A DIGITAL NEURAL NETWORK APPROACH
TO
SPEECH RECOGNITION

By

NAJMI GHANI HAIDER

A Thesis submitted for the Degree of Doctor of Philosophy
in the Department of Electrical Engineering and Electronics
at Brunel University, Uxbridge, Middlesex, March 1989.

ABSTRACT

This thesis presents two novel methods for isolated word speech recognition based on sub-word components. A digital neural network is the fundamental processing strategy in both methods.

The first design is based on the 'Separate Segmentation & Labelling' (SS&L) approach. The spectral data of the input utterance is first segmented into phoneme-like units which are then time normalised by linear time normalisation. The neural network labels the time-normalised phoneme-like segments. 78.36% recognition accuracy is achieved for the phoneme-like unit.

In the second design, no time normalisation is required. After segmentation, recognition is performed by classifying the data in a window as it is slid one frame at a time, from the start to the end of of each phoneme-like segment in the utterance. 73.97% recognition accuracy for the phoneme-like unit is achieved in this application.

The parameters of the neural net have been optimised for maximum recognition performance. A segmentation strategy using the sum of the difference in filterbank channel energy over successive spectra produced 80.27% correct segmentation of isolated utterances into phoneme-like units.

A linguistic processor based on that of Kashyap & Mittal [84] enables 93.11% and 93.49% word recognition accuracy to be achieved for the SS&L and 'Sliding Window' recognisers respectively. The linguistic processor has been redesigned to make it portable so that it can be easily applied to any phoneme based isolated word speech recogniser.

To my Parents, brothers and sisters,

Aunt Afifa, Uncle Ghazanfer and Cousin Fahad.

ACKNOWLEDGEMENT

I would like to express heartfelt thanks to my Supervisor, Dr. T.J. Stonham for his guidance, help, and encouragement throughout this research work. Thanks are also due to Dr. A. Jones, A. Badii and M.J. Binstead for their help in the first year of my research.

The S&T Scholarship from the Ministry of Science & Technology, Government of Pakistan, enabling me to undertake this research is gratefully acknowledged. I would also like to commend the staff at Education Division and Accounts Section at the Embassy of Pakistan, London, for the smooth handling of this Scholarship.

CONTENTS

1. SUBWORD APPROACH TO SPEECH RECOGNITION 1

1.1 Introduction 1
1.2 Definition of Problem 5
1.3 Overview of Techniques in Phonetic Isolated Word Recognition 10
1.3.1 The Preprocessor Section 10
1.3.1.1 Acoustic Analysis 10
1.3.1.2 Feature Extraction 14
1.3.2 The Recognition Section 16
1.3.2.1 Speech Pattern Recognition Models 16
1.3.2.2 Segmentation & Labelling 19
1.3.2.2.1 Syllable Segmentation and Labelling 20
1.3.2.2.2 Segmentation and Labelling of Phoneme-like Units 24
1.3.2.2.3 Segmentation and Labelling of Broad Phonetic Sequences 31
1.3.2.3 Word Matching 32
1.4 Summary 34

2. NEURAL NETWORKS IN SPEECH RECOGNITION 35

2.1 Introduction 35
2.2 Review of Neural Networks in Speech Recognition 37
2.3 WISARD Nets 42
2.3.1 Single Layer WISARD Nets in Speech Recognition 47
2.4 Summary 50

3. SEGMENTATION OF ISOLATED WORDS 51

3.1 Introduction 51
3.2 The Word-Set 52
3.3 Euclidean Distance Measure for Spectral Dissimilarity 54
3.4 Weighted Euclidean Distance Measures 58
3.5 Chebyshev Distance Measure for Spectral Dissimilarity 61
3.6 Sum of Channel Difference as a Spectral Dissimilarity Measure 61
3.7 Segmentation of the Word-Set 65
3.8 Summary 71

4. RECOGNISER DESIGN BASED ON THE SEPARATE SEGMENTATION AND LABELLING APPROACH 73

4.1 Introduction 73
4.2 Acoustic Analysis 73
4.3 Segmentation 75
4.4 Time Normalisation 75
4.4.1 Linear Time Normalisation 76
4.4.1.1 Linear Interpolation 77
4.5 Data Reduction 80
4.6 The Recognition Module 82
4.7 Training the Recogniser 87
4.8 Testing the Recogniser 88

4.9 Evaluation of the Recogniser 89
4.9.1 Experiments with "Unaveraged Points Interpolation Method" for Time Normalisation 90
4.9.2 Experiments with "Averaged Points Interpolation Method" for Time Normalisation 96
4.9.3 Experiments with "Across Channel Mapping" of Spectral Frame Store 99
4.9.4 Adding Normalisation Techniques to the Recogniser 102
4.9.4.1 Spectral Energy Normalisation 102
4.9.4.2 Noise Subtraction Normalisation 105
4.10 Summary 112

5. RECOGNISER DESIGN BASED ON THE SLIDING WINDOW TECHNIQUE 113

5.1 Introduction 113
5.2 Design Based on the Centisecond Labelling Approach 113
5.3 Design Based on the Sliding Window Approach 115
5.3.1 The Recognition Module 118
5.3.2 Spectral Frame Store Sliding Window 122
5.4 Training the Recogniser 124
5.5 Testing the Recogniser 125
5.5.1 Class Label Sequence Unifier 126
5.6 Evaluation of the Recogniser 129
5.6.1 Experiments with 'Direct Mapping' of Spectral Frame Store 130
5.6.2 Experiments with 'Across Channel Mapping' of Spectral Frame Store 133
5.6.3 Experiments with the 'Segment Length Threshold' 136
5.7 Summary 142

6. LINGUISTIC PROCESSOR FOR ERROR CORRECTION 144

6.1 Introduction 144
6.2 String Error Correction Method of Kashyap & Mittal 145
6.2.1 Linguistic Rules 148
6.2.2 String Distance Measure 149
6.2.3 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the SS&L Approach 152
6.2.4 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the 'Sliding Window' Approach 157
6.3 An Alternative Linguistic Processor 160
6.3.1 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the SS&L Approach 163
6.3.2 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the 'Sliding Window' Approach 165
6.4 Conclusion 167
6.5 Summary 168

7. CONCLUSION AND SUGGESTIONS FOR FUTURE WORK 170

7.1 Conclusion 170
7.2 Suggestions for Future Work 174

REFERENCES 176

APPENDICES

Appendix A - The segment labels and the corresponding sound in an utterance 188

Appendix B - Data for calculating the class label substitution weights 189

CHAPTER ONE

SUB-WORD APPROACH TO ISOLATED WORD RECOGNITION

1.1 Introduction

Work on automatic speech recognition has been going on for almost 40 years now. Researchers have realised that the constraints within which an automatic speech recogniser must perform make it a very complex problem. Current knowledge related to speech recognition is insufficient to achieve the goal of a universal (continuous speech, speaker independent) speech recogniser. The approach adopted has been to relax some of these constraints in order to enable a limited solution to the problem. Some major problems (giving rise to the different categories of speech recognisers) [1], [2] are :

i). Variation in speech between individuals (interspeaker variations).

This problem may be got around by automatic speaker adaptation. Such systems are referred to as "speaker dependent" systems. In situations where more than one speaker must use the system, "speaker independent" systems must be used. These systems compensate for interspeaker variations but have lower recognition rates than "speaker dependent" systems.

ii). Word segmentation in continuous speech.

In continuous speech, pronounciation of consecutive words affect each other and it is often difficult to determine where one word ends

and another begins. Furthermore, the characteristic acoustic patterns of words exhibit much greater variability depending on context. One way this problem has been tackled is to separate the words by pauses (minimum pause duration is usually about 100 milli-second), giving rise to isolated word recognition. Thus speech recognition systems may be further classified as "Continuous Speech Recognition Systems" (CSRS), and "Isolated Word Recognition Systems" (IWRS).

iii). Psychoacoustic experiments have shown that human beings subconsciously use their own knowledge of the language (syntactic, semantic, pragmatic) and the context within which a sentence is spoken to restore missing phonemes, syllables or words (due to noise; speaker generated or environmental) and to resolve ambiguous words in continuous speech [3],[4]. This idea has led to the application of knowledge sources to aid the recognition process. Such systems are termed "Speech Understanding Systems" (SUS). The aim here is not the recognition of each and every word in the utterance but rather the intent of the message, hence "understanding". These systems come under the field of Artificial Intelligence and are subject to considerable research effort elsewhere.

Speaker dependent isolated word recognition through whole word matching is the least complex, and to date, the most successful category of speech recognisers. Most commercial recognisers belong to this category, claiming recognition accuracy in excess of 95% for a typical vocabulary size of around 250 words (IBM's Tangora-20 has a vocabulary size of 20,000 words) [5]. From the users' point of view, this is also the most unnatural way of speaking (since the utterance has to

be preceded and followed by a pause). Thus the goal of speech recognition must be towards the recognition of continuous speech.

Although template matching and hidden Markov models are the established techniques on which to base speech recognition systems, there has been a trend towards learning systems because of difficulties of defining consistent algorithms giving adequate performance with speech data [5]. By nature, the speech signal has high degree of variability and this has made it difficult to model mathematically. This is a common problem to all aspects of pattern recognition where real data is used.

WISARD, a general purpose pattern recognition device, is an example of a learning system [6], [7]. It is based on the n-tupling method, first proposed by Bledsoe and Browning [8], and has been applied with considerable success to problems in visual pattern recognition (character recognition [9], face recognition/verification [10], etc). In 1983/4 it was decided to extend the application of these nets to the field of speech recognition. A. Badii and M.J. Binstead have experimented with speaker dependent isolated word recognition for a set of 16 words, and have reported recognition accuracy of over 90% [11], [12]. The particular 16 words were chosen so that the word-set consisted of groups of similar sounding words which would be difficult to recognise since they were confusable. Further development of this work forms the basis of this dissertation.

Remaining at the single speaker isolated word level, the aim is to recognise the word-set of Badii and Binstead through a different approach. Instead of matching the whole word during the recognition process, this should be done at the sub-word, preferably "phonemic" level.

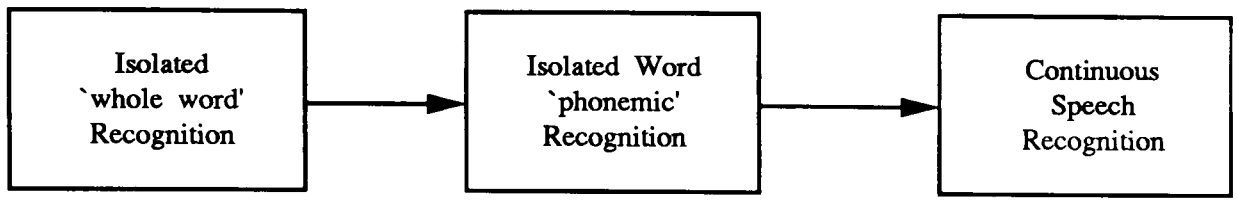


Figure 1.1 Isolated word 'phonemic' recognition as an intermediate stage towards CSR.

Research into the recognition of isolated words (based on the concept of whole word recognition) by a WISARD n-tuple recogniser has been completed and the results have been encouraging [12]. This technique appears to be at least as good as other existing methods. The next step, in my view, is the application of these nets to recognition of continuous speech. As continuous speech recognition (CSR) is a very complex and formidable task, I propose that this work should be towards an intermediate stage between IWR and CSR, as shown in figure 1.1, ie. the problem of IWR through the recognition of the different acoustic events (phoneme-like elements) that constitute the word uttered. Upon recognition of the phoneme-like elements, these may then be arranged together in the sequence in which they were recognised to produce the phonetic transcription of that word. The reason why this approach is considered as an intermediate stage between IWR and CSR is that the problem of segmenting the word into phoneme-like elements is also applicable to CSR, but CSR has in addition the word boundary segmentation problem mentioned earlier in (ii). By remaining at the IWR level, the word boundary segmentation problem is eliminated and it becomes possible to focus on the problem of segmenting the word into sub-word units. The principles needed to accomplish this task would be applicable, perhaps with some

modification, to the recognition of continuous speech. Thus one of the aims of this work is to act as a stepping stone from IWR to CSR using the WISARD n-tuple pattern recogniser. This also seems to have been the approach taken by Makino & Kido, who first designed a word recognition system based on phoneme recognition using discriminant filters [13], and then applied this approach, with some modification, to phoneme recognition in continuous speech [14]. Billi *et al* [15], also hold this view.

The sub-word unit based approach is also the established technique in the design of large vocabulary isolated word recognisers [16], [17], and it is this class of speech recognisers that this work is more specifically related to. This work is intended to be an investigation into techniques for the design of large vocabulary isolated word recognisers using WISARD nets.

1.2 Definition of Problem

IWRS store in memory a reference set of word features (also referred to as templates or prototypes) for each word in the system's vocabulary. The unknown input word is compared with the template(s) of each word to determine which prototype it is most similar to. As a consequence of this method, two of the problems that will occur as the vocabulary of the system is increased [3], are :

(i). Memory requirements for storing the word reference patterns increases.

(ii). Response time increases linearly with the size of the vocabulary.

The technique of matching all reference prototypes with the unknown input word becomes inefficient and it becomes essential to use other methods that reduce storage and computation time. This involves improvements in reference pattern representation and search strategies. Rather than store the reference pattern for the whole word, the need is for the recognition of the whole by analysis of the component subparts that constitute the word, and then grouping these together to form larger units. This makes it necessary to adopt a segmentation and labelling scheme, a technique also used in CSRS. The purpose of segmentation is to divide the utterance into discrete units (ie the subparts). The process of labelling associates a symbol corresponding to the recognised unit with each of the segments.

Spoken words can be represented as strings of phonemes. There are about 44 phonemes in standard Southern British English [18], [19]. Taking the phoneme as the unit for recognition, if it can be recognised with a certain degree of accuracy, then it should be possible to recognise an unlimited vocabulary of English words with a recogniser consisting of only 44 classifiers (one for each phoneme). Thus this approach offers a solution to the problem of increasing storage requirements with increase in the vocabulary of an IWRS based on matching whole word templates. The computational expense of recognising phonemes is also small compared with that required for whole word recognition since there are more words than phonemes (hence there are fewer classes to match the unknown input with), and also because phonemes are smaller than words. Furthermore, correction of errors can begin upon receipt of each phoneme (there is no need to wait for all the phonemes in the word to be recognised) by applying phonotactic rules [19]. These rules express which combinations of

phonemes are possible and which are prohibited in the language. This would help in speeding up the recognition time.

Thus this approach has considerable appeal and much ASR research has been directed towards automatic phoneme recognisers. The main focus of the ARPA Speech Understanding Project [20] also, has been the transformation of phonemes into words with the aid of knowledge from nonacoustic sources (syntax, semantics, pragmatics).

Although all researchers are unanimous as to the need to base the recognition strategy at the sub-word level, opinion is divided as to what the optimal unit for recognition is. The ideal unit would be the phoneme except that evidence seems to suggest that it does not exist in reality but is a linguistic unit useful for transcription of speech [16], [21]. An utterance is not merely the result of stringing phonemes together, end to end, as this would require that the articulators move from one phoneme position to the next instantaneously. Since all moving parts have mass and hence inertia, this would be mechanically impossible. What actually happens is that as the utterance proceeds from one phoneme to the next, the target articulator position for that phoneme is approached but generally not met. Thus the articulations only approximate to the ideal target positions for the phonemes in the utterance. This phenomenon is known as coarticulation [1], and as a consequence of it, cues for the identification of a phoneme may be spread over adjacent phonemes. The sound pattern of a phoneme is influenced by both the preceding and following phoneme. Some researchers have therefore proposed the use of a larger unit than the phoneme, such as the syllable [21], [22], [23], demi-syllable [24], and the diphone [25].

The diphone is composed of half of one phoneme followed by half of the next (by segmenting speech at the steady state centres of the

phonemes). In this way the coarticulatory influences of the two phonemes on each other (ie the transitions from one phoneme to the other) can be incorporated into the recognition process. More reliable detection may be effected by the use of a unit the size of a syllable since the coarticulatory transitions are captured within a larger unit. A major drawback with the use of these units as the basis for recognition is that they make the recognition set (ie. the number of classes to be recognised) too large, for example, there are over a thousand syllables in the English Language. Thus the phoneme has continued to receive attention as the unit upon which to base recognition.

Although it is not possible (due to coarticulatory effects) to define phoneme boundaries with precision [26], one approach has been to devise automatic algorithms to segment speech on the basis of acoustic boundaries [27]. The method is to look for significant changes in the acoustic characteristics over the entire utterance. The segments thus created are normally referred to as "phoneme-like elements" in acknowledgment of the fact that they are not phonemes in the linguistic sense. Based on this principle, an improved method for automatic segmentation of isolated utterances into units roughly corresponding to phonemes is suggested in Chapter 3. The segment created may now be labelled by matching with the reference prototypes. A design for a WISARD n-tuple speech recogniser based on this approach is presented in Chapter 4.

Whereas the above method is to determine the segment boundaries and then to label the segment, in the centi-second labelling method, the utterance is considered as a sequence of 10 milli-second frames. Each frame is labelled by matching with the reference prototypes and frames with the same label are grouped together to form a segment that corresponds to the phoneme. This approach is also sometimes

referred to as "Segmentation-by-Recognition" or "Recognition-then-Segmentation" since the spectral frames are first recognised and then grouped into a segment. A design for a WISARD n-tuple speech recogniser based on similar idea is presented in Chapter 5.

Both system designs mentioned above make three types of errors, namely insertion errors, deletion errors and substitution errors. Insertion and deletion errors are due to errors in segmentation of the utterance. If the segmentation process is not perfect (to date there is no segmentation algorithm that can claim error-free segmentation of speech) some sounds may be completely missed, causing deletion errors or extra segment boundaries may be inserted, causing insertion errors. Substitution errors are due to misclassification of a segment at the recognition stage. It may be possible to recover from some of these errors by applying a linguistic processor to the string of labels output by the speech recogniser. In Chapter 6, a linguistic processor due to Kashyap & Mittal [28], was implemented and applied to the output from the n-tuple recogniser. The aim was to see how much the performance of the n-tuple recogniser can be improved by complementing the recogniser with phonotactic rules and string error correction techniques. A disadvantage of the Kashyap & Mittal linguistic processor is that it has to be adapted to the speech recogniser it is being used with (phoneme replacement weights need to be calculated from the performance statistics of the speech recogniser). Modification to make this design portable is suggested in this chapter.

1.3 Overview of Techniques in Phonetic Isolated Word Recognition

The processes involved in the recognition of isolated utterances using a phonetic analysis approach are shown in figure 1.2. With reference to figure 1.2, an outline of the various techniques applied at each stage of the phonetic isolated word recogniser will be presented. A discussion of the different approaches researchers have taken in implementing such systems is also included. Thus this section is intended as a review of the techniques and work done in this branch of the field of speech recognition.

Overall, the recognition system in figure 1.2 may be divided into two parts; the pre-processer section and the recognition section.

1.3.1 The Pre-processer Section

The pre-processer section of the system comprises the acoustic analysis stage and the feature extraction stage.

1.3.1.1 Acoustic Analysis

This stage is concerned with the extraction of acoustic parameters from the speech waveform. The objective is to transform the speech signal in such a way as to enhance certain properties that would enable its detection with ease and efficiency, and also lead to a reduction in the data rate. Frequency-domain representation is the prefered method with most if not all workers in the speech recogniton field today. This is motivated by the fact that it is known that the human ear acts as a power spectrum analyser [3], [28]. Time-domain representations such as zero-crossing rate and energy measurements have also been used [29], [30]. Experience has shown that the

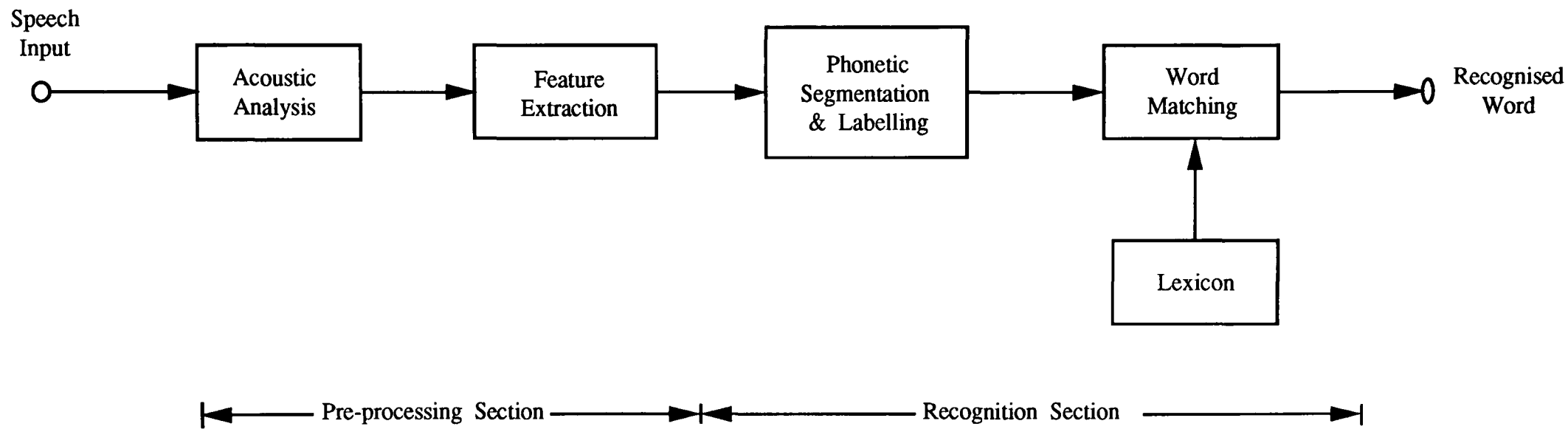


Figure 1.2 Stages in a phonetic isolated word recogniser

frequency-domain representation is richer in information than the zero-crossing and energy measurements in time-domain representation. Popular methods for analysis in the frequency domain are :

(i) *Fourier analysis* [31]

This technique enables analysis of the speech signal's spectral energy distribution and its variation with time. The speech signal is divided by filtering into about 20 to 30 frequency bands, covering the range of frequency of human speech. The output of each filter gives a measure of the energy in that frequency band. Recognition can be effected by comparing the energy levels with those of a template. It may be implemented by the use of the fast Fourier transform (FFT) or a bank of analogue band-pass filters. The latter method is used for speech transformation to the frequency domain in the work presented in this dissertation. Although the spectrum is indicative of the shape of the vocal tract, extraction of formants is difficult because the spectrum also has superimposed on it the spectrum of the glottal excitation (voice pitch). Use of a suitable analysis window (usually 10 msec) helps to reduce this effect. However, for a recognition scheme based on the extraction of formant frequencies, the LPC method is used.

(ii) *Linear predictive analysis* [32]

Linear Predictive Coding (LPC) is considered as one of the most important developments in speech recognition research. The method predicts the amplitude of a speech wave at a given instant from a weighted sum (or linear combination) of its amplitudes at a small number (8 to 14) of earlier instances. The coefficients or weights that give the best estimate of the true speech wave can then be mathematically converted into an estimate of the amplitude spectrum.

This method is equivalent to treating the vocal tract as a pipe of varying circular cross-section, ie. as a sequence of resonant cavities. Since this method models the vocal tract (and not the vocal chord vibration), the resulting spectrum is smooth due to absence of the pitch harmonics. Thus the formant structure of the speech wave is clearly identifiable. This property greatly reduces the difficulties with the estimation of formant frequencies (used to classify vowels and sonorants) and formant trajectories (used to classify diphthongs).

As it is possible to obtain the vocal tract area function and vocal tract length estimate from the LPC model of the vocal tract [33], this enables the normalisation of any speaker's vocal tract shape and length to some standard value. This is considered to be a major contribution towards the goal of achieving speaker independence in speech recognition.

The linear predictive residual proposed by Itakura [34] as a speech sound similarity measure has made it possible to perform all preprocessing and similarity measurements simply in the time domain and hence easier implementation of the LPC method.

As a result of the advantages afforded by the LPC method, it appears to be the most popular technique. However, a study by White & Neely [35] has concluded that LPC and bandpass filtering are approximately equivalent bases for measuring speech waveform similarity.

(iii) *Cepstral Analysis* [31]

This method is used to remove the glottal excitation from the speech spectra obtained through FFT such that the spectra due to the vocal tract only remains. The cepstrum is obtained by performing FFT on the logarithm of the FFT on the speech waveform. The lowest

quefrency peak in the cepstrum corresponds to glottal pitch frequency. After filtering out the pitch effects and performing the inverse FFT on the resulting cepstrum, the desired smoothed speech spectra is obtained.

1.3.1.2 Feature Extraction

After acoustic analysis, the speech signal is in a form of representation that is suitable for comparison with the system prototypes ([34], [35]). In some systems, there is a further feature extraction stage. Here, a number of additional features are calculated from the data from the acoustic analysis stage ([29], [36], [37], [38]). These features are used in the segmentation and labelling stages. Some of the most commonly extracted features are discussed below.

Reddy [27] used *zero-crossing density* and *intensity level measurements* directly on the speech waveform in order to segment it. The speech wave was first divided into a succession of "minimal segments" of 10 msec duration. Intensity levels were used to group together acoustically similar minimal segments to form larger segments. Zero-crossings were used to resolve ambiguities. Bursts and fricatives can be reliably detected using zero-crossings [29].

Schwartz & Makhoul [26] detected the presence of *fundamental frequency of voicing* from the linear prediction residual to segment into voiced and unvoiced portions. However, this can be more easily done by comparison of energy concentration in the high frequency (3700 - 5000 Hz) and low frequency (100 - 900 Hz) regions [39].

Kasuya & Wakita [40], Weinstein *et al* [39] apply a threshold to the *energy function* for silence detection (indication for stop consonants and pauses). When the energy falls below the threshold, the segment is

marked as a silence segment. Sambur & Rabiner [29] use the spectral energy's variation over time to segment the utterance of a digit into initial, medial and final regions.

K. Tanaka [38] extracts local peaks of the spectrum envelope from the output of a 46 channel filterbank which approximately correspond to *formants*. Itahashi *et al* [36] digitally filter the speech signal into four frequency bands. The three lower bands correspond to the first three formants of certain vowels. McCandless [41] has developed an algorithm for tracking formants during voiced sounds, from the linear prediction spectra. As mentioned earlier in (ii) Section 1.3.1.1, the formants and their trajectories are commonly used in the classification of vowels, sonorants and diphthongs.

Das & Stanat [42] used the *gross spectral shape* (energy in low, mid and high frequency bands) as part of 23 features derived from a 20 filter vocoder-type speech analyser for segmenting utterances. Cole *et al* [43] used the ratio of high frequency energy to low frequency energy to discriminate between certain groups of letters (B,P,V vs D,T,Z and F vs S) in their isolated English letter recogniser, FEATURE.

This completes the pre-processor section of the recogniser. Data from the feature extraction stage may now be passed to the recognition section where the recognition process is performed.

1.3.2 The Recognition Section

The recognition section of the system comprises the segmentation and labelling stage, and the word matching stage.

Having extracted the desired features in the appropriate form from the speech data input to the pre-processing section, the task of this section of the system concerns the storage of reference speech patterns consisting of these features (training) and its comparison with the unknown input (labelling). The approach is to construct suitable models that are provided with *a priori* knowledge of the speech patterns to be recognised, and against which the unknown patterns are compared to perform recognition. The model outputs a label corresponding to the class that it associates the unknown pattern to. The word matching stage attempts to match the sequence of labels output by the models to one of the entries in the systems lexicon.

1.3.2.1 Speech Pattern Recognition Models

Two methods have been most successfully used for the modelling of speech patterns; template matching and hidden Markov models (HMM). However, limitations in the performance of these models have led researchers to continue considering alternative approaches. An approach that is currently receiving much attention is the application of neural network models (also known as 'connectionist models') to speech recognition, an example of which is the WISARD n-tuple pattern recogniser. This will be the subject of the next chapter.

The concept behind the HMM approach is presented in brief. The references given may be consulted for further information. The template matching method is covered in greater detail. This is because the problem of segmentation of utterances associated with phonetic

recogniser designs based on this approach is also applicable to designs based on the WISARD n-tuple technique presented in this dissertation. In fact, the WISARD n-tuple recogniser with a tuple size of 1 is equivalent to template matching.

Hidden Markov Models

This method [44] has recently become popular (before the 1980s, template matching by dynamic time warping was the most widely used approach) and is successfully competing with the template matching method as the model for basing speech recognition systems. The technique is a statistical method for the classification of observation sequences. It made its first notable appearance in Baker's DRAGON connected speech recognition system [45], [46] at Carnegie-Mellon University. The works of Jelinek [47] and Levinson *et al* [48] are also considered important contributions. More recently, this technique has been implemented in IBM's Tangora-20 word recognition system [49], capable of handling a vocabulary of 20,000 English words with an average error rate of 5.4%.

The basic idea is to characterise each recognition unit (ie. word, syllable, phoneme, diphone etc.) by a model that generates patterns of features representative of that recognition unit. Thus each unit within the recognition set is represented by an HMM. As with the template matching method, the task of recognition is to find the model that best matches the input feature patterns (eg. LPC coefficients), and this is the model with the highest probability of generating the observed sequence of input feature patterns.

With the input utterance in the form of a sequence of frames of feature vectors, the HMM consists of a sequence of states, each state

being associated with one or more frames of the input feature vector. It can change state at each frame interval and this is determined by the *transition probabilities* for the state that it is currently in. Each state is also characterised by a probability distribution function which gives the probability (called *output probability*) that, being in that state, a particular acoustic feature vector is produced. These parameters express the behaviour of the HMM, and are determined by using either the *forward-backward algorithm* (also called the *Baum-Welch algorithm*) or the *Viterbi algorithm*. Time-scale variability, which is handled by the DTW technique in the template matching method, can also be achieved in HMMs by allowing the model to stay in the same state for succeeding frames, or to skip the next state in the Markov chain.

The advantage of the HMM approach is that it permits the modelling of phonetic units without the need to segment the utterance. The disadvantage is that it requires large amounts of training data to obtain reliable probability estimates. The set of training utterances have to be chosen with care so as to include a reasonably high number of occurances of all defined units, and also that they occur in different contexts (to account for coarticulative effects).

Template Matching

The most common approach to labelling speech segments is the template matching method. The segments corresponding to each acoustic-phonetic unit (syllable, phoneme, diphone etc.) are stored as templates in the training phase. In the recognition phase, each of the segments of the input speech are, in turn, directly compared with each of the templates using distance measures such as Euclidian distance, Chebyshev distance or the Itakura distance (if LP coefficients are

stored as templates). To compensate for variation in speaking rate, and hence the duration of different samples of the same phonetic segment, the dynamic time warping (DTW) algorithm [50] performs time normalization during the comparison between the unknown input segment and the template. The segment is assigned the label of the template that gives the best match ie. the template with the minimum test-to-reference pattern distance.

Before the template matching process can begin, the input utterance must be converted to segments of desired type ie. syllable, phoneme etc. The next section considers in detail, techniques for segmentation and labelling implemented in phonetic recogniser design based on the template matching method.

1.3.2.2 Segmentation and Labelling

This stage of the system attempts to use some or all of the features obtained from the pre-processing section to segment the speech input to units of the desired size (syllable, phoneme etc). Much work has been done on automatic segmentation of speech. There are two basic approaches to the problem, namely context dependent segmentation and context independent segmentation.

(i) *Context independent segmentation*

No explicit information is required. This approach utilizes only the acoustical information contained in the speech signal being segmented [51], [52]. Parameters (eg. spectral change from one sampling period and the next) extracted from the speech signal are compared with a threshold value and a boundary is indicated when

the threshold is exceeded. The advantage of this method is that it is easy to implement.

(ii) *Context dependent segmentation*

This approach utilizes the explicit information that is known *a priori* [39], [53]. It is known that certain parameters perform better than others at detecting certain acoustic-phonetic features in the incoming speech signal, eg. zero-crossing measurements for detecting fricatives and bursts, and energy related measurements for detecting voicing. Thus the idea is that if measurements indicate that the segment of speech under consideration is of a particular phonetic category, then the appropriate feature parameter is selected to detect the segment boundary.

Another problem is the size of the segmented units. Researchers have not been able to agree as to which type of unit the speech signal should be segmented [54]. Although segments the size of demi-syllables and diphones have been considered, three approaches appear to figure prominently :

(i) Segmentation into syllable sized units.
(ii) Segmentation into phoneme-like units.
(iii) Segmentation into broad phonetic sequences.

1.3.2.2.1 Syllable Segmentation and Labelling

Attempts at designing phonetic type-writers have shown that effects of concatenating units the size of phonemes are unpredictable

since the phonetic manifestation of a phoneme varies greatly depending on context.

O. Fujimura [21] proposed that the syllable be adopted as the minimal phonological segment unit since coarticulatory effects across syllable boundaries (wherever they are clearly defined) are mostly natural and universal (unlike phoneme boundaries). He suggests that together with prosodic information of the syllable, syllable concatenation will be efficient and require simple boundary manipulation. Classification of syllables should be in terms of classes of features : nucleus, initial and final, and then in terms of subclasses within each class ie. a hierarchical strategy. He also proposed the measure of 'vowel affinity' to constrain admissable syllable structures in English. Mermelstein [22] has however shown a case where this phonological criteria tends to break down.

Mermelstein [22] defined a "loudness" measure for the speech signal obtained from the speech power spectrum and used it to segment speech into syllabic units. Relative loudness maxima represent potential syllabic peaks and relative loudness minima as potential syllabic boundaries. The term 'syllabic unit' was introduced to refer to these speech segments to differentiate them from syllables defined at the phonological level. Tests on 430 syllables showed 6.9% syllables missed and 2.6% extra syllables. Possible contractions across words in continuous speech may cause the syllable count for a sequence of words to be smaller than the individual syllable count. A set of rules predicting such phenomena may be used to take this effect in to account.

Following Mermelstein's line of concept, Kasuya & Wakita [40] have proposed the intersyllabic segment (a sound segment bounded by two successive syllable centres) instead of the syllabic segment as the

unit of analysis. Their basis is that syllable centres are more reliably detected than syllable boundaries, and that it also reduces the number of alternative hypotheses that will have to be considered at the higher linguistic level. The RMS energy of the input signal is used to detect dips and peaks which are then classified as consonant segments or vowel-like segments. Since this work relies on detecting syllable nuclei (vowel-like segments) claiming 96.1% syllable nuclei correctly detected, the methods are also useful in syllable based work.

The syllable based approach for speech recognition is most popular in Japan. This is because Japanese is a clearly syllable based language. Japanese syllables take the simple form of a consonant followed by a vowel (CV) or monosyllables. In addition, they are limited in number up to about 100. Thus the syllable is a suitable unit upon which to base recognition of Japanese speech. The syllable has also received considerable attention in France for similar reasons.

The syllable structure is composed of a syllabic nucleus preceded and followed by consonant clusters (initial consonant clusters and final consonant clusters respectively). Consequently, the common approach is to identify vowel and consonant segments in the input signal. A popular method is syllable matching based on the dynamic time warping (DTW) algorithm using templates created from CV and VCV type syllable utterrances ([55], [56]). VCV sections are relatively easily segmented by locating the vowel portion as syllable nuclei. Moreover it also contains the transition feature between phonemes avoiding the difficulty of exactly identifying consonant boundaries. Hataoka *et al* [57] segment the input speech into CV or VCV sections by first locating stationary intervals, and then detecting the vowel interval from the stationary parts. The consonant part lies within the interval between two vowels, and is recognised by DP matching with VCV

reference templates. Although this work is aimed at phoneme recognition, it presents a method for the automatic creation of VCV reference templates applicable to syllable based recognition. Stationary interval detection rate of 100%, vowel interval detection of 97.4% and vowel deletion rate of 2.6% are claimed.

Another approach proposed by Gauvain [58] is to create the reference pattern for each word by concatenating isolated syllable templates, and then matching with the test utterance using DP. The advantage of this method is that it bypasses the problem of segmentation of the speech training and test data. The results show an error rate of 12% compared with 6% for matching whole word templates. This suggests that improvement may be expected by using syllable prototypes extracted from words (ie naturally spoken utterances) in order to take in to account the allophonic variations due to coarticulative effects within syllables. This implies the use of a small number of context sensitive templates for each syllable ([59], [60]). Höge *et al* [59] apply clustering technique [61] to the training segments to build three representative templates for each initial and final consonant segment considering three different vowel contexts (front, neutral and back).

Another method is to obtain the result of syllable recognition in the form of a syllable lattice as a mechanism against failure in the matching stage. The matching score for each syllable candidate in the lattice acts as a reliability measurement. The lattice can be fed into a linguistic processor for error correction ([56], [62]). With a lattice depth of 10 syllables, the correct syllable is to be found in the lattice 96.6% of the time.

Tanaka & Kamiya [62] perform syllable identification by using syllable descriptions in which phonological variations of syllables (eg.

vowel reduction or vowel deletion) have been accounted for. The method is to characterise the speech input in to one of 18 symbols every 8 ms. The syllable is identified (upto three candidates are chosen) by matching a sequence of these symbols with syllable patterns in descriptive form, taking in to account phonological variations. Syllable hypotheses are generated from a rewrite rule table to account for syllable substitution/deletions. Top down verification is then performed on the syllable lattice to check syllable concatenation. Accuracy of 93.7% is reported.

1.3.2.2.2 Segmentation and Labelling of Phoneme-like Units

From the practical point of view, this approach is the most appealing considering the small number of phonemes in comparison with the number of words in a language. The main advantage is that in languages which use alphabetic writing of any kind (as opposed, for example, to ideography in Chinese) the letters and their combinations reflect (with varying degrees of consistency) a linear arrangement of phoneme realizations. Sequences of phoneme-like segments can be transformed into strings of letters in conventional orthography with a minimum of rules and vocabulary lookups when it is desired that the output of a speech recogniser should be used in a speech-to-text conversion. However, its implementation has proved to be difficult. Phoneme segment boundaries cannot be identified with absolute certainty ([26], [63]). To cope with this problem the following strategies have been tried :

(i) *Phonetic lattice.*

The context within which a word is spoken causes acoustic variations that can lead to ambiguities in segmentation and labelling [26]. To cope with these ambiguities, some vagueness is incorporated in to the recognition process ([26], [64]), ie. rather than making a single decision, alternative choices in the form of a lattice are generated so that the correct class is included, and hence increasing the chances of correct recognition. The disadvantage is that many other words are found (depending on the amount of vagueness) in addition to the correct word. The BBN HWIM system performs more than a single segmentation of any region of the utterance producing a lattice of segments enabling different paths to be taken within the region [26].

(ii) *Centi-second Labelling.*

This method is also called the "Recognition-then-Segmentation" or "Segmentation-by-Recognition" approach [1]. It involves the labelling of each 10 milli-second (hence centi-second) speech spectral frame by matching with a set of spectral prototypes for each phoneme. Centi-second segments with similar labels are then grouped in to a single phone. The boundaries of a segment are determined by a change of classification between two adjacent time samples. Errors in labelling are most likely to occur at the transistional part between two phonemes.

(iii) *Separate Segmentation and Labelling.*

Together with centi-second labelling, this method has been the common approach. The segmentation of an utterance into phoneme-like units and the classification of these segments is treated as independent parts. The basis for much of the work using this method is the fact

that speech is composed of intervals such as quasi-stable intervals in which the parameters remain in almost constant state, and transient intervals in which parameters undergo gradual change [65], [27]. Reddy [27] uses intensity levels of the time domain speech waveform to determine acoustically similar 10 msec segments, and groups all segments that lie within a tolerance interval to form a sustained segment.

The mean-square distance between succesive spectra may be used as a measure of their dissimilarity. A threshold may be applied to this measure to determine segment boundaries (Beninghoff & Ross [51]). The setting of the threshold is critical. If set too high , segment boundaries may be missed (segment deletion errors), and if too low, then extra boundaries are detected (segment insertion errors).

An approach proposed by Charbonneau & Moussa [52] is to apply multidimensional scaling analysis on the 25 channel speech frequency power spectra to reduce it to a lower dimension of 7 channels. The 7 channels represent a point in a 7 dimension space in which a transition between two phonemes induces a significant variation on at least one axes. 4% segment insertion error, 19% segment deletion error are reported. 95% of the boundaries detected were correct.

L. Kot [66] proposes a segmentation technique based on the Syntactic Theory of pattern recognition developed by K.S. Fu [67], and which has also found favour with P. Mermelstein [68] and R. de Mori [69]. It makes use of the similarity relations among the momentary spectra composing the dynamic spectrum of an utterance. These relations are described by a context sensitive grammar, thus *a priori* information concerning the structure of the dynamic spectra of the speech signal is required. 90% correct segmentation of Polish utterances is reported.

The method of Bridle & Sedgwick [70] is to represent frames within a segment by an average spectrum. The aim is to minimise the spectral deviations within the segment from its corresponding average spectrum. A "segment evaluation function" measures how well any given portion of the utterance can be represented as a single segment. The segmentation process finds the sequence of segments which best represents the whole utterance.

R. Andre-Obrecht proposes a statistical approach [71], [72]. The speech signal is modelled by an AR statistical model and test statistics are used to sequentially detect changes in the parameters of the model. Jumps in the model parameters always correspond to an articulatory or an acoustic change. Most segments are stationary parts of phonemes.

The above discussion outlined some of the different approaches to segmentation of speech into phoneme-like units. The following discussion considers the labelling of these segments.

The labelling technique common to most designs of isolated word recognisers based on recognition of phoneme-like templates is the template matching method with dynamic time warping of the unknown input segment with the reference phoneme-like templates. The differences are mainly in the approaches to phoneme representation. Some of these approaches will be discussed by considering some system designs based on this technique.

To deal with time-varying patterns by algebraic operation, the method adopted by K. Tanaka [37] is to express phonemic values by a matrix composed of acoustic parameters, in which the time factor is taken into account in its column components. Phonemes are defined in terms of potential functions within this matrix space. Some of the phoneme regions are considered to be of complex shapes. A linear

combination of Gaussian subpotential functions are chosen to represent such phoneme regions. This enables the system to function with little *a priori* knowledge of individuality and coarticulation.

Since speech sounds do not fully occupy the space of all possible sounds that may be represented by a multi-dimensional feature space, a technique known as vector quantization is used to reduce the data-rate of the speech signal. Spectral data at each sampling interval (also called a "frame") is compared with a set of pre-stored reference frames (called a codebook). Each frame of the codebook is associated with a different output symbol. Each frame of the input signal is replaced by one from the codebook to which it is 'nearest' (according to a distance metric), thus the speech signal is transformed into a sequence of symbols. This process is used in the word recognition system of Sugamura *et al* [73] based on their proposed SPLIT (Strings of Phoneme-LIke Templates) method. 256 phoneme-like templates are generated by applying the vector quantization technique to a few thousand frames selected arbitrarily from training utterances. The training word is divided into 16 msec segments (frames) and each segment is compared with and assigned the symbol of the phoneme-like template which minimises the spectral distance between the two. In this way, the word templates are represented as sequences of phoneme-like templates. In the recognition stage, the spectral distance between each frame of the input word and each phoneme-like template is stored as an element of a distance matrix. Summing the elements of this matrix gives the total spectral distance between the input word and each word template. This distance measure is minimized using dynamic time warping.

The approach of Makino & Kido [13] is to first locate the typical frame of the phoneme-like segment. Phoneme recognition is achieved

by applying the Bayes decision method to the output of a discriminant filter for that typical frame. The output of the discriminant filter is a weighted summation over 7 frames preceding and following the typical frame, of the sum of the logarithmic output of a 29 channel band pass filter. The weight vectors have been preset for each of 11 features such as fricative, nasal, voiced plosive etc. The standard pattern of each phoneme was made using samples extracted manually by visual inspection. Since a preset number of frames are considered for each typical frame, no time normalisation is required. An overall correct phoneme recognition score of 75.9% is reported. Word recognition of 92.4% was achieved by calculating the likelihood between every item of the word dictionary and the phonemic sequence output from the recogniser. Probabilities of insertions, omissions, and substitutions of phonemes are also taken in to account during computation.

An interesting approach is that of computer generated phonemes (CGP) proposed by Hinton & Siegel [74]. The CGP is a vector of features (LPC coefficients, zero-crossing rate, energy) generated to represent consistent regions of speech and hence model the different phonetic sounds of the word. The CGP is created when the speech signal indicates that the vocal tract is relatively stable (by measuring similarity between successive frames), or when an unvoiced fricative (high zero-crossing rate) occurs at the beginning or end of a word. When such points are detected, an "anchor frame" is established. The CGP consists of the individual average of the features of a preset number of frames prior to and following the "anchor frame". The averaging is done to remove transient variations in the features for that segment of speech. Slowly varying speech sounds such as vowels, glides, liquids and diphthongs are represented by more than a single CGP. This technique has the effect of speech compression since a small

number of CGPs will represent a relatively large segment of speech. Each word is thus represented by a string of CGPs. To account for the different variations in utterances of the same word, each word in the classifiers vocabulary can be expressed by CGP strings of different lengths and/or CGP sequence. Classification is made by dynamic time warping between the test word and the reference templates for each word.

Whereas the methods discussed above recognised a phoneme by measuring similarity to all phoneme templates simultaneously as a single process, the approach of Morii *et al* [75] (implemented in hardware by Hiraoka *et al* [76]) is to perform discrimination of phonemes individually for vowels, semi-vowels and consonants. Consonants are further classified into groups of phonemes with similar features, and then the individual phoneme is discriminated from within this group.

A recent approach which is a knowledge based approach (hence beyond the scope of this thesis but is being mentioned here for completeness), is that of top-down phoneme verification [77], [78], [79]. Previously, the top-down approach, which is based on hypotheses verification, was applied to sentence level verification and word level verification. The expectation is that extending this approach to the phoneme level may produce more accurate recognition. Indeed, Morishima *et al* [77] achieved an average recognition accuracy for fourteen consonants of over 96% . Knowledge of possible coarticulations caused by phoneme concatenations is contained in a knowledge source. This is used to verify the phoneme sequence hypothesis derived from the word verification level.

1.3.2.2.3 Segmentation and Labelling of Broad Phonetic Sequences

The variability of phoneme sounds makes reliable identification of correct sequence of phoneme classes a difficult objective to achieve. Then there is also the problem of lexical access for very large vocabularies. The idea of segmenting an utterance into a sequence of broad (coarse) phonetic classes was proposed by Shipman & Zue [80] as another approach to this problem. They have demonstrated this method to be an effective and efficient way for discriminating among words in a large lexicon.

Their approach is to segment the utterance into a sequence of six manner of articulation labels; vowel, strong fricative, weak fricative, stop, nasal and glide. The lexicon is partitioned into equivalence classes. An equivalence class is a set of words for which the segmentation process produces the same sequence of broad phonetic labels. For a 20,000 word lexicon, there are an average of 35 words per equivalence class. The largest equivalence class has about 200 words, which is only 1% of the lexicon. Thus the unknown utterance after segmentation into sequence of broad phonetic labels, is narrowed down to one of the equivalence classes. Detailed phonetic analysis identifies the input with one of the entries in the equivalence class.

Since less detailed distinctions are needed to produce a broad phonetic representation than a detailed phonetic representation (eg. phonemes) they are relatively more robust to acoustic variability due to phonetic context, and even across different speakers (useful for speaker independent recognition). A situation when an error can occur due to variability in the speech signal is deletion of a phoneme. This will affect the sequence of broad phonetic labels representing the utterance, and hence cause the selection of an incorrect equivalence class. Some

broad phonetic classes are confusable. For example, a noisy stop gap results when the closure in the stop gap is incomplete, causing it to be labelled as a weak fricative. This problem may be overcome by allowing multiple representations of the word in the lexicon.

Huttenlocher & Zue [81] have studied how variability in the speech signal affects the broad class representation. Their study revealed that the phonetic information around stressed syllables are much more important in recognition than the information around unstressed syllables. This is due to the fact that phonetic segments around unstressed syllables are more variable than those around stressed syllables and hence less reliable in recognition. Furthermore, almost all segment deletions occur around unstressed syllables. This suggested that the words be classified according to those phonemes around stressed syllables only and to disregard those around unstressed syllables.

Lagger and Waibel [82] propose a knowledge source that uses coarse phonetic information to derive the coarse phonetic class segments. To correct for errors, dynamic programming is used to align the errorful coarse class string to the coarse class strings of the lexicon.

The effectiveness of this approach has established it as a useful alternative for large vocabulary isolated word recognition.

1.3.2.3 Word Matching

The output from the segmentation and labelling stage consists of a string or lattice of phonetic symbols associated with the unknown input utterance. The function of the word matching stage of the system is to

identify this string of symbols or lattice with an entry in the lexicon and output it as the recognised word.

The segmentation and labelling process produces three types of errors; insertion and deletion errors due to errors in segmentation, and substitution errors due to misclassification by the labelling process owing to ambiguities in the speech data for the phoneme-like segments. Thus error correction must be performed on the string (or lattice) of phonetic symbols before attempting to match it with a word in the systems lexicon. This is achieved by applying a linguistic processor to the string of phonetic symbols.

One approach is to use a distance function to measure the closeness of the phonetic string to a word in the system's lexicon, and to choose the word that is closest to the phonetic string as the final decision of the speech recogniser. The Hamming distance may be used if the two strings to be compared are of the same length. The Levenstein distance [83] is commonly used if the two strings are of different lengths. This method transforms a string s_1 into another string s_2 by substituting n_s symbols, inserting n_i symbols and deleting n_d symbols. The Levenstein distance is defined as the minimum number of these operations required to transform s_1 into s_2, as given below :

$$\mathrm{LD}(s_1 \text{ --> } s_2) = \min_{n_s, \mathrm{n_i}, n_d} (n_s + \mathrm{n_i} + n_d)$$

The phonotactic rules of a language express which combination of phonemes are possible and which are prohibited. An approach proposed by Kashyap & Mittal [84] uses these rules to eliminate the invalid phoneme combinations generated by the string transformation process. This step produces a few words that are possible correct

representations of the phonetic symbols string input to the linguistic processor. The distance between each word in the set of possible words produced by the error correction process and the phonetic symbols string is computed and the utterance is recognised as the word that gives the minimum distance and also has an entry in the lexicon.

1.4 Summary

Isolated word recognition by matching whole words is the most successful approach to speech recognition. One problem with this approach is that as the vocabulary of the speech recogniser is increased ie. large vocabulary isolated word recognition, there is a corresponding increase in computation time and memory requirements for template storage. The solution to this problem has been to base recognition at the sub-word level. The idea is to segment the word into smaller units and then to recognise the word as a sequence of these units.

Three approaches are most commonly used in implementing this method; segmentation of the word into syllable sized units, segmentation into phoneme-like units, and segmentation of the word into a sequence of broad phonetic units such as fricative, stop, nasal etc. The phoneme-like unit is chosen as the sub-word unit for the work presented in this dissertation.

To recognise the sub-word units, the two most common approaches are template matching by the dynamic time warping technique and hidden Markov models. The output from the speech recogniser is a string of labels corresponding to the sub-word units within the unknown word input to the recogniser. Recognition errors in the string of labels output by the recogniser may be corrected by a linguistic processor.

CHAPTER TWO

NEURAL NETWORKS IN SPEECH RECOGNITION

2.1 Introduction

The established models (state of the art) for basing the design of speech recognisers are the template matching and hidden Markov model approaches. These models are relatively simple descriptions of the speech recognition process, hence experience has shown that a limit in their performance is reached after which further improvements become very difficult to achieve. In the words of M. Allerhand [85]

> *"Basically these simple models are just not rich enough to describe a process as specialised as speech recognition."*

Researchers have therefore continued to explore alternative approaches to this problem.

Since human beings are the best pattern recognisers to date, many research groups have investigated if the cognitive processes of the brain may be achieved by models based on the structure of the brain [86], [87], [88]. Work on pattern recognising neural networks mainly began in 1943 with the McCulloch and Pitts Model of the neuron. The 1950s saw much interest in neural networks, one of the most popular being Rosenblatt's perceptron. In 1969, Minsky & Pappert demonstrated limitations in the perceptron approach, as a result of which interest in this area of research diminished. However,

researchers such as S. Grossberg, J.A. Anderson and J. Hopfield in U.S.A., E. Caianiello, T. Kohonen and I. Aleksander in Europe [89], and S. Amari in Japan, continued work on neural networks during the 1970s.

Recent work in the U.S. by Hopfield, Rumelhart and McClelland, Sejnowski, Grossberg and others [88], [89], in developing new network topologies and training algorithms has led to a resurgence of interest in the field of neural networks. Workers elsewhere have also made a significant contribution; Aleksander, Stonham & Wilkie's WISARD [6], a visual pattern recognition device capable of operating at TV frame rates [91], Kohonen's self organising feature maps [92], and K. Fukushima's neocognitron that can distinguish and read handwritten numbers [93]. The neocognitron can separate out and recognise 4-digit numbers in 0.8 second.

The year 1987 has seen the start of significant activity by the neural networks community at an international level. The first IEEE International Conference on Neural Networks was held in San Diego, and the International Neural Network Society was founded.

That the neural network approach is now established as a credible approach towards replicating human behaviour in machines can be seen from the fact that Japan's Ministry of International Trade and Industry (MITI) (of which S. Amari is a key adviser) is to announce the formation of a study group to formulate plans for a new national neurocomputer project expected to match the fifth generation computer project in scale [94]. Giant companies such as IBM Corp. and Texas Instruments Corp. have also invested in neurocomputers [86].

Most of the work on the speech recognition problem has been based on the algorithmic approach ie. to mathematically model the speech recognition process and implement it as a computer program on

a conventional computer. The disadvantage of this approach is that before a computer can be programmed to carry out a function, that function must first be understood and then an algorithm must be devised for implementing it. It is admitted that current knowledge on many aspects of speech recognition (as with other areas of pattern recognition) is incomplete (eg. acoustic-phonetics). Neural networks offer a way round this problem. They learn from experience (like humans do) and no programming is involved. Examples of the patterns to be recognised are input to the net and the net adjusts its parameters (ie. learns). After the net has learnt all the desired patterns, should a pattern similar to one it has "seen" during the learning phase be input to the net again, the net makes an association between the two. Currently, not much is known regarding the functioning of the brain. Neural network researchers (eg. Rumelhart & McClelland) hope that studies of the behaviour of neural nets may offer clues as to how the brain might function.

Recent books dealing with speech recognition [95], [96], [97], do not mention the alternative approach offered by neural networks. Holmes [98] gives it a very brief mention. The application of neural network models of Rumelhart & McClelland (the PDP group) and Kohonen to the speech recognition problem has enabled the neural network approach to be accepted as a genuine alternative to the traditional algorithmic approach to speech recognition.

2.2 Review of Neural Networks in Speech Recognition

Most of the initial work with neural nets dealt with pattern recognition problems related to vision (eg. character recognition). The

works of the PDP Group and Kohonen have been the most successful at demonstrating the effectiveness of neural networks in speech recognition.

(i) The TRACE Model

The TRACE model of McClelland & Elman [89] consists of a very large number of parallel processing units arranged at three levels; feature, phoneme and word levels (fig. 2.1). This architecture is motivated by the fact that perception experiments with humans indicate that different sources of information are used in recognising words and the phonemes within them. The input speech stream is divided into 5ms time slices. The model processes 500 ms of speech (100 time slices). Since speech patterns of phonemes are affected by context, McClelland & Elman prefer to identify phonemes by examining the speech stream for characteristic patterns rather than segment into separate units.

The *feature* level comprises a bank of feature detectors (acuteness, diffuseness, voiced etc.) at each time slice. At the *phoneme* and word levels, there are detectors for each phoneme and each word respectively. For each phoneme there are copies of phoneme detectors spanning several feature slices. These detectors are arranged so that they overlap. This procedure is repeated for the word level.

Processing is performed in the model by the excitatory and inhibitory interactions of the units. There are excitatory bidirectional connections between feature units and phoneme units consistent with those features, and between phoneme units and the word units representing words that contain those phonemes. Connections between units on the same level are inhibitory.

Phoneme identification is influenced by information from the feature level and also by the lexical effects produced by feedback from the word level to the phoneme level. When information from the feature level is lacking, the lexical level is able to compensate for this. It also enables the model to recover when the start of words are distorted. A difficult problem in speech recognition is to determine the word boundaries when the phoneme that a word ends with and the phoneme that the following word begins with are the same. TRACE is able to perform word segmentation in such cases.

The major drawback with TRACE is that the phoneme and word levels consist of copies of detector units overlapping in time, requiring the replication of connection patterns between features and phonemes, and between phonemes and words. Besides this there are the usual stumbling blocks for speech recognisers; speaking rate, speaker characteristics and noise.

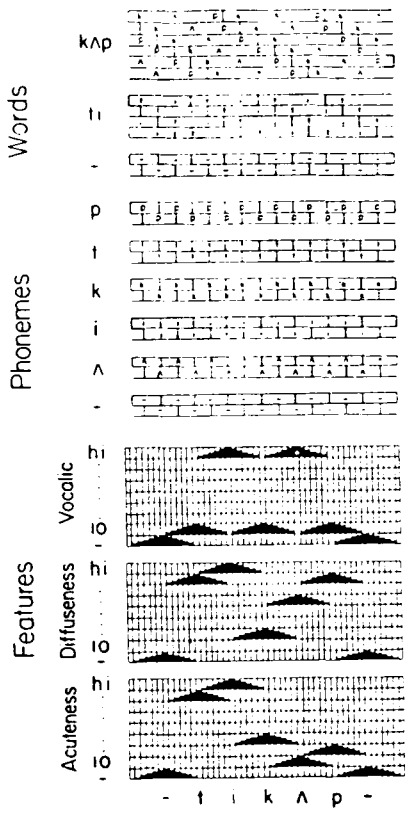


Figure 2.1 The three level architecture of the TRACE model

(ii) Kohonen's "Neural" Phonetic Typewriter

Kohonen's "neural" phonetic typewriter [99] is probably the only existing neural net based speech recogniser brought to a commercial stage. It uses an IBM PC AT as the host processor and can operate in one of two modes; transcribing dictation of unlimited text (92% to 97% accuracy), or isolated word recognition from a 1000 word vocabulary (word recognition accuracy of 96% to 98%). The basis for the design is the self-organising neural net model proposed by Kohonen. This is a two-dimensional array of output nodes (neurons) that adapts itself to input speech spectra in such a way that groups of nodes in different parts of the net become sensitised to the different phones in speech. The representations of the phonological features in speech by the network are called phonotopic maps [92].

The network is organised as a two-dimensional array of nodes. The external input to the net consists of 15 channel speech spectra. The input is connected in parallel to all nodes in the network enabling the input speech spectra to be applied simultaneously to all nodes. Lateral interactions between nodes is provided by feedback connections. The strength of these connections varies with distance as per the "Mexican hat" function.

To produce the phonotopic maps, all the weights from the external inputs to the nodes in the network are initialised to small random values. An input pattern vector (15 channel speech spectra) is applied to the net and by computing the distance between the input vector and the weight vectors for all nodes, the node that best matches the input is located. The weight vectors for this node and all other nodes in its neighbourhood (the size of this neighbourhood is predefined) are updated. The weight vectors of nodes outside this neighbourhood remain unchanged. The next input pattern vector is

presented to the net and the process is repeated. As more input vectors are presented to the net, the size of the neighbourhood of the best matching node is slowly decreased with time. After sufficient input vectors have been presented to the net (about 5,000), the net topologises itself in an orderly manner. The various nodes of the network become sensitised to spectra of different phonemes. In the speech spectral data, the spectra is clustered around phonemes and the self-organisation process in this network is able to find these clusters. The nodes are labelled by using spectra of known phonemes. If now unknown spectra are input to the net, the node with the weight vector closest to the input responds.

The net acts as a vector quantiser with the node weight vectors defining the vector quantisation of the input signal space. Comparison with vector quantisation (VQ) algorithm using K-means clustering have demonstrated that Kohonen's self organising feature map algorithm provides VQ codebooks superior to those created by the clustering algorithm [100].

An attractive feature of this neural network is that learning is unsupervised. Speech spectra are presented to the net and the self-organising process enables the net to adapt to the speech data. The net finds the relevant features in the speech signal by itself.

As a result of the recent interest in neural nets for speech recognition, other neural network models such as the Boltzmann machine [101], and single/multilayer perceptron [102], are being experimented with.

The temporal nature of speech is a major problem for speech recognisers. Dynamical networks are being assessed in this connection

[103]. The network alters its behaviour to try to follow the input and hence adapt to changes in the speech rate. Recurrent networks have been shown to be able to represent temporal dependencies internally [104].

The standard model of speech as used in LPC ignores nonlinearities in the speech production system. A multilayer neural network trained as a nonlinear predictor of the speech signal gives predictions with lower mean square error than LPC of the same order [105]. Furthermore, feeding the predicted samples back into the network enables the network to generate a signal with power spectrum similar to the spectrum of the original speech. This shows that the network is able to approximate the dynamical equations that produce speech.

In the next section the WISARD N-tuple pattern recogniser which is the basis for the speech recogniser designs presented in this work, is briefly reviewed.

2.3 WISARD Nets

Despite the disillusionment to research into neural networks caused by Minsky & Pappert's analysis of the perceptron, Aleksander and his colleagues continued to research towards a neural network based artificial vision system capable of working at TV frame rates. The result was the WISARD system (**WI**lkie, **S**tonham & **A**leksander's **R**ecognition **D**evice) [6], [7], capable of processing a 512 x 512 binary picture in real time. A review of work relating to artificial vision with single layer WISARD nets is made by Hassan [106].

The WISARD net is a hardware realisation of the N-tuple sampling method for pattern recognition proposed by Bledsoe & Browning [8]. Figure 2.2 shows the basic structure of a WISARD net based pattern recogniser. The binary pattern to be processed (ie. trained/classified) by the network is stored in a 2 dimensional binary array referred to as the 'retina' (so called since the WISARD system was designed as an artificial vision system). Each class of patterns to be recognised by the network is represented by a discriminator. Each discriminator is a single layer network of random access memories (RAMs). (Aleksander has shown that a RAM in terms of logical function is equivalent to a neuron [7], [107]). N bits are sampled from the 'retina' and grouped together to form an N-tuple. Each N-tuple sampled from the 'retina' has a corresponding RAM in each class discriminator, and the N-tuple is the address of a cell in that RAM. The number of RAMs required per class discriminator is numerically equal to the number of N-tuples needed to sample the whole 'retina'. Thus for a 'retina' size of x x y bits, and tuple size of n, the number of N-tuples required to map the whole 'retina' is $(x \times y)/n$. Each RAM in the class discriminator is a 2^n x 1 bit RAM (ie. the size of the RAM is dependent on the N-tuple size).

The mapping of the 'retina' into N-tuples may be done in one of two ways; linear or random mapping. For linear mapping, n consecutive bits are grouped to form the n-tuple. The next n-tuple would be formed from the next consecutive n bits and so on until the whole 'retina' has been mapped. For random mapping the n bits are sampled at random from all unmapped bits in the 'retina'.

The N-tuple size and number of patterns in the training set affects the classification performance of the net [108]. Experiments have shown that for a given N-tuple size, performance improves with

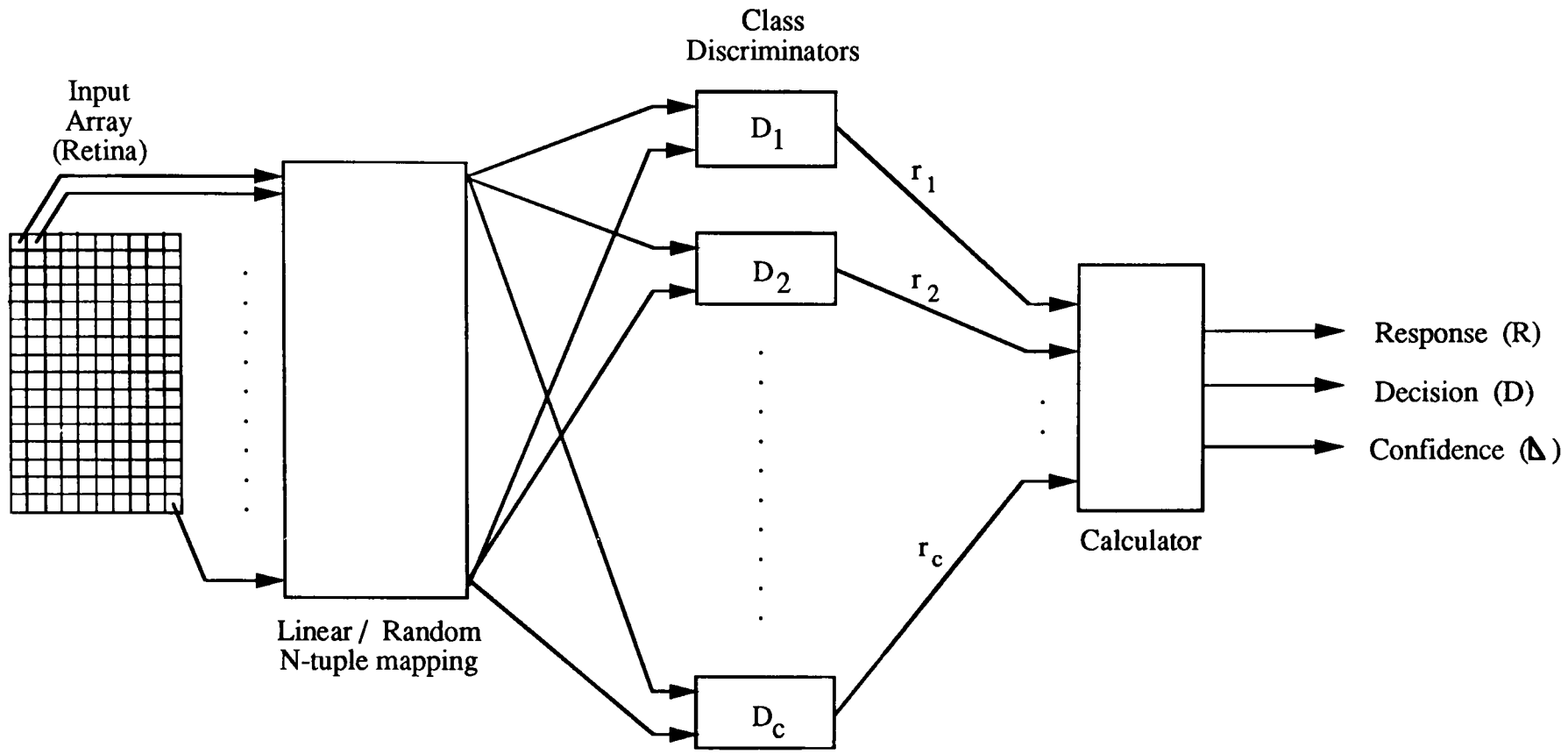


Figure 2.2 Single Layer WISARD Net Classifier

increase in size of the training set up to a certain point beyond which performance drops with further increase in the size of the training set. The net is said to have saturated. By increasing the N-tuple size the net can accommodate a training set of a larger size before saturation occurs. Performance improves with increase in N-tuple size but a larger training set may be needed (compared with that for a smaller N-tuple size).

An interesting aspect of the N-tuple method of sampling the patterns input to the network is that it enables the network to generalise and hence recognise patterns other than those in the training set [108]. The patterns that the network generalises are variations of those it has been trained on. This a useful feature to have in a speech recogniser.

Training the WISARD net is a simple and fast process. No complex training algorithms are needed as with most other neural network models (eg. error back propagation algorithm [88], and Kohonen's self organisation algorithm [92]). To train the net, initially, all class discriminators are cleared and then examples of patterns to be recognised are placed in turn, on to the 'retina'. The pattern on the 'retina' is mapped into N-tuples and a '1' is written to the RAM locations addressed by the N-tuples, in the discriminator assigned to the class to which the training pattern belongs. All training patterns belonging to a particular class are trained in to the discriminator assigned to that particular class. Provided the net does not saturate, as many patterns as necessary may be trained into the discriminator without requiring extra memory to accommodate these patterns. Also, the N-tupled representations of each pattern trained is retained in the discriminator. In contrast, a template matching based pattern recognition system requires all training patterns belonging to a

particular class to be averaged into a single template representing that class or else have a template for each training pattern in which case memory requirement increases with the number of templates.

To classify a pattern, the test pattern is placed on to the 'retina'. The 'retina' is mapped into N-tuples and the RAM locations addressed are read for each class discriminator. The score of a class discriminator for the test pattern is obtained by summing the number of 1's output by the RAMs in the discriminator. Thus the class discriminator whose N-tuple states are most like those of the test pattern will give the highest score. The test pattern is assigned the label of that class discriminator. This pattern classification scheme is both simple (as compared with other methods such as template matching) and extremely fast eg. a WISARD net with N-tuple size of 2 (ie. 2-tuple) can classify a 512 x 512 binary image in about 0.25 second. A WISARD net is therefore able to respond in real time. The response time depends mainly on the size of the 'retina' and on the RAM access time.

In general, for a WISARD net (single layer network) with c class discriminators, $R = \{r_1, r_2, r_3, \ldots . r_c\}$ is the set of responses (ie. scores) from all class discriminators to a pattern input to the network, then

(i) The response of the network, R, to the input pattern is the response of the class discriminator with the highest response.

$$R = \text{MAX}\ (R) = r_M$$

(ii) The decision of the network, D, ie. the class that the network identifies the input pattern with, is the identity of the class discriminator with the highest response (r_M).

$$D = M$$

(iii) The confidence of the decision, Δ, is the difference in response of the highest responding discriminator and the next highest one.

$$\Delta = r_M - \text{MAX}(R \cap \bar{r}_M)$$

The following section discusses the application of single layer WISARD nets in speech recognition.

2.3.1 Single Layer WISARD Nets in Speech Recognition

Most of the work with WISARD nets has been related to problems in artificial vision. In 1983, the research group of A. Badii and M.J. Binstead was initiated under the guidance of T.J. Stonham and A. Jones to investigate the recognition of isolated words by single layer WISARD nets [11]. Owing to resource constraints Badii & Binstead used a word-set comprising 16 isolated utterances from a single speaker to evaluate their isolated word speech recogniser. These utterances were chosen such that there were sets of rhyming words which would be confusable to the recogniser [12]. The input to the recogniser consisted of 8-bit samples from a 19 channel filter bank (FFT). Each word was treated as a separate class thus there were 16 discriminators in the recogniser. An average word recognition accuracy of 94.25% was achieved for this word-set with an 8-tuple WISARD recogniser. For comparison purposes, results were also obtained for recognition of this word-set using the conventional DTW based template matching scheme. 89.2% average word recognition accuracy was obtained for this approach. Thus significantly better recognition performance was achieved with the WISARD net based speech

recogniser. This was despite the fact that no time normalisation was incorporated into the WISARD net speech recogniser. Furthermore, the WISARD approach has virtually no computational overhead compared with that required for DTW.

Experiments were also performed for vowel detection [12]. The vowel samples were obtained from a 16-channel filterbank every 5 ms. The vowel samples in the set of utterances used for evaluating the WISARD net vowel detectors were handcrafted. After training the vowel detectors using these samples, classification was achieved by sliding a window across the test utterance and observing the response of the discriminators (vowel detectors). The idea is that when the sliding window encounters samples corresponding to vowels in the utterance, the discriminator trained to detect that vowel would respond strongly. The discriminators' response in non-vowel regions of the utterance would be low. To account for variations in the duration of the vowel samples, each vowel detector comprised six discriminators with each discriminator windowing a different length of time. 61.8% vowel recognition accuracy is reported.

The results obtained for isolated word recognition using WISARD nets are comparable with those reported for other methods such as template matching, and suggest it to be a useful alternative approach to speech recognition deserving further study. The work presented in this dissertation is also aimed at isolated word recognition from the word-set of Badii & Binstead but through the sub-word approach (the Badii & Binstead isolated word recogniser is based on the whole word matching approach). The advantages of the sub-word approach over the whole word matching approach were discussed in Chapter 1.

The WISARD net has features that make it suitable for use in speech recognisers. The training procedure is very simple (examples of

speech patterns to be recognised need only be presented to the net) and the nets can be trained very quickly (no iterative training algorithms required). However, training has to be supervised ie. the speech patterns to be trained must be labelled so that it can be trained into the correct discriminator. WISARD nets are easy to implement in hardware and at a low cost, eg. for the speech recogniser designs proposed in Chapters 4 and 5, the 22 class discriminators representing the phoneme-like segments to be recognised can be accommodated in a 1 Mbit RAM. The low computational overheads in classifying patterns will enable the speech recogniser real time response. The generalisation feature may enable the speech recogniser to cope with some of the variabilities in the input speech patterns. The fact that many training patterns belonging to the same class can be trained in to the same discriminator is particularly useful for implementing the sub-word approach to speech recognition. It is known that the context in which a phoneme occurs affects its spectral patterns, thus the recogniser would need to be trained with examples of the contextual variations of each phoneme to be recognised. With a WISARD net based recogniser, all contextual variations of a phoneme may be trained into a single discriminator assigned to that phoneme, and the discriminator will retain the N-tupled representation of each of these training patterns. This would not be possible in a template matching system without multiple templates for each phoneme. Also, the number of RAMs required per class discriminator is independent of the number of examples in the training set (provided the net does not saturate).

The following chapters of this dissertation discuss two designs using the single layer WISARD net for isolated word speech recognition based on the sub-word approach.

2.4 Summary

Limitations in the performance of speech recognisers based on conventional algorithmic approaches is due to the fact that current knowledge of the speech recognition process is incomplete, thus the models used are simple descriptions of the speech process. The neural network approach requires no programming. Examples of the speech data to be recognised are presented to the network and the network adapts to this data (ie. learns). A learning algorithm enables the network to find the relevant features in the speech signal.

The work of McClelland & Elman (TRACE) and Kohonen (self organising network) has established neural networks as a serious alternative approach to achieving speech recognition in machines. Currently various neural network models are being applied to different aspects of speech recognition.

The WISARD neural network model is an implementation in hardware of the N-tuple method for pattern recognition first proposed by Bledsoe & Browning. It has features that make it suitable for use in speech recognition. The training procedure is simple and fast since the iterative learning algorithms needed for training most other neural networks are not required. The net is able to respond in real time to input patterns since very little computation is required in the classification process. The net's generalisation feature may enable it to cope with small variations in the input pattern (in relation to the patterns it was trained on). The cost of implementing the WISARD net in hardware is low.

CHAPTER THREE

SEGMENTATION OF ISOLATED WORDS

3.1 Introduction

To perform isolated word recognition by the sub-word unit based approach, it is first necessary to partition the word into segments corresponding to the chosen sub-word unit. Phoneme-like segments are chosen as the sub-word unit for the work presented in this dissertation. There are two approaches to the segmentation problem :

(i) Some measurements are performed on the spectral data for the utterance and the spectral frames where these measurements exceed a preset threshold are marked as boundaries of the phoneme-like segment.

(ii) The spectral data from the utterance is considered as frames over a fixed time interval (commonly 10 ms). Each frame is labelled as belonging to a phoneme-like segment. Frames with the same label are grouped into a single segment.

In this chapter, an improved method for automatic segmentation of isolated utterances into segments of the size of phoneme-like units based on the first approach mentioned above, is presented.

Listening to a word being spoken, the phones are clearly discernable. Therefore on examining a spectral representation of the

word, it is expected to observe a similarity between the spectral patterns corresponding to a particular phone, and that the spectral patterns would change as the utterance proceeds into the following phone. These observations suggest the use of distance calculation between successive spectral frames as a measure of their dissimilarity, which is then compared to a distance threshold. The instant in time when this measure exceeds the threshold value is marked as a segment boundary (Beninghof & Ross [51]). The approach of Itahashi *et al* [36] is also based on this idea, except that the distance calculations are performed on a set of nine features extracted from filters covering formant regions. They calculated the Euclidean distance between consecutive feature frames, as defined by (3.1).

$$d_t = \{\sum_{i=1}^{9} (f_{ti} - f_{t-1,i})^2\}^{1/2} , \qquad t = 1,2,..... \qquad (3.1)$$

where f_{ti} and $f_{t-1,i}$ denote the ith feature at frame time t and $t-1$.

In the same way, a Euclidean distance measure can be defined to calculate the spectral distance between consecutive spectral frames obtained from a filterbank, which can then be used to segment speech utterances. First however, the word-set of Badii & Binstead which was used as the database to test the segmentation technique, and later on, the performance of the phoneme-based WISARD n-tuple isolated word speech recogniser, is introduced.

3.2 The Word-Set

The word-set contains sixteen words [11], [12]. The particular words were chosen so as to produce groups of similar sounding words

similar sounding words that would be difficult to recognise since they are confusable. This aspect of the word-set makes it a suitable database for phoneme-based recognition experiments. The recogniser can be trained on a phoneme-like segment from an utterance of a particular word, and tested not only on other utterances of the same word, but also on different words containing the same phoneme. The word-set is as follows:

Win	Cooler	Two	Tee
Want	Rudder	Shoe	See
One	Wonder	Tattoo	Three
Run		Toot	
Begun		Toothache	

The database as obtained from Badii & Binstead consisted of digitized time samples of fifty isolated utterances of each of these sixteen words spoken in an acoustic chamber. The utterances were bandlimited to 12.5 KHz and digitized by a 16-bit A/D converter at a sampling frequency of 25 KHz.

For the purpose of this work, frequency domain representation of each isolated utterance was required. This was achieved by outputting the digitized speech samples through a 16-bit D/A converter connected to the input of a 16-channel filterbank IC (ASA-16). The filterbank channel outputs were sampled every 0.64 ms by an 8-bit A/D converter, and every five spectral frames were averaged into one frame in order to smooth the data. Thus each averaged spectral frame corresponds to 3.2 ms of speech. In this way, spectral data consisting of a sequence of such averaged spectral frames were obtained for each utterance.

3.3 Euclidean Distance Measure for Spectral Dissimilarity

Considering spectral data from a filterbank with n channels, a spectral frame at sample time t, f_t, is represented by a point in an n-dimensional vector space, the co-ordinates being given by the filterbank channel energies [3], [51]. The spectral dissimilarity between two frames f_t and f_{t-1} may now be given by redefining the Euclidean distance given by (3.1) as

$$d_t = \{\sum_{c=1}^{16} (f_{tc} - f_{t-1,c})^2\}^{1/2} , \qquad t = 1,2,.... \qquad (3.2)$$

where c denotes the filterbank channel number.

This Euclidean distance measure has been applied to the spectral data of isolated utterances from the word-set of Badii & Binstead, obtained from a 16-channel filterbank. Plotting spectral distance between consecutive frames versus time results in plots similar to those obtained by Beninghof & Ross [51] (although they used the mean-square distance measure). The plot contains peaks and valleys, where the valleys indicate regions with similar spectral frames, and the peaks show dissimilar frames. Peaks above a threshold value are candidates for segment boundaries since spectral frames are most dissimilar at these points, indicating transition in the sound. To confirm this notion, time samples of the utterance corresponding to spectral frames within segments obtained by this segmentation process were played back through a D/A converter connected to an audio amplifier and speaker system. It was found that most of the segments corresponded to phones in the word segmented, thus the peaks indicate acoustic boundaries.

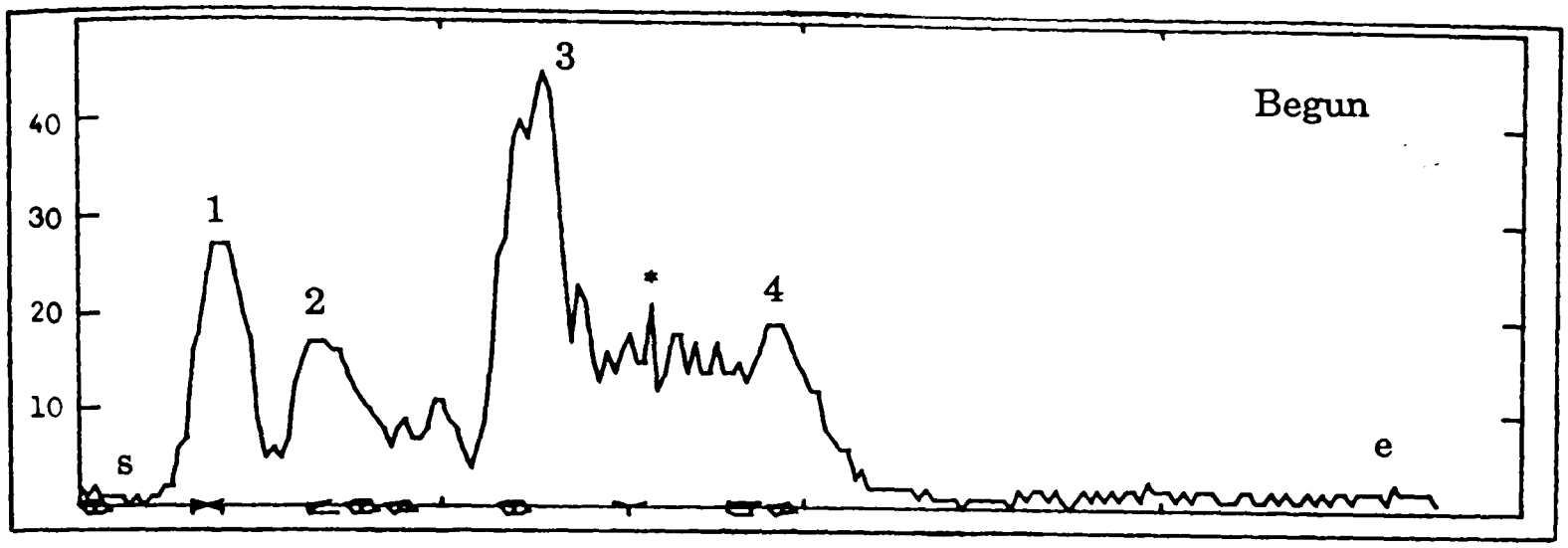


Figure 3.1 Spectral Euclidean distance *vs* Time

This method seems to be an effective technique for segmenting speech into phoneme-like units. It requires no *a priori* data; the information in the spectral data is sufficient to perform segmentation. Also, the distance calculation does not require much computation. There is however a problem in that not all the peaks detected are phone boundaries, as evidenced by the plot of spectral Euclidean distance versus time for the word 'Begun', shown in figure 3.1 .

Referring to figure 3.1, 's' marks the start of the utterance, and 'e' the end of the utterance. Peaks marked '1' to '4' are phone segment boundaries. The segments correspond to phones as follows:

Phone	Segment
/b/	's' - '1'
/ɪ/	'1' - '2'
/g/	'2' - '3'
/ʌ/	'3' - '4'
/n/	'4' - 'e'

These peaks may be easily located by applying a suitable distance threshold. However from figure 3.1, it is clear that there is a problem with this method in that other peaks such as the one marked '*' will also be picked since it is higher than the threshold, causing a segmentation error (on the diagram, only one peak is marked as an example representative of other peaks that may be marked as segment boundaries by an automatic segmentation algorithm based on the method being discussed).

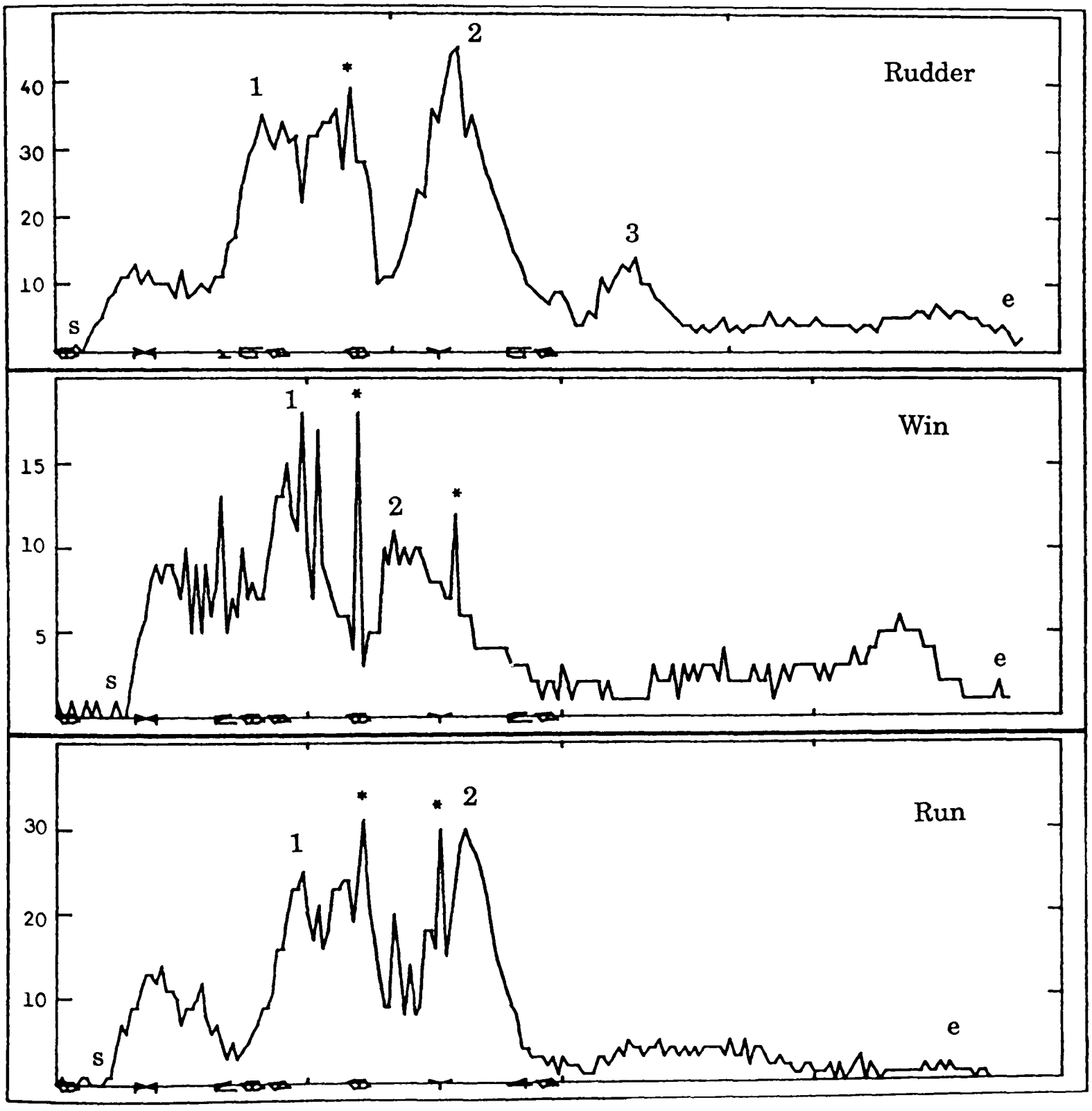
Figure 3.2 shows some examples of this type of error in other words ('Rudder', 'Win' and 'Run'). The phone segments are as follows:

Rudder		*Win*		*Run*	
Phone	Segment	Phone	Segment	Phone	Segment
/r/	's' - '1'	/w/	's' - '1'	/r/	's' - '1'
/ʌ/	'1' - '2'	/ɪ/	'1' - '2'	/ʌ/	'1' - '2'
/d/	'2' - '3'	/n/	'2' - 'e'	/n/	'2' - 'e'
/ə/	'3' - 'e'				

The simplicity of this method and its reasonably good performance (in the sense that although there are extraneous peaks, the phone boundaries are also represented) at locating possible phone boundaries suggested that effort be made to make it robust to segmentation errors of the type discussed. One possible approach is to incorporate heuristics into the segmentation process [53]. These rules are formulated from observations of the word samples and would determine when a peak that exceeded the threshold may be accepted as a segment boundary. This option was tried but it was found to be difficult to cope with the variabilities in the speech samples using *ad hoc* rules at this stage. The approach was therefore to try different

distance measures and modifications to these distance measures, to investigate if these unwanted peaks could be eliminated or at least reduced in number. If this can be achieved, heuristics applied at this stage would be easier to formulate and better results could be expected since the confusion caused by the unwanted peaks is reduced.

Figure 3.2 Spectral Euclidean distance *vs* Time



3.4 Weighted Euclidean Distance Measures

The Euclidean distance measure was modified such that the lower frequency channels had a higher weighting. This is because the lower frequencies are more important than the higher frequencies for intelligibility (Beninghof & Ross [51]). The weighted Euclidean distance measure with higher weighting given to the lower frequency channels of the 16-channel filterbank is given by

$$d_t = [\ \sum_{c=1}^{16} \{(17 - c)(f_{tc} - f_{t-1,c})\}^2\ ]^{1/2}\ ,\quad t = 1,2,..... \qquad (3.3)$$

Figure 3.3 shows the results of plotting spectral distance by lower-frequency-weighted Euclidean distance versus time. Comparison with figures 3.1 and 3.2 show no improvement as regards the removal of the unwanted peaks.

A weighted Euclidean distance measure with higher weighting to the higher frequency channels of the filterbank, given by (3.4) was also tried. Results obtained with this distance measure (figure 3.4) show that it makes matters worse (as was expected).

$$d_t = [\ \sum_{c=1}^{16} \{c(f_{tc} - f_{t-1,c})\}^2\ ]^{1/2}\ ,\quad t = 1,2,..... \qquad (3.4)$$

Figure 3.3 Spectral low-frequency-weighted Euclidean distance *vs* Time

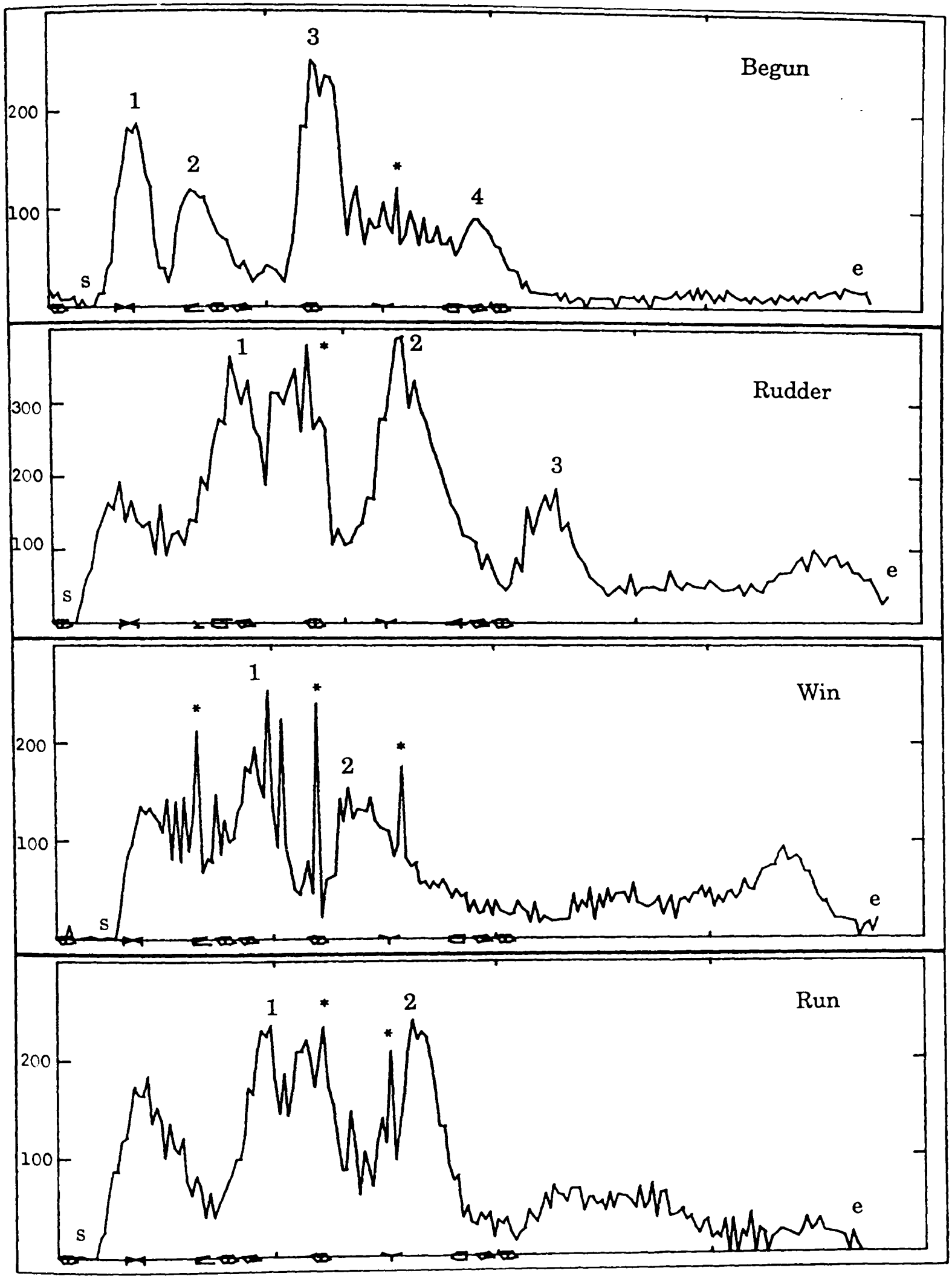
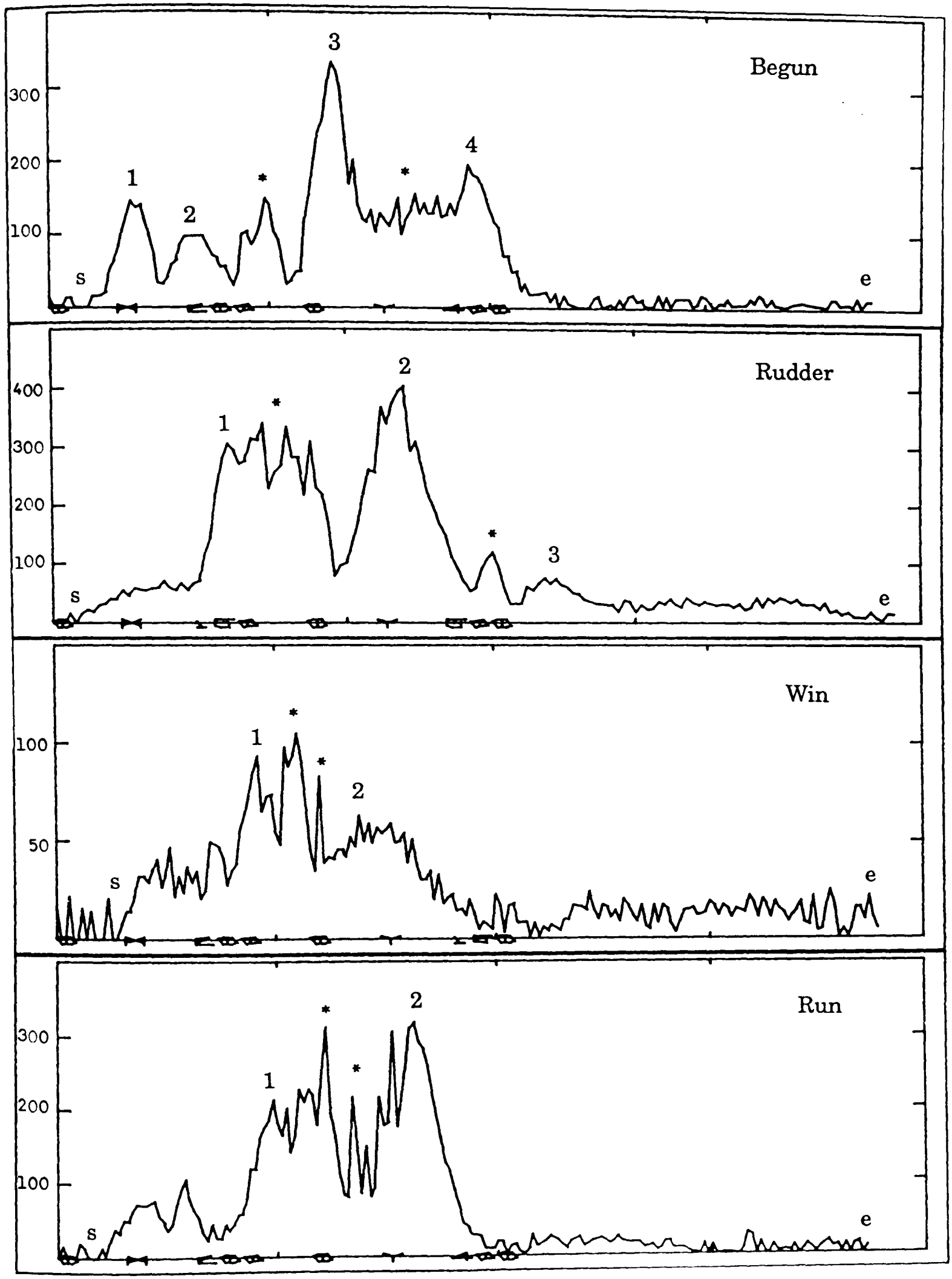


Fig. 3.4 Spectral high-frequency-weighted Euclidean distance *vs* Time



3.5 Chebyshev Distance Measure for Spectral Dissimilarity

Since the Euclidean distance measures failed in improving the segmentation errors, another spectral distance measure, the Chebyshev distance was tried. The Chebyshev distance (also known as *city-block distance*) is defined as

$$d_t = \sum_{c=1}^{16} |(f_{tc} - f_{t-1,c})| \ , \qquad t = 1,2,..... \qquad (3.5)$$

The spectral distance plots using the Chebyshev distance as the measure of spectral dissimilarity are shown in figure 3.5 . Though the unwanted peaks are still present, these results are an improvement over those obtained using the Euclidean distance measures (figures 3.1 - 3.4). The peaks marking the segment boundaries are more prominent than in the previous plots as the unwanted peaks are reduced in magnitude (relative to the segment boundary peaks). Another advantage is that the Chebyshev distance is computationally less expensive than the Euclidean distance since the squaring operations needed in the calculation of the Euclidean distance are avoided.

3.6 Sum of Channel Difference as a Spectral Dissimilarity Measure

The Chebyshev distance is the sum of the magnitudes of the differences in channel energies at frame times *t* and *t-1*. For the next experiment this distance measure was modified so that only the sum of the differences in channel energies over consecutive frames were

considered. This results in what will be referred to as the 'sum of channel difference' measure, defined in (3.6) as

$$d_t = \sum_{c=1}^{16} (f_{tc} - f_{t-1,c}) , \qquad t = 1,2,..... \qquad (3.6)$$

Figure 3.6 shows the spectral dissimilarity plots obtained using this spectral distance measure. These results are clearly better than all the previous results in achieving the objective of reducing confusion in the segmentation process due to the extraneous peaks. In this case, the phone segment boundaries are marked by the peaks and troughs on the plot. A characteristic of these plots is that the location of the segment boundaries alternates about the time axis. This greatly simplifies the process of locating these boundaries. The first boundary always occurs in the positive half of the plot because the filterbank channel activity increases as the utterance begins, and is marked by the highest peak. The next boundary is in the negative half of the plot, and is the point with the largest negative value. This process repeats for the remaining segments in the utterance. In this way, the unwanted peaks that occurred in the plots obtained using the Euclidean and Chebyshev distance measures, have been rejected since they are lower in magnitude than the peaks that indicate the segment boundaries.

Another advantage of the 'sum of channel difference' measure over the Euclidean and Chebyshev distance measures is that of the three, it requires the least computation.

Figure 3.5 Spectral Chebyshev distance *vs* Time

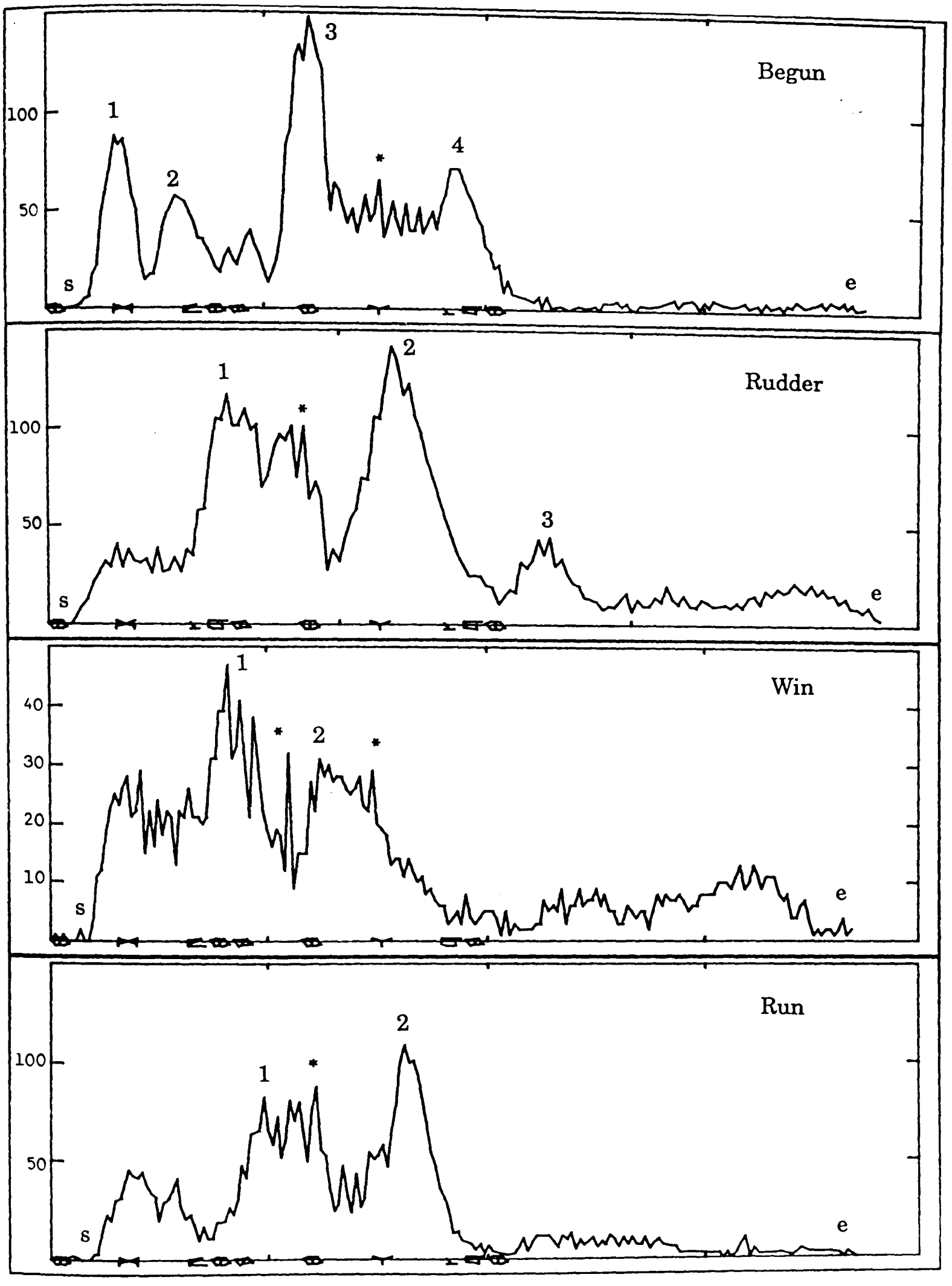
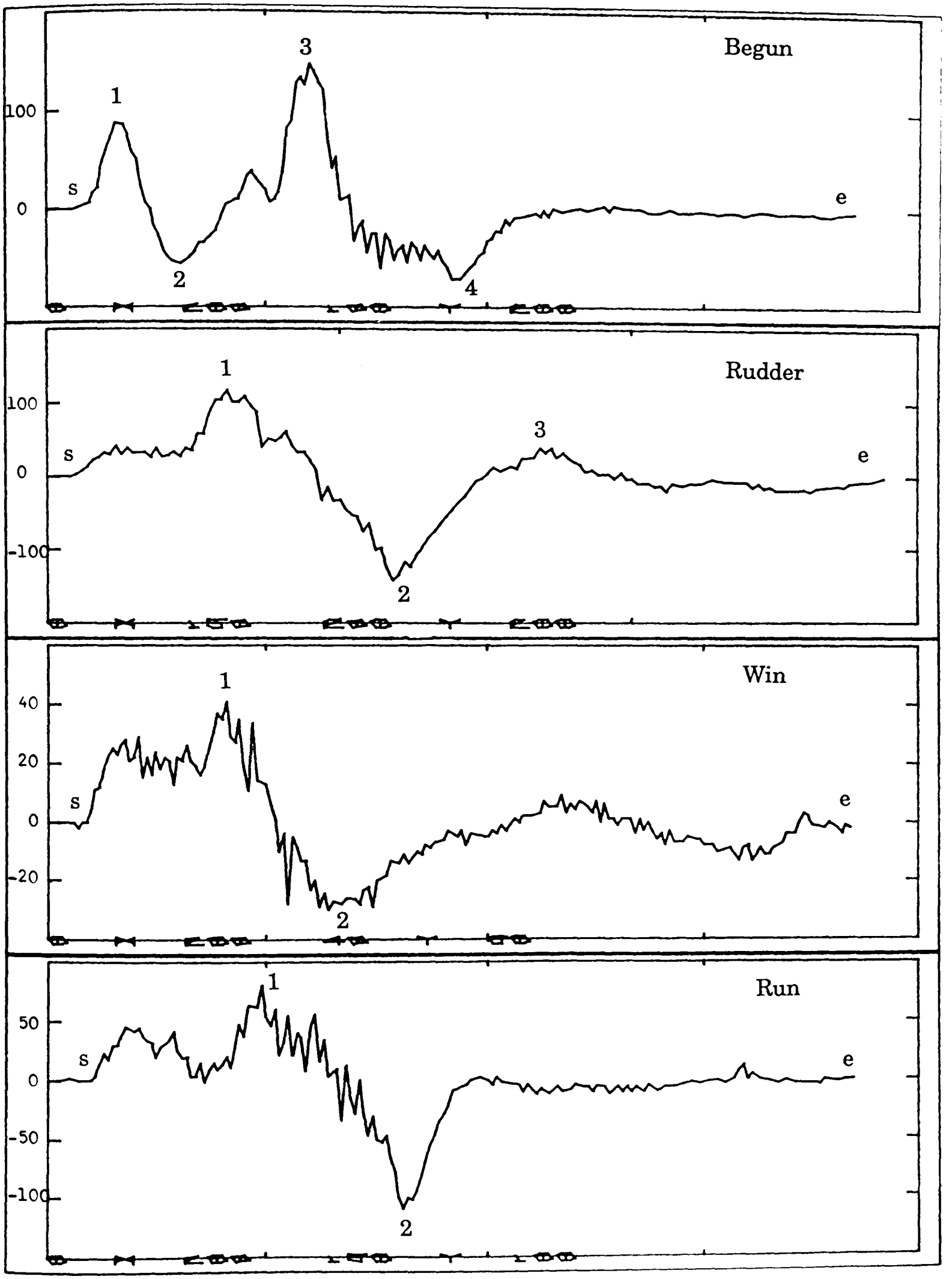


Figure 3.6 Sum of channel difference *vs* Time



3.7 Segmentation of the Word-set

The results of applying the segmentation technique using the 'sum of channel difference' measure to the utterances in the word-set are presented in this section.

Although the rule is that segment boundaries alternate about the time axis, the only exception (ie. for the utterances in the word-set) was found in the segmentation of the word 'Wonder', as illustrated in figure 3.7. This concerns the location of the boundary marked '3', which is the boundary separating the phones /n/ and /d/. To detect this boundary, a rule was incorporated into the segmentation process based on the behaviour of the spectral distance measure as observed in figure 3.7. To consider the point '3' as a boundary, firstly, the magnitude of the 'sum of channel difference' at this point must be above the threshold, and secondly, some of the spectral frames before this point must be similar ie. the spectral distance is small. This indicates that they represent the same sound and therefore a change in the sound is occurring as the point in question is approached.

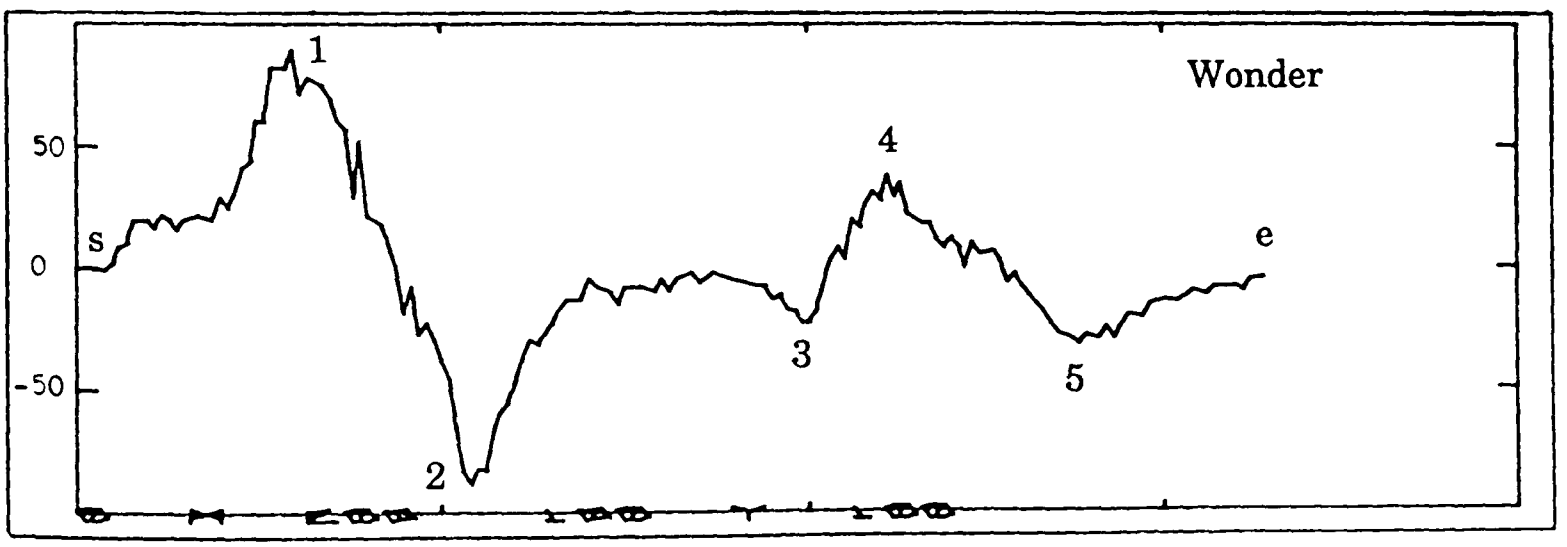


Figure 3.7 Segmentation of the utterance 'Wonder'

A problem that occurs with the segmentation of sustained vowels such as /i/ in 'See', /u/ in 'Two', /eɪ/ in 'Toothache' and /ə/ in 'Wonder', using this segmentation technique is that they are split into two segments in almost 30% of the occurances of these vowels. An example of this can be seen for the segmentation of 'Wonder' in figure 3.7 . The segment '4' - 'e' represents /ə/ but has been split into two segments at '5'. This type of segmentation error, called an insertion error, is not peculiar only to this segmentation technique as other researchers have had similar experience [109]. However, this is not a grave problem as in the training phase of a speech recogniser, these segments can be tagged as being training samples of the same vowel, and in the recognition phase, allowance is made so that two consecutive segments can be recognised as belonging to the same vowel. By approaching the problem in this way it is now possible to consider the insertion of a segment boundary in a sustained vowel segment as an acceptable segmentation result and therefore is not treated as a segmentation error.

The application of the 'sum of channel difference' segmentation method to the word-set of Badii & Binstead produces the twenty-two segment categories listed in Table 3.1. With the exception of the utterance 'Three', all the words in the word-set are segmented into phoneme-like units and can be represented by twenty phonemes. The remaining two segment categories are obtained from the segmentation of 'Three' and will be represented in this work as /Θr/ and /ri/. These segments contain parts from adjacent phonemes in the utterance. The segment /Θr/ consists of the voiceless fricative /Θ/ at the start of the utterance and part of the approximant /r/. The remaining portion of /r/ is combined with part of /i/ to give the /ri/ segment. 'Three' was also segmented as "/Θr/ /i/" (Table 3.2).

Table 3.1 Segmentation results for the 'sum of channel difference' segmentation method

Segment	Error (%)	Deletion (%)	Insertion (%)
/b/	10.20	8.16	0.00
/ʧ/	21.00	17.00	0.00
/g/	8.16	6.12	0.00
/ʌ/	17.67	1.20	0.00
/n/	33.33	22.67	0.00
/k/	13.00	10.00	1.00
/u/	15.33	13.00	0.30
/l/	68.00	68.00	0.00
/ə/	35.33	35.33	0.00
/r/	10.00	6.00	0.00
/s/	0.00	0.00	0.00
/i/	0.67	0.67	0.00
/t/	6.75	5.50	1.00
/w/	12.50	12.00	0.00
/ʃ/	52.00	4.00	44.00
/ɔ/	30.00	28.00	0.00
/æ/	10.00	8.00	0.00
/Θ/	22.00	14.00	6.00
/d/	52.00	52.00	0.00
/eɪ/	6.00	4.00	0.00
/Θr/	8.00	2.00	6.00
/ri/	2.00	2.00	0.00
Average :	19.73%	14.53%	2.65%

The twenty-two segment categories are treated as twenty-two separate classes for the purpose of recognition. Table 3.1 gives the segmentation results for the 'sum of channel difference' segmentation method applied to the utterances from the word-set of Badii & Binstead. The performance of the segmentation method is tabulated in terms of the segmentation errors. The following conditions were considered to be segmentation errors :

(i) Incorrect placement of the segment's boundaries.
(ii) Deletion of a segment boundary causing the segment to be merged with the adjacent segment.
(iii) Insertion of a boundary within a segment causing the segment to be split into two segments.

The 'Error' column in Table 3.1 is the overall error rate ie. it includes errors from all three error conditions mentioned above. The 'Deletion' and 'Insertion' error columns show the deletion and insertion errors in the overall error rate. 2697 segment samples were used to evaluate the performance of the segmentation technique.

The proposed segmentation technique has an average overall error rate of 19.73%, most of which are segment deletion errors (14.17%), and the average segment insertion error rate is 2.65%. Thus the average error rate from incorrect boundary placements is 2.91%. These results compare favourably with other segmentation methods, eg. Charbonneau & Moussa [52] have reported segment deletion error of 19% and segment insertion error of 4% for their phoneme segmentation method based on the multi-dimensional scaling technique.

There were no errors in the segmentation of /s/ while the poorest results are those for /l/ and /ə/ in the word 'Cooler' (due to deletion of

boundary between /ʊ/ and /l/, and /l/ and /ə/), /n/ and /d/ in 'Wonder' (deletion of boundary between /n/ and /d/), and /ʃ/ in 'Shoe'. /ʃ/ is the only segment that suffers significant insertion errors. Sustained vowels /ʊ/, /i/, /eɪ/ and /ə/ are exempted from segment insertion errors as these are taken into account in the training and testing procedures of the speech rescogniser.

These results are on the whole considered as encouraging keeping in mind the simplicity of the segmentation technique, but the segment deletion error rate needs to be improved to enable its use in a practical speech recognition system. The following suggestions are therefore made for improving the segmentation accuracy.

(i) Using an automatically adjustable threshold [36] for segment boundary decision will help to reduce the segment deletion errors. Some indication of the intensity of an utterance may be used to choose a suitable threshold level.

(ii) Complementing the segmentation process with more information from the acoustic signal, such as energy [22], and zero-crossing measurements [110].

(iii) Complementing the segmentation process with heuristics [53] to determine if peaks detected may be accepted as segment boundaries.

(iv) Co-operation from the speaker in producing clearly articulated utterances (tired sounding utterances are very likely to cause segmentation errors).

Table 3.2 Segmentation of the words in the word-set

Win :	/w/,/ɪ/,/n/
Want :	/w/,/ɔ/,/n/,/t/
One :	/w/,/ʌ/,/n/
Run :	/r/,/ʌ/,/n/
Begun :	/b/,/ɪ/,/g/,/ʌ/,/n/
Cooler :	/k/,/u/,/l/,/ə/
	/k/,/u/,/l/,/ə/,/ə/
Rudder :	/r/,/ʌ/,/d/,/ə/
	/r/,/ʌ/,/d/,/ə/,/ə/
Wonder :	/w/,/ʌ/,/n/,/d/,/ə/
	/w/,/ʌ/,/n/,/d/,/ə/,/ə/
Two :	/t/,/u/
	/t/,/u/,/u/
Shoe :	/ʃ/,/u/
	/ʃ/,/u/,/u/
Tattoo :	/t/,/æ/,/t/,/u/
	/t/,/æ/,/t/,/u/,/u/
Toot :	/t/,/u/,/t/
	/t/,/u/,/u/,/t/
Toothache :	/t/,/u/,/ϴ/,/eɪ/,/k/
	/t/,/u/,/ϴ/,/eɪ/,/eɪ/,/k/
Tee :	/t/,/i/
	/t/,/i/,/i/
See :	/s/,/i/
	/s/,/i/,/i/
Three :	/ϴr/,/ri/,/i/
	/ϴr/,/ri/,/i/,/i/
	/ϴr/,/i/
	/ϴr/,/i/,/i/

The word-set of Badii & Binstead in terms of the labels assigned to the twenty-two segment categories is given in Table 3.2 (Appendix A gives the sound in an utterance that these segment labels represent). Of these twenty-two segment categories, twenty correspond to phonemes. The remaining two segment categories are partial combinations of two phonemes (/Θr/ and /ri/) from the segmentation of the word 'Three'. Alternative segmentations are shown for words containing sustained vowels. For convenience, these twenty-two segment categories will be referred to in the following chapters of this work as phoneme-like segments while it is acknowledged that two of these segment categories (/Θr/ and /ri/) are not in fact phoneme-like.

The recognition of the words in the word-set as a sequence of the segments indicated in Table 3.2 is dealt with in Chapters 4 and 5. The segmentation errors were corrected before passing the segments to the speech recogniser so that the error rate of the recognition process could be evaluated.

3.8 Summary

To recognise an utterance as a sequence of phoneme-like units, a method for picking out the phoneme-like units in the utterance is required. One approach, which is presented in this chapter, is to determine the boundaries of the phoneme-like units within the utterance based on some spectral measurements. Spectral frames where these measurements exceed a preset threshold are marked as a boundary. All spectral frames within these boundaries are associated with that particular phoneme-like unit.

An improved segmentation method based on the idea of spectral dissimilarity measurements as proposed by Beninghoff & Ross is suggested in this chapter. The method is to sum the difference in channel energies for consecutive spectral frames in the utterance. Frames where this measurement exceeds the threshold level are marked as the segment boundaries. Testing this segmentation method on utterances from the word-set of Badii & Binstead produced 22 segment categories. All except two of these segment categories are phoneme-like. The two exceptions were from the segmentation of the word 'Three' in the word-set. This segmentation method has an overall error rate of 19.73% from which 14.53% are segment deletion errors and 2.65% are segment insertion errors. The remaining 2.91% errors are due to incorrect boundary placement.

CHAPTER FOUR

RECOGNISER DESIGN BASED ON THE SEPARATE SEGMENTATION AND LABELLING APPROACH

4.1 Introduction

A design for a WISARD n-tuple isolated word recogniser based on the separate segmentation and labelling (SS&L) approach is presented in this chapter. In this method the utterance is first segmented into phoneme-like segments. Each segment is then labelled in turn by the recognition process. The output from the recogniser is a string of labels. Each label in the string corresponds to a phoneme-like segment in the utterance input to the recogniser.

Figure 4.1 shows the stages in the design of the speech recogniser. These stages are discussed in the following sections.

4.2 Acoustic Analysis

This stage consists of the ASA-16 integrated circuit, which is a 16-channel filterbank. The input speech signal is transformed by the filterbank into frequency spectral frames as explained in Chapter 3, Section 3.2. The filterbank is sampled every 0.64 ms by an 8-bit A/D converter, and every five frames are averaged into one frame to smooth the spectral data. Thus each averaged frame represents 3.2 ms of speech.

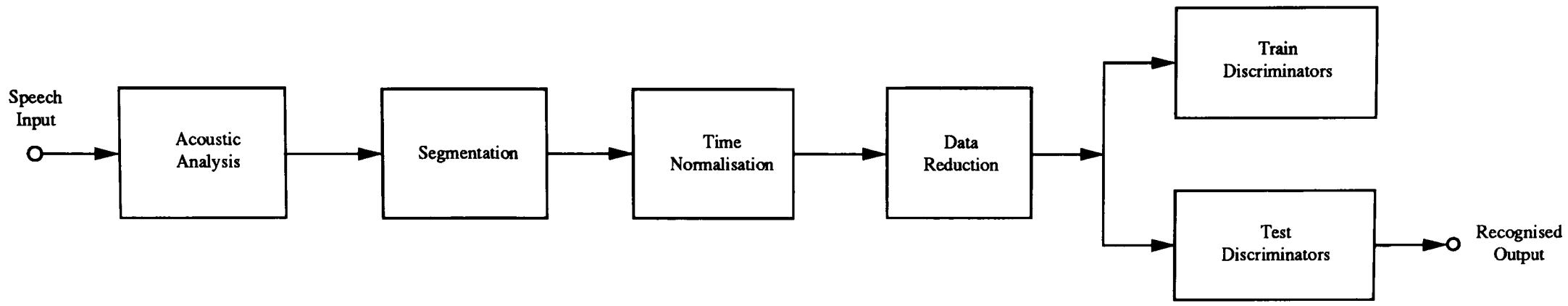


Figure 4.1 The stages in a speech recogniser based on the separate segmentation and labelling approach

4.3 Segmentation

This stage applies the segmentation technique using the 'sum of channel difference' spectral distance measure proposed in Chapter 3, Section 3.6, to the averaged spectral frames from the acoustic analysis stage. It outputs the frame number of the segment boundaries detected.

4.4 Time Normalisation

Owing to time-scale variabilities as a result of variations in the speaking rate, segments of a phoneme from different utterances have different durations. Furthermore, by nature, some phonemes such as /i/ and /u/ have longer durations than others like /b/ and /t/. This necessitates a time normalisation scheme. The idea is to stretch or squash the segments to a standard length. There are two methods [35]:

i). Linear time normalisation.
ii). Dynamic time warping (DTW).

DTW, a dynamic programming technique [50], is the superior technique and is now the established method for performing time normalisation of speech data. Whereas the linear time normalisation method performs uniform distortion of the time scale, DTW is a non-linear time normalisation technique performing selective distortion by allowing different parts of the segment to be shortened or lengthened more than others, which is what happens when a talker speaks faster or slower. It performs the optimum alignment between the unknown

and reference templates. However, it is difficult to implement with a WISARD system because both the unknown and reference patterns should be of the same type, eg. spectral data. In the WISARD system, the reference data which is stored in the discriminators consists of n-tupled data whereas the unknown input segment consists of spectral data. Thus the unknown input and reference data are of different types.

The linear time normalisation method is easily implementable with an n-tuple system since it does not involve the reference data. It operates on the data in the unknown input segment only. For this reason, it was chosen as the time normalisation technique in the design of the WISARD speech recogniser. It also requires less computation than DTW.

4.4.1 Linear Time Normalisation

This method uses the beginning and end points of the phoneme-like segment (ie the segment boundaries) to divide the segment into a fixed number of equal intervals. The frame sample values at these intervals will now represent the segment. In this way, all phoneme-like segments can be converted into the same number of frames.

It was chosen to time-normalise the phoneme-like segments into 20 frames. This was because the smallest segments in the word-set consist of 9 frames whereas the larger segments average around 40 frames thus 20 frames seems an appropriate size since it is around twice the smallest segment, and half the larger ones. As each frame comprises data from each of the filterbank's 16 channels, the time-normalised phoneme-like segment is representable by a 16 x 20 matrix.

The frame channel sample values at the time-normalised intervals within the segment are calculated by the linear interpolation method.

4.4.1.1 Linear Interpolation

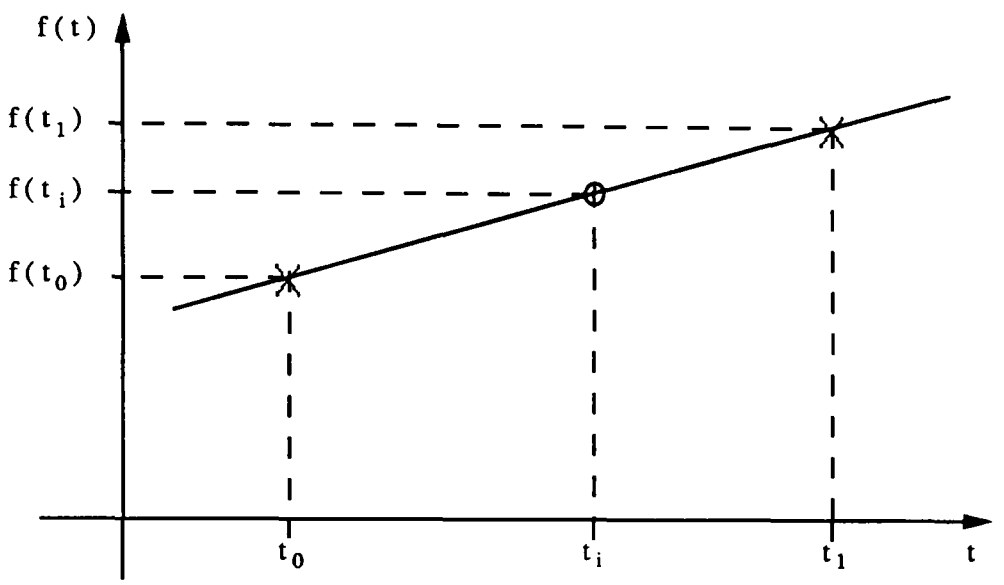


Figure 4.2 Linear interpolation of point t_i given points t_0 and t_1

Referring to figure 4.2, given points $f(t_0)$ and $f(t_1)$, the point $f(t_i)$ may be interpolated by assuming that $f(t_0)$ and $f(t_1)$ lie on a straight line. $f(t_i)$ is then defined as

$$f(t_i) = f(t_0) + \frac{f(t_1) - f(t_0)}{(t_1 - t_0)} (t_i - t_0) \qquad (4.1)$$

Definition (4.1) can be applied to the filterbank spectral frames if t_0 and t_1 represent spectral frame times, and $f(t_0)$ and $f(t_1)$ are energy levels for filterbank channel c, E_{ct_0} and E_{ct_1} , at frame times t_0 and t_1. Then from (4.1), the interpolated energy level, E_{ct_i}, is given by

$$E_{ct_i} = E_{ct_0} + \frac{E_{ct_1} - E_{ct_0}}{(t_1 - t_0)} (t_i - t_0) \qquad c = 1,2,....,16 \qquad (4.2)$$

The interpolation points are determined by dividing the phoneme-like segment into 20 equal intervals. The interpolated channel energy at these intervals are calculated in two ways, as illustrated by the following example. Consider a phoneme-like segment with more than 20 frames, say 50 frames. The interpolated channel energy values will be calculated at intervals of 2.5 frames (50/20 = 2.5). Figure 4.3 shows the channel energy values for one of the sixteen channels for the first four frames of the segment (marked 'A' to 'D'). The first interpolated energy value is to be calculated at frame time t = 2.5.

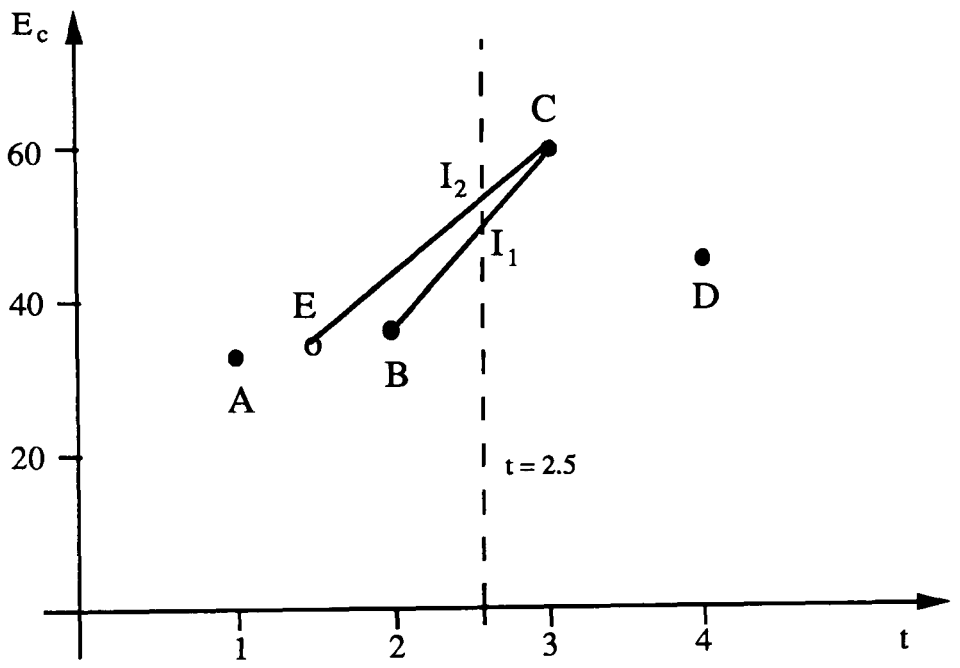


Figure 4.3 Two methods for calculating the interpolated energy value at frame time t = 2.5

Two approaches were tried in the calculation of the interpolated energy value.

(i) Since the interval at which the energy value is to be interpolated lies between frames 2 and 3 (ie. points 'B' and 'C'), it is therefore assumed that the interpolated point lies on a straight line passing through points 'B' and 'C' ie. 'I_1', and can be calculated

using (4.2). This process is repeated for the remaining interpolation points within the segment, and also for the other channels in the frame. This approach will be referred to in this work as the "unaveraged points interpolation method".

(ii) From figure 4.3, it can be seen that there are two frames (points 'A' and 'B') that precede the interpolation point at time t = 2.5. In this method, points 'A' and 'B' are first averaged to produce a new point, 'E', the value of which is the average of the energy values at points 'A' and 'B', and the position of this point is the average of the poistions of 'A' and 'B'. The interpolated point now lies on the straight line passing through this averaged point 'E' and point 'C' ie. 'I_2', and can be calculated using (4.2). This process is repeated for the remaining interpolation points within the segment, and also for the other channels in the frame. This approach will be referred to in this work as the "averaged points interpolation method".

The "averaged points interpolation method" differs from "unaveraged points interpolation method" because in the case of the latter, the calculation of the interpolated point takes into account the immediately preceding and following frames (ie. 'B' and 'C') only, while ignoring other data (eg. point 'A') which may be useful in discrimination of the phoneme-like segment. The "averaged points interpolation method" requires more computation than the "unaveraged points interpolation method" but may help improve the recognition of long phoneme-like segments such as /s/ in the word 'See' since this method considers all spectral frames within the segment in the calculation of the interpolated points, whereas the "unaveraged points

interpolation method" would ignore some of the spectral frames, which may be useful in the discrimination of the phoneme-like segment.

After time normalisation, the phoneme-like segment is representable by a 16 X 20 matrix. Since the 20th interpolated frame coincides with the last frame of the phoneme-like segment, the interpolated energy values for the former will work out to be the same as that for the latter, so there is no need to calculate the 20th interpolated frame. For this reason, only the first 19 interpolated frames are calculated for the phoneme-like segment and these are forwarded to the Data Reduction Stage of the recogniser.

4.5 Data Reduction

After the time normalisation stage, all phoneme-like segments are converted to 19 time-normalised spectral frames. Since each frame comprises 16 channel energies, the data is representable by a 16 x 19 matrix.

Following Badii & Binstead, three different types of encoding were tried; linear encoding, thermometer encoding and Gray encoding, to reduce the 8-bit channel energy data to 4 bits [11], [12].

(i) *Linear encoding*

The most-significant 4 bits of the 8 bit channel energy sample are chosen as the 4-bit encoded data.

(ii) *Thermometer encoding*

The 8-bit channel energy level has a range from 0 to 255. This interval is divided into 5 equal parts and mapped into 4 bits as follows:

[0 , 50] ---> 0000

[51 ,101] ---> 1000

[102,152] ---> 1100

[153,203] ---> 1110

[204,255] ---> 1111

(iii) *Gray encoding*

In this encoding method, the 8 bit range of values (0, 255) are divided into 16 equal subintervals. Each interval is represented by a 4-bit value chosen in such a way that the Hamming distance between the 4-bit encoded value for adjacent intervals is always 1, as given below.

[0 , 15] ---> 0000

[16 , 31] ---> 0001

[32 , 47] ---> 0011

[48 , 63] ---> 0010

[64 , 79] ---> 0110

etc.

The encoded data can now be used in training the system or in the case when the system has already been trained, to test its recognition performance.

4.6 The Recognition Module

Figure 4.5 shows the recognition module of the system. The 4-bit encoded data from the data reduction stage is stored in a spectral frame store which is a 2-dimensional bit array. Since the data is in the form of a 16 x 19 matrix where each of the sixteen channels is encoded as 4 bits, the dimensions of the spectral frame store is therefore (16 x 4) bits x 19 bits (figure 4.4).

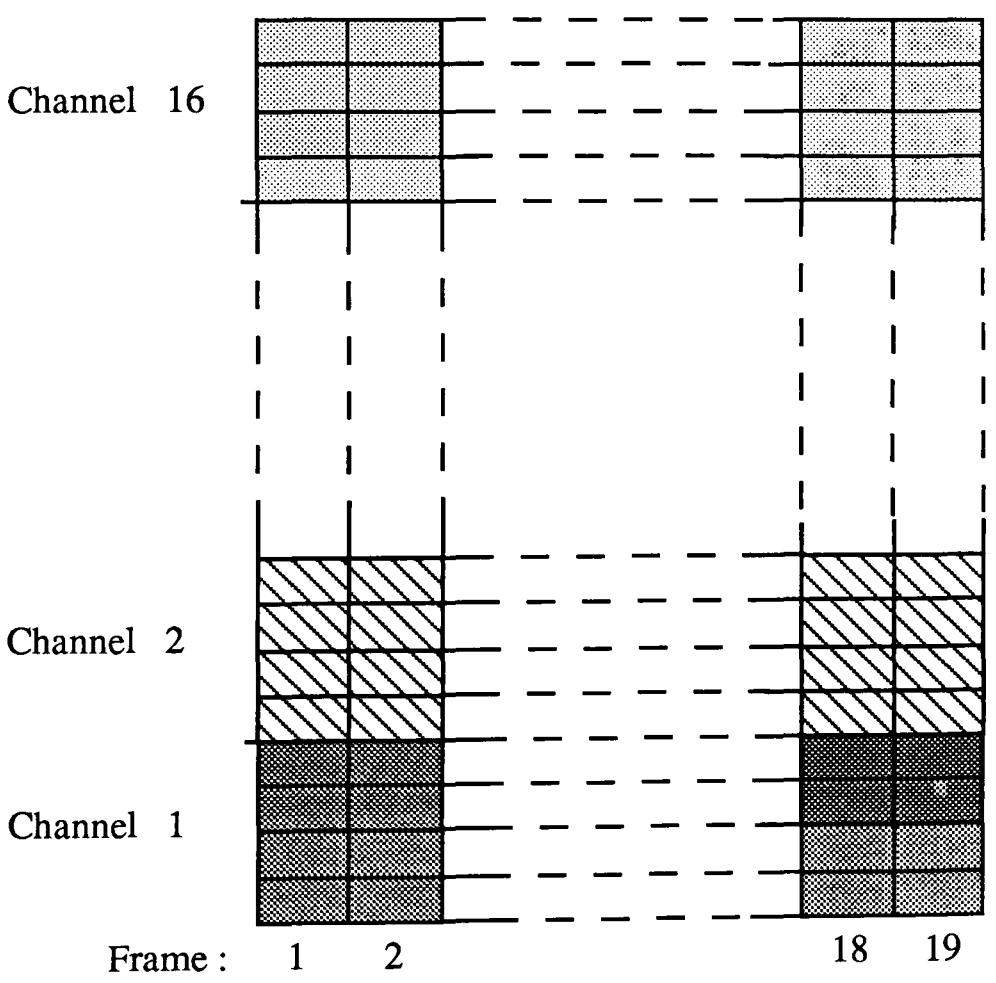


Figure 4.4 Spectral frame store layout

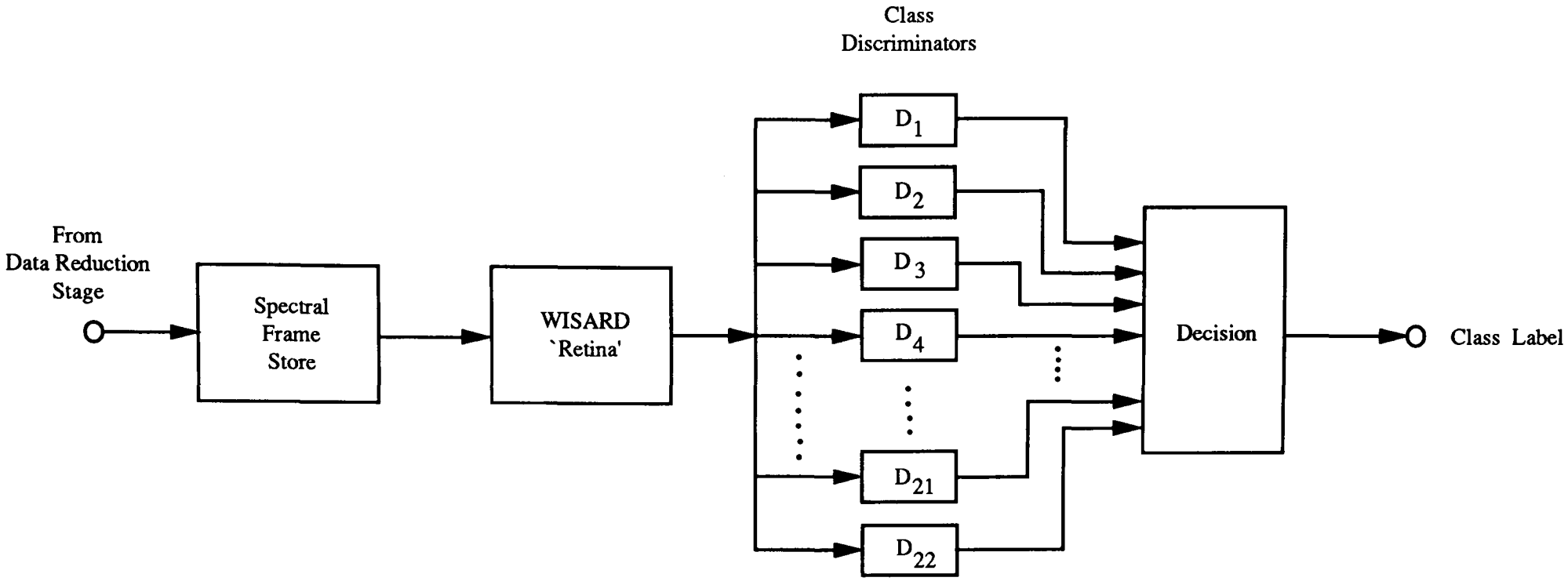


Figure 4.5 The recognition module

The function of the spectral frame store is to enable the encoded data from the data reduction stage to be manipulated when mapping it onto the WISARD 'retina'. Each column on the spectral frame store maps onto the corresponding column on the 'retina' ie. column '1' of the spectral frame store maps onto column '1' on the 'retina', and similarly for the remaining columns. Two mapping methods were used.

(i) *"Direct mapping" method*

The data in the spectral frame store column is mapped directly on to the corresponding 'retina' column in a one-to-one correspondence as shown in figure 4.6 below. Therefore each column on the 'retina' is an identical copy of the corresponding column on the spectral frame store.

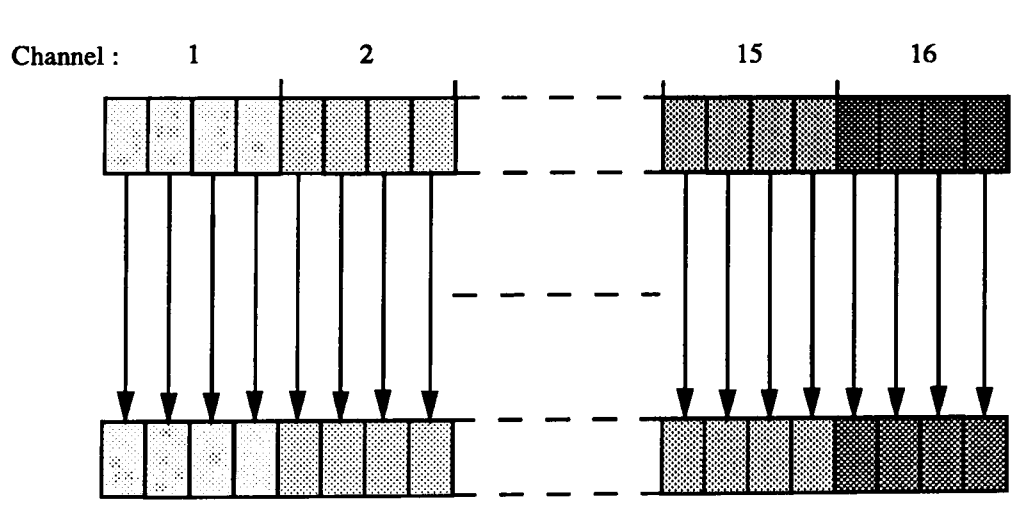


Figure 4.6 Direct mapping of spectral frame store to 'retina'

(ii) *"Across channel mapping" method*

The data in the spectral frame store column is mapped on to the corresponding 'retina' column by sampling bits from across

channels. This mapping method enables n-tuples to be formed from samples of data bits taken across spectral channels. The mapping is dependent on the n-tuple size. For a 4-tuple based system, the data bits are sampled across four adjacent channels on the spectral frame store column and mapped on the corresponding 'retina' column as shown in figure 4.7. Similarly, an 8-tuple based system would require sampling bits across eight adjacent channels in the same manner.

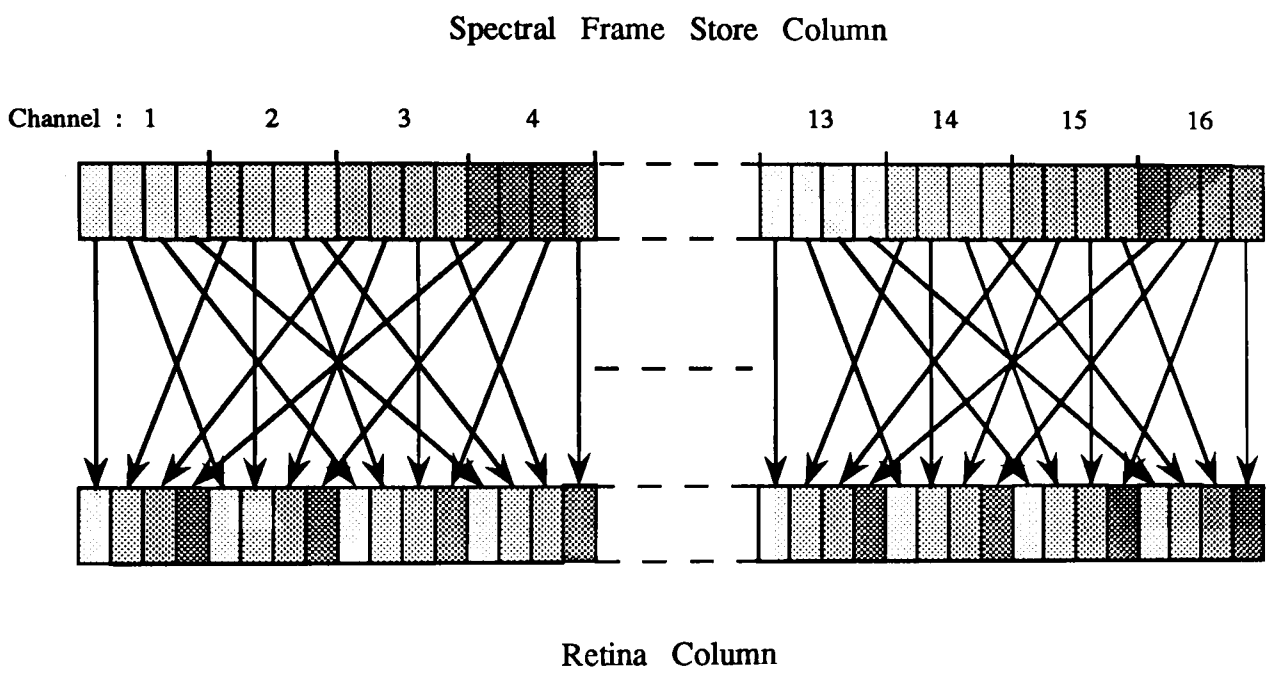


Figure 4.7 Across channel mapping to 'retina'

Since each bit in the spectral frame store maps onto a bit in the WISARD 'retina', the 'retina' is also a 2-dimensional bit array with the same dimensions as the spectral frame store, ie. (16 x 4) bits x 19 bits. *n* bits in the 'retina' are grouped together to form an n-tuple. There are two ways to map the 'retina' into n-tuples: linear mapping and random mapping, as mentioned in Chapter 2, Section 2.3. Each n-tuple

addresses a random access memory (RAM) in the class discriminator. In this recogniser design, there are 22 discriminators, one for each phoneme-like segment in the word-set to be recognised. The number of RAMs per discriminator depends on the tuple size, and is equal to the number of tuples needed to map the whole 'retina'. Tuple sizes of 4 and 8 were experimented with in this work. For a tuple size of 4 (ie. 4-tuple), 304 4-tuples (16 x 4 x 19/4) are required to map the whole 'retina'. This implies that each class discriminator contains 304 RAMs, where each RAM is a 2^4 x 1 bit RAM. Therefore each discriminator has a memory size of 4,864 bits, and the total memory requirement for all 22 discriminators in the system is 107,008 bits. For a tuple size of 8 (ie. 8-tuple), 152 8-tuples are needed to map the entire 'retina', ie. 152 RAMs are required per class discriminator. Each RAM is a 2^8 x 1 bit RAM. This gives a memory size of 38,912 bits per discriminator, ie. a memory size of 856,064 bits for all 22 discriminators. This has particular commercial significance since the hardware requirements in terms of programmable logic can be accomodated in a single 1 Mbit memory chip.

4.7 Training the Recogniser

To train the recogniser, training samples of phoneme-like segments were taken from selected words as indicated in Table 4.1. Not all the phoneme-like segments in these words were used to train the speech recogniser. These phoneme-like segments (ie. not used for training) are marked as '/-/' in the transcription of the words in the training set.

Table 4.1 Words from which the training samples were selected

Training Set	
Begun	: /b/, /ɪ/, /g/, /ʌ/, /n/
Cooler	: /k/, /u/, /l/, /-/
See	: /s/,/i/
Run	: /r/,/-/,/-/
Two	: /t/,/u/
Win	: /w/, /ɪ/, /-/
Tattoo	: /-/, /æ/, /t/, /-/
Want	: /w/, /ɔ/, /n/, /t/
Rudder	: /-/, /-/, /d/, /ə/
Shoe	: /ʃ/, /-/
Toothache	: /-/, /-/, /ϴ/, /eɪ/, /k/
Three	: /ϴr/, /ri/, /-/

To account for variations in the phoneme-like segment due to context, samples were taken from different words eg. /n/ from 'Begu**n**' and '**N**a**n**t', /t/ from '**T**wo', 'Wa**n**t' and 'Ta**t**too', and /u/ from 'T**wo**' and '**Coo**ler'.

Initially, all the discriminators are cleared (ie. all discriminator RAMs are set to '0'). Each of the phoneme-like segments is trained into its respective discriminator in the following manner. With the training sample mapped onto the WISARD 'retina', the discriminator associated with this sample class is selected. From the 'retina', n-tuples of the desired size are mapped (linear or random mapping) to the discriminator RAMs, and a '1' is written into the RAM locations addressed by the n-tuples.

4.8 Testing the Recogniser

With the phoneme-like segment to be recognised mapped onto the WISARD 'retina', n-tuples of the desired size are mapped (linear or random mapping) to RAMs in all the discriminators. RAM locations addressed by the n-tuples are read for each class discriminator. The output of a class discriminator is the sum of the output of all RAM cells (in that discriminator) addressed by the n-tuples mapped from the contents of the 'retina'. The unknown phoneme-like segment is assigned the class of the discriminator giving the highest score.

4.9 Evaluation of the Recogniser

The basic design of the phoneme recogniser is now complete. There is a choice of options in the implementation of certain parts of the design. The choices are :

(i) "Unaveraged points interpolation method" or "Averaged points interpolation method" for time normalisation.

(ii) Linear, Thermometer or Gray encoding for data reduction.

(iii) "Direct mapping" or "Across-channel mapping" of spectral frame store to WISARD 'retina'.

(iv) Linear mapping or Random mapping of WISARD 'retina' to n-tuples.

(v) Tuple size.

Tests were conducted with different combinations of these options, the aim being to determine the combination that enables the recogniser to give the best performance. The speech samples consisted of 50 utterances of each word in the word-set. 25 of these utterances (from the words in the training set) were used to train the recogniser. The remaining 25 utterances of each word were used to test the recogniser. To evaluate how the performance of the recogniser varies with the amount of training, training is increased in steps of five examples of each segment in the training set, up to twenty trainings. At each step the performance of the recogniser is noted. Tuple sizes of 4 and 8 were used.

4.9.1 Experiments with "Unaveraged Points Interpolation Method" for Time Normalisation

For these set of experiments, the following system configuration was used :

(i) Unaveraged points interpolation method for time normalisation.
(ii) Direct mapping of spectral frame store to WISARD 'retina'.

Results were obtained in turn for Linear, Thermometer and Gray encoders at the Data Reduction stage, with n-tuple sizes of 4 and 8, and linear and random mapping of the WISARD 'retina' for generating the n-tuples. The aim behind this first set of experiments is to find the system configurations that give the best results (the less promising ones would be discarded). These would then be used as the basis for the next set of experiments which would attempt to improve these results.

Linear Encoder

Tables 4.2 - 4.5 give the results for Linear encoder at the Data Reduction stage, and n-tuple sizes of 4 and 8, with linear or random mapping. In general it can be said that 8-tuple gives better results than 4-tuple, and similarly for random mapping over linear mapping. The best average recognition rate is 58.04% obtained for 8-tuple with random mapping, and 20 trainings of each member in the training set.

Thermometer Encoder

Tables 4.6 - 4.9 give the results for Thermometer encoder. These results are very much inferior to those obtained using the Linear encoder. This is mainly due to the $\log_2 5$ bits encoding accuracy and the way in which the recognition decision was performed. A phoneme-like segment was correctly recognised only if the class discriminator assigned to that segment gave the highest recognition score. The case of a 'tie' ie. two or more class discriminators (one of which is the correct class discriminator) giving the highest score, was treated as an incorrect recognition result. The results obtained for Thermometer encoding showed that a high proportion of the incorrect recognition was due to 'ties'. In the case of some of the phoneme-like segments, this was over 90%. This is because since the Thermometer encoder divides the range 0 - 255 into 5 equal subintervals only (Section 4.5), the range of these subintervals are large enough to cause the similar phoneme-like segments to generate the same 4-bit code sequence. This is reflected in the results shown in Tables 4.6 - 4.9, where in general, the average recognition accuracy deteriorates with increasing training.

As with Linear encoding, 8-tuple gives better results than 4-tuple, and similarly for random mapping over linear mapping. The best recognition rate is 36.44% for 8-tuple/random mapping and 5 trainings.

Gray Encoder

Tables 4.10 - 4.13 give the results for Gray encoder. Of the three encoders used, this method produces the best result. An average recognition accuracy of 67.17% is achieved for 8-tuple/random mapping with 20 trainings. As with the previous two encoders, 8-tuple gives better results than 4-tuple, as does random mapping compared with linear mapping.

From the results obtained for these experiments, in general, the following observations were made :

(i) The 8-tuple recogniser is superior to a 4-tuple recogniser.
(ii) Random mapping gives better results than linear mapping.
(iii) Gray encoder is the best of the three encoding methods.

Furthermore, comparison of Tables 4.2 and 4.4 with Tables 4.10 and 4.12 respectively show that they are identical. This is because for the linear mapping case, Linear encoding and Gray encoding are equivalent since both encoders quantize the range 0 - 255 into 16 equal intervals.

For both Linear encoding and Gray encoding, in the 4-tuple case it is seen that recognition accuracy peaks at 15 trainings whereas performance improves with increasing training for the 8-tuple case. This suggests that for the 4-tuple case, after 15 trainings, further training may be causing saturation of the discriminators.

Table 4.2

Linear_encoding & 4-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	62.50	54.17	50.00	8.33
/t/	46.83	79.67	98.00	83.58
/g/	45.83	50.00	66.67	58.33
/ʌ/	40.73	44.80	52.97	59.40
/n/	78.39	67.58	67.03	45.42
/k/	33.25	39.33	37.33	8.00
/u/	46.73	37.27	31.71	43.12
/l/	0.00	12.00	20.00	16.00
/ə/	55.58	80.89	68.74	52.76
/r/	60.00	60.00	62.00	90.00
/s/	4.00	4.00	8.00	4.00
/i/	40.95	40.60	41.33	24.23
/t/	92.50	84.96	85.98	78.98
/w/	51.67	30.29	52.58	37.58
/ʃ/	56.00	72.00	64.00	80.00
/ɔ/	40.00	44.00	60.00	60.00
/æ/	4.35	21.74	21.74	8.70
/ϴ/	50.00	37.50	37.50	16.67
/d/	88.00	92.00	88.00	92.00
/eɹ/	38.62	22.92	34.46	36.22
/ϴr/	28.00	44.00	36.00	32.00
/ri/	8.33	0.00	0.00	0.00
Average :	44.19	46.35	49.27	42.51

Table 4.3

Linear_encoding & 4-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	70.83	75.00	62.50	8.33
/t/	32.50	79.75	81.67	71.50
/g/	54.17	83.33	87.50	83.33
/ʌ/	29.40	56.93	74.67	77.20
/n/	80.53	37.08	14.81	5.39
/k/	12.16	45.08	55.17	38.83
/u/	76.70	48.19	41.77	59.47
/l/	0.00	0.00	0.00	8.00
/ə/	50.85	71.48	48.24	44.78
/r/	44.00	70.00	82.00	92.00
/s/	8.00	8.00	8.00	12.00
/i/	40.29	47.11	32.54	18.29
/t/	96.98	83.48	84.98	79.48
/w/	74.83	42.54	42.62	21.33
/ʃ/	56.00	80.00	80.00	92.00
/ɔ/	20.00	36.00	72.00	68.00
/æ/	8.70	8.70	8.70	4.35
/ϴ/	75.00	41.67	37.50	25.00
/d/	76.00	94.00	98.00	90.00
/eɹ/	34.77	48.72	58.17	64.43
/ϴr/	36.00	36.00	36.00	20.00
/ri/	0.00	0.00	0.00	0.00
Average :	44.44	49.69	50.31	44.71

Table 4.4

Linear_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	58.33	75.00	70.83	62.50
/t/	53.00	77.50	85.67	83.67
/g/	50.00	50.00	75.00	70.83
/ʌ/	40.70	46.40	50.53	50.63
/n/	83.78	76.39	78.42	60.89
/k/	59.58	79.92	87.92	85.83
/u/	50.95	55.36	45.69	65.36
/l/	8.00	20.00	32.00	32.00
/ə/	46.04	60.41	70.63	64.59
/r/	52.00	52.00	56.00	78.00
/s/	8.00	8.00	12.00	20.00
/i/	50.58	49.18	39.93	44.01
/t/	94.50	90.96	94.96	93.96
/w/	61.75	28.42	40.54	40.46
/ʃ/	52.00	72.00	68.00	92.00
/ɔ/	36.00	40.00	68.00	72.00
/æ/	4.35	17.39	21.74	21.74
/ϴ/	50.00	54.17	58.33	45.83
/d/	84.00	96.00	94.00	98.00
/eɹ/	30.61	37.18	42.79	38.94
/ϴr/	32.00	36.00	48.00	40.00
/ri/	8.33	0.00	0.00	0.00
Average :	46.11	51.01	56.41	57.33

Table 4.5

Linear_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	87.50	91.67	87.50	83.33
/t/	49.00	71.50	77.67	71.75
/g/	50.00	62.50	91.67	83.33
/ʌ/	14.73	26.13	40.83	44.13
/n/	87.92	77.17	81.25	48.58
/k/	45.33	51.58	63.67	59.33
/u/	61.76	70.24	48.82	74.33
/l/	0.00	0.00	4.00	16.00
/ə/	39.91	76.59	77.65	84.76
/r/	34.00	54.00	74.00	88.00
/s/	8.00	16.00	20.00	12.00
/i/	48.31	42.93	60.23	52.43
/t/	97.48	93.48	92.48	91.48
/w/	76.88	48.67	55.62	47.67
/ʃ/	48.00	76.00	76.00	96.00
/ɔ/	20.00	36.00	64.00	68.00
/æ/	4.35	17.39	8.70	8.70
/ϴ/	83.33	70.83	70.83	70.83
/d/	86.00	96.00	96.00	98.00
/eɹ/	28.52	24.36	44.23	54.33
/ϴr/	32.00	44.00	36.00	24.00
/ri/	0.00	0.00	0.00	0.00
Average :	45.59	52.14	57.78	58.04

Table 4.6

Thermometer_encoding & 4-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	4.17	0.00	0.00	0.00
/t/	24.50	26.33	22.17	8.00
/g/	4.17	20.83	16.67	8.33
/ʌ/	83.63	82.77	46.53	40.77
/n/	0.00	0.00	0.00	0.00
/k/	18.00	16.00	0.00	0.00
/u/	7.27	10.21	11.17	15.76
/l/	0.00	4.00	0.00	0.00
/a/	9.24	15.19	2.00	2.00
/r/	30.00	42.00	8.00	8.00
/s/	20.00	16.00	4.00	4.00
/i/	27.04	30.98	25.12	24.98
/t/	66.90	52.35	30.69	30.19
/w/	49.58	25.33	3.04	3.04
/ʃ/	80.00	84.00	84.00	88.00
/ɔ/	32.00	24.00	8.00	8.00
/æ/	4.35	17.39	17.39	17.39
/Θ/	37.50	0.00	0.00	0.00
/d/	18.00	2.00	0.00	0.00
/eɹ/	34.46	18.43	12.18	12.18
/Θr/	20.00	4.00	0.00	0.00
/ri/	16.67	8.33	8.33	8.33
Average :	26.70	22.73	13.60	12.68

Table 4.7

Thermometer_encoding & 4-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	4.17	0.00	0.00	0.00
/t/	32.67	67.33	55.00	44.75
/g/	50.00	66.67	62.50	58.33
/ʌ/	88.57	86.93	61.13	66.80
/n/	4.75	2.72	0.67	0.00
/k/	26.00	24.00	12.00	12.00
/u/	23.78	17.76	21.23	26.40
/l/	16.00	12.00	16.00	20.00
/a/	27.09	37.09	34.43	31.09
/r/	32.00	46.00	52.00	72.00
/s/	20.00	20.00	4.00	4.00
/i/	29.77	27.10	25.79	30.31
/t/	66.40	54.88	58.35	56.85
/w/	50.62	21.42	32.42	20.29
/ʃ/	84.00	84.00	84.00	88.00
/ɔ/	28.00	32.00	36.00	44.00
/æ/	4.35	17.39	13.04	13.04
/Θ/	37.50	0.00	0.00	0.00
/d/	32.00	34.00	28.00	24.00
/eɹ/	33.01	32.69	32.69	35.10
/Θr/	20.00	8.00	8.00	4.00
/ri/	25.00	41.67	25.00	41.67
Average :	33.44	33.35	30.10	31.48

Table 4.8

Thermometer_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	20.83	8.33	0.00	0.00
/t/	24.42	95.83	75.25	67.00
/g/	33.33	66.67	66.67	25.00
/ʌ/	61.07	65.83	65.03	59.50
/n/	12.17	2.08	1.39	0.69
/k/	26.00	22.00	0.00	0.00
/u/	41.71	38.31	40.74	28.09
/l/	24.00	20.00	24.00	24.00
/a/	40.11	40.20	28.94	23.76
/r/	12.00	38.00	52.00	66.00
/s/	20.00	20.00	8.00	8.00
/i/	22.93	24.27	29.25	29.94
/t/	69.90	60.35	57.25	56.29
/w/	58.71	25.42	44.50	16.17
/ʃ/	76.00	84.00	76.00	84.00
/ɔ/	28.00	28.00	24.00	36.00
/æ/	4.35	21.74	21.74	8.70
/Θ/	33.33	12.50	4.17	0.00
/d/	40.00	36.00	34.00	30.00
/eɹ/	35.10	25.00	20.83	30.93
/Θr/	32.00	12.00	4.00	0.00
/ri/	16.67	8.33	8.33	8.33
Average :	33.30	34.31	31.19	27.38

Table 4.9

Thermometer_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	4.17	0.00	0.00	0.00
/t/	34.67	77.75	75.58	77.42
/g/	62.50	66.67	54.17	54.17
/ʌ/	82.07	73.90	70.73	70.77
/n/	6.11	3.42	2.75	0.00
/k/	26.00	24.00	24.00	24.00
/u/	27.85	28.12	22.44	29.97
/l/	24.00	16.00	20.00	32.00
/a/	28.50	40.87	35.09	28.43
/r/	32.00	46.00	56.00	72.00
/s/	20.00	36.00	24.00	24.00
/i/	30.81	32.56	31.64	35.64
/t/	67.90	54.88	57.38	55.85
/w/	53.67	26.50	34.29	18.12
/ʃ/	84.00	84.00	84.00	92.00
/ɔ/	40.00	44.00	40.00	52.00
/æ/	8.70	17.39	13.04	8.70
/Θ/	33.33	0.00	0.00	0.00
/d/	44.00	42.00	44.00	38.00
/eɹ/	46.31	27.08	38.62	42.79
/Θr/	20.00	8.00	8.00	8.00
/ri/	25.00	33.33	16.67	25.00
Average :	36.44	35.57	34.20	35.86

Table 4.10

Gray_encoding & 4-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	62.50	54.17	50.00	8.33
/ʧ/	46.83	79.67	98.00	83.58
/g/	45.83	50.00	66.67	58.33
/ʌ/	40.73	44.80	52.97	59.40
/n/	78.39	67.58	67.03	45.42
/k/	33.25	39.33	37.33	8.00
/ʊ/	46.73	37.27	31.71	43.12
/l/	0.00	12.00	20.00	16.00
/ə/	55.58	80.89	68.74	52.76
/r/	60.00	60.00	62.00	90.00
/s/	4.00	4.00	8.00	4.00
/ʃ/	40.95	40.60	41.33	24.23
/t/	92.50	84.96	85.98	78.98
/w/	51.67	30.29	52.58	37.58
/ʃ/	56.00	72.00	64.00	80.00
/ɑ/	40.00	44.00	60.00	60.00
/æ/	4.35	21.74	21.74	8.70
/ɵ/	50.00	37.50	37.50	16.67
/d/	88.00	92.00	88.00	92.00
/eɪ/	38.62	22.92	34.46	36.22
/ɵr/	28.00	44.00	36.00	32.00
/ri/	8.33	0.00	0.00	0.00
Average :	44.19	46.35	49.27	42.51

Table 4.11

Gray_encoding & 4-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	58.33	83.33	79.17	79.17
/ʧ/	54.92	87.92	91.92	83.83
/g/	66.67	87.50	91.67	91.67
/ʌ/	71.53	62.57	68.23	69.90
/n/	77.08	71.06	75.14	39.33
/k/	2.00	14.16	14.09	18.08
/ʊ/	68.97	66.33	51.64	67.22
/l/	0.00	4.00	8.00	24.00
/ə/	67.58	76.78	80.89	86.61
/r/	60.00	66.00	72.00	86.00
/s/	4.00	8.00	12.00	12.00
/ʃ/	33.66	37.79	47.89	38.62
/t/	98.00	91.50	90.50	90.00
/w/	82.88	43.54	58.58	47.62
/ʃ/	56.00	68.00	84.00	96.00
/ɑ/	28.00	44.00	72.00	60.00
/æ/	8.70	21.74	13.04	17.39
/ɵ/	58.33	37.50	29.17	25.00
/d/	82.00	94.00	94.00	94.00
/eɪ/	62.66	58.49	70.03	66.51
/ɵr/	32.00	40.00	44.00	28.00
/ri/	0.00	8.33	0.00	0.00
Average :	48.79	53.30	56.73	55.50

Table 4.12

Gray_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	58.33	75.00	70.83	62.50
/ʧ/	53.00	77.50	85.67	83.67
/g/	50.00	50.00	75.00	70.83
/ʌ/	40.70	46.40	50.53	50.63
/n/	83.78	76.39	78.42	60.89
/k/	59.58	79.92	87.92	85.83
/ʊ/	50.95	55.36	45.69	65.36
/l/	8.00	20.00	32.00	32.00
/ə/	46.04	60.41	70.63	64.59
/r/	52.00	52.00	56.00	78.00
/s/	8.00	8.00	12.00	20.00
/ʃ/	50.58	49.18	39.93	44.01
/t/	94.50	90.96	94.96	93.96
/w/	61.75	28.42	40.54	40.46
/ʃ/	52.00	72.00	68.00	92.00
/ɑ/	36.00	40.00	68.00	72.00
/æ/	4.35	17.39	21.74	21.74
/ɵ/	50.00	54.17	58.33	45.83
/d/	84.00	96.00	94.00	98.00
/eɪ/	30.61	37.18	42.79	38.94
/ɵr/	32.00	36.00	48.00	40.00
/ri/	8.33	0.00	0.00	0.00
Average :	46.11	51.01	56.41	57.33

Table 4.13

Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	83.33	87.50	87.50	87.50
/ʧ/	63.17	88.00	91.92	85.83
/g/	75.00	83.33	91.67	91.67
/ʌ/	45.77	52.13	58.57	65.97
/n/	89.89	85.19	87.92	71.86
/k/	32.75	55.58	67.67	71.75
/ʊ/	58.25	74.48	50.62	73.78
/l/	24.00	28.00	32.00	36.00
/ə/	57.68	77.04	81.78	90.67
/r/	52.00	64.00	72.00	88.00
/s/	20.00	16.00	28.00	24.00
/ʃ/	61.35	52.24	63.10	53.64
/t/	97.50	95.00	92.00	92.00
/w/	85.92	49.54	70.83	68.75
/ʃ/	60.00	84.00	88.00	96.00
/ɑ/	56.00	64.00	76.00	68.00
/æ/	13.04	21.74	17.39	21.74
/ɵ/	87.50	70.83	66.67	75.00
/d/	86.00	96.00	96.00	98.00
/eɪ/	50.48	52.56	78.37	68.59
/ɵr/	40.00	48.00	52.00	48.00
/ri/	8.33	16.67	25.00	0.00
Average :	56.73	61.90	67.04	67.12

4.9.2 Experiments with "Averaged Points Interpolation Method" for Time Normalisation

The aim of this set of experiments is to determine which of the two interpolation methods for time normalisation produces better results. For these set of experiments, the setup was the same as in the experiments in Section 4.9.1 except that the "averaged points interpolation method" is used for time normalisation. Thus the following system configuration was used :

(i) "Averaged points interpolation method" for time normalisation.
(ii) Direct mapping of spectral frame store to WISARD 'retina'.

The results obtained for the experiments in Section 4.9.1 showed that 8-tuple gave better results than 4-tuple, and that Thermometer encoding gave poor results. It was therefore decided to discontinue experiments with n-tuple size of 4, and Thermometer encoding.

Tables 4.14 - 4.17 give the results obtained for this set of experiments. As in the experiments of Section 4.9.1, Gray encoding gives better results than Linear encoding, and random mapping produces better results than linear mapping. Also, Gray encoding and Linear encoding are equivalent for the linear mapping case (Tables 4.14 and 4.16). The highest recognition accuracy is 66.50% obtained for Gray encoding with random mapping and 20 trainings. Comparing with the corresponding results in previous experiments (Tables 4.5 and 4.6 for Linear encoding, and Tables 4.12 and 4.13 for Gray encoding) it is clear that the system configuration with the "unaveraged points interpolation method" gives the better results. This is however a surprising result because the "averaged points interpolation method"

uses data from all spectral frames in the interpolation process whereas the former method only uses the spectral frames that immediately precede and follow the interpolation point and so for phoneme-like segments with more than 19 frames, data from certain frames will be ignored in the interpolation process (Section 4.4.1.1).

Table 4.14

Linear_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	58.33	70.83	66.67	62.50
/ʧ/	55.08	79.50	85.67	81.58
/g/	50.00	50.00	75.00	70.83
/ʃ/	43.97	44.00	47.30	49.00
/n/	80.39	75.06	77.72	59.58
/k/	53.42	77.92	87.92	85.83
/u/	48.81	53.54	44.98	65.66
/l/	8.00	20.00	32.00	32.00
/ə/	47.62	61.92	68.65	68.15
/r/	52.00	52.00	56.00	76.00
/s/	8.00	8.00	8.00	12.00
/i/	53.48	48.56	41.51	46.14
/t/	94.50	89.96	95.48	93.96
/w/	63.79	28.42	41.54	40.50
/ʃ/	52.00	68.00	60.00	92.00
/ɔ/	36.00	40.00	64.00	72.00
/æ/	4.35	17.39	21.74	21.74
/Θ/	50.00	54.17	62.50	45.83
/d/	82.00	94.00	92.00	96.00
/eɪ/	32.69	37.18	35.10	37.18
/Θr/	28.00	44.00	40.00	36.00
/ri/	8.33	0.00	0.00	8.33
Average :	45.94	50.66	54.72	56.95

Table 4.15

Linear_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	87.50	91.67	87.50	83.33
/ʧ/	47.00	67.58	79.75	71.67
/g/	50.00	66.67	83.33	83.33
/ʃ/	14.70	29.37	40.00	44.90
/n/	88.58	76.50	79.22	44.58
/k/	41.25	49.42	57.50	57.17
/u/	56.65	65.67	55.76	76.14
/l/	0.00	0.00	4.00	20.00
/ə/	42.71	74.58	77.52	84.20
/r/	34.00	54.00	72.00	90.00
/s/	8.00	12.00	12.00	12.00
/i/	53.46	51.35	57.43	47.16
/t/	98.50	95.98	94.48	91.98
/w/	80.88	45.50	51.58	45.58
/ʃ/	48.00	72.00	76.00	96.00
/ɔ/	24.00	36.00	56.00	68.00
/æ/	8.70	21.74	13.04	8.70
/Θ/	83.33	70.83	75.00	66.67
/d/	84.00	96.00	96.00	98.00
/eɪ/	24.36	26.44	42.47	46.64
/Θr/	28.00	48.00	36.00	24.00
/ri/	0.00	0.00	0.00	0.00
Average :	45.62	52.33	56.66	57.27

Table 4.16

Gray_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	58.33	70.83	66.67	62.50
/ʧ/	55.08	79.50	85.67	81.58
/g/	50.00	50.00	75.00	70.83
/ʃ/	43.97	44.00	47.30	49.00
/n/	80.39	75.06	77.72	59.58
/k/	53.42	77.92	87.92	85.83
/u/	48.81	53.54	44.98	65.66
/l/	8.00	20.00	32.00	32.00
/ə/	47.62	61.92	68.65	68.15
/r/	52.00	52.00	56.00	76.00
/s/	8.00	8.00	8.00	12.00
/i/	53.48	48.56	41.51	46.14
/t/	94.50	89.96	95.48	93.96
/w/	63.79	28.42	41.54	40.50
/ʃ/	52.00	68.00	60.00	92.00
/ɔ/	36.00	40.00	64.00	72.00
/æ/	4.35	17.39	21.74	21.74
/Θ/	50.00	54.17	62.50	45.83
/d/	82.00	94.00	92.00	96.00
/eɪ/	32.69	37.18	35.10	37.18
/Θr/	28.00	44.00	40.00	36.00
/ri/	8.33	0.00	0.00	8.33
Average :	45.94	50.66	54.72	56.95

Table 4.17

Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	79.17	87.50	87.50	87.50
/ʧ/	63.17	90.00	93.92	85.83
/g/	75.00	83.33	91.67	91.67
/ʃ/	45.77	51.30	60.20	64.33
/n/	89.89	85.17	88.61	67.14
/k/	30.83	53.42	55.17	67.58
/u/	58.70	75.91	48.69	70.16
/l/	20.00	28.00	36.00	36.00
/ə/	57.53	75.39	78.58	87.48
/r/	44.00	64.00	70.00	92.00
/s/	20.00	12.00	16.00	16.00
/i/	61.07	52.24	63.33	63.87
/t/	97.50	96.00	95.00	93.00
/w/	84.92	49.58	68.79	64.75
/ʃ/	60.00	80.00	88.00	92.00
/ɔ/	56.00	56.00	72.00	68.00
/æ/	13.04	21.74	17.39	21.74
/Θ/	87.50	75.00	70.83	75.00
/d/	84.00	96.00	94.00	98.00
/eɪ/	52.56	48.72	74.52	68.91
/Θr/	40.00	48.00	52.00	52.00
/ri/	8.33	16.67	25.00	0.00
Average :	55.86	61.18	65.78	66.50

4.9.3 Experiments with "Across Channel Mapping" of Spectral Frame Store

The aim of this set of experiments is to determine which of the two methods for mapping the spectral frame store to the WISARD 'retina' is more effective; the "direct mapping" method or the "across channel mapping" method (Section 4.6). For the experiments of Sections 4.9.1 and 4.9.2, the "direct mapping" method was used for mapping the spectral frame store to the WISARD 'retina'. These experiments showed that for time normalisation, the "unaveraged points interpolation method" gave better results than the "averaged points interpolation method" and so for further experiments, the "unaveraged points interpolation method" only will be used for time normalisation. Thus the following system configuration was used :

(i) "Unaveraged points interpolation method" for time normalisation.

(ii) Across channel mapping of spectral frame store to WISARD 'retina'.

Tables 4.18 - 4.21 give the results for this set of experiments using Linear and Gray encoders. Once again the best results are obtained by Gray encoding with recognition accuracy increasing with training. However, the following differences are observed in comparison with the results obtained in the earlier set of experiments.

(i) For experiments with both Linear and Gray encoders, linear mapping of the 'retina' to n-tuples gives better results than random mapping.

Best average recognition accuracy of 69.61% is obtained with 20 trainings for Gray encoding with linear mapping (Table 4.20). This is an improvement over the results for the previous experiments since the best results obtained with the "direct mapping" method is 67.12% (Table 4.17). Overall, the results obtained in this set of experiments are an improvement over those obtained with the "direct mapping" method and this suggests that for mapping the spectral frame store to the WISARD 'retina', the "across channel mapping" method is the better method.

Table 4.18

Linear_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	87.50	100.00	100.00	95.83
/t/	55.17	77.75	75.67	79.67
/g/	58.33	83.33	87.50	79.17
/ŋ/	22.07	33.47	38.37	40.83
/n/	88.58	87.92	88.61	79.25
/k/	28.92	39.08	63.58	71.67
/u/	41.25	56.87	59.74	74.60
/l/	12.00	24.00	20.00	32.00
/ə/	33.74	60.83	75.44	78.87
/r/	40.00	50.00	58.00	82.00
/s/	28.00	16.00	24.00	20.00
/i/	55.71	52.17	49.93	44.02
/v/	95.00	97.00	95.50	92.48
/w/	74.71	49.62	64.92	65.96
/ʃ/	60.00	84.00	88.00	88.00
/ɔ/	20.00	32.00	68.00	76.00
/æ/	8.70	21.74	17.39	17.39
/Θ/	83.33	75.00	62.50	75.00
/d/	62.00	84.00	90.00	88.00
/eɪ/	33.01	30.93	37.18	44.88
/Θr/	20.00	40.00	36.00	36.00
/ri/	8.33	0.00	0.00	0.00
Average :	46.20	54.35	59.11	61.89

Table 4.19

Linear_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	79.17	83.33	79.17	70.83
/t/	47.08	79.75	81.75	75.50
/g/	54.17	75.00	87.50	87.50
/ŋ/	9.87	28.73	32.80	40.97
/n/	86.50	79.81	84.61	68.47
/k/	18.33	24.25	38.58	46.83
/u/	59.62	59.64	63.08	80.83
/l/	0.00	4.00	4.00	12.00
/ə/	43.65	67.54	71.13	84.61
/r/	38.00	56.00	76.00	94.00
/s/	8.00	12.00	20.00	24.00
/i/	43.77	38.01	37.71	31.67
/v/	98.00	95.50	97.00	92.98
/w/	78.96	55.79	62.83	59.83
/ʃ/	52.00	60.00	60.00	96.00
/ɔ/	24.00	32.00	56.00	72.00
/æ/	8.70	17.39	17.39	13.04
/Θ/	91.67	83.33	83.33	79.17
/d/	82.00	96.00	96.00	96.00
/eɪ/	29.97	22.27	34.46	54.33
/Θr/	24.00	36.00	44.00	44.00
/ri/	0.00	0.00	0.00	0.00
Average :	44.43	50.29	55.79	60.21

Table 4.20

Gray_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	100.00	95.83	91.67	83.33
/t/	71.33	83.83	85.92	87.92
/g/	75.00	91.67	87.50	87.50
/ŋ/	40.80	56.10	70.03	67.70
/n/	91.92	87.86	91.31	77.19
/k/	36.83	47.33	69.67	73.67
/u/	42.64	64.38	64.26	75.20
/l/	20.00	20.00	20.00	44.00
/ə/	48.46	63.06	88.22	93.94
/r/	58.00	68.00	80.00	94.00
/s/	28.00	20.00	24.00	28.00
/i/	64.54	67.75	66.62	68.83
/v/	96.00	97.50	95.50	94.00
/w/	84.88	59.71	69.75	73.83
/ʃ/	64.00	84.00	92.00	96.00
/ɔ/	64.00	64.00	88.00	92.00
/æ/	8.70	21.74	17.39	17.39
/Θ/	91.67	91.67	91.67	83.33
/d/	82.00	94.00	94.00	98.00
/eɪ/	51.44	53.52	53.52	47.60
/Θr/	32.00	36.00	52.00	48.00
/ri/	0.00	25.00	16.67	0.00
Average :	56.92	63.32	68.62	69.61

Table 4.21

Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	83.33	91.67	87.50	87.50
/t/	59.25	91.92	93.92	89.92
/g/	75.00	83.33	100.00	91.67
/ŋ/	45.80	47.30	57.10	61.93
/n/	86.42	85.14	87.94	78.53
/k/	14.41	22.25	36.58	48.92
/u/	57.95	69.79	63.44	73.64
/l/	20.00	16.00	28.00	28.00
/ə/	58.35	75.37	77.20	85.44
/r/	58.00	70.00	80.00	94.00
/s/	8.00	12.00	20.00	28.00
/i/	40.97	49.41	59.12	55.05
/v/	98.00	96.50	96.00	94.48
/w/	82.92	62.79	71.88	67.79
/ʃ/	56.00	68.00	80.00	92.00
/ɔ/	60.00	56.00	76.00	76.00
/æ/	13.04	21.74	21.74	13.04
/Θ/	87.50	83.33	83.33	79.17
/d/	86.00	96.00	94.00	98.00
/eɪ/	56.73	54.65	62.66	60.90
/Θr/	32.00	48.00	48.00	52.00
/ri/	0.00	8.33	16.67	0.00
Average :	53.62	59.52	65.50	66.18

4.9.4 Adding Normalisation Techniques to the Recogniser

The basic design of the speech recogniser is complete and the best average recognition accuracy of 69.61% is obtained for a system with the following configuration :

(i) "Unaveraged points interpolation method" for time normalisation.
(ii) Gray encoder for the Data Reduction stage.
(iii) Across channel mapping of spectral frame store to 'retina'.
(iv) n-tuple size of 8.
(v) Linear mapping of 'retina' to n-tuples.

The following normalisation techniques ([3]) are introduced into the speech recogniser design to attempt further improvement in the recognition accuracy :

(i) Spectral energy normalisation.
(ii) Noise subtraction normalisation.

The aim of these normalisation techniques is to make the spectral representation of the speech segments from the filterbank more robust by reducing their variability and hence less confusable. It is therefore expected that improved recognition accuracy will result.

4.9.4.1 Spectral Energy Normalisation

The spectral energy normalisation method employed in the speech recogniser was first proposed by Shearme & Leach [35], [111]. Given a spectral frame, the method is to divide the energy in each

filterbank channel by the sum of the energy from all filterbank channels for that frame, as given by

$$F_c = f_c / \sum_{i=1}^{16} f_i \qquad c = 1,2,3,...,16 \qquad (4.3)$$

where f_c is the energy in filterbank channel c and F_c is the normalised energy for that channel.

Tables 4.22 and 4.23 give the results for the speech recogniser with spectral energy normalisation. As expected, these results are an improvement over those obtained in the earlier experiments. The best average recognition accuracy is 79.22% obtained with random mapping and 15 trainings. This is an improvement of almost 10% over the best recognition results in the earlier experiments (69.61%). This is comparable with results for phoneme recognition achieved by other researchers (Kashyap & Mittal achieved 78% phoneme recognition for 40 words but these were from six different speakers [84]).

With energy normalisation, the recognition of /s/ has improved vastly, but a drop in recognition accuracy for /i/ has resulted. This is mainly due to confusion with /u/. No improvements were effected for recognition of /æ/. There has also been an improvement in the recognition of /Θr/ and /ri/.

Table 4.24 gives results obtained with spectral energy normalisation for Gray encoding/8-tuple random mapping but with the "averaged points interpolation method" for time normalisation. Although the experiments of Section 4.9.2 showed that the "unaveraged points interpolation method" gave better results than the "averaged points interpolation method", with spectral energy normalisation, the latter method gives slightly better results. Best average recognition accuracy of 79.61% is achieved compared with 79.22% for the "unaveraged points interpolation method".

Table 4.22

Gray_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		**Accuracy (%)**		
/b/	87.50	95.83	95.83	95.83
/t/	81.67	73.75	59.42	71.58
/g/	66.67	87.50	87.50	87.50
/ʌ/	78.77	87.80	83.80	80.53
/n/	88.61	89.94	91.28	88.58
/k/	85.67	89.67	87.50	89.58
/u/	77.43	83.94	82.12	87.40
/l/	64.00	72.00	76.00	76.00
/ə/	81.22	85.35	90.98	91.50
/r/	72.00	78.00	86.00	92.00
/s/	64.00	60.00	80.00	68.00
/ʃ/	4.08	7.50	21.91	18.41
/t/	97.50	98.00	98.00	98.00
/w/	82.96	61.83	70.92	75.96
/ʃ/	92.00	92.00	96.00	96.00
/ə/	96.00	96.00	100.00	100.00
/æ/	17.39	17.39	21.74	21.74
/Θ/	100.00	95.83	91.67	79.17
/d/	90.00	94.00	94.00	94.00
/eɹ/	65.39	59.46	59.46	59.46
/Θr/	20.00	32.00	44.00	40.00
/ri/	16.67	33.33	25.00	33.33
Average :	69.52	72.32	74.69	74.75

Table 4.23

Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		**Accuracy (%)**		
/b/	91.67	100.00	100.00	100.00
/t/	81.75	94.00	89.92	88.00
/g/	70.83	83.33	95.83	95.83
/ʌ/	78.17	83.80	75.80	71.70
/n/	96.67	95.31	94.00	92.67
/k/	81.50	85.50	89.67	89.67
/u/	85.00	83.57	78.26	80.26
/l/	56.00	80.00	76.00	68.00
/ə/	91.82	95.89	94.04	94.09
/r/	92.00	94.00	94.00	94.00
/s/	64.00	68.00	68.00	68.00
/ʃ/	5.31	13.08	38.65	37.54
/t/	96.00	98.00	98.00	98.00
/w/	87.00	80.92	83.00	81.00
/ʃ/	92.00	96.00	96.00	96.00
/ə/	96.00	96.00	96.00	96.00
/æ/	17.39	21.74	21.74	17.39
/Θ/	91.67	91.67	87.50	91.67
/d/	94.00	94.00	94.00	94.00
/eɹ/	57.38	59.46	82.53	90.23
/Θr/	16.00	24.00	40.00	36.00
/ri/	16.67	50.00	50.00	33.33
Average :	70.86	76.74	79.22	77.88

Table 4.24. **Gray_encoding & 8-tuple Random_mapping**
"Averaged points interpolation method" for Time Normalisation

Trainings :	5	10	15	20
Class		**Accuracy (%)**		
/b/	87.50	100.00	100.00	100.00
/t/	81.75	94.00	89.92	88.00
/g/	70.83	87.50	95.83	95.83
/ʌ/	80.53	84.60	75.80	70.10
/n/	96.67	96.00	94.67	93.33
/k/	81.50	85.58	89.67	91.75
/u/	86.51	87.21	82.55	81.50
/l/	56.00	80.00	80.00	64.00
/ə/	96.83	95.89	92.26	94.09
/r/	92.00	94.00	94.00	94.00
/s/	64.00	64.00	68.00	80.00
/ʃ/	5.98	18.86	33.22	36.12
/t/	96.00	98.00	98.00	98.00
/w/	87.00	80.92	84.00	82.00
/ʃ/	92.00	96.00	96.00	96.00
/ə/	96.00	96.00	96.00	96.00
/æ/	17.39	21.74	21.74	17.39
/Θ/	91.67	95.83	91.67	100.00
/d/	94.00	94.00	96.00	94.00
/eɹ/	57.38	63.30	86.38	90.23
/Θr/	16.00	24.00	44.00	32.00
/ri/	8.33	50.00	41.67	33.33
Average :	70.72	77.61	79.61	78.53

4.9.4.2 Noise Subtraction Normalisation

The objective is to try to reduce the effect of background noise on the speech spectral data which in turn should result in improvement in the recogniser's performance. The method is to determine the energy level of the background noise and to subtract this energy from the energy of the input signal over the duration of the utterance. The remaining energy level would be that due to the speech signal only.

In the speech recogniser design, the background noise was represented by a spectral frame just before the start of the utterance. This spectral frame was subtracted from each of the spectral frames obtained over the duration of the utterance [3]. The resulting spectral frames represent the speech signal only and may now be energy normalised.

Tables 4.25 - 4.28 give the results for the experiments with noise subtraction normalisation. Since the energy normalisation experiments have indicated that the "averaged points interpolation method" gives slightly better results than the "unaveraged points interpolation method", results were therefore obtained for both interpolation methods (Tables 4.25 - 4.26 for the former method, and Tables 4.27 - 4.28 for the latter method). These results continue to show that with energy normalisation, the "averaged points interpolation method" gives higher recognition accuracy. 78.23% average recognition accuracy is obtained for Gray encoding/8-tuple random mapping with 20 trainings. Although this is lower than that achieved with energy normalisation only (79.61%), if recognition scores for /æ/, /Θr/ and /ri/ are ignored since in all the experiments, these three classes have been the most difficult to recognise, it is found that the results obtained with noise subtraction normalisation have in fact been an improvement. Referring to Table

4.24 (ie. results of the energy normalisation experiment), if recognition scores for /æ/, /Θr/ and /ri/ are excluded, then for the remaining classes, the average recognition accuracy for 15 trainings is 86.52% (scores for the 15 trainings case are considered since this has the best average recognition accuracy). This compares with 87.23% (Table 4.27, 20 trainings) and 87.68% (Table 4.28, 15 trainings) for the noise subtraction normalisation experiments.

Tables 4.29 and 4.30 show the word recognition accuracy (for the Badii & Binstead word-set) for the 8-tuple based speech recogniser with Gray encoding and the "averaged points interpolation method" for time normalisation. A word from the word-set was considered to have been correctly recognised only if all the phoneme-like segments in that word were correctly recognised by the speech recogniser. The best word recognition accuracy is 57.2%, achieved with random mapping of the n-tuples and 15 trainings (Table 4.30). Comparing with the corresponding accuracy at the phoneme-like segment level (Table 4.28), it is seen that a higher word recognition accuracy is obtained for 10 trainings than with 20 trainings although the latter has a higher accuracy at the phoneme-like segment level. About 70% of the words that were considered to be errors in word recognition, had misclassification of one phoneme-like segment only. These errors may be easily corrected by a linguistic processor. However, it may not be possible to correct all of these errors, eg. if the phoneme-like segment /i/ in the word 'Tee' is incorrectly recognised as /u/, then 'Tee' will be recognised as 'Two' which is a word in the word-set of the speech recogniser.

Table 4.25 **Gray_encoding & 8-tuple Linear_mapping**
"Unaveraged points interpolation method" for Time Normalisation

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	83.33	100.00	91.67	87.50
/ʧ/	85.58	93.92	91.83	91.83
/g/	95.83	100.00	100.00	95.83
/ʃ/	80.50	85.40	76.53	75.73
/n/	93.25	89.83	96.64	96.61
/k/	81.33	83.33	91.67	95.83
/u/	61.62	68.59	65.97	68.33
/l/	56.00	76.00	64.00	56.00
/ɚ/	86.78	91.44	91.11	93.89
/r/	70.00	88.00	90.00	96.00
/s/	68.00	76.00	88.00	88.00
/ɾ/	58.40	61.98	64.44	62.10
/t/	96.96	97.48	96.50	95.48
/w/	80.00	84.00	85.00	90.00
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	96.00	96.00	100.00	100.00
/æ/	17.39	17.39	17.39	21.74
/Θ/	70.83	87.50	75.00	70.83
/d/	94.00	94.00	94.00	94.00
/eɹ/	49.68	49.68	49.68	59.46
/Θr/	12.00	20.00	36.00	32.00
/ri/	0.00	8.33	0.00	25.00
Average :	69.89	75.86	75.70	77.10

Table 4.26 **Gray_encoding & 8-tuple Random_mapping**
"Unaveraged points interpolation method" for Time Normalisation

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	75.00	95.83	100.00	95.83
/ʧ/	83.75	89.75	89.75	85.75
/g/	87.50	95.83	95.83	95.83
/ʃ/	74.90	81.47	71.77	72.53
/n/	91.97	94.61	97.28	95.94
/k/	87.50	95.83	95.83	95.83
/u/	79.14	77.38	78.26	77.44
/l/	60.00	84.00	80.00	56.00
/ɚ/	92.83	95.50	98.61	97.22
/r/	80.00	94.00	96.00	96.00
/s/	80.00	88.00	88.00	80.00
/ɾ/	43.72	48.21	51.35	49.01
/t/	96.98	96.48	98.00	97.50
/w/	78.00	82.00	84.00	87.00
/ʃ/	96.00	96.00	100.00	96.00
/ɔ/	96.00	92.00	96.00	96.00
/æ/	21.74	21.74	21.74	21.74
/Θ/	62.50	62.50	70.83	70.83
/d/	94.00	94.00	94.00	94.00
/eɹ/	55.60	65.07	74.84	86.38
/Θr/	12.00	24.00	24.00	24.00
/ri/	0.00	16.67	8.33	16.67
Average :	70.42	76.86	77.93	76.71

Table 4.27 **Gray_encoding & 8-tuple Linear_mapping**
"Averaged points interpolation method" for Time Normalisation

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	91.67	100.00	91.67	91.67
/ʧ/	87.67	93.92	93.92	91.83
/g/	91.67	95.83	95.83	100.00
/ʃ/	60.97	79.73	65.97	61.93
/n/	91.92	91.92	95.25	95.94
/k/	79.17	91.67	97.92	97.92
/u/	61.30	60.40	61.36	72.27
/l/	64.00	76.00	76.00	72.00
/ɚ/	90.17	93.50	94.56	95.22
/r/	64.00	88.00	84.00	92.00
/s/	84.00	92.00	92.00	92.00
/ɾ/	60.03	63.07	63.88	63.62
/t/	96.98	95.96	96.46	95.96
/w/	79.00	80.00	86.00	89.00
/ʃ/	100.00	100.00	100.00	96.00
/ɔ/	96.00	96.00	100.00	100.00
/æ/	17.39	17.39	17.39	21.74
/Θ/	79.17	83.33	79.17	83.33
/d/	92.00	94.00	94.00	94.00
/eɹ/	57.38	53.52	63.30	72.76
/Θr/	12.00	32.00	28.00	16.00
/ri/	16.67	8.33	16.67	25.00
Average :	71.51	76.66	76.97	78.19

Table 4.28 **Gray_encoding & 8-tuple Random_mapping**
"Averaged points interpolation method" for Time Normalisation

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	95.83
/ʧ/	87.83	93.92	91.83	87.75
/g/	91.67	95.83	100.00	95.83
/ʃ/	69.97	83.03	78.05	70.13
/n/	95.22	97.28	98.64	98.64
/k/	77.33	95.83	97.96	97.92
/u/	81.38	74.46	79.56	77.56
/l/	64.00	80.00	76.00	68.00
/ɚ/	93.50	96.17	97.30	97.94
/r/	92.00	96.00	98.00	98.00
/s/	84.00	84.00	88.00	84.00
/ɾ/	56.62	45.46	47.92	49.54
/t/	97.00	98.00	96.98	97.00
/w/	73.00	79.00	84.85	85.00
/ʃ/	96.00	96.00	100.00	96.00
/ɔ/	92.00	92.00	92.00	92.00
/æ/	21.74	21.74	21.74	21.74
/Θ/	75.00	79.17	62.50	75.00
/d/	94.00	94.00	94.00	94.00
/eɹ/	55.60	61.22	86.49	90.23
/Θr/	12.00	20.00	28.00	24.00
/ri/	0.00	16.67	8.33	25.00
Average :	72.99	77.07	78.36	78.23

Table 4.29 **Word Recognition for Gray_encoding & 8-tuple Linear_mapping**
"Averaged points interpolation method" for Time Normalisation

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	25.0	75.0	58.3	58.3
Run	52.0	80.0	84.0	96.0
Want	84.0	80.0	92.0	96.0
Begun	62.5	70.8	62.5	66.7
Wonder	20.0	64.0	52.0	44.0
Rudder	24.0	24.0	4.0	0.0
Win	36.0	36.0	48.0	60.0
Two	80.0	80.0	84.0	100.0
Shoe	88.0	92.0	84.0	96.0
Toot	32.0	20.0	8.0	16.0
Tattoo	16.0	16.0	20.0	28.0
Toothache	12.5	0.0	4.2	8.3
Cooler	44.0	64.0	76.0	72.0
Tee	52.0	56.0	56.0	60.0
Three	4.0	12.0	4.0	4.0
See	40.0	48.0	56.0	60.0
Average :	42.00	51.11	49.56	54.08

Table 4.30 **Word Recognition for Gray_encoding & 8-tuple Random_mapping**
"Averaged points interpolation method" for Time Normalisation

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	52.0	87.5	87.5	87.5
Run	72.0	100.0	100.0	100.0
Want	80.0	84.0	84.0	92.0
Begun	58.3	79.2	75.0	58.3
Wonder	48.0	68.0	68.0	48.0
Rudder	52.0	40.0	28.0	16.0
Win	12.0	28.0	44.0	40.0
Two	92.0	88.0	84.0	88.0
Shoe	96.0	92.0	96.0	96.0
Toot	60.0	52.0	52.0	36.0
Tattoo	20.0	24.0	20.0	24.0
Toothache	12.5	8.3	16.7	29.2
Cooler	52.0	76.0	76.0	68.0
Tee	56.0	48.0	56.0	56.0
Three	0.0	4.0	0.0	0.0
See	36.0	32.0	28.0	32.0
Average :	49.93	56.94	57.20	54.44

	b	ɪ	g	Λ	n	k	u	l	ə	r	s	i	t	w	ʃ	ɔ	æ	θ	d	eɪ	θr	ri	Samples
b	23	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	24
ɪ	1	45	.	.	.	.	2	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	49
g	.	.	24	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	24
Λ	.	.	.	96	.	.	.	.	15	.	.	.	6	.	.	6	.	.	.	.	.	.	123
n	.	.	.	.	146	.	1	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	148
k	.	.	1	.	.	48	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	49
u	1	16	2	.	9	.	144	.	.	.	.	7	2	.	.	.	.	.	.	.	.	.	181
l	.	1	.	.	.	.	5	19	.	.	.	.	.	.	.	.	.	.	.	.	.	.	25
ə	.	.	1	.	1	.	.	.	108	.	.	.	.	.	.	.	1	.	.	.	.	.	111
r	.	.	.	.	.	.	.	.	.	49	.	.	.	.	.	.	.	.	1	.	.	.	50
s	.	.	.	.	.	.	.	.	.	.	22	.	3	.	.	.	.	.	.	.	.	.	25
i	.	2	2	.	8	.	38	.	.	.	.	46	.	.	.	.	.	.	.	.	.	.	96
t	.	.	.	.	.	3	3	.	.	.	.	.	193	.	.	.	.	.	.	.	.	.	199
w	.	.	.	.	.	.	11	.	.	2	.	.	.	84	.	.	.	.	2	.	.	.	99
ʃ	.	.	.	.	.	.	.	.	.	.	.	.	.	.	25	.	.	.	.	.	.	.	25
ɔ	.	.	.	.	.	.	.	.	2	.	.	.	.	.	.	23	.	.	.	.	.	.	25
æ	2	2	2	.	.	1	6	.	.	.	.	.	3	.	.	.	5	.	2	.	.	.	23
θ	.	.	1	.	.	.	.	.	.	.	.	.	8	.	.	.	.	15	.	.	.	.	24
d	.	.	.	.	2	.	.	.	.	.	.	.	1	.	.	.	.	.	47	.	.	.	50
eɪ	.	.	.	.	2	.	2	.	.	.	.	.	.	.	.	.	.	.	1	32	.	.	37
θr	.	.	.	.	.	1	.	.	.	.	.	.	17	.	.	.	.	.	.	.	7	.	25
ri	.	.	1	.	.	.	6	.	.	.	.	.	1	.	.	.	.	.	3	.	.	1	12

Table 4.31 Confusion matrix for recogniser with Gray encoding / 8-tuple Random mapping and 15 trainings

Word recognition accuracy of 54.08% is achieved with linear mapping of the n-tuples and 20 trainings (Table 4.29). This performance is lower than that achieved with random mapping (57.2%). Fewer trainings were also required in the random mapping case (15 trainings).

Table 4.31 gives the confusion matrix for the phoneme-like segments for the recogniser with Gray encoding/8-tuple random mapping and 15 trainings (since this gave the best word recognition accuracy). With the exception of recognition of /æ/, /Θr/ and /ri/, the performance of the speech recogniser is encouraging. The poor recognition of the segments /Θr/ and /ri/ suggests that the 'sum of channel difference' segmentation method used at the segmentation stage of the speech recogniser may be inconsistent in the segmentation of the word 'Three'. This would cause variabilities in the speech data for these segments and hence making them difficult to recognise. The confusion of /Θr/ with /t/ is due to the greater number of /t/ samples in the training set thus the /t/ discriminator was trained to a greater degree than the /Θr/ discriminator. This is confirmed by the confusion of /Θ/ with /t/. The greater training of the /t/ discriminator causes it to show stronger response to spectral data similar to that on which it was trained (/Θr/, /Θ/ and /t/ all start with a burst). The poor recognition of /æ/ is due to inconsistency in the articulation of this phone in the word 'Tattoo'. The speaker tended to shorten this phone and in two samples of the utterance, it was deleted. The segment /i/ tends to get confused with /u/. This suggests similarity between spectral data for /i/ and /u/. However, the fact that /i/ is more frequently confused with /u/ than vice-versa is probably because there are more samples of /u/ in the training set than of /i/ (Table 4.1), hence the discriminator assigned to /u/ has a tendency to respond stronger. Some samples of /u/ are

confused with the segment /ɪ/. These are samples of /u/ that were produced when the segmentation process partitioned the sustained vowel /u/ into two segments (Section 3.7,Chapter 3).

The low word recognition accuracy makes it necessary to incorporate a linguistic processor in the design of the speech recogniser. The linguistic processor would be the final stage in the speech recognition system and its function is to perform error correction on the class labels (corresponding to the phoneme-like segments in the word) output by the 'Recognition Module' and hence improve the word recognition accuracy. This subject is covered in Chapter 6.

The 'Separate Segmentation and Labelling' method for the sub-word approach to recognition of isolated words is just one scheme that is implementable with the WISARD n-tuple pattern recogniser. In the next chapter, the 'Sliding Window' approach is used to secure the same goal.

4.10 Summary

A design for a WISARD n-tuple isolated word speech recogniser based on the 'Separate Segmentation & Labelling' (SS&L) approach has been presented in this chapter. The spectral representation of the utterance input to the speech recogniser is obtained from a 16 Channel filterbank IC. Using the segmentation method suggested in Chapter 3, the spectral data is partitioned into phoneme-like segments. Since the phoneme-like segments have varying durations, they are converted to a fixed size by time normalisation using the linear interpolation method. For data reduction, the 8-bit spectral data from the filter bank is encoded to 4-bits using the Linear, Thermometer or Gray encoders. Improvements in performance were achieved with spectral channel normalisation and noise subtraction normalisation. Various configurations are possible for the speech recogniser design. A series of experiments with different configurations are conducted to arrive at the setup for the speech recogniser that gives the best recognition performance. The best recognition accuracy for the phoneme-like segments in the word-set used in these experiments is 79.61%. The best word recognition accuracy achieved is 57.2%. The word recognition accuracy of the speech recogniser may be improved by incorporating a linguistic processor into the design of the speech recognition system (Chapter 6).

CHAPTER FIVE

RECOGNISER DESIGN BASED ON THE SLIDING WINDOW TECHNIQUE

5.1 Introduction

A design for a WISARD n-tuple phoneme-like segment speech recogniser based on the sliding window approach is presented in this chapter. This method involves sliding a window of a fixed duration over the spectral data of the utterance, and labelling the windowed data as belonging to one of the phoneme-like segments. In this design, two window sizes are used. This idea is similar to the centisecond labelling technique and in fact the speech recogniser design discussed in this chapter was initially based on this method but was later modified to the sliding window approach for reasons given in the next section.

5.2 Design Based on the Centisecond Labelling Approach

In the centisecond labelling technique [1], spectral frames are averaged over a 10 ms time duration to produce a centisecond frame. Each centisecond frame is submitted to the recognition module for labelling. Consecutive centisecond frames with the same label are grouped together to form a single segment with that label. In this way, the whole utterance is segmented and recognised at the same time.

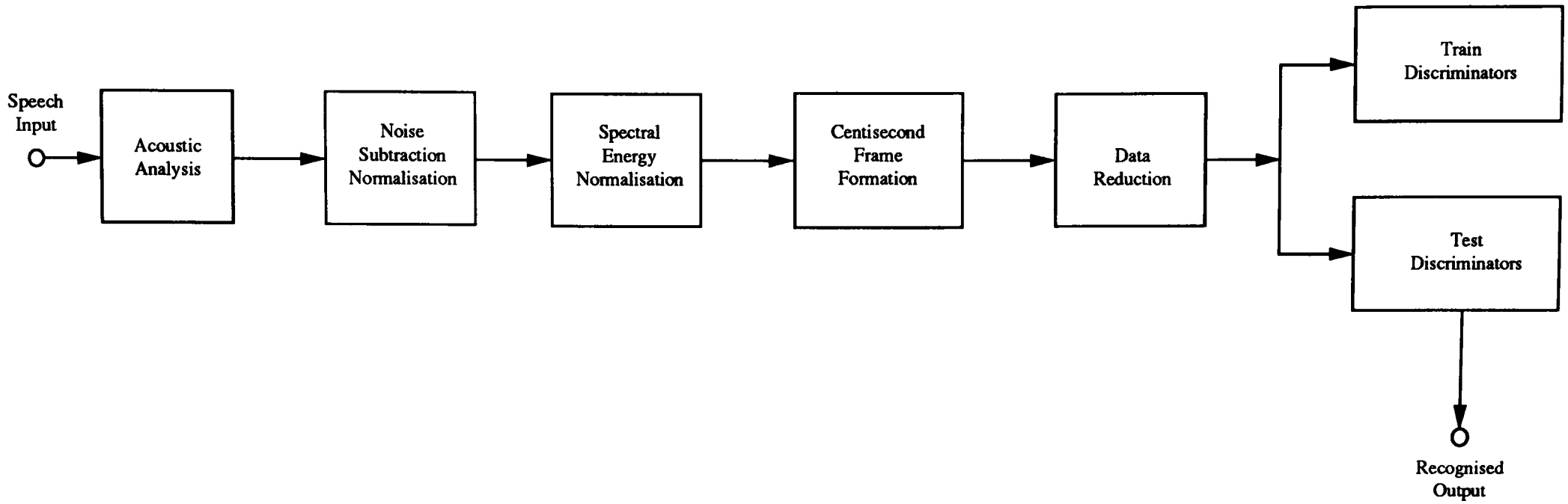


Figure 5.1 Recogniser design based on the centisecond labelling approach

The overall layout of the recogniser design based on this concept is shown in figure 5.1. Preliminary experiments with this design suggested that certain modifications were necessary. Since each spectral frame from the filterbank represents 3.2 ms of speech, three such frames are averaged into a centisecond frame (the time duration for three frames is actually 9.6 ms, but this is approximately a centisecond). After the data reduction stage, each of the 8-bit 16 channel samples in each centisecond frame is reduced to 4 bits, thus each centisecond frame is a single column of (16 x 4) bits (16 channels encoded as 4 bits by the data reduction stage). As the centisecond frame is the unit of recognition, the WISARD 'retina' is therefore a single column of 16 x 4 bits. This implies that for a tuple size of 4, only 16 4-tuples are needed to cover the whole 'retina', and therefore only 16 2^4 x 1 bit RAMs are required per class discriminator, and in the 8-tuple case, only 8 2^8 x 1 bit RAMs per class discriminator. Although at first this appears attractive on the basis of discriminator memory requirements for the design, tests showed that this discriminator size is too small since training causes saturation of the discriminator RAMs.

5.3 Design Based on the Sliding Window Approach

A solution to this problem is not to average the spectral frames over 10 ms into a single frame but to consider spectral frames in groups of three frames instead ie. to have a window the size of three frames which is slid across the spectral data for the utterance. Thus although the spectral frames are being considered over the same time duration, the number of RAMs per class discriminator in a 4-tuple

system has been increased from 16 to 48, and from 8 to 24 for the 8-tuple system. This discriminator size is sufficient for the short phoneme-like segments such as /b/ and /ɪ/ but not for the discriminators assigned to the longer phoneme-like segments such as /s/, /u/ and /i/. A larger discriminator size is needed for recognition of these segments. Furthermore, as there are many more frames in the longer phoneme-like segments than in the shorter ones, the discriminators assigned to these longer segments will have far more training samples than discriminators assigned to the shorter ones and therefore may tend to show greater response to the input speech patterns (on account of their greater training) than the shorter segment discriminators. This would increase the chances of misclassification of the shorter segments.

As a result of these problems, the approach taken is to recognise spectral frames from the shorter phoneme-like segments separately from those belonging to the longer segments ie. to have two separate banks of discriminators; one for the shorter segments, the other for the longer ones. The discriminators assigned to the shorter phoneme-like segments can process spectral data over a time duration of around 10 ms, whereas the longer phoneme-like segment discriminators may work over a much greater time duration. This requires that a mechanism be incorporated into the system design of figure 5.1 that will determine the length of the segment to which the unknown input spectral frames belong, ie. a segmentation process, which will then enable the recognition module to select the appropriate discriminator bank. The modified system design is shown in figure 5.2. Although this is now a departure from the 'Recognition-then-Segmentation' philosophy of the centisecond labelling technique since the speech spectral data is being segmented beforehand to locate the segment boundaries and hence

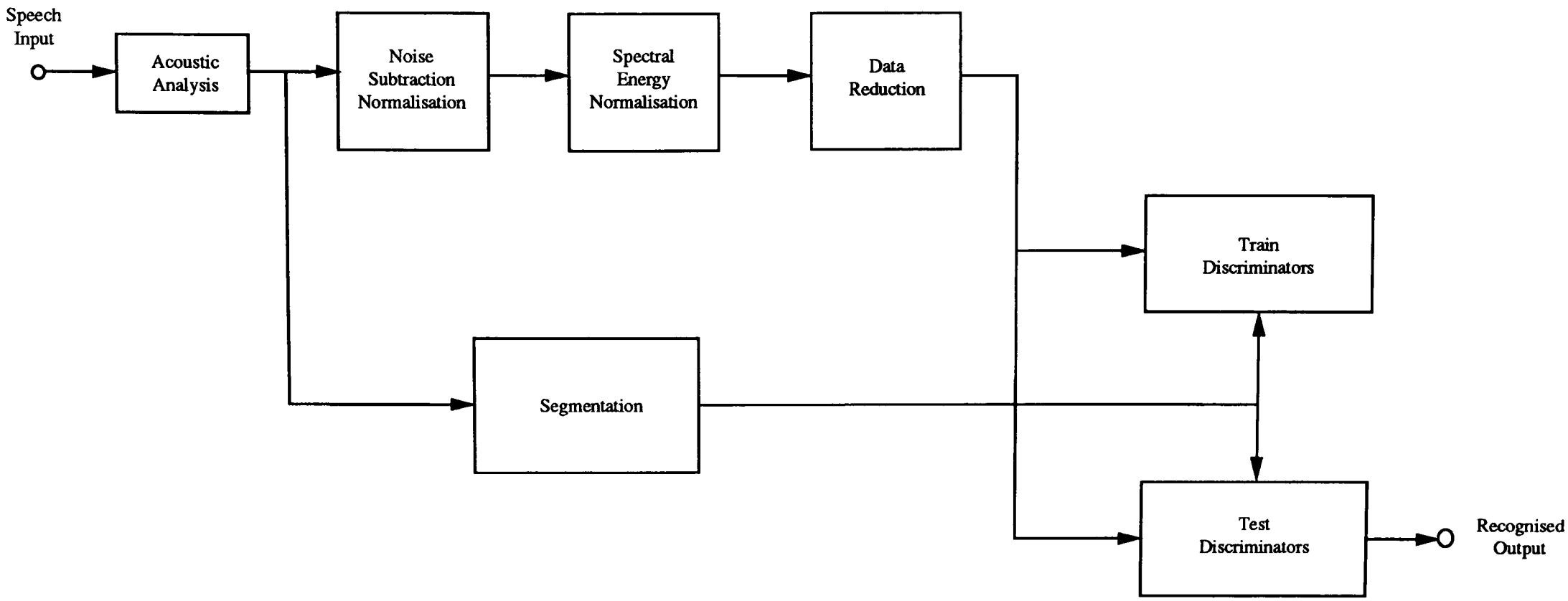


Figure 5.2 Recogniser design based on the sliding window approach

determine the segment duration, it still has in common the idea of recognising the phoneme-like segment by labelling spectral data as a sequence of subsegments that are around 10 ms in duration. In this recogniser design, two subsegment sizes are used; four frames (ie. 12.8 ms) for short phoneme-like segments, and fifteen frames (ie. 48 ms) for the long phoneme-like segments. This size is chosen because phoneme perception experiments suggest that a certain duration of speech signal is required for phoneme identification. Upto 30 ms is required for vowels ([112]) and 30 ms or longer for stops ([113]).

The stages in the recogniser system design are shown in figure 5.2. The acoustic analysis, noise subtraction, channel normalisation, segmentation and data reduction stages are the same as those in the recogniser design in Chapter 4. This setup was chosen for the design as it gave the best results with the recogniser design of Chapter 4.

5.3.1 The Recognition Module

Figure 5.3 shows the recognition module for this design. There are two banks of discriminators. Discriminator bank 'S' is assigned the recognition of short phoneme-like segments such as /b/ and /ʧ/, and discriminator bank 'L' is assigned the recognition of long phoneme-like segments such as /s/, /u/ and /i/.

The segmentation stage uses the 'sum of channel difference' segmentation method of Chapter 3 to determine the segment boundaries, and hence the segment length for the input spectral frames. This information is passed on to the Discriminator Bank Selector.

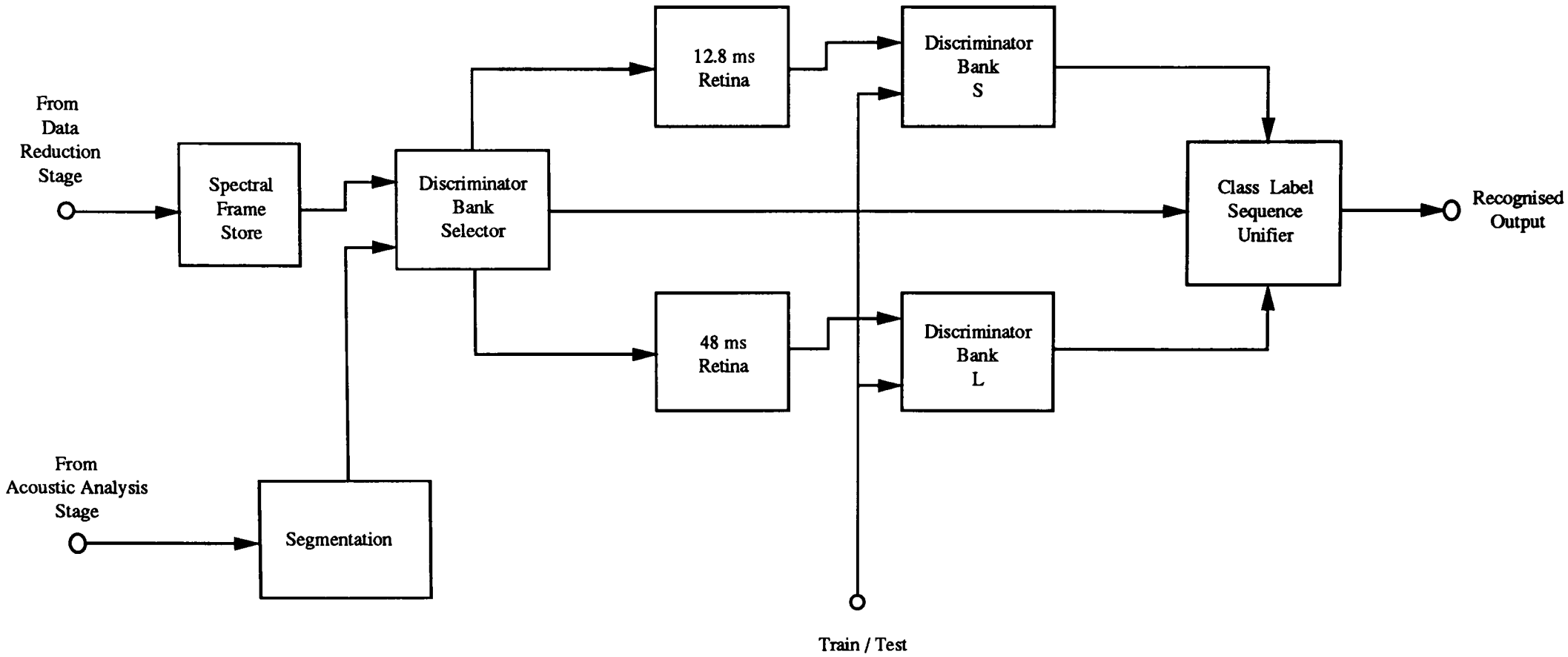


Figure 5.3 Layout of the Recognition Stage

Depending on the length of the segment, the Discriminator Bank Selector selects the appropriate discriminator bank to operate on the data in the spectral frame store.

Discriminator bank 'S' has a 'retina' size of (16 x 4) x 4 bits ie. it operates on four data reduced spectral frames ie. 12.8 ms time duration since each spectral frame is of a duration of 3.2 ms. The 'retina' for discriminator bank 'L' operates on fifteen data reduced spectral frames and therefore has a size of (16 x 4) x 15 bits, which is equivalent to a time duration of 48 ms. Discriminator bank 'L' has been assigned a bigger 'retina' for the reasons discussed earlier in Section 5.2.

To keep the design flexible, all the phoneme-like segments to be recognised are represented in both discriminator banks. This was done to avoid having to determine which phoneme-like segments are to be represented in either discriminator banks in order to assign the discriminators to the phoneme-like segments. Depending on the segment length threshold used by the Discriminator Bank Selector to select the appropriate discriminator bank, some of the phoneme-like segments will be represented in only one of the discrminator banks while some will be represented in both banks.

The memory requirements for the discriminator banks are as follows :

(i) *Discriminator Bank 'S' (short phoneme-like segments)*

The 'retina' size is (16 x 4) x 4 bits.

For an n-tuple size of 4, 64 4-tuples are needed to map the 'retina'. For each 4-tuple there is a corresponding 2^4 x 1 bit RAM in the class discriminator. Therefore each class discriminator has

a size of 1K bits (64 x 2^4). Since there are 22 phoneme-like segments to be recognised, and hence 22 class discriminators, discriminator bank 'S' therefore has a size of 22K bits.

Similarly, for the 8-tuple case, 32 8-tuples are required to map the 'retina'. This implies that each discriminator has a size of 8K bits (32 x 2^8), and therefore discriminator bank 'S' has a size of 176K bits.

(ii) *Discriminator Bank 'L' (long phoneme-like segments)*

The 'retina' size is (16 x 4) x 15 bits.

For the 4-tuple case, 240 4-tuples are needed to map the 'retina'. This gives each class discriminator a size of 3,840 bits, and hence discriminator bank 'L' a size of 84,480 bits.

For the 8-tuple case, 120 8-tuples are needed to map the 'retina'. This gives each class discriminator a size of 30,720 bits, and hence discriminator bank 'L' a size of 675,840 bits.

Therefore, the total discriminator memory requirement for this recogniser design is 107,008 bits for the 4-tuple based recogniser, and 856,064 bits for the 8-tuple based recogniser.

5.3.2 Spectral Frame Store Sliding Window

After data reduction, the 4-bit encoded spectral data for the whole isolated word is stored in the spectral frame store. By applying an energy threshold in the manner suggested by Rabiner & Sambur [114], the start of the utterance is detected. From this point, spectral frames equivalent to 1 second of speech (ie. 312 spectral frames) are retained by the spectral frame store. The segmentation stage identifies the frames in the spectral frame store that mark the start and end of the phoneme-like segments in the word. A window is opened in the spectral frame store, and is aligned with the start of the phoneme-like segment. The window has the same size and dimensions as the 'retina', ie. (16 x 4) x 4 bits if discriminator bank 'S' is selected, or (16 x 4) x 15 bits if discriminator bank 'L' is selected. The spectral frames that lie within this window are mapped onto the corresponding 'retina'. After operating on the bits mapped in the 'retina' (ie. n-tupling the 'retina' and training or testing the discriminators) the window is moved forward one frame as shown in figure 5.4. The spectral frames within the window are again mapped onto the retina and operated on. This process is repeated until all the spectral frames within the phoneme-like segment in the spectral frame store have been processed. Thus this method involves sliding a window across the phoneme-like segment one frame at a time, and operating on the contents of the window at these window positions.

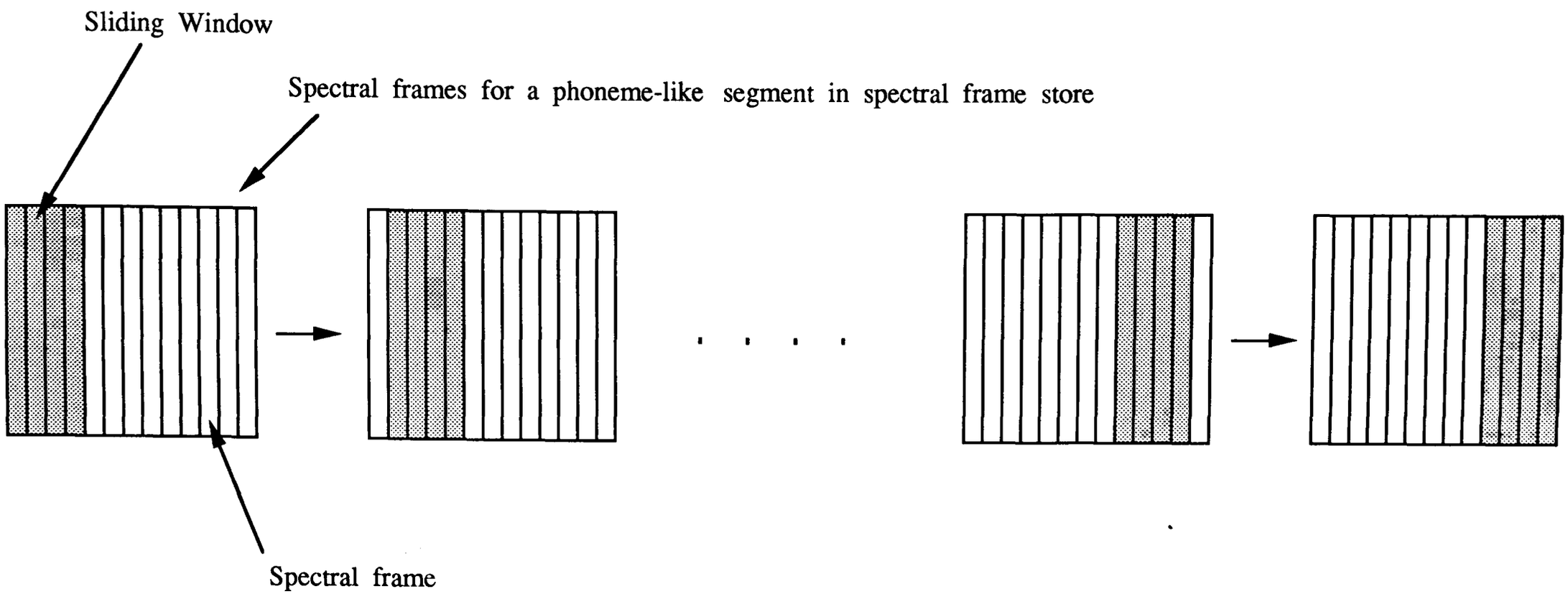


Figure 5.4 Window sliding across the spectral frames of a phoneme-like segment in the spectral frame store

5.4 Training the Recogniser

To train the recogniser, the training samples are the same as those in the training set given in Table 4.1 (Chapter 4, Section 4.7) used to evaluate the recogniser design in Chapter 4. This will enable a comparison of the performance of these two recogniser designs.

Before training of the discriminators can commence, all discriminators in both discriminator banks are cleared. The training samples after passing through the preprocessor and data reduction stages are loaded into the spectral frame store. The segmentation stage identifies the frames associated with the phoneme-like segment to be trained. Depending on the length of this segment, a 12.8 ms or 48 ms window is aligned with the start of the segment and the corresponding discriminator bank is selected. The class discriminator assigned to the phoneme-like segment being trained is set to train mode. The 4-bit encoded spectral data within the window is mapped onto the 'retina'. The bits in the 'retina' are then grouped into n-tuples (4-tuple or 8-tuple) using linear or random mapping, and a '1' is written into the selected discriminator's RAM cells addressed by the n-tuples. The window is moved forward by one frame and the data within the window is trained into the discriminator as before. Thus to train the discriminator on the whole phoneme-like segment, the window is slid one frame at a time from the start to the end of the segment. At each stage the data within the window is trained into the discriminator. This process is repeated for all the phoneme-like segments in the training set.

5.5 Testing the Recogniser

The procedure for testing the recogniser is similar to that for training. The 4-bit encoded spectral data from the unknown input word is loaded into the spectral frame store. The segmentation stage marks the boundaries of the phoneme-like segments in the unknown input word. The phoneme-like segments are to be recognised in the sequence in which they occur in the unknown input word. Starting with the first phoneme-like segment, a 12.8 ms or a 48 ms window (depending on the length of this phoneme-like segment) is aligned with the start of the segment, and the corresponding discriminator bank is selected. All the class discriminators in the selected discriminator bank are set to test mode. The data within the window is mapped onto the 'retina'. Using linear or random mapping, the 'retina' is mapped to n-tuples (4-tuples or 8-tuples). RAM locations addressed by these n-tuples are read for each class discriminator in the selected discriminator bank. The output of each discriminator (called the discriminator score) is the sum of the contents of all RAM cells (in that discriminator) addressed by the n-tuples mapped from the 'retina'. The spectral data within the window is assigned the label of the class discriminator giving the highest score. The window is then moved forward by one frame and the process for labelling the window contents is repeated. Thus to recognise the phoneme-like segment, the window is slid from the start to the end of the segment and the spectral data windowed is labelled for each window position as it slides across the segment. The remaining phoneme-like segments in the unknown input word are processed in the same manner.

As the window is slid across the spectral data for the phoneme-like segment, the recogniser outputs a class label for the windowed spectral data at each window position. Thus the result of recognition is

a sequence of class labels representing the phoneme-like segment. Since the spectral data scanned by the window belong to the same phoneme-like segment, it would be expected that all the labels in the label sequence output by the recogniser are of the same class. Frequently however, the label sequence contains labels of other classes due to misclassification by the recogniser. Therefore there is a need to unify this label sequence into a single class label as the final decision as to the identity of the unknown phoneme-like segment. This function is performed by the 'Class Label Sequence Unifier'.

5.5.1 Class Label Sequence Unifier

Instead of outputting a single class label as the recognition result at each window position across the phoneme-like segment, the recognition process generates a list of alternative choices at each window position as the window slides across the phoneme-like segment ie. a lattice of class labels. Each column in the lattice gives the possible choices of class labels for that window position over the segment. The columns are arranged such that the class label of the discriminator giving the highest recognition score is at the top of the column. The next position in the column is occupied by the class label of the discriminator that gives the next highest score and so on. A final overall score for each class is obtained by summing the positions that the class label occupies in each column in the lattice. The column positions are treated as penalty points assigned to the class label occupying that column position. The top of the column is position '0' and has a penalty of 0 point. The penalty is therefore numerically equal to the column position. Thus the lower down the column a class label is, the less likely it is that it is the correct class and hence the

higher the penalty assigned to it. The class label that produces the lowest score when each class label's position is summed across all columns in the lattice, is chosen as the identity of the unknown phoneme-like segment being recognised. This technique is based on the expectation that when a misclassification occurs at any window position over the phoneme-like segment, the discriminator that actually represents that phoneme-like segment will be among the top scoring discriminators for that window position and hence its class label should be found near the top of the lattice column. Consequently the penalty will be small and the misclassification will therefore not significantly affect the penalty score. The following example illustrates this method.

(0)	d*	d*	d*	b	ɪ *	ɪ	ɪ	ɪ	ɪ
(1)	b*	b*	b*	d*	b*	b	d	d	d
(2)	ɪ	ɪ	ɪ	ɪ*	d	d	d	d	d
(3)	l	l	l	l	l	l	l	l	l
(4)	u	u	u	u	æ	æ	i	i	i
(5)	æ	æ	æ	æ	u	i	æ	æ	æ
(6)	i	i	i	i	i	u	u	u	u

Figure 5.5 Lattice output for recognition of /ɪ/

Figure 5.5 shows the lattice input to the Class Label Sequence Unifier for the phoneme-like segment /ɪ/ (only the first seven choices are shown for each column in order to simplify the illustration). Class discriminators that gave identical scores ie. the scores 'tied', are marked by a '*'. In this case each of the classes involved are assigned

the same penalty, P. If there are n 'tied' classes, then P is given by the sum of the positions occupied in the lattice by these classes, p_i, divided by the number of 'tied' classes, n, as defined by (5.1).

$$P = \sum_{i=0}^{n-1} p_i \; / \; n \qquad (5.1)$$

For example, from figure 5.5, /d/ and /b/ are 'tied' at positions '0' and '1' respectively for lattice column '0'. Applying (5.1), each are assigned a penalty of 0.5 ((0+1)/2 = 0.5).

Summing the positions in the lattice for each class and applying (5.1) where classes are 'tied', the penalty scores are as follows:

/ɹ/ --> 8.0

/b/ --> 9.0

/d/ --> 10.0

/l/ --> 27.0

/æ/ --> 43.0

/u/ --> 45.0

/i/ --> 47.0

The segment is recognised as /ɹ/ since it has the lowest penalty score.

Another approach to reduce the lattice to a single class decision regarding the identity of the phoneme-like segment is to sum the discriminator scores across the lattice for each class. The class with the highest overall score is chosen as the identity of the phoneme-like segment being recognised.

5.6 Evaluation of the Recogniser

To arrive at a configuration for the recogniser that gives the best results, the following options were experimented with :

(i) 'Direct mapping' or 'Across channel mapping' of spectral frame store to WISARD 'retina'.
(ii) Linear or Gray encoding for data reduction.
(iii) Linear mapping or Random mapping of WISARD 'retina' to n-tuples.
(iv) Tuple size (4-tuple and 8-tuple)

For the data reduction option, the Thermometer encoder was not included in these experiments because the results obtained with the speech recogniser design of Chapter 4 showed it to be much inferior than the Linear and Gray encoders.

A 'segment length threshold' is used to select the discriminator bank 'S' or 'L'. For the experiments in Sections 5.5.2 and 5.5.3, the 'segment length threshold' was set to 20, ie. if the length of a phoneme-like segment is less than 20 frames, then it is considered as a short phoneme-like segment and Discriminator Bank 'S' would be selected to train/recognise it, otherwise it would be considered as a long phoneme-like segment and hence Discriminator Bank 'L' is selected. In Section 5.5.3, other values for the 'segment length threshold' are experimented with to improve the recognition accuracy of the recogniser.

5.6.1 Experiments with 'Direct Mapping' of Spectral Frame Store

Tables 5.1 - 5.8 give the results obtained for this set of experiments.

For the 4-tuple case with both Linear and Gray encoders, recognition performance peaks at 15 trainings, whereas performance improves with training for the 8-tuple case, with the exception of Gray encoding/8-tuple Random mapping (Table 5.8).

Comparing Table 5.1 with Table 5.5, and Table 5.3 with Table 5.7 shows that Linear encoding with linear mapping and Gray encoding with linear mapping are equivalent.

For the Linear encoder, better results are obtained with linear mapping than random mapping, whereas for the Gray encoder, random mapping gives better results than linear mapping.

The highest recognition accuracy is 72.23% achieved with Gray encoding/8-tuple random mapping, and 15 trainings (Table 5.8).

Table 5.1

Linear_encoding & 4-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	91.67	100.00
/ʧ/	22.92	31.25	35.42	31.25
/g/	87.50	83.33	83.33	79.17
/ʍ/	16.20	21.07	15.40	22.60
/n/	71.69	76.42	68.31	71.00
/k/	50.17	56.42	66.83	66.83
/ω/	28.15	51.12	41.16	40.13
/l/	16.00	24.00	28.00	28.00
/ə/	89.94	89.94	86.76	83.32
/r/	38.00	50.00	60.00	58.00
/s/	40.00	80.00	68.00	28.00
/ɪ/	78.21	27.25	19.62	20.83
/t/	92.00	69.42	57.88	47.79
/w/	4.04	19.17	38.33	37.42
/ʃ/	100.00	100.00	100.00	96.00
/ɔ/	16.00	16.00	20.00	32.00
/æ/	8.70	4.35	8.70	4.35
/Θ/	33.33	33.33	37.50	25.00
/d/	58.00	60.00	62.00	62.00
/eɹ/	58.81	66.82	74.84	65.07
/Θr/	0.00	52.00	48.00	48.00
/ri/	50.00	16.67	25.00	25.00
Average :	47.98	51.11	51.67	48.72

Table 5.2

Linear_encoding & 4-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	95.83
/ʧ/	27.08	24.67	29.00	20.67
/g/	37.50	79.17	79.17	58.33
/ʍ/	26.67	35.67	34.07	32.43
/n/	79.69	77.69	71.61	66.89
/k/	48.08	52.25	64.75	70.92
/ω/	68.75	56.22	40.36	35.40
/l/	16.00	24.00	28.00	36.00
/ə/	90.11	88.20	85.48	86.43
/r/	46.00	68.00	80.00	76.00
/s/	4.00	20.00	36.00	44.00
/ɪ/	54.62	13.52	10.81	14.90
/t/	93.00	69.79	63.27	51.62
/w/	14.25	35.54	39.62	35.54
/ʃ/	0.00	0.00	0.00	4.00
/ɔ/	4.00	44.00	44.00	60.00
/æ/	0.00	0.00	0.00	0.00
/Θ/	12.50	25.00	33.33	33.33
/d/	58.00	58.00	58.00	58.00
/eɹ/	74.20	75.96	72.12	70.35
/Θr/	8.00	24.00	24.00	24.00
/ri/	25.00	58.33	58.33	50.00
Average :	40.15	46.63	47.63	46.57

Table 5.3

Linear_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	91.67	87.50	91.67	95.83
/ʧ/	27.00	29.08	37.25	41.25
/g/	58.33	79.17	83.33	70.83
/ʍ/	43.33	51.33	40.63	47.17
/n/	94.61	89.83	91.19	93.22
/k/	83.33	89.58	93.75	95.83
/ω/	58.53	61.93	63.69	69.31
/l/	24.00	28.00	32.00	32.00
/ə/	89.86	91.06	88.98	90.76
/r/	65.33	66.00	64.00	68.00
/s/	36.00	80.00	76.00	76.00
/ɪ/	37.65	40.56	36.37	37.64
/t/	93.48	91.94	87.90	89.90
/w/	35.29	29.33	28.50	50.62
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	36.00	44.00	56.00	68.00
/æ/	13.04	4.35	4.35	13.04
/Θ/	16.67	33.33	50.00	41.67
/d/	58.58	62.00	62.00	64.00
/eɹ/	59.46	67.15	70.99	72.76
/Θr/	16.00	36.00	44.00	60.00
/ri/	8.33	41.67	25.00	8.33
Average :	52.11	59.26	60.35	63.01

Table 5.4

Linear_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	83.33	91.67	95.83	95.83
/ʧ/	37.50	41.33	37.33	37.33
/g/	20.83	66.67	75.00	79.17
/ʍ/	23.60	25.23	30.97	36.67
/n/	98.00	93.19	95.28	93.28
/k/	46.17	52.25	68.92	77.17
/ω/	64.76	71.54	72.18	74.57
/l/	16.00	20.00	28.00	28.00
/ə/	93.50	89.61	90.06	89.11
/r/	50.00	60.00	78.00	84.00
/s/	4.00	20.00	36.00	24.00
/ɪ/	69.94	76.75	65.27	53.54
/t/	94.00	94.00	93.00	93.00
/w/	14.12	35.46	45.58	44.62
/ʃ/	60.00	0.00	4.00	8.00
/ɔ/	20.00	60.00	72.00	80.00
/æ/	8.70	0.00	0.00	4.35
/Θ/	0.00	0.00	0.00	12.50
/d/	58.00	60.00	60.00	60.00
/eɹ/	66.51	80.12	80.45	80.45
/Θr/	0.00	0.00	0.00	0.00
/ri/	41.67	75.00	50.00	33.33
Average :	44.12	50.58	53.54	54.04

Table 5.5

Gray_encoding & 4-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	91.67	100.00
/t/	22.92	31.25	35.42	31.25
/g/	87.50	83.33	83.33	79.17
/N/	16.20	21.07	15.40	22.60
/n/	71.69	76.42	68.31	71.00
/k/	50.17	56.42	66.83	66.83
/u/	28.15	51.12	41.16	40.13
/l/	16.00	24.00	28.00	28.00
/ə/	89.94	89.94	86.76	83.32
/r/	38.00	50.00	60.00	58.00
/s/	40.00	80.00	68.00	28.00
/i/	78.21	27.25	19.62	20.83
/t/	92.00	69.42	57.88	47.79
/w/	4.04	19.17	38.33	37.42
/ʃ/	100.00	100.00	100.00	96.00
/ɔ/	16.00	16.00	20.00	32.00
/æ/	8.70	4.35	8.70	4.35
/Θ/	33.33	33.33	37.50	25.00
/d/	58.00	60.00	62.00	62.00
/eɹ/	58.81	66.82	74.84	65.07
/Θr/	0.00	52.00	48.00	48.00
/ri/	50.00	16.67	25.00	25.00
Average :	47.98	51.11	51.67	48.72

Table 5.6

Gray_encoding & 4-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	100.00
/t/	39.58	37.17	45.50	28.92
/g/	66.67	83.33	83.33	75.00
/N/	30.90	26.80	25.23	24.33
/n/	93.28	91.89	91.19	91.25
/k/	56.07	64.58	85.50	66.75
/u/	59.76	77.99	57.22	63.53
/l/	21.74	24.00	32.00	36.00
/ə/	91.80	92.11	90.06	88.67
/r/	50.00	56.00	62.00	58.00
/s/	25.00	68.00	84.00	80.00
/i/	69.75	45.29	43.16	32.75
/t/	93.50	91.98	92.00	89.40
/w/	11.12	44.71	58.79	30.62
/ʃ/	88.00	96.00	96.00	84.00
/ɔ/	16.00	52.00	64.00	68.00
/æ/	13.04	17.39	21.74	21.74
/Θ/	0.00	16.67	20.83	25.00
/d/	58.00	60.00	58.00	58.00
/eɹ/	78.04	84.29	84.29	84.29
/Θr/	4.00	16.00	36.00	32.00
/ri/	16.67	91.67	91.67	75.00
Average :	49.03	60.62	64.47	59.69

Table 5.7

Gray_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	91.67	87.50	91.67	95.83
/t/	27.00	29.08	37.25	41.25
/g/	58.33	79.17	83.33	70.83
/N/	43.07	51.33	40.63	47.17
/n/	94.61	89.83	91.19	93.22
/k/	83.33	89.58	93.75	95.83
/u/	58.53	61.93	63.69	69.31
/l/	24.00	28.00	32.00	32.00
/ə/	89.94	91.06	88.98	90.76
/r/	66.00	66.00	64.00	68.00
/s/	36.00	80.00	76.00	76.00
/i/	37.65	40.56	36.37	37.64
/t/	93.48	91.94	87.90	89.90
/w/	35.29	29.33	28.50	50.62
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	36.00	44.00	56.00	68.00
/æ/	13.04	4.35	4.35	13.04
/Θ/	16.67	33.33	50.00	41.67
/d/	58.00	62.00	62.00	64.00
/eɹ/	59.46	67.15	70.99	72.76
/Θr/	16.00	36.00	44.00	60.00
/ri/	8.33	41.67	25.00	8.33
Average :	52.11	59.26	60.35	63.01

Table 5.8

Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	95.83
/t/	43.58	45.42	53.58	53.58
/g/	75.00	83.33	83.33	87.50
/N/	58.07	54.57	46.47	46.53
/n/	82.69	98.64	98.67	97.31
/k/	85.50	93.75	95.83	95.83
/u/	56.17	77.56	74.60	79.46
/l/	32.00	32.00	32.00	36.00
/ə/	78.22	94.17	93.89	91.83
/r/	82.00	82.00	82.00	78.00
/s/	56.00	88.00	100.00	96.00
/i/	81.55	72.69	63.08	53.04
/t/	81.50	94.00	92.50	91.46
/w/	57.62	62.71	70.88	64.79
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	56.00	84.00	84.00	80.00
/æ/	17.39	17.39	17.39	26.09
/Θ/	0.00	8.33	33.33	33.33
/d/	64.00	60.00	62.00	62.00
/eɹ/	90.00	84.29	86.38	86.38
/Θr/	0.00	12.00	40.00	44.00
/ri/	41.67	58.33	83.33	50.00
Average :	60.67	68.14	72.23	70.41

5.6.2 Experiments with 'Across Channel Mapping' of Spectral Frame Store

Tables 5.9 - 5.16 give the results obtained for the recogniser with 'across channel mapping' of the spectral frame store. Tables 5.9 - 5.12 show the results obtained with the Linear encoder at the Data Reduction stage, and Tables 5.13 - 5.16 are for the Gray encoder.

For the 4-tuple case, recognition performance improves with training for linear mapping with both Linear and Gray encoders (Tables 5.9 and 5.13). With random mapping, recognition accuracy peaks at 15 trainings for both Linear and Gray encoders (Tables 5.10 and 5.14), and further training causes a drop in the recognition accuracy. For the 8-tuple based recognisers, recognition accuracy peaks at 15 trainings for all cases (Tables 5.11, 5.12, 5.15, 5.16).

The Gray encoder gives significantly better results than the Linear encoder, in fact, the recogniser with Gray encoding/4-tuple random mapping is superior to the 8-tuple based recogniser with Linear encoding.

Random mapping gives better results than linear mapping. The results obtained with Gray encoding/4-tuple random mapping are comparable with those for Gray encoding/8-tuple linear mapping.

Whereas experiments with 'direct mapping' of spectral frame store showed that Linear encoding with Linear mapping is equivalent to Gray encoding with Linear mapping (Section 5.5.1), these experiments show that this relation does not hold for 'across channel mapping' of the spectral frame store.

The best recognition accuracy is 69.99%, achieved with Gray encoding/8-tuple random mapping. This is lower than that obtained for the recogniser with 'direct mapping' of the spectral frame store (72.23%).

Table 5.9

Linear_encoding & 4-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	50.00	83.33	95.83	95.83
/ʧ/	18.58	27.00	14.41	18.67
/g/	8.33	20.83	29.17	41.67
/ʍ/	17.90	27.57	29.30	35.70
/n/	56.06	40.44	40.64	48.67
/k/	68.75	46.00	48.08	46.00
/u/	63.87	58.91	34.74	47.97
/l/	12.00	16.00	20.00	36.00
/ə/	57.28	77.96	77.67	73.78
/r/	24.00	44.00	40.00	44.00
/s/	4.00	36.00	32.00	8.00
/ʃ/	7.50	0.00	11.35	12.25
/t/	88.94	73.42	63.27	57.77
/w/	5.08	10.21	6.12	23.33
/ʃ/	0.00	0.00	0.00	0.00
/ɔ/	4.00	0.00	16.00	12.00
/æ/	0.00	0.00	0.00	0.00
/Θ/	4.17	12.50	25.00	20.83
/d/	58.00	58.00	56.00	56.00
/eʊ/	60.58	68.27	66.18	70.35
/Θr/	8.00	24.00	24.00	12.00
/ri/	0.00	16.67	8.33	8.33
Average :	28.05	33.69	33.55	34.96

Table 5.10

Linear_encoding & 4-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	100.00	100.00
/ʧ/	27.08	27.08	22.92	12.41
/g/	45.83	83.33	83.33	62.50
/ʍ/	29.23	33.27	34.93	34.07
/n/	67.53	69.50	72.33	70.39
/k/	48.08	48.08	52.25	56.42
/u/	68.39	51.75	27.81	43.38
/l/	16.00	24.00	24.00	28.00
/ə/	93.28	89.83	82.00	77.82
/r/	60.00	66.00	84.00	70.00
/s/	12.00	28.00	32.00	44.00
/ʃ/	48.91	35.24	19.95	21.98
/t/	89.94	68.85	57.25	42.58
/w/	18.17	45.54	47.62	44.58
/ʃ/	0.00	0.00	0.00	0.00
/ɔ/	4.00	40.00	60.00	64.00
/æ/	13.04	4.35	4.35	4.35
/Θ/	12.50	25.00	20.83	25.00
/d/	56.00	58.00	58.00	58.00
/eʊ/	81.90	78.04	76.28	84.29
/Θr/	4.00	24.00	28.00	24.00
/ri/	8.33	66.67	66.67	58.33
Average :	40.91	48.29	47.93	46.64

Table 5.11

Linear_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	62.50	91.67	87.50	83.33
/ʧ/	27.08	33.33	35.33	29.17
/g/	37.50	41.67	20.83	12.50
/ʍ/	16.40	15.60	17.20	18.03
/n/	74.94	85.11	81.75	76.94
/k/	36.25	44.17	48.17	62.58
/u/	41.02	56.76	45.83	57.51
/l/	16.00	20.00	28.00	32.00
/ə/	83.76	84.20	76.46	82.08
/r/	36.00	26.00	32.00	38.00
/s/	56.00	52.00	76.00	52.00
/ʃ/	47.12	11.94	26.14	14.02
/t/	88.92	87.92	84.85	85.38
/w/	16.25	19.25	10.12	8.17
/ʃ/	52.00	48.00	40.00	36.00
/ɔ/	24.00	60.00	68.00	68.00
/æ/	17.39	13.04	8.70	8.70
/Θ/	0.00	0.00	0.00	0.00
/d/	60.00	58.00	58.00	58.00
/eʊ/	45.19	49.04	66.51	59.14
/Θr/	0.00	0.00	0.00	0.00
/ri/	8.33	25.00	25.00	8.33
Average :	38.48	41.94	42.56	40.45

Table 5.12

Linear_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	70.83	91.67	91.67	91.67
/ʧ/	33.25	33.33	39.42	33.25
/g/	16.67	66.67	75.00	70.83
/ʍ/	31.73	35.70	39.10	49.80
/n/	94.58	93.22	93.28	95.97
/k/	46.25	54.33	81.25	85.50
/u/	69.37	78.26	67.78	76.40
/l/	20.00	28.00	32.00	36.00
/ə/	92.17	91.44	90.06	90.78
/r/	52.00	54.00	76.00	78.00
/s/	12.00	16.00	20.00	24.00
/ʃ/	71.08	59.66	56.18	48.60
/t/	94.00	94.00	93.00	91.98
/w/	21.04	36.42	48.50	49.54
/ʃ/	56.00	12.00	8.00	0.00
/ɔ/	28.00	52.00	64.00	64.00
/æ/	13.04	8.70	13.04	13.04
/Θ/	0.00	0.00	8.33	12.50
/d/	58.00	58.00	58.00	60.00
/eʊ/	76.28	76.28	80.12	90.23
/Θr/	0.00	0.00	4.00	4.00
/ri/	50.00	50.00	75.00	41.67
Average :	45.74	49.53	55.17	54.90

Table 5.13

Gray_encoding & 4-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	100.00	91.67
/ʧ/	37.33	35.42	33.33	27.08
/g/	45.83	29.17	33.33	41.67
/ʌ/	9.90	17.13	20.43	34.13
/n/	74.28	64.17	71.67	73.75
/k/	66.75	70.92	89.67	79.25
/u/	50.88	80.05	59.54	64.65
/l/	28.00	28.00	32.00	32.00
/ə/	83.70	69.78	78.19	70.28
/r/	18.00	44.00	40.00	44.00
/s/	0.00	28.00	28.00	32.00
/ɲ/	52.75	2.52	9.48	11.48
/t/	94.50	90.46	81.31	73.81
/w/	16.17	15.21	5.12	27.42
/ʃ/	48.00	96.00	80.00	52.00
/ɔ/	4.00	12.00	20.00	20.00
/æ/	8.70	4.35	4.35	4.35
/Θ/	4.17	8.33	12.50	20.83
/d/	58.00	60.00	58.00	60.00
/eɪ/	72.43	54.97	66.82	74.52
/Θr/	0.00	24.00	16.00	12.00
/ri/	8.33	8.33	0.00	0.00
Average :	39.89	42.66	42.72	43.04

Table 5.14

Gray_encoding & 4-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	100.00	100.00	100.00	95.83
/ʧ/	35.42	29.17	31.25	27.00
/g/	75.00	83.33	83.33	75.00
/ʌ/	40.50	38.10	33.30	29.13
/n/	95.89	91.19	90.56	91.94
/k/	52.17	70.92	85.42	83.33
/u/	54.69	64.71	54.52	64.37
/l/	20.00	20.00	24.00	28.00
/ə/	88.67	89.33	89.39	89.39
/r/	64.00	74.00	76.00	62.00
/s/	24.00	64.00	64.00	56.00
/ɲ/	73.36	51.02	48.60	50.20
/t/	93.50	91.50	81.92	73.67
/w/	31.38	56.71	64.83	54.79
/ʃ/	84.00	92.00	92.00	88.00
/ɔ/	36.00	60.00	64.00	76.00
/æ/	13.04	0.00	0.00	4.35
/Θ/	16.67	33.33	33.33	62.50
/d/	58.00	58.00	58.00	58.00
/eɪ/	72.43	81.90	84.29	84.29
/Θr/	0.00	24.00	44.00	32.00
/ri/	25.00	50.00	50.00	25.00
Average :	52.44	60.15	61.49	59.58

Table 5.15

Gray_encoding & 8-tuple Linear_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	91.67	91.67	91.67	91.67
/ʧ/	47.67	53.83	43.50	35.25
/g/	75.00	83.33	79.17	70.83
/ʌ/	46.67	44.10	31.87	33.37
/n/	91.86	85.11	85.75	78.86
/k/	50.50	58.58	64.83	87.50
/u/	46.87	65.73	61.44	68.59
/l/	28.00	28.00	32.00	28.00
/ə/	88.50	88.67	88.00	88.65
/r/	74.00	72.00	76.00	74.00
/s/	68.00	76.00	92.00	92.00
/ɲ/	77.46	50.37	48.25	22.44
/t/	91.46	85.85	84.88	81.81
/w/	53.50	56.62	51.58	53.54
/ʃ/	88.00	100.00	100.00	96.00
/ɔ/	64.00	76.00	76.00	80.00
/æ/	17.39	17.39	17.39	21.74
/Θ/	12.50	20.83	25.00	20.83
/d/	60.00	66.00	62.00	62.00
/eɪ/	68.91	67.15	74.84	78.68
/Θr/	0.00	8.00	40.00	16.00
/ri/	16.67	41.67	66.67	33.33
Average :	57.21	60.77	63.31	59.78

Table 5.16

Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	95.83
/ʧ/	43.58	47.42	55.50	49.42
/g/	75.00	87.50	87.50	79.17
/ʌ/	65.20	62.70	52.10	54.60
/n/	98.64	95.94	97.31	98.64
/k/	81.33	89.58	97.92	95.83
/u/	58.57	79.13	72.97	76.97
/l/	28.00	36.00	32.00	32.00
/ə/	93.50	92.78	92.11	93.22
/r/	80.00	80.00	82.00	80.00
/s/	44.00	84.00	92.00	80.00
/ɲ/	79.55	66.79	56.90	50.81
/t/	93.50	92.50	90.42	90.44
/w/	50.62	59.67	67.79	70.83
/ʃ/	92.00	96.00	100.00	92.00
/ɔ/	64.00	76.00	80.00	80.00
/æ/	13.04	17.39	17.39	21.74
/Θ/	8.33	20.83	25.00	33.33
/d/	60.00	62.00	60.00	60.00
/eɪ/	82.53	82.53	86.38	86.38
/Θr/	4.00	16.00	32.00	28.00
/ri/	41.67	75.00	66.67	58.33
Average :	61.50	68.89	69.99	68.52

5.6.3 Experiments with the 'Segment Length Threshold'

The results of the experiments in Sections 5.5.1 and 5.5.2 show that the best results are obtained for the recogniser with Gray encoding and 8-tuple random mapping, and 'direct mapping' of the spectral frame store. This configuration is adopted for the experiments presented in this section. The aim of these experiments is to determine whether the recognition accuracy of the speech recogniser may be improved by using a different value for the 'segment length threshold' used to select the required discriminator bank in the recognition system.

Tables 5.17 - 5.23 give the results for this set of experiments. Segment length thresholds of 17 frames to 28 frames were used. These results show that the recognition accuracy may be improved from the 72.23% obtained with a 'segment length threshold' of 20 frames, to 73.97% for 'segment length threshold' of 24 frames (Table 5.21).

Table 5.24 gives the word recognition accuracy for the speech recogniser with a 'segment length threshold' of 24 frames. 40.49% word recognition accuracy is achieved with 20 trainings. As observed in the case of the speech recogniser design of Chapter 4, it is found that although a recogniser design may have the highest recognition accuracy at the phoneme-like segment level, this does not imply that it also has the highest word recognition accuracy. From Tables 5.21 and 5.24, it is seen that the recogniser with 15 trainings has the highest recognition accuracy at the phoneme-like segment level, but the recogniser with 20 trainings has a higher word recognition accuracy. About 50% of the word recognition errors were due to misclassification of only one of the phoneme-like segments in the word.

Table 5.17 Results for 'Segment Length Threshold' = 17
Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	95.83
/ʧ/	41.50	47.42	53.58	53.58
/g/	75.00	83.33	83.33	87.50
/ŋ/	63.23	62.50	51.23	52.13
/n/	98.67	98.64	98.67	97.97
/k/	75.08	91.67	97.92	95.83
/ω/	64.42	81.65	77.87	81.91
/l/	0.00	4.00	8.00	16.00
/ə/	94.11	94.83	93.89	93.17
/r/	82.00	84.00	84.00	82.00
/s/	56.00	88.00	100.00	96.00
/ʃ/	82.88	72.69	63.08	53.04
/t/	98.50	98.00	96.00	94.46
/w/	50.62	67.75	75.92	69.83
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	56.00	92.00	92.00	88.00
/æ/	17.39	17.39	17.39	26.09
/Θ/	4.17	8.33	33.33	33.33
/d/	48.00	60.00	56.00	54.00
/eɹ/	90.23	88.15	90.23	88.15
/Θr/	0.00	12.00	40.00	44.00
/ri/	41.67	58.33	83.33	50.00
Average :	60.70	68.48	72.34	70.58

Table 5.18 Results for 'Segment Length Threshold' = 18
Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	95.83
/ʧ/	43.58	45.42	53.58	53.58
/g/	75.00	83.33	83.33	87.50
/ŋ/	60.83	60.10	49.63	50.77
/n/	98.67	98.64	98.67	97.28
/k/	75.08	91.67	97.92	95.83
/ω/	63.18	80.41	77.05	81.49
/l/	8.00	20.00	20.00	28.00
/ə/	94.11	95.50	94.56	91.75
/r/	80.00	82.00	82.00	80.00
/s/	56.00	88.00	100.00	96.00
/ʃ/	82.21	72.69	63.08	53.71
/t/	97.00	97.00	95.00	93.96
/w/	50.62	66.75	74.92	68.46
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	56.00	88.00	88.00	84.00
/æ/	17.39	17.39	17.39	26.09
/Θ/	4.17	8.33	33.33	33.33
/d/	54.00	56.00	58.00	53.67
/eɹ/	88.15	86.06	88.15	86.06
/Θr/	0.00	12.00	40.00	44.00
/ri/	41.67	58.33	83.33	50.00
Average :	60.98	68.34	72.44	70.51

Table 5.19 Results for 'Segment Length Threshold' = 19
Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	95.83	95.83	95.83	95.83
/ʧ/	43.58	45.42	53.58	53.58
/g/	75.00	83.33	83.33	87.50
/ŋ/	59.33	59.37	50.47	50.53
/n/	98.67	98.64	98.67	97.31
/k/	75.08	91.67	97.92	97.92
/ω/	61.57	78.80	75.83	80.69
/l/	28.00	28.00	28.00	20.00
/ə/	92.11	94.83	94.56	92.50
/r/	84.00	82.00	84.00	80.00
/s/	56.00	88.00	100.00	96.00
/ʃ/	82.21	72.69	63.08	53.04
/t/	96.00	96.00	94.50	93.46
/w/	48.62	63.71	71.88	68.79
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	56.00	88.00	88.00	84.00
/æ/	17.39	17.39	17.39	26.09
/Θ/	4.17	8.33	33.33	33.33
/d/	54.00	54.00	54.00	56.00
/eɹ/	88.15	86.06	88.15	92.31
/Θr/	0.00	12.00	40.00	44.00
/ri/	41.67	58.33	83.33	50.00
Average :	61.70	68.29	72.54	70.59

Table 5.20 Results for 'Segment Length Threshold' = 22
Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	87.50	95.83	95.83	95.83
/ʧ/	43.50	47.42	57.58	55.58
/g/	70.83	79.17	79.17	83.33
/ŋ/	50.53	49.77	44.07	44.93
/n/	98.67	98.64	98.00	97.97
/k/	75.08	93.75	95.83	95.83
/ω/	58.28	75.50	73.04	79.79
/l/	48.00	60.00	60.00	64.00
/ə/	82.39	85.11	86.22	84.83
/r/	80.00	80.00	80.00	76.00
/s/	56.00	88.00	100.00	96.00
/ʃ/	80.21	72.69	64.54	55.04
/t/	91.00	91.00	89.50	88.46
/w/	41.58	52.67	60.83	56.75
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	48.00	88.00	84.00	80.00
/æ/	8.70	8.70	13.04	17.39
/Θ/	4.17	8.33	33.33	33.33
/d/	72.00	72.00	72.00	72.00
/eɹ/	74.52	76.60	76.60	78.68
/Θr/	0.00	12.00	40.00	44.00
/ri/	41.67	58.33	91.67	50.00
Average :	59.67	67.89	72.51	70.44

Table 5.21 Results for 'Segment Length Threshold' = 24
Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	83.33	95.83	95.83	91.67
/ʧ/	45.50	55.67	57.67	53.06
/g/	66.67	79.17	79.17	83.33
/ŋ/	42.50	40.07	38.43	41.46
/n/	98.64	98.64	98.00	95.95
/k/	75.08	91.67	95.83	95.92
/ω/	56.20	70.35	70.14	76.80
/l/	52.00	76.00	80.00	84.00
/ə/	75.04	82.76	88.20	95.50
/r/	74.00	74.00	74.00	72.00
/s/	56.00	88.00	100.00	96.00
/ʃ/	79.55	72.69	65.21	52.08
/t/	88.50	88.50	87.00	85.93
/w/	36.54	45.58	51.71	63.64
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	60.00	92.00	96.00	88.00
/æ/	8.70	8.70	13.04	17.39
/Θ/	4.17	8.33	33.33	33.33
/d/	84.00	84.00	84.00	84.00
/eɯ/	66.51	74.20	88.15	91.89
/Θr/	0.00	12.00	40.00	44.00
/ri/	41.67	58.33	91.67	50.00
Average :	58.85	68.02	73.97	72.54

Table 5.22 Results for 'Segment Length Threshold' = 26
Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	83.33	95.83	95.83	91.67
/ʧ/	53.75	59.92	65.92	61.75
/g/	62.50	83.33	79.17	79.17
/ŋ/	36.10	38.37	38.33	36.00
/n/	99.33	98.64	98.00	97.97
/k/	73.00	89.58	93.75	95.83
/ω/	56.97	72.15	70.09	76.46
/l/	80.00	80.00	84.00	88.00
/ə/	70.74	78.26	86.82	91.17
/r/	64.00	64.00	64.00	62.00
/s/	56.00	88.00	100.00	96.00
/ʃ/	78.21	68.61	63.21	53.04
/t/	88.00	88.00	86.50	85.46
/w/	30.17	52.58	60.67	72.75
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	68.00	96.00	92.00	88.00
/æ/	8.70	8.70	13.04	13.04
/Θ/	4.17	8.33	33.33	33.33
/d/	90.00	90.00	90.00	88.00
/eɯ/	74.20	74.20	88.15	90.23
/Θr/	0.00	12.00	40.00	44.00
/ri/	41.67	50.00	75.00	41.67
Average :	59.95	68.02	73.54	72.07

Table 5.23 Results for 'Segment Length Threshold' = 28
Gray_encoding & 8-tuple Random_mapping

Trainings :	5	10	15	20
Class		Accuracy (%)		
/b/	79.17	91.67	91.67	87.50
/ʧ/	28.75	57.75	67.92	59.58
/g/	45.83	87.50	83.33	75.00
/ŋ/	32.87	35.90	37.50	38.40
/n/	98.67	98.64	98.67	98.64
/k/	73.00	89.58	91.67	93.75
/ω/	55.77	73.35	71.60	76.86
/l/	84.00	80.00	88.00	92.00
/ə/	59.87	63.26	73.39	80.26
/r/	48.00	48.00	52.00	48.00
/s/	56.00	88.00	100.00	96.00
/ʃ/	78.21	68.61	62.91	54.38
/t/	84.40	84.40	82.90	82.38
/w/	44.29	53.29	60.38	63.50
/ʃ/	100.00	100.00	100.00	100.00
/ɔ/	76.00	84.00	84.00	84.00
/æ/	8.70	8.70	13.04	13.04
/Θ/	4.17	8.33	25.00	25.00
/d/	90.00	92.00	90.00	92.00
/eɯ/	66.51	68.27	78.37	86.38
/Θr/	0.00	12.00	40.00	44.00
/ri/	33.33	41.67	66.67	33.33
Average :	56.71	65.22	70.86	69.27

Table 5.24 **Word Recognition for Gray_encoding & 8-tuple Random_mapping**

'Segment Length Threshold' = 24

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	20.0	29.2	32.0	54.2
Run	60.0	60.0	60.0	60.0
Want	12.0	24.0	28.0	68.0
Begun	37.5	41.7	41.7	41.7
Wonder	0.0	0.0	0.0	0.0
Rudder	8.0	8.0	8.0	8.0
Win	0.0	0.0	0.0	0.0
Two	60.0	68.0	80.0	88.0
Shoe	84.0	88.0	88.0	88.0
Toot	4.0	48.0	40.0	52.0
Tattoo	8.0	8.0	4.0	4.0
Toothache	0.0	0.0	0.0	0.0
Cooler	40.0	56.0	64.0	72.0
Tee	72.0	64.0	60.0	52.0
Three	0.0	4.0	16.0	4.0
See	32.0	56.0	60.0	56.0
Average :	27.34	34.68	36.36	40.49

Table 5.25 shows the confusion matrix for the speech recogniser with 'segment length threshold' of 24, and 20 trainings (Table 5.21).

The /ʊ/ segment is frequently confused with /u/. This is because when the sustained vowel /u/ is split into two segemnts by the segmentation process, one of the two segments is of short duration. It is the short /u/ segment that the /ʊ/ segment is confused with (and vice-versa).

Generally, the confusion is with phoneme-like segments that are similar. The /Θ/ segment is confused with the /t/ segment. The /t/ segment has more training samples than the /Θ/ segment hence the stronger response of the /t/ discriminator. A significant number of /t/ samples are misclassified as /b/. These are /t/ segments from the beginning of the word 'Tattoo'. Normally, the /t/ segment has a duration of greater than 24 frames (the 'segment length threshold') thus the Discriminator Bank 'L' would be used to recognise it. The /t/ segments from the beginning of 'Tattoo' have a shorter duration than the 'segment length threshold' and hence the speech recogniser attempts to classify it using Discriminator Bank 'S'. In this discriminator bank, the discriminator assigned to /b/ has most probably had more training than that assigned to /t/ (since the duration of /t/ is normally greater than the 'segment length threshold' thus it would mostly be trained into the /t/ discriminator in Discriminator Bank 'L'), therefore the /b/ discriminator gives a stronger response to the spectral patterns from /t/ (/b/ and /t/ would have similar spectral patterns since they are both 'stops'; voiced and unvoiced respectively).

The 'Sliding Window' technique is better at discriminating the segments /Θr/ and /ri/ in 'Three' than the 'Separate Segmentation and Labelling' approach.

	b	ɪ	g	Λ	n	k	u	l	ə	r	s	i	t	w	ʃ	ɔ	æ	θ	d	eɪ	θr	ri	Samples
b	22	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	.	24
ɪ	3	26	4	.	.	2	14	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	49
g	.	.	20	.	.	.	.	.	.	.	.	.	1	.	.	.	.	1	.	2	.	.	24
Λ	.	.	.	51	.	.	.	.	21	2	.	.	4	.	.	23	.	.	22	.	.	.	123
n	.	.	.	.	142	.	2	.	3	.	.	1	.	.	.	.	.	.	.	.	.	.	148
k	1	.	.	.	.	47	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	49
u	9	10	2	.	4	1	139	2	.	.	.	7	.	2	.	.	.	.	3	.	.	2	181
l	.	.	1	.	.	.	2	21	.	.	.	.	.	.	.	.	.	.	1	.	.	.	25
ə	.	.	1	.	1	.	.	.	106	.	.	.	.	.	.	.	.	.	3	.	.	.	111
r	.	.	.	.	.	.	.	.	7	36	.	.	.	.	.	.	.	.	7	.	.	.	50
s	.	.	.	.	.	.	.	.	.	.	24	.	1	.	.	.	.	.	.	.	.	.	25
i	.	2	.	.	2	1	29	5	.	.	.	50	.	.	.	.	.	.	3	.	.	4	96
t	14	.	.	.	.	2	2	.	.	.	.	1	171	.	.	.	.	.	6	.	3	.	199
w	.	.	.	.	3	.	2	.	21	3	.	.	.	63	.	.	.	.	7	.	.	.	99
ʃ	.	.	.	.	.	.	.	.	.	.	.	.	.	.	25	.	.	.	.	.	.	.	25
ɔ	.	.	.	.	.	.	.	.	3	.	.	.	.	.	.	22	.	.	.	.	.	.	25
æ	3	1	3	.	.	.	2	.	.	.	.	.	3	1	.	.	4	.	6	.	.	.	23
θ	.	.	.	.	.	1	.	.	.	.	.	.	15	.	.	.	.	8	.	.	.	.	24
d	.	.	.	.	.	.	1	.	.	.	.	.	7	.	.	.	.	.	42	.	.	.	50
eɪ	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	2	34	.	.	37
θr	1	.	.	.	.	1	.	.	.	.	.	.	12	.	.	.	.	.	.	.	11	.	25
ri	.	.	.	.	.	.	6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	6	12

Table 5.25 Confusion matrix for recogniser with Gray encoding / 8-tuple Random mapping and 15 trainings The `segment length threshold' = 24

5.7 Summary

A design for a WISARD n-tuple isolated word speech recogniser based on the 'Sliding Window' approach has been presented in this chapter. A 16-channel filterbank IC enables spectral data to be obtained from the utterance input to the speech recogniser. Using the segmentation method suggested in Chapter 3, the spectral data is partitioned into phoneme-like segments. After spectral normalisation and noise subtraction normalisation, the phoneme-like segment is passed on to the data reduction stage where the 8-bit spectral data from the filter bank is encoded to 4-bits using the Linear, and Gray encoders. To train or classify the phoneme-like segment, a window of a fixed duration (less than the duration of the phoneme-like segment) is aligned with the start of the segment and slid one spectral frame at a time across to the end of the phoneme-like segment. As the window is slid across the phoneme-like segment, the spectral data windowed is trained or classified. There are two banks of discriminators in this speech recogniser (Discriminator Bank 'S' and Discriminator Bank 'L'). By comparing the duration of the phoneme-like segment to be trained or classified, with a 'segment length threshold', one of these discriminator banks would be selected. If the number of spectral frames in a phoneme-like segment exceeds the 'segment length threshold', it is considered to be a long phoneme-like segment and Discriminator Bank 'L' is selected for training or classification, otherwise it is a short phoneme-like segment and Discriminator Bank 'S' is selected. Two window sizes (12.8ms and 48 ms) are used in this design. The 12.8 ms window is used with Discriminator Bank 'S' while the 48 ms window is used with Discriminator Bank 'L'.

Various configurations are possible for the speech recogniser design. A series of experiments with different configurations are conducted to arrive at the setup for the speech recogniser that gives the best recognition performance. The best recognition accuracy for the phoneme-like segments in the word-set used in these experiments is 73.97%. The best word recognition accuracy achieved is 40.49%. By applying the output of this speech recogniser to a linguistic processor, the word recognition accuracy of this speech recogniser may be improved (Chapter 6).

CHAPTER SIX

LINGUISTIC PROCESSOR FOR ERROR CORRECTION

6.1 Introduction

The isolated word speech recogniser designs discussed in Chapters 4 and 5 transform the incoming acoustic signal into a string of labels from a set of 22 class labels, 20 of which are phonemes. Due to errors by the speech recognisers in the labelling of the phoneme-like segments, some of the class labels in the string output by the speech recogniser will be incorrect. The output string may be considered as a noisy representation of the acoustic signal at the recogniser's input and a string dissimilarity measurement such as the Levenstein distance [83] can be used to match this output string with an entry in the system's lexicon.

There are three types of errors that can occur in the recognition process; insertion and deletion errors by the segmentation process, and substitution errors as a result of misclassification of a phoneme-like segment by the recogniser. These errors at the phoneme level lower the word recognition rate of the recogniser. It is possible to improve the word recognition rate by including in the design of the speech recogniser a mechanism for correcting the errors in the label string. For example, the word recognition system of Makino & Kido [13] has a phoneme recognition accuracy of 75.9% but by allowing correction of insertions, omissions and substitutions of phonemes, a word recognition accuracy of 92.4% is achieved.

Just as the syntactic rules of a language dictate the meaningful arrangement of words in the construction of sentences, the phonotactic

rules of the language determine which combination of phonemes are valid and which are prohibited in the construction of words. As the output of the speech recognisers presented in Chapters 4 and 5 are strings of class labels (almost all of which are phonemes) representing the input word, it is therefore possible to apply a combination of phonotactic rules and error correction rules as a linguistic processor to the label string output by the recogniser in an attempt to improve the word recognition accuracy. A linguistic processor based on this idea is proposed by Kashyap & Mittal [84].

In this chapter, two designs for linguistic processing of the string of labels from the isolated word speech recognisers of Chapters 4 and 5 are presented. The first design is an implementation of the method of Kashyap & Mittal for error correction in strings. The second design is a modification of the linguistic processor of Kashyap & Mittal. An alternative approach for class label substitutions is suggested.

6.2 String Error Correction Method of Kashyap & Mittal

Given a word-set, W, containing all the words in the speech recogniser's lexicon, and the phonetic transcription of these words, L, the set of class labels (phonemes), P, that the words in the word-set are constructed from, can be determined. A set of linguistic rules that express the concatenation of two class labels for all members in L can then be generated. This procedure is applied to the word-set of Badii & Binstead used to evaluate the speech recogniser designs of Chapters 4 and 5. These speech recognisers recognise the words in the word-set as concatenations of segments labelled from a set of twenty-two classes, twenty of which are phonemes, the remaining two being combinations of two phonemes (/ϴr/ and /ri/) from the segmentation of the word 'Three'.

The set of class labels for the word-set are given by

$$P = \{/b/,/ʉ/,/g/,/ʌ/,/n/,/k/,/u/,/l/,/ə/,/r/,/s/,/i/,$$
$$/t/,/w/,/ʃ/,/ɔ/,/æ/,/ɵ/,/d/,/eʉ/,/ɵr/,/ri/\}$$

The class labels in P may be divided into three groups :

(i) *Vowels* : /ʉ/, /ʌ/, /u/, /ə/, /i/, /ɔ/, /æ/, /eʉ/

(ii) *Consonants* : /b/, /g/, /n/, /k/, /l/, /r/, /s/, /t/, /w/, /ɵ/, /d/, /ʃ/, /ɵr/

(iii) *Vowel/Consonant* : /ri/

The class label /ɵr/ is grouped with the consonants since it consists of the phonemes /ɵ/ and /r/, both of which are consonants. The class label /ri/ may be considered as a consonant or vowel since it consists of the phoneme /r/ which is a consonant and /i/, a vowel. In a string of class labels containing the label /ri/, from the point of view of the class label immediately preceding /ri/, /ri/ is considered a consonant, but for the class label immediately following /ri/, it is a vowel.

The transcription of the word-set in terms of the concatenations of the class labels in P, ie. L, is given in Table 6.1. Alternative transcriptions are given for words containing sustained vowels since the segmentation process used in the speech recogniser designs of Chapters 4 and 5 tended to split some occurrences of sustained vowels into two segments (Section 3.7, Chapter 3). The word 'Three' was also segmented as "/ɵr/ /ri/ /i/" or "/ɵr/ /i/" (Section 3.7, Chapter 3).

Table 6.1 Transcription of the words in the Lexicon, L

Win :	/w/,/ɪ/,/n/
Want :	/w/,/ɔ/,/n/,/t/
One :	/w/,/ʌ/,/n/
Run :	/r/,/ʌ/,/n/
Begun :	/b/,/ɪ/,/g/,/ʌ/,/n/
Cooler :	/k/,/u/,/l/,/ə/
	/k/,/u/,/l/,/ə/,/ə/
Rudder :	/r/,/ʌ/,/d/,/ə/
	/r/,/ʌ/,/d/,/ə/,/ə/
Wonder :	/w/,/ʌ/,/n/,/d/,/ə/
	/w/,/ʌ/,/n/,/d/,/ə/,/ə/
Two :	/t/,/u/
	/t/,/u/,/u/
Shoe :	/ʃ/,/u/
	/ʃ/,/u/,/u/
Tattoo :	/t/,/æ/,/t/,/u/
	/t/,/æ/,/t/,/u/,/u/
Toot :	/t/,/u/,/t/
	/t/,/u/,/u/,/t/
Toothache :	/t/,/u/,/ϴ/,/eɪ/,/k/
	/t/,/u/,/ϴ/,/eɪ/,/eɪ/,/k/
Tee :	/t/,/i/
	/t/,/i/,/i/
See :	/s/,/i/
	/s/,/i/,/i/
Three :	/ϴr/,/ri/,/i/
	/ϴr/,/ri/,/i/,/i/
	/ϴr/,/i/
	/ϴr/,/i/,/i/

6.2.1 Linguistic Rules

The linguistic rules expressing the list of valid concatenation of class labels for the word-set of Badii & Binstead have been devised. They are as follows :

(i) *Vowel/Vowel* : /i/ /i/, /u/ /u/, /eɪ/ /eɪ/, /ə/ /ə/, /ri/ /i/

(ii) *Consonant/Consonant* : /n/ /t/, /n/ /d/, /ϴr/ /ri/

(iii) *Consonant/Vowel* : c_1 /ɪ/, c_2 /u/, c_3 /ʌ/, c_4 /ə/,
c_5 /i/, c_6 /ɔ/, c_7 /æ/, c_8 /eɪ/,

where
$c_1 \in C_1$, C_1 = [/b/, /w/]
$c_2 \in C_2$, C_2 = [/k/, /t/, /ʃ/]
$c_3 \in C_3$, C_3 = [/g/, /r/, /w/]
$c_4 \in C_4$, C_4 = [/l/, /d/]
$c_5 \in C_5$, C_5 = [/s/, /t/, /ϴr/]
$c_6 \in C_6$, C_6 = [/w/]
$c_7 \in C_7$, C_7 = [/t/]
$c_8 \in C_8$, C_8 = [/ϴ/]

(iv) *Vowel/Consonant* : /ɪ/ c_9, /u/ c_{10}, /ʌ/ c_{11}, /ɔ/ c_{12},
/æ/ c_{13}, /eɪ/ c_{14}

where
$c_9 \in C_9$, C_9 = [/g/, /n/]
$c_{10} \in C_{10}$, C_{10} = [/l/, /t/, /ϴ/]
$c_{11} \in C_{11}$, C_{11} = [/n/, /d/]
$c_{12} \in C_{12}$, C_{12} = [/n/]
$c_{13} \in C_{13}$, C_{13} = [/t/]
$c_{14} \in C_{14}$, C_{14} = [/k/]

6.2.2 String Distance Measure

Let $z = b_1,b_2,....,b_n$, $b_i \in P \forall i$ represent the class label string output by the recogniser. If the recognition process was error-free, then z will match exactly one of the entries in L, hence the word in W that caused z is easily identified. Since the speech recogniser is prone to error, z will in general not belong to L. The method is to substitute the class labels b_i in z by other class labels and hence generate strings that match an entry in L. A distance measure is used to select one string from the strings generated by the substitution process as the word in the word-set, W, that caused z. The distance measure used in this string error corrector is based on the probability of the error in the classification of the phoneme-like segment by the speech recogniser.

For each class label (or phoneme) b_i observed in z, there is a possibility that it is actually the correct phoneme, or it may have been substituted for some other phoneme. Therefore to perform correction, this class label in z must be replaced by the actual ie. correct class label. The possible substitutions for each class label in z are determined from the recognition performance of the speech recogniser. From the performance statistics, weights, $w(a,b)$, are assigned to each substitution as a measure of the likelihood of such a substitution for that class label. The smaller the value of this weight, the more likely it is that this substitution occurred during the recognition process. The weight $w(a,b)$ indicates the chance that when some class label a is observed, the actual class label is some other class label b.

Therefore, for each class label in P, a vector B_i is defined. The elements of B_i are the possible substitutions for that class label together with the weight for that substitution. Also included in this vector is the deletion symbol λ. If during the error correction process a symbol is to be deleted from z, it is replaced by λ.

To illustrate how this error correction process works, consider a string $z = b_1b_2b_3$.

(i) Replace b_i in z by vector B_i ie. $z = B_1B_2B_3$

Thus b_i --> B_i = $\begin{bmatrix} b_{i1}(w_{i1}) \\ . \\ . \\ b_{in}(w_{in}) \end{bmatrix}$, $w_{ik} \triangleq w(b_i,b_{ik})$

where $b_{i1} = b_i$, and $b_{in} = \lambda$

b_{ik} is the phoneme or class label to be substituted for b_i, and w_{ik} is the weight assigned to that substitution.

(ii) Combine vectors B_1 and B_2 to form a single vector B_{12}. The components of B_{12} are determined as follows :

Let $B_1 = \begin{bmatrix} b_{11}(w_{11}) \\ . \\ . \\ b_{1p}(w_{1p}) \end{bmatrix}$ $B_2 = \begin{bmatrix} b_{21}(w_{21}) \\ . \\ . \\ b_{2q}(w_{2q}) \end{bmatrix}$, $b_{1p} = b_{2q} = \lambda$

Form strings by combining each element $b_{1j}(w_{1j})$ in B_1 with all elements in B_2 as given below :

$$b_{1j}b_{2k}(w_{1j} + w_{2k}), \qquad j = 1,2....,p; \quad k = 1,2....,q$$

Thus the new vector B_{12} has elements consisting of the string $b_{1j}b_{2k}$ with weight $(w_{1j} + w_{2k})$.

(iii) From all elements $b_{1j}b_{2k}(w_{1j} + w_{2k})$ in B_{12}, delete b_{2k} if $b_{2k} = b_{1j}$ so that no two adjacent symbols in the string are identical.

(iv) Apply the linguistic rules (Section 6.2.1) to the elements in B_{12} and delete those elements, $b_{1j}b_{2k}(w_{1j} + w_{2k})$, that are not allowed by the linguistic rules.

(v) For strings $b_{1j}b_{2k}$ from the elements in B_{12} that are identical, delete the elements with the larger weight.

(vi) Combine B_{12} with B_3 to form the vector B_{123} by repeating steps (ii) to (v) but with B_{12} in place of B_1 and B_3 in place of B_2.

(vii) For all elements in B_{123}, from all strings containing the symbol λ, delete λ; eg. if an element of B_{123} contains the string "$s\lambda i$", the string becomes "si" after deleting λ.

(viii) The string from the element in B_{123} with the lowest weight that matches an entry in L is taken to be the word that caused the speech recogniser to output the string z.

Thus if x represents an element in B_{123}, then the distance between z and x, $D(z,x)$, is the sum of the weights of the various class labels in x.

In the following two sections, the string error correction method illustrated above is applied to the output from the speech recognisers based on the 'Separate Segmentation and Labelling' (SS&L) approach (Chapter 4), and the 'Sliding Window' approach (Chapter 5), in order to improve their word recognition accuracy.

6.2.3 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the SS&L Approach

In order to apply this linguistic processor to the string of class labels output by the speech recogniser, the weights for class label substitutions must first be determined for this particular speech recogniser. The weights for the class label substitutions were obtained from the statistics of the recognition performance of the speech recogniser for the samples from the word-set of Badii & Binstead. Kashyap & Mittal do not show in their paper how they worked out the weights. For the work presented in this chapter, the following method was used to determine the weights :

$w(a,b)$ denotes the weight assigned to the replacement of the class label a in z by another class label b, and is defined as

$$w(a,b) = 1 - \mathrm{p}(a,b) \qquad (6.1)$$

$\mathrm{p}(a,b)$ is the probability that the class label a is observed at the ouput of the speech recogniser but the actual (ie. correct) class label is b (ie. the speech recogniser has misclassified some class b as some other class a). If n_b is the total number of samples of class b input to the speech recogniser, and $n_{a,b}$ is the number of times that the speech recogniser misclassifies class b as class a, then $\mathrm{p}(a,b)$ is given by

$$\mathrm{p}(a,b) = n_{a,b} / n_b \qquad (6.2)$$

Following Kashyap & Mittal, $w(a,b) = 0$ if $b = a$, and

$w(a,b) = 1$ if $b = \lambda$

Table 6.2 Class label substitutions and associated weights

/b/ --> b (0)
æ (0.9)
ɪ (0.99)
u (0.99)
λ (1)

/Λ/ --> Λ (0)
λ (1)

/n/ --> n (0)
i (0.92)
eɪ (0.95)
u (0.96)
d (0.96)
ə (0.99)
w (0.99)
λ (1)

/s/ --> s (0)
λ (1)

/w/ --> w (0)
r (0.99)
λ (1)

/ʃ/ --> ʃ (0)
λ (1)

/ri/ --> ri (0)
i (0.99)
λ (1)

/ɪ/ --> ɪ (0)
æ (0.9)
u (0.91)
l (0.96)
i (0.96)
ri (0.96)
λ (1)

/k/ --> k (0)
æ (0.96)
b (0.97)
θr (0.98)
t (0.99)
λ (1)

/ə/ --> ə (0)
Ɔ (0.93)
Λ (0.99)
æ (0.99)
w (0.99)
λ (1)

/eɪ/ --> eɪ (0)
λ (1)

/θ/ --> θ (0)
g (0.99)
λ (1)

/Ɔ/ --> Ɔ (0)
Λ (0.88)
λ (1)

/g/ --> g (0)
æ (0.94)
ri (0.96)
θ (0.96)
k (0.98)
l (0.99)
r (0.99)
i (0.99)
u (0.99)
ə (0.99)
λ (1)

/l/ --> l (0)
λ (1)

/r/ --> r (0)
w (0.98)
λ (1)

/æ/ --> æ (0)
Λ (0.99)
ə (0.99)
r (0.99)
eɪ (0.99)
Ɔ (0.99)
λ (1)

/θr/ --> θr (0)
λ (1)

/i/ --> i (0)
u (0.96)
ɪ (0.99)
eɪ (0.99)
λ (1)

/u/ --> u (0)
ri (0.40)
i (0.59)
æ (0.75)
l (0.77)
w (0.86)
eɪ (0.87)
ɪ (0.94)
n (0.98)
t (0.98)
r (0.99)
k (0.99)
λ (1)

/t/ --> t (0)
θr (0.33)
θ (0.78)
s (0.85)
æ (0.87)
k (0.94)
Λ (0.97)
ʃ (0.97)
g (0.98)
ɪ (0.98)
d (0.98)
b (0.99)
λ (1)

/d/ --> d (0)
ri (0.84)
æ (0.93)
r (0.98)
w (0.98)
g (0.99)
θ (0.99)
eɪ (0.99)
ɪ (0.99)
λ (1)

The choices for class label replacements and the associated weights for the speech recogniser based on the 'Separate Segmentation and Labelling' (SS&L) approach are given in Table 6.2. Appendix B gives the data from which the weights were calculated. The configuration of the speech recogniser is as follows :

(i) "Averaged points interpolation method" for time normalisation.
(ii) Gray encoder for the Data Reduction stage.
(iii) Across channel mapping of spectral frame store to 'retina'.
(iv) n-tuple size of 8.
(v) Random mapping of 'retina' to n-tuples.

The speech recogniser with this configuration was selected for experimentation with the linguistic processor because experiments in Chapter 4 show that it offered the best results. The word recognition accuracy of this speech recogniser is given in Table 4.30 (Chapter 4). The highest word recognition accuracy is 57.2% achieved with 15 trainings although 78.36% recognition accuracy was achieved at the phoneme-like segment level (Chapter 4, Table 4.28). Table 6.3 gives the word recognition accuracy of the same speech recogniser after application of the linguistic processor to the speech recogniser's output. The word recognition accuracy improves with training, peaking at 93.11% for 15 trainings. Further training causes performance to drop. With the exception of the word 'Three', all the other words in the word-set are recognised with atleast 80% accuracy. These results are comparable with those obtained by Badii & Binstead for recognition of the word-set by the 'whole word' recognition approach.

Although the word recognition accuracy of the speech recogniser designs proposed in this work are similar, the advantage of the proposed speech recognisers over that of Badii & Binstead is that

words other than those in the Badii & Binstead word-set, that are combinations of the phoneme-like segments recognised by these speech recognisers may also be recognised by simply including them in the linguistic processor. In comparison, the Badii & Binstead speech recogniser would require an extra discriminator for each new word added to the recogniser's lexicon. Also, the total memory requirement for all class discriminators in the proposed recogniser designs is below 1 Mbit, whereas the Badii & Binstead recogniser requirement is more than 3 Mbits.

Table 6.3 **Word Recognition for Recogniser based on the 'Separate Segmentation and Labelling' Approach**

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	100.0	100.0	100.0	100.0
Run	100.0	100.0	100.0	100.0
Want	100.0	100.0	100.0	100.0
Begun	95.8	95.8	95.8	100.0
Wonder	92.0	96.0	96.0	100.0
Rudder	88.0	96.0	100.0	100.0
Win	100.0	96.0	96.0	96.0
Two	100.0	100.0	96.0	96.0
Shoe	96.0	96.0	100.0	96.0
Toot	100.0	100.0	100.0	88.0
Tattoo	76.0	88.0	88.0	88.0
Toothache	83.3	95.8	100.0	100.0
Cooler	96.0	100.0	100.0	92.0
Tee	96.0	92.0	80.0	84.0
Three	28.0	36.0	50.0	44.0
See	88.0	88.0	88.0	84.0
Average :	89.94	92.48	93.11	91.75

6.2.4 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the 'Sliding Window' Approach

The choices for class label replacements and the associated weights for the speech recogniser based on the 'Sliding Window' approach are given in Table 6.4. The weights were calculated using the formula defined in (6.1) in Section 6.2.3. Appendix B gives the data from which the weights were calculated. The configuration of the speech recogniser is as follows :

(i) Gray encoder for the Data Reduction stage.
(ii) Direct mapping of spectral frame store to 'retina'.
(iii) n-tuple size of 8.
(iv) Random mapping of 'retina' to n-tuples.
(v) Segment Length Threshold of 24 frames.

The speech recogniser with this configuration was selected for experimentation with the linguistic processor because experiments in Chapter 5 show that it offered the best results. The best word recognition accuracy of this speech recogniser for the word-set of Badii & Binstead is 40.49% achieved with 20 trainings (Table 5.24, Chapter 5). Table 6.5 gives the word recognition accuracy of the same speech recogniser after application of the linguistic processor to the speech recogniser's output. The word recognition accuracy improves with training, peaking at 93.49% for 10 trainings. Further training causes performance to drop. This speech recogniser has improved the recognition accuracy for the word 'Three' (80% with 15 trainings) which was difficult to recognise with the SS&L based speech recogniser.

Table 6.4 Class label substitutions and associated weights

/b/ --> b (0)
æ (0.81)
ɪ (0.92)
t (0.94)
u (0.94)
θr (0.96)
k (0.98)
λ (1)

/ʌ/ --> ʌ (0)
λ (1)

/n/ --> n (0)
w (0.93)
u (0.97)
i (0.98)
r (0.98)
ə (0.99)
λ (1)

/s/ --> s (0)
λ (1)

/w/ --> w (0)
λ (1)

/ʃ/ --> ʃ (0)
λ (1)

/ri/ --> ri (0)
i (0.96)
l (0.97)
u (0.97)
ɪ (0.99)
λ (1)

/ɪ/ --> ɪ (0)
u (0.95)
æ (0.95)
i (0.96)
g (0.98)
ri (0.98)
λ (1)

/k/ --> k (0)
θr (0.93)
t (0.99)
i (0.99)
θ (0.99)
λ (1)

/ə/ --> ə (0)
ʌ (0.82)
w (0.85)
ɔ (0.86)
r (0.89)
eɪ (0.99)
k (0.99)
λ (1)

/eɪ/ --> eɪ (0)
æ (0.95)
g (0.97)
l (0.97)
ɪ (0.99)
ə (0.99)
u (0.99)
λ (1)

/θ/ --> θ (0)
g (0.99)
λ (1)

/g/ --> g (0)
θ (0.97)
d (0.99)
λ (1)

/l/ --> l (0)
i (0.94)
u (0.99)
λ (1)

/r/ --> r (0)
w (0.98)
ʌ (0.99)
λ (1)

/æ/ --> æ (0)
eɪ (0.99)
λ (1)

/θr/ --> θr (0)
t (0.99)
λ (1)

/i/ --> i (0)
u (0.92)
eɪ (0.96)
ɪ (0.98)
g (0.99)
l (0.99)
n (0.99)
æ (0.99)
λ (1)

/ɔ/ --> ɔ (0)
ʌ (0.82)
λ (1)

/u/ --> u (0)
ri (0.67)
ɪ (0.71)
i (0.83)
æ (0.90)
l (0.95)
w (0.96)
d (0.98)
eɪ (0.99)
t (0.99)
n (0.99)
λ (1)

/t/ --> t (0)
θ (0.33)
θr (0.44)
d (0.86)
g (0.87)
æ (0.87)
k (0.92)
ri (0.96)
ɪ (0.98)
ʌ (0.99)
λ (1)

/d/ --> d (0)
æ (0.65)
w (0.77)
ʌ (0.79)
l (0.85)
r (0.86)
ə (0.88)
eɪ (0.89)
b (0.92)
ɪ (0.92)
t (0.96)
g (0.97)
u (0.97)
i (0.97)
ɔ (0.98)
λ (1)

Table 6.5 **Word Recognition for Recogniser based on the 'Sliding Window' Approach**

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	100.0	100.0	100.0	95.8
Run	92.0	80.0	80.0	80.0
Want	92.0	100.0	100.0	100.0
Begun	91.7	100.0	100.0	91.7
Wonder	100.0	100.0	100.0	100.0
Rudder	100.0	100.0	100.0	100.0
Win	96.0	100.0	88.0	92.0
Two	80.0	92.0	88.0	96.0
Shoe	96.0	100.0	100.0	100.0
Toot	100.0	100.0	100.0	100.0
Tattoo	80.0	84.0	76.0	80.0
Toothache	58.3	95.8	95.8	91.7
Cooler	96.0	100.0	100.0	100.0
Tee	96.0	88.0	80.0	72.0
Three	36.0	64.0	80.0	60.0
See	60.0	92.0	100.0	96.0
Average :	85.88	93.49	92.99	90.95

6.3 An Alternative Linguistic Processor

A disadvantage of the linguistic processor design due to Kashyap & Mittal is that the class labels replacement weight is calculated from the recognition performance statistics of the speech recogniser and is therefore dependent on the particular speech recogniser the linguistic processor is being used with. This weight calculation is a laborious process since acquiring satisfactory performance statistics requires that a large number of samples be considered in the measurements. A modification to the linguistic processor of Kashyap & Mittal is suggested which overcomes this problem and makes it portable (ie. independent of the speech recogniser it is being used with).

The Kashyap & Mittal linguistic processor assumes that when a word is input to the speech recogniser, the speech recogniser outputs a single class label corresponding to each phoneme-like segment in that word. *A priori* information regarding class label substitution is therefore required to perform correction of errors due to misclassification of one or more phoneme-like segments by the recogniser. However, this information may be obtained from the speech recogniser itself if instead of making a single decision as to the identity of a phoneme-like segment, the recogniser outputs alternative choices in decreasing order of likelihood ie. a lattice of class labels. Each column of the lattice gives the choice of alternative class labels for the corresponding phoneme-like segment in the word input to the recogniser (there are as many columns in the lattice as there are phoneme-like segments in the word). The 'lattice depth' determines the number of class labels in the lattice column. Thus for example, if the 'lattice depth' is set to 5, the recogniser outputs a choice of 5 class labels for each phoneme-like segment in the input word. The maximum lattice depth is 22 since there are a total of 22 classes in the speech recogniser.

The linguistic processor suggested in this section is based on the same concept as the Kashyap & Mittal linguistic processor, but it differs in the manner in which the string distance is obtained.

To illustrate how this linguistic processor functions, let z be the output of the speech recogniser for a word consisting of three phoneme-like segments. Then $z = S_1S_2S_3$ where

$$S_i = \begin{bmatrix} s_{i0}(w_{i0}) \\ . \\ . \\ s_{in}(w_{in}) \end{bmatrix} \qquad , w_{ik} = k \qquad k = 0,1,....,n$$
$$, s_{in} = \lambda$$

z is therefore a lattice with lattice depth n, and S_i is the column in the lattice corresponding to the ith phoneme-like segment. s_{ik} is the class label that the recogniser assigns to that phoneme-like segment. s_{i0} is the label of the class discriminator in the speech recogniser that gave the highest recognition score for that phoneme-like segment. s_{i1} is the label of the class discriminator with the next highest score, and so on. λ is the segment deletion symbol. The weight w_{ik} is assigned to the class label s_{ik} and is numerically equal to its position in the lattice column. The deletion symbol λ , has the highest weight (as in the Kashyap & Mittal linguistic processor). Treating S_i as a vector, where $s_{ik}(w_{ik})$ are the components of S_i,

(i) Combine lattice columns (or vectors) S_1 and S_2 to form a single vector S_{12}. Each component of S_{12} consists of a string with associated weight, formed by combining each element $s_{1j}(w_{ij})$ in S_1 with all elements in S_2 as given below :

$$s_{1j}s_{2k}(w_{1j} + w_{2k}), \qquad j = 1,2....,n; \quad k = 1,2....,n$$

(ii) From all elements $s_{1j}s_{2k}(w_{1j} + w_{2k})$ in S_{12}, delete s_{2k} if $s_{2k} = s_{1j}$ so that no two adjacent symbols in the string $s_{1j}s_{2k}$ are identical.

(iii) Apply the linguistic rules (Section 6.2.1) to the elements in S_{12} and delete those elements, $s_{1j}s_{2k}(w_{1j} + w_{2k})$, that are not allowed by the linguistic rules.

(iv) For strings $s_{1j}s_{2k}$ from the elements in S_{12} that are identical, delete the elements with the larger weight.

(v) Combine S_{12} with S_3 to form a new vector S_{123} by repeating steps (i) to (iv) but with S_{12} in place of S_1 and S_3 in place of S_2.

(vi) For all elements in S_{123}, from all strings containing the symbol λ, delete λ.

(vii) The string from the elements in S_{123} with the lowest weight that matches an entry in the systems lexicon, L, is taken to be the identity of the word input to the speech recogniser.

This linguistic processor was applied to the output from the speech recognisers based on the 'Separate Segmentation and Labelling' and 'Sliding Window' approaches. The speech recognisers are the same as those used in the experiments in Sections 6.2.3 and 6.2.4 respectively. The following two sections give the results for these experiments. Results are given for the linguistic processor with lattice depths of 5, 10, 15 and 20 to investigate how word recognition accuracy improves with increasing lattice depth.

6.3.1 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the SS&L Approach

Tables 6.6 and 6.7 give the word recognition results. The results are given for lattice depths of 5 and 10 only. The nature of the errors showed that increasing lattice depth further will not improve the results significantly. The idea of outputting the recognition results as a phonetic lattice is that in the case that the speech recogniser misclassifies a phoneme-like segment, the correct class will at least be included in the lattice and hence offering the chance of recovering from this error. Increasing the lattice depth increases the chance of finding the correct class within the lattice, but this also increases the branching factor ie. the number of alternative words that will be generated by the error correction process. If a phoneme-like segment is misclassified such that the correct class is quite low down the lattice, hence increasing the overall weight of the correct word, this may result in one or more of the alternative words generated by the linguistic processor having a lower weight than the correct word causing the linguistic processor to err. In such cases, increasing the lattice depth will not improve the word recognition accuracy. From tables 6.6 and 6.7 it can be seen that this has been the case for the words 'Shoe', 'Tattoo', 'Tee', 'Three' and 'See'. Other words like 'Wonder' and 'Rudder' have benefitted from the increase in lattice depth.

Best results are obtained for 5 trainings. Increasing training causes an initial drop in word recognition accuracy which improves with further training but does not improve upon that for 5 trainings. Increasing the lattice depth from 5 to 10 has improved that word recognition accuracy from 85.99% to 87%.

Table 6.6 Word Recognition for Recogniser based on the 'Separate Segmentation and Labelling' Approach

Lattice Depth = 5

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	100.0	100.0	100.0	100.0
Run	100.0	100.0	100.0	100.0
Want	100.0	100.0	100.0	100.0
Begun	100.0	100.0	100.0	100.0
Wonder	96.0	96.0	100.0	100.0
Rudder	96.0	96.0	96.0	88.0
Win	96.0	96.0	88.0	92.0
Two	100.0	100.0	100.0	100.0
Shoe	96.0	96.0	100.0	100.0
Toot	100.0	100.0	100.0	100.0
Tattoo	44.0	40.0	40.0	48.0
Toothache	95.8	95.8	100.0	100.0
Cooler	100.0	100.0	96.0	96.0
Tee	80.0	68.0	80.0	80.0
Three	16.0	4.0	24.0	24.0
See	56.0	36.0	40.0	44.0
Average :	85.99	82.99	85.25	85.75

Table 6.7 Word Recognition for Recogniser based on the 'Separate Segmentation and Labelling' Approach

Lattice Depth = 10

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	100.0	100.0	100.0	100.0
Run	100.0	100.0	100.0	100.0
Want	100.0	100.0	100.0	100.0
Begun	100.0	100.0	100.0	100.0
Wonder	100.0	100.0	100.0	100.0
Rudder	100.0	100.0	100.0	100.0
Win	100.0	100.0	96.0	96.0
Two	100.0	100.0	100.0	100.0
Shoe	96.0	96.0	100.0	100.0
Toot	100.0	100.0	100.0	100.0
Tattoo	44.0	40.0	40.0	48.0
Toothache	100.0	100.0	100.0	100.0
Cooler	100.0	100.0	100.0	100.0
Tee	80.0	68.0	80.0	80.0
Three	16.0	4.0	24.0	24.0
See	56.0	36.0	40.0	44.0
Average :	87.00	84.00	86.25	87.00

6.3.2 Application of the Linguistic Processor to the Output from Speech Recogniser Based on the 'Sliding Window' Approach

Tables 6.8 - 6.11 give the results for increasing lattice depth from 5 to 20 in steps of 5. With a lattice depth of 5, the word recognition accuracy is seen to improve with training until 10 trainings. Increasing training initially lowers the word recognition accuracy (15 trainings) which then improves with further training. 65.51% word recognition accuracy with 20 trainings is the best performance for this lattice depth. This compares with 85.99% achieved with 5 trainings for the SS&L based speech recogniser (Table 6.6).

Word recognition accuracy improves with increasing lattice depth. Whereas for the results obtained with lattice depth of 5 showed the average word recognition accuracy to dip at 15 trainings, the results obtained with increasing lattice depth show that the performance peaks at 15 trainings. The highest word recognition accuracy is 87.48% obtained for lattice depth of 20.

For lattice depths greater than 10, it is seen that no further improvements were achieved for the recognition of words as 'See', 'Three', 'Tee', 'Two', 'Shoe', 'Win' and 'One'. This is because in the case of errors in the recognition of the phoneme-like segments in these words, the class labels of these segments were too low down the lattice enabling other words to have a lower weight hence it was not possible to correct this error. In the case of 'Wonder' and 'Rudder', it was possible to achieve improvements with a lattice depth of 20 since these are long words.

Table 6.8 **Word Recognition for Recogniser based on the 'Sliding Window' Approach**

Lattice Depth = 5

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	79.2	91.7	91.7	91.7
Run	80.0	80.0	80.0	80.0
Want	32.0	96.0	100.0	100.0
Begun	100.0	95.8	100.0	91.7
Wonder	12.0	12.0	4.0	16.0
Rudder	8.0	12.0	8.0	12.0
Win	16.0	48.0	8.0	60.0
Two	88.0	92.0	88.0	96.0
Shoe	96.0	100.0	100.0	92.0
Toot	92.0	92.0	92.0	88.0
Tattoo	16.0	16.0	20.0	20.0
Toothache	12.5	20.8	12.5	16.7
Cooler	84.0	80.0	84.0	88.0
Tee	80.0	80.0	80.0	76.0
Three	24.0	44.0	56.0	44.0
See	40.0	84.0	84.0	76.0
Average :	53.73	65.27	63.01	65.51

Table 6.9 **Word Recognition for Recogniser based on the 'Sliding Window' Approach**

Lattice Depth = 10

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	79.2	91.7	91.7	91.7
Run	80.0	80.0	80.0	80.0
Want	40.0	100.0	100.0	100.0
Begun	100.0	100.0	100.0	100.0
Wonder	12.0	12.0	12.0	16.0
Rudder	12.0	12.0	12.0	12.0
Win	28.0	64.0	84.0	72.0
Two	88.0	92.0	88.0	96.0
Shoe	96.0	100.0	100.0	92.0
Toot	96.0	96.0	96.0	96.0
Tattoo	20.0	20.0	32.0	20.0
Toothache	95.8	83.3	87.5	87.5
Cooler	88.0	84.0	84.0	88.0
Tee	84.0	80.0	80.0	76.0
Three	24.0	44.0	56.0	44.0
See	40.0	84.0	84.0	76.0
Average :	61.44	71.44	74.20	71.70

Table 6.10 **Word Recognition for Recogniser based on the 'Sliding Window' Approach**

Lattice Depth = 15

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	79.2	91.7	91.7	91.7
Run	80.0	80.0	80.0	80.0
Want	40.0	100.0	100.0	100.0
Begun	100.0	100.0	100.0	100.0
Wonder	12.0	12.0	12.0	16.0
Rudder	12.0	16.0	16.0	12.0
Win	28.0	64.0	84.0	72.0
Two	88.0	92.0	88.0	96.0
Shoe	96.0	100.0	100.0	92.0
Toot	96.0	96.0	100.0	100.0
Tattoo	20.0	20.0	88.0	84.0
Toothache	95.8	95.8	100.0	100.0
Cooler	92.0	92.0	92.0	96.0
Tee	84.0	80.0	80.0	76.0
Three	28.0	48.0	56.0	44.0
See	40.0	84.0	84.0	76.0
Average :	61.94	73.22	79.48	77.23

Table 6.11 **Word Recognition for Recogniser based on the 'Sliding Window' Approach**

Lattice Depth = 20

Trainings :	5	10	15	20
Word		**Accuracy (%)**		
One	79.2	91.7	91.7	91.7
Run	84.0	84.0	88.0	84.0
Want	100.0	100.0	100.0	100.0
Begun	100.0	100.0	100.0	100.0
Wonder	96.0	100.0	96.0	96.0
Rudder	64.0	64.0	40.0	32.0
Win	28.0	64.0	84.0	72.0
Two	88.0	92.0	88.0	96.0
Shoe	96.0	100.0	100.0	92.0
Toot	100.0	100.0	100.0	100.0
Tattoo	84.0	80.0	92.0	88.0
Toothache	100.0	100.0	100.0	100.0
Cooler	96.0	100.0	100.0	96.0
Tee	84.0	80.0	80.0	76.0
Three	28.0	48.0	56.0	44.0
See	40.0	84.0	84.0	76.0
Average :	79.20	86.73	87.48	83.98

6.4 Conclusion

The work presented in this chapter shows that the linguistic processor enables the word recognition accuracy of the speech recogniser to be improved significantly, eg. the highest word recognition accuracy of the SS&L based speech recogniser is improved from 57.2% to 93.11% by the Kashyap & Mittal liguistic processor. The linguistic processor is an essential part of the speech recognition system based on the sub-word approach.

Comparing the Kashyap & Mittal linguistic processor with the modified linguistic processor of Section 6.3, the former must be adapted (*a priori* determination of the class label substitution weights) to the particular speech recogniser it is to be used with, whereas this procedure is eliminated in the latter design, hence making it easier to use. The Kashyap & Mittal linguistic processor gives the better results (by about 7%) but the design is not portable. Although the phoneme-like segment recognition accuracy of the speech recogniser affects the performance of both linguistic processors, this affects the modified linguistic processor more than that of Kashyap & Mittal. For example, the 'Sliding Window' based speech recogniser is better at recognising /Θr/, /ri/ and /i/ than the SS&L based speech recogniser. This is reflected in the accuracy of the recognition of 'Three' and 'See' for the two recognisers (Tables 6.2 and 6.1 respectively for the Kashyap & Mittal linguistic processor, and Tables 6.8 and 6.4 respectively for the modified linguistic processor).

6.5 Summary

Although the speech recogniser's recognition accuracy at the phoneme-like segment level may be between 70% and 80% (Chapters 4 and 5), it is possible to achieve higher word recognition accuracy from the speech recogniser by applying a linguistic processor to its output. This chapter demonstrates how the word recognition accuracy (for the word-set of Badii & Binstead) of the SS&L and 'Sliding Window' speech recognisers proposed in Chapters 4 and 5 respectively, may be improved, for example, with a linguistic processor due to Kashyap & Mittal, word recognition accuracies of 93.11% and 93.49% were achieved for the SS&L and 'Sliding Window' based speech recognisers respectively. Prior to application of the linguistic processor, these speech recognisers had word recognition accuracy of 57.2% and 40.49% respectively.

The implementation of the Kashyap & Mittal linguistic processor uses linguistic rules specifying valid concatenations of the class labels (or phonemes) derived from the words from the Badii & Binstead word-set, together with a procedure for substitution of class labels in the string of class labels output by the speech recogniser, to perform error correction.

An alternative linguistic processor is also suggested. This linguistic processor is a modification of the Kashyap & Mittal linguistic processor. The Kashyap & Mittal linguistic processor needs to be adapted to the particular speech recogniser it is to be used with since the class label substitution weights used in the error correction process need to be determined from the recognition performance statistics of that speech recogniser. The modification to the Kashyap & Mittal linguistic processor is to eliminate this requirement hence making it easy to implement with any speech recogniser. Applying the modified linguistic processor to the SS&L and 'Sliding Window' based speech

recognisers, word recognition accuracy of 87% and 87.48% respectively are achieved for the word-set of Badii & Binstead.

The word recognition accuracy of the speech recognisers after error correction by the linguistic processors is comparable to that achieved by Badii & Binstead using the 'whole word' based recognition approach.

CHAPTER SEVEN

CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

7.1 Conclusion

Two designs for isolated word recognition systems based on the sub-word approach have been proposed in this work ('Separate Segmentation & Labelling' (SS&L) recogniser in Chapter 4, and 'Sliding Window' recogniser in Chapter 5). These systems recognise isolated words as concatenations of phoneme-like segments. The basis for both designs is the single layer WISARD net. Earlier work by Badii & Binstead [11], [12], demonstrated the effectiveness of the N-tuple pattern matching technique for isolated word recognition. The proposed designs are aimed at achieving a more flexible system.

The 'whole word' matching approach of the Badii & Binstead recogniser places increasing memory requirements with increase in the size of the lexicon as each additional word requires a separate discriminator. The sub-word approach adopted for the proposed recognisers overcomes this problem since each word is recognised in terms of the phoneme-like segments in the word. Thus other words (besides those in the word-set used to evaluate the recogniser designs) that are combinations from the set of phoneme-like segments recognised by these speech recognisers may also be recognised by simply allowing for them in the linguistic processor section of the recognition system. No additional discriminators are required to accomodate these words. Furthermore, although the word recognition

accuracy is similar to that reported for the Badii & Binstead speech recogniser, this performance has been achieved at a significantly lower memory cost (less than 1 Mbit memory is needed for the proposed designs whereas the Badii & Binstead recogniser requires over 3 Mbit).

At the phoneme-like segment level, recognition accuracy between 70% to 80% was achieved with both recogniser designs. Similar figures have been reported for recognition systems based on other approaches (eg. template matching). This suggests that the N-tuple method using single layer WISARD nets is at least as good as other existing approaches for designing speech recognisers.

Of the two proposed recogniser designs, the SS&L recogniser has a higher phoneme-like segment recognition accuracy. It also has a faster response time since it time-normalises the spectral data for the whole phoneme-like segment in to a single window and classification is performed on this window only, whereas the 'Sliding Window' recogniser bases its decision on the overall result of classifying data in a short duration window sliding one spectral frame at a time from the begining to the end of the phoneme-like segment. The response time of the 'Sliding Window' recogniser may be improved by sliding the window more than a single spectral frame at time. At the word recognition level, both recogniser designs have similar performances. However, convergence was achieved with a lower lattice depth for the SS&L recogniser when the modified Kashyap & Mittal linguistic processor (Chapter 6, Section 6.3) was used.

Since the proposed recogniser designs were evaluated using software simulation, a measure of the response time was not obtained. It is however felt that at least the SS&L recogniser should be capable of near real time response. This optimism is borne out of the fact that other systems with far greater computational overheads have been

implemented in hardware giving near real time response (eg. the Kohonen "Neural" Phonetic Typewriter [99] has a mean delay on the order of 250 ms per word). Furthermore, a WISARD net based image recognition system has been shown to recognise a 512 x 512 bit image in around 0.25 second, whereas the maximum bit array size processed by the proposed recognisers is 64 x 19. The preprocessor and linguistic processor sections of the recognisers require simple computation. For example, segmentation of the input utterance into phoneme-like segments only requires the computation of the sum of difference in channel energies over successive spectral frames and its comparison to a preset threshold (Chapter 3). Error correction by the linguistic processor requires simple addition of phoneme substitution weights (Chapter 6). Also, the 16-channel filterbank IC enables the spectral representation of an input utterance to be obtained without any computation.

Accuracy of segmenting the input utterance into phoneme-like segments is essential to the performance of both proposed recogniser designs. The sum of the difference in the filterbank channel energies over successive spectral frames was suggested in Chapter 3 for performing segmentation of isolated utterances. Although the segmentation accuracy compares favourably with other reported methods [52], further improvement is needed to enable its use in a practical recognition system. Another problem was in the segmentation of the word 'Three' from the Badii & Binstead word-set. The segments indicated for this word contained parts from following phonemes in the utterance and were therfore not phoneme-like (all other words in the word-set were segmented into phoneme-like segments). Supplementing the segmentation process with more information from the speech signal such as zero-crossing rate and energy measurements, and the use of an

adjustable threshold (as in Itahashi *et al* [36]) could help to at least reduce some of these problems.

A linguistic processor plays an important role in improving the word recognition accuracy of a recognition system based on the sub-word approach. Without the linguistic processor, the proposed recognition systems have a word recognition accuracy around 50%. The linguistic processor is able to improve this to around 90%. Two designs for linguistic processor were experimented with (Chapter 6). The first is an implementation of a design proposed by Kashyap & Mittal [84]. This design needs to be adapted to the recognition system it is to be used with. Modification to this design is suggested to overcome this requirement hence making the design portable (Chapter 6, Section 6.3). There is however an important shortcoming in both designs in that they cannot correct for segment deletion errors in the segmentation process (they can correct substitution errors and segment insertion errors). The effects of this can be reduced somewhat if the segmentation process could be adjusted such that segment deletion errors are minimized (for the segmentation method suggested in this work, this could be achieved by using a lower threshold).

WISARD nets require supervised training ie. training samples must be labelled so that they can be trained into the correct discriminators. This process can be time consuming as it is necessary to ensure that the training samples are properly segmented and are representative of the recognition set. A self organising network like that of Kohonen is easier to train in this respect since training is unsupervised.

7.2 Suggestions for Future Work

The objective of this work was to investigate if the proposed recogniser designs could be used as the basis for a large vocabulary isolated word recogniser. The results obtained for the recognition of the 16 words in the word-set of Badii & Binstead demonstrate that such a goal can be achieved using the proposed designs. In the present state, only 20 phonemes in the English Language (which is slightly less than half the total number of phonemes in English) are represented in the recognisers. It is recommended that further work be done to extend the designs to cover all English phonemes. The following areas of the design need attention :

(i) Improvement in segmentation accuracy and also that the segments detected by the segmentation process are phoneme-like.

(ii) Linguistic processor should be able to correct segment deletion errors.

(iii) The words used to train the recogniser should be carefully selected so that the phoneme-like segments in these words are representative of those in all words in the word-set (to account for contextual variations of each phoneme-like segment in the word-set).

The proposed recogniser designs could be implemented as a plug-in card for the IBM PC AT. The plug-in card would contain the filterbank, A/D circuit and the WISARD net classifier. The

segmentation and linguistic processor functions would be performed by the PC.

Another approach to large vocabulary isolated word recognition is suggested. The need for accurate segmentation of the utterance into phoneme-like segments is essential for the success of the proposed recogniser designs. The self organising Kohonen net does not require prior segmentation. Activations of different areas of the net correspond to different phonemes. Kohonen has reported about 90% phoneme recognition for the Finnish Language [99]. A two layer structure with a 2-dimensional Kohonen net as the first layer and the WISARD net as the second layer is suggested. The Kohonen net will act as the 'retina' for the WISARD net. Each phoneme discriminator in the WISARD net windows the area on the Kohonen net that gets activated by the corresponding phoneme. Thus the WISARD net acts as the labeller for each spectral frame input to the Kohonen net.

REFERENCES

[1] Vaissiere, J.,
'Speech Recognition: A tutorial', in *Computer Speech Processing*, Fallside, F., Woods, W.A. (eds.). Prentice-Hall International, London, pp. 191-242, 1985.

[2] Reddy, D.R.,
'Speech recognition by machine : A review', *Proc. IEEE*, pp. 501-531, April 1976.

[3] White, G.M.,
'Speech Recognition: A tutorial overview', *Computer*, vol.9, pp. 40-53, May 1976.

[4] Thomas, T.,
'Why speech recognisers make mistakes', *Systems International*, pp. 31-34, October 1987.

[5] Peckham, J.,
'Talking to machines', *IEE Review*, pp. 385-388, November 1988.

[6] Aleksander, I., Thomas, W.V., and Bowden, P.A.,
'WISARD - A radical step forward in image recognition', *Sensor Review*, pp. 120-124, July 1984.

[7] Aleksander, I., and Burnett, P.,
'Thinking machines', Oxford University Press, 1987.

[8] Bledsoe, W., and Browning, I.,
'Pattern recognition and reading by Machine', *Proc. Eastern Joint Computer Conf.*, pp. 225-232, 1959.

[9] Nappey, J.,
'Aspects of n-tuple character recognition for a blind reading aid', Ph.D. Thesis, Brunel University, Electrical Engineering and Electronics Department, 1977.

[10] Stonham, T.J.,
'Practical face recognition and verification with WISARD', in *Aspects of Face Processing*, Ellis, H.D., Jeeves, M.A., Newcombe. F., Young, A., Nyhoff, M., (eds). 1986.

[11] 'First Report', Brunel Speech Group, Brunel University, Electrical Engineering and Electronics Department.

[12] Badii, A., Binstead, M.J., Jones, A.J., Stonham, T.J., and Valenzuela, C.L.,
'Applications of n-tuple sampling and genetic algorithms to Speech Recognition', in *Neural Computing Architectures*, Aleksander, I., (ed), pp. 172-216, North Oxford Academic, 1989.

[13] Makino, S., and Kido, K.,
'A speaker independent word recognition system based on phoneme recognition for a large size (212 words) vocabulary', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, (17.8), 1984.

[14] Moriai, S., Makino, S., and Kido, K.,
'Phoneme recognition in continuous speech using phoneme discriminant filters', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 2251-2254, 1986.

[15] Billi, R., Massia, G., and Nesti, F.,
'Word preselection for large vocabulary speech recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 65-68, 1986.

[16] Wilpon, J.G., Juang, B.H., and Rabiner, L.R.,
'An investigation on the use of acoustic sub-word units for Automatic Speech Recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 821-824, 1987.

[17] Svendsen, T., and Soong, F.K.,
'On the automatic segmentation of speech signals', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 77-80, 1987.

[18] Ladefoged, P.,
'The phonetic basis for computer speech processing', in *Computer Speech Processing*, Fallside, F., Woods, W.A. (eds.). Prentice-Hall International, London, pp. 3-27, 1985.

[19] Nolan, F.,
'The nature of speech', in *Electronic Speech Recognition*, Bristow, G., pp. 18-48, Collins, 1986.

[20] Klatt, D.H.,
'Review of the ARPA Speech Understanding Project', *J. Acoustic Society of America*, Vol. 62, pp. 1345-1366, December 1977.

[21] Fujimura, O.,
'Syllable as a unit of speech recognition', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-23, pp. 82-87, February 1975.

[22] Mermelstein, P.,
'Automatic segmentation of speech into syllabic units', *Journal Acoustical Society of America*, Vol. 58, pp. 880-883, October 1975.

[23] De Mori, R., Laface, P., and Piccolo, E.,
'Automatic detection and description of syllabic features in continuous speech', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-24, pp. 365-379, 1976.

[24] Rosenberg, A.E., Rabiner, L.R., Levinson, S.E., and Wilpon, J.G.,
'A preliminary study on the use of demi-syllables in automatic speech recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 967-970, 1981.

[25] Klatt, D.N.,
'Scriber and Lafs : two new approaches to speech analysis', in *Trends in Speech Recognition*, Lea, W.A., (ed), Prentice Hall, Englewood Cliffs, New Jersey, pp. 529-555, 1980.

[26] Schwartz, R., and Makhoul, J.,
'Where the phonemes are: Dealing with ambiguity in acoustic-phonetic recognition', *IEEE Trans. ASSP*, Vol. ASSP-23, pp. 50-53, February 1975.

[27] Reddy, D.R.,
'Segmentation of speech sounds', *J. Acoustic Society of America*, Vol. 40, pp.307-312, August 1966.

[28] Flanagan, J.L.,
'Speech analysis, synthesis and perception', Springer-Verlag, New York, 1965.

[29] Sambur, M.R., and Rabiner, L.R.,
'A speaker-independent digit recognition system', *Bell System Tech. Journal*, Vol. 54, pp. 81-108, 1975.

[30] Lau, Y., and Chan, C.,
'Speech recognition based on zero crossing rate and energy', *IEEE Trans. ASSP*, Vol. ASSP-33, pp. 320-323, February 1985.

[31] Fallside, F.,
'Frequency-domain analysis of speech', in *Computer Speech Processing*, Fallside, F., Woods, W.A. (eds.). Prentice-Hall International, London, pp. 41-80, 1985.

[32] Atal, B.S.,
'Linear predictive coding of speech', in *Computer Speech Processing*, Fallside, F., Woods, W.A. (eds.). Prentice-Hall International, London, pp. 81-124, 1985.

[33] Wakita, H.,
'Direct estimation of the vocal tract shape by inverse filtering of acoustic speech waveforms', *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-21, pp. 417-427, October 1973.

[34] Itakura, F.,
'Minimum prediction residual princple applied to speech recognition', *IEEE Trans. ASSP*, Vol. ASSP-23, pp. 67-72, February 1975.

[35] White, G.M., and Neely, R.B.,
'Speech recognition experiments with linear prediction, bandpass filtering, and dynamic programming', *IEEE Trans. ASSP*, Vol. ASSP-24, pp. 183-188, April 1976.

[36] Itahashi, S., Makino, S., and Kido, K.,
'Discrete-Word Recognition Utilizing a Word Dictionary and Phonological Rules', *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-21, pp. 239-249, June 1973.

[37] Tanaka, K.,
'A parametric representation and a clustering method for phoneme recognition - Application to Stops in a CV environment', *IEEE Trans. ASSP*, Vol. ASSP-29, pp. 1117-1127,
December 1981.

[38] Tanaka, K.,
'A dynamic processing approach to phoneme recognition (Part I) - Feature extraction', *IEEE Trans. ASSP*, Vol. ASSP-27,
pp. 596-608, December 1979.

[39] Wienstein, C.J., McCandless, S.S., Mondshein, L.F.,
and Zue, V.W.,
'A system for acoustic-phonetic analysis of continuous speech', *IEEE Trans. ASSP*, Vol. ASSP-23, pp. 54-67, February 1975.

[40] Kasuya, H., and Wakita, H.,
'Automatic detection of syllable nuclei as applied to segmentation of speech', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 652-655, 1977.

[41] McCandless, S.S.,
'An algorithm for automatic formant extraction using linear prediction spectra', *IEEE Trans. ASSP*, Vol. ASSP-22,
pp. 135-141, April, 1974.

[42] Das, S.K., and Stanat, D.F.,
'Segmentation of utterances of a known phrase using linear threshold techniques', *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-20, pp. 142-150, June 1972.

[43] Cole, R.A, Stern, R.M., Phillips, M.S., Brill, S.M., Pilant, A.P., and Specker, P.,
'Feature-based speaker-independent recognition of isolated English letters', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 731-733, 1983.

[44] Rabiner, L.R., and Juang, B.H.,
'An introduction to Hidden Markov Models', *IEEE Trans. ASSP*, Vol. ASSP-34, pp. 4-16, January 1986.

[45] Baker, J.K.,
'The DRAGON system - An overview', *IEEE Trans. ASSP*, Vol. ASSP-23, pp. 24-29, February 1975.

[46] Baker, J.K.,
'Stochastic modelling for automatic speech understanding', in *Speech recognition*, Reddy, D.R. (ed), Academic Press, pp. 521-542, 1975.

[47] Jelinek, F.,
'Continuous speech recognition by statistical methods', *Proc. IEEE*, Vol. 64, pp. 532-556, April 1976.

[48] Levinson, S.E., Rabiner, L.R. and Sondhi, M.M.,
'An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition', *Bell System Tech. Journal*, Vol. 62, pp. 1035-1074, April 1983.

[49] Averbuch, A., Bahl, L., Bakis, R., Brown, P., Daggett, G., Das, S., Davies, K., De Gennaro, S., de Souza, P., Epstein, E., Fraleigh, D., Jelinek, F., Lewis, B., Mercer, R., Moorhead, J., Nadas, A., Nahamoo, D., Picheny, M., Shichman, G., Spinelli, P., Van Compernolle, D. and Wilkens, H.,
'Experiments with the Tangora 20,000 word speech recognizer', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 701-704, 1987.

[50] Sakoe, H., and Chiba, S.,
'Dynamic programming algorithm optimization for spoken word recognition', *IEEE Trans. ASSP*, Vol. ASSP-26, pp. 43-49, 1978.

[51] Beninghoff, JR., W.J., and Ross, M.J.,
'Investigation of an efficient representation of speech spectra for segmentation and classification of speech sounds', *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-18, pp. 33-42,
June 1972.

[52] Charbonneau, G.R., and Moussa, T.,
'Segmentation of continuous speech by using multidimensional scaling techniques', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 2012-2014, 1982.

[53] Dixon, N.R.,
'An application hierarchy for heuristic rules in automatic phonemic segmentation of continuous speech', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 671-674, 1977.

[54] Shoup, J.,
'Phonological aspects of speech recognition', in *Trends in Speech Recognition*, Lea, W.A., (ed), Prentice Hall, Englewood Cliffs, New Jersey, pp. 125-165, 1980.

[55] Watanabe, T.,
'Segmentation-free syllable recognition in continuously spoken Japanese', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 320-323, 1983.

[56] Watanabe, T.,
'Syllable recognition for continuous Japanese speech recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 2295-2298, 1986.

[57] Hataoka, N., Amano, A., and Yajima, S.,
'VCV segmentation and phoneme recognition in continuous speech', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 2299-2302, 1986.

[58] Gauvain, J.,
'A syllable-based isolated word recognition experiment', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*,
pp. 57-60, 1986.

[59] Höge, H., Littel, B., Marschall, E., Schmidbauer, O., and Sommer, R.,
'Syllable-based acoustic-phonetic decoding and wordhypothesis generation in fluently spoken speech', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1561-1564, 1986.

[60] Togawa, F., Hakaridani, M., Iwahashi, H., and Ueda, T.,
'Voice-activated word processor with automatic learning for dynamic optimization of syllable-templates', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1121-1124, 1986.

[61] Rabiner, L.R., and Wilpon, J.G.,
'Considerations in applying clustering techniques to speaker independent word recognotion', *Journal Acoustical Society of America*, Vol. 66, pp. 663-673, 1979.

[62] Tanaka, A., and Kamiya, S.,
'A speech processing based on syllable identification by using phonlogical patterns', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 2231-2234, 1986.

[63] Broad, D.J.,
'Formants in automatic speech recognition', *Int. J. Man-Machine Studies*, Vol. 4, pp. 411-424, July, 1972.

[64] Green, P.D., and Wood, A.R.,
'A representational approach to knowledge based acoustic-phonetic processing in speech recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1205-1208, 1986.

[65] Sakai, T., and Doshita, S.,
'The automatic speech recognition system', *IEEE Trans. Elec. Commun.*, Vol. 12, pp. 835-846, December, 1963.

[66] Kot, L.,
'A syntax-controlled segmentation of speech signal on the basis of dynamic spectra', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 2015-2017, 1982.

[67] Fu, K.S.,
'*Syntactic methods pattern recognition*', Academic Press, 1974.

[68] Mermelstein, P.,
'A phonetic context controlled strategy for segmentation and phonetic labelling of speech', *IEEE Trans. ASSP*, Vol. ASSP-23, pp. 79-82, February 1975.

[69] De Mori, R.
'A descriptive technique for automatic speech recognition', *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-21, pp. 89-100, April 1973.

[70] Bridle, J.S., and Sedgwick, N.C.,
'A method for segmenting acoustic patterns with applications to automatic speech recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 656-659, 1977.

[71] Andre-Obrecht, R.,
'Automatic segmentation of continuous speech signals', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*,
pp. 2275-2278, 1986.

[72] Andre-Obrecht, R.,
'A new statistical approach for the automatic segmentation of continuous speech signals', *IEEE Trans. ASSP*, Vol. ASSP-36, pp. 29-40, January 1988.

[73] Sugamura, N., Shikano, K., Furui, S.,
'Isolated word recognition using phoneme-like templates', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 723-726, 1983.

[74] Hinton, H.S., and Siegel, L.J.,
'Speaker independent isolated word automatic speech recognition using computer generated phonemes', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 727-730, 1983.

[75] Morii, S., Niyada, K., Fuji, S., and Hoshimi, M.,
'Large vocabulary speaker-independent Japanese speech recognition system', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 866-869, 1985.

[76] Hiraoka, S., Morii, S., Hoshimi, M., and Niyada, K.,
'Compact isolated word recognition system for large vocabulary', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 69-72, 1986.

[77] Morishima, S., Harashima, H., and Miyakawa, H.,
'A proposal of a knowledge based isolated word recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 713-716, 1986.

[78] Aikawa, K., Sugiyama, M., and Shikano, K.,
'Spoken word recognition based on top-down phoneme segmentation', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 33-36, 1985.

[79] Kawabata, T.,
'Word spotting method based on top-down phoneme verification', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1438-1441, 1987.

[80] Shipman, D.W., and Zue, V.W.,
'Properties of large lexicons : Implications for advanced isolated word recognition systems', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 546-549, 1982.

[81] Huttenlocher, D.P., and Zue, V.W.,
'A model of lexical access from partial phonetic information', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, (26.4), 1984.

[82] Lagger, H., and Waibel, A.,
'A coarse phonetic knowledge source for template independent large vocabulary word recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 862-865, 1985.

[83] Okuda, T., Tanaka, E., and Kasai, T.,
'A method for the correction of garbled words based on the Levenstein metric', *IEEE Trans. on Computers*, Vol. C-25, pp. 172-177, February 1976.

[84] Kashyap, R.L., and Mittal, M.C.,
'A new method for error correction in strings with applications to spoken word recognition', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 675-678, 1977.

[85] Allerhand, M.,
'*Knowledge-based speech pattern recognition*', Kogan Page, London, 1987, pp. 45.

[86] Hecht-Nielsen, R.,
'Neurocomputing: picking the human brain', *IEEE Spectrum*, pp.36-41, March 1988.

[87] Lippmann, R.P.,
'An introduction to computing with neural nets', *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[88] Rumelhart, and McClelland, (eds)
'Parallel Distributed Processing', Vol. 1, Cambridge, MA, MIT Press, 1986.

[89] Rumelhart, and McClelland,
'Parallel Distributed Processing', Vol. 2, Cambridge, MA, MIT Press, 1986.

[90] Aleksander, I., (ed)
'Neural Computing Architectures', North Oxford Academic.

[91] Aleksander, I., Stonham, T.J., and Wilkie, B.A.,
'Computer vision systems for industry: WISARD and the like', *Digital Systems for Industrial Automation*, Vol. 1, No. 4, pp. 305-320, 1982.

[92] Kohonen, T., Mäkisara, K., and Saramäki, T.,
'Phonotopic maps - Insightful representation of phonological features for speech recognition', *Proc. IEEE 7th Inter. Conf. on Pattern Recognition*, pp. 182-185, 1984.

[93] Fukushima, K.,
'A neural network for visual pattern recognition', *IEEE Computer*, pp. 65-75, March 1988.

[94] Swinbanks, D.,
'New initiative in Japan to develop neural networks', *Nature*, Vol. 336, pp. 7, 3 November 1988.

[95] Fallside, F., Woods, W.A., (eds.)
'*Computer Speech Processing*', Prentice-Hall International, London, 1985.

[96] Bristow, G., (ed)
'*Electronic Speech* Recognition', Collins, 1986.

[97] Parsons, T.,
'*Voice and speech processing*', McGraw-Hill, 1987.

[98] Holmes, J.N.,
'*Speech synthesis and recognition*', Van Nostrand Reinhold (UK), 1988.

[99] Kohonen, T.,
'The "neural" phonetic typewriter', *IEEE Computer*, pp. 11-22, March 1988.

[100] Naylor, J., and Li, K.P.,
'Analysis of a neural network algorithm for vector quantization of speech parameters', *Abstracts of the First Annual International Neural Networks Society Meeting*, Vol. 1, Supplement 1, pp. 310, 1988.

[101] Trehern, J.F., Jack, M.A., and Laver, J.,
'Speech processing with a Boltzmann machine', *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 721-724, 1986.

[102] Kämmerer, B., and Küpper, W.,
'Experiments for isolated-word recognition with single and multi-layer perceptrons', *Abstracts of the First Annual International Neural Networks Society Meeting*, Vol. 1, Supplement 1, pp. 302, 1988.

[103] Byrne, W., and Shamma, S.,
'Dynamical networks for speech recognition', *Abstracts of the First Annual International Neural Networks Society Meeting*, Vol. 1, Supplement 1, pp. 291, 1988.

[104] Anderson, S., Merrill, J., and Port, R.,
'Effects of network topology on speech categorization', *Abstracts of the First Annual International Neural Networks Society Meeting*, Vol. 1, Supplement 1, pp. 286, 1988.

[105] Tishby, N.,
'Nonlinear dynamical modelling of speech using neural networks', *Abstracts of the First Annual International Neural Networks Society Meeting*, Vol. 1, Supplement 1, pp. 279, 1988.

[106] Hassan, N.K.,
'An adaptive artificial vision system for recognition and handling of industrial parts', Ph.D. Thesis, Brunel University, Electrical Engineering and Electronics Department, 1987.

[107] Aleksander, I.,
'Emergent intelligent properties of progressively structured pattern recognition nets', *Pattern Recognition Letters*, Vol. 1, pp. 375-384, July 1983.

[108] Aleksander, I., and Stonham, T.J.,
'Guide to pattern recognition using random-access memories', *Computers and Digital Techniques*, Vol.2, No. 1, pp. 29-40, February 1979.

[109] Tappert, C.C.,
'Experiments with a tree-search method for converting niosy phonetic representation into standard orthography', *IEEE Trans. ASSP*, Vol. ASSP-23, pp. 129-135, February 1975.

[110] Ito, M.R., and Donalson, R.W.,
'Zero-crossing measurements for analysis and recognition of speech sounds', *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-19, pp. 235-242, September 1971.

[111] Shearme, J.N., and Leach, P.F.,
'Some experiments with a simple word recognition system', *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-16, pp. 256-261, June 1968.

[112] Thomas, I.B., Hill, P.B., Carroll, F.S., and Garcia, B.,
'Temporal order in the perception of vowels', *Journal Acoustical Society of America*, Vol. 48, No. 4, pp. 1010-1013, March 1970.

[113] LaRiviere, C., Winitz, H., and Herriman, E.,
'Vocalic transitions in the perception of voiceless initial stops', *Journal Acoustical Society of America*, Vol. 57, No. 2, pp. 470-475, February 1975.

[114] Rabiner, L.R., and Sambur, M.R.,
'An algorithm for determining the endpoints of isolated utterances', *Bell Syst. Tech. J.*, Vol. 54, pp. 297-315, February 1975.

APPENDIX A

The segment labels and the corresponding sound in an utterance

Table 'A' shows the class labels of the sub-word segments recognised by the speech recogniser designs presented in this dissertation and the corresponding sound in an utterance that they represent.

Table A

/b/	Begun
/ɪ/	wIn
/g/	beGun
/ʌ/	begUn
/n/	beguN
/k/	Cooler
/u/	tOOt
/l/	cooLer
/ə/	coolER
/r/	Run
/s/	See
/i/	sEE
/t/	Tee
/w/	Win
/ʃ/	SHoe
/ɔ/	wAnt
/æ/	tAttoo
/θ/	tooTHache
/d/	wonDer
/eɪ/	toothAche
/θr/	THRee
/ri/	thREE

APPENDIX B

Data for calculating the class label substitution weights

Table 'B' gives the total number of samples of each class ie. n_b (Chapter 6, Section 6.2.3, page 152) submitted to the recogniser for recognition over tests with 5, 10, 15 and 20 trainings.

Table B

Class	n_b
/b/	96
/ɹ/	196
/g/	96
/ʌ/	492
/n/	592
/k/	196
/u/	724
/l/	100
/ə/	444
/r/	200
/s/	100
/I/	384
/t/	796
/w/	396
/ʃ/	100
/ɔ/	100
/æ/	92
/θ/	96
/d/	200
/eɹ/	148
/θr/	100
/ri/	48

Sections B.1 and B.2 tabulate data for $n_{a,b}$ ie. the number of times that any given class a is observed at the output of the recogniser but the actual class input to the recogniser was some other class b (Chapter 6, Section 6.2.3, page 152), for the SS&L based recogniser and the 'Sliding Window' recogniser respectively. In formulating Tables 6.2 and 6.4, cases for which p(a,b) is less than 0.005 have been ignored.

B.1 'Separate Segmentation & Labelling' Based Recogniser

Table B.1 gives data for $n_{a,b}$ for the SS&L based recogniser.

Table B.1

Observed Class	Actual Class								
/g/	/l/	/r/	/æ/	/ri/	/i/	/u/	/θ/	/k/	/ə/
No. of errors	1	1	6	2	5	9	4	3	2

Observed Class	Actual Class
/ʌ/	/ə/
No. of errors	1

Observed Class	Actual Class					
/ɪ/	/l/	/i/	/æ/	/u/	/ri/	/n/
No. of errors	4	17	9	63	2	1

Observed Class	Actual Class		
/b/	/ɪ/	/æ/	/u/
No. of errors	2	9	4

Observed Class	Actual Class					
/n/	/ə/	/i/	/u/	/eɪ/	/w/	/d/
No. of errors	6	30	27	7	2	8

Observed Class	Actual Class					
/k/	/b/	/ʌ/	/t/	/æ/	/θr/	
No. of errors	3	1	6	4	2	

Observed Class	Actual Class											
/u/	/ɹ/	/n/	/l/	/r/	/i/	/t/	/æ/	/ri/	/eɪ/	/k/	/w/	/ə/
No. of errors	11	10	23	1	158	12	23	29	19	1	57	1

Observed Class	Actual Class											
/t/	/b/	/g/	/k/	/ʌ/	/s/	/ʃ/	/æ/	/θr/	/θ/	/u/	/ɹ/	/d/
No. of errors	1	2	11	13	15	3	12	77	21	3	3	4

Observed Class	Actual Class
/r/	/w/
No. of errors	9

Observed Class	Actual Class
/ɹ/	/l/
No. of errors	

Observed Class	Actual Class				
/ə/	/ʌ/	/n/	/æ/	/ɔ/	/w/
No. of errors	45	1	1	7	2

Observed Class	Actual Class
/s/	/s/
No. of errors	

Table B.1 (contd.)

Observed Class	Actual Class
/w/	/r/
No. of errors	1

Observed Class	Actual Class
/ʃ/	/ʃ/
No. of errors	

Observed Class	Actual Class
/ɔ/	/ʌ/
No. of errors	57

Observed Class	Actual Class
/θ/	/g/
No. of errors	1

Observed Class	Actual Class				
/æ/	/ɒ/	/ʌ/	/r/	/eɹ/	/ɔ/
No. of errors	5	3	2	2	1

Observed Class	Actual Class
/θr/	/t/
No. of errors	1

Observed Class	Actual Class											
/d/	/g/	/n/	/ɒ/	/r/	/ʌ/	/æ/	/t/	/ri/	/θ/	/eɹ/	/w/	/ɹ/
No. of errors	1	2	1	3	2	6	1	7	1	1	8	1

Observed Class	Actual Class
/eɹ/	/eɹ/
No. of errors	

Observed Class	Actual Class		
/i/	/ɪ/	/u/	/eɹ/
No. of errors	2	26	2

Observed Class	Actual Class
/ri/	/i/
No. of errors	3

B.2 'Sliding Window' Recogniser

Table B.2 gives data for $n_{a,b}$ for the 'Sliding Window' based recogniser.

Table B.2

Observed Class	Actual Class	
/g/	/θ/	/d/
No. of errors	3	1

Observed Class	Actual Class
/ʌ/	/ə/
No. of errors	1

Observed Class	Actual Class				
/ʊ/	/g/	/i/	/u/	/æ/	/ri/
No. of errors	2	14	35	5	1

Observed Class	Actual Class					
/b/	/ʊ/	/t/	/æ/	/θr/	/u/	/k/
No. of errors	15	48	17	4	46	4

Observed Class	Actual Class				
/n/	/ə/	/w/	/r/	/u/	/i/
No. of errors	3	27	4	21	7

Observed Class	Actual Class				
/k/	/u/	/t/	/i/	/θr/	/θ/
No. of errors	3	4	3	7	1

Observed Class	Actual Class									
/u/	/n/	/l/	/w/	/i/	/t/	/æ/	/ri/	/eʊ/	/ʊ/	/d/
No. of errors	4	5	17	65	5	9	16	1	56	3

Observed Class	Actual Class									
/t/	/g/	/ʌ/	/k/	/d/	/u/	/æ/	/i/	/θ/	/θr/	/ri/	/ʊ/
No. of errors	12	7	15	28	1	12	1	74	66	2	3

Observed Class	Actual Class	
/r/	/ʌ/	/w/
No. of errors	6	6

Observed Class	Actual Class		
/l/	/i/	/u/	/t/
No. of errors	23	10	1

Observed Class	Actual Class					
/ə/	/ʌ/	/w/	/r/	/eʊ/	/k/	/ɔ/
No. of errors	89	59	21	1	1	14

Observed Class	Actual Class
/s/	/s/
No. of errors	

Table B.2 (contd.)

Observed Class	Actual Class
/w/	/u/
No. of errors	2

Observed Class	Actual Class
/ʃ/	/ʃ/
No. of errors	

Observed Class	Actual Class
/ɒ/	/ʌ/
No. of errors	90

Observed Class	Actual Class
/θ/	/g/
No. of errors	1

Observed Class	Actual Class	
/æ/	/eɪ/	/ə/
No. of errors	2	1

Observed Class	Actual Class			
/ri/	/l/	/i/	/u/	/ɹ/
No. of errors	3	14	22	2

Observed Class	Actual Class													
/d/	/b/	/ɹ/	/g/	/u/	/l/	/ɒ/	/w/	/ʌ/	/r/	/i/	/t/	/æ/	/eɪ/	/ɔ/
No. of errors	8	15	3	24	15	53	90	101	27	13	32	32	16	2

Observed Class	Actual Class						
/eɪ/	/ɹ/	/g/	/l/	/ɒ/	/æ/	/i/	/u/
No. of errors	2	3	3	2	5	1	6

Observed Class	Actual Class
/θr/	/t/
No. of errors	4

Observed Class	Actual Class							
/i/	/g/	/l/	/n/	/u/	/t/	/æ/	/eɪ/	/ɹ/
No. of errors	1	1	6	55	2	1	6	4