --so look for further matches
    set found=true
    append sizes, count(getAt(temp,i))
    setAt(PosArray,piw,rez)               --note position in word
    set piw=piw+1
   end if
  end repeat
  if not found then
   set the text of cast "NewBox" to the text of cast "NewBox" &&
getAt(InDifs, CurrentD)
   append ErrList, getAt(InDifs, CurrentD)
   set phonType=GetSegType(getAt(InDifs, CurrentD))
   append FinalVals, -1
   set currentD=currentD+1
  else
   set largest=nBS(currentD)              --find largest sub-string
   AppendToFinal(largest,currentD-1)      --use its duration values
   piwstuff(largest)                      --position-in-word checks
   set currentD=currentD+(count(getAt(TmpRec,largest)))
   set TmpRec=[]
   set TmpNumRec=[]
  end if
 end repeat
```

```
end

--checks the current Timing Database entry for further matches
on checkAhead TempInd, TempJ, curD
  global InDifs, Temp, Times, TmpRec, TmpNumRec
  set scratchTlist=[]
  set scratchNameList=[]
  append scratchTlist, getAt(getAt(times, TempInd),TempJ)
  append scratchNameList, getAt(getAt(Temp,TempInd),TempJ)
  set endReached=FALSE
  set i=2
  set nextOne=TempJ+1
  repeat while not endReached and (curD+1<=count(InDifs)) and¬
(nextOne<=count(getAt(Temp,TempInd)))
    if getAt(getAt(Temp, TempInd),nextOne)=getAt(InDifs,curD+1) then
      set tval=getAt(getAt(Times, TempInd),nextOne)
      set nval=getAt(getAt(Temp, TempInd),nextOne)
      append scratchTlist, tval
      append scratchNameList, nval
      set nextOne=nextOne+1
      set curD=curD+1
      set i=i+1
    else
      set endReached=TRUE
    end if
  end repeat
  append TmpRec, scratchNameList
  append TmpNumRec, scratchTlist
end

--function to return the largest sub-string of several candidates
on nBS InPos
  global PosArray, TmpRec
  set BestSize=0
  set BestOne=0
  repeat with i=1 to count(TmpRec)
    set tmpcount=count(getAt(TmpRec,i))
    if (tmpcount>BestSize) then
      set BestOne=i
      set BestSize=tmpcount
    else if (tmpcount=BestSize) then
      set dif1=getAt(PosArray,i)-InPos
      set dif2=getAt(PosArray,BestOne)-InPos
      if (dif1<0) then set dif1=-1*dif1
      if (dif2<0) then set dif2=-1*dif2
      if (dif1<dif2) then set BestOne=i
      if dif1=dif2 then
        set sizeDif1=count(InDifs)-getAt(sizes,BestOne)
        set sizeDif2=count(InDifs)-getAt(sizes,i)
```

```
      if sizeDif1<0 then set sizeDif1=-1*sizeDif1
      if sizeDif2<0 then set sizeDif2=-1*sizeDif2
      if sizeDif2<=sizeDif1 then
        set BestOne=i
      end if
    end if
  end if
 end repeat
 return BestOne
end


--position-in-word checks
on piwStuff largest
 global piws, PosArray
 set startpos=getAt(PosArray,largest)
 repeat with i=1 to count(getAt(TmpRec,largest))
  append piws, (startpos+i)-1
 end repeat
end


--procedure to perform a further search if the first one fails
--to return a duration for each input segment
on SecondSearch
 global difcount, NUMTEMPS, PosArray, InDifs, Ltemp,Temp,times,Errlist,
sizes,piws, TmpRec, TmpNumRec, FinalVals
 --initialisation section
 set ErrList=[]
 set difcount=count(InDifs)
 set posArray=[]
 set TmpRec=[]
 set TmpNumRec=[]
 set CurrentD=1
 set stop=FALSE
 --end of initialisation section
 repeat while (CurrentD<=difcount)
  set diph=getAt(InDifs,currentD)
  set piw=1
  set sizes=[]
  set found=FALSE
  repeat with i=1 to count(temp)
   set rez=getPos(getAt(temp,i),diph)
   if rez>0 then
    checkAhead(i,rez,currentD)
    set found=true
    append sizes, count(getAt(temp,i))
    setAt(PosArray,piw,rez)
    set piw=piw+1
   end if
```

```
  end repeat
  if not found then        --output error message
    set the text of cast "NewBox" to the text of cast "NewBox" && "After Second
Search: "
    set the text of cast "NewBox" to the text of cast "NewBox" &&
getAt(InDifs, CurrentD)
    append ErrList, getAt(InDifs, CurrentD)
    append FinalVals, -1
    set currentD=currentD+1
  else
    set largest=nBS(currentD)                    --select the largest sub-string
    AppendToFinal(largest,currentD-1)            --note the relevant duration values
    piwstuff(largest)                            --position-in-word checks
    set currentD=currentD+(count(getAt(TmpRec,largest)))
    set TmpRec=[]
    set TmpNumRec=[]
  end if
 end repeat
end

--output predicted segmental duration values to text boxes
on output
 global InDifs
 global FinalVals
 set the text of field "output" to " "
 set the text of field "TimeRezBox" to " "
 repeat with i=1 to count(InDifs)
  put getAt(InDifs,i) after field "output"
  put " " after field "output"
 end repeat
 repeat with i=1 to count(FinalVals)
  put getAt(FinalVals,i) after field "TimeRezBox"
  put " " after field "TimeRezBox"
 end repeat
end

--procedure to append the segmental duration values to the final duration list
on AppendToFinal biggest, prevpos
 global FinalVals                        --final duration list
 global TmpNumRec
 if TmpNumRec=[] then
  alert "error!"
 else
  set j=1
  repeat while j<=count(getAt(TmpNumRec,biggest))
   set val=getAt(getAt(TmpNumRec,biggest),j)
   setAt(FinalVals,prevpos+1,val)
   set j=j+1
   set prevpos=prevpos+1
  end repeat
```

```
 end if
end

--function to automatically identify a segment's category
--in terms of liquid, fricative, nasal or plosive

on GetSegType seg
 global plos, nas, fric, stp, vowels, fricMenu, plosMenu, liqMenu, nasMenu
 --check liquids
 repeat with i=1 to count(liqMenu)
  set res=getPos(getAt(liqMenu,i), seg)
  if res>0 then                          --segment found
   set segType=getAt(liqMenu,i)
   set st=reverseLookUp(segType)
   return st
  end if
 end repeat

 --check fricatives
 repeat with i=1 to count(fricMenu)
  set res=getPos(getAt(fricMenu,i), seg)
  if res>0 then                          --segment found
   set segType=getAt(fricMenu,i)
   set st=reverseLookUp(segType)
   return st
  end if
 end repeat

 --check nasals
 repeat with i=1 to count(nasMenu)
  set res=getPos(getAt(nasMenu,i), seg)
  --res is the position within the "Menu" type lists
  if res>0 then                          --segment found
   set segType=getAt(nasMenu,i)
   set st=reverseLookUp(segType)
   return st
  end if
 end repeat

 --check plosives
 repeat with i=1 to count(plosMenu)
  set res=getPos(getAt(plosMenu,i), seg)
  if res>0 then                          --segment found
   set segType=getAt(plosMenu,i)
   set st=reverseLookUp(segType)
   return st
  end if
 end repeat
end
--end of section
```

```
--function to categorise segments
on reverseLookUp segType

--plosive-vowel combinations
 if segType=["ca","ba","pa","ta","da","ga"] then
  return "plosa"
 else if segType=["cah","bah","pah","tah","dah","gah"] then
  return "plosah"
 else if segType=["cay","bay","pay","tay","day","gay"] then
  return "plosay"
 else if segType =["ce","be","pe","te","de","ge"] then
  return "plose"
 else if segType=["ceye","beye","peye","teye","deye","geye"] then
  return "ploseye"
 else if segType=["cee","bee","pee","tee","dee","gee"] then
  return "plosee"
 else if segType=["ci","bi","pi","ti","di","gi"] then
  return "plosi"
 else if segType=["cl","bl","pl","tl","dl","gl"] then
  return "plosl"
 else if segType=["co","bo","po","to","do","go"] then
  return "ploso"
 else if segType=["coo","boo","poo","too","doo","goo"] then
  return "plosoo"
 else if segType=["cor","bor","por","tor","dor","gor"] then
  return "plosor"
 else if segType=["coh","boh","poh","toh","doh","goh"] then
  return "plosoh"
 else if segType=["coy","boy","poy","toy","doy","goy"] then
  return "plosoy"
 else if segType=["cooh","booh","pooh","tooh","dooh","gooh"] then
  return "plosooh"
 else if segType=["cu","bu","pu","tu","du","gu"] then
  return "plosu"
 else if segType=["cew","bew","pew","tew","dew","gew"] then
  return "plosew"

  --assorted diphones
 else if segType=["cl","bl","pl","tl","dl","gl"] then
  return "plosl"
 else if segType=["cn","bn","pn","tn","dn","gn"] then
  return "plosn"
 else if segType=["cr","br","pr","tr","dr","gr"] then
  return "plosr"
 else if segType=["cs","bs","ps","ts","ds","gs"] then
  return "ploss"
 else if segType=["shn","dgn","thn","vn","fn","sn","zn","chn"] then
  return "fricn"
```

```
--nasal-vowel combinations
else if segType=["na","ma"] then
   return "nasa"
  else if segType=["nah","mah"] then
   return "nasah"
  else if segType=["nay","may"] then
   return "nasay"
  else if segType=["ne","me"] then
   return "nase"
  else if segType=["nee","mee"] then
   return "nasee"
  else if segType=["neye","meye"] then
   return "naseye"
  else if segType=["ni","mi"] then
   return "nasi"
  else if segType=["no","mo"] then
   return "naso"
  else if segType=["noo","moo"] then
   return "nasoo"
  else if segType=["nooh","mooh"] then
   return "nasooh"
  else if segType=["nor","mor"] then
   return "nasor"
  else if segType=["noh","moh"] then
   return "nasoh"
  else if segType=["noy","moy"] then
   return "nasoy"
  else if segType=["nu","mu"] then
   return "nasu"
  else if segType=["nur","mur"] then
   return "nasur"
  else if segType=["new","mew"] then
   return "nasew"
  --liquid-vowel combinations
  else if segType=["wa","la"] then
   return "liqa"
  else if segType=["wah","lah"] then
   return "liqah"
  else if segType=["way","lay"] then
   return "liqay"
  else if segType=["we","le"] then
   return "liqe"
  else if segType=["weye","leye","reye"] then
   return "liqeye"
  else if segType=["wee","lee"] then
   return "liqee"
  else if segType=["wi","li"] then
   return "liqi"
  else if segType=["wo","lo"] then
   return "liqo"
```

```
else if segType=["woo","loo"] then
 return "liqoo"
else if segType=["wooh","looh"] then
 return "liqooh"
else if segType=["wor","lor"] then
 return "liqor"
else if segType=["woh","loh"] then
 return "liqoh"
else if segType=["woy","loy"] then
 return "liqoy"
else if segType=["wu","lu"] then
 return "liqu"
else if segType=["wew","lew"] then
 return "liqew"
else if segType=["sha","dga","tha","va","fa","sa","za","ha","cha"] then
 return "frica"
else if segType=["shah","dgah","thah","vah","fah","sah","zah","hah","chah"] then
 return "fricah"
else if segType=["shay","dgay","thay","vay","fay","say","zay","hay","chay"] then
 return "fricay"
else if segType=["she","dge","the","ve","fe","se","ze","he","che"] then
 return "frice"
else if
segType=["sheye","dgeye","theye","veye","feye","seye","zeye","heye","cheye"] then
 return "friceye"
else if segType=["shee","dgee","thee","vee","fee","see","zee","hee","chee"] then
 return "fricee"
else if segType=["shi","dgi","thi","vi","fi","si","zi","hi","chi"] then
 return "frici"
else if segType=["sho","dgo","tho","vo","fo","so","zo","ho","cho"] then
 return "frico"
else if segType=["shoo","dgoo","thoo","voo","foo","soo","zoo","hoo","choo"] then
 return "fricoo"
else if
segType=["shooh","dgooh","thooh","vooh","fooh","sooh","zooh","hooh","chooh"]
then
 return "fricooh"
else if segType=["shor","dgor","thor","vor","for","sor","zor","hor","chor"] then
 return "fricor"
else if segType=["shoh","dgoh","thoh","voh","foh","soh","zoh","hoh","choh"] then
 return "fricoh"
else if segType=["shoy","dgoy","thoy","voy","foy","soy","zoy","hoy","choy"] then
 return "fricoy"
else if segType=["shu","dgu","thu","vu","fu","su","zu","hu","chu"] then
 return "fricu"
else if segType=["shew","dgew","thew","vew","few","sew","zew","hew","chew"]
then
 return "fricew"
else if segType=["ac","ab","ap","at","ad","ag"] then
 return "astp"
```

```
else if segType=["ahc","ahb","ahp","aht","ahd","ahg"] then
 return "ahstp"
else if segType=["ayc","ayb","ayp","ayt","ayd","ayg"] then
 return "aystp"
else if segType=["ec","eb","ep","et","ed","eg"] then
 return "estp"
else if segType=["eyec","eyeb","eyep","eyet","eyed","eyeg"] then
 return "eyestp"
else if segType=["eec","eeb","eep","eet","eed","eeg"] then
 return "eestp"
else if segType=["ic","ib","ip","it","id","ig"]  then
 return "istp"
else if segType=["oc","ob","op","ot","od","og"] then
 return "ostp"
else if segType=["ooc","oob","oop","oot","ood","oog"] then
 return "oostp"
else if segType=["orc","orb","orp","ort","ord","org"] then
 return "orstp"
else if segType=["ohc","ohb","ohp","oht","ohd","ohg"] then
 return "ohstp"
else if segType=["oyc","oyb","oyp","oyt","oyd","oyg"] then
 return "oystp"
else if segType=["oohc","oohb","oohp","ooht","oohd","oohg"] then
 return "oohstp"
else if segType=["uc","ub","up","ut","ud","ug"] then
 return "ustp"
else if segType=["ewc","ewb","ewp","ewt","ewd","ewg"] then
 return "ewstp"

 --vowel-nasal combinations
else if segType=["an","ang","am"] then
 return "anas"
else if segType=["ahn","ahm"] then
 return "ahnas"
else if segType=["ayn","aym"] then
 return "aynas"
else if segType=["en","eng","em"] then
 return "enas"
else if segType=["een","eem"] then
 return "eenas"
else if segType=["eyen","eyem"] then
 return "eyenas"
else if segType=["in","ing","im"] then
 return "inas"
else if segType=["on","ong","om"] then
 return "onas"
else if segType=["oon","oom"] then
 return "oonas"
else if segType=["oohn","oohm"] then
 return "oohnas"
```

```
else if segType=["orn","orm"] then
 return "ornas"
else if segType=["ohn","ohm"] then
 return "ohnas"
else if segType=["oyn","oym"] then
 return "oynas"
else if segType=["un","ung","um"] then
 return "unas"
else if segType=["urn","urm"] then
 return "urnas"
else if segType=["ewn","ewm"] then
 return "ewnas"

 --vowel-liquid combinations
else if segType=["aw","al"] then
 return "aliq"
else if segType=["ahw","ahl"] then
 return "ahliq"
else if segType=["ayw","ayl"] then
 return "ayliq"
else if segType=["ew","el"] then
 return "eliq"
else if segType=["eyew","eyel"] then
 return "eyeliq"
else if segType=["eew","eel"] then
 return "eeliq"
else if segType=["iw","il"] then
 return "iliq"
else if segType=["ow","ol"] then
 return "oliq"
else if segType=["oow","ool"] then
 return "ooliq"
else if segType=["oohw","oohl"] then
 return "oohliq"
else if segType=["orw","orl"] then
 return "orliq"
else if segType=["ohw","ohl"] then
 return "ohliq"
else if segType=["oyw","oyl"] then
 return "oyliq"
else if segType=["uw","ul"] then
 return "uliq"
else if segType=["eww","ewl"] then
 return "ewliq"


 --vowel-fricative combinations
else if segType=["ash","adg","ath","av","af","as","az","ah","ach"] then
 return "afric"
else if segType=["ahsh","ahdg","ahth","ahv","ahf","ahs","ahz","ahh","ahch"] then
```

```
  return "ahfric"
 else if segType=["aysh","aydg","ayth","ayv","ayf","ays","ayz","ayh","aych"] then
  return "ayfric"
 else if segType=["esh","edg","eth","ev","ef","es","ez","eh","ech"] then
  return "efric"
 else if
segType=["eyesh","eyedg","eyeth","eyev","eyef","eyes","eyez","eyeh","eyech"] then
  return "eyefric"
 else if segType=["eesh","eedg","eeth","eev","eef","ees","eez","eeh","eech"] then
  return "eefric"
 else if segType=["ish","idg","ith","iv","if","is","iz","ih","ich"] then
  return "ifric"
 else if segType=["osh","odg","oth","ov","of","os","oz","oh","och"] then
  return "ofric"
 else if segType=["oosh","oodg","ooth","oov","oof","oos","ooz","ooh","ooch"] then
  return "oofric"
 else if
segType=["oohsh","oohdg","oohth","oohv","oohf","oohs","oohz","oohh","oohch"]
then
  return "oohfric"
 else if segType=["orsh","ordg","orth","orv","orf","ors","orz","orh","orch"] then
  return "orfric"
 else if segType=["ohsh","ohdg","ohth","ohv","ohf","ohs","ohz","ohh","ohch"] then
  return "ohfric"
 else if segType=["oysh","oydg","oyth","oyv","oyf","oys","oyz","oyh","oych"] then
  return "oyfric"
 else if segType=["ush","udg","uth","uv","uf","us","uz","uh","uch"] then
  return "ufric"
 else if segType=["ewsh","ewdg","ewth","ewv","ewf","ews","ewz","ewh","ewch"]
then
  return "ewfric"
 end if
end
--end of section


--set up the Timing Database
on setListUp
 global temp, times    --temp contains the list of segmented words
 set temp=[]           --times contains the corresponding durations
 set times=[]

--NB plos=plosive    nas=nasal    fric=fricative    stp=stop liq=liquid

 append temp,["liqu","unas"]
 append times,[188.005, 348.186]
 append temp,["plosr","ray","ayt"]
 append times,[46.168, 123.129, 346.485]
 append temp,["friceye","eyen","nd"]
 append times,[169.456, 345.737, 266.190]
```

```
append temp,["fricur","urs","st"]
append times,[465.510, 0, 260.136]
append temp,["frica","at"]
append times,[128.526, 295.986]
append temp,["nasa","an"]
append times,[155.918, 330.998]
append temp,["nase","en","nee"]
append times,[164.263, 128.277, 380.000]
append temp,["nasu","us","st"]
append times,[168.889,   209.887,   380.816]
append temp,["plosa","an"]
append times,[90.385, 330.998]
append temp,["liqo","ot"]
append times,[195.193,484.036]
append temp,["liqoo","ood"]
append times,[234.331, 319.977]
append temp,["plosoo","ood"]
append times,[109.637, 319.977]
append temp,["liqu","un"]
append times,[188.005, 348.186]
append temp,["liqeye","eyec"]
append times,[131.179, 437.029]
append temp,["liqi","ich"]
append times,[170.113, 374.989]
append temp,["plosu","ut"]
append times,[53.968, 387.330]
append temp,["plosi","iv"]
append times,[42.268, 389.977]
append temp,["plosay"]
append times,[273.197]
append temp,["liqi","il"]
append times,[224.558, 249.977]
append temp,["liqe","en"]
append times,[154.739, 360.000]
append temp,["nasay","ayc"]
append times,[143.039, 330.000]
append temp,["nasor"]
append times,[439.977]
append temp,["frice","e-r"]
append times,[127.007, 294.082]
append temp,["ploseye","eyem"]
append times,[86.848, 548.776]
append temp,["plosoh"]
append times,[339.977]
append temp,["nasoh"]
append times,[450.000]
append temp,["ustp"]
append times,[423.15]
append temp,["fricro","om"]
append times,[174.286, 92.426, 400.000]
```

```
append temp,["frica","av"]
append times,[68.662, 380.00]
append temp,["frici","ing"]
append times,[123.016, 339.569]
append temp,["frice","e-r","ree"]
append times,[150.408, 166.100, 304.036]
append temp,["plosi","ic","co","oz"]
append times,[37.370, 120.884, 88.526, 419.977]
append temp,["nasah","ahc"]
append times,[136.19, 474.989]
append temp,["frica","an","nd"]
append times,[76.054, 169.229, 266.190]
append temp,["plosow","own"]
append times,[42.268, 470.431]
append temp,["nase","ev","vu"]
append times,[289.864, 0, 320.181]
append temp,["liqah","ahs","st"]
append times,[122.245,   268.299,   273.537 ]
append temp,["ustp","bow","owt"]
append times,[230.340, 0, 480]
append temp,["fricoh"]
append times,[550.454]
append temp,["liqu","un","nt"]
append times,[326.168, 0, 500.000]
append temp,["plosl","lay","ays"]
append times,[47.823, 111.519, 363.741]
append temp,["nasu","um","mb","bu"]
append times,[142.540, 299.546, 0, 292.494]
append temp,["fricee","eel"]
append times,[144.195, 279.977]
append temp,["liqi","ith","thow","owt"]
append times,[166.327,  245.850,  0, 480.0]
append temp,["plosi","ig","gi","in"]
append times,[122.902, 0, 60.000, 287.279]
--entries containing 'wild-card' segments
append temp,["*e","e-r","ree"]
append times,[150.408, 166.100, 304.036]
append temp,["*eye","eyem"]
append times,[86.848, 548.776]
append temp,["*i","ith"]
append times,[177.687, 329.977]
append temp,["*a","av"]
append times,[68.662, 380.00]
append temp,["*o","ot"]
append times,[141.791, 484.036]
append temp,["*a","at"]
append times,[128.526, 295.986]
append temp,["*a","a*"]
append times,[128.526, 295.986]
append temp,["*i","is"]
```

```
append times,[158.730, 402.154]
append temp,["*ro","om"]
append times,[174.286, 92.426, 400.000]
append temp,["*u","ut"]
append times,[53.968, 387.330]
append temp,["*i","ich"]
append times,[170.113, 374.989]
append temp,["*oo","ood"]
append times,[234.331, 319.977]
append temp,["*u","un"]
append times,[188.005, 348.186]
append temp,["*i","il"]
append times,[224.558, 249.977]
append temp,["*e","en"]
append times,[154.739, 360.000]
append temp,["*ay","ayc"]
append times,[143.039, 330.000]
append temp,["*e","e-r"]
append times,[127.007, 294.082]
append temp,["*a","an"]
append times,[90.385, 330.998]
append temp,["*o","ot"]
append times,[195.193,484.036]
append temp,["*i","ic","co","oz"]
append times,[37.370, 120.884, 88.526, 419.977]
append temp,["*ow","own"]
append times,[42.268, 470.431]
append temp,["*e","ev","vu"]
append times,[289.864, 0, 320.181]
append temp,["*ah","ahs","st"]
append times,[122.245,   268.299,   273.537 ]
append temp,["*b","bow","owt"]
append times,[230.340, 0, 480]
append temp,["*oh"]
append times,[550.454]
append temp,["*u","un","nt"]
append times,[326.168, 0, 500.000]
append temp,["*l","lay","ays"]
append times,[47.823, 111.519, 363.741]
append temp,["*u","um","mb","bu"]
append times,[142.540, 299.546, 0, 292.494]
append temp,["*ee","eel"]
append times,[144.195, 279.977]
append temp,["*i","ith","thow","owt"]
append times,[166.327,  245.850,  0, 480.0]
append temp,["*i","ig","gi","in"]
append times,[122.902, 0, 60.000, 287.279]
append temp,["*ah","ahc"]
append times,[136.19, 474.989]
```

```
--'"pure" entries containing no re-categorised segments
 append temp,["in"]
 append times,[282.063]
 append temp,["hee"]
 append times,[290.000]
 append temp,["tooh"]
 append times,[352.449]
 append temp,["ha","av"]
 append times,[68.662, 380.00]
 append temp,["it"]
 append times,[350.000]
 append temp,["eye"]
 append times,[368.322]
 append temp,["thay"]
 append times,[520.227]
 append temp,["wi","ith"]
 append times,[177.687, 329.977]
 append temp,["no","ot"]
 append times,[141.791, 484.036]
 append temp,["tha","at"]
 append times,[128.526, 295.986]
 append temp,["on"]
 append times,[379.977]
 append temp,["shee"]
 append times,[420.658]
 append temp,["az"]
 append times,[388.367]
 append temp,["at"]
 append times,[295.986]
 append temp,["beye"]
 append times,[380.000]
 append temp,["thi","is"]
 append times,[158.730, 402.154]
 append temp,["wee"]
 append times,[435.193]
 append temp,["yooh"]
 append times,[498.617]
 append temp,["fro","om"]
 append times,[174.286, 92.426, 400.000]
 append temp,["dooh"]
 append times,[330.000]
 append temp,["bu","ut"]
 append times,[53.968, 387.330]
 append temp,["or"]
 append times,[359.048]
 append temp,["an"]
 append times,[330.998]
 append temp,["wi","ich"]
 append times,[170.113, 374.989]
 append temp,["woo","ood"]
```

```
append times,[234.331, 319.977]
append temp,["say"]
append times,[430.227]
append temp,["orl"]
append times,[404.626]
append temp,["wu","un"]
append times,[188.005, 348.186]
append temp,["wi","il"]
append times,[224.558, 249.977]
append temp,["hooh"]
append times,[395.351]
append temp,["tha","at"]
append times,[161.497, 295.986]
append temp,["we","en"]
append times,[154.739, 360.000]
append temp,["may","ayc"]
append times,[143.039, 330.000]
append temp,["the","e-r"]
append times,[127.007, 294.082]
append temp,["if"]
append times,[351.882]
append temp,["ca","an"]
append times,[90.385, 330.998]
append temp,["ma","an"]
append times,[155.918, 330.998]
append temp,["wo","ot"]
append times,[195.193,484.036]
append temp,["teye","eyem"]
append times,[86.848, 548.776]
append temp,["goh"]
append times,[339.977]
append temp,["noh"]
append times,[450.000]
append temp,["in","nt","tooh"]
append times,[281.587, 0, 304.286]
append temp,["coo","ood"]
append times,[109.637, 319.977]
append temp,["up"]
append times,[423.152]
append temp,["uth","thu"]
append times,[166.440, 451.474]
append temp,["tha","at"]
append times,[137.868, 295.986]
append temp,["yee","eeu"]
append times,[146.553, 410.000]
append temp,["owt"]
append times,[480.000]
append temp,["new"]
append times,[400.000]
append temp,["su","um"]
```

```
append times,[156.372, 282.902]
append temp,["tay","ayc"]
append times,[63.311, 330.000]
append temp,["thee","eez"]
append times,[145.306, 380.000]
append temp,["cu","um"]
append times,[103.220, 322.585]
append temp,["see"]
append times,[439.977]
append temp,["ge","et"]
append times,[56.780, 340.000]
append temp,["noh"]
append times,[450]
append temp,["st","tay","ayt"]
append times,[264.263, 0, 346.485]
append temp,["tooh"]
append times,[352.449]
append temp,["ohn","nl","lee"]
append times,[343.810,0,270.839]
append temp,["the","en"]
append times,[173.129, 360.000]
append temp,["en","nee"]
append times,[83.152, 380.000]
append temp,["now"]
append times,[430.000]
append temp,["may"]
append times,[  367.982]
append temp,["tha","an"]
append times,[168.073, 330.998]
append temp,["gi","iv"]
append times,[42.268, 389.977]
append temp,["ub","bow","owt"]
append times,[168.299, 49.592, 480.000]
append temp,["az"]
append times,[388.367]
append temp,["day"]
append times,[273.197]
append temp,["orl","ls","soh"]
append times,[268.980, 0, 563.968]
append temp,["fur","urs","st"]
append times,[465.510, 0, 260.136]
append temp,["feye","eyen","nd"]
append times,[169.456, 345.737, 266.190]
append temp,["fur","urs","st"]
append times,[465.510, 0, 260.136]
append temp,["way"]
append times,[475.125]
append temp,["mu","us","st"]
append times,[168.889, 209.887, 380.816]
append temp,["yooh","oohz"]
```

```
append times,[197.755, 469.977]
append temp,["mor"]
append times,[439.977]
append temp,["leye","eyec"]
append times,[131.179, 437.029]
append temp,["eev","vu","un"]
append times,[233.878,145.941, 348.186]
append temp,["me","en","nee"]
append times,[164.263, 128.277, 380.000]
append temp,["eech"]
append times,[390]
append temp,["thoh","ohz"]
append times, [224.649, 400]
append temp,["mi","is","st","tu"]
append times,[164.444, 349.297, 0, 281.043]
append temp,["ohv","vu"]
append times,[275.646, 294.875]
append temp,["wur","url","ld"]
append times,[195.556,   263.220,   283.696]
append temp,["see","eem"]
append times,[187.143, 379.977]
append temp,["dgu","us","st"]
append times,[121.224, 227.778,  380.816]
append temp,["leye","eyef"]
append times,[123.424, 479.524]
append temp,["bee","eec","cu","um"]
append times,[163.946, 0, 100.726, 330.023]
append temp,["te","el"]
append times,[117.778, 253.243]
append temp,["hee","eeu"]
append times,[73.129, 410.000]
append temp,["dow","own"]
append times,[49.025, 474.739]
append temp,["ba","ac"]
append times,[40,300]
append temp,["boh","ohth"]
append times,[35.125, 439.977]
append temp,["goo","ood"]
append times,[45.329, 319.977]
append temp,["bi","it","tw","wee","een"]
append times,[45.261, 199.161, 90.476, 102.358, 342.086]
append temp,["st","ti","il"]
append times,[231.701, 95.828, 249.977]
append temp,["ha","an","nd"]
append times,[76.054, 169.229, 266.190]
append temp,["bi","ic","co","oz"]
append times,[37.370, 120.884, 88.526, 419.977]
append temp,["ve","e-r","ree"]
append times,[150.408, 166.100, 304.036]
append temp,["thi","ing"]
```

```
append times,[123.016, 339.569]
append temp,["ahf","ft","tu"]
append times,[280.635, 225.215, 296.508]
append temp,["dow","own"]
append times,[42.268, 470.431]
append temp,["ne","ev","vu"]
append times,[289.864, 0, 320.181]
append temp,["un","nu","uth","thu"]
append times,[189.796, 260.816, 0, 451.474]
append temp,["un","nd","du"]
append times,[331.020, 0, 300]
append temp,["sc","cooh","oohl"]
append times,[192.676,  0, 320]
append temp,["cay","aym"]
append times,[114.082, 427.483]
append temp,["wur","urc"]
append times,[188.027, 399.977]
append temp,["sl","loh"]
append times,[133.832, 410]
append temp,["ohl","ld"]
append times,[165.669, 283.696]
append temp,["gr","ray","ayt"]
append times,[46.168, 123.129, 346.485]
append temp,["ha","av"]
append times,[75.692, 373.039]
append temp,["moh","ohs","st"]
append times,[334.603, 0, 380.816]
append temp,["weye","eyel"]
append times,[134.354, 511.111]
append temp,["pah","aht"]
append times,[83.832, 358.844]
append temp,["thr","ree"]
append times,[135.737, 419.977]
append temp,["ta","ad"]
append times,[90.385,290091]
append temp,["few"]
append times,[479.977]
append temp,["cheye","eyel","ld"]
append times,[125.215, 283.628, 283.696]
append temp,["wu","un"]
append times,[163.107, 348.186]
append temp,["bi","ig","gi","in"]
append times,[122.902, 0, 60.000, 287.279]
append temp,["wi","ith","thow","owt"]
append times,[166.327,  245.850,  0, 480.0]
append temp,["fee","eel"]
append times,[144.195, 279.977]
append temp,["nu","um","mb","bu"]
append times,[142.540, 299.546, 0, 292.494]
append temp,["pl","lay","ays"]
```

```
  append times,[47.823, 111.519, 363.741]
  append temp,["wu","un","nt"]
  append times,[326.168, 0, 500.000]
  append temp,["soh"]
  append times,[550.454]
  append temp,["ub","bow","owt"]
  append times,[230.340, 0, 480]
  append temp,["lah","ahs","st"]
  append times,[122.245, 268.299, 273.537]
  append temp,["dee","ori","ing"]
  append times,[81.678, 313.537, 339.569]
  append temp,["ahs","sc"]
  append times,[236.621, 321.882]
end
--finish setting up Timing Database

--set up various speech segment categories
on setTypesUp
  --plos=plosive    nas=nasal   fric=fricative   stp=stop
  global plos, liq, nas, fric, stp, vowels, liqMenu, fricMenu, plosMenu, nasMenu

  --plosive-vowel combinations
  set plosa=["ca","ba","pa","ta","da","ga"]
  set plosah=["cah","bah","pah","tah","dah","gah"]
  set plosay=["cay","bay","pay","tay","day","gay"]
  set plose=["ce","be","pe","te","de","ge"]
  set ploseye=["ceye","beye","peye","teye","deye","geye"]
  set plosee=["cee","bee","pee","tee","dee","gee"]
  set plosi=["ci","bi","pi","ti","di","gi"]
  set ploso=["co","bo","po","to","do","go"]
  set plosoo=["coo","boo","poo","too","doo","goo"]
  set plosor=["cor","bor","por","tor","dor","gor"]
  set plosoh=["coh","boh","poh","toh","doh","goh"]
  set plosoy=["coy","boy","poy","toy","doy","goy"]
  set plosooh=["cooh","booh","pooh","tooh","dooh","gooh"]
  set plosu=["cu","bu","pu","tu","du","gu"]
  set plosew=["cew","bew","pew","tew","dew","gew"]
  --nasal-vowel combinations
  set nasa=["na","ma"]
  set nasah=["nah","mah"]
  set nasay=["nay","may"]
  set nase=["ne","me"]
  set nasee=["nee","mee"]
  set naseye=["neye","meye"]
  set nasi=["ni","mi"]
  set naso=["no","mo"]
  set nasoo=["noo","moo"]
  set nasooh=["nooh","mooh"]
  set nasor=["nor","mor"]
  set nasoh=["noh","moh"]
```

```
set nasoy=["noy","moy"]
set nasu=["nu","mu"]
set nasur=["nur","mur"]
set nasew=["new","mew"]
--liquid-vowel combinations
set liqa=["wa","la"]
set liqah=["wah","lah"]
set liqay=["way","lay"]
set liqe=["we","le"]
set liqeye=["weye","leye","reye"]
set liqee=["wee","lee"]
set liqi=["wi","li"]
set liqo=["wo","lo"]
set liqoo=["woo","loo"]
set liqooh=["wooh","looh"]
set liqor=["wor","lor"]
set liqoh=["woh","loh"]
set liqoy=["woy","loy"]
set liqu=["wu","lu"]
set liqew=["wew","lew"]
--fricative-vowel combinations
set frica=["sha","dga","tha","va","fa","sa","za","ha","cha"]
set fricah=["shah","dgah","thah","vah","fah","sah","zah","hah","chah"]
set fricay=["shay","dgay","thay","vay","fay","say","zay","hay","chay"]
set frice=["she","dge","the","ve","fe","se","ze","he","che"]
set friceye=["sheye","dgeye","theye","veye","feye","seye","zeye","heye","cheye"]
set fricee=["shee","dgee","thee","vee","fee","see","zee","hee","chee"]
set frici=["shi","dgi","thi","vi","fi","si","zi","hi","chi"]
set frico=["sho","dgo","tho","vo","fo","so","zo","ho","cho"]
set fricoo=["shoo","dgoo","thoo","voo","foo","soo","zoo","hoo","choo"]
set fricooh=["shooh","dgooh","thooh","vooh","fooh","sooh","zooh","hooh","chooh"]
set fricor=["shor","dgor","thor","vor","for","sor","zor","hor","chor"]
set fricoh=["shoh","dgoh","thoh","voh","foh","soh","zoh","hoh","choh"]
set fricoy=["shoy","dgoy","thoy","voy","foy","soy","zoy","hoy","choy"]
set fricu=["shu","dgu","thu","vu","fu","su","zu","hu","chu"]
set fricew=["shew","dgew","thew","vew","few","sew","zew","hew","chew"]

--consonant clusters (under construction)
set plosl=["cl","bl","pl","tl","dl","gl"]
set plosn=["cn","bn","pn","tn","dn","gn"]
set plosr=["cr","br","pr","tr","dr","gr"]
set ploss=["cs","bs","ps","ts","ds","gs"]
set fricn=["shn","dgn","thn","vn","fn","sn","zn","chn"]

--vowel-stop combinations
set astp=["ac","ab","ap","at","ad","ag"]
set ahstp= ["ahc","ahb","ahp","aht","ahd","ahg"]
set aystp=["ayc","ayb","ayp","ayt","ayd","ayg"]
set estp=["ec","eb","ep","et","ed","eg"]
set eyestp=["eyec","eyeb","eyep","eyet","eyed","eyeg"]
```

```
set eestp=["eec","eeb","eep","eet","eed","eeg"]
set istp=["ic","ib","ip","it","id","ig"]
set ostp=["oc","ob","op","ot","od","og"]
set oostp=["ooc","oob","oop","oot","ood","oog"]
set orstp=["orc","orb","orp","ort","ord","org"]
set ohstp=["ohc","ohb","ohp","oht","ohd","ohg"]
set oystp=["oyc","oyb","oyp","oyt","oyd","oyg"]
set oohstp=["oohc","oohb","oohp","ooht","oohd","oohg"]
set ustp=["uc","ub","up","ut","ud","ug"]
set ewstp=["ewc","ewb","ewp","ewt","ewd","ewg"]

--vowel nasals
set anas=["an","ang","am"]
set ahnas=["ahn","ahm"]
set aynas=["ayn","aym"]
set enas=["en","eng","em"]
set eenas=["een","eem"]
set eyenas=["eyen","eyem"]
set inas=["in","ing","im"]
set onas=["on","ong","om"]
set oonas=["oon","oom"]
set oohnas=["oohn","oohm"]
set ornas=["orn","orm"]
set ohnas=["ohn","ohm"]
set oynas=["oyn","oym"]
set unas=["un","ung","um"]
set urnas=["urn","urm"]
set ewnas=["ewn","ewm"]

--vowel liquids
set aliq=["aw","al"]
set ahliq=["ahw","ahl"]
set ayliq=["ayw","ayl"]
set eliq=["ew","el"]
set eyeliq=["eyew","eyel"]
set eeliq=["eew","eel"]
set iliq=["iw","il"]
set oliq=["ow","ol"]
set ooliq=["oow","ool"]
set oohliq=["oohw","oohl"]
set orliq=["orw","orl"]
set ohliq=["ohw","ohl"]
set oyliq=["oyw","oyl"]
set uliq=["uw","ul"]
set ewliq=["eww","ewl"]

--vowel fricatives
set afric=["ash","adg","ath","av","af","as","az","ah","ach"]
set ahfric=["ahsh","ahdg","ahth","ahv","ahf","ahs","ahz","ahh","ahch"]
set ayfric=["aysh","aydg","ayth","ayv","ayf","ays","ayz","ayh","aych"]
```

```
set efric=["esh","edg","eth","ev","ef","es","ez","eh","ech"]
set eyefric=["eyesh","eyedg","eyeth","eyev","eyef","eyes","eyez","eyeh","eyech"]
set eefric=["eesh","eedg","eeth","eev","eef","ees","eez","eeh","eech"]
set ifric=["ish","idg","ith","iv","if","is","iz","ih","ich"]
set ofric=["osh","odg","oth","ov","of","os","oz","oh","och"]
set oofric=["oosh","oodg","ooth","oov","oof","oos","ooz","ooh","ooch"]
set oohfric=["oohsh","oohdg","oohth","oohv","oohf","oohs","oohz","oohh","oohch"]
set orfric=["orsh","ordg","orth","orv","orf","ors","orz","orh","orch"]
set ohfric=["ohsh","ohdg","ohth","ohv","ohf","ohs","ohz","ohh","ohch"]
set oyfric=["oysh","oydg","oyth","oyv","oyf","oys","oyz","oyh","oych"]
set ufric=["ush","udg","uth","uv","uf","us","uz","uh","uch"]
set ewfric=["ewsh","ewdg","ewth","ewv","ewf","ews","ewz","ewh","ewch"]


set
plosMenu=[plosa,plosah,plosay,plose,ploseye,plosee,plosi,ploso,plosoo,plosor,plosoh
,plosoy,plosu,plosew,plosooh,plosl,plosn,plosr,ploss,astp,ahstp,aystp,estp,eyestp,eestp
,istp,ostp,oostp,orstp,ohstp,oystp,ustp,ewstp,oohstp]

set
liqMenu=[liqa,liqah,liqay,liqe,liqeye,liqee,liqi,liqo,liqoo,liqooh,liqor,liqoh,liqoy,liqu,
liqew,aliq,ahliq,ayliq,eliq,eyeliq,eeliq,iliq,oliq,ooliq,oohliq,orliq,ohliq,oyliq,uliq,
ewliq]

set
fricMenu=[frica,fricah,fricay,frice,friceye,fricee,frici,fricn,frico,fricoo,fricooh,fricor,
fricoh,fricu,fricew,afric,ahfric,ayfric,efric,eyefric,eefric,ifric,ofric,oofric,oohfric,orfric
,ohfric,ufric,ewfric]

set
nasMenu=[nasa,nasah,nasay,nase,naseye,nasee,nasi,naso,nasoo,nasooh,nasor,nasoh,
nasoy,nasu,nasur,nasew,anas,ahnas,aynas,enas,eyenas,eenas,inas,onas,oonas,oohnas,
ornas,ohnas,oynas,unas,urnas,ewnas]
end
```
--*end of section*

14. Appendix G - Letter-To-Sound Rules

This section contains the Letter-to-Sound rules used in the Text-to-phoneme conversion program. NB. These rules are for American English.

```
/* ************ English.c *************** */

/*
**      English to Phoneme rules.
**
**      Derived from:
**
**          AUTOMATIC TRANSLATION OF ENGLISH TEXT TO PHONETICS
**              BY MEANS OF LETTER-TO-SOUND RULES
**
**                      NRL Report 7948
**
**                  January 21st, 1976
**          Naval Research Laboratory, Washington, D.C.
**
**
**      Published by the National Technical Information Service as
**      document "AD/A021 929".
**
**
**
**      The Phoneme codes:
**
**              IY      bEEt            IH      bIt
**              EY      gAte            EH      gEt
**              AE      fAt             AA      fAther
**              AO      lAWn            OW      lOne
**              UH      fUll            UW      fOOl
**              ER      mURdER          AX      About
**              AH      bUt             AY      hIde
**              AW      hOW             OY      tOY
**
**              p       Pack            b       Back
**              t       Time            d       Dime
**              k       Coat            g       Goat
**              f       Fault           v       Vault
**              TH      eTHer           DH      eiTHer
**              s       Sue             z       Zoo
**              SH      leaSH           ZH      leiSure
**              HH      How             m       suM
**              n       suN             NG      suNG
**              l       Laugh           w       Wear
**              y       Young           r       Rate
**              CH      CHar            j       Jar
```

```
**              WH    WHere
**
**
**      Rules are made up of four parts:
**
**              The left context.
**              The text to match.
**              The right context.
**              The phonemes to substitute for the matched text.
**
**      Procedure:
**
**              Seperate each block of letters (apostrophes included)
**              and add a space on each side.  For each unmatched
**              letter in the word, look through the rules where the
**              text to match starts with the letter in the word.  If
**              the text to match is found and the right and left
**              context patterns also match, output the phonemes for
**              that rule and skip to the next unmatched letter.
**
**
**      Special Context Symbols:
**
**              #       One or more vowels
**              :       Zero or more consonants
**              ^       One consonant.
**              .       One of B, D, V, G, J, L, M, N, R, W or Z (voiced
**                      consonants)
**              %       One of ER, E, ES, ED, ING, ELY (a suffix)
**                      (Found in right context only)
**              +       One of E, I or Y (a "front" vowel)
**
*/


/* Context definitions */
static char Anything[] = "";    /* No context requirement */
static char Nothing[] = " ";    /* Context is beginning or end of word */

/* Phoneme definitions */
static char Pause[] = " ";      /* Short silence */
static char Silent[] = "";      /* No phonemes */

#define LEFT_PART 0
#define MATCH_PART      1
#define RIGHT_PART      2
#define OUT_PART  3

typedef char *Rule[4];          /* Rule is an array of 4 character pointers */
```

```
/*0 = Punctuation */
/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule punct_rules[] =
        {
        {Anything,    " ",          Anything,     Pause  },
        {Anything,    "-",          Anything,     Silent  },
        {".",         "'S",         Anything,     "z"    },
        {"#:.E",      "'S",         Anything,     "z"    },
        {"#",         "'S",         Anything,     "z"    },
        {Anything,    "'",          Anything,     Silent  },
        {Anything,    ",",          Anything,     Pause  },
        {Anything,    ".",          Anything,     Pause  },
        {Anything,    "?",          Anything,     Pause  },
        {Anything,    "!",          Anything,     Pause  },
        {Anything,    0,            Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule A_rules[] =
        {
        {Anything,    "A",          Nothing,      "AX"   },
        {Nothing,     "ARE",        Nothing,      "AAr"  },
        {Nothing,     "AR",         "O",          "AXr"  },
        {Anything,    "AR",         "#",          "EHr"  },
        {"^",         "AS",         "#",          "EYs"  },
        {Anything,    "A",          "WA",         "AX"   },
        {Anything,    "AW",         Anything,     "AO"   },
        {" :",        "ANY",        Anything,     "EHnIY"},
        {Anything,    "A",          "^+#",        "EY"   },
        {"#:",        "ALLY",       Anything,     "AXlIY"},
        {Nothing,     "AL",         "#",          "AXl"  },
        {Anything,    "AGAIN",      Anything,     "AXgEHn"},
        {"#:",        "AG",         "E",          "IHj"  },
        {Anything,    "A",          "^+:#",       "AE"   },
        {" :",        "A",          "^+ ",        "EY"   },
        {Anything,    "A",          "^%",         "EY"   },
        {Nothing,     "ARR",        Anything,     "AXr"  },
        {Anything,    "ARR",        Anything,     "AEr"  },
        {" :",        "AR",         Nothing,      "AAr"  },
        {Anything,    "AR",         Nothing,      "ER"   },
        {Anything,    "AR",         Anything,     "AAr"  },
        {Anything,    "AIR",        Anything,     "EHr"  },
        {Anything,    "AI",         Anything,     "EY"   },
        {Anything,    "AY",         Anything,     "EY"   },
        {Anything,    "AU",         Anything,     "AO"   },
```

```
{"#:",          "AL",          Nothing,       "AXl"  },
{"#:",          "ALS",         Nothing,       "AXlz"},
{Anything,      "ALK",         Anything,      "AOk" },
{Anything,      "AL",          "^",           "AOl"  },
{" :",          "ABLE",        Anything,      "EYbAXl"},
{Anything,      "ABLE",        Anything,      "AXbAXl"},
{Anything,      "ANG",         "+",           "EYnj"},
{Anything,      "A",           Anything,      "AE"   },
{Anything,      0,             Anything,      Silent  },
};


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule B_rules[] =
        {
        {Nothing,       "BE",          "^#",          "bIH"  },
        {Anything,      "BEING",       Anything,      "bIYIHNG"},
        {Nothing,       "BOTH",        Nothing,       "bOWTH"},
        {Nothing,       "BUS",         "#",           "bIHz" },
        {Anything,      "BUIL",        Anything,      "bIHl" },
        {Anything,      "B",           Anything,      "b"    },
        {Anything,      0,             Anything,      Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule C_rules[] =
        {
        {Nothing,       "CH",          "^",           "k"    },
        {"^E",          "CH",          Anything,      "k"    },
        {Anything,      "CH",          Anything,      "CH"   },
        {" S",          "CI",          "#",           "sAY"  },
        {Anything,      "CI",          "A",           "SH"   },
        {Anything,      "CI",          "O",           "SH"   },
        {Anything,      "CI",          "EN",          "SH"   },
        {Anything,      "C",           "+",           "s"    },
        {Anything,      "CK",          Anything,      "k"    },
        {Anything,      "COM",         "%",           "kAHm"},
        {Anything,      "C",           Anything,      "k"    },
        {Anything,      0,             Anything,      Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
```

```
static Rule D_rules[] =
        {
        {"#:",          "DED",      Nothing,     "dIHd" },
        {".E",          "D",        Nothing,     "d"    },
        {"#:^E",        "D",        Nothing,     "t"    },
        {Nothing,       "DE",       "^#",        "dIH"  },
        {Nothing,       "DO",       Nothing,     "dUW"},
        {Nothing,       "DOES",     Anything,    "dAHz"},
        {Nothing,       "DOING",    Anything,    "dUWIHNG"},
        {Nothing,       "DOW",      Anything,    "dAW"},
        {Anything,      "DU",       "A",         "jUW" },
        {Anything,      "D",        Anything,    "d"    },
        {Anything,      0,          Anything,    Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule E_rules[] =
        {
        {"#:",          "E",        Nothing,     Silent  },
        {":^",          "E",        Nothing,     Silent  },
        {" :",          "E",        Nothing,     "IY"    },
        {"#",           "ED",       Nothing,     "d"     },
        {"#:",          "E",        "D ",        Silent  },
        {Anything,      "EV",       "ER",        "EHv"  },
        {Anything,      "E",        "^%",        "IY"    },
        {Anything,      "ERI",      "#",         "IYrIY"},
        {Anything,      "ERI",      Anything,    "EHrIH"},
        {"#:",          "ER",       "#",         "ER"   },
        {Anything,      "ER",       "#",         "EHr"  },
        {Anything,      "ER",       Anything,    "ER"   },
        {Nothing,       "EVEN",     Anything,    "IYvEHn"},
        {"#:",          "E",        "W",         Silent  },
        {"T",           "EW",       Anything,    "UW"  },
        {"S",           "EW",       Anything,    "UW"  },
        {"R",           "EW",       Anything,    "UW"  },
        {"D",           "EW",       Anything,    "UW"  },
        {"L",           "EW",       Anything,    "UW"  },
        {"Z",           "EW",       Anything,    "UW"  },
        {"N",           "EW",       Anything,    "UW"  },
        {"J",           "EW",       Anything,    "UW"  },
        {"TH",          "EW",       Anything,    "UW"  },
        {"CH",          "EW",       Anything,    "UW"  },
        {"SH",          "EW",       Anything,    "UW"  },
        {Anything,      "EW",       Anything,    "yUW"},
        {Anything,      "E",        "O",         "IY"   },
        {"#:S",         "ES",       Nothing,     "IHz"  },
        {"#:C",         "ES",       Nothing,     "IHz"  },
```

```
{"#:G",        "ES",        Nothing,      "IHz"   },
{"#:Z",        "ES",        Nothing,      "IHz"   },
{"#:X",        "ES",        Nothing,      "IHz"   },
{"#:J",        "ES",        Nothing,      "IHz"   },
{"#:CH",       "ES",        Nothing,      "IHz"   },
{"#:SH",       "ES",        Nothing,      "IHz"   },
{"#:",         "E",         "S ",         Silent  },
{"#:",         "ELY",       Nothing,      "lIY"   },
{"#:",         "EMENT",     Anything,     "mEHnt"},
{Anything,     "EFUL",      Anything,     "fUHl" },
{Anything,     "EE",        Anything,     "IY"    },
{Anything,     "EARN",      Anything,     "ERn"   },
{Nothing,      "EAR",       "^",          "ER"    },
{Anything,     "EAD",       Anything,     "EHd"   },
{"#:",         "EA",        Nothing,      "IYAX"},
{Anything,     "EA",        "SU",         "EH"    },
{Anything,     "EA",        Anything,     "IY"    },
{Anything,     "EIGH",      Anything,     "EY"    },
{Anything,     "EI",        Anything,     "IY"    },
{Nothing,      "EYE",       Anything,     "AY"    },
{Anything,     "EY",        Anything,     "IY"    },
{Anything,     "EU",        Anything,     "yUW"},
{Anything,     "E",         Anything,     "EH"    },
{Anything,     0,           Anything,     Silent  },
};


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule F_rules[] =
        {
        {Anything,     "FUL",       Anything,     "fUHl" },
        {Anything,     "F",         Anything,     "f"     },
        {Anything,     0,           Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule G_rules[] =
        {
        {Anything,     "GIV",       Anything,     "gIHv" },
        {Nothing,      "G",         "I^",         "g"     },
        {Anything,     "GE",        "T",          "gEH"  },
        {"SU",         "GGES",      Anything,     "gjEHs"},
        {Anything,     "GG",        Anything,     "g"     },
        {" B#",        "G",         Anything,     "g"     },
        {Anything,     "G",         "+",          "j"     },
```

```
        {Anything,    "GREAT",     Anything,     "grEYt"},
        {"#",         "GH",        Anything,     Silent  },
        {Anything,    "G",         Anything,     "g"     },
        {Anything,    0,           Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule H_rules[] =
        {
        {Nothing,     "HAV",       Anything,     "hAEv"},
        {Nothing,     "HERE",      Anything,     "hIYr" },
        {Nothing,     "HOUR",      Anything,     "AWER"},
        {Anything,    "HOW",       Anything,     "hAW"},
        {Anything,    "H",         "#",          "h"     },
        {Anything,    "H",         Anything,     Silent  },
        {Anything,    0,           Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule I_rules[] =
        {
        {Nothing,     "IN",        Anything,     "IHn"   },
        {Nothing,     "I",         Nothing,      "AY"    },
        {Anything,    "IN",        "D",          "AYn"   },
        {Anything,    "IER",       Anything,     "IYER"},
        {"#:R",       "IED",       Anything,     "IYd"   },
        {Anything,    "IED",       Nothing,      "AYd"   },
        {Anything,    "IEN",       Anything,     "IYEHn"},
        {Anything,    "IE",        "T",          "AYEH"},
        {" :",        "I",         "%",          "AY"    },
        {Anything,    "I",         "%",          "IY"    },
        {Anything,    "IE",        Anything,     "IY"    },
        {Anything,    "I",         "^+:#",       "IH"    },
        {Anything,    "IR",        "#",          "AYr"   },
        {Anything,    "IZ",        "%",          "AYz"   },
        {Anything,    "IS",        "%",          "AYz"   },
        {Anything,    "I",         "D%",         "AY"    },
        {"+^",        "I",         "^+",         "IH"    },
        {Anything,    "I",         "T%",         "AY"    },
        {"#:^",       "I",         "^+",         "IH"    },
        {Anything,    "I",         "^+",         "AY"    },
        {Anything,    "IR",        Anything,     "ER"    },
        {Anything,    "IGH",       Anything,     "AY"    },
        {Anything,    "ILD",       Anything,     "AYld"},
```

```
        {Anything,      "IGN",          Nothing,        "AYn" },
        {Anything,      "IGN",          "^",            "AYn" },
        {Anything,      "IGN",          "%",            "AYn" },
        {Anything,      "IQUE",         Anything,       "IYk"  },
        {Anything,      "I",            Anything,       "IH"   },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule J_rules[] =
        {
        {Anything,      "J",            Anything,       "j"     },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule K_rules[] =
        {
        {Nothing,       "K",            "N",            Silent  },
        {Anything,      "K",            Anything,       "k"     },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule L_rules[] =
        {
        {Anything,      "LO",           "C#",           "lOW" },
        {"L",           "L",            Anything,       Silent  },
        {"#:^",         "L",            "%",            "AXl" },
        {Anything,      "LEAD",         Anything,       "lIYd" },
        {Anything,      "L",            Anything,       "l"     },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule M_rules[] =
        {
        {Anything,      "MOV",          Anything,       "mUWv"      },
```

```
        {Anything,    "M",          Anything,     "m"     },
        {Anything,    0,            Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule N_rules[] =
        {
        {"E",         "NG",         "+",          "nj"    },
        {Anything,    "NG",         "R",          "NGg"   },
        {Anything,    "NG",         "#",          "NGg"   },
        {Anything,    "NGL",        "%",          "NGgAXl"},
        {Anything,    "NG",         Anything,     "NG"    },
        {Anything,    "NK",         Anything,     "NGk"   },
        {Nothing,     "NOW",        Nothing,      "nAW"   },
        {Anything,    "N",          Anything,     "n"     },
        {Anything,    0,            Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule O_rules[] =
        {
        {Anything,    "OF",         Nothing,      "AXv"   },
        {Anything,    "OROUGH",     Anything,     "EROW"  },
        {"#:",        "OR",         Nothing,      "ER"    },
        {"#:",        "ORS",        Nothing,      "ERz"   },
        {Anything,    "OR",         Anything,     "AOr"   },
        {Nothing,     "ONE",        Anything,     "wAHn"  },
        {Anything,    "OW",         Anything,     "OW"    },
        {Nothing,     "OVER",       Anything,     "OWvER" },
        {Anything,    "OV",         Anything,     "AHv"   },
        {Anything,    "O",          "^%",         "OW"    },
        {Anything,    "O",          "^EN",        "OW"    },
        {Anything,    "O",          "^I#",        "OW"    },
        {Anything,    "OL",         "D",          "OWl"   },
        {Anything,    "OUGHT",      Anything,     "AOt"   },
        {Anything,    "OUGH",       Anything,     "AHf"   },
        {Nothing,     "OU",         Anything,     "AW"    },
        {"H",         "OU",         "S#",         "AW"    },
        {Anything,    "OUS",        Anything,     "AXs"   },
        {Anything,    "OUR",        Anything,     "AOr"   },
        {Anything,    "OULD",       Anything,     "UHd"   },
        {"^",         "OU",         "^L",         "AH"    },
        {Anything,    "OUP",        Anything,     "UWp"   },
        {Anything,    "OU",         Anything,     "AW"    },
```

```
{Anything,    "OY",        Anything,    "OY"   },
{Anything,    "OING",      Anything,    "OWIHNG"},
{Anything,    "OI",        Anything,    "OY"   },
{Anything,    "OOR",       Anything,    "AOr"  },
{Anything,    "OOK",       Anything,    "UHk"  },
{Anything,    "OOD",       Anything,    "UHd"  },
{Anything,    "OO",        Anything,    "UW"   },
{Anything,    "O",         "E",         "OW"   },
{Anything,    "O",         Nothing,     "OW"   },
{Anything,    "OA",        Anything,    "OW"   },
{Nothing,     "ONLY",      Anything,    "OWnlIY"},
{Nothing,     "ONCE",      Anything,    "wAHns"},
{Anything,    "ONT",       Anything,    "OWnt"},
{"C",         "O",         "N",         "AA"   },
{Anything,    "O",         "NG",        "AO"   },
{" :^",       "O",         "N",         "AH"   },
{"I",         "ON",        Anything,    "AXn"  },
{"#:",        "ON",        Nothing,     "AXn"  },
{"#^",        "ON",        Anything,    "AXn"  },
{Anything,    "O",         "ST ",       "OW"   },
{Anything,    "OF",        "^",         "AOf"  },
{Anything,    "OTHER",     Anything,    "AHDHER"},
{Anything,    "OSS",       Nothing,     "AOs"  },
{"#:^",       "OM",        Anything,    "AHm"},
{Anything,    "O",         Anything,    "AA"   },
{Anything,    0,           Anything,    Silent  },
};


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule P_rules[] =
{
{Anything,    "PH",        Anything,    "f"     },
{Anything,    "PEOP",      Anything,    "pIYp"  },
{Anything,    "POW",       Anything,    "pAW"},
{Anything,    "PUT",       Nothing,     "pUHt"},
{Anything,    "P",         Anything,    "p"     },
{Anything,    0,           Anything,    Silent  },
};


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule Q_rules[] =
{
{Anything,    "QUAR",      Anything,    "kwAOr"},
{Anything,    "QU",        Anything,    "kw"    },
```

```
        {Anything,      "Q",            Anything,       "k"     },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART       RIGHT_PART        OUT_PART
*/
static Rule R_rules[] =
        {
        {Nothing,       "RE",           "^#",           "rIY"   },
        {Anything,      "R",            Anything,       "r"     },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART       RIGHT_PART        OUT_PART
*/
static Rule S_rules[] =
        {
        {Anything,      "SH",           Anything,       "SH"    },
        {"#",           "SION",         Anything,       "ZHAXn"},
        {Anything,      "SOME",         Anything,       "sAHm"},
        {"#",           "SUR",          "#",            "ZHER"},
        {Anything,      "SUR",          "#",            "SHER"},
        {"#",           "SU",           "#",            "ZHUW"},
        {"#",           "SSU",          "#",            "SHUW"},
        {"#",           "SED",          Nothing,        "zd"    },
        {"#",           "S",            "#",            "z"     },
        {Anything,      "SAID",         Anything,       "sEHd"},
        {"^",           "SION",         Anything,       "SHAXn"},
        {Anything,      "S",            "S",            Silent  },
        {".",           "S",            Nothing,        "z"     },
        {"#:.E",        "S",            Nothing,        "z"     },
        {"#:^##",       "S",            Nothing,        "z"     },
        {"#:^#",        "S",            Nothing,        "s"     },
        {"U",           "S",            Nothing,        "s"     },
        {" :#",         "S",            Nothing,        "z"     },
        {Nothing,       "SCH",          Anything,       "sk"    },
        {Anything,      "S",            "C+",           Silent  },
        {"#",           "SM",           Anything,       "zm"    },
        {"#",           "SN",           "'",            "zAXn"},
        {Anything,      "S",            Anything,       "s"     },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART       RIGHT_PART        OUT_PART
```

```
*/
static Rule T_rules[] =
        {
        {Nothing,       "THE",          Nothing,        "DHAX"},
        {Anything,      "TO",           Nothing,        "tUW" },
        {Anything,      "THAT",         Nothing,        "DHAEt"},
        {Nothing,       "THIS",         Nothing,        "DHIHs"},
        {Nothing,       "THEY",         Anything,       "DHEY"},
        {Nothing,       "THERE",        Anything,       "DHEHr"},
        {Anything,      "THER",         Anything,       "DHER"},
        {Anything,      "THEIR",        Anything,       "DHEHr"},
        {Nothing,       "THAN",         Nothing,        "DHAEn"},
        {Nothing,       "THEM",         Nothing,        "DHEHm"},
        {Anything,      "THESE",        Nothing,        "DHIYz"},
        {Nothing,       "THEN",         Anything,       "DHEHn"},
        {Anything,      "THROUGH",Anything,             "THrUW"},
        {Anything,      "THOSE",        Anything,       "DHOWz"},
        {Anything,      "THOUGH",       Nothing,        "DHOW"},
        {Nothing,       "THUS",         Anything,       "DHAHs"},
        {Anything,      "TH",           Anything,       "TH"  },
        {"#:",          "TED",          Nothing,        "tIHd" },
        {"S",           "TI",           "#N",           "CH"  },
        {Anything,      "TI",           "O",            "SH"  },
        {Anything,      "TI",           "A",            "SH"  },
        {Anything,      "TIEN",         Anything,       "SHAXn"},
        {Anything,      "TUR",          "#",            "CHER"},
        {Anything,      "TU",           "A",            "CHUW"},
        {Nothing,       "TWO",          Anything,       "tUW" },
        {Anything,      "T",            Anything,       "t"   },
        {Anything,      0,              Anything,       Silent  },
        };


/*
**      LEFT_PART  MATCH_PART       RIGHT_PART        OUT_PART
*/
static Rule U_rules[] =
        {
        {Nothing,       "UN",           "I",            "yUWn"},
        {Nothing,       "UN",           Anything,       "AHn" },
        {Nothing,       "UPON",         Anything,       "AXpAOn"},
        {"T",           "UR",           "#",            "UHr" },
        {"S",           "UR",           "#",            "UHr" },
        {"R",           "UR",           "#",            "UHr" },
        {"D",           "UR",           "#",            "UHr" },
        {"L",           "UR",           "#",            "UHr" },
        {"Z",           "UR",           "#",            "UHr" },
        {"N",           "UR",           "#",            "UHr" },
        {"J",           "UR",           "#",            "UHr" },
        {"TH",          "UR",           "#",            "UHr" },
```

```
{"CH",      "UR",      "#",       "UHr"  },
{"SH",      "UR",      "#",       "UHr"  },
{Anything,  "UR",      "#",       "yUHr"},
{Anything,  "UR",      Anything,  "ER"   },
{Anything,  "U",       "^ ",      "AH"   },
{Anything,  "U",       "^^",      "AH"   },
{Anything,  "UY",      Anything,  "AY"   },
{" G",      "U",       "#",       Silent  },
{"G",       "U",       "%",       Silent  },
{"G",       "U",       "#",       "w"    },
{"#N",      "U",       Anything,  "yUW"},
{"T",       "U",       Anything,  "UW"  },
{"S",       "U",       Anything,  "UW"  },
{"R",       "U",       Anything,  "UW"  },
{"D",       "U",       Anything,  "UW"  },
{"L",       "U",       Anything,  "UW"  },
{"Z",       "U",       Anything,  "UW"  },
{"N",       "U",       Anything,  "UW"  },
{"J",       "U",       Anything,  "UW"  },
{"TH",      "U",       Anything,  "UW"  },
{"CH",      "U",       Anything,  "UW"  },
{"SH",      "U",       Anything,  "UW"  },
{Anything,  "U",       Anything,  "yUW"},
{Anything,  0,         Anything,  Silent  },
};


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule V_rules[] =
        {
        {Anything,  "VIEW",    Anything,  "vyUW"},
        {Anything,  "V",       Anything,  "v"    },
        {Anything,  0,         Anything,  Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule W_rules[] =
        {
        {Nothing,   "WERE",    Anything,  "wER" },
        {Anything,  "WA",      "S",       "wAA"},
        {Anything,  "WA",      "T",       "wAA"},
        {Anything,  "WHERE",   Anything,  "WHEHr"},
        {Anything,  "WHAT",    Anything,  "WHAAt"},
        {Anything,  "WHOL",    Anything,  "hOWl"},
        {Anything,  "WHO",     Anything,  "hUW"},
```

```
        {Anything,    "WH",         Anything,     "WH"  },
        {Anything,    "WAR",        Anything,     "wAOr"},
        {Anything,    "WOR",        "^",          "wER" },
        {Anything,    "WR",         Anything,     "r"   },
        {Anything,    "W",          Anything,     "w"   },
        {Anything,    0,            Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule X_rules[] =
        {
        {Anything,    "X",          Anything,     "ks"    },
        {Anything,    0,            Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule Y_rules[] =
        {
        {Anything,    "YOUNG",      Anything,     "yAHNG"},
        {Nothing,     "YOU",        Anything,     "yUW"},
        {Nothing,     "YES",        Anything,     "yEHs"},
        {Nothing,     "Y",          Anything,     "y"     },
        {"#:^",       "Y",          Nothing,      "IY"    },
        {"#:^",       "Y",          "I",          "IY"    },
        {" :",        "Y",          Nothing,      "AY"    },
        {" :",        "Y",          "#",          "AY"    },
        {" :",        "Y",          "^+:#",       "IH"    },
        {" :",        "Y",          "^#",         "AY"    },
        {Anything,    "Y",          Anything,     "IH"    },
        {Anything,    0,            Anything,     Silent  },
        };


/*
**      LEFT_PART  MATCH_PART      RIGHT_PART        OUT_PART
*/
static Rule Z_rules[] =
        {
        {Anything,    "Z",          Anything,     "z"     },
        {Anything,    0,            Anything,     Silent  },
        };

Rule *Rules[] =
        {
```

```
punct_rules,
A_rules, B_rules, C_rules, D_rules, E_rules, F_rules, G_rules,
H_rules, I_rules, J_rules, K_rules, L_rules, M_rules, N_rules,
O_rules, P_rules, Q_rules, R_rules, S_rules, T_rules, U_rules,
V_rules, W_rules, X_rules, Y_rules, Z_rules
};
```

15. References

Abercrombie, D. (1964). Syllable quantity and enclitics in English. *In:* Abercrombie, D., Fry, D.B., MacCarthy, P.A.D., Scott, N.C. & Trim, J.L.M. (editors) *In Honour of Daniel Jones*. London, Longmans. 216-222.

Adamson, M.J. & Damper, R.I. (1996). Recurrent Networks for English Text-to-Phoneme Conversion. *In:* Lawrence, R. (editor) *Proc. Institute of Acoustics.* 18(9), 27-34. St. Albans, IoA.

Allen, G.D. (1975). Speech rhythm: its relation to performance universals and articulatory timing. *Journal of Phonetics,* 3, 75-86.

Allen, J. (1976). Speech synthesis from unrestricted text. *Proc. Institute of Electrical Engineers,* 64(4), 433-442.

Allen, J., Hunnicut, S. & Klatt, D. (1987). *From Text to Speech: The MITalk system.* Cambridge University Press, Cambridge.

Allen, J. (1995). Linguistic aspects of speech synthesis. *Proc. National Acad. Sci. USA,* 92, 9946-9952.

Alm, N., Todman, J., Elder, L. & Newell, A.F. (1993). Computer Aided Conversation for Severely Physically Impaired Non-speaking People. *In: Proc. INTERCHI '93, 24-29 April 1993.* 236-241.

Arnott, J.L., Alm, N. & Murray, I.R. (1993). Enhancing a communication prosthesis with vocal emotion effects. *In: Proc. ESCA Workshop - Speech and Language Technology for Disabled Persons, KTH, Stockholm, Sweden, 31 May - 2 June 1993.* 165-168.

Bartkova, K. & Sorin, C. (1987). A model of segmental duration for speech synthesis in French. *Speech Communication,* 6(3), 245-260.

Bell Labs (1998). *http://www.research.att.com/cgi-bin/cgiwrap/mjm/voices.cgi*

Benoît, C. (1992). Semantically Unpredictable Sentences: Theoretical Basis for Generation. *In: SAM ESPRIT Project Final Report.*

Benoît, C. & Pols, L.C.W. (1992). On the assessment of synthetic speech. *In:* Bailly, G., Benoît, C. & Sawallis, T.R. (editors) *Talking Machines - Theories, Models, and Designs,* London, Elsevier Science Publishers. 435-441.

Bhaskararao, P. (1994). Subphonemic Segment Inventories for Concatenative Speech Synthesis. *In:* Keller, E. (editor) *Fundamentals of Speech Synthesis and Speech Recognition.* Chichester, England, John Wiley & Sons. 69-85.

Black, A.W. & Taylor, P. (1994). CHATR: a generic speech synthesis system. *In: Proc. COLING'94 15th Int. Conf. on Computational Linguistics, Kyoto, 5-9 August 1994.* Vol. 2, 983-986.

Breen, A.P. (1992a). Speech synthesis models: a review. *Electronics & Communication Engineering Journal,* February, 19-31.

Breen, A.P. (1992b). A comparison of statistical and rule-based methods of determining segmental durations. *In: Proc. ICSLP 92,* Banff, 1199-1202.

Breen, A.P. & Easton, M. (1994). A method of estimating segmental durations. *Proc. Institute of Acoustics,* 343-350.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984). *Classification and regression trees.* Monterey, CA, Wadsworth & Brooks.

Campbell, W. (1992). Syllable-based segmental duration. *In:* Bailly, G. & Benoît, C. & Sawallis, T.R. (editors) *Talking Machines - Theories, Models, and Designs,* London, Elsevier Science Publishers. 211-224.

Carlson, R., Granström, B. & Klatt, D.H. (1979). Some notes on the perception of temporal patterns in speech. *In:* Lindblom, B. & Ohman, S. (editors) *Frontiers of Speech Communication Research,* Academic Press, London. 233-243.

Carlson, R. & Granström, B. (1986). A search for durational rules in a real-speech data base. *Phonetica,* 43, 140-154.

Carlson, R. (1991). Synthesis: Modeling Variability and Constraints. *In: Proc. Eurospeech '91, 2nd European Conference on Speech Communication and Technology, 24-26 September 1991.* Vol. 3, 1043-1048.

Carlson, R., Granström, B., Neovius, L. & Nord, L. (1992). The "listening speed" paradigm for synthesis evaluation. *In: Proc. FONETIK'92, Sixth Swedish Phonetics Conference, 20-22 May 1992.*

Charpentier, F. & Stella, M.G. (1986). Diphone Synthesis Using an Overlap-Add Technique for Speech Waveform Concatenation. *In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing 86,* 2015-2018.

Childers, D.G. & Lee, C.K. (1991). Vocal quality factors: Analysis, synthesis and perception. *J. Acoust. Soc. Am.,* 90(5), 2394-2410.

Clarke, G.M. & Cooke, D. (1983). *A Basic Course in Statistics.* 2nd ed. London, Edward Arnold.

Cohen, M.M. & Massaro, D.W. (1993). Modeling Coarticulation in Synthetic Visual Speech. *In:* Magnenat-Thalmann, N. & Thalmann, D. (editors) *Models & Techniques in Computer Animation.* London, Springer-Verlag. 139-156.

Coker, C., Umeda, N. & Browman, C. (1973). Automatic synthesis from ordinary English text. *IEEE Trans. Aud. Electr.* AU-21(3), 293-298.

Cole, R.A. & Cooper, W.E. (1975). The perception of voicing in English affricates and fricatives. *J. Acoust. Soc. Am.,* 58, 1280-1287.

Crystal, D. (1987). *The Cambridge Encyclopedia of Language.* Cambridge, Cambridge University Press.

Delin, J. & Zacharski, R. (1997). Pragmatic Determinants of Intonation Contours for Dialogue Systems. *Int. J. Speech Technology,* 109-120.

Dettweiler, H. & Hess, W. (1985). Concatenation rules for demisyllable speech synthesis. *Acustica,* 57, 268-283.

Di Francesco, R. & Moulines, E. (1989). Detection of the Glottal Closure by Jumps in the Statistical Properties of the Signal. *In: Proc. Eurospeech 89, Paris.* Vol. 2, 39-41.

Dixon, N.R. & Maxey, H.D. (1968). Terminal analog synthesis of continuous speech using the diphone method of segment assembly. *IEEE Trans.,* AU-16, 40-50.

Donavon, R.E. (1996). Trainable Speech Synthesis. *PhD Thesis.* Cambridge, Cambridge University Press.

Dudley, H., Riesz, R.R. & Watkins, S.A. (1939). A synthetic speaker. *J. Franklin Inst.,* 227, 739-764.

Dudley, H. & Tarnoczy, T.H. (1950). The Speaking Machine of Wolfgang von Kempelen. *J. Acoust. Soc. Am.,* 22(2), 151-166.

Dutoit, T. & Leich, H. (1993). MBR-PSOLA: Text-to-Speech Synthesis Based on an MBE Resynthesis of the Segments Database. *Speech Communication,* 13, 435-440.

Dutoit, T. (1997). *An Introduction to Text-to-Speech Synthesis.* Netherlands, Kluwer Academic.

Edgington, M., Lowry, A., Jackson, P., Breen, A.P. & Minnis, S. (1996a). Overview of current text-to-speech techniques: Part 1 - text and linguistic analysis. *BT Technology Journal,* 14(1), 68-83.

Edgington, M., Lowry, A., Jackson, P., Breen, A.P. & Minnis, S. (1996b). Overview of current text-to-speech techniques: Part 2 - prosody and speech generation. *BT Technology Journal,* 14(1), 84-99.

Eefting, W. (1991). The effect of "information value" and "accentuation" on the duration of Dutch words, syllables, and segments. *J. Acoust. Soc. Am.,* 89(1), 412-424.

Elsendoorn, B.A. & 't Hart, J. (1982). Exploring the possibilities of speech synthesis with Dutch diphones. *IPO Annual Progress Report,* 17, 63-65.

Fallside, F. & Young, S.J. (1978). Speech output from a computer-controlled water supply network. *Proc. Institute of Electrical and Electronic Engineers,* 125(2), 157-161.

Fant, G. (1960). *Acoustic Theory of Speech Production.* The Hague, Mouton.

Fant, G., Kruckenberg, A. & Nord, L. (1991). Durational correlates of stress in Swedish, French and English. *Journal of Phonetics,* 19, 351-365.

Ferrero, F.E., Pelamatti, G.M. & Vagges, K. (1979). Perceptual Category Shift of Voiceless Italian Fricatives as a Function of Duration Shortening. *In:* Lindblom, B. & Ohman, S. (editors). *Frontiers of Speech Communication Research.* London, Academic Press. 159-165.

Festival (1997). *http://www.cstr.ed.ac.uk/projects/festival/userin.html*

Francis, W. N. & Kucera, H. (1982). *Frequency Analysis of English Usage: Lexicon and Grammar.* Boston, Houghton Mifflin Co.

Fujimura, O. & Lovins, J. (1978). Syllables as concatenative phonetic units. *In:* Bell, A. & Hooper, J. (editors) *Syllables and Segments.* Amsterdam, North-Holland. 107-120.

Fujisaki, H., Nakamura, K. & Imoto, T. (1975). Auditory Perception of Duration of Speech and Non-speech Stimuli. *In:* Fant, G. & Tatham, M.A.A. (editors) *Auditory Analysis and Perception of Speech.* London, Academic Press.

Hallahan, W.I. & Vitale, A.J. (1997). Software Text-To-Speech. *Int. J. Speech Technology,* 1, 121-134.

Halliday, M.A.K. (1970). *A course in spoken English: Intonation.* London, Oxford University Press.

't Hart, J., Collier, R. & Cohen, A. (1990). *A perceptual study of intonation. Cambridge, Cambridge University Press.*

Hawkins, S. & Warren, P. (1991). Factors affecting the given-new distinction in speech. *In: Proc. 12th Int. Congress of Phonetic Sciences, Aix-en-Provence, France, 19-24 August 1991.* Vol. 4, 66-69.

Hess, W.J. (1992). Speech Synthesis - A Solved Problem? *In:* Vandewalle, J., Boite, R., Moonen, M. & Oosterlinck, A. (editors) *Signal Processing VI: Theories and Applications.* Elsevier Science Publishers B.V.

Hirokawa, T., Itoh, K. & Sato, H. (1993). High quality speech synthesis system based on waveform concatenation of phoneme segment. *IEICE Trans. Fundamentals of Electronics, Communications & Computer Sciences,* E76-A, 11, 1964-1970.

Hirsh, I.J. (1959). Auditory perception of temporal order. *Journal of the Acoustical Society of America,* 31, 759-767.

Hirshberg, J. (1993). Pitch accent in context: predicting intonational prominence from text. *Artificial Intelligence,* 63, 305-340.

Holmes, J.N., Mattingly, I.G. & Shearme, J.N. (1964). Speech synthesis by rule. *Language and Speech,* 7, 127-143.

Holmes, J.N. (1973). The Influence of Glottal Waveform on the Naturalness of Speech from a Parallel Formant Synthesizer, *IEEE Transactions on Audio and Electroacoustics,* AU-21, 3, 298-305.

Huber, K. (1990). A statistical model of duration control for speech synthesis. *In: Proc. 5th European Signal Processing Conference (EUSIPCO-90), Barcelona, Spain, 18-20 Sept. 1990,* 1127-1130.

Huckvale, M. (1996). *SFS for Users.* University College London.

Hunt, A.J. & Black, A.W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In: *Proc. ICASSP-96, Atlanta, GA, 7-10 May 1996.*

Idson, W.L. & Massaro, D.W. (1981). The role of perceived duration in the identification of vowels. *Journal of Phonetics,* 8(4), 407-425.

Isard, S.D. & Miller D.A. (1986). Diphone synthesis techniques. *In: Proc. IEE International Conference on Speech Input/Output; Techniques and Applications, 24-26 March 1986.* London, IEE, Pub. no. 258.

Iwahashi, N. & Sagisaka, Y. (1995). Speech segment network approach for optimization of synthesis unit set. *Computer Speech and Language,* 9, 335-352.

Jekosch, U. & Pols, L.C.W. (1994). A Feature-Profile For Application-Specific Speech Synthesis Assessment And Evaluation. *In: Proc. Int. Conf. on Acoustics, Speech & Signal Processing.* Vol. 3, 1319-1322.

Johnston, R.D. (1996). Beyond intelligibility - the performance of text-to-speech synthesisers. *BT Technol. J.,* 14(1), 100-111.

Kaiki, N., Takeda, K. & Sagisaka, Y. (1990). Statistical analysis for segmental duration rules in Japanese speech synthesis. *In: Proc. Int. Conf. on Spoken Language Processing, Kobe, Japan.* Vol. 1, 17-20.

Kelly, J.L. & Gerstman, L.J. (1961). An artificial talker driven from a phonetic input. *J. Acoust. Soc. Am.,* 33, 835.

Klatt, D.H. (1972). Acoustic theory of terminal analog speech synthesis. *In: Proc. Int. Conf. Speech Communication Processing, Boston, MA, 24-26 April 1972.*

Klatt, D.H. (1975). Vowel lengthening is syntactically determined in a connected discourse. *J. Phonetics,* 3, 129-140.

Klatt, D.H. (1976). Linguistic uses of segmental duration in English: Acoustic and perceptual evidence. *J. Acoust. Soc. Am.,* 59, 1208-1221.

Klatt, D.H. (1979). Synthesis by rule of segmental durations in English sentences. *In:* Lindblom & Ohman (editors) *Frontiers of Speech Communication Research,* Academic Press, London. 287-299.

Klatt, D.H. (1980). Software for a Cascade/Parallel Formant Synthesizer. *J. Acoust. Soc. Am.,* 67, 971-995.

Klatt, D.H. (1987). Review of Text-to-Speech Conversion for English. *J. Acoust. Soc. Am.,* 82(3), 737-793.

Klatt, D.H. & Klatt, L.C. (1990). Analysis, synthesis and perception of voice quality variations among female and male talkers. *J. Acoust. Soc. Am.,* 87(2), 820-857.

Kohler, K.J. (1991). Prosody in speech synthesis: the interplay between basic research and TTS application. *Journal of Phonetics,* 19, 121-138.

Kortekaas, R.W.L. & Kohlrausch, A. (1997). Psychoacoustical evaluation of the pitch-synchronous overlap-and-add speech-waveform manipulation technique usxing single-formant stimuli. *J. Acoust. Soc. Am.,* 101(4), 2202-2213.

Kraft, V. (1993). Auditory Detection of Discontinuities in Synthesis-By-Concatenation. *In: Proc. Eurospeech '93.* 929-932.

Kraft, V. (1994). Does the resulting speech quality improvement make a sophisticated concatenation of time-domain synthesis units worthwhile? *In: Proc. 2nd ESCA/IEEE Workshop on Speech Synthesis, New Paltz, NY.* 65-68.

Kraft, V. & Portele, T. (1995). Quality Evaluation of Five German Speech Synthesis Systems. *Acta Acustica,* 3, 351-365.

KTH (1998). *http://www.speech.kth.se/cgi-bin/SayTrans*

Laureate (1998). *http://innovate.bt.com/showcase/laureate/demo/body.htm*

Lawrence, W. (1974). The phoneme, the syllable, and the parameter track. *In: Proc. Speech Communication Seminar, Stockholm, 1-3 August 1974.*

Lehiste, I. (1977). Isochrony reconsidered. *Journal of Phonetics,* 5, 253-63.

Levinson, S.E., Coker, C.H. & Schmidt, C.E. (1981). Adaptive emulation of human speech by an articulatory speech synthesiser. *Bell Laboratories Report.*

Levitin, D. J. (1998). Memory for Musical Attributes. *In:* Cook, P.R. (ed.) *Music, Cognition and Computerized Sound: An Introduction to Psychoacoustics.* MIT Press.

Linggard, R. (1985). *Electronic Synthesis of Speech.* Cambridge, Cambridge University Press.

Logan, J.S., Greene, B.G. & Pisoni, D.B. (1989). Segmental intelligibility of synthetic speech produced by rule. *J. Acoust. Soc. Am.,* 86(2), 566-581.

Macromedia. (1998). *http://www.macromedia.com*

Markel, J.D. & Gray, A.H. (1976). *Linear Prediction of Speech.* New York, Springer-Verlag.

Mattingly, I.G. (1971). Synthesis By Rule As A Tool For Phonological Research. *Language & Speech,* 14, 47-56.

McAllister, M. (1989). The Problems of Punctuation Ambiguity in Fully Automatic Text-to-Speech Conversion. *In: Proc. Eurospeech 89, Paris.* Vol. 1, 538-541.

McCulloch, N., Bedworth, M. & Bridle, J. (1987). NETspeak - a re-implementation of NETtalk. *Computer Speech and Language,* 2, 284-301.

Minnis, S. (1994). The PARSODY System: Automatic Prediction of Prosodic Boundaries for Text-to-Speech. *In: Proc. COLING '94.* Association of Computational Linguistics.

Möbius B. & van Santen J. (1996). Modeling Segmental Duration in German Text-to-Speech Synthesis. *In: Proc. Int. Conf. on Spoken Language Processing, Philadelphia.* Vol. 4, 2395-2398.

Morton, K. (1987). SYNTHesis - EXpert: An Expert System for Improving Naturalness in Synthetic Speech. *In: Proc. Expert Systems '87, Brighton, December 1987.*

Morton, K. (1991). Expectations for assessment techniques applied to speech synthesis. *In: Proc. Institute of Acoustics,* 13(2), 501-508.

Moulines, E. & Charpentier, F. (1990). Pitch Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones. *Speech Communication,* 9, 453-467.

Murray, I.R. & Arnott, J.L. (1993). Toward the simulation of emotion in synthetic speech - A review of the literature on human vocal emotion. *J. Acoust. Soc. Am.,* 93(2), 1097-1108.

Nation, P. (1993). Vocabulary Size, Growth, and Use. *In:* Schreuder, R. & Weltens, B. (editors) *The Bilingual Lexicon.* Amsterdam, John Benjamins Publishing Company. 115-134.

Naval Research Laboratory. (1976). Automatic translation of English text to phonetics by means of letter-to-sound rules. *In: NRL Report 7948.*

Nooteboom, S.G. (1973). The perceptual reality of some prosodic durations. *Journal of Phonetics,* 1, 23-45.

Nooteboom, S.G. (1975). On the internal, auditory representation of syllable nucleus durations. *In:* Fant, G. & Tatham, M.A.A. (editors) *Auditory Analysis and Perception of Speech.* London, Academic Press, 413-430.

Olive, J.P. (1977). Rule synthesis of speech from dyadic units. *In: Proc. Int. Conf. Acoust. Speech Signal Process. ICASSP-77,* 568-570.

Oller, D.K. (1973). The effect of position in utterance on speech segment duration in English. *J. Acoust. Soc. Am.,* 54, 1235-1247.

O'Shaughnessy, D. (1977). Fundamental frequency by rule for a text-to-speech system. *In: Proc. Int. Conf. on Acoustics, Speech and Signal Processing.* IEEE.

O'Shaughnessy, D. (1984). A multispeaker analysis of durations in real French paragraphs. *J. Acoust. Soc. Am.,* 76(6), 1664-1672.

O'Shaughnessy, D. (1987). *Speech Communication - Human and Machine.* Wokingham, Addison-Wesley.

O'Shaughnessy, D., Barbeau, L., Bernardi, D. & Archambault, D. (1988). Diphone Speech Synthesis. *Speech Communication,* 7, 55-65.

Ostendorf, M., Price, P., Bear, J. & Wightman, C.W. (1990). The use of relative duration in syntactic disambiguation. *In: Proc. 3rd DARPA Speech and Natural Language Workshop, June 1990.*

Ozum, I.Y & Bulut, M.M. (1994). Knowledge Based Speech Synthesis by Concatenation of Phoneme Samples. *In: Proc. 7th Mediterranean Electrotechnical Conference, Antalya, Turkey, 12-14 April 1994.* New York, IEEE, 1, 51-54.

Parke, F.I. & Waters, K. (1996). *Computer Facial Animation.* Wellesley, MA, A.K. Peters.

Pavlovic, C.V., Rossi, M. & Espesser, R. (1990). Use of the magnitude estimation technique for assessing the performance of text-to-speech synthesis systems. *J. Acoust. Soc. Am.,* 87(1), 373-382.

Pearson, S., Holm, F. & Hata, K. (1997). Combining Concatenation and Formant Synthesis for Improved Intelligibility and Naturalness in Text-to-Speech Systems. *Int. J. of Speech Technology,* 1(2), 103-108.

Pelachaud, C. & Prevost, S. (1995). Talking Heads: Physical, Linguistic and Cognitive issues in Facial Animation. *In: Course Notes - Computer Graphics International, June 1995.*

Peterson, G., Wang, W. & Silvertsen, A. (1958). Segmentation Techniques in Speech Synthesis. *J. Acoustical Society of America,* 30, 739-748.

Pierrehumbert, J. (1981). Synthesizing Intonation. *J. Acoust. Soc. Am.*, 70(4), 985-995.

Pitrelli, J. & Zue, V. (1989). A hierarchical model for phoneme duration in American English. *In: Proc. Eurospeech '89, European Conf. on Speech Comm & Technology, Paris.* Vol. 2, 324-327.

Port, R. (1981). Linguistic timing factors in combination. *J. Acoust. Soc. Am.* 69, 262-274.

Portele, T., Hofer, F. & Hess, W. (1994). Structure and representation of an inventory for German Speech Synthesis. *In: Proc. ICSLP 94 International Conference on Spoken Language Processing, Tokyo, Japan.* Acoustical Society of Japan, Vol. 4, 1759-1762.

Posmyk, R. (1989). Time-Domain Synthesizer for preserving Microprosody. *In: Eurospeech '89,* Vol. 2, 191-194.

Potapova, R.K. (1975). Auditory estimate of syllable and vowel duration in sentences. *In:* Fant, G. & Tatham, M.A.A. (editors) *Auditory Analysis and Perception of Speech.* London, Academic Press. 465-476.

Rabiner, L.R. (1969). A model for synthesizing speech by rule. *IEEE Trans. Audio,* 17, 7-13.

Rahim, M.G., Goodyear, C.C., Kleijn, W.B., Schroeter, J. & Sondhi, M.M. (1993). On the use of neural networks in articulatory speech synthesis. *J. Acoust. Soc. Am.,* 93(2), 1109-1121.

Riley, M.D. (1992). Tree-based modelling of segmental durations. *In:* Bailly, G., Benoît, C. & Sawallis, T.R. (editors) *Talking Machines - Theories, Models, and Designs.* London, Elsevier Science Publishers. 265-273.

Robinson, A.J. (1994). An Application of Recurrent Nets to Phone Probability Estimation. *IEEE Transactions on Neural Networks,* 5(2), 298-305.

Rodet, X. (1980). Time-domain formant-wave-function synthesis. *In: Spoken language generation and understanding. In: Proc. NATO ASI, Bonas, France, 23 June - 5 July 1979.* Reidel, Dordrecht, 429-440.

Rowden, C. (1992). Synthesis. *In:* Rowden, C. (editor) *Speech Processing.* London, McGraw-Hill. 184-222.

Rudnicky, A.I., Hauptmann, A.G. & Lee, K.-F. (1994). Survey of Current Speech Technology. *Communications of the ACM,* 37(3), 52-57.

Sagisaka, Y. (1990). Speech Synthesis from Text. *IEEE Communications Magazine,* January, 35-41.

Sahandi, R. & Vine, D.S.G. (1996). A Genetic Algorithm Approach for obtaining Diphone Durations for English Speech Synthesis. *In: DSP Scandinavia 96, Copenhagen, Denmark, 18-19 June 1996.*

Sahandi, R. & Vine, D.S.G. (1997a). A Concatenation Framework for Text-to-Speech Synthesis using CVC Syllables. *In: Proc. SALT Workshop on Evaluation in Speech and Language Technology, University of Sheffield, UK, June 17-18.* 172-179.

Sahandi, R. & Vine, D.S.G. (1997b). Text-to-Speech Animation for the Hearing Impaired. *In: Proc. KT97 Int. Conf. on Knowledge Transfer, London, 15-16 July 1997.*

Saito, S. (editor) (1992). *Speech science & technology.* Tokyo, Ohmsha.

Sanderman, A.A. & Collier, R. (1996). Prosodic rules for the implementation of phrase boundaries in synthetic speech. *J. Acoust. Soc. Am.,* 100(5), 3390-3397.

Sejnowski, T. & Rosenberg, C.R. (1987). Parallel Networks that Learn to Pronounce English Text. *Complex Systems,* 1, 145-168.

Selkirk, E. (1984). *Phonology and Syntax.* Cambridge, MA, MIT Press.

Sharman R.A. (1994). Concatenative speech synthesis using subphoneme segments. *In: Proc. Institute of Acoustics,* 16(5), 367-374.

Shih C. & Sproat R., (1996). Issues in Text-to-Speech Conversion for Mandarin. *Computational Linguistics and Chinese Language Processing,* 1(1), 37-86.

Siegel, S. & Castellan, N. (1988). *Nonparametric Statistics for the Behavioural Sciences.* London, McGraw-Hill.

Silverman, K., Beckman, M., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J. & Hirschberg, J. (1992). ToBI: a standard for labelling English prosody. *In: Proc. ISCLP92.* Vol. 2, 867-870.

Sonntag, G.P. & Portele, T. (1996). A framework to evaluate and verify the presence of linguistic concepts in the prosody of spoken utterances. *In: Proc. 3rd SPEAK!-Workshop, Budapest.*

Sonntag, G.P. & Portele, T. (1997). A method for prosody evaluation. *In: Proc. SALT Workshop on Evaluation in Speech and Language Technology, University of Sheffield, UK, June 17-18 1997.* 188-194.

Sorin, C., Larreur, D. & Llorca, R. (1987). A Rhythm-Based Prosodic Parser for Text-to-Speech Systems in French. *In: Proc. 11th Int. Congress on Phonetic Sciences, Tallin.* Vol. 1, 125-128.

Takeda, K., Sagisaka, Y. & Kuwubara, H. (1989). On sentence-level factors governing segmental duration in Japanese. *J. Acoust. Soc. Am.,* 86(6).

Takeda, K., Katsuo, A. & Sagisaka, Y. (1992). On the basic scheme and algorithms in non-uniform unit speech synthesis. *In:* Bailly, G., Benoît, C. & Sawallis, T.R. (editors) *Talking Machines: Theories, Models and Designs,* London, Elsevier Science Publishers B.V. 93-105.

Tatham, M.A.A. (1969). Classifying Allophones. *Occasional Papers,* 3. Department of Linguistics, University of Essex.

Tatham, M.A.A. (1970). Articulatory Speech Synthesis by Rule: Implementation of a Theory of Speech Production. *Working Papers.* The Ohio State University, Computer and Information Science Research Center.

Tatham, M.A.A., Morton, K. & Mansell, P. (1971). Speech Synthesis and Models of Speech Production - II. *Interim Report: Science Research Council Award B/SR/6733.*

Tatham, M.A.A. (1991). Units of representation for speech synthesis. *In: Proc. Institute of Acoustics,* 13(2), 519-526.

Tatham, M.A.A. & Lewis, E. (1992). Prosodic assignment in SPRUCE text-to-speech synthesis. *In:* Lawrence, R. (editor) *Proc. Institute of Acoustics.* Vol. 14. St. Albans, IoA.

Tatham, M.A.A. & Morton, P.K.J. (1996). Natural voice output in interactive information systems. *In:* Lawrence, R. (editor) *Proc. Institute of Acoustics.* 18(9), 43-50. St. Albans, IoA.

Traber, C. (1992). F0 Generation with a Database of Natural F0 Patterns and with a Neural Network. *In:* Bailly, G., Benoît, C. & Sawallis, T.R. (editors) *Talking Machines - Theories, Models, and Designs.* London, Elsevier Science Publishers. 287-304.

TruVoice (1998). *http://www.westcler.k12.oh.us/johnny/truvoice.html*

Umeda, N. (1977). Consonant duration in American English. *J. Acoust. Soc. Am.,* 61, 846-858.

Valbret, H., Moulines, E. & Tubach, J.P. (1992). Voice transformation using PSOLA technique. *Speech Communication* 11, 175-187.

van Santen, J.P.H. (1992). Contextual effects on vowel duration. *Speech Communication,* 11, 513-546.

van Santen, J.P.H. (1993). Timing in Text-to-Speech Systems. *In: Proc. Eurospeech 93, Berlin.* 1397-1404.

van Santen, J.P.H. (1994). Assignment of segmental duration in text-to-speech synthesis. *Computer Speech and Language,* 8, 95-128.

Veldhuis, R. (1998). A computationally efficient alternative for the Liljencrants-Fant model and its perceptual evaluation. *J. Acoust. Soc. Am.,* 103(1), 566-571.

Vine, D.S.G. & Sahandi, R. (1996). Optimisation of segmental time durations in concatenative speech synthesis. *In: Proc. KT96 International Conference on Knowledge Transfer, London, 22-26 July, 1996.*

Vine, D.S.G. & Sahandi, R. (1997). Timing Databases for Concatenative Speech Synthesis. *In: Proc. DSP Scandinavia 97, Stockholm, Sweden, 3-4 June, 1997.* 21-24.

Voiers, W.D. (1983). Evaluating Processed Speech Using the Diagnostic Rhyme Test. *Speech Technology,* January-February, 30-39.

Vonwiller, J., Cleirigh, C., Garsden, H., Kumpf, K., Mountstephens, R. & Rogers, I. (1997). Development and Application of an Accurate and Flexible Automatic Aligner. *Int. J. Speech Technology,* 1, 151-160.

Williams, B. (1994). Welsh letter-to-sound rules: rewrite rules and two-level rules compared. *Computer Speech and Language,* 8, 261-277.

Witten, I.H. (1982). *Principles of Computer Speech.* London, Academic Press.

Witten, I.H. (1986). *Making Computers Talk: An Introduction to Speech Synthesis.* London, Prentice-Hall International (UK) Ltd.

Yarrington, D. & Foulds, R. (1993). Personalizing Synthesized Voices. *In: ESCA Workshop, KTH, Stockholm, 31 May-2 June.* 169-172.

Yazu, T. & Yamada, K. (1986), The speech synthesis system for an unlimited Japanese vocabulary. *In: Proc. IEEE-ICASSP86.* 2019-2022.

Yost, W.A. (1994). *Fundamentals of Hearing - An Introduction.* 3rd ed. London, Academic Press.

OPTIMISATION OF SEGMENTAL TIME DURATIONS IN CONCATENATIVE SPEECH SYNTHESIS

Daniel S.G.Vine
Dr. Reza Sahandi

Department of Computing
Bournemouth University
Talbot Campus
Fern Barrow
Poole
Dorset
United Kingdom
BH12 5BB

Tel: (+44) 1202 595531
Fax: (+44) 1202 595314
Email: dvine@bournemouth.ac.uk

Abstract

A segmental timing module for English text-to-speech synthesis has been developed which allocates time durations to speech segments. Segmental duration plays an important part in the quality of concatenative speech synthesis. It is essential that the overall timing of the synthesised speech is correct, so that the speech will sound more natural to listeners. Traditional timing modules use sets of duration rules which attempt to account for a range of timing factors. These include syntactic, lexical and semantic factors.

The system developed avoids an explicit set of rules, and is based on diphones. It uses a database to obtain appropriate diphone durations for whole words. Contextual and positional effects are accounted for where possible, by extracting the largest possible sub-string of duration values from any database entry. Initial results show a noticeable improvement in the intelligibility of speech produced, compared with the output of a timing module which uses only a single duration value for any diphone. The paper also contains a brief discussion on the synchronisation of the speech with lip-readable animation.

Introduction

A segmental timing module for use in an English text-to-speech synthesiser has been developed. The timing module allocates suitable time durations to speech segments for output. The speech segments are concatenated to form continuous speech. In concatenative speech synthesis, segments are output sequentially, each for a certain length of time before the next one is played. Every speech segment must, therefore, be assigned an appropriate time duration for which it will be output.

Many factors have an influence upon segmental timing. These include semantic factors, syntactic factors, and lexical factors (Klatt, 1976). Semantic factors include contrastive stress and variations in speaking rate. For instance, speakers will often slow their speaking rate when semantically novel words are encountered. The main syntactic factors that influence segmental timing are the locations of phrase and clause boundaries. These may be determined manually, or identified from the input text using a stochastic approach (Edgington *et al.,* 1996). Lexical factors incorporate the combination of speech segments that is used for synthesis, along with stress markers, which define a speech segment's stress level. This representation is produced from text by the lexical component of a speech synthesiser, and is then input to a timing module, which predicts a time duration for each speech segment (Klatt, 1976).

It is essential that the overall timing of the synthesised speech is correct, in order to make speech sound more natural to listeners. Segmental duration also plays an important part in the prosody of concatenative speech. For example, it can be used to slow the speaking rate to emphasize a particular word. If natural-sounding speech is to be produced, then considerable attention should be paid to the time duration allocated to each segment.

Duration Models

Segmental duration models for text-to-speech synthesis attempt to account for some or all of the timing factors, including syntactic, lexical and semantic factors. Durations are usually predicted by modifying a default set of time durations using rules or formulae, or by directly specifying the typical durations of each segment for every type of context. Duration models fall into four categories: sequential rule systems, equation systems, table look-up systems and binary trees (Campbell, 1990). Segmental duration models mostly use phonemes as the underlying unit for measurement (Klatt 1976, Pitrelli & Zue 1989), although models exist for other segments such as diphones (Huber, 1990) and syllables (Edgington *et al.,* 1996). (N.B. A diphone consists of the transition from the centre of one phoneme to the centre of the next).

Klatt's (1979a) rule-based model is widely used, and forms the basis for many other models. In this model, an 'inherent' duration is assigned to each phoneme. This duration is derived by considering a phoneme at the start of a word, preceding a stressed vowel. Durational rules cause a percentage change in the segment's inherent duration, taking into account factors like syntactic information, position in a word, stress markers, etc. In such models, a second value must be supplied in order to limit the extent to which a phoneme can be shortened. Klatt termed this a 'minimum' duration value, corresponding to the minimum duration for that phoneme if stressed. One problem with such models is that the inherent durations for phonemes are difficult to estimate (Huber, 1990), and vary between different studies (Carlson & Granstrom 1986, Umeda 1977; Klatt 1979b). A further difficulty is that it is difficult for such models to express a relationship between a context effect and an inherent duration modification (Port, 1981).

Many duration models differ in complexity and scope. For diphones, it is possible to limit the major duration considerations to *vowel-consonant diphones*, i.e. diphones containing the transition from a vowel to a consonant (O'Shaughnessy *et al.*, 1988). It may not be appropriate to adapt a phoneme duration model for diphones, since a decision must be made as to how a phoneme's duration should be shared between two diphones (O'Shaughnessy *et al.*, 1988).

Duration models are generally based upon studies of recorded utterances. Durational rules may be developed using a speech database (Carlson & Granstrom, 1986). Carlson *et al* used context sensitive rules when searching the database for data relevant to the particular rule being formulated. The main disadvantage of rule-based models is that they have failed to reach an integrated view of how timing should be used in speech synthesis (Port, 1981). When so many factors such as position within a word, semantic prominence and lexical stress have an influence upon duration, it becomes a difficult task to

account for them all. Alternative techniques for investigating segmental durations are becoming more widespread, such as statistical approaches (Edgington *et al.* 1996, Riley 1992).

The approach taken in this project is to avoid explicit sets of rules. The system developed uses a look-up from a duration database to obtain relevant time durations for English words. Input words are initially converted to a list of diphones, and then the timing module provides a set of corresponding duration values for the diphones. A concatenative speech synthesiser produces speech output using a database of pre-recorded human speech segments and the duration values obtained from the duration database.

Concatenative Speech Synthesiser

The concatenative speech synthesiser produces speech output by concatenating pre-recorded diphone waveforms. The waveforms were digitally recorded using the voice of a single, male speaker. Factors such as pitch, volume and distance from the microphone were kept as constant as possible. Diphones were obtained by manually editing the waveforms. The speech for any input diphone list is produced for the durations specified, by concatenating these waveforms.

Timing Module

When a new word is presented to the text-to-speech system, it is first converted to a set of diphone symbols, which are then input to the duration module. The duration module assembles a set of time durations from a database, corresponding to the diphone list. The database consists of diphone labels for English words, along with their corresponding duration values.

To construct the database, diphone time durations were collected experimentally for commonly used English words (see figure 1). For each word, an input string of diphones was supplied, along with an initial set of time values. These values were used to synthesise speech for the word, using a concatenative speech synthesiser, and the results were judged by a human listener. For each word, the time values were iteratively altered, until the speech output was judged to be the best possible. The final string of diphone values were then stored in the database along with the appropriate diphone symbols. Using synthesised speech to support the building of a duration database is an alternative approach that is being tested. A more common method is to use recordings of continuous speech, labelled with segmental durations, when constructing timing systems. However, it is possible that durations derived from spoken speech may not be as appropriate for use in speech synthesis, compared to those obtained from synthesised speech.

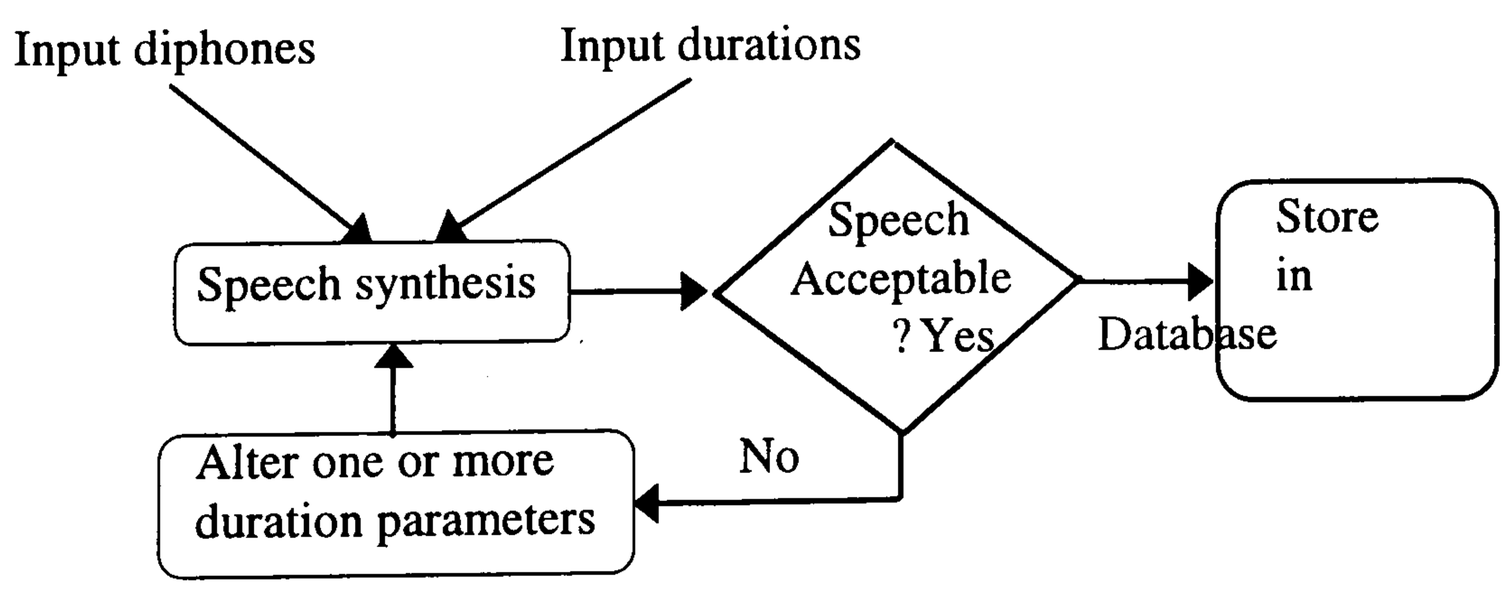


Fig.1: Creation of diphone duration database

To assign time durations, an algorithm has been developed which attempts to preserve any contextual effects due to neighbouring diphones. Context is accounted for by extracting the largest possible sub-strings of duration values from the database when assembling the output set of duration values. The algorithm also makes use of positional information within a word. If several identical sub-strings are found in the database which correspond to a section of the input diphone string, then the one that is most similar to the input sub-string, with respect to position, is chosen. In this look-up procedure, if a diphone is not represented in the database, then it is assigned an average duration value by default.

An initial approach that was investigated was the use of a genetic algorithm (Goldberg, 1989) to optimise duration values. This was one way of predicting diphone durations without relying upon grammatical or syntactic rules. The present database system was derived from the genetic algorithm investigation, although the use of a genetic algorithm itself has been discontinued.

The accuracy of the extracted time durations may be increased by using a larger duration database. For any output string of time durations, a sub-string of values taken from a single database entry is more likely to be accurate. By increasing the database size, it is more likely that any output string will be derived from fewer sub-strings, and hence will have a greater contextual accuracy. Further study is now required into the optimum number of words for the database.

Synchronisation issues

Concatenative speech has been combined with moving lip images, producing animated speech synthesis. In a prototype system, a set of stored lip images forms the necessary visual phonemes, or 'visemes', for portraying speech. Speech animation is achieved with a minimal set of nine visemes since one viseme can correspond to several acoustic classes. The animation consists of displaying pre-stored lip images in sequence according to the speech output.

For synchronisation with speech, the length of the image sequence for each diphone is adjustable. This is achieved by displaying a smaller or larger number of intermediate images as the diphone is output. This method is convenient from a speech production point of view: the duration for which a diphone can be output is a controllable parameter, and so the animation sequence is automatically adjusted to last for this time. At present, three pre-stored lip images are specified for each diphone. Thus, each different lip image is displayed repeatedly for a third of the total duration. The animation is currently being improved so that a greater number of intermediate images are specified for each diphone. This will improve the 'lip-readability' of the system.

Results

The output of the current timing module was used to synthesise speech using a waveform concatenation synthesiser. When compared with the output of a timing module which only contained a single duration value for any diphone, the resulting speech was noticeably superior, although some discontinuities were still present. However, the timing module could be further improved by increasing the size of the duration database, since the database may not contain entries for some diphones. In such cases, a single default value is used. Increasing the database size would fill these 'gaps' in the database, and supply a greater number of contexts for any particular diphone. The performance and accuracy of the system will be monitored as the database size increases. No comparisons have yet been made with traditional rule-based timing systems.

Conclusions

A method has been described for producing optimal time durations for speech segments, for use in concatenative speech synthesis. A database has been produced, consisting of manually edited speech segment durations for commonly used English words. For any input diphone string, duration values are extracted from this database, in such a way as to account for contextual and positional data where possible. Whilst this method has been used for English diphone synthesis, it may also be used for other speech segment types and for other languages, providing that an appropriate database is created.

References

Campbell, W.N. (1990). Syllable-based segmental duration. *In:* Bailly, G., Benoit, C. and Sawallis, T.R. (eds.) *Talking Machines: Theories, Models and Designs.* Amsterdam, Elsevier Science Publishers, 211-224.

Carlson, R. and Granstrom, B. (1986). A Search for Durational Rules in a Real-Speech Data Base. *Phonetica,* 43, 140-154.

Edgington, M. *et al.* (1996). Overview of current text-to-speech techniques: Part 2 - prosody and speech generation. *BT Technol. Journal,* 14(1), 84-99.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning,* Wokingham, England, Addison-Wesley.

Huber, K. (1990). A statistical model of duration control for speech synthesis. *In: Proc. 5th European Signal Processing Conference (EUSIPCO-90), Sept. 18-20, Barcelona, Spain,* 1127-1130.

Klatt, D.H. (1976). Linguistic uses of segmental duration in English: Acoustic and perceptual evidence. *J. Acoust. Soc. Am.,* 59, 1208-1221.

Klatt, D.H. (1979a). Synthesis by rule of segmental durations in English sentences. *In: Proc. 9th Int. Congress of Phonetic Sciences, Copenhagen,* 290-297.

Klatt, D.H. (1979b). Synthesis by Rule of Segmental Durations in English Sentences. *In:* Lindblom, B. and Ohman, S. (eds.) *Frontiers of Speech Communication Research,* London, Academic Press, 287-300.

O'Shaughnessy, D. *et al.* (1988). Diphone Speech Synthesis. *Speech Communication,* 7, 55-65.

Pitrelli, J. and Zue, V. (1989). A hierarchical model for phoneme duration in American English. *In: Proc. Eurospeech '89, European Conf. on Speech Comm & Technology, Paris,* 2, 324-327.

Port, R. (1981). Linguistic timing factors in combination. *J. Acoust. Soc. Am.,* 69, 262-274.

Riley, M.D. (1992). Tree-based modelling of segmental durations. *In:* Bailly, G., Benoit, C. and Sawallis, T.R. (eds.) *Talking Machines: Theories, Models and Designs.* Amsterdam, Elsevier Science Publishers, 265-273.

Umeda, N. (1977). Consonant duration in American English. *J. Acoust. Soc. Am.,* 61, 846-858.

Paper to be submitted at Digital Signal Processing Scandinavia '97, 3-4 June 1997, Sky City, Stockholm.

Authors: Daniel S.G.Vine
Dr. Reza Sahandi

Address: Department of Computing
Bournemouth University
Talbot Campus
Fern Barrow
Poole
Dorset
United Kingdom
BH12 5BB

Tel: (+44) 1202 595502
Fax: (+44) 1202 595314
Email: dvine@bournemouth.ac.uk

Segmental Duration Prediction for Concatenative Speech Synthesis.

Abstract

Speech synthesis via waveform concatenation requires a database of of highly accurate segmental timing data for smooth, intelligible speech to be output. Timing accuracy is essential for smooth transitions at segment boundaries. A concatenative speech synthesizer using a waveform inventory consisting of consonant-vowel (CV), vowel-consonant (VC) and vowel-consonant-vowel (CVC) syllables is discussed. Speech is formed by outputting these segments sequentially, with given duration values. If appropriate durations are chosen, acceptable speech can be synthesized without the need to apply smoothing algorithms.

A timing database for waveform concatenation synthesis has been built. The database consists of segmented words with their corresponding duration values, and is constructed using an iterative synthesis/adjustment approach. A segmental duration prediction algorithm produces duration values independently of segment type. This is achieved by assembling sub-strings of duration values taken from one or more database entries. The algorithm is guided by three rules which do not require a lengthy text analysis, and allows segmental durations to be output given only an input stream of phonemes. An assessment of the timing system indicates the need for a larger timing database.

Segmental Duration Prediction for Concatenative Speech Synthesis.

1. Introduction

Speech synthesis via waveform concatenation requires a database of highly accurate segmental timing data for smooth, intelligible speech to be output. Accuracy in timing is essential since it determines the quality of concatenative speech output. In this research, a concatenative speech synthesizer uses a waveform inventory consisting of consonant-vowel (CV), vowel-consonant (VC) and vowel-consonant-vowel (CVC) syllables. The syllables are pre-recorded segments of human speech. Continuous speech is synthesized for whole words by outputting waveforms from the inventory in sequence, with a duration value being applied to each segment.

In waveform concatenation, accurate segmental timing data is necessary, since the segmental duration determines the *initial portion* of each waveform to be output, enabling smoother speech to be produced at segment boundaries. In this manner, each waveform is output from its initial position up to a point determined by the segment duration value. This is shown in figure 1, in which portions of the syllables /DI/ and /Is/ are used to synthesize the word 'this' (the portions are indicated by arrows). The first two windows show the complete syllables, as stored in the waveform inventory and the final window shows the synthesized word 'this'. As the diagram shows, only the portion of the /DI/ waveform is output before /Is/. For intelligible speech to be output, waveform concatenation requires a high degree of timing accuracy, since the timing data is specific to a particular inventory of speech waveforms. This is unlike other speech production methods in which each segment is squashed or stretched to fit a particular duration (Isard & Miller, 1986). Since such speech synthesis methods use smoothing procedures at segmental boundaries, a wider range of segmental durations can be tolerated.

A timing database for waveform concatenation synthesis has been built. An iterative synthesis/adjustment approach involving a human listener was adopted to assign appropriate segmental durations to segmented words (Vine & Sahandi, 1996). This technique implicitly enhances the speech rhythm through the listening and improving cycle. Thus, timing data is produced that goes some way towards incorporating a correct perception of speech rhythm. This is advantageous, because traditional timing systems have no explicit speech rhythm model, and as a result, the speech rhythm may not be correct (Breen, 1992).

The timing database consists of segmented words along with their corresponding duration values. Speech can be synthesized using a simple look-up method for each word contained in the timing database. A database consisting of pre-defined words has only a limited application. It is much more beneficial to use a timing database to predict segmental durations for novel words. The following section describes a segmental duration prediction system for unrestricted text input, using such a database.

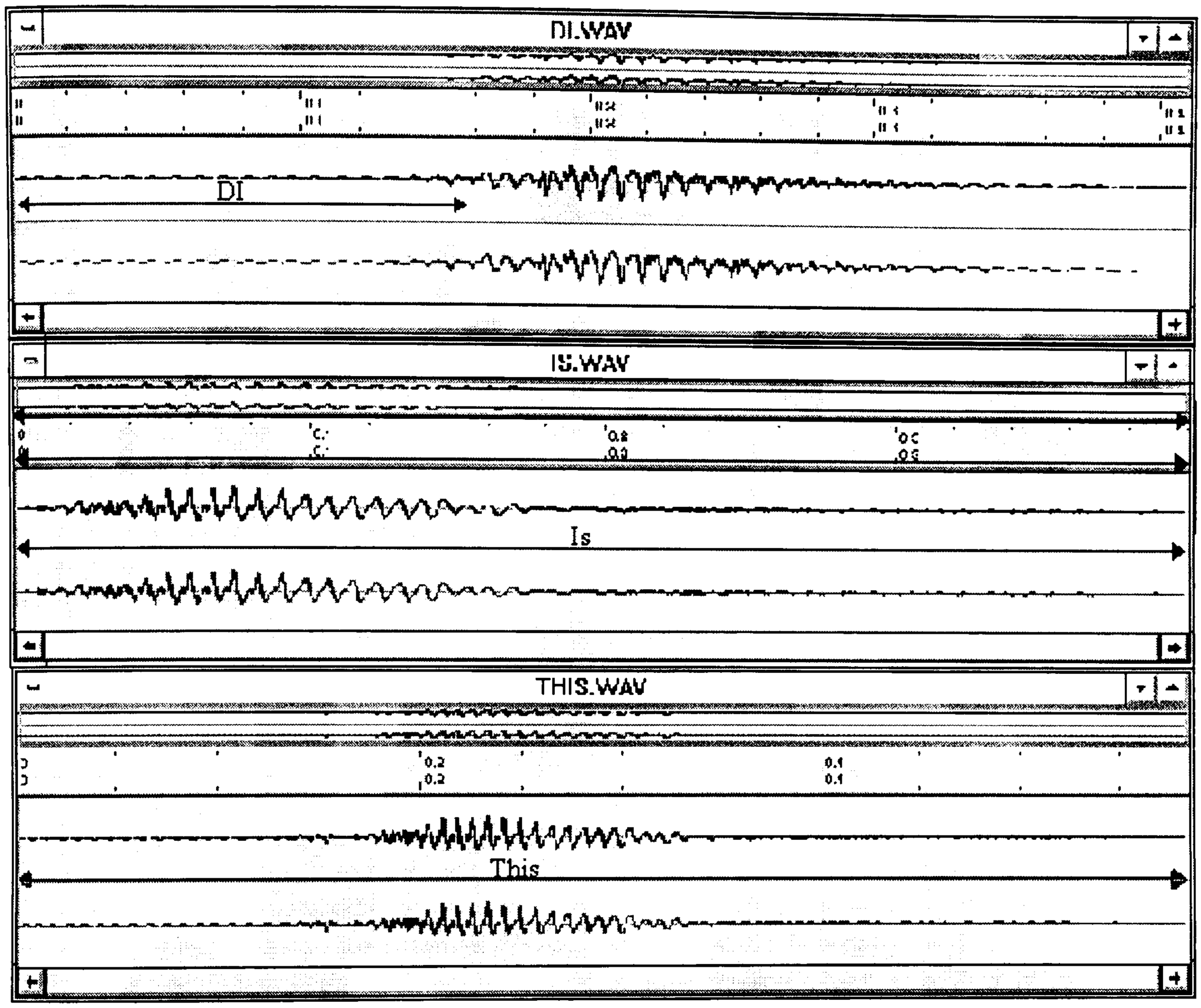


Fig.1: Portions of waveforms used to synthesize the word 'this'

2. Segmental Duration Prediction

An input word is first converted into a segmental representation, for which timing information must be predicted. The segmented word is input to the timing system, which uses the timing database to predict appropriate segmental durations. The database consists of a list of segmented words that are commonly used in English. This list is accompanied by a list of corresponding durations for each segmented word. So, for example, the words 'card' and 'this' could be stored as follows, where the numerical values represent segmental duration in milliseconds:

[[kA:, A:d][106, 381], [DI, Is][189, 396],...].

The duration prediction algorithm works by assembling sub-strings of durations from the database. In this paper, a 'sub-string' refers to an unbroken sequence of segmental duration values, taken from a single database entry. When a list of input segments representing the input word is received, the duration prediction algorithm searches the database for sub-strings that match some or all of the input list. If several sub-strings

containing the same segment are found, the prediction algorithm applies a series of three rules to determine which duration sub-string to use in the output list of durations. These rules are described in section 2.1. If no duration value is available for a particular segment, a default value is used.

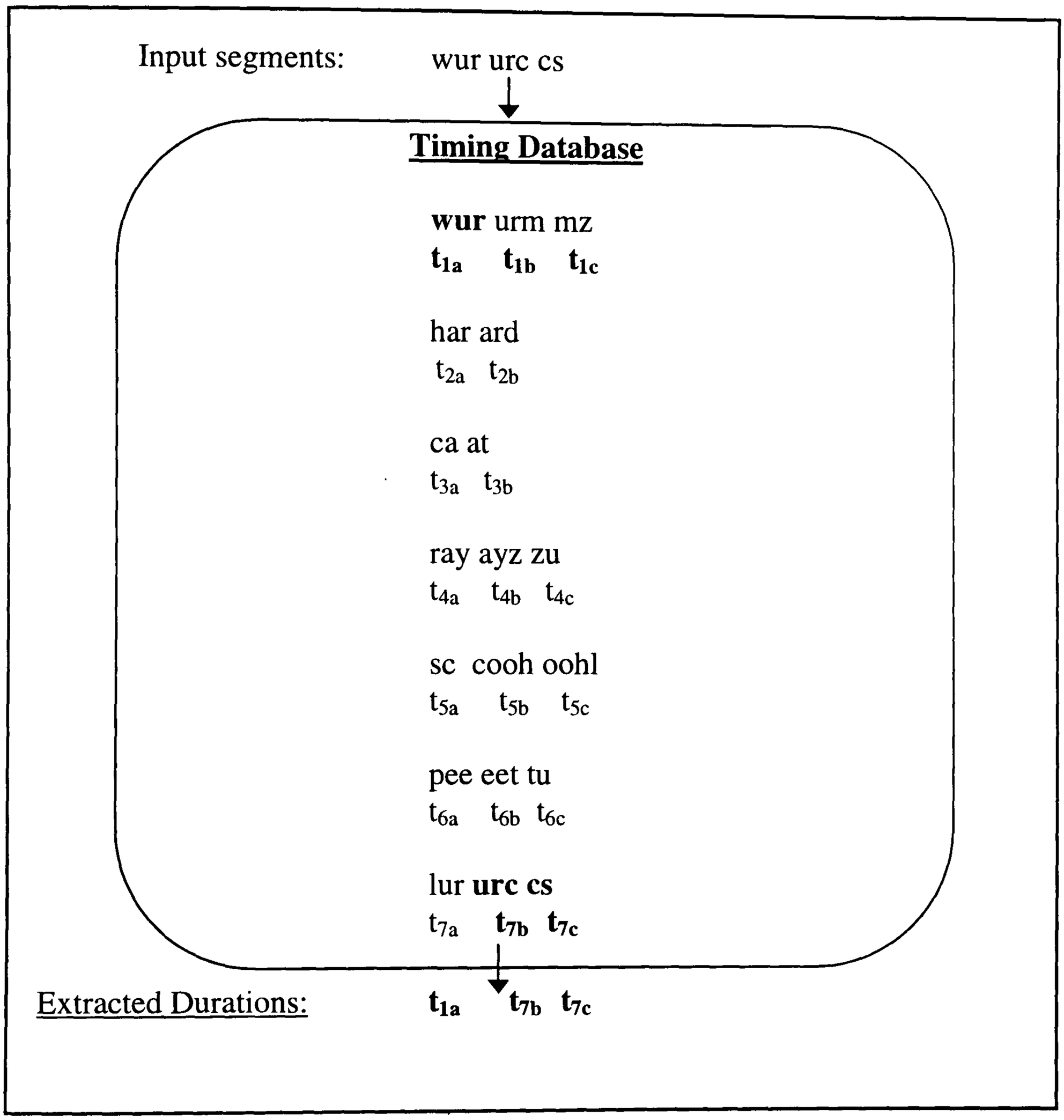


Fig.2 Duration prediction algorithm

The duration prediction algorithm is illustrated in figure 2, which shows a representation of timing database. The database shows the words 'worms', 'hard', 'cat', 'razor', 'school', 'Peter' and 'lurks' with a simplistic, segmental representation, accompanied by their corresponding durations. To predict durations for the input word 'works', the algorithm looks for all instances of the initial segment, 'wur'. This is found in the first database entry, and so the corresponding duration, $\mathbf{t_{1a}}$, is used as the first predicted duration. The final segments ('urc cs') are found together as part of the final database entry, and so their respective durations are added to the output duration list.

It should be noted that the duration prediction algorithm operates independently of the types of segment used, and even allows a mixture of segment types to be stored in the waveform inventory. A range of segment sizes and types can thus be used to synthesize speech, allowing great flexibility. As a result, CV, VC and CVC syllables can coexist in the waveform inventory. This is highly convenient for waveform concatenation synthesis, because its output can be significantly improved through the availability of larger speech segments (Tatham, 1991). The timing database is thus capable of simultaneously storing a mixture of segment types. For example, the word 'connect' can be synthesized using the segments /k@n/, /ne/, /ek/, /kt/. This word would simply be stored in the timing database, along with appropriate segmental duration data.

2.1 Duration Look-up Rules

Rule 1: Largest possible sub-string

The duration prediction algorithm should extract the largest possible sub-strings of durations from the timing database (where 'largest' refers to the number of segments in the string).

This rule takes into account the fact that a segment's duration is affected by its neighbouring ones (Van Santen, 1992). This rule works on the basis that it is more likely that a sub-string taken from one database entry will produce more natural speech than durations taken from a variety of locations. It should be noted that the *largest possible sub-string* rule is similar to Hunt and Black's (1996) approach to the selection of concatenation units from a large speech database. Hunt and Black used a "concatenation cost function", which encouraged the selection of multiple consecutive phonemes from the synthesis database.

Rule 2: Position in word

If several sub-strings of equal length are found, the sub-string closest in position to the input sub-string is selected.

When searching through large duration databases, it is likely that several sub-strings of the same length may be found by Rule 1. In such cases, the 'position in word' rule is used to differentiate between them. This rule is used because a segment's position within a word has an effect on its duration (Oller, 1973).

Rule 3: Length of database entry

If rules 1 and 2 fail to arrive at a single sub-string, the sub-string whose database entry length most closely matches the input segment string's length is chosen.

If the first two rules fail to arrive at a single sub-string, then a 'length of database entry' rule is used, since there remain some sub-strings of identical size and position-in-word. This rule examines the database entries for these remaining sub-strings, and compares

their respective database entry lengths. The entry whose length (in terms of number of segments) most closely matches that of the input segment string is noted. The sub-string of durations corresponding to that database entry is then appended to the output duration list.

2.2 Database size issues

When the timing system is presented with unrestricted text, it is possible that duration data for a particular segment is not available in the desired context. Therefore, the larger the database size, the greater the likelihood of obtaining an appropriate match. If timing data is absent altogether for a particular speech segment, a default duration must be used, which tends to decrease the naturalness of the output speech. It is therefore imperative that the timing database is of sufficient size.

According to Nation (1993), 89% coverage of a written academic text can be achieved with a vocabulary size of 5,000 words. A timing database of 5,000 words would thus cater for a considerable percentage of the likely text inputs. The possible number of unfamiliar input words for which segmental durations must be predicted would be reduced substantially.

It has been found that the smaller the sub-strings of durations used when synthesizing a word, the worse the naturalness of the speech. This is of particular significance for the waveform concatenation method, and is used in a performance measure described in the next section.

3. Evaluating the current system

An experiment was conducted to test the effect of increasing database size on the duration prediction algorithm's output. The experiment monitored the percentage of an input set of test words for which a complete set of durations was produced by the duration prediction algorithm. Complete sets of durations were defined as a 'complete word match'. If no duration could be found for a segment, a default value had to be used, due to the lack of timing data. Ideally, no default values should be used, since they are likely to degrade the intelligibility of the output speech.

In the experiment, a set of 150 words taken randomly from a dictionary was input to the timing system and the number of complete word matches was noted for this set. This was carried for every 100 words added to the timing database. Figure 2 shows the percentage of all input words, for which a complete word match was recorded. The results show that as more words were added to the database, this percentage increased. Figure 3 indicates that a larger database size reduces the number of necessary default durations.

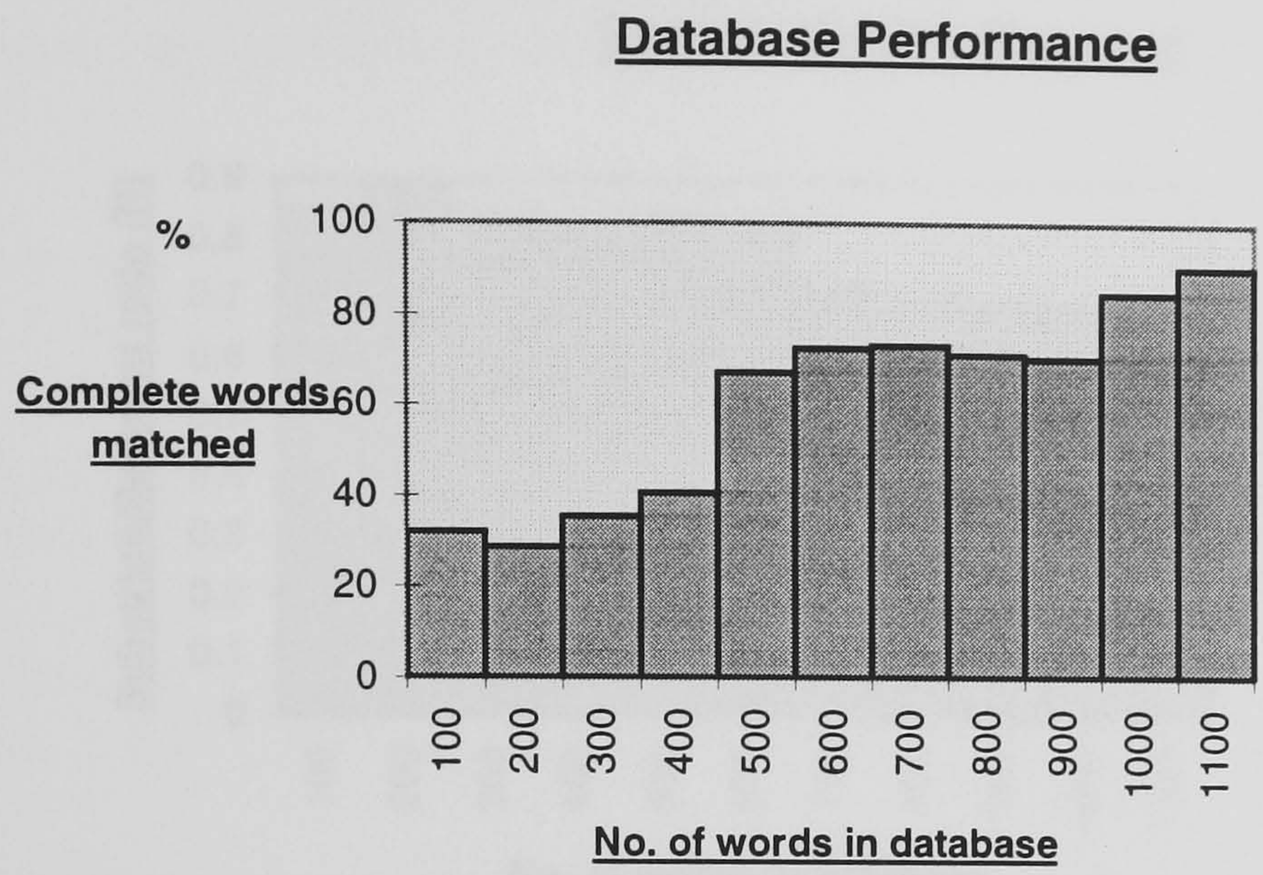


Fig.3: Database performance

The speech output quality for a word can be judged in terms of the number of database entries from which its set of durations is taken (i.e. the number of sub-strings which make up the final set of time values). Thus, the database is of better quality if, for a set number of input words, fewer sub-strings are used, compared to the total number of segments required.

In the above experiment, the number of sub-strings making up the predicted time values for each word was recorded. Thus, for a test set size n, where n=150 words, equation (1) is defined. This ratio R is shown in figure 4, for increasing database sizes.

$$R = \frac{\sum_{i=1}^{n} \text{Number of sub-strings required for test word } W_i}{\sum_{i=1}^{n} \text{Number of segments required for test word } W_i} \qquad \text{(Equation 1)}$$

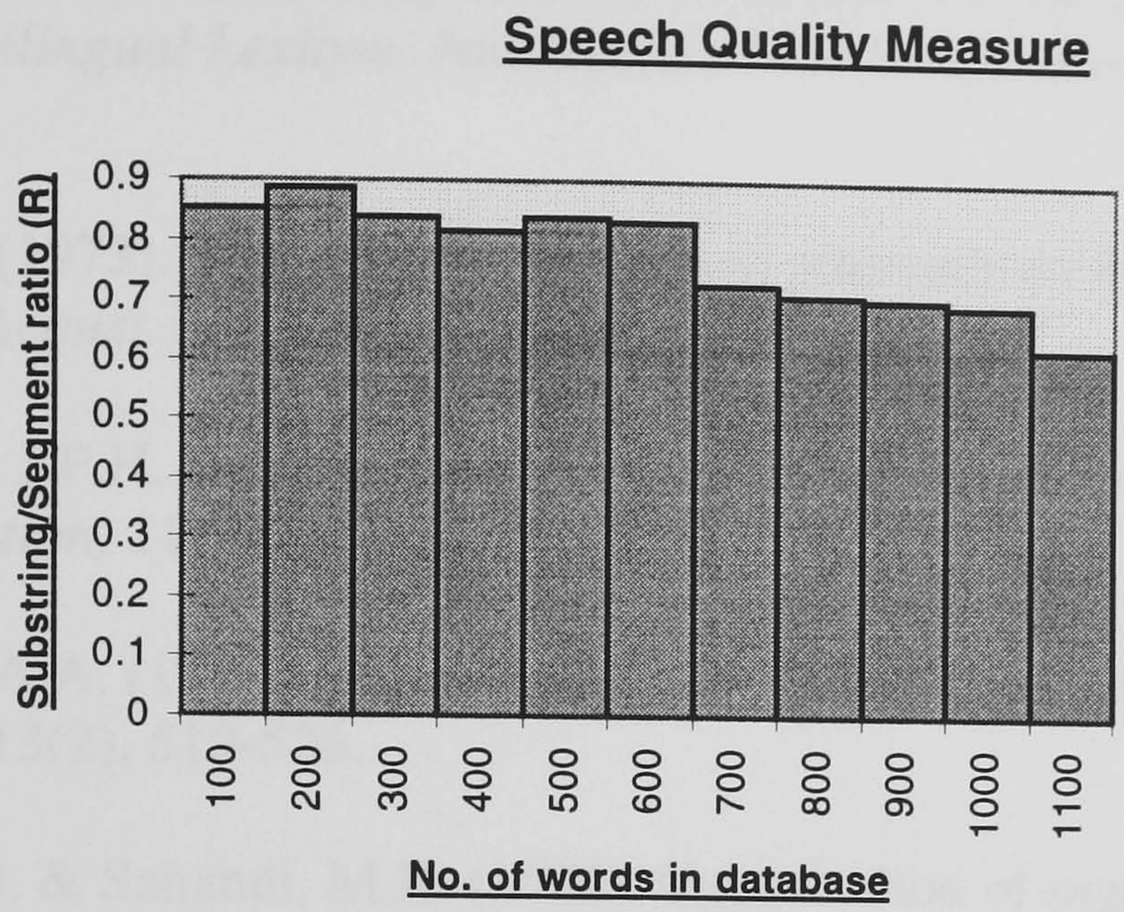


Fig.4: Sub-strings/segments ratio R

Figure 4 shows an overall decrease in the sub-string/segment ratio R, as the database size is increased. This is in accordance with the trend that a larger database allows better duration prediction.

4. Conclusion

A timing system has been discussed, which predicts segmental durations for text-to-speech synthesis via waveform concatenation. The timing system makes use of a timing database and a duration look-up algorithm which is guided by a small set of rules. The timing system is independent of segment type, and requires only a string of phonemes as its input. A quantitative assessment of the timing system was carried out using performance measures relating to database size. The results indicate the need for a larger timing database.

5. References

Breen, A.P. (1992). A comparison of statistical and rule-based methods of determining segmental durations. *Proc. ICSLP 92, Banff.* 1199-1202.

Hunt, A.J. & Black, A.W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In: *Proc. ICASSP-96, Atlanta, GA, 7-10 May.*

Isard, S.D. & Miller D.A. (1986). Diphone synthesis techniques. *In: IEE International Conference on Speech Input/Output; Techniques and Applications. 24-26 March 1986.* London, IEE, Pub. no. 258.

Nation, P. (1993). Vocabulary Size, Growth, and Use. *In:* Schreuder, R. & Weltens, B. (eds.) *The Bilingual Lexicon.* Amsterdam, John Benjamins Publishing Company. 115-134.

Oller, D.K. (1973). The effect of position in utterance on speech segment duration in English. *J. Acoust. Soc. Am.,* 54, 1235-1247.

Van Santen, J.P.H. (1992). Contextual effects on vowel duration. *Speech Communication,* 11, 513-546.

Tatham, M.A.A. (1991). Units of representation for speech synthesis. *Proc. Institute of Acoustics,* 13(2), 519-526.

Vine, D.S.G. & Sahandi, M.R. (1996). Optimisation of segmental time durations in concatenative speech synthesis. *In: KT96 International Conference on Knowledge Transfer, London, UK, 22-26 July.*

Paper to be submitted for Speech and Language Technology (SALT) Club Workshop on Evaluation in Speech and Language Technology, June 17-18, University of Sheffield, UK.

Authors: Dr. Reza Sahandi
Daniel S.G.Vine

Address: School of Design, Engineering & Computing
Bournemouth University
Talbot Campus
Fern Barrow
Poole
Dorset
United Kingdom
BH12 5BB

Tel: (+44) 1202 595258
Fax: (+44) 1202 595314
Email: rsahandi@bournemouth.ac.uk
dvine@bournemouth.ac.uk

A Concatenation Framework for Text-to-Speech Synthesis using CVC Syllables

Abstract

Speech synthesis produced by concatenation of speech waveforms has several associated problems. Speech waveforms are inherently random in nature, and often produce waveform discontinuities when concatenated. A framework for waveform concatenation speech synthesis is described, that produces natural-sounding speech at the word level with minimal discontinuities. The concatenation framework utilises a mixture of waveform types, including consonant-vowel (CV), vowel-consonant (VC) and consonant-vowel-consonant (CVC) syllables. Smoothing at segment boundaries is avoided by outputting only a portion of each segment, specified by a segmental duration parameter. Waveform discontinuities are also minimised by using larger speech segments where possible.

A text-to-segment module converts the input text to a suitable stream of segments for output by selecting the largest speech segments currently available in the waveform inventory. Segmental durations are then predicted for the segment stream. This is achieved using a database of durations for segmented whole words, which has been created manually by synthesizing commonly used words. A duration look-up algorithm

enables the prediction of segmental durations for individual words using a pattern matching approach.

1. Introduction

Speech synthesis produced by concatenation of speech waveforms has several associated problems. Speech waveforms are generally unpredictable in nature, and often produce waveform discontinuities in the form of 'clicks' when simply concatenated (O'Shaughnessy, 1987). Furthermore, speech waveforms require additional processing to manipulate prosodic attributes such as intonation, rhythm and variability (Tatham & Lewis, 1996). For this reason, speech waveforms are often converted to a form that allows some parametric control, such as linear predictive coded (Witten, 1982). Alternatively, concatenation can be assisted via Pitch-Synchronous Overlap-Add techniques (Charpentier & Moulines, 1990), which allow prosodic attributes of the speech to be altered (Edgington *et al.*, 1996).

Different types and sizes of segment can be concatenated, including phonemes, diphones (Isard & Miller, 1986), demisyllables (Dettweiler & Hess, 1985) and syllables (Tatham & Lewis, 1996). In general, using smaller segments has the advantage of requiring a smaller number of stored units; larger-sized segments have greater storage requirements, but can represent coarticulation effects with a lesser degree of processing. Some approaches use a combination of different types of segment (Dettweiler & Hess, 1985). It is common for concatenative techniques to involve minimising a concatenation cost function, which takes into account factors such as potential waveform distortion between consecutive segments (Iwahashi & Sagisaka 1995, Hunt & Black 1996).

In this research, a concatenative speech synthesizer generates speech for individual words using a mixture of consonant-vowel (CV), vowel-consonant (VC) and consonant-vowel-consonant (CVC) syllables. Whilst the units for concatenation are syllables, they are used in a manner which has much in common with diphone and demisyllable speech synthesis. An inventory of syllable waveforms has been digitally recorded by a single, male speaker, at a sampling rate of 44100 Hz.

Speech, when synthesized solely by concatenating entire waveforms selected from an inventory, intelligibility can be seriously degraded. However, this approach can be substantially improved by outputting a specified portion of each selected waveform. The portion of each waveform to be output is specified by a segmental duration parameter (Vine & Sahandi, 1996). This 'portion' refers to a section of the stored segment starting from the onset of the segment t_0, and finishing at any point t_0+s, where s is the specified segmental duration. Thus, depending on s, a smaller or larger portion of a waveform is output. Segmental duration plays a major part in the intelligibility of the resulting synthetic speech, and will be discussed briefly later in the paper.

With this technique, a segmental duration can be specified so that concatenation occurs at an acoustically stable portion of the current waveform. In this way, the current waveform can finish with an amplitude similar to that of the following waveform. This minimises the probability of waveform discontinuities at segment boundaries, which normally cause audible 'clicks'. Thus, by using an optimal choice of segmental timing data alone, smoothing algorithms are dispensed with at concatenation boundaries.

To generate text-to-speech, a timing database of segmental durations is required. A timing database has been developed which includes segmental durations for commonly used English words. For any input word, a duration look-up algorithm uses the timing database to assemble an appropriate set of segmental duration values using a pattern matching approach. This approach involves selecting sequences of appropriate segments from the timing database along with their corresponding time durations, and is described in the following sections.

2. Concatenation Scheme

The units for concatenation are CV, VC and CVC syllables, which are digitally recorded by a single, male speaker. At present, the CVC inventory is still being developed. However, CVC syllables can be made up by concatenating CV and VC syllables. Using CVC segments has the advantage that coarticulation effects rarely extend beyond syllable boundaries (Fujimura & Lovins, 1978). When concatenating segments, the choice of segments making up the output word is important, whilst the duration assigned to each segment is crucial. These aspects of the concatenation scheme are discussed below.

2.1 Formation of CVC syllables

When CV segments are concatenated to VC syllables to form CVC units, it is important to avoid concatenation points within vowels where possible (Saito, 1992), because this is detrimental to the vowel quality. However, whilst this is sometimes unavoidable, better results can be obtained by making a demi-syllabic cut where the consonant-to-vowel transition ends and the vowel portion begins (Bhaskararao, 1994). This allows for coarticulation effects and enables the listener to anticipate the following VC segment. However this can sometimes produce an audible waveform discontinuity; in such cases, only the consonant portion of the CV syllable should be output.

In creating a CV-VC pair, the initial consonant can accept some degree of duration variation. For instance, a CV segment beginning with a fricative or affricate can take a wide duration range without the likelihood of introducing waveform discontinuities. For example, the words 'shoot' or 'jig' allow considerable variation in their initial 'sh' and 'j' portions respectively. To some extent, CV-VC pairs starting with sustained sounds such as liquids can also vary in duration without compromising the speech output. Finally, even unsustained sounds such as unvoiced plosives can tolerate some duration variation. For example, in synthesizing the word 'ton', the initial /t/ can have any duration value between 45 and 348 ms.

2.2 Concatenation Nodes

When CV and VC syllables are concatenated, there is a risk that a waveform discontinuity will result. This problem can be reduced by choosing specific concatenation points in the CV segment's waveform. These points will be referred to as nodes. Figure 1

shows a series of concatenation nodes for the CV segment /m{/, which is recorded in stereo. These are labelled C_n, where n is an integer between 1 and 5. The nodes indicate the instants in the 'm' allophone where the signal crosses the time axis shown. The corresponding segmental durations are shown by the arrows.

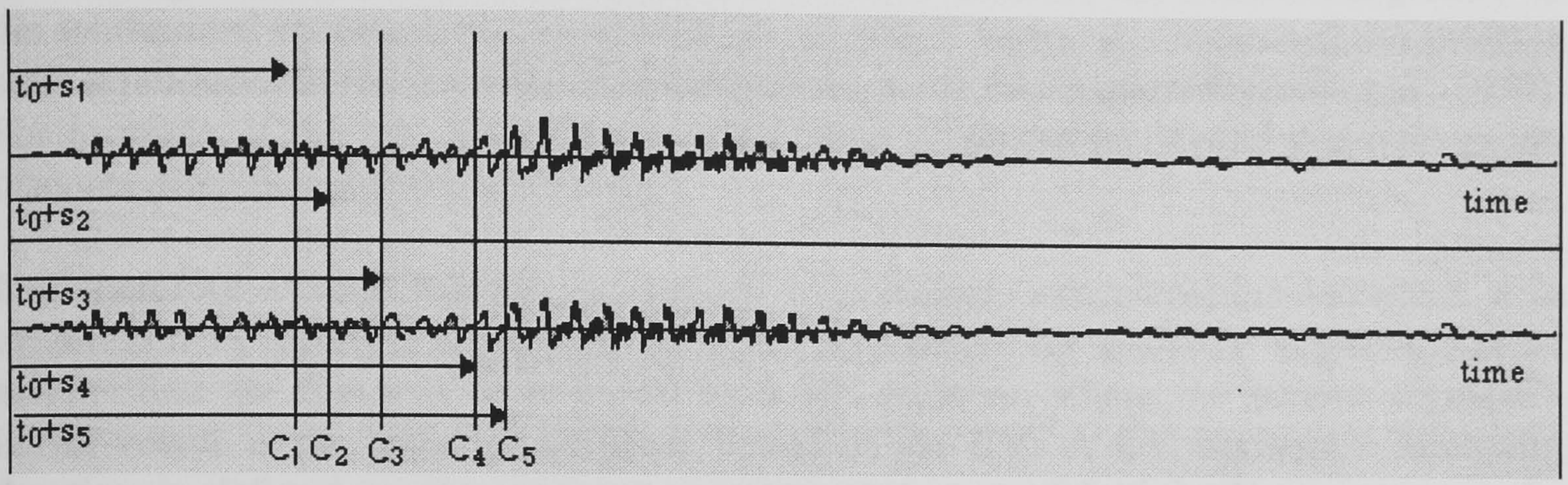


Fig.1: Concatenation nodes for the syllable /m{/

An experiment was carried out to verify the efficacy of the concatenation nodes, by synthesizing different CVC words which share the same initial CVC segment. For instance, the concatenation nodes for the /m{/ segment were established by visual inspection of the waveforms in the time domain. Initially they were tested by synthesizing the word 'mat', using a /m{/ segment followed by an entire /{t/ segment. The efficacy of using the same concatenation nodes for /m{/ was then tested by synthesizing other CVC syllables. These consisted of an initial /m{/ segment, followed by various VC syllables starting with /{/, e.g. /{n/ or /{d/. It was found that for the concatenation nodes shown, minimal waveform discontinuities were produced compared to CVC segments synthesized using random durations for the initial /m{/ syllable.

2.3 Speech Segment Omission

It is sometimes necessary to omit segments from the output stream, in order to reduce the number of concatenation points required and to improve the overall fluency of the output speech. For example, the word 'dogmatic' can be represented initially by the following CV and VC symbols:

dQ Qg gm m{ {t tI Ik

In the above segmental representation there is a trade-off between coarticulation and rhythm. For instance, whilst the /gm/ segment provides a transition from the /g/ to the /m/ sound, it could have the effect of reducing the overall speaking rate for the synthesized word. This would be detrimental to the word's overall rhythm, making it sound 'laboured'. The /gm/ segment should therefore be omitted from the final utterance. Such an omission would not degrade the overall smoothness of the word too dramatically, since the remaining segments correspond well with syllable boundaries.

2.4 Phonemic Category Shifts

When concatenating a CV segment followed by a VC, the CV segment's duration is highly important. Apart from the challenges of creating realistic coarticulation and minimising waveform discontinuities, one must also consider the basic intelligibility of the synthesized speech. If the duration of an initial CV segment is shortened too much, a shift is produced in the phonemic category perceived by the listener (Ferrero *et al.*, 1979). For example, if the 'sh' sound in the word 'ship' is shortened beyond a certain point, listeners often perceive the word 'chip'.

It is therefore evident that for the method of waveform concatenation described, it is meaningful to record a 'minimum duration' permissible for each CV segment. Let us assume that the CV will be followed by a VC segment, where the second segment's initial vowel is the same as the final vowel of the CV. A CV segment's minimum duration is defined as *the minimum duration that a word-initial CV segment can be assigned without listeners experiencing a perceptual category shift.* This definition is related in concept to Klatt's (1976) 'minimum duration' term. However, Klatt's concepts of a minimum and maximum duration for a phoneme apply to a different synthesis method in which segments are 'squashed' or 'stretched' to fit a specified duration. This concatenative scheme does not require a maximum duration for segments, since its value depends upon the actual duration of the recorded waveform.

3. Applying the Concatenation Framework to Text-to-Speech

3.1 Timing Database Creation

To automate the concatenation scheme, a segmental duration module has been designed, which produces segmental durations using a timing database (Vine & Sahandi, 1996). The database consists of segmental duration details for commonly used English words. Each word is synthesized by concatenating the segments according to the concatenation scheme.

3.2 Text-to-Segment Conversion Module

The speech synthesizer uses CV, VC and CVC syllables to produce speech for an input text. The text must therefore be converted to a stream of symbols which correspond to CV, VC and CVC syllables. This is achieved by a text-to-segment conversion module (see figure 2). The module contains a text-to-phoneme conversion system, which breaks up the input text into a stream of phoneme symbols, using a set of letter-to-sound rules. Since the phoneme symbols produced may not correlate with the stored segments in the inventory, the phonemes are thus converted to CV and VC syllables by grouping them into pairs. For example, the word 'cats', is converted from the phoneme stream to CV and VC segments as shown:

k { t s => k{ {t ts

Speech output can be improved by using larger speech segments (Tatham, 1991). This reduces the number of segment boundaries per word and hence the likelihood of waveform discontinuities. The text-to-segment conversion module ensures that for any given text, the largest possible speech segments are selected from the waveform inventory. Where possible, CV-VC syllable pairs are replaced by a single CVC syllable, when available in the waveform inventory. The text-to-segment conversion module relies upon a Segment Equivalency Table, which contains equivalencies between CV-VC pairs and CVC syllables that are stored in the waveform inventory. For each consecutive pair of syllables in the stream of segments for output, the text-to-segment module searches the Segment Equivalency Table for an appropriate CVC syllable. For example, for the word 'cats', the /k{/, /{t/ pair could be replaced by a single segment /k{t/, if it is contained in the Segment Equivalency Table. This reduces the number of segment boundaries for a word.

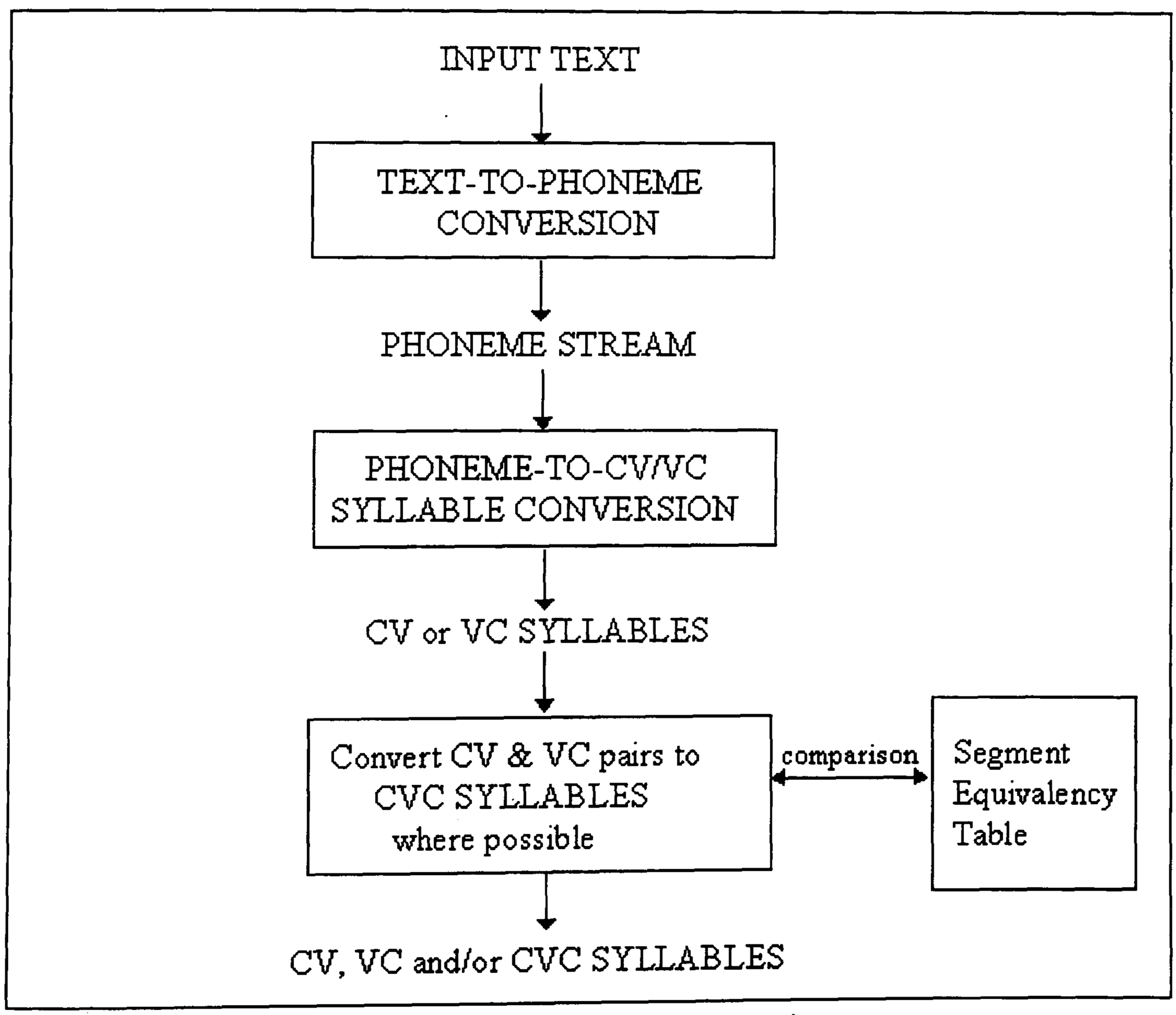


Fig.2: Text-to-segment conversion

The text-to-segment conversion module at present deals with CV, VC and CVC syllables, and is potentially independent of segment type. Currently, the CVC inventory is not exhaustive, but is continuously being developed. Whenever additional speech segments are added to the waveform inventory, the Segment Equivalency Table is also updated

with appropriate equivalencies. This allows the speech waveform inventory to contain a range of segment types.

3.3 Segmental Duration Module

The segmental duration module assigns a time value to each speech segment for output, using a timing database and a duration look-up algorithm. The main principle is to extract sequences of durations from entries in the timing database for the relevant segments of the input word. The first rule is to take the largest unbroken sequences of consecutive segments possible from the timing database, along with their associated durations. If several segment sequences of equal length are found, then their respective 'position in word' attributes are taken into account. If each segment sequence has the same position in word, then a final rule is used which favours the sequence whose original database entry is closest in length to the input word's length in segments.

At present, the segment omission principle which was discussed earlier, is included inherently in the database. So when a word such as 'instigated' is included in the database, any segments omitted are simply assigned a duration of zero. In this example, this word would initially be segmented as follows:

In ns st tI Ig geI eIt tI Id

Experimental results show that superior speech is output if the /Ig/ segment is omitted altogether. So when this word is stored in the timing database, /Ig/ is assigned a duration of zero, effectively omitting it. When a novel word is being synthesized, this segment omission is transparent to the duration look-up algorithm. For instance, the input word 'unmitigated' would be segmented as follows:

Vn nm mI It tI Ig geI eIt tI Id

The duration look-up algorithm searches the timing database for the largest unbroken sequences of segments possible. It is likely that the last five segments for the word 'unmitigated' would be extracted from the timing database entry for the word 'instigated' (the first five segments would be compiled from other database entries). Since the /Ig/ segment was originally assigned a duration of zero, /Ig/ is also omitted from the word 'unmitigated' during the speech synthesis. Whilst at present, segment omission is 'built in' to the timing database, it would be possible to implement this process using a set of segment omission rules. Further study is needed to define the contexts in which segment omissions should occur in the text-to-segment conversion module.

4. Results

The concatenation scheme described produces natural-sounding speech synthesized for whole words. On the sentence level, whilst no prosodic manipulation such as pitch alteration is possible, the speech is nonetheless intelligible. This is no doubt due to the

listener being assisted by the presence of surrounding words to give contextual clues.

5. Conclusion

A concatenation scheme based on CV, VC and CVC syllables for waveform speech synthesis has been described. The scheme is used in building a database of timing data for segmented whole words, which is later used to predict segmental durations for text-to-speech. A text-to-segment module automatically controls the translation of text to an appropriate combination of segments, attempting to make optimal use of the available waveforms in the inventory. Segmental durations are assigned automatically via a duration look-up algorithm which makes use of the timing database. Once the segment stream is chosen, the waveforms are output according to the concatenation scheme described. The text-to-segment module requires further work with respect to implementing an omission scheme for segments at syllable boundaries.

6. References

Bhaskararao, P. (1994). Subphonemic Segment Inventories for Concatenative Speech Synthesis. *In:* Keller, E. (editor) *Fundamentals of Speech Synthesis and Speech Recognition.* Chichester, John Wiley & Sons. 69-85.

Charpentier, F. & Moulines, E. (1990). Pitch Synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication,* 9, 453-467.

Dettweiler, H. & Heiss, W. (1985). Concatenation rules for demisyllable speech synthesis. *Acustica,* 57, 268-283.

Edgington, M., Lowry, A., Jackson, P., Breen, A.P. & Minnis, S. (1996). Overview of current text-to-speech techniques: Part 2 - prosody and speech generation. *BT Technology Journal*, 14(1), 84-99.

Ferrero, F.E., Pelamatti, G.M. & Vagges, K. (1979). Perceptual Category Shift of Voiceless Italian Fricatives as a Function of Duration Shortening. *In:* Lindblom, B. & Ohman, S. (eds.). *Frontiers of Speech Communication Research.* London, Academic Press. 159-165.

Fujimura, O. & Lovins, J. (1978). Syllables as concatenative phonetic units. *In:* Bell, A. & Hooper, J. (eds.) *Syllables and Segments.* Amsterdam, North-Holland. 107-120.

Hunt, A.J. & Black, A.W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In: *Proc. ICASSP-96, Atlanta, GA, 7-10 May 1996.*

Isard, S.D. & Miller D.A. (1986). Diphone synthesis techniques. *In: IEE International Conference on Speech Input/Output; Techniques and Applications. 24-26 March 1986.* London, IEE, Pub. no. 258.

Iwahashi, N. & Sagisaka, Y. (1995). Speech segment network approach for optimization of synthesis unit set. *Computer Speech and Language,* 9, 335-352.

Klatt, D.H. (1976). Linguistic uses of segmental duration in English: Acoustic and perceptual evidence. *J. Acoust. Soc. Am.,* 59, 1208-1221.

O'Shaughnessy, D. (1987). *Speech Communication - Human and Machine.* Wokingham, Addison-Wesley Publishing Company.

Saito, S. (editor) (1992). *Speech science & technology.* Tokyo, Ohmsha.

Tatham, M.A.A. (1991). Units of representation for speech synthesis. *Proc. Institute of Acoustics,* 13(2), 519-526.

Tatham M.A.A. & Lewis, E. (1996). Improving text-to-speech synthesis. *In:* R. Lawrence (editor) *Proceedings of the Institute of Acoustics,* 18(9), 35-42.

Vine, D.S.G. & Sahandi, M.R. (1996). Optimisation of segmental time durations in concatenative speech synthesis. In: *KT96 International Conference on Knowledge Transfer, London, UK, 22-26 July 1996.*

Witten, I.H. (1982). *Principles of Computer Speech.* London, Academic Press.

AN EVALUATION OF NATURALNESS IN TIME-DOMAIN CONCATENATIVE SPEECH SYNTHESIS

Daniel S G Vine & Reza Sahandi
SPECTA Group, Bournemouth University, Fern Barrow, Poole, BH12 5BB, UK.
Email: {dvine|rsahandi}@bournemouth.ac.uk

ABSTRACT

The potential of a time-domain concatenative speech synthesis (TDCSS) method, based on joining portions of CV, VC, CVC and CC waveforms, is evaluated. In TDCSS, a segmental duration parameter specifies a portion of each stored waveform, to be concatenated. This paper compares the output of TDCSS with five established text-to-speech synthesisers, ranking a set of stimuli according to naturalness.

1. INTRODUCTION

A time-domain concatenative speech synthesis (TDCSS) technique for text-to-speech (TTS) has been developed at Bournemouth University. Speech is synthesised using an inventory of stored CV, VC, CC and CVC waveforms. To synthesise a word, appropriate waveforms are selected from the inventory and a segmental duration value is assigned to each waveform. The segmental duration parameter specifies the portion to be selected from each waveform for concatenation (see Figure 1). Each portion is taken from the start of a stored waveform, and finishes at a point t milliseconds later, where t is the segmental duration. In effect, the segmental duration variable determines the number of speech samples that are extracted from a stored waveform. In the concatenation process, the speech segments are abutted, without either an overlap or a gap between segments. In TDCSS, the segmental duration parameter actually controls both the speech timing and smoothness of the join between acoustic segments. Accurate segmental durations are essential if the resulting speech is to sound natural, since the duration value effectively determines the choice of concatenation point and hence, the quality of the synthetic speech.

In TDCSS, the speech portions for concatenation are not stretched or squashed in any way. Rather, the intrinsic duration of each extracted speech segment is used. With this method, overall word duration can be adjusted, by varying segmental durations. Furthermore, the number of 'clicks' in the synthetic speech can by reduced by choosing concatenation points that correspond to appropriate zero crossings in the time domain. Vowel quality can be preserved by avoiding concatenation points within vowels where possible (Saito, 1992). For instance, when using CV and VC segments, to form CVC syllables, vowel onset time can be controlled, by varying the portion of CV segment to be concatenated. Good results can usually be obtained using a demi-syllabic cut, where the consonant-to-vowel transition ends and the vowel portion begins. This allows for coarticulation effects and enables the listener to anticipate the following VC segment. A more detailed explanation can be found in Sahandi and Vine (1997).

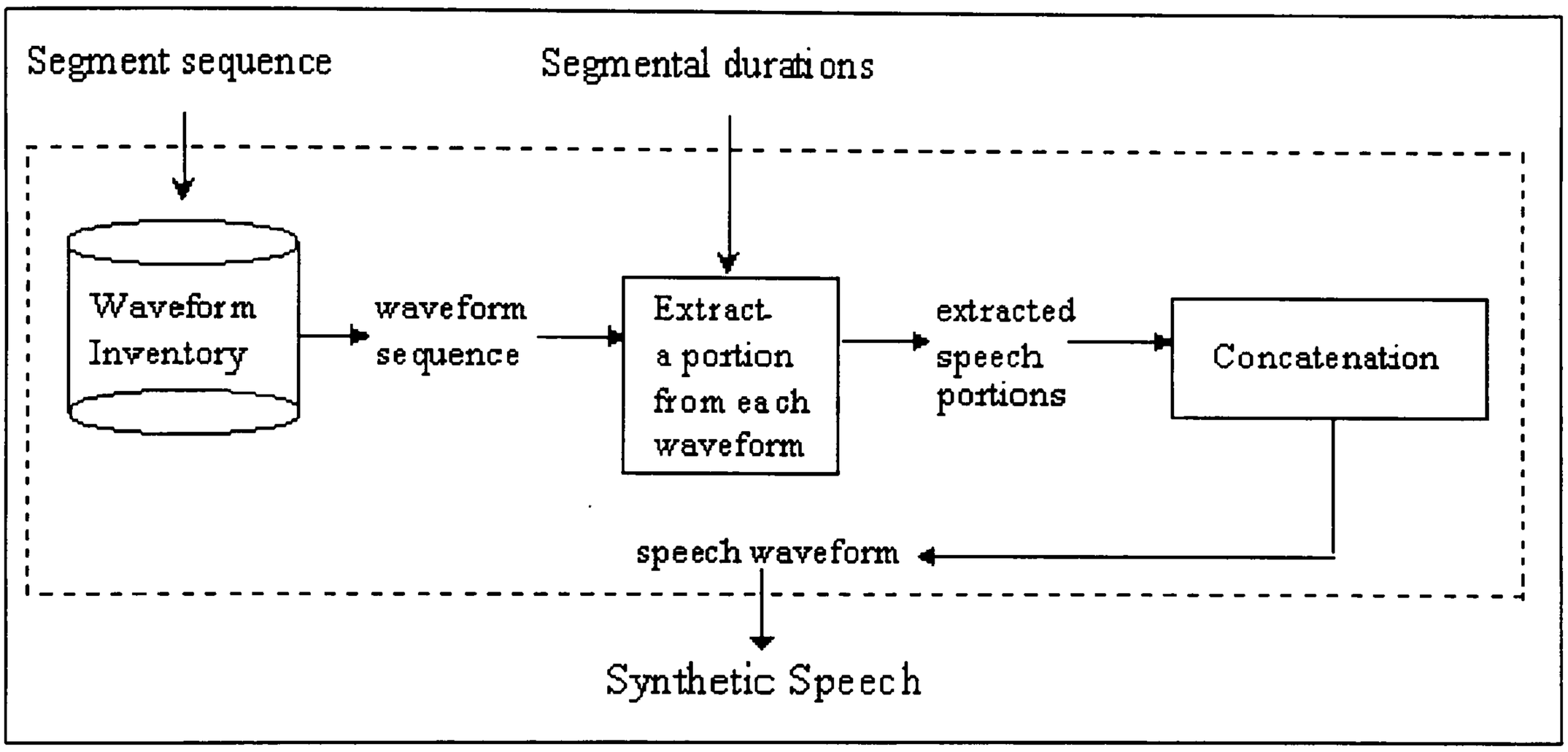


Figure 1: An overview of TDCSS

TDCSS reduces the need for smoothing at segmental boundaries, by attempting to choose optimal groups of speech segments. Some concatenation systems select an optimal set of segments from a segment inventory (e.g. Iwahashi & Sagisaka, 1995), on the basis of minimising a concatenation cost function (based on measures such as spectral distance). In contrast, for TDCSS, concatenation points are selected via the segmental duration parameter. For TTS, a statistical segmental duration prediction system is used, based on a database of segmental duration information. It should be noted that the segmental duration database was compiled by an 'expert judge', who listened to the synthetic speech and adjusted the timing according to their perception of speech quality.

In this paper, the potential of the TDCSS synthesis method is evaluated. This is an exploratory investigation, to examine the output achievable using TDCSS, before fully developing an entire TTS synthesis system. For optimal naturalness, hand-refined segmental durations were chosen for the TDCSS stimuli. This study assumes that appropriate segmental durations will be predicted via a statistical segmental timing algorithm (see Vine & Sahandi, 1997). In future, TDCSS will be applied to TTS, and may be combined with techniques, such as TD-PSOLA (Moulines & Charpentier, 1990), for pitch manipulation. In a preliminary listening test, the intelligibility of TDCSS was also evaluated. Four participants listened to ten synthetic words of between one and five syllables. All subjects correctly identified the ten words.

1.1 EVALUATION TECHNIQUES

The assessment of TTS systems is a necessary task, since there are "no physical measurements which result in indices that quantify perceptual attributes of synthesised speech" (Pavlovic, Rossi & Espesser, 1990). Anyone who investigates synthetic speech for any great period of time finds it increasingly

difficult to judge its naturalness effectively (Dixon & Maxey, 1968). It is also necessary to compare rival commercial speech synthesisers. Researchers need to study which aspects of a synthesis technique improve the overall speech (Morton, 1991), and yet the task of assessing synthetic speech is complicated by many interacting factors. Since there are few objective procedures for assessing synthetic speech, subjective assessment techniques are often favoured (Arden, 1997).

Some aspects of speech can be measured quantitatively, such as intelligibility and comprehension. Quantitative speech quality measures such as *Reference Distortion* techniques have been described (Johnston, 1996). Other attributes can only be measured qualitatively, including ease of listening, ease of comprehension and naturalness. Although intelligibility and naturalness are desirable, they remain ill-defined. Dutoit (1997) has defined intelligibility as being linked to how well the phoneme-to-phoneme transitions are realised, whilst naturalness is a result of how well the segmental features harmonise with the supra-segmental features. A further term, *acceptability* has been defined by Pavlovic, Rossi and Espesser (1990) as being "the overall users' satisfaction with the communication situation". Whilst this criterion is useful, the perception of acceptability may be relative to the user's expected application for the speech. In concatenative speech synthesis, it is important that discontinuities in the form of clicks are eliminated when conjoining speech segments (Hess 1992, Kraft 1993), since they may affect both intelligibility and naturalness. This is particularly the case for TDCSS, so in this study, naturalness is defined as "human-sounding" (e.g. Childers & Lee, 1991).

Intelligibility can be investigated in a reasonably straightforward manner, using segmental intelligibility tests. Segmental intelligibility tests examine intelligibility at the phoneme level (Carlson *et al.,* 1992) and include the Modified Rhyme Test, Diagnostic Rhyme Test (Voiers, 1983) and Cluster Identification Test (Jekosch & Pols, 1994). The *Modified Rhyme Test* involves the presentation of isolated, monosyllabic words to subjects (Logan, Greene & Pisoni, 1989), and can be used with untrained listeners. In the *Diagnostic Rhyme Test*, listeners are presented with an isolated word taken from a 192-word vocabulary, and must identify it in a pair of words (which differ only by their initial consonant). *Cluster Identification Tests* involve the presentation of nonsense words within short carrier phrases to subjects, who then attempt to identify the word (Kraft & Portele, 1995). The nonsense words usually have a given syllable structure, such as C_iVC_f (where *i* and *f* indicate the number of initial and final consonants in the cluster). Each stimulus is only presented once, and subjects are given the overall list of words to be played. Responses are then scored according to correctness. Cluster Identification Tests are more inclusive and reproducible than Diagnostic Rhyme Tests, requiring fewer trained listeners (usually four is enough).

Sentence-level intelligibility tests include the Semantically Unpredictable Sentences (SUS) test. This test minimises the contextual semantic information given in sentences by using raw sentence structures filled by a random selection of words taken from predefined lists of possible candidates (Benoît & Pols, 1992). An example sentence is *"The table walked through the blue truth"* (Dutoit, 1997). One approach for evaluating synthetic speech is to use *single pair comparison tests.* These operate on the word or sentence level, and are used to determine the ranking of systems. According to Johnston (1996), these tests can either focus on specific attributes (e.g. loudness, distortion, etc.) or allow an open response (e.g. "Which

do you prefer?"). In this paper, a single pair comparison test was used to rank the TDCSS synthesiser against five established TTS synthesisers, in terms of perceived naturalness.

2. INVESTIGATING NATURALNESS IN TDCSS

In this investigation, the potential of the TDCSS method was assessed by comparing its output with that of five established TTS synthesisers. In the experiment, hand-refined TDCSS stimuli were used, in order to explore the potential naturalness of unsmoothed TDCSS stimuli, given the availability of appropriate segmental durations. It is also intended in future to investigate a smoothed version of TDCSS.

2.1 STIMULI

Six different speech synthesisers were used to produce the stimuli for comparison. These include TDCSS, KTH, Laureate, Festival, Bell Labs and TruVoice. KTH refers to the interactive multilingual TTS service of KTH in Stockholm (KTH, 1998). Laureate refers to BT's TTS synthesiser (Edgington *et al.* 1996, Laureate 1998). Festival refers to Edinburgh University's Festival TTS synthesiser, version 1.2.1 (Festival, 1997). Bell Labs and TruVoice refer to TTS synthesisers produced by AT&T and Centigram respectively (Bell Labs 1998, TruVoice 1998). Half of these speech synthesisers are for American English and half are for a British English accent. All stimuli were synthesised using the above TTS synthesisers' default settings. Male voices were chosen, to be comparable with the TDCSS stimuli.

The stimuli consisted of isolated, single words of between one and five syllables (see Table 1). Isolated words were chosen in order to reduce the contribution of prosodic elements, such as intonation, to the subjects' perception of naturalness.

Still	Computer	Hurts	Concatenation
Definitely	State	Talking	-

Table 1: Words used as synthetic stimuli

For the TDCSS stimuli, an initial phonetic transcription was produced by hand for each word. This was converted to an appropriate sequence of CC, CV and VC speech units. The syllables were taken from a waveform inventory (Sahandi & Vine, 1997) recorded by a male speaker, with a 16-bit depth, and a sampling rate of 44100 Hz. A variable portion of each syllable was selected for concatenation, following the scheme outlined in Section 1. Segmental durations were initially produced using a data-driven segmental duration prediction algorithm (Vine & Sahandi, 1996). They were then hand-refined where necessary according to the author's judgement, to increase the naturalness of the final synthetic speech.

2.2 SUBJECTS

Fifteen subjects were used, aged between 21 and 50. Nine subjects were male and six were female. Nine subjects were native speakers of English whilst six were not. All subjects had normal hearing, and did not have extensive experience of speech synthesis. Experienced listeners were avoided, because synthetic speech tends to sound more natural as listening experience increases (Nusbaum, Francis & Henly, 1995). Subjects were told that the purpose of the experiment was to compare the naturalness of a number of different speech synthesisers.

2.3 METHOD

In a pair comparison experiment, six speech synthesisers were ranked in terms of perceived naturalness. The experiment was carried out in a quiet room and conducted using an interactive computer program. Seven different words were produced by each synthesiser, and the total set of synthetic words were arranged in such a way that every possible combination of stimuli was used. The pairs of stimuli were presented in a randomised order, so as to minimise any possible boredom that may arise. After each pair was presented, the subject indicated which of the stimuli they thought was "the most natural sounding". A 'Repeat' button was provided. Subjects were provided with a list of the words that they would encounter, in order to reduce the cognitive load associated with word identification, and to enable them to concentrate purely on judging naturalness.

2.4 RESULTS

Figure 2 shows the overall rankings for the speech synthesisers in terms of average number of times preferred per subject. When ranked against five established TTS synthesisers, the TDCSS stimuli were judged to be the fifth most natural out of six. It was noted that several subjects favoured a British speech accent to an American one. This is perhaps unsurprising, since none of the subjects were American. It was often the case that shorter TDCSS stimuli, such as monosyllables and disyllables, tended to be judged more favourably. It should be pointed out that only CV, VC and CC speech segments were used to synthesise the TDCSS stimuli; further improvements are attainable if CVC syllables are introduced.

The speech synthesisers used were based on a variety of different synthesis methods, including TD-PSOLA (Laureate), LPC (Bell Labs) and formant synthesis (KTH). The overall results follow the general trend that concatenative synthesisers (particularly the TD-PSOLA method) are perceived as being more natural (Klaus, Fellbaum & Sotscheck, 1997). It is also interesting that a relatively unsophisticated synthesis method, such as TDCSS, was perceived as slightly more natural than an established formant TTS synthesiser (KTH).

It is important to consider that in this experiment, only a small set of isolated words was used for the stimuli. It is likely that further differences between the synthesisers would emerge, in terms of predicted intonation at the sentence level. Indeed, the evaluation of prosodic features remains a difficult task, and it is not fully clear as to how synthetic speech should be assessed on the sentence level (Morton, 1991).

This is because the various components of a TTS synthesiser can interact with the prosodic component. A potential solution to this problem is to reduce the experimental stimuli to their prosodic content, such as fundamental frequency contour, intensity or rhythm (Sonntag & Portele, 1997).

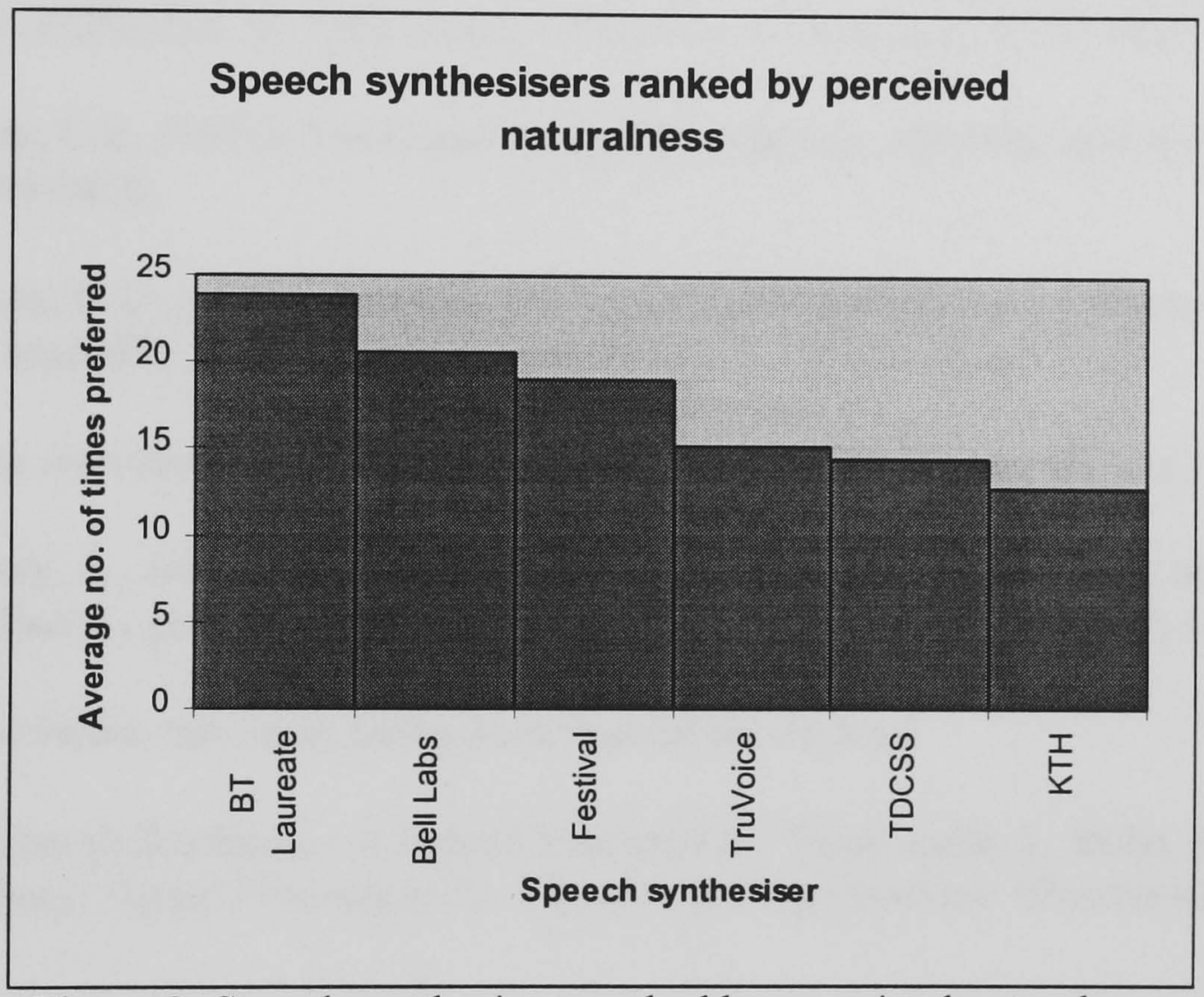


Figure 2: Speech synthesisers ranked by perceived naturalness

3. CONCLUSION

The naturalness of speech produced using TDCSS has been evaluated, relative to five TTS synthesisers. In a single pair comparison experiment, TDCSS stimuli were ranked against speech produced by five established TTS synthesisers. TDCSS was ranked fifth most natural out of the six synthesisers. If accurate segmental durations can be predicted, and appropriate smoothing techniques used, the TDCSS speech synthesis method can potentially be adapted to practical applications.

4. REFERENCES

Arden, P. (1997). Subjective assessment methods for text-to-speech synthesis. *In: Proc. SALT Workshop on Evaluation in Speech and Language Technology, University of Sheffield, UK, June 17-18.* 9-16.

Bell Labs (1998). *http://www.research.att.com/cgi-bin/cgiwrap/mjm/voices.cgi*

Benoît, C. & Pols, L.C.W. (1992). On the assessment of synthetic speech. *In:* Bailly, G., Benoît, C. & Sawallis, T.R. (eds.) *Talking Machines - Theories, Models, and Designs,* London, Elsevier Science Publishers. 435-441.

Carlson, R., Granström, B., Neovius, L. & Nord, L. (1992). The "listening speed" paradigm for synthesis evaluation. *In: Proc. FONETIK'92, Sixth Swedish Phonetics Conference, 20-22 May 1992.*

Childers, D.G. & Lee, C.K. (1991). Vocal quality factors: Analysis, synthesis and perception. *J. Acoust. Soc. Am.,* 90(5), 2394-2410.

Dixon, N.R. & Maxey, H.D. (1968). Terminal analog synthesis of continuous speech using the diphone method of segment assembly. *IEEE Trans.,* AU-16, 40-50.

Dutoit, T. (1997). *An Introduction to Text-to-Speech Synthesis.* Netherlands, Kluwer Academic.

Edgington, M., Lowry, A., Jackson, P., Breen, A.P. & Minnis, S. (1996). Overview of current text-to-speech techniques: Part 2 - prosody and speech generation. *BT Technology Journal,* 14(1), 84-99.

Festival (1997). *http://www.cstr.ed.ac.uk/projects/festival/userin.html*

Hess, W.J. (1992). Speech Synthesis - A Solved Problem? *In:* Vandewalle, J., Boite, R., Moonen, M. & Oosterlinck, A. (editors) *Signal Processing VI: Theories and Applications.* Elsevier Science Publishers B.V.

Iwahashi, N. & Sagisaka, Y. (1995). Speech segment network approach for optimization of synthesis unit set. *Computer Speech and Language,* 9, 335-352.

Jekosch, U. & Pols, L.C.W. (1994). A Feature-Profile For Application-Specific Speech Synthesis Assessment And Evaluation. *In: Proc. Int. Conf. on Acoustics, Speech & Signal Processing.* Vol. 3, 1319-1322.

Johnston, R.D. (1996). Beyond intelligibility - the performance of text-to-speech synthesisers. *BT Technol. J.,* 14(1), 100-111.

Klaus, H., Fellbaum, K. & Sotscheck, J. (1997). Subjective evaluation and comparison of the speech quality of text-to-speech systems for the German language. *Acustica,* 83(1), 124-136.

Kraft, V. (1993). Auditory Detection of Discontinuities in Synthesis-By-Concatenation. *In: Proc. Eurospeech '93.* 929-932.

Kraft, V. & Portele, T. (1995). Quality Evaluation of Five German Speech Synthesis Systems. *Acta Acustica,* 3, 351-365.

KTH (1998). *http://www.speech.kth.se/cgi-bin/SayTrans*

Laureate (1998). *http://innovate.bt.com/showcase/laureate/demo/body.htm*

Logan, J.S., Greene, B.G. & Pisoni, D.B. (1989). Segmental intelligibility of synthetic speech produced by rule. *J. Acoust. Soc. Am.,* 86(2), 566-581.

Morton, K. (1991). Expectations for assessment techniques applied to speech synthesis. *Proc. Institute of Acoustics,* 13(2), 501-508.

Moulines, E. & Charpentier, F.(1990). Pitch Synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication,* 9, 453-467.

Nusbaum, H.C., Francis, A.L. & Henly, A.S. (1995). Measuring the Naturalness of Synthetic Speech. *Int. J. Speech Technology,* 1, 7-19.

Pavlovic, C.V., Rossi, M. & Espesser, R. (1990). Use of the magnitude estimation technique for assessing the performance of text-to-speech synthesis systems. *J. Acoust. Soc. Am.,* 87(1), 373-382.

Sahandi, R. & Vine, D.S.G. (1997). A Concatenation Framework for Text-to-Speech Synthesis using CVC Syllables. *In: Proc. SALT Workshop on Evaluation in Speech and Language Technology, University of Sheffield, UK, June 17-18.*

Saito, S. (editor) (1992). *Speech science & technology.* Tokyo, Ohmsha.

Sonntag, G.P. & Portele, T. (1997). A method for prosody evaluation. *In: Proc. SALT Workshop on Evaluation in Speech and Language Technology, University of Sheffield, UK, June 17-18.* 188-194.

TruVoice (1998). *http://www.westcler.k12.oh.us/johnny/truvoice.html*

Vine, D.S.G. & Sahandi, M.R. (1996). Optimisation of segmental time durations in concatenative speech synthesis. *In: Proc. KT96 International Conference on Knowledge Transfer, London, UK, 22-26 July.*

Vine, D.S.G. & Sahandi, M.R. (1997). Timing Databases for Concatenative Speech Synthesis. *In: DSP Scandinavia 97, Stockholm, Sweden, 3-4 June.*

Voiers, W.D. (1983). Evaluating Processed Speech Using the Diagnostic Rhyme Test. *Speech Technology,* January-February, 30-39.