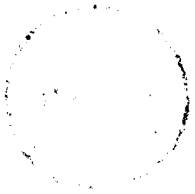
UNIVERSITY OF ESSEX

Department of Electrical Engineering Science

WISPA: A System for Word Identification in Speech by Phonetic Analysis



A thesis presented in partial fulfilment of the requirements for the degree of Doctor of Philosophy

John P Yardley

Division of Numerical Analysis and Computer Science

National Physical Laboratory

Teddington, Middlesex, England

May 1981

To Teresa, Victoria and Francesca.

Summary

The WISPA system for the automatic recognition of continuous speech is a research tool designed to evaluate various strategies for recognition which use the human model as their basis. This is reflected in a system which encodes speech into phonetic features rather than acoustic patterns and which is trained by "intelligently" combining several utterances of the same word into a single template. The benefits of a phonetic encoding scheme are less speaker dependence and greater processing potential. Indeed, the baseline system using an LSI-11 microcomputer is able to continuously process vocabularies of up to 30 words in real-time.

A fundamental component of the WISPA system is the NPL Speech Input Device (SID) which provides "front-end" hardware preprocessing. While WISPA has been designed for ultimate use with the latest generation of SID, the mark III, it gives excellent performance on the less powerful SID II.

The thesis discusses the background to the development of WISPA, its design, philosophy and performance measurement. It also presents an overview of speech recognition theory and technology in order to contrast the WISPA system with those developed elsewhere. Finally, it summarises the design concepts of WISPA and looks at the implications for future work.

Acknowledgements

I should like to acknowledge the contribution of Brian Pay, team leader of the NPL Speech Recognition Project, whose early work formed the basis for the subsequent development of WISPA, and with whom I "thrashed out" many of the ideas presented in this thesis. I am also indebted to Roger Manning and Ralph Rengger for their time in discussing the hardware aspects of the project; to David Schofield whose help was invaluable in the drafting of chapter 2; and to Martyn Armstrong, a sandwich course student from City University, who helped write the code for WISPA. Finally, I should like to thank my wife and my mother for proof reading the initial drafts.

CONTENTS

CHAPTER 3: THE WISPA SYSTEM - CONTINUED

CHAPTER 4: PERFORMANCE MEASUREMENT

CHAPTER 5: RESUME, FUTURE WORK & CONCLUSIONS

CHAPTER 6: REFERENCES

CHAPTER 1

Introduction

1.1 Objectives and Overview

The work described in this thesis is concerned with the development of a task-independent continuous speech recognition system in which the training process forms the basis for subsequent recognition. This is an unusual approach, since in most experimental and commercial recognisers, relatively little attention is given to training algorithms and their effect on recognition performance.

The objective has been to develop a speech recognition system structure in which the relationship between training and recognition is well-defined, but the actual training/recognition strategy may be altered. Using this derived structure, it has been aimed to show that a strategy employing phonetic analysis can demonstrate a low-cost, commercially viable continuous speech recognition system.

The author undertook the work as an external student, while working at the National Physical Laboratory (NPL). The remainder of this chapter outlines

the background and history of speech recognition at NPL and explains the reasons that the project evolved.

Chapter 2 looks at speech recognition theory and technology to introduce some of the terminology and also to relate the author's work to that done elsewhere.

Chapter 3 discusses the design of the WISPA speech recognition system - one of the practical targets during the research period - and Chapter 4 describes the performance measurement methods used to compare experimental strategies. As such, these two chapters represent the bulk of the work done for the thesis.

Chapter 5 summarises the findings of the thesis and studies the implications for future work.

1.2 Background

In 1965, one of the groups in the newly formed Computer Science Division at NPL was researching optical character recognition under the leadership of Ted Newman. Newman had, for some years, been giving thought to the problems of manual testers of thermometers, who experienced difficulty in re-focussing their eyes from the temperature scales on the thermometers to the report sheets to be filled in. He decided that a system of automatic speech recognition would be of great help to such testers. Furthermore, it seemed likely that some of the auto-correlation techniques he had developed for character recognitition might be applicable to formant analysis in speech. So, at that time he was joined part-time by Brian Pay and Roger Manning, who

began to investigate the use of analogue LCR delay-lines in measuring formant frequencies.

By 1968, Pay and Manning were fully occupied by the project having disgarded the analogue delay-line in favour of a wholly digital auto-correlator using clipped speech and digital shift-registers. This followed some successful work by British Telecommunications Research (now part of Plessey) who, under contract to the Department, had looked at correlation techniques using ternary shift-registers proposed by Newman.

Up to this time, Pay and Manning had also been conducting a whole series of psychological experiments designed to isolate the features in speech crucial to the perception of its meaning. This they did by artificially distorting speech in a controlled way, and testing its subjective effect on perception. The results of these experiments led to an early commitment to the technique of voicing classification by the measurement of two formants.

In 1969, the group was able to demonstrate various speaker-independent word recognisers, based on NPL developed feature detectors and a modified system of "sequential-logic", patented some years earlier by Newman. These early recognisers, called Speech Analysis Machines or SAMs had only limited vocabularies and were implemented completely in hardware. Nevertheless, their effectiveness in real applications were subsequently proved in two particular systems for control and data-entry. The first was a speech-driven slide projector [Rengger 1973], which was a demonstration application tested by many hundreds of visitors to NPL and various exhibitions. The second, was a pilot scheme for medical interviewing and diagnosis [Budgett 1978], which used a

multi-speaker three word vocabulary - "yes", "no" and "don't know". It was interesting that the latter application was able to utilise work done some five years earlier.

The advent of laboratory based mini-computers and the purchase of a DEC PDP-8 in 1970 led to a change in the direction of research. Instead of constructing the complete system in hardware, it was decided to reduce its role to feature detection and to execute the sequential-logic process for word recognition as a software routine. This heralded the hardware preprocessor known as the Speech Input Device or SID [Rengger and Manning 1973]. Ralph Rengger joined the group at this time to assist initially in the engineering of SID and SAM, and the interfacing of SID to the PDP-8.

Subsequent work by Pay showed that the sequential-logic methods he had developed for execution on the computer were applicable equally well to both the recognition of isolated words and continuous speech. As a result, by 1974 the group had a very effective continuous speech recogniser, significantly ahead of other researchers. The system was able to recognise strings of numerals from a single speaker with a word recognition rate of 98%. The training procedure needed to achieve that rate was very labour intensive, requiring the efforts of a skilled person with expert knowledge of the system.

In 1975 attention returned to applications of speech recognition. Upon joining the group at this time, the author was concerned about the difficulty of applying the recognition software to an application system. As a result, a software interface to the BASIC language was developed which gave users the ability to experiment with man-machine protocols quickly and easily, and to

establish some ground rules for effective speech input - the system and interface is described in a later report by Pay and Evans [Pay and Evans 1978]. Many interactive systems were developed for a variety of tasks, and considerable effort was applied to the analysis of data produced under operational conditions [Pay 1978]. A popular demonstration was the simulation of a speech-controlled aircraft in which the pilot could perform several navigational and control functions.

Figure 1.1: Brian Pay (standing) and the author testing the pilot-cockpit demonstration.

About the middle of 1975, work began on developing a new generation of SID, the mark III (SID I was a prototype "breadboard" device, while SID II was an "engineered" version of this prototype). Technology had moved some way since the original design of SID and it was now possible to consider feature extraction techniques which hitherto had not, or could not, be realised in a practical system. A particularly useful innovation was the analogue delay-line, now available as a compact MOS device. In addition, earlier applications, like using speech over a telephone line, gave a better idea of the design criteria for SID III.

By the autumn of 1977, the training problem was under serious consideration as a primary goal of this thesis, and research work had begun on finding methods to automatically train the SID/PDP-8 system for new speakers and vocabularies. First efforts were not too successful in that the automatic training procedures could never be made to yield the recognition performance achieved using the manual procedures; their role was therefore limited to assisting the manual training by extracting areas of commonality in training utterances. Nevertheless, the work gave a valuable insight into the importance of training in overall system design. It became clear that training and recognition could not be regarded in isolation and must have a well-defined relationship. Furthermore, the way in which utterances were combined into single templates appeared to be an area hardly considered by other researchers; many used a single template for each training utterance or used simple normalisation rules for merging templates.

Experience gained with interactive systems and training procedures during the previous two years led to the decision in 1978 to re-design the speech system

from the ground upwards. The new system was to be based on the LSI-11 micro-processor and SID hardware, with provision for the new generation SID, to be known as SID III, then nearing completion. Automatic training facilities were to be an intrinsic part of the design structure, as well as the ability to experiment with more than one training and recognition strategy (it is this work that is described in subsequent chapters).

In parallel with the author's development of the new system, to be known as WISPA, work was in hand on processes for extracting a final recognition decision from a sequence of "spotted" words. This was termed the semantic segmentation process. Other ongoing work on feature extraction from SID III enabled a basic SID-III/LSI-11 system to be operational by the end of 1980.

Current research has centred around fundamental analysis of SID III data, and experiments with training/recognition strategies. Much effort has gone into automatic strategy testing, using a large speech database of SID data, table-driven software that may be modified in "batch-mode", and a series of pre-defined experiments (see appendix IV).

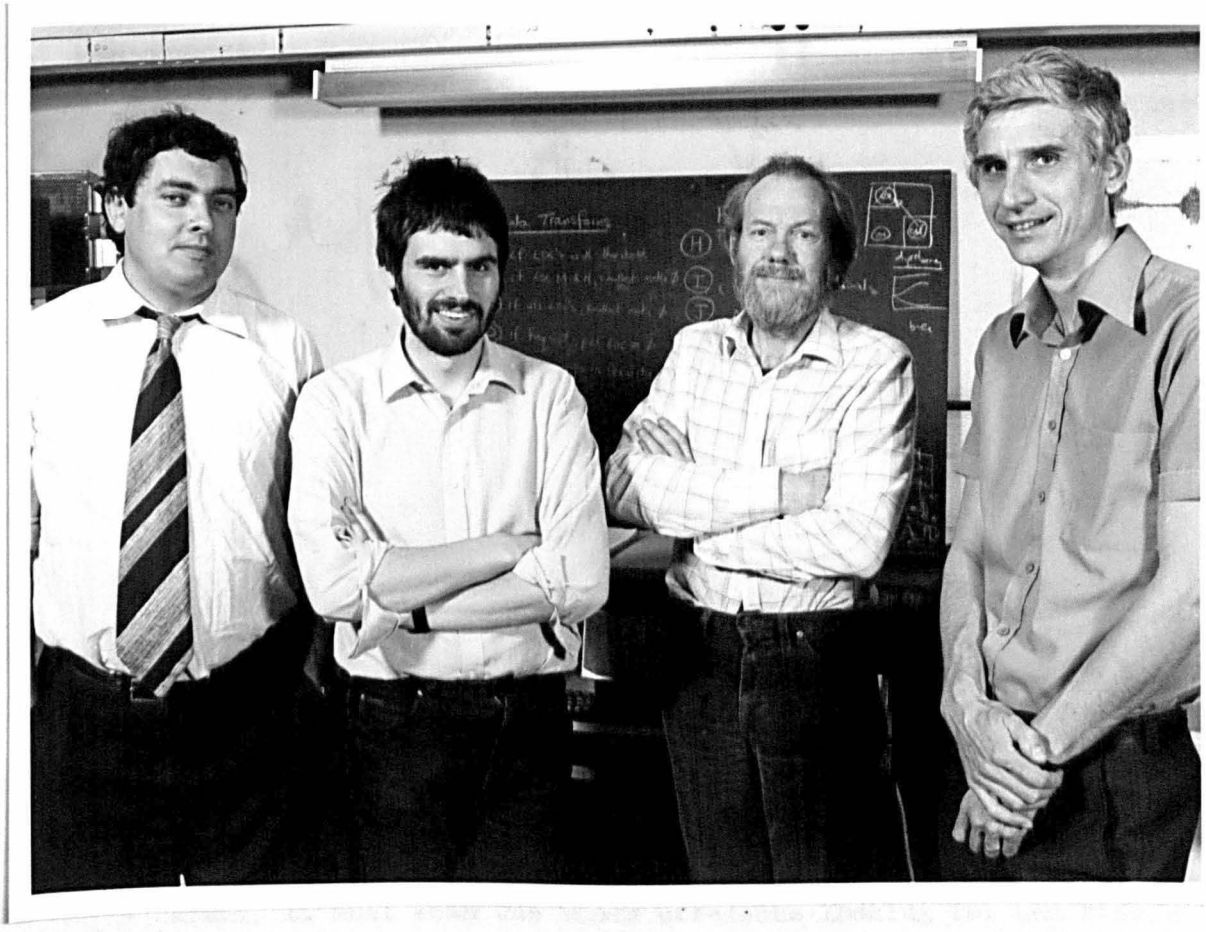


Figure 1.2: The NPL Speech Recognition Group 1981.

(Left to right: Roger Manning, John Yardley, Brian Pay & Ralph Rengger)

1.3 Comparison of NPL Speech Recognition Philosophies

As a prelude to a more detailed discussion of speech recognition and the approach taken in WISPA, we introduce some basic concepts of the NPL speech recognisers and compare the earlier philosophies with those described in this

thesis.

One of the primary processes in any speech recognition system is to compare (or match) a test utterance of unknown meaning with a stored template of each word in a vocabulary for which recognition is defined. Having compared each word, it must decide which, if any, is sufficiently like the test utterance to be recognised.

A system that allows the user to only utter a single word at a time, (ie he must precede and follow the word with silence), is termed an isolated word recogniser. One that permits words to be joined together is a connected word recogniser. If, in addition, the system permits connected words outside the vocabulary to be used - although, of course it cannot recognise them - it is termed a continuous speech recogniser (more about this in chapter 2).

A continuous speech recogniser does not know where in the utterance each word begins or ends since there are not usually distinct gaps of silence to tell it. As a result, it must scan the whole utterance looking for the best fit for each word, and so the comparison process is executed many more times than for isolated word recognition.

The most effective comparison algorithms involve computation-time proportional to the product of the number of elements in the section of test utterance and the number of elements in the template. If the test utterance and template are resolved to the same degree, this approximates to the square of elements in the template. Clearly, by reducing the element resolution of the template, we may reduce computation-time according to a square law.

The basis of the NPL philosophy is that the resolution can be reduced to allow real-time execution of a continuous speech recognition process with modest processing power (ie less than 0.15 mips) and large vocabularies (30 - 60 words). This is achieved by using hardware (SID) and software to preprocess the speech into phonetic features [*] - that is features based on the phoneme as the basic unit of speech. In so doing, it is sought to remove that information in the speech waveform which is not relevent to the perception of its meaning. If this is performed correctly, that includes information about the speaker's identity, gender, and indeed, state-of-mind (eg frightened, stressed, tired, etc). The real-time implications of phonetic encoding are in fact a spin-off from the fundamental principle that the phoneme-based unit is the correct unit into which to segment speech. The phoneme is, by definition, the minimum unit of speech which, if changed, can yield a difference in meaning.

This basic philosophy applies to both the original PDP-8 system and to the WISPA system. The differences between systems lay in

1. the phonetic encoding scheme
2. the comparison process
3. the training philosophy

The method of phonetically encoding speech employed by the PDP-8 system produces features each described by one of several mutually-exclusive phonetic

---

* The term "phonetic feature" is used throughout the thesis to describe information which may be directly related to phonological events in general. Although in some cases the term may correspond to "accepted" articulatory and distinctive features, it is not intended to be interpreted in such a specific sense.

groups. At the lowest level, this involves classification into one of the following:

Fricative	F1	"f" as in face
	F2	"s" as in see
	F3	"sh" as in she
Voiced	V1	"n" as in nine
	V2	"oo" as in tool
	V3	"o" as in hot
	V4	"aa" as in far
	V5	"e" as in met
	V6	"ee" as in seed
Quiet	Q1	" " as in

These can then be further classified according to their sequencing (eg a diphthong, D1 say, might be defined as V2 followed by V3). Each feature is appended with its duration, expressed as a count of the number of SID samples used in producing the feature - SID is sampled at 10 ms intervals. A typical "four" might be stored as:

F1	15
V4	26

and "nine" as:

V1	20	
D3	25	where: D3 represents the sequence V4,V5,V6
V1	16	

The main disadvantage of this scheme is the mutual exclusivity of features. Experiments have shown that features that are perceived as the same, often contain parameters which overlap in value. In the WISPA system this is overcome by allowing overlapping features in the data-structure. Furthermore, the WISPA system grades the parameters in each feature to permit selective comparisons to be made.

The comparison method on the PDP-8 system is a string-matching technique which is highly optimised for speed. The optimisation is too severe to permit significant deviation between template and test utterance, and relies on words being stored as several templates to accommmodate variations. WISPA makes use of a very generalised dynamic programming process and initially uses little optimisation. The objective is only to optimise when it can be proved that the optimisation does not impair performance. The WISPA costing algorithm, for measuring the distance between features, is more "intelligent" than its predecessor; it is able to automatically take account of the statistical "importance" of features by observing a series of training utterances.

The use of multiple templates for each word in the PDP-8 system is particularly prevalent when the system is trained for multi-speaker use. The effect of this and the mutually-exclusive features means that some words can often be recognised erroneously, due to a proliferation of acceptable fits. For example, two formant frequencies varying by only a single level, can result in three different classifications of voicing. In a two-feature word, say, where one feature is allowed with a choice of three types of voicing, it is easy to see that the template forms a very loose specification. The WISPA philosophy is that each word should be represented by a single template which contains information about the statistical importance of each feature. Further, that the process for matching utterances at recognition-time, should be identical to the matching process used when building templates.

Finally, a note on the size of vocabularies. There are relatively few speech recognition applications that require a vocabulary much in excess of 30 words at any one time. By the use of the syntax and semantics of the application,

it usually possible to restrict the size of vocabularies to within this limit. Indeed, some very ambitious speech understanding tasks are demonstrated on the ARPA projects (see chapter 2), using average branching factors of less than 40. This thinking is reflected in the PDP-8 and WISPA systems, which both operate schemes for sub-setting large vocabularies into several smaller ones, each applicable at different points in the interaction. Although this thesis is not directly concerned with the syntactic and semantic aspects of speech understanding, a subsetting mechanism is fundamental to the system design.

CHAPTER 2

Speech Recognition Theory and Technology

2.1 Introduction

This chapter looks at speech recognition in general, its theory, terminology and technology, and by way of example, some other researchers' approach to the problem. The objective is to make succeeding chapters more understandable, and to relate the author's work to that done elsewhere.

2.2 Basic Theory

Aspects of speech have been studied under disciplines such as acoustics, phonetics, physiology, engineering, psychology, each of which has an extensive literature and tradition. The pioneering work on the scientific study of speech from the point of view of electrical communication is well covered in the book by J L Flanagan [1972] who himself has been associated with much of the work, as leader of a group at Bell Telephones.

The speech communication channel from one human to another may be represented thus:

Transmitter (speaker)			Receiver (listener)
BRAIN -->	MOUTH -->	ACOUSTIC SIGNAL -->	EAR --> BRAIN
Encode			Decode

The brain decides what message is to be conveyed by speech, and encodes it into a sequence of articulate sounds via the vocal mechanism. The acoustic waveform carries the information about the message in a way which reflects the positions taken up by the mouth etc in the appropriate order. The listener, by ear and brain, must decode the message back into meaningful units. The transaction will be successful provided the transmitter and the receiver mutually agree on the coding rules.

Although speech recognition models the right-hand half of this chain, an understanding of the speech generation process (the left hand side) has proved to be useful in making theoretical models which can be used for recognition purposes.

2.2.1 The Brain

Experiments on perception [Pierce and Karlin 1947] show that the brain seems unable to make use of information at more than 50 bits per second (bps). For a machine to capture the acoustic signal faithfully, it requires more than 20,000 bps, so that a major objective of speech recognition is to remove redundancy. A single figure cannot be given for the information content of speech because it contains different sorts of information, such as literal

meaning, and the identity and state of the speaker. A current target for intelligible speech is 100 bps, which has not yet been achieved by available communication vocoders.

The efforts of phoneticians [Bloch and Trager 1942] have defined linguistic units called phonemes. (Appendix III gives a summary of some of the terms used in phonetics.) Speech can be regarded as a sequence of phonemes and processes in the brain convert this sequence into literal meaning, eg "words". The listener uses his inherent linguistic knowledge and the context of meaning to interpret the sequence meaningfully, even though the signal at the phoneme level may be poorly defined.

This is an important point, since in normal conversational speech, humans speak only sufficiently clearly to be understood, and in fact, individual words are not 100% recognisable in isolation.

2.2.2 The Ear

Disregarding the effects of displacements in time, it generally true to say that if two acoustic signals sound identical, then their information content is the same. The ear, or rather the "hearing" system of ear + brain, must then process both signals to produce the same result. A mathematical model of the ear (the hearing process) could be used for data reduction not only of speech but of any sound; this problem is therefore of great interest to communications engineers. The anatomy of the ear is extremely complicated, and the mechanism of its operation is far from fully understood [Moore 1979;

Fourcin 1976]. However, models of hearing do exist, derived mainly from perception experiments. They can be used to derive the broad acoustic parameters of recognition systems. For example, the transmission properties of the ear can indicate the relative importance of different gross features of the acoustic signal. No doubt, if the mechanism of hearing (the ear+brain) were understood, this might be used as a model for actual recognisers.

Though the transmission properties are useful in the acoustic analysis of speech, the fundamental mechanism of hearing has not provided a guide to the design of automatic recognisers.

2.2.3 The Mouth

The mouth is much better understood than the ear, and useful analytical models can be used for the speech generation process. In this context, a "model" means a simplified representation of the actions of the parts of the mouth; the mathematics of this simplified representation can be used in place of the mathematics of the mouth itself. These models are therefore useful in designing speech recognition equipment, and indeed, one such model of the mouth has formed the basis of much of the work at NPL.

The vocal cords produce an excitation of cavities in the mouth and nose. The size and activity of these cavities are controlled by the lips and tongue. Other sounds are excited by a stream of air only. Phonemes can be related to these activities; sounds are largely characterised by the sizes of the several resonant cavities, and consonants in particular represent different

types of dynamic effects.

Figure 2.1 indicates diagrammatically some of the features of the human vocal mechanism and their effects on speech.

The significance of a model is that it allows signal processing techniques to be used to recover, from the acoustic signal, model parameters requiring many fewer bits per second. These techniques include auto-correlation, linear predictive coding, Fourier transforms and spectrum analysis generally. The processing may be carried out using special purpose equipment (analogue or digital), or it may be performed entirely digitally by general-purpose computing devices.

2.3 Types of Recognisers and Terminology

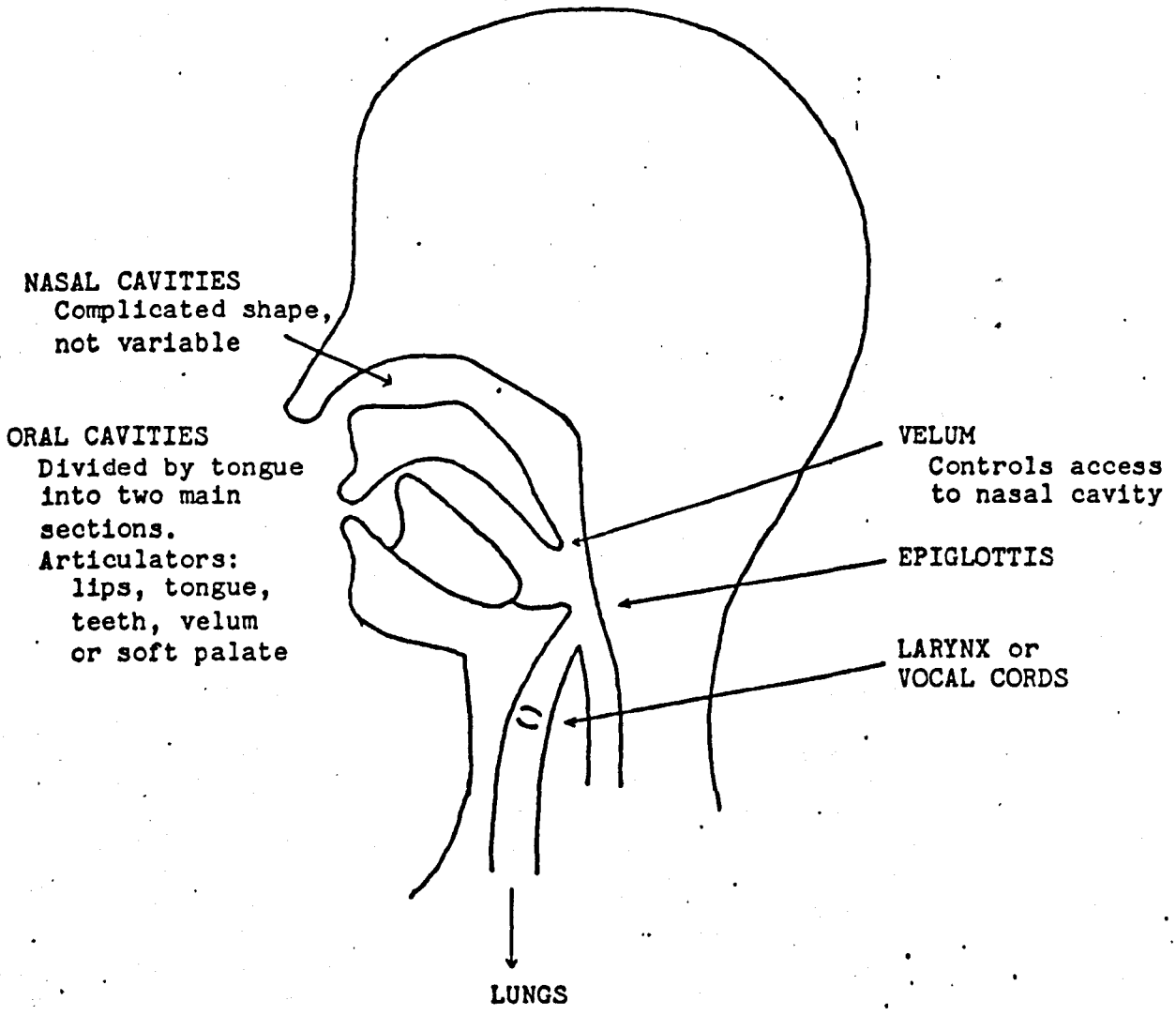
Speech recognition systems can be classified in several different ways. Firstly, on the unit of recognition:

Isolated speech recognisers

The utterance to be recognised, which may be a word or phrase, must be clearly delimited by initial and final periods of silence. Such devices are currently available and are suitable for control and command applications and simple data entry.

Word Spotters

Word spotters are devices for the detection of key-words embedded in speech. Word spotting may be a useful operation in itself - for example, extracting commands from within a phrase - or can form the front-end of a connected word or continuous speech recogniser. As an independent task, it has received considerable attention by researchers into military applications of speech recognition, such as Sperry Univac. The constraints imposed by isolated word recognisers



When we produce voiced sounds, the larynx provides the "kick" to cause certain cavities in the head to resonate. The periodicy of the larynx determines the pitch of the voice; it is independent of the phoneme (meaningful unit of sound) we are producing. The mean value for a male speaker is 125 Hz, for a female 180 Hz.

The nasal passages consist of a considerable number of small cavities of differing shapes and sizes. These make a very broad band, low resonance system (low Q) which does little more than enhance the fundamental (larynx) period.

The mouth cavities, on the other hand, are nominally two in number and their shapes and sizes are under complete control by the lips and tongue. These resonant cavities are relatively narrow band and of high resonance (high Q). These resonances influence the waveform shape considerably, depressing the fundamental component.

We produce vowels simply by controlling the shape and size of the mouth cavities, primarily by positioning the tongue. Since one mechanism controls both resonant cavities, the frequencies are inter-dependent. The two resonances are referred to as the 1st and 2nd formants and their values provide a good measure of the vowel sound, i.e. position of the tongue.

Unvoiced fricatives such as S in "Sit", SH in "SHore" and F in "Forget" are set up by positioning the tongue and lips, and air is exhaled while leaving the larynx static and open. The effect is of a cavity driven by an incoherent source. This imparts a bandwidth limitation on an otherwise random noise.

"Stops" are produced by a rapid change; For example, a rapid release of the lips leading into the vowel "EE" produces the voiced stop "B". The same release but with no voicing produces the unvoiced stop "P".

Whereas lips and tongue can stop or constrain air from coming out of the mouth, there are other pieces of apparatus that can stop air entering the mouth cavities. These are the epiglottis and soft palate. For example they are used to produce the G in "Gate" and K in "Kettle".

Figure 2.1: The human vocal mechanism.

are often too strict to make systems feasible for use in military environments, where the operator is often under a high workload and hence unable to conform to strict protocols.

Connected word recognisers

This allows the use of free flowing speech to input a group of words, but the user is constrained to use only words within the predefined vocabulary, and the group must be clearly delimited by initial and final periods of silence. Devices of this type are commercially available for up to five connected words. Laboratory feasibility has been demonstrated for larger utterances.

Continuous speech recogniser

This accepts natural, free-flowing speech, with no constraints forced on the speaker by the recognition system. An example of constrained speech is where the speaker is required to introduce gaps between utterances, eg "7gap9gap2gap1" instead of "7921". A recogniser capable of recognising continuous speech will be able to recognise words in its vocabulary when these words are spoken in a continuous flow. For example, if the vocabulary is the numerals "0", "1", ..."9", then the recogniser can be used to evaluate any number - "132" or "92638461" - even when embedded in other phrases such as "the answer is 9". Techniques exist but devices are not yet commercially available.

Secondly, the use by different speakers:

Speaker-dependent recogniser

A recogniser requiring training on individual speakers and using the appropriate "training set" for each speaker. Training consists of the user saying the words, once or several times. This is the commonest type of system at present.

Speaker-independent recogniser

This can recognise items even though said by different speakers. Some devices of this type are available now [Moshier 1979]. Performance is less good than for speaker-dependent systems and input is therefore less elaborate.

At present, practical recognisers work in a very restricted context of utterances. A system going beyond this is called a

Speech Understanding System

> A system which is capable of recognising sentences of natural or natural-seeming form and analysing them to extract their meaning. Still a research topic, though considerable progress has been made. Also used to describe systems which use semantics and grammatical rules to assist in the recognition of spoken words.

The quality of speech that can be accepted is independent of all these classes. Recognisers can use a close-speaking microphone (the most usual case), input from a telephone (also possible) or input from a microphone some distance from the speaker (not practical at present). Background noise can be limited to breathing noises (the minimum useful system), and can include steady or unsteady noises. Speech-like noise, or low level conversations in the background, are difficult to cope with at present.

2.4 Recognition Techniques

This section gives a more detailed account of the operation and stages of development of the different types of recognisers.

2.4.1 Isolated Speech Recognisers

Devices which are limited to recognising isolated words or phrases often work by matching the utterance with stored patterns. Pattern matching methods make no attempt to analyse speech in terms of relationships between words; each

word to be recognised is independently characterised by a pattern of features, in fact, a whole phrase can be treated as one pattern. There are several problems to be overcome when matching speech by machine:

(i) Compensating for different speakers.
(ii) Variations in the timing of the word or phrase.
(iii) Variations in stress or intonation.
(iv) Intrusive noises (ums and ahs) and missing phonemes.

For present systems, (i) is usually dealt with by "training" the recogniser, (ii) by normalising the time scale of the utterance. The effect of (iii) can be minimised by choosing speech features which do not contain information known to relate to stress or intonation (eg pitch). (iv) can prove difficult and may demand that the user learns not to make variations which are under his control.

As an example of an isolated word recogniser which uses pattern matching techniques, we describe the Threshold Technology Inc recogniser. This uses three subsystems: a preprocessor, a feature extractor and a classifier. The first two of these are implemented in hardware, the third by program in a computer, usually a DEC LSI-11 in current models.

The Threshold preprocessor performs a spectrum analysis; the feature extractor then characterises the input in terms of 32 binary features. There are five broad class features:

Vowel/Vowel like (V/VL)
Long Pause (LP) - greater than 100 mS
Short pause (SP) - less than 100 mS
Unvoiced noise-like consonants (UVNLC) - fricatives and unvoiced stops
Burst - sudden energy increase

There are also 27 phonetic features based on spectrum information.

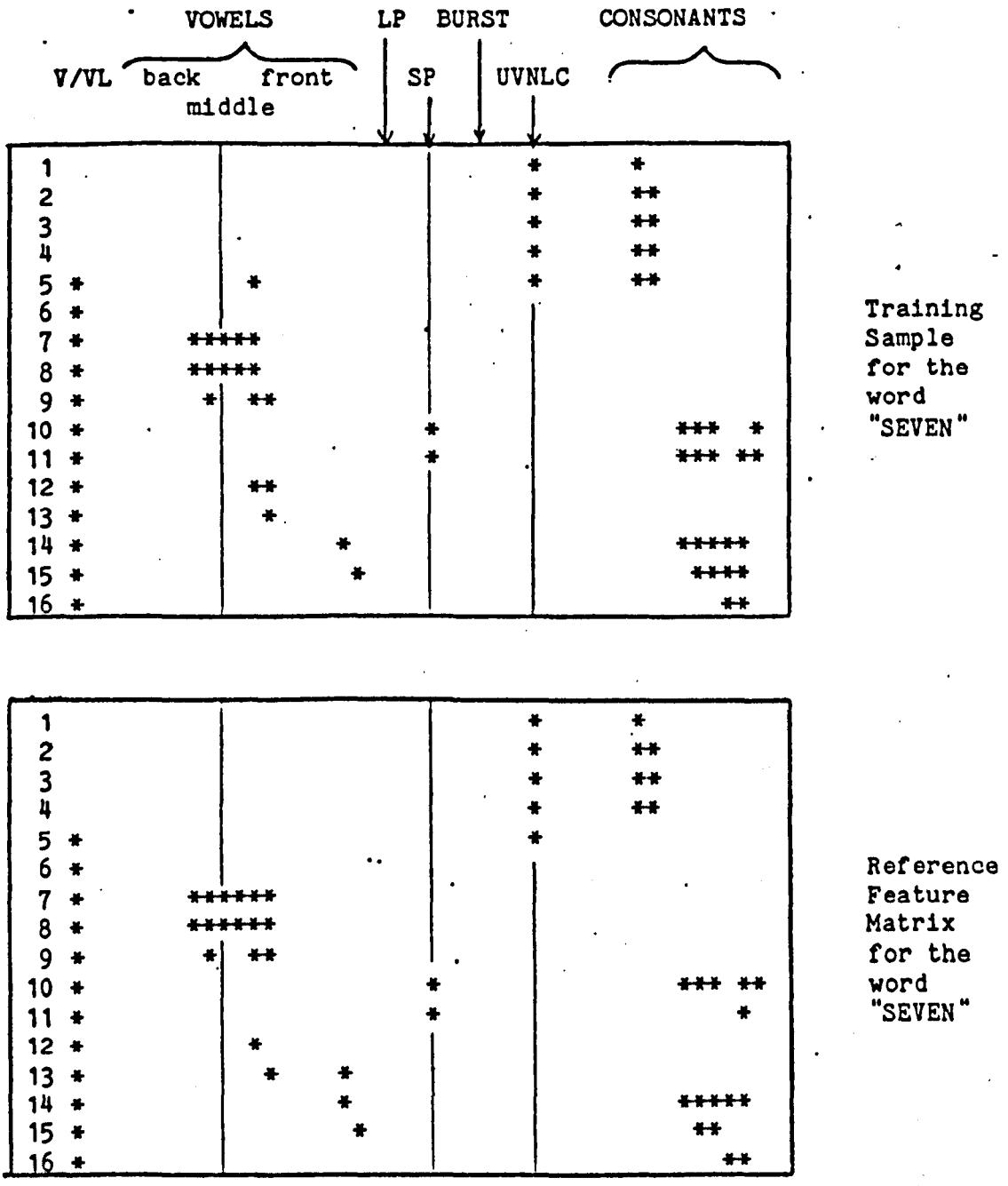


Figure 2.2: Patterns for the Word "seven" as used in the Threshold Technology recognisers.

Features and their time of occurrence are stored in a short-term memory until the end of the utterance is detected. The whole utterance is then rescaled

into a normalised timescale of 16 segments. This gives a normalised pattern of 512 bits (32 features in 16 segments).

In the training mode the speaker repeats each word, typically five times, and a reference pattern for this word/speaker combination is then stored. In operation, the pattern for the spoken word is compared with the stored patterns for all the words currently valid, using pattern recognition logic. The word producing the highest overall match is selected and the system decides whether the match is good enough to represent recognition. The time normalisation and pattern matching are carried out by software in the minicomputer.

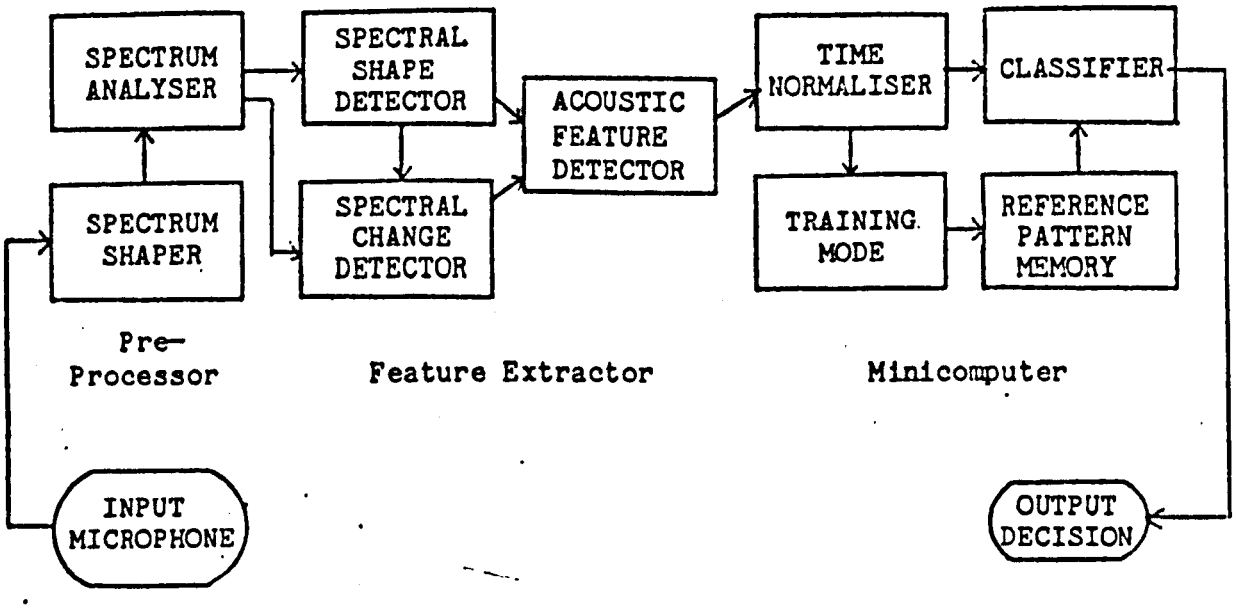


Figure 2.3: Schematic diagram of the Threshold Recogniser.

Each pattern consists of a vector of features for each instant in time (figure 2.2). A speaker does not repeat an utterance with exactly the same timing, so the timescale must be normalised before comparing patterns. Isolated word recognisers originally used a linear rescaling, but better results are

obtained if nonlinear timescale distortions are allowed, within limits. The optimum distortion for any pair of patterns can be found, and this is now commonly done using a mathematical technique called dynamic programming.

The Threshold system has been described as the typical isolated word recogniser. Threshold is currently introducing systems incorporating its QUIKTALK feature which greatly reduces the required pause time between words, thus approaching the goal of connected speech. Existing systems can be upgraded since the change is essentially an improvement in the software.

2.4.2 Word Spotters

Word spotters commonly form the basis of both connected and continuous word recognisers. In some devices, like that from NEC [Sakoe 1977; Sakoe and Chiba 1978], they may work unconditionally on every word in the vocabulary, leaving a secondary process to string spotted words together. In others, the spotter is controlled by its output in a single pass.

Although they may not appear to perform a significantly more complex task, the computational effort required by a word spotter can be at least an order greater than that required by an isolated word recogniser. This is because words cannot be segmented before matching takes place, and so it is often the matching process which is used to do the segmentation. In general, the amount of matching necessary is a function of the duration of the test (unknown) utterance, the resolution of the word templates (ie number of features per template) and the number of templates in the vocabulary. Most matching techniques require a processing time roughly proportional to the square of the

number of features in the template, so that high-resolution templates can consume enormous processing power.

2.4.3 Connected Word Recognition

If a user does not have to leave a gap of 100ms or so between words, not only is a system much easier to use by untrained people, but input can be significantly faster for data items such as numbers and phrases. It has usually been assumed that a recogniser which could work for connected words would be based on analysis of the speech in terms of phonetic structure, rather than simply applying pattern matching techniques for whole words. New problems arise with connected speech:

(i) Segmentation into words is difficult, because speakers do not leave gaps between words; speech can contain gaps, but these depend on the phonetic structure, not the word boundaries. Partial elisions can occur of the form "sicseven" for "six seven"

(ii) The time normalisation is now more difficult since it extends over a much longer time.

The Nippon Electric Co has introduced a commercial product (the DP-100) which tackles these problems by word pattern matching, for phrases of up to five words. It is therefore aimed at systems with fairly short transactions, or data entry of up to five digit numbers. Each phrase must start and finish with a period of silence. The cost and complexity are significantly greater than that of isolated word recognisers at present. The device is capable of a 120 word vocabulary for connected speech (or 1000 for isolated words) and for this performance has a decision time of 250 ms.

The input voice signal is analysed in 16 frequency bands at 20 ms intervals, and this information, in a compressed form, is the basis of the patterns to be matched. The reference templates are 4000 bits per word, and are continuously updated during use. The non-uniform distortion of the time axis to find the best match is done by dynamic programming in a two level manner; first, matching is done between candidate words and all partial patterns (of permissible length for a "word", 0.2 to 1.0 seconds) in the utterance. The best matches are carried forward by a second optimisation to obtain the best word structure. This process requires considerable computational power.

2.4.4 Continuous Speech

Continuous speech can be regarded as a sequence of units called phonemes. A phoneme is a linguistic unit within a word which, if removed or replaced, can change the meaning of the word (CAT to COT for example). Most languages use about 40 different phonemes.

A phone is the term for the acoustic manifestation of a phoneme. Phones can differ widely from occasion to occasion for the same phoneme. Phones differing through systematic variations in properties are known as allophones. Speech recognition systems sometimes use "phones" which are determined statistically (by clustering techniques) from a representative sample of the speech to be recognised.

In continuous speech, words are made up of sequences of phones; the problem is to determine where the word boundaries are. Gaps of silence do not provide such boundaries. The segmentation problem becomes impossible if all

combinations of vocabulary items are tried because of the number of possible combinations. Combinations must be reduced by using syntactic information (restrictions on word combinations through grammar rules) and by using statistical information about the combinations of phones.

Much research work has been aimed at interpreting a spoken sentence in the context of a task, for example making an enquiry about a database. Therefore, continuous speech work is associated mainly with language understanding systems. This approach goes well with restrictions on the forms of input sentences.

Another long term aim is a voice-driven typewriter or automatic dictaphone for natural language. Many people think that such a system will need to substantially "understand" the spoken input, so that all current work on natural language is potentially relevant to such a development. Another view is that the dictaphone requires only a phoneme vocoder as its front end, and a phoneme-to-text orthograhic transcriber. Both these are being developed independently for other applications - the vocoder for narrow-bandwidth communication, and (at least) text to phoneme transcribers for speech output systems.

2.4.5 The ARPA Project on Speech Understanding Systems

The Advanced Research Projects Agency (ARPA) of the US Department of Defense sponsored a five-year programme on Speech Understanding Systems (SUS) from 1971-76. Funding was set at $3 million per year, and the initial contractors

were Bolt Beranek and Newman (BBN), Carnegie-Mellon University (CMU), MIT Lincoln Laboratory, Stanford Research Institute and System Development Corporation. Four subcontractors were Haskins Laboratories, University of Michigan (later transferred to University of California at Berkeley), Speech Communications Research Laboratory (SCRL) and Sperry Univac. Disciplines represented included acoustics, signal processing, phonetics, phonology, linguistics, artificial intelligence, computer programming, system architecture and engineering.

The essence of the project was to demonstrate the understanding of continuous speech, in the sense of correctly interpreting sentences in the context of a task. Actual systems were to be constructed and demonstrated, and this was achieved by BBN [Woods et al 1976; Wolf 1977], CMU [Lowerre 1976; CMU 1976, 1977; Hayes-Roth and Lesser 1977; Erman et al 1976; Reddy et al 1976] and jointly by SDC/SRI [Silva 1975; Walker 1976]. Significant work was done during the project on network representation of knowledge sources, on linear predictive coding, and on the parsing of ambiguous and incomplete sentences. The project was generally considered a success [Medress et al 1978; Lea 1979]. There is an alternative view, which points to the very limited transfer of technology from the study to industry. The table shows the target task and the performance of the CMU system HARPY (actually the least ambitious), which most closely approached it [CMU 1977].

The fact that performance was not achieved in real-time is less significant than the fact that a very artificial syntax had to be used. The other two systems achieved less success (BBN 70% semantic accuracy on a small test set; SDC 30% semantic accuracy on an ample test set), but this is offset by the

fact that they had much greater linguistic generality. CMU also have an ultimate aim of great generality.

	TARGET	HARPY
Connected speech	Yes	Yes
Number of speakers	Many	5
(Cooperative speakers of the General American Dialect)		
Environment	Quiet room	Computer terminal room
Input	Good microphone	Ordinary microphone
Training of system to each speaker	Slight	Considerable (20-30 utterances)
Training of speakers	Adaptation	None
Vocabulary	1000 words	1011 words
Syntax and Task	Artificial	Very artificial (Branching factor 10)
Speed	A few times real time	80 times real time using .35 mips PDP10 and 256 K 36 bit words
Errors	Less than 10% semantic error	9% sentence error (5% semantic error)

Figure 2.4: Results from the ARPA SUS Project 1976.

Since the completion of the project, SCRL has been retained to look at possible further work for ARPA support, but no new projects in speech understanding have been funded. CMU does not rely on external funding for its work, and it continues to be an important centre for speech recognition research, under Dr Raj Reddy. Further work on HARPY is described in [Bisiani 1979].

2.4.6 Work at IBM

The work at IBM merits discussion in its own right both because of its ambitiousness and the sheer effort that is applied to the problem. IBM was not a member of the ARPA study, and the IBM work at Thomas J Watson Research Laboratories, Yorktown Heights, has aimed at the "voice-driven typewriter" rather than at a "speech understanding" system.

Recognition is divided between an acoustic processor and a linguistic decoder. IBM workers have used several different acoustic processors in experiments on substantially the same tasks, using various languages as stages towards natural language. The New Raleigh Language is an artificial language with a 250-word vocabulary, defined by a state transition table with sentences averaging eight words in length. Several of the acoustic processors at IBM have now demonstrated 100% recognition of a set of 125 sentences in this language, though not in real-time.

Current work uses the Laser-1000 language. From a corpus of 1.8 million words from the literature of laser patents, all those sentences which consisted entirely of the 1000 most common words of the corpus were selected. These sentences comprise the Laser-1000 language; it has a vocabulary of 1000 words, much higher acoustic complexity than the New Raleigh Language, and an average sentence length of 24 words. A typical sentence is:

> "Some of the light, therefore, passes through the surface while the reflected portion is reduced accordingly."

90% correct word recognition has been achieved on 50 test sentences (with a

single, trained speaker) based on 900 training sentences recorded several years before. An IBM 370/168 computer was used, and recognition took 360 times the time of the speech itself [Bahl 1979].

Various acoustic processors have been used by IBM. The acoustic processor MAP [Dixon and Silverman 1977] has beeh under development since 1967. It is a segmenting processor, using complex heuristics to produce strings of phoneme level output. Processors developed since 1973 include CSAP (CentiSecond Acoustic Processor) in various forms; WBAP (a Word Based Acoustic Processor which avoids the use of phones and matches words directly) and a processor named HEAR [Baker 1979]. These processors, and subsequent linguistic decoders, are based on a statistical model of speech, applied at all levels [Jelinek 1975]. For example the phones, which the acoustic processor is trying to generate from the input acoustic parameters, have statistical transition probabilities which can be observed from the corpus or a training subset. These probabilities, as well as the observed acoustic parameters, are taken into account in classifying segments into phones.

In the same way, a word can be made up of phones in different ways, depending on the speaker, the context etc. The word can be represented by a graph showing the possible sequences, and transition probabilities can be attached at each point of choice. These graphs are set up by hand, but the probabilites can be refined from measurements on the corpus. When the linguistic decoder is evaluating the likelihood of a sentence, given input from the acoustic processor (as phones), the likelihood is composed of an acoustic part and a language model part. Decoding uses a tree search technique, following forward the most promising continuations to try to keep

within the maximum size of memory available for partial results.

Laser-1000 is a natural language, and the IBM workers think that natural language grammars are not well enough defined to be useful in linguistic decoding. For each pair of words the system has an estimate of the likelihood of the pair being followed by each of the 1000 words. This involves large amounts of store, but especially an extensive preprocessing of "representative" speech. Even after counting 1.5 million of the 1.8 million words in the corpus, 23% of the trigrams in the test sentences chosen from the excluded material had not appeared in the 1.5 million word base. What would happen if the text were from a completely different source, newspaper stories for example, can be imagined. This is likely to be the main problem in extending the statistical approach.

A further problem with automatic speech recognition is that each utterance may be realised phonetically in many ways. Examples have been given of the phrase "either person" [Cohen 1975] which can be produced in over a thousand distinguishable ways, even when disregarding pitch and stress. Obviously all these productions cannot be stored; fortunately they can be related to one another by defining in effect a phonological grammar. The grammar in particular contains rules of coarticulation - how one sound influences a neighbouring one, and how words are joined together in continuous speech. IBM has done considerable work in this area and can provide an impressive demonstration of systematic variations applied to speech output.

The workers at IBM are constantly changing the numbers and identifying features of acoustic parameters, phones and techniques used. The most

advanced experiments have concentrated on input speech of high quality, with processors trained on the speaker. Their best techniques can now correctly recognise more than 80% of the words in a sentence on average. Most of the errors are for small "function words" which are not pronounced distinctly.

2.5 Conclusions

Clearly, there has been and still is much interest in speech recognition. This is reflected in the wealth of literature that exists and in the large amount of funding available to both commercial and academic projects. Many of the academic researchers have tended to take a "critical-path" approach to the speech understanding problem, concentrating their efforts on the discipline that, at the time, gives the greatest improvement in performance. The result has been that aspects such as syntax and semantics have been exploited at the expense of unconditional speech recognition work. The development of task-independent recognisers has been largely in the hands of two or three commercial companies who have been unable to produce both a cheap and effective continuous speech recognition device. This has provided significant motivation to the evolution of WISPA, since one of its design aims has been that it should be commercially exploitable at low cost.

CHAPTER 3

The WISPA System for the Automatic Recognition of Continuous Speech.

3.1 Introduction

This chapter describes the design and implementation of the WISPA Speech Recognition System. The WISPA system is designed as a tool with which one may experiment with various training/recognition strategies, rather than a specific piece of hardware or software designed to implement a specific strategy. In making this distinction, it becomes easier to separate the strategy from the implementation details, and thus gain a better understanding of the way in which the strategy succeeds or fails. The implementation of the strategy is governed by the structure of the system.

The structure is established by identifying those tasks which are specific to the problem of speech recognition in general. Here, we ask what level of speech recognition we require or, more formally, what we wish to be the eventual intrinsic capabilities of the system. We may design-in capabilities that we may yet not have the strategy to realise. For example, a system may have capability to process continuous speech, but may model an isolated word recogniser.

The design of the recogniser, then, requires the specification of:

1) System Capability
2) System Structure
3) System Strategy

The objective is to allow for the maximum variation in strategy by choosing a structure that reflects a very general capability.

3.2 System Capability

The WISPA system capability is determined mainly by exploitation potential, previous experience with earlier systems, and level of research effort available. The eventual goal is a system capable of:

1. Acoustical recognition of continuously spoken words.
2. Trainability and adaptability to many speakers using utterances of isolated words.

3.2.1 Acoustical Recognition

The term acoustical is used here to denote that the recognition is performed using no contextual information relating to task. A more definitive statement would be that acoustical recognition is performed using no more information than is contained in:

(i) the acoustical waveform

(ii) the phonetics of human speech

(iii) the vocabulary

The use of the acoustic waveform and the vocabulary are obvious prerequisites to any speech recogniser. Phonetic information may perform an important role in acoustical recognition, although the use of it is not demanded. Indeed, many recognisers [eg Bridle and Brown 1978; Sakoe 1977; Martin 1975] adopt a pure pattern matching approach, and make no special use of the fact that the source data is speech.

There are, of course, other sources of information - often termed knowledge sources [Reddy et al 1976] - such as semantics and syntax which can be applied to continuous speech recognition. Semantics is the term applied to the meaning [Lyons 1968] of the utterance, while syntax applies to its construction or grammar [Bloomfield 1935].

Apart from the semantic and syntactic implications of a finite vocabulary (or subset thereof), there is no mechanism for specifying a set of syntactic or semantic rules in such an acoustical recogniser. A purely acoustical recogniser is, nonetheless, still extremely important since:

1. There is a whole class of applications where these knowledge sources are restricted or unavailable.

2. Some high-integrity applications (for examples see Beek et al 1977] may be offered semantically or syntactically incorrect utterances, which must be recognised as such.

This is not to say that other knowledge sources are unimportant, merely that if relevant, they may be applied at a higher level. Further, it is seems clear from recent studies [Lea 1979], that it is the area of acoustical recognition of continuous speech that is least advanced in the field of speech recognition. Better performance at this level (in speed, accuracy and cost) would yield a considerably wider field of application.

3.2.2 Trainability and Adaptability

The requirement of the WISPA system is that it should be capable of building up vocabularies of words from their utterances in isolation. Furthermore, it must be possible to combine several utterances of the same word into a single word entry. This is the training process. It is not necessary to specify a further multi-speaker capability since there is no inherent constraint that either training or recognition utterances must be made by the same subject.

An additional requirement is that it should be possible to use utterances generated at recognition-time as training utterances. This is the process of adaptation.

The degree of success of the training and adaptation processes depends on the system strategy, but what is important is that the system has the capability to implement the strategy.

3.3 System Structure

The approach for choosing the structure for the WISPA system is a "top-down" one. Starting with a very inspecific scheme comprising several functional modules, we break down each module until it is sufficiently well defined to be constructable.

The overall structure may itself be divided into two sections: the logical structure concerned with the organisation of the modules with respect to each other and the data structure dealing with definition of the interfaces between modules and the system data-bases.

3.3.1 The Logical Structure

The overall system may first be considered as two distinct processes: training and recognition. These are represented in their most basic forms in figures 3.1 and 3.2.

Although some processes are functionally equivalent (ie do the same thing) for both training and recognition - this is true of preprocessing, matching and merging - the output and execution of each of these modules may vary according to whether the system is training or recognising. Structurally, it is more elegant to share the functionally equivalent modules between both processes, and control the operation of the common modules by means of an executive.

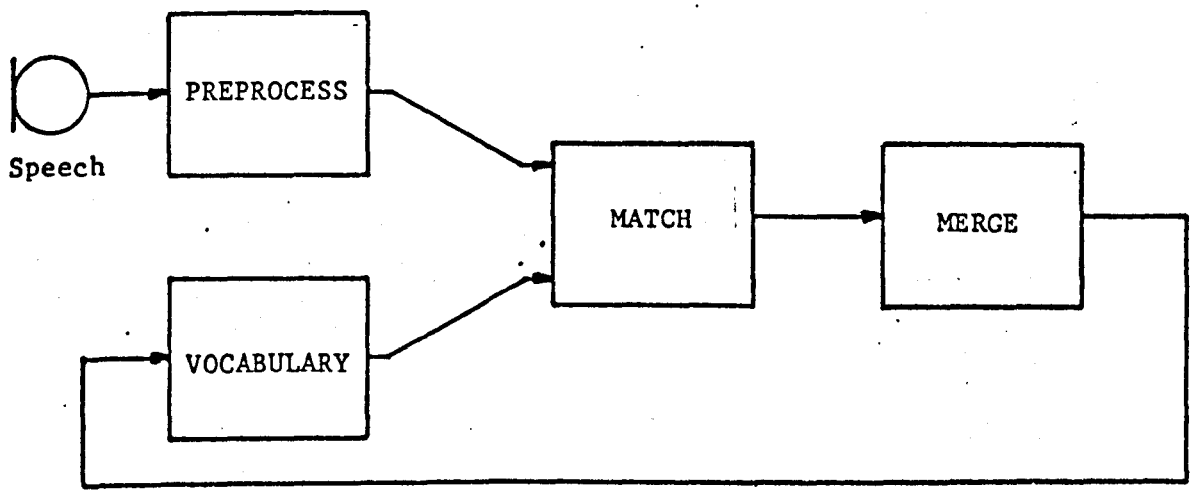


Figure 3.1: The training process.

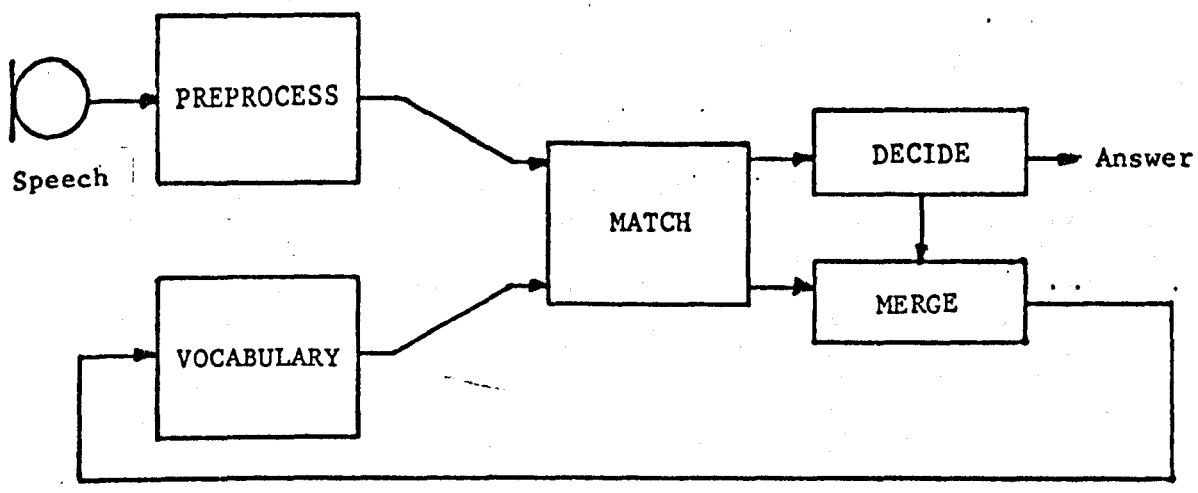


Figure 3.2: The recognition process.

3.3.1.1 Preprocessing

The primary purpose of preprocessing speech is to convert it into a form that may be conveniently analysed. In this respect, a microphone could conceivably be the only preprocessing element in the system - converting the speech from a pressure wave into an electrical signal. A further refinement might be to convert the analogue signal into a digital form. However, to seriously contemplate the operation of a practical speech recognition device in anything approaching real-time, one would expect the preprocessor to encode the speech into some more concise form.

Why should this be the case? Boddie [1977] illustrates the coding inefficiency of the direct comparison of speech waveforms by the use of a simple example. He says:

> "The following sentence:
>
> "The objective of this section is to develop these techniques"
>
> contains 60 characters and can be spoken in about 3 seconds. Using a 5 bit code for alphabetic characters, real-time transmission at speech rates would require only 100 bps. If the utterance were transmitted with a string of 42 phonemes coded with 6 bits each, then 84 bps would be necessary. However, if the speech signal itself were band-limited at 3 kHz, sampled at 6 kHz, and quantised to 256 levels or 8 bits, real-time transmissions would require a rate of 48,000 bps. If one could analyse this digitised speech in real-time, it would have to process one 8-bit speech sample every 160 us. A microcomputer with a cycle time of 500 ns would only have 320 cycles between speech samples. That does not allow for very many instructions to match the data with the reference waveforms. The direct storage of the waveform would require 6,000 bytes of memory for every second of speech. It should be clear that it is desirable to have a method of rapidly extracting the useful information, thereby reducing the amount of data storage and computation time".

The decision as to which information is useful is made by the strategy, but

what is required of the preprocessing structure is a scheme that allows the speech to be encoded in any chosen way; this should encompass the most trivial case of encoding, that of a straight analogue-to-digital conversion.

The encoded speech must be passed on to the matching module in some well defined way. Since speech is sequential in time, it is reasonable that the interface between the preprocessing and matching modules should be a feature string that retains the sequential nature of the data.

3.3.1.2 Vocabulary

The vocabulary is the set of words for which recognition is defined. The term word is used throughout to indicate a unit of meaning, and as such may be part of, or many conventional words. (Note: the word, as used here, should not be regarded as a synonym for the speech "token" [Rabiner, Rosenberg and Levinson 1978], which is defined as a single word uttered by a single speaker). Each word may be stored in three parts:

1. The key - this represents the perceived meaning of the word
2. The template - this represents the way in which the word is uttered
3. The subset - this indicates the subset of the vocabulary to which the word belongs

The template would normally be stored in a form similar to the feature string generated by the preprocessor. Since each template may represent several training utterances, it is necessary to store information about the diversity of the feature strings which represent the utterances.

Although we have made clear the role of an acoustic recogniser as a device not using contextual information, we have not precluded the use of this information at a higher level. In so doing, we must ensure that the structure of the recogniser does not prevent the application of other knowledge sources to the problem. This is particularly relevant to the design of the vocabulary structure, where we may wish to isolate sets of words dynamically, according to the recognition task. This gives rise to the subset descriptor for each stored word.

The vocabulary is simply a list of templates. There is obviously work to be done in the selection and manipulation of these templates for both recognition and training, and this requires a separate process to undertake the task, a vocabulary manager.

3.3.1.3 Matching

Matching is the process of comparing the feature string from the preprocessor with the feature string of the template, and generating two outputs: the distance and the trace. The distance is a measure of similarity between the strings, and the trace is a function describing the transformation from one string to another. The trace may be thought of as the sequence of editing operations that converts one string (from the preprocessor) into another (the template). The execution of the matching process assumes the existence of a cost function which defines the edit distance between features from the preprocessor and template.

This concept of string matching is based on the more general "String-to-String

Correction Problem". The paper by Wagner and Fischer [1974] is a definitive presentation of the problem and is the source of much of the nomenclature used here.

Within this framework, it is the strategy which determines the format of the feature string, the edit distance between features, and the matching criteria (eg match for minimum distance).

The usual sort of matching process is one that evaluates a trace giving minimum distance between two strings. As an illustration, one may consider two strings of characters, string X: <abbacd>, and string Y: <abace>, where one might wish to match for minimum distance. We may refer to string X and string Y simply as X and Y.

If we label the elements of each string thus:

X:		Y:	
	X[1] = a		Y[1] = a
	X[2] = b		Y[2] = b
	X[3] = b		Y[3] = a
	X[4] = a		Y[4] = c
	X[5] = c		Y[5] = e
	X[6] = d		

and define the cost function as:

```
COST(X[i],Y[j]) = 0     if X[i] = Y[j]
                = 1     if X[i] ≠ Y[j]
COST(null,Y[j]) = 1
COST(X[i],null) = 1
```

then a matching operation giving minimum distance might map X to Y thus:

a	->	a	COST=0
b	->	b	COST=0
b	->	<null>	COST=1
a	->	a	COST=0
c	->	c	COST=0
d	->	e	COST=1

which could be represented by:

DISTANCE(X,Y) = 2

TRACE(X,Y) = {(1,1),(2,2),(3,0),(4,3),(5,4),(6,5)}

The distance is simply the sum of the costs of each trace operation.

The trace is a set of ordered pairs representing the operations necessary to edit X into Y; so that (1,1) means X[1] is changed to Y[1] and the cost associated with the operation is zero (as one might expect since the elements are the same). The element labelled zero is the null element. If the first number of the pair is null, this implies that an element has been inserted into Y, whereas if the second number of the pair is a null, this implies that an element has been deleted from X.

In the context of speech recognition, if the characters are speech features, then the cost is a function of perceptual similarity between features.

Matching in Training - In the case of training, the matching process is normally only executed once for each training utterance - comparing one preprocessed string with one template string - returning a single distance and trace. The trace is then used by the merger to combine the template and feature strings into an updated template.

Matching in Recognition - At the very least, recognition demands execution of the matching process for every word in the vocabulary (or its subset). Furthermore, if we allow for continuous speech recognition, we must provide for matching any sub-string within the feature string, with any word template in the vocabulary. This must be an intrinsic capability of the system, indepenent of the strategy, and is an essential pre-requisite for subsequent word-spotting.

The repeated application of the matching process implies the existence of some controlling procedure, which decides which word templates to match with which sub-strings. This gives rise to the concept of an executive to control the overall recognition process.

3.3.1.4 Merging

In merging, we seek to combine a feature string with a template to form a new, updated template. Where no template exists, the feature string is simply translated from one data structure to another.

3.3.1.5 Deciding

Under the control of the executive, the decision process takes information from the matching process - a series of distance measures - and decides which words, if any, were uttered and in what order. Its output is a string of keys. Depending upon the output from the decision process, the merger may incorporate a sub-string of the feature string into the appropriate template,

thus adapting to the latest utterance. Figure 3.2 shows a control path directly from the decider to the merger, but in practice it is better accomplished via the executive.

3.3.1.6 Rationalisation

If continuous speech recognition is to operate in real-time, it is often extremely important that operations in so-called "critical paths" be executed as quickly as possible. This is particularly true of matching.

It is therefore useful to incorporate a mechanism in the structure to:

(i) perform a conversion process which can transform the the data-base into a format amenable to fast operation

(ii) where possible, execute the conversion before recognition takes place

This conversion process is termed rationalisation.

As an example of a possible rationalisation process we may consider the case where a parameter of a template comprises the average of the corresponding parameters in a series of training feature strings. This average might be stored as the sum of parameter values and the total number of training utterances, so that the average could be computed by dividing the sum by the total. If, furthermore, a recognition decision is based on whether or not this average is above a fixed threshold, then the average can be computed and a single bit used to represent the state of the average (ie above or below the the threshold) before recognition takes place. However, it cannot be stored

as just a single bit in the template, since the sum and total information are needed to update it. By the same token, we need not use up valuable processor memory and speed operating on two multiple-bit numbers at recognition time.

In developing the system structure, we allow for rationalisation of both the feature string and template. Nevertheless, we are not forced to use rationalisation in our strategy simply by allowing for it in the structure. In such a case rationalisation is a null process (in the same way as preprocessing can be a null process).

3.3.1.7 The Executive

The need for an executive as an overall management process has evolved in looking at the modules of basic training and recognition system. In recognition, the executive performs the role of controlling the matcher and merger, as well as responding to the decider and selecting words in the vocabulary. In training, it performs a similar but less demanding role.

In addition to the speech processing tasks, there are tasks associated with user-interaction, speech and vocabulary storage, and performance monitoring, all of which are independent of strategy.

The role of an executive is a common feature in many speech recognition systems, two examples of which are Rover in the Hearsay System [Reddy et al,1976] and the executive in the Stanford Research Institute Speech Understanding System [Walker, Paxton, et al 1977].

3.3.1.8 The Derived Logical Structure

Having looked at the functional requirements of each module, we are now in a position to define the logical structure of a complete recognition/training system. This is shown in figure 3.3.

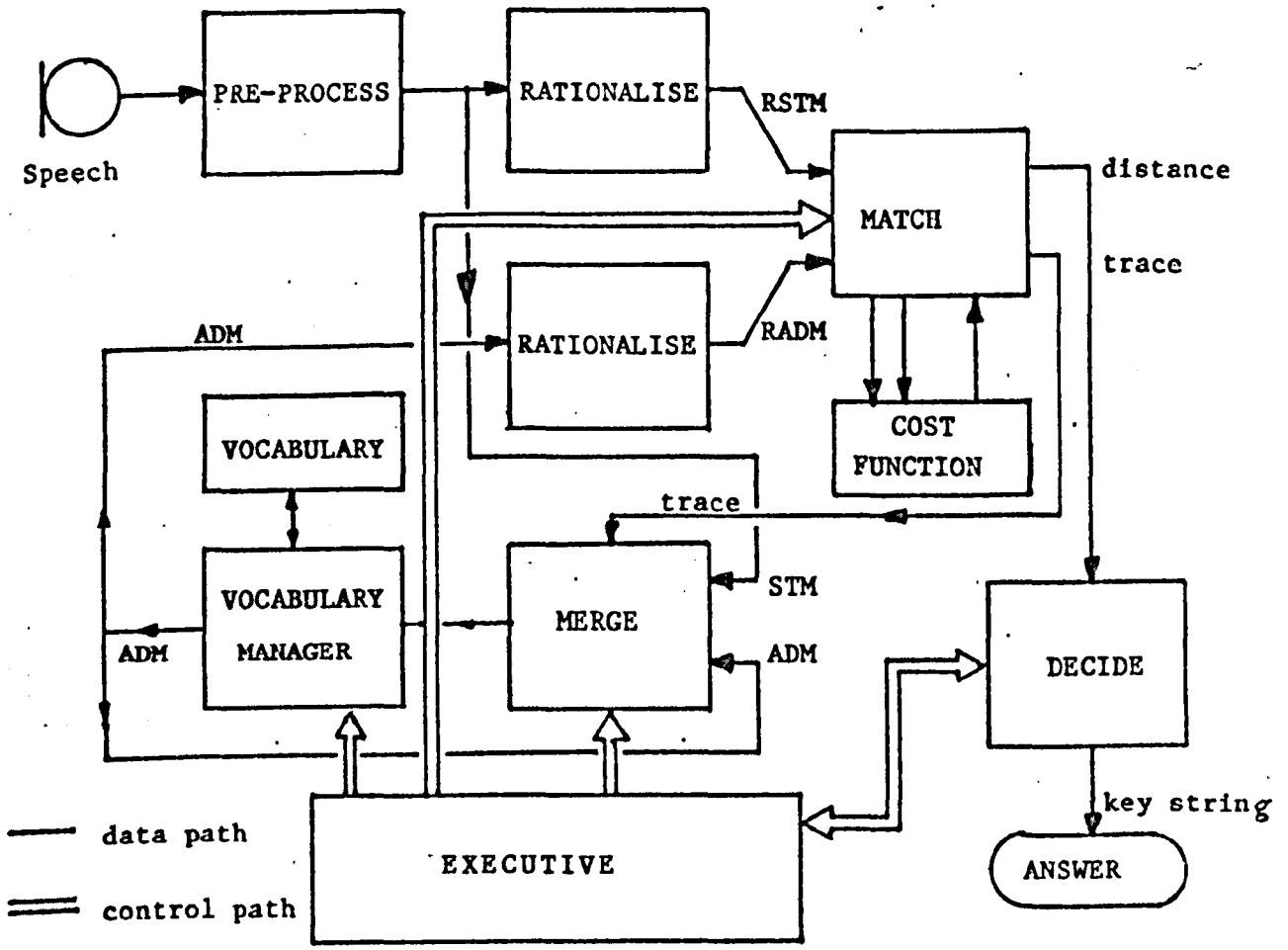


Figure 3.3: The logical structure.

We have managed to derive this structure without once considering the actual way in which the speech is processed, matched and recognised. There has been

mention of feature strings and templates, but no specific description of their composition. The question now is whether we can go further in defining the structure without regard for the strategy, for even if we defer the specification of the modules, we must first establish the interfaces between them.

3.3.2 The Data Structure

One approach to the design of the data structure is to make the interfaces between modules as general as possible. We may say, for example, that the feature string is a linked-list, with each element comprising a data section of n bits. The size and content of the data section (ie the function of the bits) can then be defined for the strategy under investigation, and the individual modules designed accordingly.

The drawback of this approach is that while it retains maximum generality, it does not encourage the independence of modules. If, say, we modify the preprocessing technique such that it results in a change of the interface to the rationaliser, then we must also modify the rationaliser.

It is at this point that we must begin to consider the strategy; if we can establish the data structure for a given strategy, we can then test and amend it in the light of its suitability to a class of strategies. In addition, we must consider the machine on which the structure is implemented. We have not assumed in any of the forgoing that the speech recognition process be performed necessarily on a digital computer.

To summarise, the overall structure may be considered in two levels of generalisation; the logical structure generalises the system to the modelling of speech recognition and training; the data structure restricts the generalisation to model to a class of strategies, in the case of WISPA, a strategy employing phonetic analysis.

Let us now consider the detailed data structure. From the logical structure in figure 3.3, there are seven module interfaces that may be specified. These are shown in table 3.1.

INTERFACE	NAME	CODE
Preprocessor/ STM Rationaliser	Short-Term Memory	STM
STM Rationaliser/ Matcher	Rationalised Short-Term Memory	RSTM
Merger/Vocabulary Manager	Admissible Sequence	ADM
Vocabulary Manager/ ADM Rationaliser	Admissible Sequence	ADM
ADM Rationaliser/ Matcher	Rationalised Admissible Sequence	RADM
Vocabulary Manager/ Backing Store	Vocabulary	VOCAB
Matcher/Decider	Match Matrix	MM
Matcher/Merger	Trace Vector	TV
Decider/Human	Key String	KS

Table 3.1: WISPA module interfaces (see figure 3.3).

Each interface is assigned a name and code, and assumed to exist in the memory

of a digital computer, whose basic unit of storage is the 8-bit byte. We can now investigate the interfaces by name to establish a data structure for the logical structure and hence completely define the system structure as a whole.

3.3.2.1 Short-Term Memory (STM)

The short-term memory of the preprocessor is analogous to the so-called short-term memory of the human [Newell and Simon 1967]. It is a temporary store in which data is held for fairly immediate processing, ie matching or incorporation into the vocabulary.

The purpose of the preprocessor is to encode the analogue speech with the aim of removing information in the waveform that is not thought to be relevant to its meaning. The method of encoding adopted for the WISPA preprocessor is to convert the speech into a string of phonetic features. It is the STM that stores the phonetic features.

Each phonetic feature, is described by a phonetic class (abbreviated to class) and the class is further described by a phonetic type (abbreviated to type). With each type there is an associated weight. Phonetic features may contain up to 8 classes, and each class may contain up to 16 types, subject to the constraint that the total number of types must not exceed 16. The weight may take on up to 256 values.

Types are numbered such that their numbers indicate the class to which they belong, also, each class is a partition of types. Figure 3.4 shows a typical

arrangement with four separate classes.

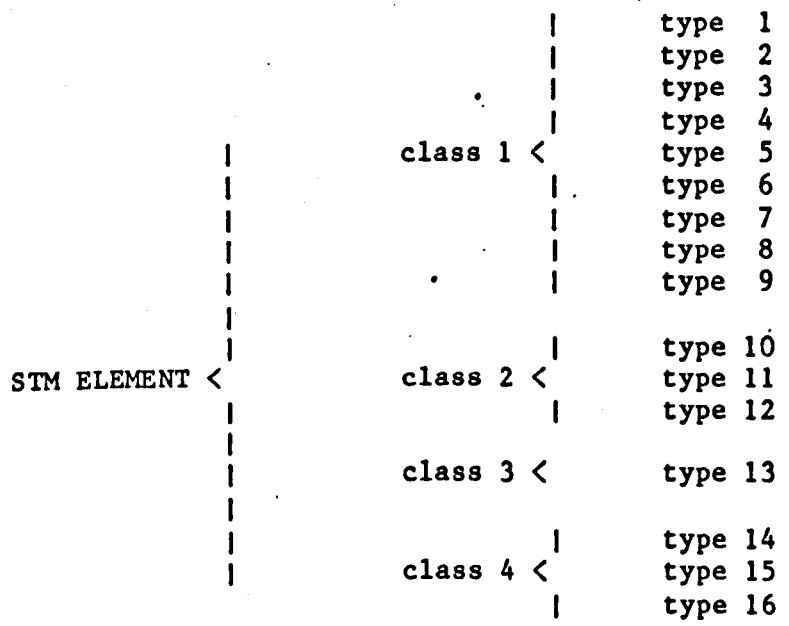


Figure 3.4: Typical class partitions in an STM element.

While each type is internally uniquely numbered, it can be referred to unambiguously by its "type within class". In this case, types are specified as a "class-type" pair, and type numbering is only unique within a class; so that type 10, for example might be referred to as class 2, type 1.

Each STM element is stored in a sequence of contiguous bytes. The number of bytes is dependent on the number of types encountered upon the generation of the element. Figure 3.5 shows this arrangement.

<class bitmap>	Notes:	Bits set in class code according to classes present.
<byte count>		
<type code 1>		
<weight 1>		
<type code 2>		Byte count equals 2 times number of types present. ie 2n
<weight 2>		
.		
.		
.		
<type code n>		
<weight n>		

Figure 3.5: Storage arrangement for a STM element.

As an example, if we assume the classes are partitioned according to figure 3.4, then a typical STM element might be figure 3.6.

BINARY	DECIMAL	MEANING
00000101	5	classes 1 and 3
00000110	6	byte count = 6
00000010	2	type 2 (class 1)
00000101	5	weight = 5
00000111	7	type 7 (class 1)
00000011	3	weight = 3
00001011	13	type 13 (class 3)
00000111	7	weight = 7

Figure 3.6: Typical STM element.

Within this structure, the meaning of the phonetic class and type is dependent upon the strategy. The weight value associated with each listed type, contains time information in the WISPA system but can be used for any parameter within the accuracy range of a byte.

3.3.2.2 Rationalised Short-Term Memory (RSTM)

The short-term memory is rationalised for three main reasons:

1. To make each element occupy a fixed amount of memory, so that elements can be directly addressed (rather than indirectly).
2. To reduce the resolution of the weight information.
3. To expedite the extraction and processing of type information.

This is accomplished by storing each RSTM element in 8 contiguous bytes. The first byte is a copy of the class byte in the corresponding STM element, and the second byte is unused. The subsequent 3 pairs of bytes each hold a bitmap representation of those types which fall into one of three grades. In the bitmap representation, a bit is allocated for each type, and set or cleared according to whether its weight falls inside a fixed range for the grade. The first bitmap relates to grade 1 threshold, the second to grade 2 and the third to grade 3. Figure 3.7 shows the RSTM format.

```
< res  >< class >
< grade 1 bitmap >
< grade 2 bitmap >
< grade 3 bitmap >
```

Figure 3.7: RSTM element format.

If we consider again the example in figure 3.6, we can see in figure 3.8 how this STM element is represented as an RSTM element. In this example we have selected a weight range of 6-7 for grade 1, 4-7 for grade 2 and 1-7 for grade 3.

```
00000000 00000101        classes 1 and 3
00010000 00000000        grade 1: type 13
00010000 00000010        grade 2: types 2,13
00010000 01000010        grade 3: types 2,7,13
```

Figure 3.8: RSTM element corresponding to STM element in figure 3.6.

The criteria for grade selection are, again, strategy dependent. In the example in figure 3.8, grade n is a subset of grade n+1, although this does not have to be the case.

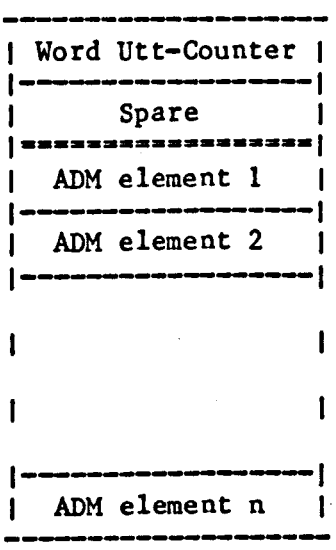
3.3.2.3 Admissible Sequence (ADM)

The admissible sequence is the internal structure for word templates. Its role is to represent, in a single sequence, one or more STMs that are generated in response to a vocabulary word uttered in isolation.

The ADM is constructed by initially copying the STM of the first training utterance. As additional training utterances are received, their STMs are matched against the existing ADM and a mapping function between the ADM and STM is established, this is the trace. If there is a direct correspondence between an ADM element and an STM element, this is recorded in a special data area in the ADM element. If an ADM element does not map to any STM element, then this element has been deleted (see 3.3.1.3) and remains unaltered by the new utterance. If an STM element does not map to any ADM element, then this element has been inserted (see 3.3.1.3), and a new ADM element must be created.

The format of the ADM is nominally the same as that of the STM, so that there

is a direct correspondence between ADM elements and STM elements. There is additional information which records variation in training sequences. The overall ADM comprises two computer words relating to the training of word - these are (i) the total number of training utterances (the word utterance counter) and (ii) a spare, extra information word - plus a contiguous string of ADM elements, thus:



Each ADM element contains information falling into one of two catagories:

1. The head: information relating to the whole ADM element.
2. The tail: information relating to each type in the ADM element.

The first category, the head, contains four items: the type-count, the class bitmap, the class utterance-counter, which is the number of contributing utterances used during training, and finally, the class weight-sum, which is derived by summing the maximum type weight of each contributing STM element.

In the second category, the tail, corresponding values for type number, type utterance-counter and type weight-sum are stored, referring only to specific phonetic types. In addition, an age value is recorded, giving the value of the word utterance-counter at the time the element is updated. Figure 3.9 shows the internal arrangement of an ADM element. Each element occupies a contiguous sequence of 16-bit words.

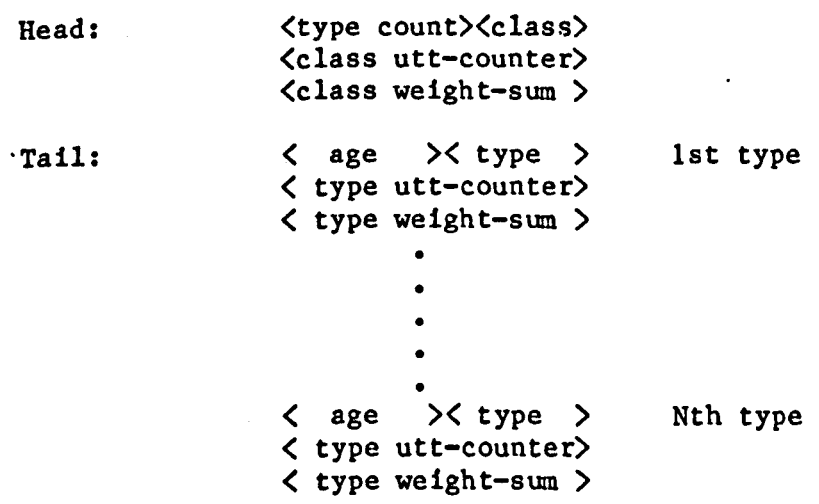


Figure 3.9: Storage arrangement for an ADM element.

3.3.2.4 Rationalised Admissible Sequence (RADM)

The admissible sequence is rationalised for basically the same reasons as the STM (see 3.3.2.2). An extra benefit of ADM rationalisation is the reduction in storage space required. Although it is not a design criterion, it is useful to keep the storage overhead to a minimum to allow for large resident vocabularies. The format of the RADM is identical to that of the RSTM. This results in the cost function being able to use the same addressing scheme for both RSTM and RADM elements. Figure 3.10 shows the RADM element as a four

word block, with class and graded-type bitmaps.

```
< res  >< class >
< grade 1 bitmap >
< grade 2 bitmap >
< grade 3 bitmap >
```

Figure 3.10: RADM element format.

The way in which types are graded for inclusion in each bitmap is, again strategy dependent. However, it is likely that use is made of the type weight-sum and type probability (obtained from the type utt-counter).

3.3.2.5 The Vocabulary

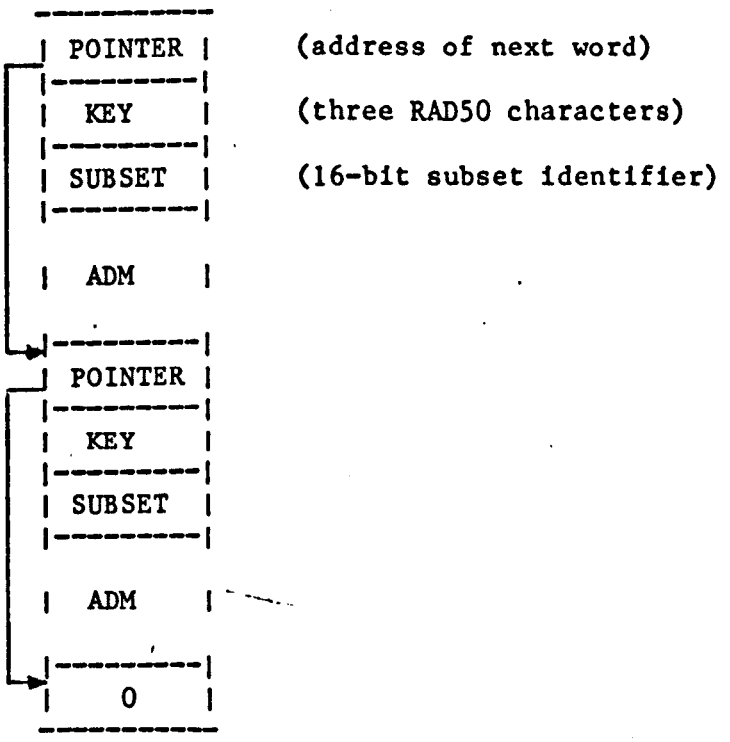


Figure 3.11: The vocabulary structure.

The vocabulary is a linked list of words, where each list element comprises a

head and a tail. The head contains a pointer to the next word, the word key and the word subset. The tail contains the ADM.

The pointers are addresses relative to the first word entry. A zero pointer indicates that there are no more words. Figure 3.11 shows the vocabulary structure.

3.3.2.6 The Match Matrix

The function of the ideal matcher is to produce a series of continuous, time-varying outputs (one for each word), representing the probability that a given word exists at a given time in the test utterance. Figure 3.12 shows how the outputs from such a matcher can be represented graphically. As well as the probability, it is useful to have a measure of the section of test utterance that gives rise to the probability; that is, which portion of the the utterance was used in determining the probability.

The more accurate the matching, the more trivial is the task of the decider. Indeed, a perfect matcher should produce binary outputs (probabilities of 0 or 1), leaving the decider to string sequentially recognised keys (and perhaps resolve word subset conflicts).

In practice, it is not possible to produce continuously varying outputs from a discrete input source. The best we can do is maintain the resolution of input source (ie the RSTM) and generate a (reciprocal) probability measure, called the fit for every word, at every new RSTM element. Associated with each fit, is the length of RSTM that gives rise to that fit. (Note that the term fit

rather than probability is used here to indicate that the measure is un-normalised, and that it is not valid to compare word-fits with one another on an absolute basis).

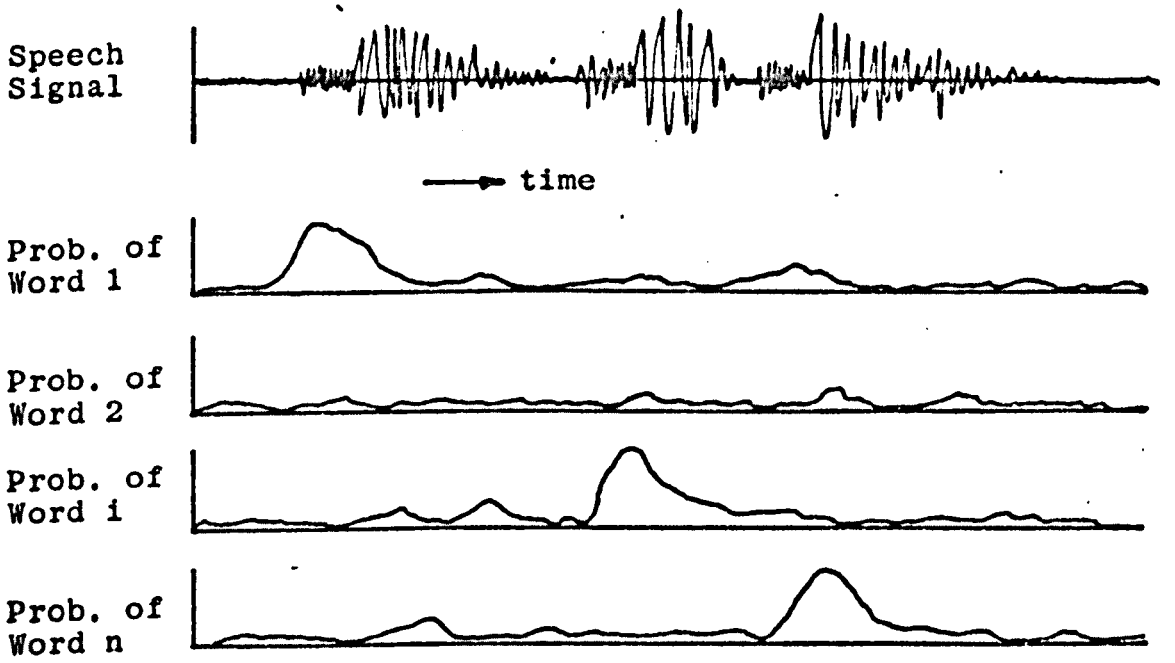


Figure 3.12: Outputs from an ideal matcher.

The upshot of this is the match matrix of fit/length pairs. The columns of the matrix represent the words of the vocabulary, and the rows represent the RSTM elements. Figure 3.13 shows a generalised match matrix for an RSTM of m elements and a vocabulary of n words.

	Word 1	Word 2	.	Word j	.	Word n
RSTM 1	F(1,1)	F(1,2)	.	F(1,j)	.	F(1,n)
	L(1,1)	L(1,2)	.	L(1,j)	.	L(1,n)
RSTM 2	F(2,1)	F(2,2)	.	F(2,j)	.	F(2,n)
	L(2,1)	L(2,2)	.	L(2,j)	.	L(2,n)
.	.	.	.	.	.	.
RSTM i	F(i,1)	F(i,2)	.	F(i,j)	.	F(i,n)
	L(i,1)	L(i,2)	.	L(i,j)	.	L(i,n)
.	.	.	.	.	.	.
RSTM m	F(m,1)	F(m,2)	.	F(m,j)	.	F(m,n)
	F(m,1)	L(m,2)	.	L(m,j)	.	L(m,n)

Figure 3.13: The generalised match matrix.

A fit value occurring at row i and column j, ie F(i,j), represents the degree to which word j matches a sequence of L(i,j) RSTM elements commencing at RSTM element i. It is, of course, possible to establish a fit value for all the lengths in the range (1,m-i), but it is a constraint of the matrix structure that only one be chosen. It is usual for the strategy to select only the length which gives the best fit (although best is not necessarily least).

3.3.2.7 The Trace Vector

The trace vector (as described in 3.3.1.3) comprises a list of ordered pairs. The data structure for this is simply a 2xN array and a scalar indicating the length, N, of the array.

3.3.2.8 The Key String

The key string, as implied by the name, is simply a string of keys of the recognised words. With each key there is an associated number indicating the level of confidence with which the word is recognised.

3.4 System Strategy

Having defined the structure of the system, we are now in a position to consider the recognition/training strategies we model on it.

All that is described actually exists (in terms of hardware and software) and is fully implemented as a system. However, as one might expect in any research program, the design and functionality of modules is constantly under review, and so the strategy described here represents only the base-line from which experiments are conducted; it is not meant to be a definitive description of the ultimate WISPA system.

3.4.1 Philosophy

We begin by discussing the design philosophy of the WISPA system in general, and continue by examining the modular system construction in detail.

There are two basic approaches to speech recognition that are adopted by many workers in the field. These can be broadly catagorised as the pattern matching approach and the linguistic approach. Both begin by band-limiting

the speech waveform to the limits required by a normal human recogniser; this turns out to be in the region of 100 Hz - 5 KHz.

The pattern matching approach requires some time/frequency transformation on the band-limited signal to be performed at regular intervals, and the utterance to be stored as a sequence of spectra. Test utterances are then compared as patterns against reference templates stored in a similar manner. Essentially, the approach is one of converting test and reference utterances into "pictures" and comparing the pictures in the same way one might compare images from a camera. In the linguistic approach, the band-limited signal is converted into a string of phonemes which may be compared by a string matching technique.

At its extreme, the pattern matching technique takes no advantage of the fact that the acoustic waveform is human speech - it might just as well be a bird twittering or a telephone ringing. Such systems are known to work extremely well in particular environments, but tend to be intolerant of speakers who have not trained the system individually. This is because the spectral pattern shape can be altered even though the phonetic content is unchanged. An example of this might be where a system is trained on a normally stressed "six" and asked to recognise a "six" where the initial and terminal "s" sounds are artificially stressed. In such an example, both versions of six have the same perceived meaning and are phonetically identical (but allophonically different), yet because the spectral shape of the utterance is drastically altered, the match between each version is poor. Intuitively, it seems wrong not to make use of linguistic rules - in this case, the ineffectiveness of stressing an "s" sound.

The linguistic approach, in contrast to the pattern matching approach, relies completely on the linguistic nature of the signal. Again at the extreme, it attempts to make a perfect phonetic transcription of the utterance for comparison with templates comprising phoneme strings. Although this seems a more elegant approach, it suffers from two major drawbacks. First, it is not easy to automatically transcribe speech in terms of phonemes (as described by the phonetician) because there is no simple relationship between a segment of speech waveform and its corresponding phoneme. Second, the phonetic description of a word varies according to whether the word is spoken in isolation or continuous speech - this is generally referred to as the coarticulation problem. In the case of continuous speech, the description of the word is dependent on the surrounding words, so the speaker can actually say a word differently when he embeds it among other words. Nevertheless, there are distinct advantages to a linguistically orientated speech recognition system, which are summarised by Medress [1979], who says:

> "While a linguistic framework is a more sophisticated approach to speech recognition and therefore entails a more intensive development period than does acoustic pattern matching, it has some important theoretical advantages. First, a linguistic approach is ideally suited to continuous speech, because it does not require knowledge of word boundaries to accomplish recognition, and because it provides a natural framework for dealing with acoustic-phonetic variability that so often occurs with more natural speech input. Second, a linguistically-orientated system should be less sensitive to acoustic and dialectic differences between users, and would provide several options for efficiently adjusting to such differences if they are significant. Finally, by comparing phonetic rather than acoustic patterns, a linguistic approach can more efficiently and more effectively note critical vocabulary differences, and therefore handle a larger vocabulary and a greater variety of input utterances."

The WISPA system strategy is essentially linguistically based, although certain techniques that are traditionally associated with pattern matching are

used to advantage. However, the basic characteristic of WISPA strategy is the phonetic transcription scheme adopted for the description of utterances. It is in this area that the system described is unique. Many systems using the linguistic approach attempt to transcribe the utterance in the same way as a skilled phonetician would, producing a sequence of phonemes. In this sense they seek to describe the utterance the listener perceives. The WISPA system differs fundamentally from this in that it seeks to describe the way in which the sound is made. It then modifies this description in the light of what it is thought it is possible for the human to perceive. This gives rise to a phonetic encoding scheme that is considerably more stable and quite different from that used in many other recognisers.

3.4.2 Preprocessing

The preprocessor comprises the sub-system between the microphone and short-term memory. It may be conveniently divided into two naturally distinct sections: that executed on hardware and that executed on software (figure 3.14). The hardware is known locally as the Speech Input Device (SID) and designed to perform the first level of data reduction. The preprocessing software is executed on the same processor as the rest of the system components, but might equally well be run on a dedicated processor.

The fundamental aim of the preprocessing strategy is to produce "similar" short-term memories for utterances that are perceived as phonetically the same (but perhaps allophonically different).

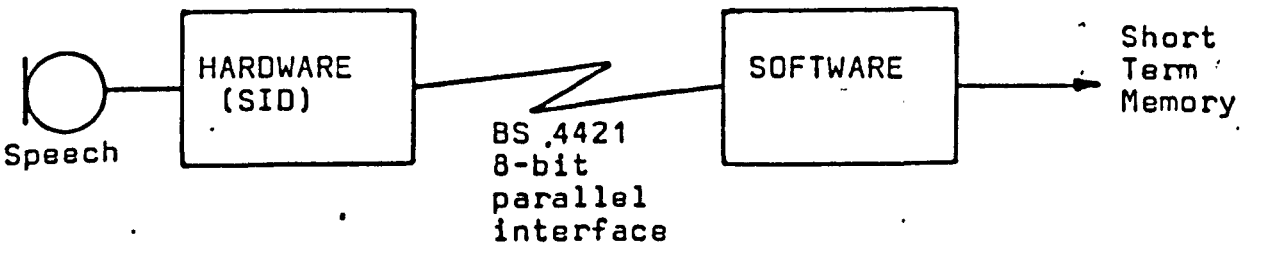


Figure 3.14: The preprocessor components.

Similarity between short-term memories is an equivalence relation that can be formally defined in terms of the trace and distance between them. A loose definition of what would be expected of similar STMs is as follows:

1. The number of STM elements containing a non-empty grade 1 entry should be nominally the same.
2. There should be a one-one correspondence between STM elements.
3. Where a correspondence between STM elements is established, grade 1 entries should be the same.

The thinking behind this definition is that harsh non-linearities in the preprocessed utterance should be removed before the matching process begins. Contrary to many other strategies, it is not felt that time-warping [eg Sakoe and Chiba 1978; Rabiner, Rosenberg, Levinson 1978] should be the primary aim of the matcher, especially in view of the extreme processing overhead that this implies. Rather, the matcher should cope only with phonetic differences in speech caused by dialectic variations or coarticulation.

3.4.2.1 The Speech Input Devices (SID)

The function of SID is to extract as much perception-relevant data from the analogue speech waveform as might permit the real-time operation of a recogniser of continuous speech. The design of SID is established by experimentally distorting speech in order to isolate the parameters which are crucial to the perception, then developing hardware capable of extracting those parameters. Technically there is no reason why its operation cannot be emulated in software, but with existing technology this would prohibit its use in a real-time environment.

The use of SID implies a strict commitment to part of the overall strategy, and as such its function is regarded as a fixed, un-alterable part of the system (ie a system constraint). For this reason, it is important that the design of SID be correct in that it does not discard any information which could be relevant to perception. Although SID is a fixed part of the system, its evolution and improvement are reflected by various revisions of the hardware, which are physically interchanged in the system. The current revision is Mark III, although, as yet, only Mark II is integrated into the system.

The structure of the STM is designed to be largely SID-independent. This is accomplished by using the preprocessing software to make sure that the SID data conforms to the conventions of the STM

Revisions to SID give rise to either improvements in the way parameters are measured or the generation of new parameters. In the case of new parameters,

it is necessary to include their definition in the preprocessing software and the cost function for matching (see 3.3.1.3). Aside from this, there are no further system modifications required.

The design and development of SID does not form part of the work done for this thesis, and so it is not appropriate to discuss it here (see appendices I and II for more information). Furthermore, the current version of SID represents many years work in the area of fundamental speech analysis, and it would be impossible to fully justify that effort without digressing. However, it is important to describe the function of SID, because of its considerable influence over the system design, and the fact that knowledge of its operation is a prerequisite to understanding the overall system.

Of the three SID versions, we may confine our interest to the Mark II and Mark III (known as SID II and SID III). SID II is actually implemented in a system for which performance figures are available. SID III exists and its associated preprocessing software is under development.

SID III has a similar functional philosophy to SID II, but greatly improved specification in that it provides more parameters than SID II, more consistently and under a greater range of operating conditions.

3.4.2.1.1 SID II

SID II continuously samples the analogue speech waveform. Every 10 ms it presents a digital code, of up to 8 bits, at its output interface which describes the information in the signal during the current sampling interval.

The description takes the form of one of three, mutually exclusive classes, plus a quality descriptor for the appropriate class. The class is encoded in 2 bits and the quality is encoded in either 2 or 6 bits depending on the class. This is passed as a single byte as shown in figure 3.15.

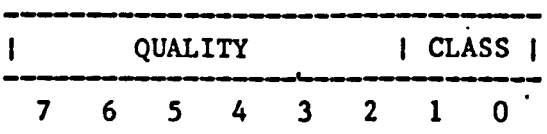


Figure 3.15: The general SID II sample.

The three sample classes are voiced (V), fricative (F) and quiet (Q). Strictly, a fricative is a special type of unvoiced sound (see appendix III), but historically the term has been used at NPL to classify any unvoiced sound.

The Voiced Sample - Voicing uses the class code "00". The voiced quality descriptor uses all 6 bits available, and represents the first and second formant frequencies (f1 and f2), each encoded in 3 bits (8-levels). Figure 3.16 shows the format of a voiced sample, table 3.2 the encoding scheme for formant frequencies.

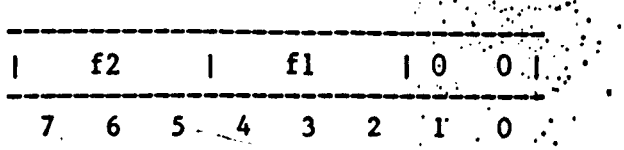


Figure 3.16: The voiced sample.

f1		f2	
Code	Band (Hz)	Code	Band (Hz)
000	< 240	000	< 800
001	240 - 340	001	800 - 1080
010	340 - 430	010	1080 - 1200
011	430 - 510	011	1200 - 1460
100	510 - 580	100	1460 - 1720
101	580 - 640	101	1720 - 1840
110	640 - 750	110	1840 - 2400
111	> 750	111	> 2400

Table 3.2: Formant encoding for voicing.

The technique for the measurement of formant frequencies is that of digital auto-correlation. The two frequencies selected are those which give a correlation in each formant band, and there is no measurement of the degree of correlation. The auto-correlation process assumes that after filtering and clipping, there will only be a single frequency remaining in the band. This frequency is therefore the frequency at which the signal correlates. If more than one frequency remains, it is not necessarily the best correlation, since the priority encoder following the correlators asserts the highest barrier crossed. Appendix I gives the design details of SID II.

The Fricative Sample - Frication uses the class code "10". Two bits are sufficient to describe fricative quality, giving the sample format of

figure 3.17.

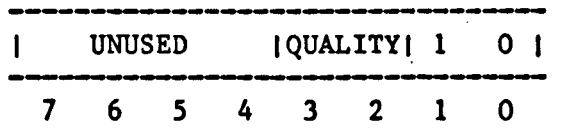


Figure 3.17: The fricative sample.

There are three fricative qualities which are determined by the band in which the speech signal contains greatest coherence. Each band is identified by its associated perceptual sound, in this case "sss","shh","fff", as in table 3.3

Code	Band (Hz)	Sound
00	< 2000	FFF
01	2000 - 3500	SHH
10	> 3500	SSS

Table 3.3: Fricative quality encoding.

The Quiet Sample - Quiet uses the class code "11". The class may be qualified by indicating the transition between unconditional quiet and the onset of either voicing or frication (table 3.4). This condition can occur when there is just sufficient energy in the signal to indicate the presence of voicing or frication, but not enough energy to give an accurate measure of quality.

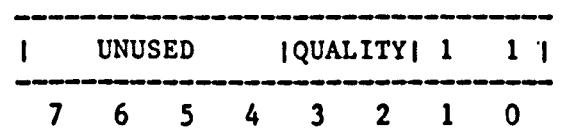


Figure 3.18: The quiet sample.

Code	Meaning
00	Unconditional
01	Tending fricative
10	Tending voiced

Table 3.4: Quiet quality encoding.

3.4.2.1.2 SID III

SID III comprises a set of independent hardware modules, each coupled to a speech bus; the speech bus is an analogue line giving modules access to the pre-amplified speech signal. Modules are each responsible for the measurement of a particular feature or class of features. The digitally-encoded output from each module is gated onto an output highway, which may be sampled by the software every 10 ms. Since the SID III measures many features, and each is independently available to the software, the output highway must, of necessity, be very wide - in fact, 64 bits.

In contrast to SID II, SID III makes no mutually exclusive decisions regarding the selection of parameters for transmission to the software (SID II, for example, is unable to represent a voiced-fricative sound since a sound is classified as either voiced or fricative, and there is no mechanism for describing it as both). It is up to the software to decide the relationship between the outputs of modules, and hence which data is relevant. The implication of this is that the preprocessing software for SID III needs to be more complicated than that for SID II, which is indeed the case.

The SID III Highway - The 64-bit SID III highway is transmitted to the preprocessing software as a sequence of four 16-bit words. Figure 3.19 shows the coding scheme for these words.

Vo Freq. (HF)	Vo Freq. (MF)	Vo Freq. (LF)	0	Qu	Fr	Vo
15 14 13 12	11 10 9 8	7 6 5 4	3	2	1	0

Vo LOC (HF)	Vo LOC (MF)	Vo LOC (LF)	Pitch
31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16

Noise Width	Noise Height	Noise Freq.	Nasal Ampl.
47 46 45 44	43 42 41 40	39 38 37 36	35 34 33 32

Sequence No.	0 0 0 0	LOC Scl. Factr.	Amplitude
63 62 61 60	59 58 57 56	55 54 53 52	51 50 49 48

Figure 3.19: Sample encoding for SID III.

The highway is partitioned into groups of 1 bit or 4 bits. Each group represents the independent measurement of a particular parameter as shown in tables 3.5A and 3.5B.

Group bits	Group name	Group meaning
0	Voicing	Sample contains voiced information
1	Fricative	Sample contains fricative information
2	Quiet	Sample contains no speech
4-7	Voicing Frequency (Low)	Indicates which one of 16 equally-spaced bands in the range 350-700 Hz contains low frequency voicing
8-11	Voicing Frequency (Medium)	Indicates which one of 16 equally-spaced bands in the range 700-1400 Hz contains medium frequency voicing
12-15	Voicing Frequency (High)	Indicates which one of 16 equally-spaced bands in the range 1400-2800 Hz contains high frequency voicing
16-19	Pitch Frequency	Indicates which one of 16 equally-spaced bands in the range 60-200 Hz contains the pitch frequency
20-23	Level of Correlation (Low Frequency)	Correlation level for the LF voicing auto-correlator
24-27	Level of Correlation (Medium Frequency)	Correlation level for the MF voicing auto-correlator
28-31	Level of Correlation (High Frequency)	Correlation level for the HF voicing auto-correlator

Table 3.5A: SID III parameters (bits 0-31).

32-35	Nasal Amplitude	Signal energy in 0-350 Hz band
36-39	Noise Frequency	Signal frequency in the 1-5 KHz band giving maximum energy - 8 equally-spaced bands
40-43	Noise Height	Signal energy of Noise Frequency
44-47	Noise Width	Bandwidth inside 1-5 KHz band which includes Noise frequency and those frequencies within 6 db of Noise Height - 8 discrete widths
48-51	Speech Amplitude	Peak Amplitude of signal
52-55	Level of Correlation Scale Factor	Scaling factor for Levels of Correlation in all three bands
60-63	Sequence Number	Cyclic counter incremented for each new sample

Table 3.5B: SID III parameters (bits 32-63).

The technique for detecting the presence of voicing is one of analogue auto-correlation (correlation is measured as a difference), where a pre-set level of correlation in the 60-200 HZ band is required.

A similar auto-correlation technique (here correlation is measured as a product) is used for the measurement of formant frequencies. Here, the band-limited speech signal is fed into three separate analogue delay-lines for the 350-700 Hz, 700-1400 Hz and 1400-2800 Hz bands. The position of the delay-line tapping which registers the best correlation with the original signal in each band is encoded into 4 bits (there are 16 tappings on the delay-lines), and the result placed on the highway (bits 4-15). At the same time, the "Level-of-Correlation" value is obtained, representing the product of the signal and the delayed signal over a certain period. The

Level-of-Correlation values are also encoded and placed on the highway (bits 20-31). Their resolution may be further increased by the use of "Scale Factoring" (bits 52-55).

Fricative information is available in the form of a frictive detector (bit 1), and three spectral shape encoders: "Noise Frequency" (bits 36-39), "Noise Height" (bits 40-43) and "Noise Width" (bits 44-47). The spectral shape is obtained from 8 filters in the 1-5 KHz band, although the bit allocation for frequency and width allows for the use of 16 filters.

The quiet detector (bit 2) looks for the absence of speech rather than the absence of signal. Built-in knowledge of the energy rhythm of a speech sound enables the detector to discriminate between speech and non-speech by applying structural tests to the signal. This is a considerable advance on simple level-detection methods, and enables SID III to operate well in low signal/noise ratio environments.

3.4.2.2 Preprocessing Software

The preprocessing software operates on SID II data in four sequential processes. The algorithms governing the execution of the processes are regarded as fixed, and adjustments to the operation are made by the modification of a local database of system variables. This concept of "table-driven" software is used throughout the system, and is a feature of the internal structure of modules.

The four processes are:

1. Sampling
2. Encoding
3. Smoothing
4. Compression

Figure 3.20 shows their organisation.

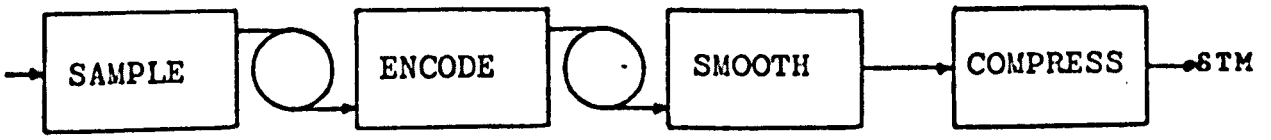


Figure 3.20: The preprocessing software.

There are two important premisses that are part of the preprocessing strategy. The first is that each SID sample is known to exhibit greater resolution than is necessary for equivalent human speech recognition. This is true mainly of formant frequencies where work at NPL and other workers [Gerstman 1968; Schwartz 1971; Wakita and Kasuya 1977; Skinner 1977] has shown that vowels may be catagorised by areas on a first and second (and third) formant plot.

The second premiss is that there is a certain maximum period of time during which it is impossible for the human to perceive the order of linguistic events. This results in the discarding of order information within a limited

time-slot. The premiss is founded on the results of some early unpublished experiments by E.A.Newman of NPL to find the minimum period required by a human to perceive two audible "clicks" as distinct events. These were followed later by some experiments by R.E.Rengger using the SID III and a purpose-built speech reconstitution device called the SID III Monitor, designed to test the effects on perception of various transformations to SID data. These experiments showed that up to eight SID III samples could be randomly shuffled, without destroying the perceived meaning of the various test phrases (digit strings).

Both of these premisses are reflected in the system strategy, but their degree of implementation is variable; highlighting the benefit of table-driven software.

3.4.2.2.1 Sampling

The sampling process is one of taking and storing data from SID II. SID II does not "buffer" its data, so it is important to sample and store whenever new data is available (ie every 10 ms), independent of the current processor workload. This is accomplished by making the process "interrupt-driven".

Each time SID has new data available for transfer, it interrupts the processor. The processor responds by executing the sampling process until the data is safely stored away, after which it continues with the interrupted process.

The processor is usually unable to use the SID data immediately, so the

sampling process must store it until required. A convenient mechanism for this short-term storage is the circular buffer. In this scheme a linear array is treated as circular in that new data overwrites the oldest data in the array, providing that data has been used. Figure 3.21 illustrates the concept graphically.

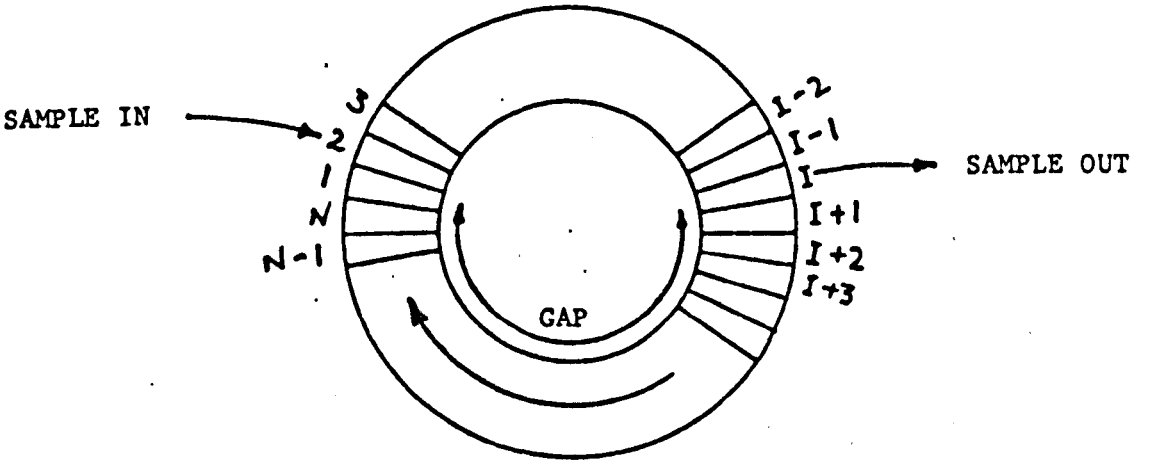


Figure 3.21: Circular buffer arrangement.

The sampling process is controlled by a suite of subroutines (table 3.6). Sampling is commenced under control of the executive, and terminated either by a speech completion condition (signalled by the compression sub-process) or an error condition (eg buffer full or SID failure).

Subroutine	Purpose
SETSID	Sets circular-buffer size, defines SID, protects interrupt vectors
CLRSID	Un-protects interrupt vectors
LISTEN	Starts sampling process
SHUTUP	Terminates sampling process
USESID(D,M,S)	Passes single SID sample and releases sample buffer-space Data (read only) D = value of sample Mode (write only) M = 0 requests NEXT sample M = 1 requests LATEST sample Status (read only) S = 0 no sample ready S > 0 gap between next and latest samples

Table 3.6: The sampling subroutines.

3.4.2.2.2 Encoding

The encoding process for SID data is basically the conversion of SID samples into class/type samples, referred to internally as a encoded samples. The encoded sample uses class and type definitions in exactly the same way as an STM element (see 3.3.2.1).

The procedure for setting the class is fairly trivial for SID II samples. Voicing and frictive are determined by the corresponding SID sample class bits, while quiet is registered for only unconditional quiet (quality 0). Quiet tending fricative (quality 1) is classed as type 1 fricative, and quiet tending voiced (quality 2) is classed type 1 voiced. This is summarised in

table 3.7.

SID II Sample		Encoded Sample	
Class	Quality	Class	Type
Voiced	Any	Voiced	Mapped
Fricative	Any	Fricative	Mapped
Quiet	Unconditional	Quiet	1
Quiet	Tending fric.	Fricative	1
Quiet	Tending voic.	Voiced	1

Table 3.7: Transformation of classes.

Having established the class of the encoded sample, the process uses the quality information to generate type bits. This is done by mapping the quality into the type.

The data format for encoded samples is similar to that used for rationalised STM and ADM elements (see 3.3.2.2), except that there is no need for a grading scheme. Existing implementations use 9 voicing types, 3 fricative types and 1 quiet type, resulting in the format shown in figure 3.22. The format uses two 16-bit words, the first is a class bitmap and the second a type bitmap.

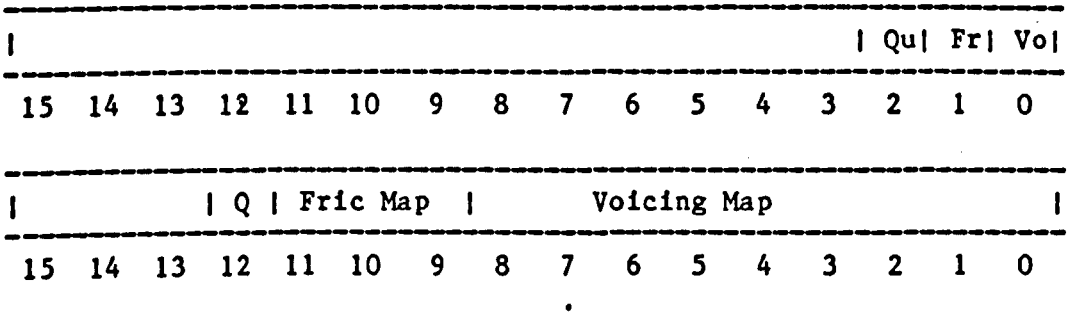


Figure 3.22: Encoded sample format.

Voiced Mapping - By plotting formant frequencies (f1 and f2) one against another, it is possible to experimentally establish areas in the {f1,f2} space, which correspond to single perceived phonemes. Within these areas the change in {f1,f2} has little effect on the human interpretation of the sound. This is not a newly discovered phonomena and has been demonstrated by many other workers in the field (see 3.4.2).

One technique for the type-classification of formants is to partition the {f1,f2} space into several mutually-exclusive areas, each labelled with a unique type identifier. The partitioned area is the type map. The formant frequencies can then be "plugged-in" to the map and the corresponding type identifier "read-off". In practice, the map comprises about six perceptually identifiable areas, resulting in a data reduction of 64:6, or about 10:1.

The disadvantage of this technique is that classification at area boundaries tends to be arbitrary, often oscillating between one type and another. To overcome this, a noval system of intersecting areas is used, where a type-area is defined by its limit of perceptual uniqueness, independent of other types. As a result, type-areas tend to be larger and thus less speaker-dependent, and

two or more types are free to overlap. This overlap is perfectly permissible in the data structure of the encoded sample, since types are represented in the bitmap format.

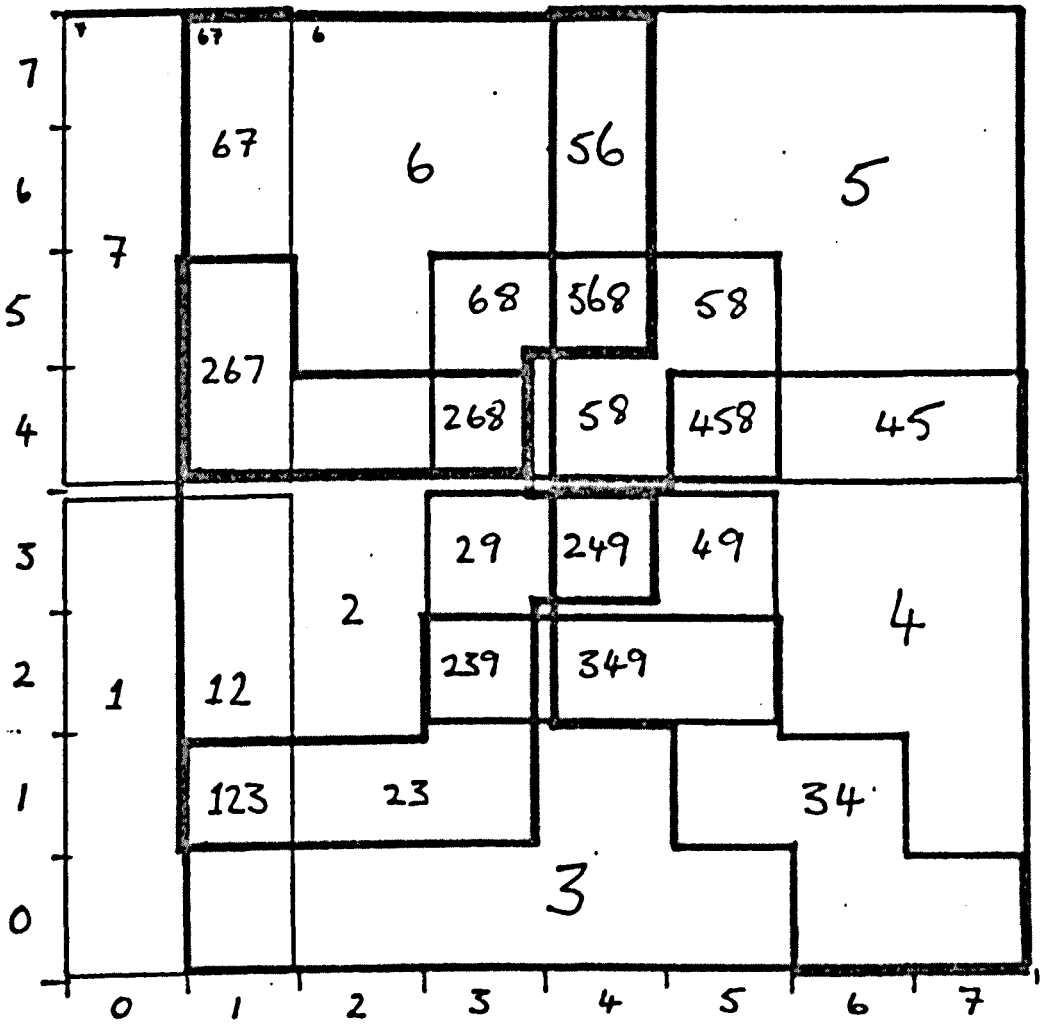


Figure 3.23: A typical voicing map.

Voicing map definition is a system variable, since selection of areas is a prime target for experimentation. In the intersecting area scheme, it is possible to define additional areas to increase resolution where required, although such areas do not necessarily correspond to a phonetically describable sound. For this reason, it is not considered either useful or meaningful to refer to types by anything other than their numeric identifier.

Figure 3.23 shows a typical voicing map for SID II.

Fricative and Quiet Mapping - The mapping of frictive and quiet qualities is trivial since SID II makes a phonetic assessment of frictives and produces a binary decision about unconditional quiet. Nevertheless, their qualities are still mapped to their types even though it is an identity mapping.

3.4.2.2.3 Smoothing

Even after the reduction in data resolution achieved by encoding, there can still be rapid oscillatory changes between encoded samples. These manifest themselves as unstable type bits or even unstable classes. To overcome this instability, the encoded samples are smoothed before the final sub-process of compression.

The smoothing process executes by first filling a circular buffer with encoded samples. Once full, a smooth sample is generated representing the aggregate of all encoded samples in the buffer. Subsequently, new encoded samples are added to the buffer by overwriting the oldest sample. After each new encoded sample is added, a smooth sample is produced as the aggregate of the previous n encoded samples, where n is the size of the circular buffer.

The size of circular buffer and parameters affecting the aggregation are system variables. The trivial case of no-smoothing occurs when the buffer size is 1.

The initial filling of the circular buffer gives rise to a delay in the

production of smoothed samples from encoded samples. The delay is equal to the size of the buffer. Figure 3.24 shows a pictorial view of the smoothing scheme.

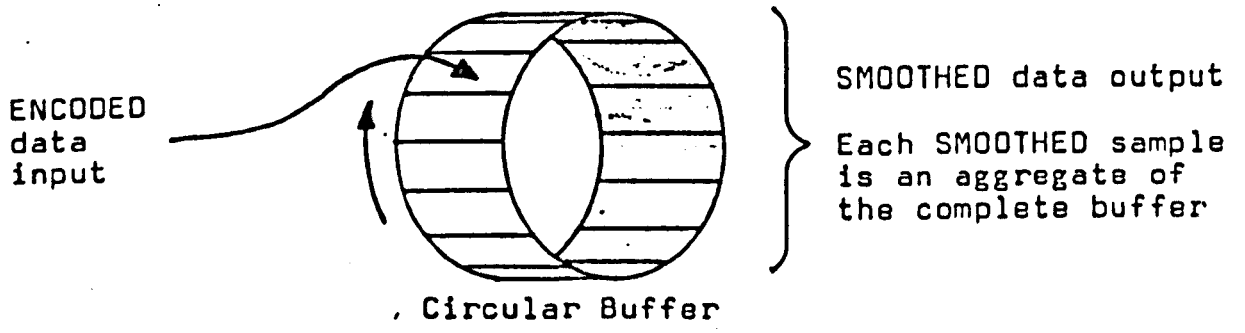


Figure 3.24: The smoothing process.

There are many possible techniques that may be adopted for generalised data smoothing. It is difficult to justify the choice of any one particular technique on a theoretical basis, when, at the end of the day, the choice is limited to the method that gives the best result. Nevertheless, with such a variety of approaches, it makes good sense to isolate the aim of smoothing and select a method that achieves the aim. This, at least, forms a good basis for further experimentation.

The need for smoothing arises because of the lack of hysteresis in the decision making processes. For example, a particular level of sustained, low-energy voicing may result in a sequence of alternating voiced then quiet samples. The subsequent compression process regards class changes as

significant, and so it is important to remove short-term transitions where they are not thought to be relevant to the perception. This is not the same as removing transitions altogether, rather it is the deletion of "tracks" that show no signs of permanence. In short, we wish to introduce some hysteresis.

Unfortunately, since the encoded samples do not represent a simple transformation of the speech signal, it is not possible to apply a discrete analytical technique to the smoothing of data. An alternative is the application of a statistical technique, and one particularly applicable to encoded data is the detection of modal types. The statistical term mode refers to the most popular value, so that the modal type is the type which occurs most frequently.

The method works by first constructing a histogram of types from samples inside the circular buffer. The modal type(s) is then established and a smooth sample constructed, comprising the modal type and any other types whose frequencies of distribution fall within a specified range of the modal frequency of distribution. Where a smooth sample evolves containing non-modal types which relate to mutually exclusive classes, such types are expunged from the smooth sample. Where the histogram is multimodal, containing modal types from mutually exclusive classes, types are expunged if they did not exist in the previous smooth sample.

The smooth sample is a good aggregate measure of a sample group because it is representative without being "average". Average measures can often be meaningless, in that they give no information about the contributing samples (eg the average height of two men says little about their individual heights).

Furthermore, it requires sufficient distribution weight to modify the smooth sample to ensure that a transition has some stability. This fulfils the hysteresis requirement without degrading the transient response.

The system variables applicable to the smoothing process are the size of the circular buffer and limit (called the type inclusion limit) of distribution frequency for types to be included in the smooth sample. Initial experiments use a circular buffer size of 8, and a type inclusion limit of 1 (ie 1 less than the modal frequency of distribution).

As an illustration of the smoothing process, consider the 8 samples in figure 3.25. These generate a histogram (figure 3.26) where the mode is voicing type 6. The frequency of distribution for this type is 4 samples, so that the smooth sample may contain each type with a frequency of distribution of 4 and 4-1 samples. This means that fricative type 3 and voicing type 8 are eligible for inclusion, whereas voicing type 1 is not, since it is present in only 1 sample. SID II defines voicing and fricative as mutually exclusive classes, so that fricative type 3, as a non-modal type must be excluded from the resulting smooth sample: "V .8.6....."

Sample number	Raw	SID II DATA Classified	Encoded	Smoothed
10	12	F 2,0	F .....3..	
11	12	F 2,0	F .....3..	
12	12	F 2,0	F .....3..	
13	0	V 0,0	V 1	
14	210	V 2,4	V ...6...2.	
15	214	V 3,4	V .8.6...2.	
16	254	V 3,5	V .8.6.....	
17	254	V 3,5	V .8.6.....	V .8.6.....

Figure 3.25: An illustration of smoothing.

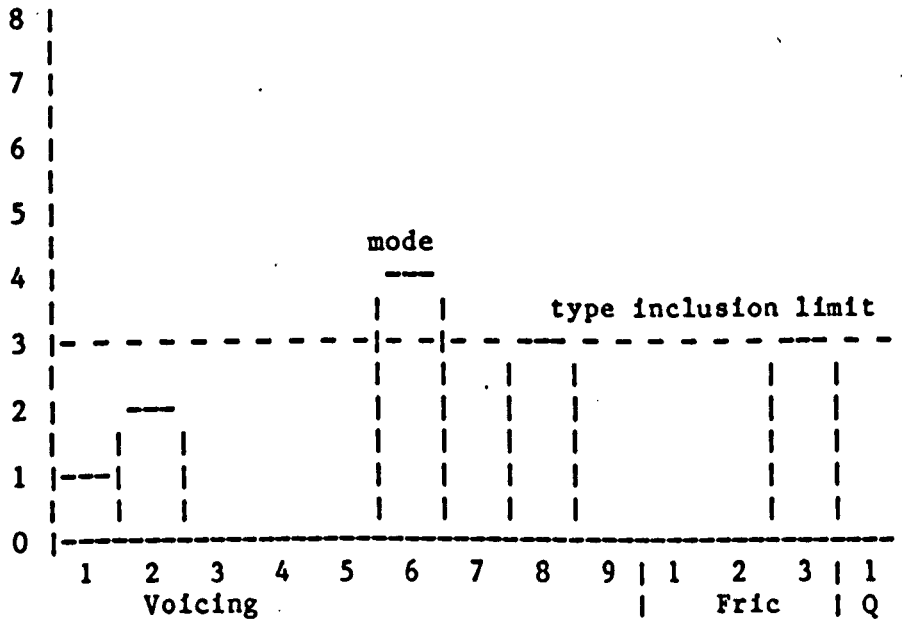


Figure 3.26: Smoothing histogram for samples in figure 3.25.

3.4.2.2.4 Encoding and Smoothing

The processes of encoding and smoothing do not reduce the sample size of the data but re-classify each sample so that the overall variation is reduced. If

we look at the data produced in response to an actual utterance we can see more clearly how this takes place.

Figure 3.27 shows the 3 levels at which data from the word "six" is transformed: the raw data, the encoded data and the smoothed data (the classified data is simply the raw data in a more readable form). The raw data contains a total of 15 data transitions (where the data changes in some way). This is reduced to only 7 transitions in the smoothed data, and as a result gives the data far more compression potential.

The word "six" is a good example of typical speech data, since it contains the three speech classes: voicing, frication and quiet. However, most data reduction occurs in voiced samples, of which "six" has relatively few; so that "six" is not particularly representative of typical transition reductions.

Sample number	Raw	SID II DATA Classified	Encoded	Smoothed
1	3	Q 0,0	Q 1	
2	3	Q 0,0	Q 1	
3	3	Q 0,0	Q 1	
4	12	F 2,0	F .....3..	
5	12	F 2,0	F .....3..	
6	12	F 2,0	F .....3..	
7	12	F 2,0	F .....3..	
8	12	F 2,0	F .....3..	F .....3..
9	12	F 2,0	F .....3..	F .....3..
10	12	F 2,0	F .....3..	F .....3..
11	12	F 2,0	F .....3..	F .....3..
12	12	F 2,0	F .....3..	F .....3..
13	0	V 0,0	V 1	F .....3..
14	210	V 2,4	V ...6...2.	F .....3..
15	214	V 3,4	V .8.6...2.	F .....3..
16	254	V 3,5	V .8.6.....	F .....3..
17	254	V 3,5	V .8.6.....	V .8.6.....
18	314	V 3,6	V ...6.....	V ...6.....
19	314	V 3,6	V ...6.....	V ...6.....
20	250	V 2,5	V ...6.....	V ...6.....
21	250	V 2,5	V ...6.....	V ...6.....
22	10	V 2,0	V .....3..	V ...6.....
23	10	V 2,0	V .....3..	V ...6.....
24	13	Q 2,0	V 1	V ...6.....
25	3	Q 0,0	Q 1	V ...6.....
26	3	Q 0,0	Q 1	V ...6..3..
27	3	Q 0,0	Q 1	V ...6..3.1
28	3	Q 0,0	Q 1	V ...6..3.1
29	3	Q 0,0	Q 1	Q 1
30	3	Q 0,0	Q 1	Q 1
31	3	Q 0,0	Q 1	Q 1
32	3	Q 0,0	Q 1	Q 1
33	3	Q 0,0	Q 1	Q 1
34	3	Q 0,0	Q 1	Q 1
35	3	Q 0,0	Q 1	Q 1
36	3	Q 0,0	Q 1	Q 1
37	2	F 0,0	F 1	Q 1
38	6	F 1,0	F .....2.	Q 1
39	12	F 2,0	F .....3..	Q 1
40	12	F 2,0	F .....3..	Q 1

Figure 3.27A: Sampling, encoding and smoothing of the utterance "six".

Sample number	Raw	SID II DATA Classified	Encoded	Smoothed
41	12	F 2,0	F .....3..	F .....3..
42	12	F 2,0	F .....3..	F .....3..
43	12	F 2,0	F .....3..	F .....3..
44	12	F 2,0	F .....3..	F .....3..
45	12	F 2,0	F .....3..	F .....3..
46	12	F 2,0	F .....3..	F .....3..
47	12	F 2,0	F .....3..	F .....3..
48	12	F 2,0	F .....3..	F .....3..
49	12	F 2,0	F .....3..	F .....3..
50	12	F 2,0	F .....3..	F .....3..
51	12	F 2,0	F .....3..	F .....3..
52	12	F 2,0	F .....3..	F .....3..
53	12	F 2,0	F .....3..	F .....3..
54	12	F 2,0	F .....3..	F .....3..
55	2	F 0,0	F 1	F .....3..
56	3	Q 0,0	Q 1	F .....3..
57	3	Q 0,0	Q 1	F .....3..
58	3	Q 0,0	Q 1	F .....3..
59	3	Q 0,0	Q 1	F .....3..

Figure 3.27B: Sampling, encoding and smoothing of the utterance "six".

3.4.2.2.5 Compression

Compression is the final stage of preprocessing and gives rise to the generation of the short-term memory. The compression strategy is based on three main hypotheses, these are:

(i) Within a certain maximum period, the order in which phonetic events occur is unimportant.

(ii) Beyond a certain minimum period, the duration of steady-state phonetic events is unimportant.

(iii) Certain class and type transitions are of articulatory importance.

The STM is able to reflect these hypotheses in three corresponding ways, by:

(i) Creating STM elements with a fixed maximum duration and within which the order of types occurrence is lost.

(ii) Restricting the creation of contiguous "similar" STM elements to a fixed maximum number, the clip threshold.

(iii) Creating new STM elements whenever a mutually exclusive type chahge occurs. (For SID II this implies class changes as well).

The maximum duration of an STM element in (i) is denoted "m", and the maximum number of contiguous similar STM elements in (ii) is denoted "n". Both of these values are system variables.

The compression process begins by fetching a smooth sample and generating a new STM element (see 3.3.2.1). The new element comprises a class byte, corresponding to that of the initial sample, and a word for each type encountered in the sample. One half of this word specifies the type it represents and the other half the accummulated weight of the type (ie the number of samples containing it), which, for the initial sample, is 1. At the same time, the type bitmap becomes the mask for the STM element.

Subsequent smooth samples are fetched and their type bitmaps are "anded" with the mask - "anding" is a logical bitwise "and" operation. If this results in a zero mask, then the sample is used to create a new STM element. Otherwise, the sample is absorbed into the STM element by incrementing the weights of corresponding types and creating new type entries where necessary. The significance of this procedure is that each element represents samples that have at least 1 type in common. The type (or types) in common is that which

has maximum weight, and it is this weight which defines the overall weight of the element.

Once an STM element represents m samples, the process forces a new element to be created, so that each element compresses between 1 and m samples. Note also that the order information is lost, and that the element simply gives the distribution of types over some period, less than or equal to m.

The compression process may perhaps be better understood by reference to the flowchart in figure 3.29 and the short-term memory listing in figure 3.28. The listing is generated in response to the utterance of the word "six" listed in figure 3.27.

```
8    F 3[8]
1    F 3[1]
8    V 6[8] 8[1]
4    V 6[4] 3[3] 1[2]
8    Q 1[8]
4    Q 1[4]
8    F 3[8]
8    F 3[8]
3    F 3[3]
```

Figure 3.28: STM for the utterance "six".

Each line in figure 3.28 details one STM element. The first column shows the weight of the element, known as the class weight, which is always equal to the maximum type weight. The next column, gives the class of the element, and is followed by the distributions of each type within the class in the form: type[weight]. The 3rd STM element, for example, represents 8 samples containing voicing type 6, and 1 sample containing voicing type 8.

Conditions

C1: Weight of STM(I)=m
C2: STM(I) "similar" to STM(I-1)
C3: Sample AND MASK = 0
C4: STM(I)="Quiet"
C5: Q=TIMEOUT

Variables

I: STM element counter
S: "similar" element counter
Q: "Quiet" element counter
m: maximum STM weight
n: maximum number of "similar elements

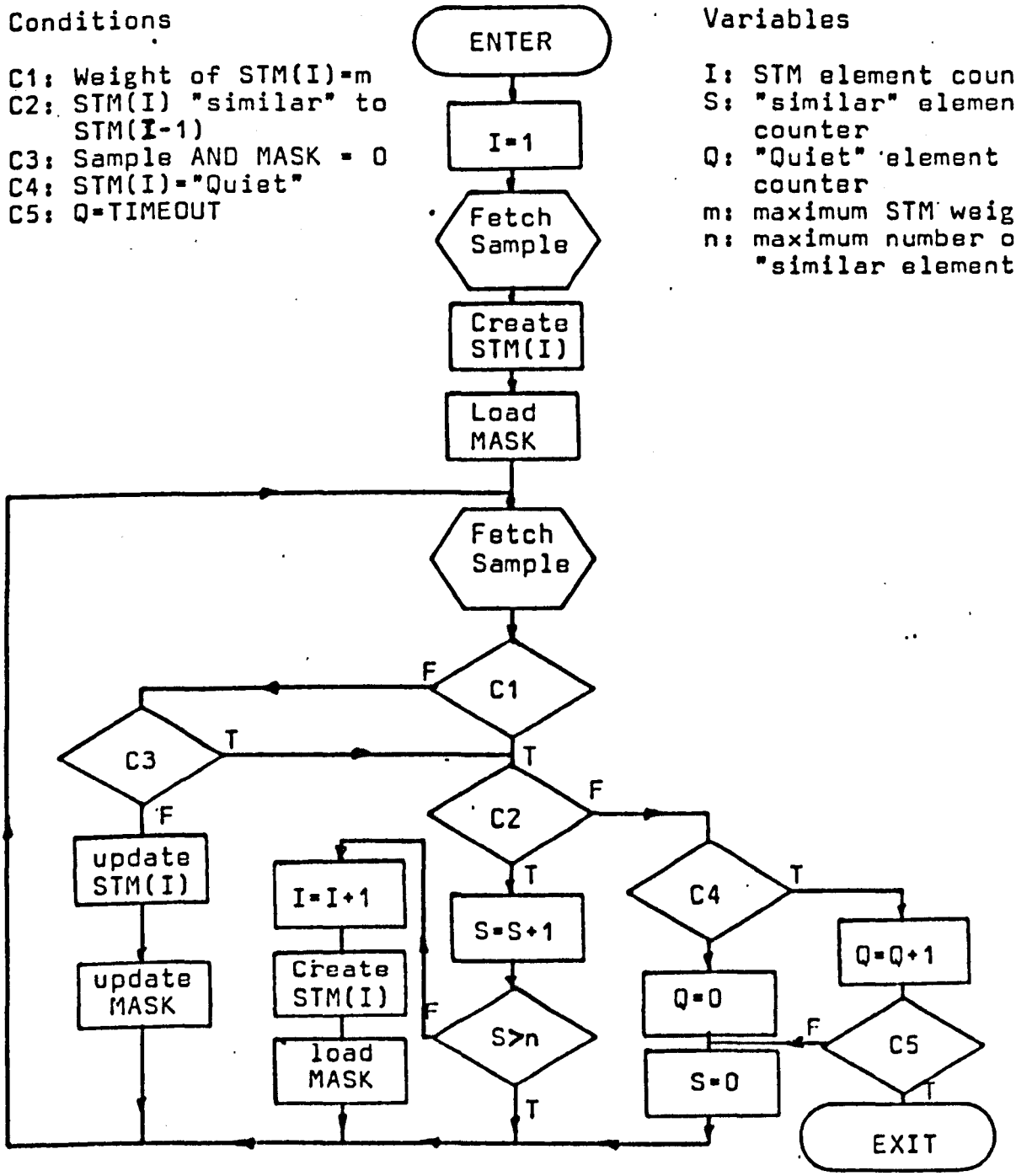


Figure 3.29: The compression process.

3.4.2.2.6 Scheduling of the Preprocessor

We have discussed the scheduling of the sampling process, explaining the need for it to be interrupt driven. However, the creation of STM elements takes place at a much slower rate and in view of the complexity of processing in STM generation, there is no point in attempting to construct an STM element until we have sufficient SID samples to absorb. Indeed, it may be dangerous to drive the STM builder from SID interrupts since processing activity may lock-out further interrupts.

A convenient method for generating STM elements quickly, but without occupying more processor time than necessary, is to "clock-drive" the STM building process. In this scheme, the process is "woken-up" periodically to use all the SID samples that have been created since it was last "awake". The process is run at a lower priority, safe in the knowledge that it is not impeding the sampling of SID, and furthermore, that if it executes too slowly, samples are retained in the circular buffer.

The preprocessor operates concurrently with the remainder of the system processes (except STM rationalisation), which are free to use STM elements as and when available. Operation is started by the executive and halted by the compression process on detection of a speech termination condition such as a quiet time-out.

3.4.3 Rationalisation of the STM

Rationalisation of the STM takes place sequentially with preprocessing and is

scheduled to occur after an STM element has been created. The process consists of grading each type within the element according to its weight. The data structure allows for three distinct grades - grade 1, grade 2 and grade 3. (See figure 3.8)

Corresponding to each grade is some condition Cn. The process creates three bitmaps where bits are set according to the state of the condition for each type. The strategy uses the same following simple condition to set the bitmaps for each of the three grades:

BIT(g,b) = 1 if weight of type b greater than threshold T(g)
= 0 otherwise

where BIT(g,b) is the bth bit in the map for the gth grade

The values of the thresholds are system variables. Figure 3.30 shows the effect of rationalising the STM for the word "six" in figure 3.28. In this example the values for T(1),T(2) and T(3) are 4,2 and 0.

Class	Grade 1	Grade 2	Grade 3
.F.	......3..	......3..	......3..
.F.	....	....	......3..
..V	...6.....	...6.....	.8.6.....
..V	....	...6..3..	...6..3.1
Q..	1	1	1
Q..	....	1	1
.F.	......3..	......3..	......3..
.F.	......3..	......3..	......3..
.F.	....	......3..	......3..

Figure 3.30: Rationalised STM for the utterance "six".

Visual inspection of figure 3.30 reveals the significance of the grading scheme. The grade 1 column highlights the types which occur with significant weight and are thus likely to strongly influence the the perceptability of the utterance. The grade 3 column, on the other hand, resolves finer weight detail, and thus reflects the more subtle variations in perceptability.

3.4.4 Matching

A single patching process consists of the evaluation of a trace and distance (see 3.3.1.3) between a rationalised STM and a rationalised ADM. During training, a single match is executed to establish the way in which the STM is mapped into the ADM. In recognition however, the matching process is executed many times in order to produce the match matrix (see 3.3.2.6) for the utterance.

The general match matrix pair, F(i,j) and L(i,j), are derived by comparing the RADM for the jth word with a series of RSTM sub-sequences commencing with the ith RSTM element (see figure 3.13). This process involves matching the whole RADM with first a single RSTM element, RSTM(j), then two RSTM elements, RSTM(j) and RSTM(j+1), and so on, until the distance measure gets progressively worse for some specified number of consecutive computations; the actual number being specified as a system variable. At this point, the lowest distance and length of RSTM that gives rise to this distance, are recorded in the match matrix as the fit and length, F(i,j) and L(i,j).

The matching process may be executed many hundreds of times in order to produce the match matrix, so it is important that that each process be

expedited as quickly as possible. This is achieved in four ways, by:

1. Reducing the number of elements to be matched
2. Matching by dynamic programming
3. Simplifying evaluation of cost
4. Optimising production of match matrix

Item 1 follows largely as a result of the way in which the speech is preprocessed and rationalised, so we shall begin by discussing the dynamic programming procedure for a single matching process.

3.4.4.1 Dynamic Programming

The dynamic programming procedure is based loosely on a technique described by Wagner and Fischer [1974] for measuring the similarity between strings of characters, with particular reference to spelling corrections. It commences by constructing a penalty matrix for the strings being matched.

We may describe the procedure by considering the matching of a rationalised ADM containing 3 elements: RADM(1), RADM(2) and RADM(3), with a rationalised STM containing 4 elements: RSTM(1), RSTM(2), RSTM(3) and RSTM(4). The relationship between RADM and RSTM elements and the penalty matrix, P is shown in figure 3.31.

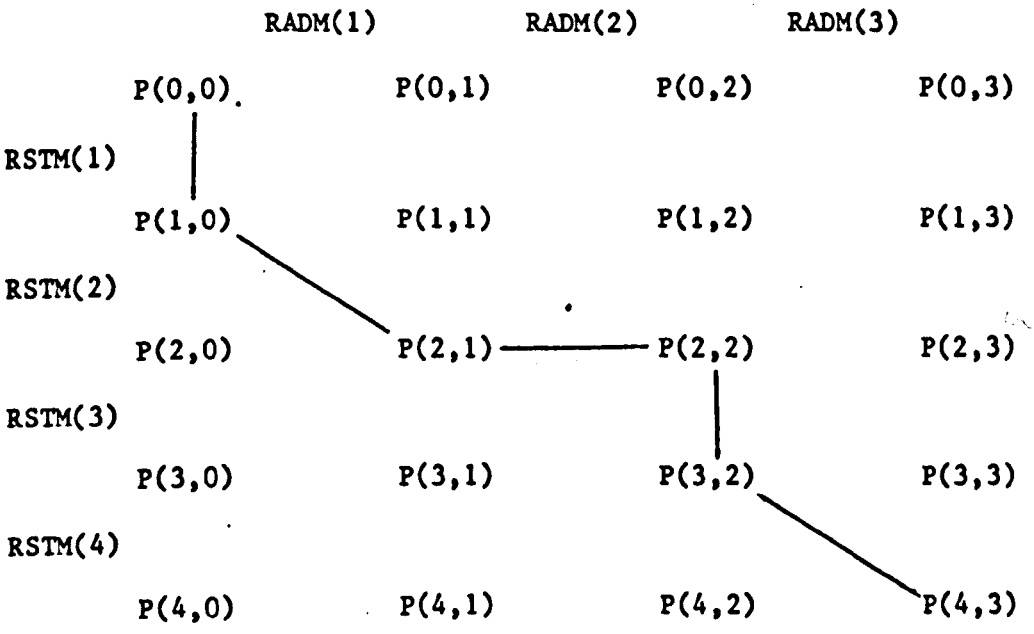


Figure 3.31 The penalty matrix and an example trace.

Penalty matrix elements are defined recursively by the relations:

$$P(0,0) = 0 \qquad (3.1)$$

$$P(0,j) = P(0,j-1) + COST(null,RADM(j)) \quad \text{for all } j>1 \qquad (3.2)$$

$$P(i,0) = P(i-1,j) + COST(RSTM(i),null) \quad \text{for all } i>1 \qquad (3.3)$$

$$M1(i,j) = P(i-1,j-1) + COST(RSTM(i),RADM(j)) \quad \text{for all } i,j>1 \qquad (3.4)$$

$$M2(i,j) = P(i,j-1) + COST(null,RADM(j)) \quad \text{for all } j>1 \qquad (3.5)$$

$$M3(i,j) = P(j-1,i) + COST(RSTM(i),null)) \quad \text{for all } i>1 \qquad (3.6)$$

$$P(i,j) = MIN\ (M1(i,j),M2(i,J),M3(i,j)) \quad \text{for all } i,j>1 \qquad (3.7)$$

where: COST(null,RADM(j)) is the cost of INSERTING RADM(j)

COST(RSTM(i),null) is the cost of DELETING RSTM(i)

COST(RSTM(i),RADM(j)) is the cost of CHANGING RSTM(i) to RADM(j)

In general, each element of P is evaluated by computing the total cost of three possible routes into the element (figure 3.32) and selecting the minimum. The total cost of each route is obtained by adding the lowest of the insertion, change or deletion costs to the corresponding previous penalty matrix element. As a result, the element P(i,j) always represents the minimum cost of matching the first i RSTM elements with the first j RADM elements; and so for the example in figure 3.31, P(4,3) is the minimum cost or distance between the RSTM and RADM.

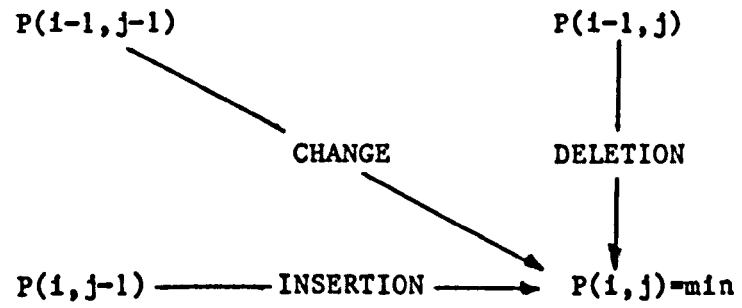


Figure 3.32: Evaluation of P(i,j).

Having obtained the penalty matrix, it is a straightforward process to back-track through the matrix to find the sequence of edit operations which result in the evaluated distance measure, that is, the trace.

If we consider once again figure 3.31, the lines joining elements of P represent a trace through P. Starting from P(4,3) we establish that its value is derived from P(3,2) - we know this from the values at P(4,2),P(3,2) and P(3,3) and the corresponding costs of moving from each to P(4,3). Similarly we find from which element P(3,2) is derived, and so on until we arrive at P(0,0).

We can then interpret the trace as the sequence of edit operations necessary to convert the STM into the ADM. In figure 3.31 there are five discrete operations:

1. Delete RSTM(1)
2. Change RSTM(2) into RADM(1)
3. Insert RADM(2)
4. Delete RSTM(3)
5. Change RSTM(4) into RADM(3)

These may be expressed as a mapping between the RSTM and RADM thus:

RSTM(1) -> <null>
RSTM(2) -> RADM(1)
<null> -> RADM(2)
RSTM(3) -> <null>
RSTM(4) -> RADM(3)

The cost of changes, insertions and deletions are all determined by the cost function, and the purpose of the dynamic programming is to match for minimum overall cost. We can now look at the cost function in some detail to see how it relates elements of the RSTM and RADM.

3.4.4.2 The Cost Function

The cost function operates on two variables - an RSTM element and an RADM element - to give a single scalar result. Since the storage format of the

RSTM and RADM is identical, the function can be considered as operating on a single finite space, R x R, where R is the set of all possible rationalised elements (including the <null> element). We denote the function thus:

COST(A,B) where A,B belong to R

Since the bits in each graded bitmap of a rationalised element are highly dependent on one another, it is not easy to estimate the exact size of the R x R space. However, it is trivial to show that R x R far exceeds the maximum number of elements that could be usefully stored in computer memory, hence precluding evaluation of cost by a look-up table. The format of rationalised element is, in fact, designed for algorithmic evaluation, so this is not a particularly surprising result.

Although both parameters of the cost function belong to the same space, the function is not necessarily symmetric; ie

$$COST(A,B) \neq COST(B,A) \qquad A,B \text{ belong to } R$$

For this reason, the order of parameters is important and so, the first argument always refers to the RSTM element and the second to the RADM element. A special code is used to indicate the presence of a <null> argument, so that the function may detect insertions and deletions.

The minimum possible cost is zero and the maximum defined by a system variable called the unit penalty. The total cost is computed in one of three ways depending on whether the edit function is an insertion, a deletion or a

change.

Insertion Cost - This implies that the RSTM parameter is <null>. Insertion costs are based on three system variables - IC1, IC2 and IC3 - and the state of the RADM bitmaps, so that:

$$\begin{aligned} COST(null,B) &= IC1 \quad \text{if grade 1 RADM bitmap non-empty} \\ &= IC2 \quad \text{if grade 1 RADM bitmap empty} \\ &= IC3 \quad \text{if grade 1 and 2 RADM bitmaps empty} \end{aligned} \tag{3.8}$$

Deletion Cost - This implies that the RADM parameter is <null>. Deletion costs are based on three system variables - DC1, DC2 and DC3 - and the state of the RSTM bitmaps, so that:

$$\begin{aligned} COST(A,null) &= DC1 \quad \text{if grade 1 RSTM bitmap non-empty} \\ &= DC2 \quad \text{if grade 1 RSTM bitmap empty} \\ &= DC3 \quad \text{if grade 1 and 2 RSTM bitmaps empty} \end{aligned} \tag{3.9}$$

Change Cost - The change cost is computed by summing the sub-cost of comparing the bitmaps of each respective grade in the RSTM and RADM. The sub-cost algorithm is based on simple and fast bit-manipulation operations, which check the status of the RSTM and RADM bitmaps and assign a sub-cost, previously defined as a system variable - C1, C2 or C3. Table 3.8 shows the sub-cost assignments.

A,B Word Status	A,B Bit Status	Sub-cost
A empty B empty		C2
A empty B non-empty		C1
A non-empty B empty		C2
A non-empty B non-empty	A is a subset of B	C1
	B is a subset of A	C1
	A ^ B empty	C3
	A ^ B non-empty	C2
Note: "^" is the "intersection" or "bitwise AND"		

Table 3.8: Sub-costings.

The total cost is obtained by summing the weighted sub-costs. The weighting is chosen to reflect the "importance" of each grade and to ensure that maximum cost does not exceed unit penalty, so that;

if we denote SUBCOST(G1(A),G1(B)) as S1,
and SUBCOST(G2(A),G2(B)) as S2,
and SUBCOST(G3(A),G3(B)) as S3,

then $COST(A,B) = (W1 \times S1) + (W2 \times S2) + (W3 \times S3)$ (3.10)

where $W1+W2+W3 = \frac{\text{Unit Penalty}}{\text{Max. Sub-cost}}$ (3.11)

3.4.4.3 Optimising Production of the Match Matrix

There are basically three levels at which the use of the processor may be

optimised to expedite the generation of the match matrix. These are:

1. The match matrix level
2. The penalty matrix level
3. The cost function level

Match Matrix Level - Optimisation here is concerned with the way in which the executive routine (MEMGEN) makes successive calls to the matcher. The primary target is to reduce the number of calls necessary by avoiding the generation of elements in the match matrix that are not used in the subsequent decision process.

By studying complete match matrices, it is possible to establish a relationship between contiguous entries in each column. Once this relationship is known, one may use the values of existing elements to predict future values, and hence avoid computing those whose fits are likely to be too great to be of interest.

Another strategy is to use a parallel decision process to determine the entries worthwhile computing, in a single-pass. This involves applying decision logic to each matrix row as it is produced, rather than waiting until the whole matrix is complete. The decision logic then decides which elements in the next row need evaluating.

Penalty Matrix Level - The generation of the penalty matrix offers considerable scope for optimisation. At this level, there are three main possibilities;

1. Reduction in the number of matrix elements generated
2. Reduction in the number of cost function computations
3. Reduction in the number of penalty matrices necessary to generate each match matrix element

1. It is probable that the trace through penalty matrix representing a good fit follows a more or less diagonal path across the matrix. The further the trace deviates from this diagonal path, then the less likely the word is to correspond to the sub-sequence. This is true particularly of the WISPA system, because the primary function of the dynamic programming algorithm is feature matching, not time-warping. Accepting the fact that good fits produce nominally diagonal traces, there is no point in computing penalty matrix elements which do not contribute to the construction of the trace, namely those some distance from the diagonal. Since we are mainly interested in good fits, we may restrict evaluation of matrix elements to those within a band along the diagonal. The width of the band is determined by the number of contiguous deletions or insertions allowable in the trace. Matrix elements that are not computed are assigned an artificial, high value to satisify the evaluation requirements of elements inside the band.

2. The calculation of insertion and deletion costs is required for every penalty matrix element. Each element in a given column requires the same insertion cost, while each element in a given row requires the same deletion cost. The change cost is normally unique. There may be no point in re-computing insertion and deletion costs when their values can be computed

once, then looked-up on subsequent calls. The decision as to whether costs should be computed or looked-up depends largely upon the time taken to execute the cost function.

3. Inspection of the penalty matrix in figure 3.31 shows that the matrix used for comparision between the RADM and the single RSTM(1) entry is a sub-set of that for the RADM and two RSTM entries, RSTM(1) and RSTM(2). In the general case, the penalty matrix for i RTSM elements is always a subset of that for n RSTM elements, providing $i < n$. The significance of this is that when evaluating a single match matrix element, there need only be a single penalty matrix constructed with sufficient rows for the longest RSTM sub-sequence.

Cost Function Level - Economies in the evaluation of cost function are possible by careful choice of weightings for sub-costings. If a power-of-two weighting ratio can be adopted, then sub-cost computation becomes extremely efficient for a digital processor. By using weight values of 4,2 and 1 for W1, W2 and W3 respectively, equation 3.10 reduces to

$$COST(A,B) = S3 + 2(S2 + 2(S1)) \qquad (3.12)$$

The multiplication by two can be accomplished by a simple left-shift operation, so that the function can be computed in very few instructions. Since it is the cost function which is executed most, economies of processor time here have greatest effect on overall recognition speed. Indeed, it may even be worthwhile to perform the cost function on a dedicated hardware peripheral device.

While the the choice of weightings is extremely important to execution speed, it must be compatible with the system variables, in particular the grading scheme adopted for rationalised elements.

3.4.4.4 Example Matches

We may complete our discussion of matching by looking at two examples of the matching process. In the first (figure 3.33), there is the trace between the RSTM for a version of "six" and a RADM based on the single utterance of the word "six" in figure 3.30. In the second (figure 3.34), the same RADM is used but matched against an RSTM for the word "sox".

```
.F. .....3.. .....3.. .....3.1 to  .F. .....3.. .....3.. .....3.. =  0
.F. .....3.. .....3.. .....3.. to  .F. ........ ........ .....3.. =  6
..V ..6..... ..6..... 8.6..... to  ..V ..6..... ..6..... 8.6..... =  0
<NULL>                         to  ..V ........ ..6..3.. ..6..3.1 =  7
Q.. .......1 .......1 .......1 to  Q.. .......1 .......1 .......1 =  0
Q.. .......1 .......1 .......1 to  Q.. ........ .......1 .......1 =  4
.F. ........ .....3.1 .....3.1 to  <NULL>                         =  7
.F. .....3.. .....3.. .....3.. to  .F. .....3.. .....3.. .....3.. =  0
.F. .....3.. .....3.. .....3.. to  .F. .....3.. .....3.. .....3.. =  0
.F. ........ .....3.. .....3.1 to  .F. ........ .....3.. .....3.. =  0

                                                    TOTAL COST =  24
```

Figure 3.33: "six" to "six" trace.

```
.F. .....3.. .....3.. .....3.. to  .F. .....3.. .....3.. .....3.. =  0
.F. .....3.. .....3.. .....3.. to  .F. ........ ........ .....3.. =  6
..V ....43.. ....43.. ....43.. to  ..V ..6..... ..6..... 8.6..... = 14
..V ....43.. ....43.. ....43.. to  ..V ........ ..6..3.. ..6..3.1 =  7
Q.. .......1 .......1 .......1 to  Q.. .......1 .......1 .......1 =  0
Q.. .......1 .......1 .......1 to  Q.. ........ .......1 .......1 =  4
.F. .....3.. .....3.. .....3.1 to  .F. .....3.. .....3.. .....3.. =  0
.F. .....3.. .....3.. .....3.. to  .F. .....3.. .....3.. .....3.. =  0
.F. ........ ........ .....3.. to  .F. ........ .....3.. .....3.. =  4

                                                    TOTAL COST =  35
```

Figure 3.34: "sox" to "six" trace.

The traces are based on the following (octal) values of system variables for the cost function:

IC1 = 17	DC1 = 17	C1 = 0	W1 = 4
IC2 = 7	DC2 = 7	C2 = 1	W2 = 2
IC3 = 3	DC3 = 3	C3 = 2	W3 = 1

Both mappings are fairly typical of those generally obtained between similar sounding words, and correspond well to what one might expect if the mapping was done by hand.

3.4.5 Merging

The merging process is used to incorporate the short-term memory into the the admissible sequence, based on the trace between the RSTM and RADM. We have discussed the structure of the ADM (see 3.3.2.3) and its elements each of which we may summarise thus:

[Wc.Uc] C T1[W1.U1.A1] T2[W2.U2.A2] . . . Tn[Wn.Un.An]

where:
- Wc — Average weight of element (computed from sum)
- Uc — Number of utterances mapped to element
- C — Class of element
- Tn — Type number
- Wn — Average weight of type (computed from sum)
- Un — Number of utterances mapped to type
- An — Age of type

The trace provides the relationship between elements of the STM and ADM. Where a change takes place, the STM element is <u>included</u> in the corresponding ADM element; where a deletion takes place, a new ADM element is created

corresponding to the deleted STM element; finally, where insertion occurs, no action is taken in modifying the ADM.

The process of inclusion always involves the following operations:

1. Update class weight-sum
2. Increment class utterance counter
3. Modify class byte (if necessary)

and for each type in the STM element:

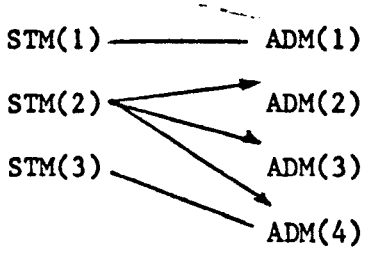
(a) if type exists in ADM:

1. Update type weight-sum
2. Increment type utterance counter
3. Update age

(b) if type does not exist in ADM

1. Create new type entry
2. Load type weight sum
3. Set type utterance counter to 1
4. Load age

The creation of an ADM element may be regarded simply as the inclusion of the STM element with an empty ADM element. However, the position at which the element is created can be less simple to obtain. If, for example, STM(1) traces to ADM(1), STM(2) traces to <null> and STM(3) traces to ADM(4), then there are three possible locations for a new ADM element thus:



The procedure to select the appropriate location is one of finding the distance (using the cost function) between the deleted STM element and the

possible neighbours for the new ADM element. Having found the lowest neighbour, the elements either side of it are costed and lower of the two is termed the second neighbour. The new ADM element is then created between the lowest and second neighbours.

In conclusion, we may look at an example merge, shown in figure 3.35. Here, we take a version of "six" (from figure 3.28) and create a new ADM (figure 3.35A). We then match this with the STM of a second version of "six" (figure 3.35B) to produce a trace (figure 3.35C). The trace is then used to incorporate the second version into the ADM (figure 3.35D).

```
[8.1]   F 3[8.1.1]
[1.1]   F 3[1.1.1]
[8.1]   V 6[8.1.1] 8[1.1.1]
[4.1]   V 6[4.1.1] 3[3.1.1] 1[2.1.1]
[8.1]   Q 1[8.1.1]
[4.1]   Q 1[4.1.1]
[8.1]   F 3[8.1.1]
[8.1]   F 3[8.1.1]
[3.1]   F 3[3.1.1]
```

Figure 3.35A: ADM created from utterance "six" in figure 3.28.

```
8    F 3[8] 1[1]
6    F 3[6]
8    V 6[8] 8[2]
8    Q 1[8]
5    Q 1[5]
4    F 1[4] 3[3]
8    F 3[8]
8    F 3[8]
3    F 3[3] 1[2]
```

Figure 3.35B: STM for a second utterance of "six".

```
STM< 1> to ADM< 1>  COST:  0
STM< 2> to ADM< 2>  COST:  6
STM< 3> to ADM< 3>  COST:  0
<NULL>  to ADM< 4>  COST:  7
STM< 4> to ADM< 5>  COST:  0
STM< 5> to ADM< 6>  COST:  4
STM< 6> to <NULL>   COST:  7
STM< 7> to ADM< 7>  COST:  0
STM< 8> to ADM< 8>  COST:  0
STM< 9> to ADM< 9>  COST:  0
```

Figure 3.35C: Trace from STM to ADM (c.f. figure 3.33).

```
[8.2]   F 3[8.2.2] 1[1.1.2]
[3.2]   F 3[3.2.2]
[8.2]   V 6[8.2.2] 8[1.2.2]
[4.1]   V 6[4.1.1] 3[3.1.1] 1[2.1.1]
[8.2]   Q 1[8.2.2]
[4.2]   Q 1[4.2.2]
[4.1]   F 1[4.1.2] 3[3.1.2]
[8.2]   F 3[8.2.2]
[8.2]   F 3[8.2.2]
[3.2]   F 3[3.2.2] 1[2.1.2]
```

Figure 3.35D: Resultant ADM.

3.4.6 Rationalisation of the ADM

The ADM is rationalised in much the same way as is the STM, resulting in elements of an identical format. The meaning of the graded types, however, is different. Whereas the STM elements are rationalised by grading each type according only to its weight, the grading scheme for ADM elements is designed to make use of their weight-sums, utterance-counts, and ages. In grading each type, we seek to construct a rationalised ADM element that optimally produces the minimum cost for each of the RSTM elements which contributes to its creation.

Two example base-line algorithms for type grading are as follows:

Algorithm 1:

$$BIT(g,b) = 1 \text{ if } \frac{TWS(b)}{WUC} > T(g)$$

$$= 0 \text{ otherwise}$$

Algorithm 2:

$$BIT(g,b) = 1 \text{ if } \frac{TUC(b)}{WUC} > E(g) \text{ AND } \frac{TWS(b)}{TUC(b)} > T(g)$$

$$= 0 \text{ otherwise}$$

where BIT(g,b) is the bth bit in the map for the gth grade
TUC(b) is type utterance-counter for the bth type
TWS(b) is the type weight-sum for the bth type
WUC is the word utterance-counter
T(g) is the weight threshold for grade g
E(g) is the existence probability threshold for grade g

The effect of algorithm 1 is that types are graded according to their average weight. The average is obtained independently of the number of utterances actually containing the type, so it is an average for all the training utterances.

Algorithm 2 first checks the type utterance-counter against some pre-defined threshold, E(g) and if this is exceeded, it continues by testing the average weight. In this algorithm however, the average is taken on only the set of utterances containing the type.

The values for T(g) and E(g) (g=1,2,3) are system variables, and are set as a function of thresholds for the rationalisation of the STM. The effect of these algorithms is discussed in Chapter 4. Note that neither of these algothithms use "age" values, simply because the best way of using this parameter is yet to be established.

3.4.7 Deciding

There are many strategies for the extraction of a single key string from a match matrix of the type described here. Techniques applicable to the WISPA system are investigated in some depth by Armstrong [1980]. He shows that the WISPA system trained on isolated utterances, and giving a word spotting probability of P, can, using the technique of semantic segmentation [*] yield the theoretically predictable phrase recognition probability of P**N, where N is the number of connected words. This highlights the importance of an intrinsically high word spotting capability, especially for recognition of continuously spoken strings of several words. As a result, it is convenient to use a simple word spotting algorithm in order to evaluate the "front-end" of the system.

The word spotting algorithm adopted scans each column in the match matrix to obtain a "normalised" fit. This is the quotient of the fit and the cummulative weight of the RSTM elements used in obtaining the fit, the number of RSTM elements being given by the length value. The normalised fit is thus the fit per unit time, and is independent of the duration of the word.

Having converted each fit/length pair into a normalised value, the procedure continues by searching for minima within each column. Those minima within some given threshold are deemed as the start points of a spotted word, the identity of which is given by the column. This information is commonly displayed on a spotting matrix, which indicates the section of utterance

---

* The term "semantic" as used here applies at the phonetic feature level rather than at the word level. In this sense it refers to the meaning of strings of features.

thought to contain the word, and the normalised fit (see figure 3.36).

The spotting matrix does no more than indicate spotted words, and makes no attempt to resolve any ambiguities such as totally overlapping words. It it the semantic segmentation procedure that makes the final decision as to which word string is recognised from amoung the continuous speech. This it does from information obtained directly from the match matrix (see section on future work in chapter 5).

```
                                          1 2 3 4 5 6 7 8 9

STM        G1         G2         G3

.F. ......3.. ......3.. ......3..  . . . . . 3 . . .
.F. ......3.. ......3.. ......3..  . . . . . 3 . . .
..V ...6..... ...6....1 ...6..321  . . . . . 3 . 6 .
Q.. .........1 .........1 .........1  . . . . . 3 . 6 .
Q.. ......... ......... .........1  . . . . . 3 . 6 .
.F. ......3.. ......3.. ......3.1  . . . . . 3 1 6 .
.F. ......3.. ......3.. ......3..  . 5 . . . 3 1 . .
.F. ......... ......... ......3..  . 5 . . . 3 1 . .
..V .8..5.... .8..5.... 98..54321  6 5 . . . . 1 . .
..V .......2. .......2. ......321 *6 5 . . . . 1 . .
..V ......3.. ......3.. ......321 *6*5 . . . . 1 . .
..V ......32. ......32. ......32. *6 5 . . . . 1 . .
..V .......2. .......2. .......2. *6 5 . . . . 1 . 2
..V .......2. .......2. 9....4.2. *6 5 . . . . . . 2
..V 9....4... 9....4... 9....4...  6 . . . . . . . 2
..V .....4... .....4... .....4...  6 . . . . . . . 2
..V .....4... .....4... .8..54...  6 . . . . . . . 2
..V .8..5.... .8..5.... .8.65.32.  6 . . . . . . . 2
..V ......3.. ......3.. ......32.  6 . . . . . . . 2
..V ......... ......... ......3..  6 . . . . . . . 2
Q.. .........1 .........1 .........1  . . . . . . . . .
Q.. .........1 .........1 .........1  . . . . . . . . .
```

Figure 3.36: Spotting matrix for the phrase "six-seven-nine". Note the overlapping fricative features - the terminal F3 ("sss" sound) in "six" overlaps the initial F3 in "seven".

3.4.8 Vocabulary Management

The management of vocabularies is a system task having little effect on the speech processing strategy. Efficient vocabulary manipulation is, nonetheless, an important feature of any speech system since it provides insight into the operation of different strategies by allowing the examination of their effect on modified vocabularies.

The main purpose of the vocabulary manager is to provide an effective sub-setting procedure and to keep memory overhead to a minimum.

The sub-setting mechanism for the WISPA system involves allocating each word in a vocabulary a sub-set descriptor. This is a single 16-bit computer word in which each bit represents one or more of up to 16 sub-sets to which the word may belong.

Memory usage is kept to a minimum by allowing only rationalised ADMs to be resident during recognition and restricting a single ADM to memory during training.

3.4.9 The Executive

The executive acts as the main control process to recognition and training. It is responsible for the scheduling of the system processes and the allocation of system resources. It also provides the operational interface between the user and the system. In this role, the executive looks after the user interaction and provides complete diagnostic and monitoring facilities.

Under the control of the executive, the user may inspect all the system databases at strategic points in the processing and thus ascertain exactly how the strategy performs.

WISPA provides two command interfaces between the user and system; one for training and one for recognition. The syntax of the commands is summarised in the "help" messages available to system users (figures 3.37 and 3.38).

The sub-system for watching system databases is called the process monitor. The user may selectively enable database monitoring by using special commands to turn monitor points on or off. The status of each monitoring point can be obtained by issuing a command to display the process monitor status table. Figures 3.39 and 3.40 show the monitor points provided by the initial WISPA system executive.

```
Command prompts are of the form: COMMAND [Default-Key:subset] ?

There are 10 valid commands. Speech input is assume to come from
SID-II or, if specified, from file defined by "filespec".

1       Key[/filespec]   Train the Word "Key". Make "Key" the Default-Key.
2       [/filespec]      Train the Default-Key. Input as per 1.
3       Key#             Display the ADM for "Key".
4       Key#Subset       Change subset descriptor of Key to Subset.
5       Key#0            Delete "Key" from the Vocabulary.
6       M                Display the Monitor Status Table.
7       >filespec        Reset the Monitor output file.
8       +n[+n][-n]       Set(+) or clear(-) the nth Monitor bit.
9       ?                Help.
10      <escape>         Write Vocabulary to disk.

where:  "Key"            is an identifier of up to 3 characters
        "Subset"         is an OCTAL number of up to 6 digits
        "filespec"       is a standard RT-11 file specification
        "n"              is a single hexadecimal digit
```

Figure 3.37: Help message for the training sub-system.

```
Command prompts are of the form:   COMMAND [Default-Subset] ?

There are 7 valid commands. Speech input is assumed to come from
SID-II or, if specified, from file defined by "filespec".

1       Subset          Change the Default-Subset to Subset.
2       [/filespec]     Apply recognition algorithms on all the
                        words in the default subset.
3       M               Display the Monitor Status Table.
4       >filespec       Reset the Monitor output file.
5       +n[+n][-n]      Set(+) or clear(-) the nth Monitor bit.
6       ?               Help.
7       <escape>        Write Vocabulary to disk.

where:  "Subset"        is an OCTAL number of up to 6 digits
        "filespec"      is a standard RT-11 file specification
        "n"             is a single hexadecimal digit
```

Figure 3.38: Help message for the recognition sub-system.

```
Bit  Dataset                          State

  1  Short-Term Memory                 ON
  2  Admissible Sequence               ON
  3  Trace                             OFF
  4  Penalty Matrix                    OFF
  5  Rationalised inputs to matcher    OFF
  6  Pruning enabled                   OFF

Process Monitor Output File - TT:
Free space in ADM:    200
Free space in Vocabulary:  2048
```

Figure 3.39: Training process monitor status table.

```
Bit  Dataset                          State

  1  Trace                             OFF
  2  Match Matrix                      OFF
  3  Spotting Matrix                   ON
  4  Embedded single-word decision     OFF
  5  Isolated single-word decision     OFF
  6  Continuous word decision          ON

Process Monitor Output File - TT:
Free space in Rationalised Vocabulary:  1975
```

Figure 3.40: Recognition process monitor status table.

3.4.10 System Variables

Throughout the discussion of the system strategy, we have referred to the use of system variables. The purpose of a database of system variables is to isolate those parameters whose optimal values for best system performance can only be obtained by experiment. Having isolated them, we can adjust them without the need to reprocess the program code.

The actual technique used for this, is to place the system variables in a global data area which may be loaded from a file at run-time. The file containing the system variables then becomes the definition of the system. A sample system variable file is shown in appendix VI.

3.5 Conclusion

We have examined the WISPA system by looking at it in three parts; the system capability, the system structure, and the system strategy. The objective has been to evolve a system which may act as a vehicle for examining the problem of continuous speech recognition. The evolution has been accomplished by applying orderly system design procedures to what is universally accepted as a difficult problem. The system has fully met this objective and proved, in practice, to be a powerful aid to understanding the effects of different recognition and training strategies.

Table 3.9 summarises the four main system processes that characterise WISPA from the user viewpoint. In chapters 4 and 5 we go on to look at the system in action and the implications for future work.

1. Preprocessing	Use of hardware/software preprocessor permits fast compression of speech into short strings of phonetic features, containing only perception-relevant information. Preprocessor encodes "time" information into the phonetic features, thus removing the need for the matching process to warp the time scales. (Phonetic features are language-based, hence have greater speaker-independence than simple frequency measures.)
2. Matching	Matching expedited by the use of dynamic programming techniques on short feature strings, making real-time execution feasible on only a micro-processor. The matcher operates as a true "word spotter", enabling the system to recognise continuous speech, as well as connected and isolated words.
3. Deciding	Decision logic designed for the context-free recognition of continuous speech, but internal database allows for application of syntax and semantics.
4. Training	System may be trained completely automatically by non-expert users. Extensive vocabulary manipulation facilities, including "sub-setting". Single template words permit fast processing in multi-speaker environment.

Table 3.9: The main WISPA system processes.

CHAPTER 4

Performance Measurement

4.1 Introduction

In any experimental system it is necessary to adopt some techniques for performance measurement before the effectiveness of a strategy can be objectively evaluated. Ideally, some standard "benchmarks" should be applied to a single source of speech data to gain a comparative assessment of the performance of the speech recognition device. Despite the not inconsiderable efforts of some people to organise this, no such common speech database or benchmarking system has yet evolved, and so researchers have developed their own, loosely conforming to some established criteria. The failure to produce a universally acceptable measurement system lies to some extent with the variation in types of system; for example, the simple tests that can be applied to an isolated word recogniser may be meaningless to a speech understanding system. Although descriptive techniques have been proposed for the analysis of speech recognition systems [eg Moore, 1976], none have yet been universally adopted.

The approach that is taken for the testing of WISPA is simple, and easily interpreted. It consists of the construction of a database of

hardware-preprocessed speech, obtained from a variety of subjects, followed by a series of well-defined experiments executed using the database. The results are used to give an absolute measure of performance as well as peripheral information indicating possible problem areas. The complexity of the database and difficulty of the experiments can be adjusted to suit the performance of the system - that is, as the performance of the system approaches 100% the tasks can be made more demanding.

Lea [1979] lists six important, but vague, performance criteria that may form part of a speech recognition system specification. These are:

1. Recognition Accuracy
2. Training the Machine
3. Training the User; User Acceptance
4. Speed
5. Cost
6. Time for delivery

Broadly, these are listed in order of academic rather than commercial importance, and here, we are concerned mainly with the measurement of the first two. We go on to look at these in more detail later. At the recognition level of any speech driven application, WISPA is designed to be trained on isolated utterances by the non-expert user. The system does not require any particular discipline of the user, save that he speaks into a microphone. Any constraints applied to the user by way of special vocabularies or syntaxes are all application-dependent and as such can only be evaluated in the context of that application. The development of a device which operates fast and cheaply is one of the design goals of the WISPA system, so to an extent, high-speed and low-cost are intrinsic features of the system. The measurement of speed and cost is straightforward and need not be discussed here.

4.1.1 Recognition Accuracy and Answer Matching

The most fundamental measure of a speech recognition system's performance is its recognition accuracy; this is the percentage of all utterances that it correctly identifies. Associated with the recognition accuracy is the error rate and the reject rate. The error rate is the percentage of utterances wrongly identified by the system , while the reject rate is the percentage of utterances the system is unable to identify. The sum of the recognition accuracy, the error rate and the reject rate is 100%.

In the case of an isolated word recogniser, it is simple to decide whether or not a word has been correctly recognised. In a continuous speech system, like WISPA, it is more difficult. For example, if a phrase is uttered as "one-four-six" and recognised as "one-six", is it correct? There can be no doubt that the phrase is wrong, but clearly there are individual words which appear to be correctly identified. It would be useful to be able to distinguish between a phrase recognition rate and a word recognition rate. The word recognition rate is an especially useful measure because it is independent of the number of words in the phrase, but to evaluate it we need to find the correspondence between words in the "input" phrase and words in the "output" phrase. To do this we use an answer matching process.

The answer matching process developed for the WISPA system uses the same string matching technique as that used for comparing phonetic features. Indeed, the section describing this (3.3.1.3) may be applied directly to answer matching. By assigning each word a key and defining a trivial cost function between keys, we can evaluate the distance and trace between any

given pair of key strings. The trace is particularly useful since it gives the correspondence between individual keys, and thus enables the construction of a manageable confusion matrix. The confusion matrix summarises the relationship between input and output words for a given set of speech data. Each row in the matrix represents a word "said" by the speaker and each column represents a word "recognised" by the system. There is a row and column for every word in the vocabulary. In addition, there is an extra row for a word not said yet still recognised, an inserted word; and an extra column for a word said but not recognised, a rejected or deleted word. The confusion matrix gives an immediate summary of the performance of the system on individual words.

To construct a confusion matrix, we begin by initialising all elements to zero. We then apply the string matching routine to each input and output pair of strings - this gives the mapping for each key (see figure 4.1). Using the

Input string (said)	Output String (recognised)	Mapping			Result
one-two-three	two-two-three	one	->	two	wrong
		two	->	two	correct
		three	->	three	correct
one-two-three	one-three	one	->	one	correct
		two	->		deleted
		three	->	three	correct
one-two-three	two-three-four	one	->		deleted
		two	->	two	correct
		three	->	three	correct
			->	four	inserted
one-two	one-two-three	one	->	one	correct
		two	->	two	correct
			->	three	inserted

Figure 4.1: Examples of answer matching.

key and its mapping to index a pointer to a matrix element, we can then increment that element.

Figure 4.2 shows a confusion matrix and statistical results from a system trained on the ten digits, zero to nine, and attempting to recognise 100 multiple-digit phrases. Note that although the sum of "correct", "wrong" and "deleted" (or rejected) words should, before rounding be 100%, the "inserted" word rate must be regarded seperately. Insertions do not necessarily occur at the expense of other classifications and as such are quoted simply as a percentage of the sample size.

Results from experimental data in file DL1:EXPT05.BEP

Confusion Matrix:

		Words recognised										
		0	1	2	3	4	5	6	7	8	9	<null>
W	0	30	0	0	0	0	0	0	0	0	0	0
o	1	0	30	0	0	0	0	0	0	0	0	0
r	2	0	0	30	0	0	0	0	0	0	0	0
d	3	0	0	1	23	0	0	0	0	0	1	5
s	4	0	0	0	0	27	0	0	0	0	0	3
	5	0	2	0	0	1	27	0	0	0	0	0
s	6	0	0	0	0	0	0	29	0	0	0	1
a	7	0	1	3	0	1	0	0	25	0	0	0
i	8	0	0	0	0	0	2	0	0	22	0	6
d	9	0	4	0	0	0	2	0	0	0	23	1
	<null>	0	0	0	0	0	0	0	0	0	0	0

100 Phrases Spoken

68[68%] correct

300 Words Spoken

266[88%] correct
18[6%] wrong
0[0%] inserted
16[5%] deleted

Figure 4.2: Sample confusion matrix and summary.

Justification for Answer Matching - There are some situations where answer matching can give misleading or optimistic information. For example the second string in figure 4.1 shows "one" and "three" as correct and "two" as deleted, when in fact there are other possible mappings that could give a different result. In the absence of any information giving the position-in-time of the answers with respect to the input utterance, the algorithm chooses the best string match. In fact, experiments have shown that in over 90% of cases, the answer match algorithm gives the same result as a manually executed match, making use of "time" information. As a further measure of confidence, the word recognition rates correlate well with the theoretically predicted values of the nth root of the phrase recognition rate, where n is the number of words in the phrase.

4.1.2 Training the Machine

Great importance is attached to machine training in the WISPA system, and so it is essential that the effect of variable amounts of training be measured. This is accomplished by designing experiments which apply a series of vocabularies, each trained in a different way, to the same recognition task.

The initial training stategies for WISPA require that each word be individually trained, although not necessarily by every user. This is in contrast to strategies that require only some words to be trained by every user, and remaining words being defined by inference from those trained [Furui 1980].

4.2 Execution and Design of Experiments

Clearly, there is an infinity of experiments that may be performed and so it is best to choose a set which gives a cross-section of tasks and may be executed in a reasonable amount of time. Initial trials with the first configured WISPA system began with ten separate experiments operating on a total of 1200 uttered digits. The ten experiments run as a time-shared job under the multi-user executive TSX-PLUS (multi-user RT-11) and take about 2 hours to complete (depending on system workload). These experiments are described in Appendix IV.

Experiments are only executed in response to some change in the system strategy. This most usually takes the form of a modification to the System Variable table or program code (including bug fixes). The experiment is defined by a batch stream (known as an Indirect Command File in the RT-11 operating system) which contains commands to run the appropriate programs and commands to the programs themselves - these might be vocabulary file specifications or speech file specifications. Recognition results are stored in a file which is subsequently processed to give a confusion matrix and statistical summary of the experiment (see figure 4.2).

The design of experiments is influenced by three main factors:

1. The Speech Database
2. Recognition Task
3. Training Task

These, in turn, are determined by the difficulty of the task for which we wish to measure performance. We shall look at each of these factors in turn, discussing their specification criteria with reference to the requirements of "initial" WISPA system testing. The initial WISPA system is essentially that whose strategy is described in chapter 3. It is the baseline system from which system development has begun.

4.2.1 The Speech Database

The definition of the speech database can be summarised by five criteria. These are as follows:

1. Storage format of speech data
2. Definition of vocabularies
3. Construction of utterances
4. Demands on each subject
5. Choice and number of subjects

Storage Format of Speech Data - There are three principal alternatives for speech storage: (i) magnetic tape to store the analogue signal, (ii) magnetic tape or disc to store the digitised signal, and (iii) magnetic tape or disc to store the SID preprocessed speech. Analogue magnetic tape must be accessed serially and is difficult to address automatically. It is invaluable as a master record of the original speech data but unsuitable for direct program access. Digitised speech, on the other hand, can be easily addressed but requires in excess of 250,000 bit/sec to faithfully record the signal, and without extremely large mass-storage devices this technique permits storage of only 200-300 utterances. The storage of SID preprocessed speech provides the

best medium for the database since it requires only 800(SID II)/6400(SID III) bit/sec, making it feasible to store up to 10,000 utterances on a single hard disc. It does mean that it is impossible to test the effects of hardware changes or environmental conditions (eg noise), but given that the original speech is stored on magnetic tape, these may be re-processed as required. The current database stores utterances preprocessed by both SID II and SID III.

Definition of Vocabularies - Ideally the choice of words for vocabularies should be as application-independent as possible. Although there is a variety of common vocabularies which fulfil this requirement, the numeral set is undoubtedly the most general. There are many applications that require digit recognition at some point in their interaction. In addition, it is a set for which there are considerable performance statistics covering many recognition systems, and as such is often regarded a "benchmark" vocabulary. The numeral set forms the basis of the WISPA database vocabulary.

Construction of Utterances - The construction of database utterances relates to the linking of words into connected words or continuous speech. For an isolated word recogniser the construction is trivial - only one word may be uttered; for a speech understanding system the construction may be complex - utterances are defined by permissible syntaxes or semantics. The WISPA database stores both isolated words and groups of three connected digits. The sequences of digits are chosen to give an example of every possible pairing of digits for each speaker.

Demands on Each Subject - The number of utterances that should be collected from each subject is a trade-off between the statistical significance of the sample set and the cooperativeness of the subject. The collection of speech from subject can, in practice, often be difficult to organise. It is neither reasonable nor desirable to expect a subject to enter a complete dataset in one sitting, and so an individual's database is more conveniently built over several sessions. The WISPA trials commence with each speaker providing 100 isolated digit utterances and 100 digit phrase utterances during 5 seperate sessions.

Choice and number of Subjects - Subjects may be broadly classified according to sex, age (child/adult), and regional accent. For any given classification (eg adult male Cockney) there should be several subjects - forming a group. This allows tests for single-speaker training and multi-speaker recognition both inside and outside a group.

4.2.2 The Recognition Task

The difficulty of the recognition task should be pitched at a level where changes in performance can be easily seen. There is no point in making the task so easy that recognition is perfect, since any subsequent system improvements cannot be detected. Also for a given task, the nearer the system gets to "perfection", the greater the size of the database must become to detect any changes.

A recognition task can be made more difficult in a number of different ways, by:

1. Increasing the vocabulary size
2. Including words in the vocabulary that sound similar
3. Increasing the the number of words in the test utterance
4. Introducing words not defined during training
5. Changing the training task

There are two basic recognition tasks chosen for the first WISPA system strategies. The first is the recognition of 50 isolated digits (per subject) and the second the recognition of 100 groups of 3 connected digits (per subject).

Each task is performed on both a single speaker and a group of speakers, and uses vocabularies trained on a single speaker or group of speakers. The recognition task may or may not use a vocabulary which contains training utterances from the recognition subject. The experiments designed to test the performance of the system on these tasks vary in difficulty, but are adequate to give a positive measure of the system strategy. Appendix IV describes the WISPA recognition tasks.

4.2.3 The Training Task

We may execute a given recognition task using vocabularies that have been trained in different ways and so establish how the training affects the recognition performance. The training task specifies the method of training. The method of training may be varied either by selection of the training subject(s), or by the number of training utterances presented, or by a combination of the two.

The subject can be chosen as an individual, in which case his is the only voice used in training, or as a group, in which case utterances from each member of the group are used. WISPA uses vocabularies trained on 1,5 and 10 presentations of each digit from (a) individual subjects and (b) all subjects within a group (ie one vocabulary represents the whole group). Appendix IV gives the details of the vocabularies.

4.3 Testing the System

We illustrate the application of the WISPA performance measurement techniques by looking at four consecutive experimental runs. These runs are in fact executed to test three strategy changes made on the baseline system. Each run is defined in a "run information form" containing a record of the system variable file plus any changes to the programs, and a tabulated record of the results of each experiment for each subject and/or group. Appendix V gives a sample run information form.

The four runs may be summarised thus:

Run 1 - Baseline strategy: defined as a series of specific versions of each component subprogram in the system and system variable file "PARAM" version 2A (see Appendix VI). Database files generated by SID II.

Run 2 - System variable file altered (now version 2B) to reduce "clip threshold" from 3 elements to 2 elements (see 3.4.2.2.5). The purpose of this strategy change is to see the effect of clipping steady state phonetic features to a maximum of 160mS rather than a previous value of 240mS.

Run 3 - This involves a program change to introduce a "pruning" algorithm for use in training on multiple utterances. The objective is to monitor the probability of individual admissible sequence elements and physically delete or prune out those which fall below a fixed value of 0.4. The system variable file is maintained as version 2B.

Run 4 - Version 2B system variable file altered (now version 2C) to change insertion and deletion costs from 7,3 and 1 to 7,5 and 3 for Grades 1,2 and 3 respectively. Here the object is to see the effect of favouring, in certain circumstances, a substitution (during matching) over an insertion and deletion.

Table 4.1 shows the word recognition results obtained on the four runs, for each experiment. Experiments 1 through to 5 were executed on 3 separate individuals in a group and their results averaged, while experiments 6 through 10 were performed on the same three individuals as a group. The recognition task for these runs uses a continuous speech algorithm which is not pre-warned of the number of digits to expect in the phrase.

The strategy changes between runs may be speculative in that they are done simply to see the effect of some change, or more often are performed as a result of a detailed study of the training and recognition of individual words. In the previous example, run 2 tested a speculative strategy change because there was no evidence that a lowering of the "clip threshold" would improve recognition. However, if the change did not impair performance then it could be used to improve the process speed (by reducing the average number of STM and ADM elements). Runs 3 and 4 on the other hand, tested strategies

that were selected as a result of intensive investigation into the process by which individual word templates evolved.

Experiment number	Run 1 (baseline)	Run 2 (new clip threshold)	Run 3 (pruned)	Run 4 (new costs)
01	84.67	91.33	91.33	94.67
02	93.33	87.33	92.67	96.67
03	93.33	91.33	94.67	94.67
04	67	67	67	75
05	63	62	77	83
06	88	88	86	86
07	88	80	86	87
08	81	76	86	86
09	73	71	74	78
10	51	48	78	80

Table 4.1: Word recognition percentages for the first 4 runs.
(see appendix IV for definition of experiments)

CHAPTER 5

Resume, Future Work and Conclusions

5.1 Resume

The design and implementation of the WISPA system is only the the first stage in a program of speech recognition work designed to produce a fully automatic speech recogniser utilising the NPL Speech Input Device (SID). There are several more stages of development ahead before the system is ready for commercial exploitation, and each stage should bring improved performance. An important benefit of the implemention strategy of WISPA is that when the experimental system has reached the limit of its potential, its production viability is immediately apparent. This is because the development hardware - SID (II or III) and the LSI-11 microcomputer - is likely to be no more powerful than the production hardware. This contrasts with other research systems that work well on large mainframes often in many times real-time, but await technology advances before becoming viable.

Results with the baseline system are good (ie 97% WRR on isolated digits and 83% WRR on connected digits), bearing in mind the simplicity of the SID II device, upon which the initial trials are based. However, from results with

the earlier system (ie 98% WRR on connected digits) employing the PDP/8 and manual training, it would appear that there is still some performance potential in SID II. While it cannot be claimed that the current WISPA recognition parameters (ie those in the system variable file) are optimal for best performance, it is felt unlikely that their variation can significantly make up the 15% shortfall in WRR. Detailed examination of individual recognition attempts appears to show that there is a limit to which the variation in utterances can be accommodated in a single template for a given word - even for a single speaker. It is believed that this can be overcome by a technique of horizontal adaptation to generate several templates per word where necessary. In fact, the manual training procedures for the PDP-8 system capitalise on the use of multiple-template words. Horizontal adaptation is discussed in the section on future work.

The technique of rationalisation which is used to produce features in a graded bitmap format is an excellent data structure for phonetic information and it appears that the resolution it provides is adequate for accurate word identification. The concept of feature grading can be modified and extended and we shall look at the possibilities for this later in the chapter, again, in the section on future work.

Many of the recognition inaccuracies that occur, stem from the incorrect feature classification by the preprocessor. This is due partly to the simplicity of SID II and partly to the inability of the software to fully compensate for mis-classifications (such as by the use of phonological rules). The SID II hardware is now superceded by the SID III device, which makes available considerably more speech information to the software than does its

predecessor. The "back-end" software for SID III must be significantly different in order to make best use of this extra information. As a result, most effort is directed towards the early commissioning of SID III into WISPA, rather than continuing improvement to the SID II back-end software.

Concept	Descriptive Section Nos.
1. Matching process identical for both training and recognition.	(3.3.1.3) (3.4.4)
2. Phonetic feature strings represented in rationalised (RSTM/RADM) and un-rationalised forms (STM/ADM). The rationalised form permits fast matching and access, while the un-rationalised form retains extensive information for long-term adaptation.	(3.3.1.6) (3.3.2.1-4)
3. The graded bitmap format for rationalised feature strings.	(3.3.2.2) (3.3.2.4) (3.4.3) (3.4.6)
4. The smoothing process.	(3.4.2.2.3)
5. The discarding of order information in the compression sub-process.	(3.4.2.2.5)
6. The truncation of feature strings in the clipping process.	(3.4.2.2.5)
7. The match matrix data structure.	(3.3.2.6)
8. The cost function as simple bit manipulations.	(3.4.4.2)
9. The process monitor.	(3.4.9)

Table 5.1: Important design concepts of the WISPA system.

In summary, we list (table 5.1) some of the important design concepts of the WISPA system that distiguish it from other research systems. While some of these features are specific to a system supporting the SID front-end, most are generally applicable to training and recognition systems.

5.2 Future Work

The development of the WISPA system has generated many ideas for improvements and adjustments to the base-line strategy. Here, we discuss five main topics for future work that are seen as the most obvious extensions to the current research. Some of this work is already in progress and hopefully will be implemented in future systems.

5.2.1 The SID III Speech Preprocessor

SID III represents a significant technological advance over SID II in three ways. Firstly, it measures similar information to a greater resolution and with greater accuracy. Secondly, it provides further data from the speech signal not measured by SID II. Thirdly, it is capable of operation in higher levels of noise and over lower bandwidth channels. Although it has these enhanced capabilities, it offloads some extra responsibilities onto the software. The future development of SID III software may be described under the headings of new phonetic features and software encoding

New Phonetic Features - Information provided by SID III relating to levels of correlation in the formant bands can provide valuable clues to transitional features such as stops. In addition, SID III provides data to enable the detection of nasels and voiced-fricatives. These extra features can be easily accommodated in the short-term memory and admissible sequence data structures by assigning, where necessary, unused class and type bits. Current work indicates that stops may be reliably detected by following the shape of the

level-of-correlation envelope. The stop itself forms an interesting feature because of its transient nature; it does not make sense to grade type information in stops in the same way as is done for the more steady features such as voicing. It is therefore likely that stops could be represented adequately by bits in the class word of a feature which need not be further qualified by graded types. The stop then, would merely be combined within an accompanying steady feature.

Software Encoding - The term software encoding is used to describe any feature extraction process made conditionally by the software, rather than explicitly by the hardware. In the SID II preprocessor, only voicing types are generated in this way - ie the voicing type is conditional upon the values of the formant frequencies, whereas all classes, plus the fricative and quiet types are selected by the hardware. The software for SID III, on the other hand, must first decide which class bit or bits need to be asserted. If it decides that voicing is present, for example, it has to establish the type by inspecting each formant band for frequency and level of correlation. Similarly, if frication is present, say, then the noise spectrum information must be interpreted. Indeed, there is no case where any one SID III feature unconditionally determines the content of the encoded sample.

5.2.2 Semantic Segmentation

The final decision process which extracts the key-word sequence from the match matrix is termed "semantic segmentation" (remember that semantic as used here refers to meaning at the phonetic feature level - see 3.4.7). The technique

adopted for this, and described by Armstrong [1980], traces all possible recognition routes through the matrix and sums the fits for each word in the route as well as certain fixed penalties for unused speech. In the absence of any semantic or syntactic information at the word level, the route with the lowest overall cost is chosen as the recognition answer. This method works quickly and accurately providing that the resolution of the match matrix is coarse and spotted words are sparsely distributed throughout the matrix. The first criterion is met by the fact that STM elements occur only once every 60mS on average. The second demands an intrinsically high word spotting capability, with good discrimination between words. The word spotting performance of the baseline system causes the segmentation process to be executed in about one tenth real-time for ten words. The processing time for segmentation is proportional to the square of the number of spotted words, so overall segmentation time can best be reduced by speed optimisation incorporating improvments in the word spotting performance. Two immediate possibilities for this are (i) re-coding the program in assembler language and (ii) including some node detection algorithm into the process. Node detection involves locating points in short-term memory where all recognition routes converge - this may be defined as any point not straddled by a spotted word - the segmentation process can then be restricted to operating between nodes. This method works well on the PDP/8 system.

5.2.3 Vertical and Horizontal Adaptation

Elements in time, such as those of the STM and ADM are denoted vertical elements, while elements in vocabulary space are described as horizontal. We touched on the subject of vertical adaptation in chapter 4 when referring to

the pruning of admissible sequences. The concept is one of removing redundant elements from an ADM when their probability of occurrance drops below some pre-defined threshold. Future work here is in the selection of age/probabilty criteria for best overall performance.

Horizontal adaptation is the automatic creation and deletion of ADMs. Whenever the training sub-system is presented with an utterance for an existing key, it assumes that utterance to have perceptively the same sound as other training utterances for the key. If it is not the same, either because a different word has be said - eg saying "nought" instead of "zero" - or because the speaker says the word in a different way, then the resulting template may be degraded so as not to fit any of the training utterances well. A solution to this is employ the distance measure between the existing ADM and incoming STM to decide whether the word can be reliably incorporated. If it cannot, then a new template can be contructed which maps to the same key. By recording the age and usage of each ADM for a given key, whole ADMs may be pruned out in the same way as their individual elements. The attraction of this system is that the creation of new ADMs is completely transparent to the user, who need have no prior knowledge of how a word "should" be pronounced.

5.2.4 Alternative Grading Schemes

The graded bitmap data struture is used to represent feature weights in order of value; types with a high probability or duration are flagged as grade 1 and those with a low probability or duration are flagged as grade 3. Used in this way, the grades order the data. This need not necessarily be the case, and one scheme under consideration is to use each bitmap to represent the

conformity of types to some particular criteria. For example, grade 1 could represent long duration types, grade 2 short duration types and grade 3 could indicate high probability types. Schemes of this nature yield even more data compression, and thus allow faster matching to be achieved.

5.2.5 Phonological Rules

The baseline WISPA strategy treats each phonetic feature independently without regard for context. However, since it known that the source data is speech, which obeys certain rules in the contruction of phonemes, it may prove useful to apply these rules to the system. As an illustration, there are some occasions when mis-classifications by the preprocessor can lead to linguistic nonsense - ie feature strings that reflect sounds which cannot be physically uttered. There are other times when the isolated training utterance is not representative of the word as spoken in continuous speech - such as the terminal "t" in "start", say, which is often present in all training samples but rarely there in phrases containing an embedded "start". To some extent, the lower fit values that would normally result from such changes can be overridden by applying some simple phonological rules to the either the matching or the construction of an ADM. The basic problem is one of identifying the rules as they might apply to the WISPA phonetic features. The solution to this problem will, no doubt, require a significant amount of research.

5.3 Conclusions

In chapter 1, we discussed how we set out to develop a speech recognition system in which the training process was to form the basis for subsequent recognition. This has been successfully accomplished on an experimental system which adheres to the basic philosophy of recognition by phonetic analysis. The fact that the system works at all (see chapter 4) justifies the approach taken, but more important, if one relates the performance of the system to its speed, cost and capability, it is further advanced than any other known to the author.

The question of how the human performs speech recognition has always been a subject of much debate, although it is now generally accepted that what he does not do is apply spectral pattern recognition techniques. This would be of academic interest save for the fact that, in the author's opinion, the use of pure pattern matching is rapidly reaching the limit of its power to solve the automatic speech recognition problem. This means that we must look elsewhere for solutions, if more global systems, for instance for use with many speakers, are ever to be developed. Intuitively, it seems wrong not to make optimal use of the fact that the source data is speech, and apply our knowledge of the construction of the speech to its recognition. Furthermore, if we can apply our understanding of the way the human learns to recognise speech, then so much the better. WISPA is an attempt to do this by providing a framework to test certain training and recognition hypotheses related to a model of the human. Results with the system appear to confirm that the model is a "good" one which will provide a sound basis for future recognition systems.

CHAPTER 6

References

6.1 Abbreviations

Throughout the list of references the following abbreviations are used for the reference author's affiliation:

BBN	Bolt, Beranek & Newman
CMU	Carnegie-Mellon University
IBM	International Business Machines
JSRU	Joint Speech Research Unit
MIT	Massachusetts Institute of Technology
NEC	Nippon Electric Company
NPL	National Physical Laboratory
RADC	Rome Air Development Centre
SCRL	Speech Communications Research Laboratory
SDC	System Development Corporation
SRI	Stanford Research Institute
TTI	Threshold Technology Incorporated
UCSD	University of California at San Diego

6.2 References

Armstrong M R (City University, London) 1980
"Semantic Segmentation of Speech Data"
Final-Year Project Report for Computer Science Degree, City University, 1980

Bahl L R et al (IBM) 1979
"Recognition Results with Several Experimental Acoustical Processors"
Proc. IEEE Int. Conf. Acoustics Speech & Signal Processing,
Washington, 2-4 April 1979

Baker J (IBM) 1979
"Performance Statistics of the HEAR Acoustic Processor"
Proc. IEEE Int. Conf. Acoustics Speech & Signal Processing,
Washington, 2-4 April 1979

Beek B, Neuberg E P, Hodge D C (RADC) 1977
"Assessment of the Technology of Automatic Speech Recognition for Military Applications"
IEEE Trans. Acoustics Speech & Signal Processing, Vol ASSP-25, No 4, Aug 1977, pp 310-321

Bisiani R (CMU) 1979
"Recent Improvements in the Harpy Speech Recognition System"
Proc. IEEE Conf. on Decision & Control, San Deigo, 10-12 Jan 1979

Bloch B, Trager G L 1942
"Outline of Linguistic Analysis"
Linguistic Society of America, Baltimore. Waverly Press 1942

Bloomfield L 1935
"Language"
Unwin University Press 1935, 1973 edition

Boddie J R (Bell Labs) 1977
"Speech Recognition for a Personal Computer System"
Byte, Vol 2, No 7, Jul 1977, pp 64-71

Bridle J S, Brown M D (JSRU) 1974
"An Experimental Automatic Word Recognition System - Interim Report"
JSRU Research Report No 1003, Dec 1974

Bridle J S, Brown M D (JSRU) 1978
"Connected Word Recognition using Whole Word Templates"
Proc. Inst. of Acoustics (GB) Conf. Nov 1978

Budgett E H (NPL) 1978
"An Investigation of Man-Computer Interaction Using an Automatic Speech Recognition System"
NPL Report COM 105, May 1978

CMU 1976
"Working Papers in Speech Recognition. 4: The Hearsay II system"
NTIS Springfield VA, Feb 1976, 169 pp

CMU 1977
"Speech Understanding Systems. A Summary of Results of the Five Year Reseach Effort at Carnegie-Mellon University"
NTIS AD-A049 288/4GA

Cohen P S, Mercer R L (IBM) 1975
"The Phonological Component of an Automatic Speech Recognition System"
In the book "Speech Recognition" edited by D R Reddy, Academic Press, New York, 1975

Dixon N R, Silverman H F (IBM) 1977
"The 1976 Modular Acoustic Processor (MAP)"
IEEE Trans. Acoustics Speech & Signal Processing, Vol ASSP-25, No 5, Oct 1977, pp 367-379

Erman L D, Fennell R D, Lesser V R, Reddy D R (CMU) 1976
"System Organisations for Speech Understanding Systems: Implications of Network & Multiprocessor Computer Architectures for AI"
IEEE Trans. on Computers, Vol C-25, No 4, Apr 1976

Flanagan J L (Bell Labs) 1972
"Speech Analysis, Synthesis and Perception"
2nd edition, Springer-Verlag, New York, 1972

Fourcin A J, et al 1976
"Speech Processing by Man and Machine - Group Report"
Dahlem Workshop on Recognition of Complex Acoustic Signals, 1976

Furui S 1980
"A Training Procedure for Isolated Word Recognition Systems"
IEEE Trans. Acoustics Speech & Signal Processing, Vol ASSP-28, No 2, April 1980, pp 129-136

Gerstman L J 1968
"Classification of Self-Normalised Vowels"
IEEE Trans. on Audio & Electro-Acoustics, Vol AU-16, No 1, March 1968, pp 78-80

Hayes-Roth F, Lesser V (CMU) 1977
"Focus of Attention in the Hearsay-II Speech Understanding System"
NTIS, Springfield VA, Jan 1977, 140 pp

Jelinek F, Bahl L R, Mercer R L (IBM) 1975
Design of a Linguistic Statistical Decoder for Recognition of Continuous Speech"
IEEE Trans. on Information Theory, Vol IT-21, 1975, p 250

Lea W A (SCRL) 1979
"Trends in Speech Recognition"
Prentice Hall, 1980

Lowerre B T (CMU) 1976
"Harpy Speech Recognition System"
Diss. Abst. Int. Pt. B - Sci & Eng, Vol 37, No 3, Sep 1976

Lyons J 1968
"Introduction to Theoretical Linguistics"
Cambridge University Press, 1968

Martin T B (TTI) 1976
"Programable Word Recognition Apparatus"
US Patent No 3,883,850 May 1975

Medress M F et al 1978
"Speech Understanding Systems: Report of the Steering Committee"
Artificial Intelligence, 9, 1978 (A short review of ARPA SUS Project)

Medress M F (Sperry Univac) 1979
"Speech Communications Technology at Sperry Univac"
Sperry Univac Defense Systems Division Report No. PX 12989, May 1979

Moore B C J 1979
"Introduction to the Psychology of Hearing"
Macmillan, London 1979

Moore R K (University of Essex) 1976
"A Descriptive Technique for the Analysis and Design of Speech Understanding Systems"
Ph. D. Thesis, EES-MMS-SPE76, Essex University, 1976

Moshier S (Dialog Systems Inc) 1979
"Talker Independent Speech Recognition in Commercial Environments"
50th Anniv. Celebration of Acoustical Soc. of America, MIT, Jun 11-15, 1979

Newell A, Simon H A (Carnegie Institute of Technology) 1967
"Overview: Memory and Process in Concept Formation"
In the book "Concepts and Structure of Memory" edited by B Kleinmuntz, Wiley, New York, 1967

Pay B E (NPL) 1978
"Analysis of Pilots' Voice Records for Automated Speech Recognition"
NPL DNACS Report No 2/78, 1978

Pay B E and Evans C R (NPL) 1978
"An Approach to the Automatic Recognition of Speech"
NPL DNACS Report No 7/78, Oct 1978

Pierce J R and Karlin J E 1947
"Information Rate of the Human Channel"
Proc. IRE, Vol 45, 1947, p 368

Rabiner L R, Rosenberg A E, Levinson S E (Bell Labs) 1978
"Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition"
IEEE Trans. Acoustics Speech & Signal Processing, Vol ASSP-26, No 6, Dec 1978, pp 575-582

Reddy D Raj, Erman L D, Fennel R D, Neely R B (CMU) 1976
"The Hearsay-I Speech Understanding System: An Example of the Recognition Process"
IEEE Trans. on Computers, Vol C-25, No 4, April 1976

Rengger R E (NPL) 1973
"Speech Controlled Equipment"
NPL Technical Memorandum TM 80, 1973

Rengger R E and Manning D R (NPL) 1973
"A Device to Enable a Computer to Accept Speech - SID Mark I"
NPL Technical Memorandum TM 79, 1973

Sakoe H (NEC) 1977
"System for Recognizing Speech Continuously Spoken with Number of Words Preselected"
US Patent No 4,049,913, Sep 1977

Sakoe H (NEC) 1977
"Automatic Continuous Speech Recognition System Employing Dynamic Programming"
US Patent No 4,059,725, Nov 1977

Sakoe H and Chiba S (NEC) 1978
"Dynamic Programming Algorithm Optimisation for Spoken Word Recognition"
IEEE Trans. Acoustics Speech & Signal Processing, Vol ASSP-26, No 1,
Feb 1978, pp 43-49

Schwartz R M (MIT) 1971
"Automatic Normalisation for Recognition of Vowels of All Speakers"
S.B. Thesis, MIT, June 1971

Silva G (SDC) 1975
"Automatic Speech Understanding Systems"
Computers and Humanities, Vol 9, No 5, Sep 1975, pp 237-244

Skinner 1977
"Speaker-Invariant Characteristics of Vowels, Liquids and Glides using
Relative Formant Frequencies"
J. Acoust. Soc. Am. 62, Supple. 1, 55(A)

Wagner R A (Vanderbilt Univ), Fischer M J (MIT) 1974
"The String to String Correction Problem"
JACM, Vol 21, No 1, Jan 1974, pp 168-173

Wakita H and Kasuya H 1977
"A Study of Vowel Normalisation & Identification in Connected Speech"
Conference Record, IEEE Int. Conf. on Acoust. Speech & Signal Proc.,
Siverman 1977, pp 648-651

Walker D E (SRI) 1976
"Speech Understanding Through Syntactic & Semantic Analysis"
IEEE Trans. on Computers, Vol C-25, No 4 Apr 1976

Walker D E, Paxton W H et al 1977
"Procedures for Integrating Knowledge in a Speech Understanding System"
Proc. 5th Int. Joint Conf. on Artificial Intelligence, CMU, Pittsburg,
1977, pp 36-42

Wolf J J (BBN) 1977
"HWIM, A Natural Language Speech Understander"
Proc. IEEE Conf. on Decision & Control, New Orleans, Vol 1,
7-9 Dec 1977, pp 560-565

Woods W A et al (BBN) 1976
"Speech Understanding Systems"
NTIS, Springfield VA, Aug 1976, 26 pp

6.3 Sources of Information on Automatic Speech Recognition

Books

J L Flanagan. Speech analysis, synthesis and perception. Second edition, Springer 1972.

> This is a basic textbook which still provides a good background to speech. It is rather heavy going in places.

W A Lea, Editor. Trends in Speech Recognition. Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

> This is a weighty compilation volume on the technology of speech recognition, with accounts of much existing work. It suffers somewhat from a lack of balance typical of compilations, in that some subjects are described in excessive detail and others dealt with more superficially. It is a good reference for getting a view of work over the last decade without having to read back-numbers of journals.

L R Rabiner and R W Shafer. Digital processing of speech signals. Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

> Intended as a teaching text for a one-semester course on speech processing. Reviews mathematical techniques such as Fourier analysis, Linear Predictive coding etc very much in the style of (mathematical) electronics engineers.

Journals

IEEE Transactions on Acoustics, Speech and Signal Processing. (ASSP). Published bimonthly by the Institute of Electrical and Electronic Engineers (subscriptions 445 Hoes Lane, Piscataway, NJ 08854, USA).

> This is the most concentrated journal especially on the engineering and mathematical (signal processing mainly) side. It also includes relevant book reviews and abstracts of forthcoming papers. Interesting papers in Volume ASSP-28, 1980, are:

Comparison of monosyllabic word recognition in continuously spoken sentences. S B Davis and P Mermelstein (p 357).

Application of dynamic time warping to connected digit recognition. L R Rabiner and C E Schmidt (p 377).

Voiced/Unvoiced/Silence discrimination of speech by delta modulation. C K Un and H H Lee (p 398).

The IEEE also holds conferences on ASSP, with published proceedings.

The Proceedings of the IEEE periodically devotes an issue to topics in speech and signal processing. These issues are useful because they give wide-ranging reviews - eg volume 63 no 5 (April 1975) on digital signal processing, and volume 64 no 4 (April 1976) on man:machine communication by voice.

Electronics International is a good source for general news and articles on speech. It also prints occasional survey articles, the most recent being:

B LeBoss. Speech I/O is making itself heard. May 22 1980, p 95.

W R Iverson. Military has its eye on speech. June 5 1980, p 93.

These are often dominated by speech output which is currently developing very rapidly.

Patents

Patents associated with speech communication are very numerous. Those related to recognition have shown a swing from electrical engineering towards mathematical methods and systems. It seems that the increasing software content of recognition systems will cause problems for patent protection. Patents are a better index of commercial activity than of new information, except in rare cases (eg the Nippon Electric Patents).

Manufacturers' literature

A checklist of known current suppliers of speech recognition equipment is:

THRESHOLD TECHNOLOGY Inc

1829 Underwood Boulevard,
Delran, New Jersey 08075, USA.

Threshold International Electronics Ltd
4 Walk Wood Rise
Beaconsfield, Bucks, UK
HP9 1TX
Telephone Beaconsfield (049 46) 4816

NIPPON ELECTRIC COMPANY

NEC Building, 33-1 Shiba-Gochome,
Minato-ku, Tokyo 108, Japan.

NEC America, Inc.
532 Broadhollow Road, Melville,
NY 11746, USA

VERBEX Inc (was Dialog Systems Inc)

32 Locust Street
Belmont, MA 02178, USA

INTERSTATE ELECTRONICS Corp

707 E Vermont Avenue,
Anaheim, California 92803, USA

AURICLE

Now a subsidiary of Threshold.

HITACHI

Not clear whether any equipment delivered so far.

Low cost and "hobby" suppliers

HEURISTICS Inc

900 N San Antonio Rd, Los Altos
CA 94022, USA

PHONICS Inc

POB 62275, Sunnyvale
CA 94087, USA.

CENTIGRAM Corp

Sunnyvale, California, USA

SCOTT INSTRUMENTS

815 North Elm, Denton,
Texas 76201, USA

VOICETEK

PO Box 388, Goleta, California 93017, USA

Consumer products

TOSHIBA has demonstrated voice-controlled TV and HiFi. Various toys are available.

Other firms and centres of research

IBM, at the Thomas J. Watson Research Centre, Yorktown Heights, New York, USA.

BELL TELEPHONE LABORATORIES, at their acoustics research department, Murray Hill, New Jersey, and also at Bell Northern Research, Montreal, Canada,

SPERRY UNIVAC DEFENSE SYSTEMS St Paul, Minnesota, USA

CARNEGIE-MELLON UNIVERSITY Pittsburg, USA.

NATIONAL PHYSICAL LABORATORY (NPL), Teddington, Middlesex, UK.

JOINT SPEECH RESEARCH UNIT (JSRU), Princess Elizabeth Way, Cheltenham, Gloucester GL52 5AJ, UK.

BOLT BERANEK AND NEWMAN (BBN) Cambridge, Massachusetts, USA.

HASKINS LABORATORIES Inc, New Haven, Connecticut, USA.

TEXAS INSTRUMENTS (TI).

ITT Defense Communications Division, San Diego, California.

SPEECH COMMUNICATIONS RESEARCH LABORATORY (SCRL) Santa Barbara, California.

SIGNAL TECHNOLOGY Inc, Santa Barbara, California.

PERCEPTION TECHNOLOGY Inc.

LOGICON, San Diego, California

NASA Ames Research Centre, Moffat Field, California.

ROYAL AIRCRAFT ESTABLISHMENT, Farnborough, Hants, UK.

MARCONI AVIONICS, Rochester, Kent.

ICL Ltd (at Stevenage, UK)

UK universities

University College London, Department of Phonetics and Linguistics. Roger K Moore.

University of Keele, Department of Communications and Neuroscience. W A Ainsworth.

Queen's University Belfast, Department of Electrical Engineering. R Linggard.

University of Sussex, Laboratory of Experimental Psychology. C J Darwin and Prof. H C Longuet-Higgins.

University of Leeds, Department of Linguistics and Phonetics. P Roach.

North Staffs Polytechnic, Department of Computing. P Green.

University of Aston, Department of Electrical and Electronic Engineering. M Ackroyd.

University of London, School of Oriental and African Studies. D C Bennet.

APPENDIX I

The Speech Input Device (SID) Mark II

by

R E Rengger and D R Manning

This appendix is a reproduction of the report by Rengger & Manning [1973], titled "A Device to Enable a Computer to Accept Human Speech: SID Mk I". The SID I is a prototype version of SID II and is identical in design and performance.

Introduction

As man evolved he developed speech as a very efficient means of communication based on his audio generator and receptor. It would therefore seem natural to allow a human to communicate with other intelligent machines, such as computers, by using his highly developed speech mechanisms. The Mark I Speech Input Device or "SID" is a prototype unit which enables human speech to be accepted by a computer.

Before describing the device in detail it will be useful to briefly explain the human speech generating system.

Air from the lungs is passed up the throat and into the mouth. The passage of air up the throat is controlled by the vocal chords, or larynx, which can restrict or completely block the air flow. When a person fricates, such as the "s" sound in sun or the "f" sound in four, the larynx is held open. When a vowel sound is generated, such as the "or" sound in four, the larynx vibrates and allows bursts of air to enter the mouth. In both cases the air entering the mouth cavities resonates so that oscillations, dependent on the cavities, are imposed on to the driving waveform. In the case of fricatives this is a steady stream of air, in the case of vowels it is the basic larynx pitch. When vowel sounds are being produced, movements of the tongue within the mouth cavity modify the size of the cavity and hence the resonating frequency. The tongue splits the cavity into two major areas so that there are predominantly two resonating frequencies for each vowel sound. Consonants are generally produced by transitional events. An example is the sudden stop in the voicing followed by the rapid energy build-up of a plosive sound "b" such as in the word "Abbey". Nasalisation further complicates the speech pattern by adding a by-pass to the resonating cavities.

SID MK. 1 - Principles of Operation

This prototype device is designed to detect some of the speech characteristics described previously, in particular, it detects the following:-

1. Vowel quality

The technique used is to monitor, in two channels, the resonant frequencies imposed onto the larynx frequency by the mouth cavities. These frequencies are dependent on the cavity size and not on the pitch of the larynx stimulating them, so they are similar in both male and female speakers. The reasons for choosing two channels and the special auto-correlation technique developed to analyse these frequencies are fully described in a Technical Memorandum by D R Manning and B E Pay entitled, "Speech Recognition Techniques".

2. Fricative quality

Using similar techniques to 1 above, the predominant frequency contained within a band of frequencies representing the fricatives is detected. This enables the separation of fricatives into two groups. A detailed account of the analyses of fricatives is contained in a Technical Memorandum by this author entitled "The Analysis of Fricatives in Human Speech".

3. Silence

The gaps of silence between words and also those within an utterance (glottal stops) may be monitored by determining insufficient energy in both the voiced and fricated frequency bands.

Sid continually monitors speech and reports to its output the existence of the above features as they occur. A block diagram of SID Mk.1 is shown in Figure 1 and there follows a detailed description of the component parts.

Transducer and AGC

The transducer is a dynamic microphone with a frequency range from 50 to 15000 Hz. This covers the spread of frequencies produced in normal speech. The features detected have a wide range of energy, the quietest being the fricative sound "f", the loudest being the vowel sound "ar". An automatic gain control, AGC is therefore incorporated to compensate, when necessary, for this variation. It has a sensitivity down to 300mV to "boost up" a typical f sound and rates of attack and delay of 1μS and 10mS respectively, the latter being chosen to be compatible with an average male larynx pitch.

Voicing, Fricative or Silence detector - VFQ

The VFQ monitors whether the speaker is uttering voiced or fricated sounds, or has stopped doing either. This enables speech sounds to be divided into two classes.

It consists of two bandpass filters, one centred at 300Hz and covering the range of frequencies used in voiced sounds, the other centred at 20KHz covering the frequencies outside the voiced band. The frequency response curves of these two filters are shown in figure 2A. The outputs from the two filters are full wave rectified and compared to ascertain which is the greater at any time. This comparison consists of a subtracter so that if the result is positive the sound is termed "fricative", if the result is negative the sound is termed "voiced".

Thresholds employing hysteresis are used so that if neither filter contains sufficient energy then silence can be detected. This hysteresis also overcomes noise in the system by using different amplitude thresholds for onset and decay of energy. The transfer characteristic between voicing and fricating is shown in figure 2B. The outputs are binarised and form three of the SID output data lines, see figure 1.

Preprocessing for the Voiced Channels

When the VFQ reports that the sound is voiced, the output from the AGC is fed to two bandpass filters, LF and HF, see Figure 3. These two filters, one centred at 900Hz, the other at 2.5KHz split the voiced sounds into two channels representing the lower and higher major resonating frequencies described earlier. Having achieved this separation the signals may be infinitely clipped without loss of data, so that the special auto-correlation technique used to analyse the frequencies present may be realised by a digital system.

Vowel frequency analysis

As the two voiced channels are similar, only the lower frequency channel will be described.

The binarised lower resonating frequencies obtained from the pre-processing are fed to a fully tappable delay line, see Figure 1. This delay line is clocked at 20KHz approximately 20 times higher than the

signal frequencies, so the data is not appreciably modified.

By selecting two tapping points along the line and correlating the signals at these points a waveform is produced in which the mark-space ratio varies with the frequency of the data signal. If this waveform is integrated over approximately a larynx period, 10mS, then a correlation curve A, is produced, see Figure 4B, which consists of a frequency dependent voltage. By selecting tapping points further apart a second frequency dependent voltage B, is formed such that its period of oscillation with frequency is exactly three times higher than that of curve A. When these two curves, A and B, are compared in a differential amplifier, figure 4A, working under saturating conditions the result is that the amplifier changes state at only one frequency within the range of interest. This is called the barrier frequency. The reason for using two correlation curves to produce a barrier is to enable the comparative measure between curves A and B to increase the rate of charge, with frequency of the barrier.

The SID uses 14 such barriers, 7 in the low frequency range of 200 to 800Hz and 7 in the high frequency range 800 to 3KHz. The actual barrier frequencies, and an example of the determination of the required tapping point separation is given in Figure 5. In order to reduce the complexity of the hardware only 8 correlator-integrator units were provided for each channel. This necessitated a compromise in the obtainable barrier frequencies as correlators had to be "doubled up" for more than one barrier. However when analysing speech the resolution required is never as high as that provided by the SID barriers, so the best barrier for a particular separation may be chosen from those provided.

Fricative analysis

When the VFQ reports the presence of a fricative the output from the AGC is infinitely clipped without further filtering. As the fricative classification is dependent upon the lowest frequency present, this technique is acceptable. This binarised signal is fed to a fully tappable delay line, clocked at 80KHz. Two barriers are set up (as described previously for voiced sounds) at 1KHz and 2.5KHz to separate "s" sounds from "f" sounds. An s has most of its energy above 2.5KHz while an f is much more broad band and so does not exhibit a peak above this frequency. No discrimination is made between other fricatives such as t, th, sh, ch, etc. so these will fall into one of the above groups. There are hence only two types of fricative reported to the output, namely "High" and "Low".

Barrier "Smoothing"

The mechanics of the mouth limit the rate at which vowel frequencies may change to something greater than 20mS per barrier, so to prevent noise spikes from reaching the output, each barrier signal is fed to a low pass filter, whose output is discriminated by a schmitt trigger containing hysteresis between its positive and negative switching levels. This removes all pulses less than 15mS.

Finally barrier "Lock Ups" are applied. As a barrier is based on correlation curves, see Figure 4B which give readings even when the input frequency is well outside the barrier range, then some control has to be applied to the data. This is achieved by locking each barrier to its high state when the next highest frequency barrier has been passed and is thus in its high state. A barrier may only be in a low state, therefore, when the signal frequency in the channel is truly below the barrier frequency.

Considering this another way; if the signal frequency is higher than say the highest "LF" barrier then by definition all the "LF" barriers must be in their high states.

Sid parallel output

There are 19 "bits" of information fed to the parallel output by Sid Mk.1:-

7 High Frequency voiced barriers

7 Low Frequency voiced barrier

2 Fricative groups

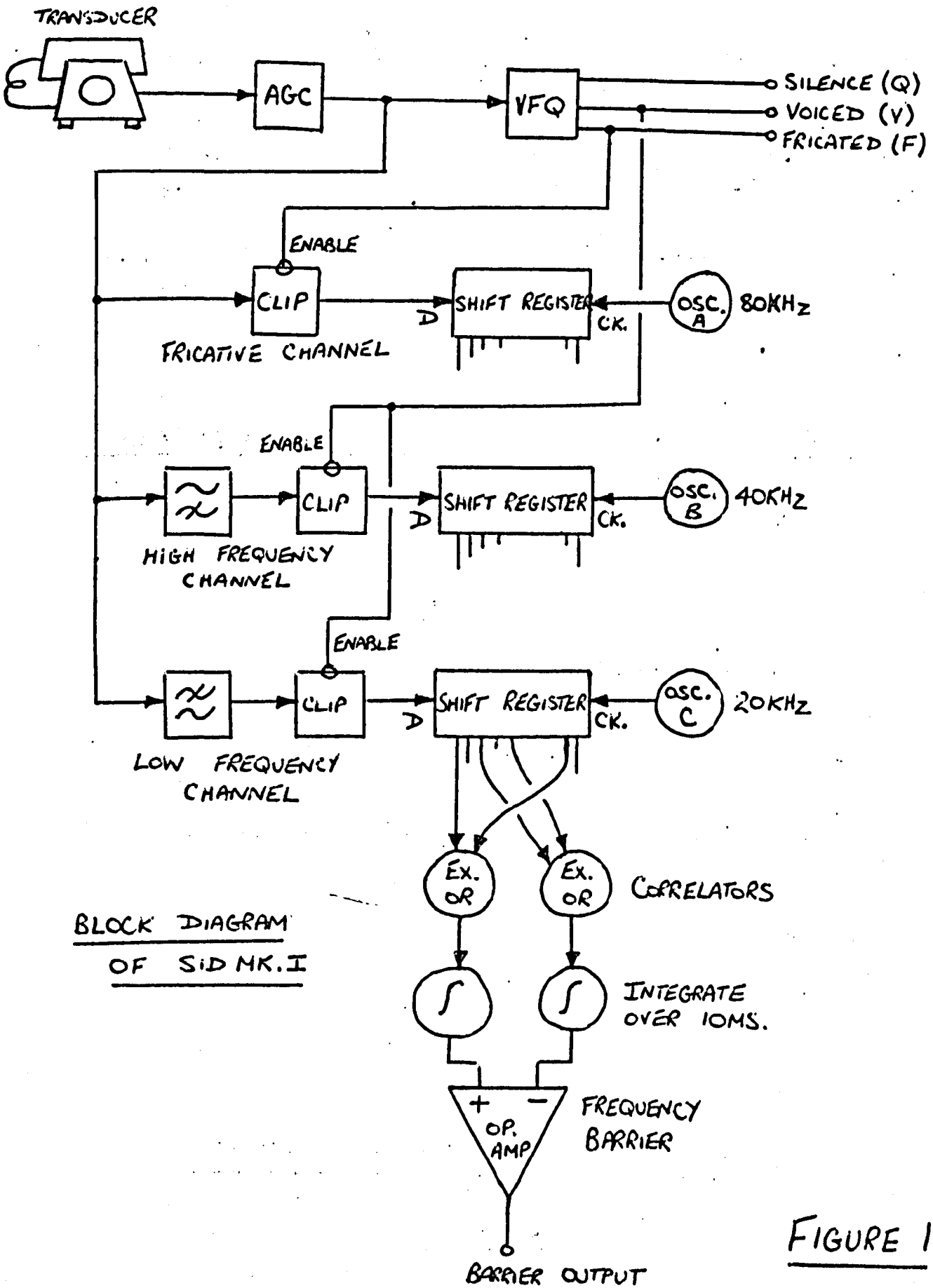
3 Quality types - voiced, fricated, silence

The output has the capability of delivering 40 bits so extensions to the system are possible. A time versus "bit" description of the word "fishes" is shown in figure 6. This was produced using a SID Mk.1 connected to a specially developed 40 bit parallel real time display. Normally a Sid would be connected via a suitable interface to the computer requiring a speech input, the necessary logic to interpret the data from Sid being contained within the computer software. "Sid" contains a clock, of period 10mS, which can be monitored by the computer to enable the sampling rate to be reduced, when required, to 100Hz. This limits the amount of data without losing any information as a human cannot modify his mouth shape this quickly.

Summary

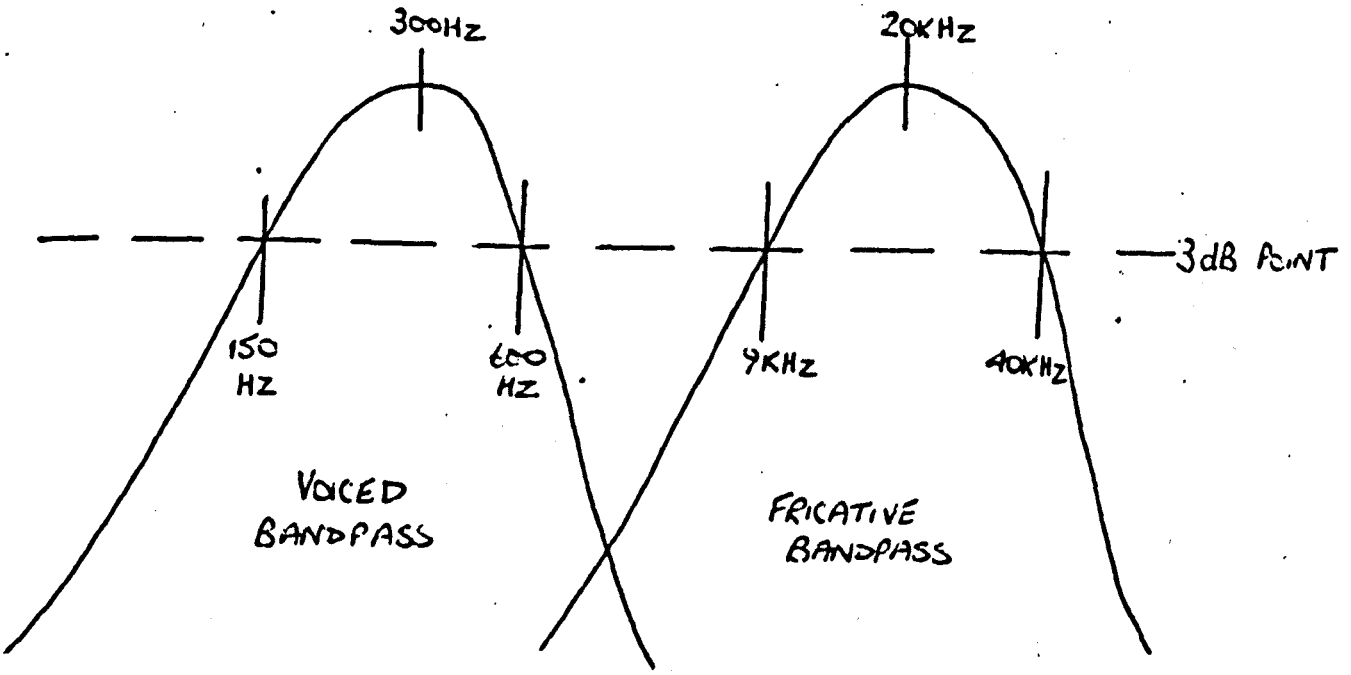
A SID Mk.I enables human speech to be converted into a form which a computer may accept. It works in real time and monitors speech features which are characteristic of the words being spoken. The features chosen are not modified by male or female voices. The device is not dependent upon the vocabulary that is being used. This dependence only occurs in the recognition logic software contained within the computer interfaced to Sid and so is easily modifiable. Sid is relatively simple when compared to the computer it is serving and so can be considered as a peripheral device.

A SID Mk.I forms the basis of a general purpose speech terminal for any computer.



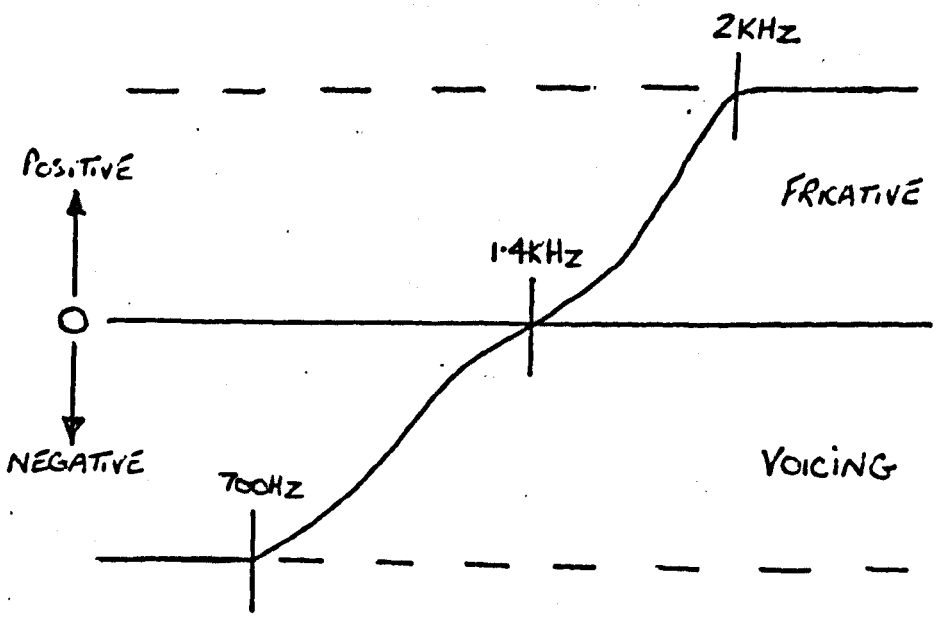
BLOCK DIAGRAM OF SID MK.I

FIGURE 1



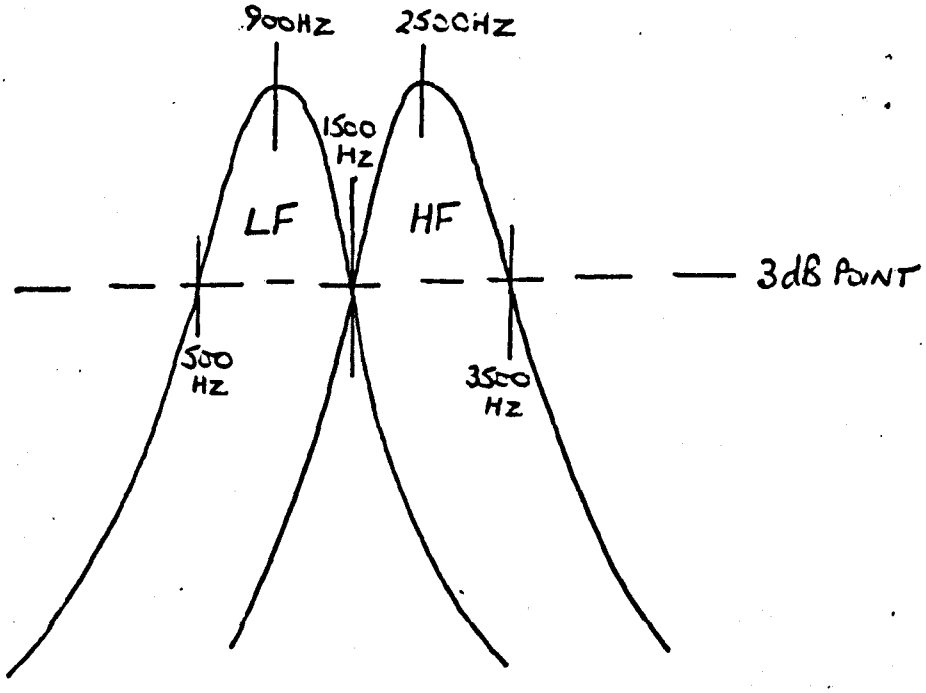
FREQUENCY RESPONSE CURVES FOR THE VFQ SEPERATOR

FIGURE 2A



FRICATING - VOICING TRANSFER CHARACTERISTIC

FIGURE 2B



FREQUENCY RESPONSE CURVES FOR THE TWO VOICED CHANNELS — HF AND LF

FIGURE 3

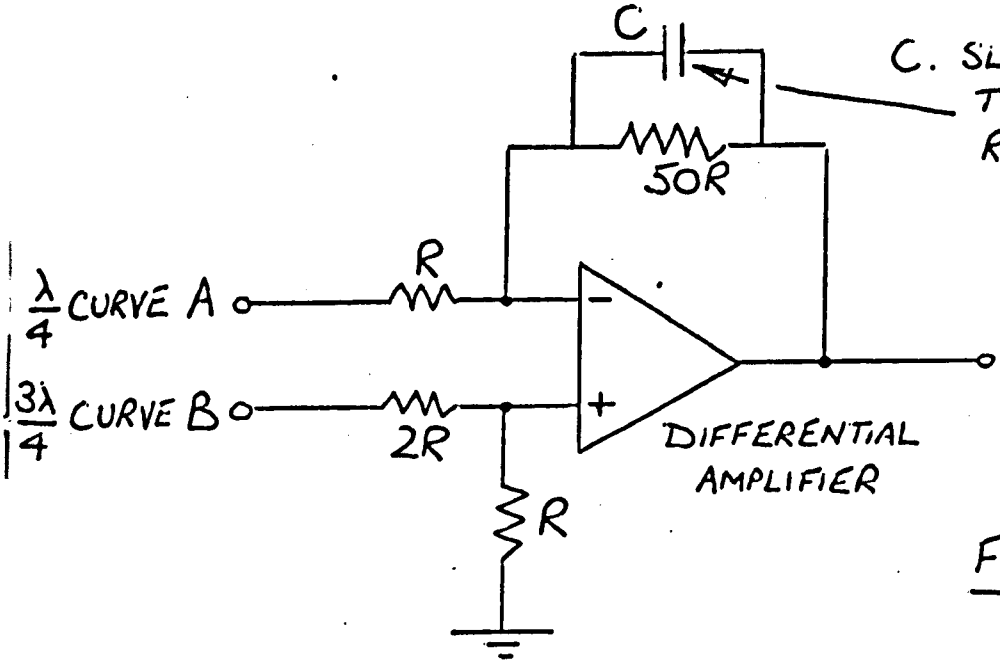


FIGURE 4A

A AND B ARE FREQUENCY DEPENDANT VOLTAGES DERIVED FROM THE CORRELATORS

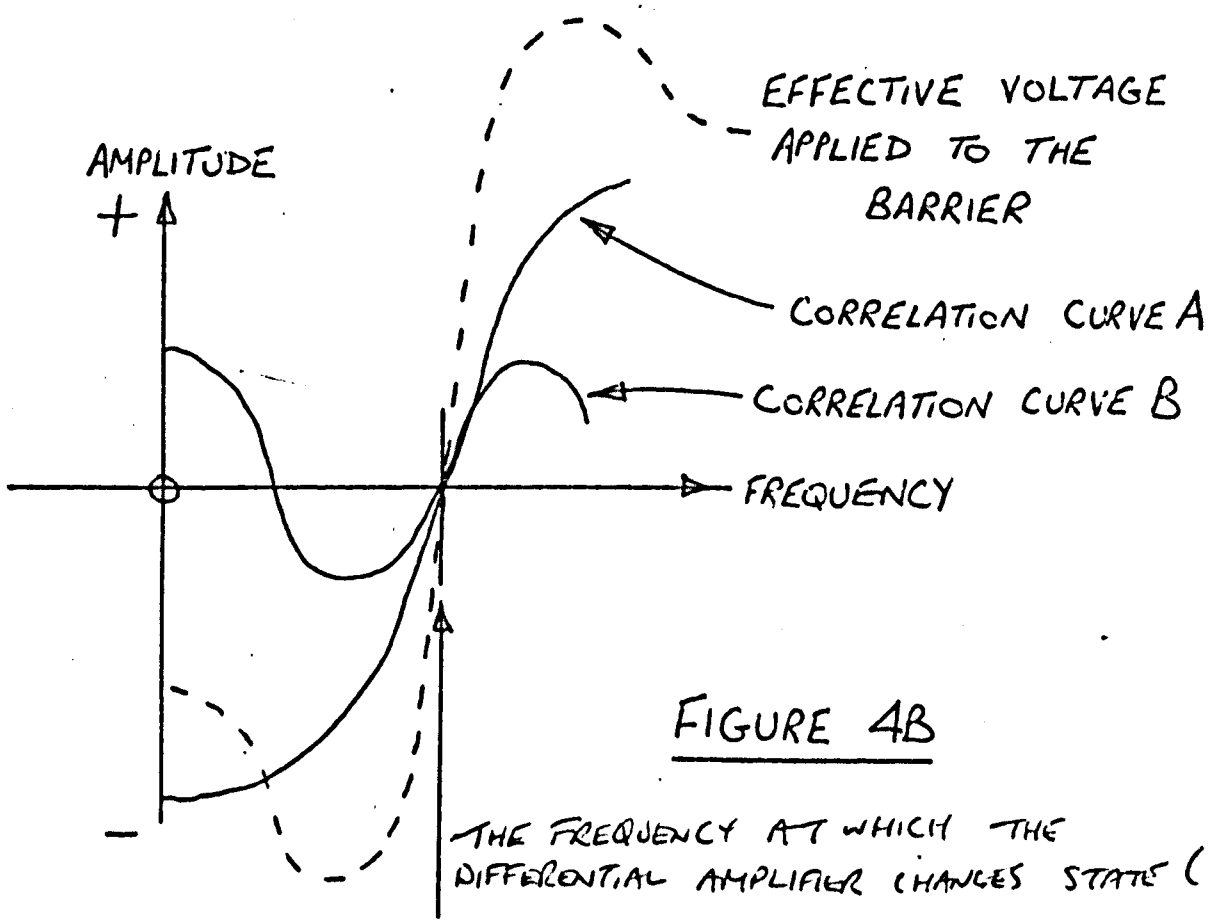
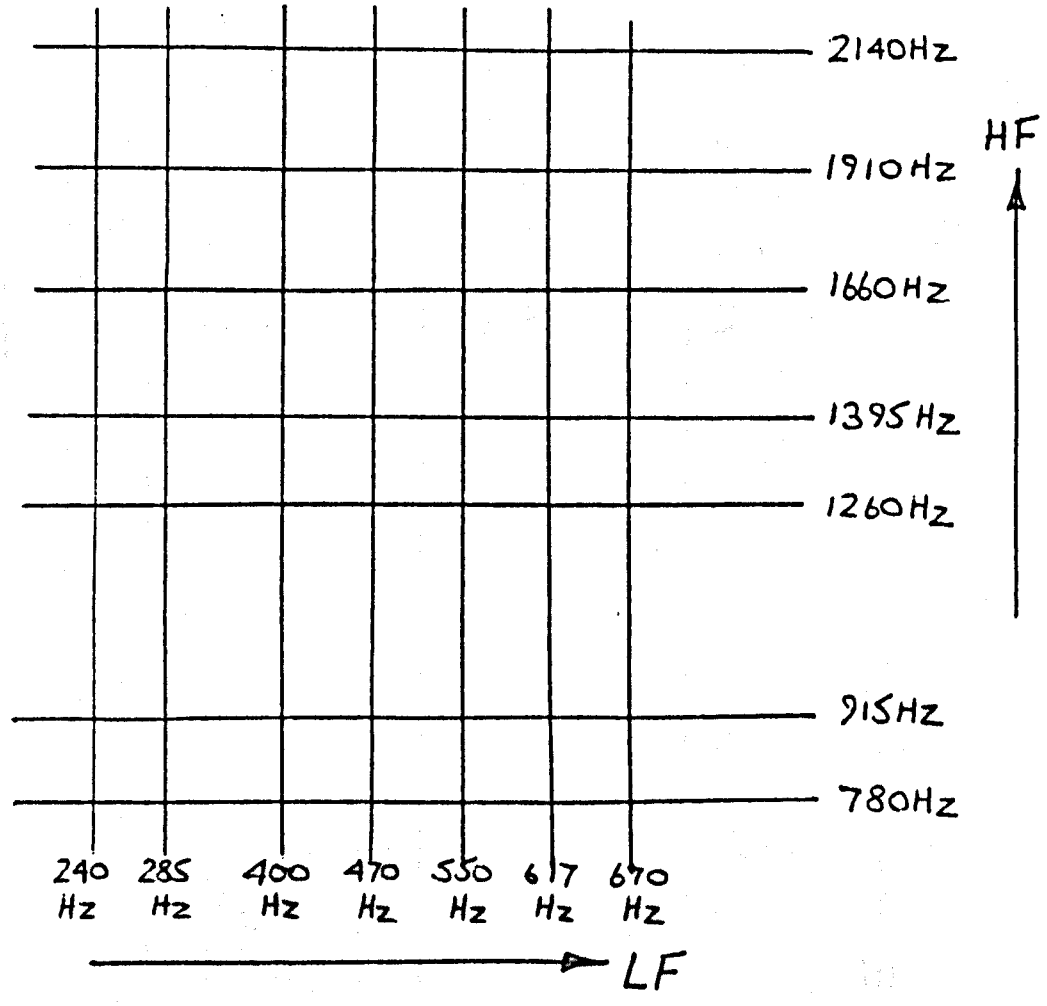


FIGURE 4B

SID MK I BARRIER POSITIONS



EXAMPLE OF CALCULATION OF TAPPING POINTS

A LOW FREQUENCY BARRIER REQUIRED AT 400Hz

$\lambda = 400\,Hz$, PERIOD $T = 2500\mu S$

C'K PERIOD FOR LF = $50\mu S$ (20KHz)

THE PERIOD EQUIVALENT TO $\frac{\lambda}{4} = \frac{2500}{4}\mu S$

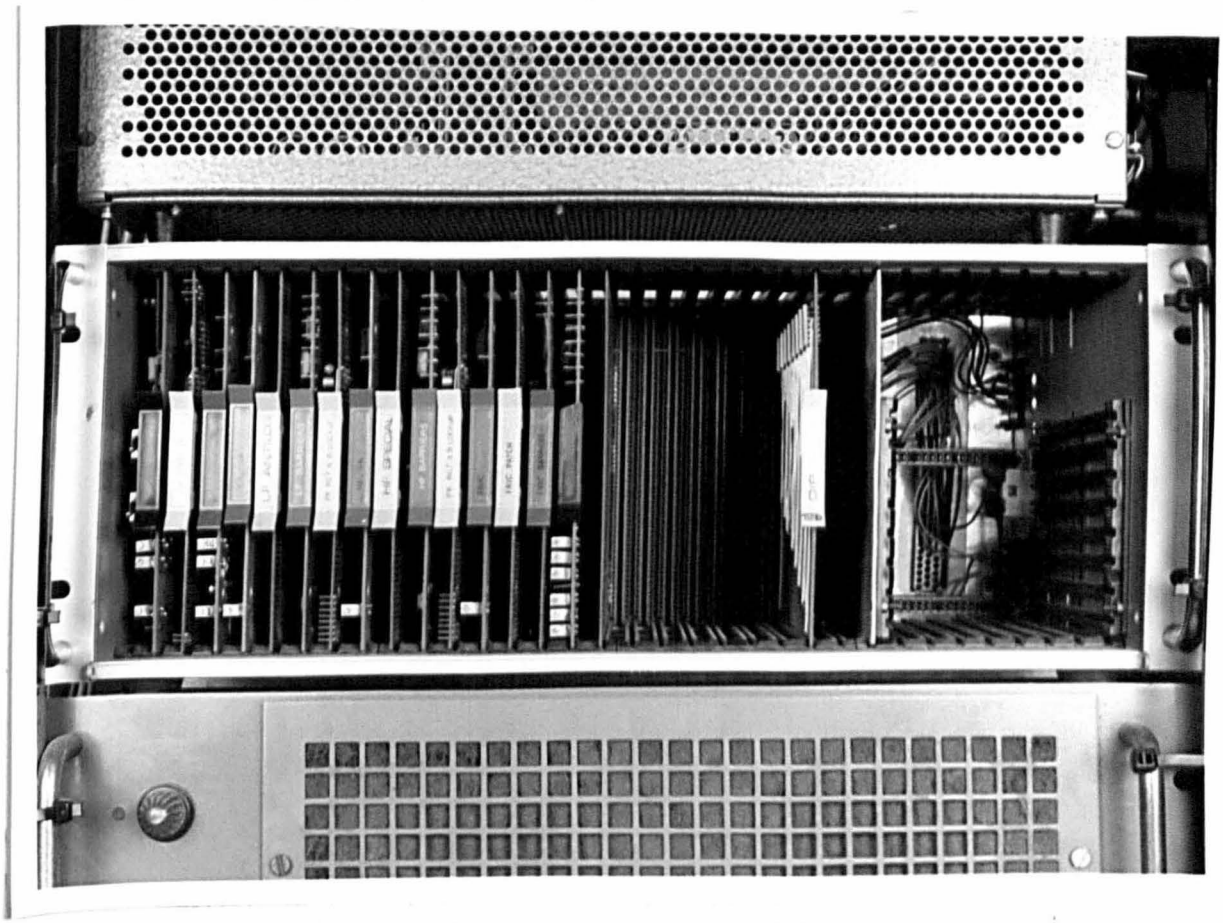
$\therefore$ DELAY TAP SEPARATION FOR $\frac{\lambda}{4}$ CURVE $= \frac{2500}{4.50} = 12.5$ "TAPS"

FIGURE 5

THIS IS ROUNDED UP TO 13 "TAPS"

FIGURE 6

Q - SILENCE
F - FRICATED SOUND
V - VOICED SOUND



The Speech Input Device Mark II

APPENDIX II

The Speech Input Device (SID) Mark III

by

R E Rengger

INTRODUCTION

The purpose of developing Automatic Speech Recognition devices is to give to a computer system the capability of interacting intelligently with a human, by responding directly to his or her spoken comments.

At the front end of any automatic speech recognition system therefore there has to be a transducing device that converts the pressure signal emitted from the speakers mouth into a form that may be accepted by the computer containing the recognition logic.

If the recognition logic resides in a digital computer, then the transducing hardware must produce a digital code that contains the information that conveys the meaning of the speech.

In its simplest form the transducing hardware need only be a microphone connected to an analogue to digital converter. As long as the converting rate is well above the highest frequency signal in the speech waveform and there are sufficient levels to accurately reproduce the relative amplitudes of the signals, no data will be lost. However this is a very inefficient technique as it requires extensive, high speed, computer memory, to deal in real time, with the vast amount of data produced. Most of which, in fact, is redundant to the meaning of the speech.

A more efficient system will intelligently reduce the amount of data sent to the computer by extracting from the speech signal ONLY the information that conveys the meaning of the speech, all other data being discarded.

A transducing system that meets this specification has been designed as part of the National Physical Laboratory study of automatic speech recognition techniques. The transducing hardware is called a Mark 3 NPL Speech Input Device or SID3 for short.

Our early work assumed that the task of discovering the parameters in speech that conveyed its meaning would be made easier if we studied speech utterred under "ideal conditions". The term "ideal" in this context meant the use of a wide bandwidth, high quality, studio microphone in a relatively quiet environment, connected to the analysing system by a high fidelity channel.

We have, however, found by experience that this is not the best platform from which to start as the analysing techniques developed for this special situation can not always be transferred to more natural conditions. To overcome this problem we now specify that any speech processing proceedure to be adopted, must be capable of working properly under the following "less than ideal" conditions:-

1. NOISY ENVIRONMENTS
   The analysing hardware must function consistently in normal room ambient noise and also when other sounds such as aircraft noise or equipment hum are present. It should be tolerant to signal to noise ratios typically as low as 10dB.

2. LOW QUALITY CHANNELS AND MICROPHONES
   Restrictions to the bandwidth of the microphone, or the channel connecting the microphone to the analysing hardware, must not effect the analysing hardware anymore than it would a human listening to the same speech. For instance, the public telephone frequency bandwidth of 300Hz to 3KHz should be tolerated.

3. UNCONSTRAINED SPEECH
   The system must consistently analyse speech whether the words are spoken in isolation or strung together as is more natural. Variations in accent between speakers should not cause any major differences in the data produced for similar words, although variations may occur when linguistic features are pronounced in different orders for similar words.

This "worst case" approach has resulted in the SID3 speech analysing hardware that extracts from a speech signal information relating to the meaning of the speech. The amount of data produced represents a considerable reduction compared to direct AtoD conversion and is also far more consistent throughout all of the above conditions.

SID3 contains a range of modules, each designed to analyse a particular part of speech. There are two types of modules. Firstly those that detect a specified speech quality and secondly those that describe various features of that quality.

A block diagram of SID3 is shown in Figure 1.

The SID3 hardware analyses an analogue signal that may contain speech, and produces continuously at its output 64 bits of digital information, that represent the way 16 speech parameters vary. The results of this analysis are tolerant to changes in the acoustic environment in which the speech is utterred and also to changes of speaker.

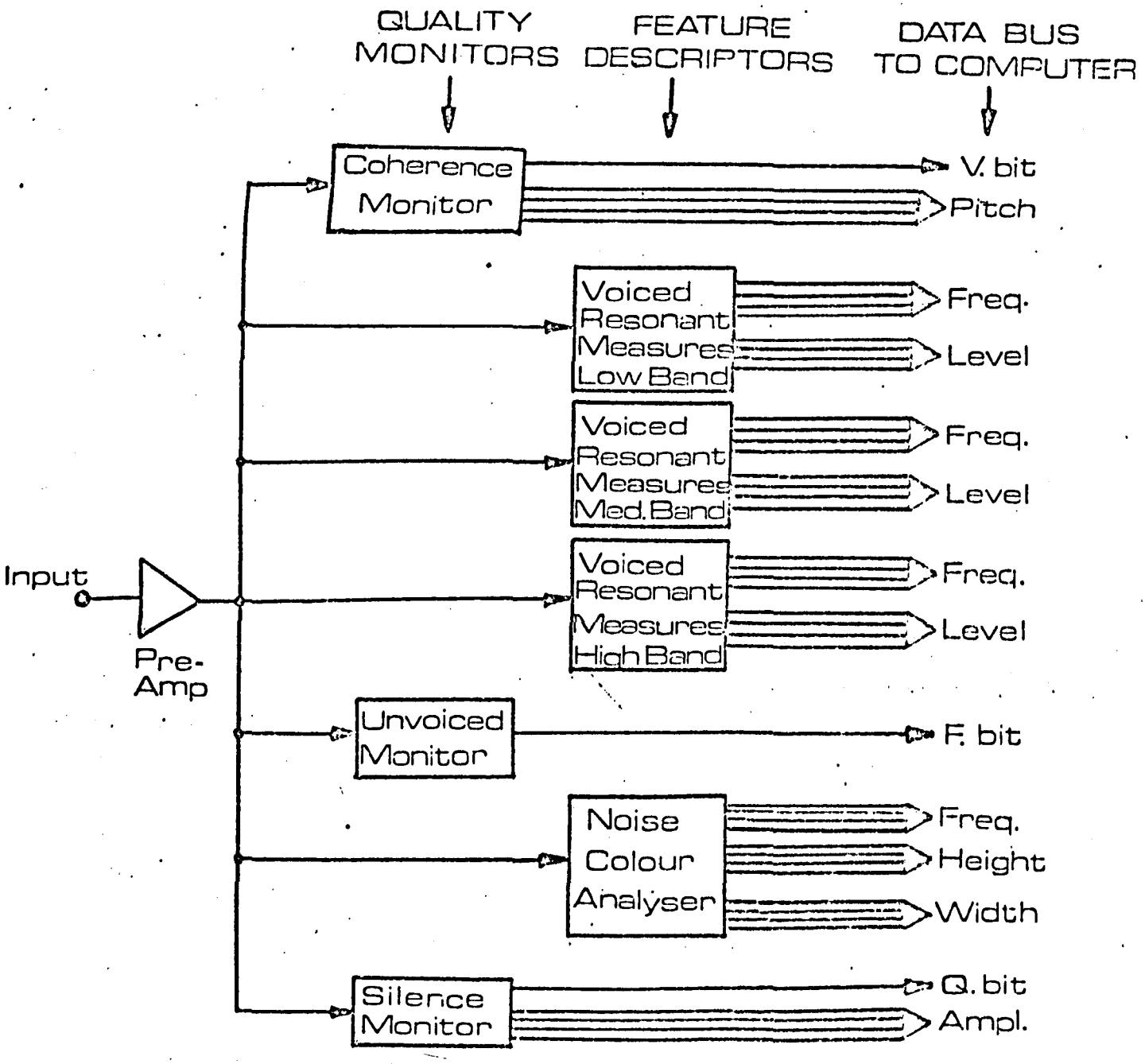
The 16 parameters chosen cover the following aspects :-

Speech quality in terms of voiced, unvoiced and no speech areas.

Voiced pitch, resonant frequency and level of resonance.

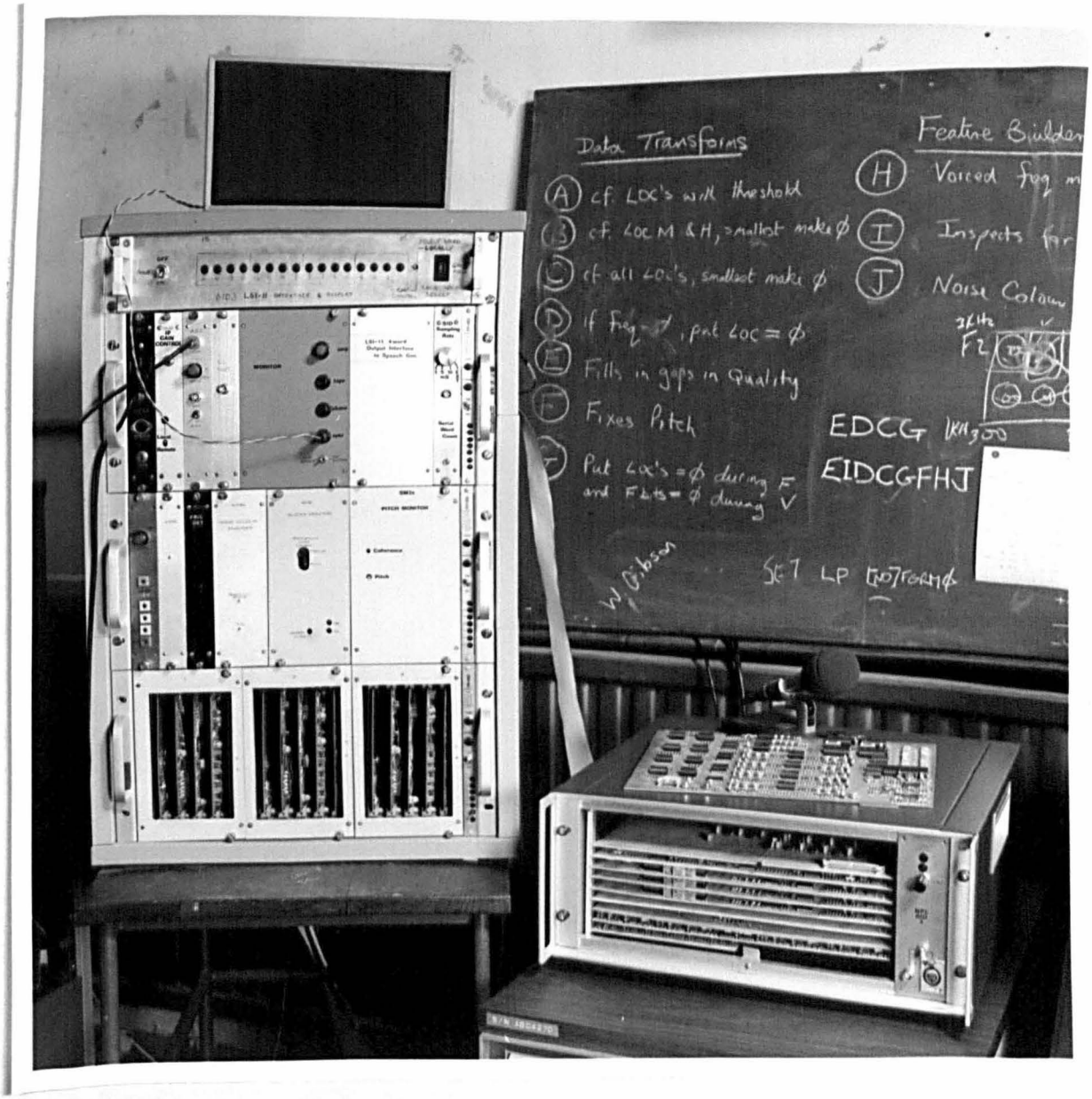
Unvoiced spectrum colour.

All the parameters are produced in parallel at the output of the device where they may be sampled by a computer. To recognise any speech in the analogue signal fed to SID3, the computer must support software containing recognition algorithms, having knowledge of the way the chosen parameters vary when speech is present in the signal.



BLOCK DIAGRAM OF NPL SID3

Figure 1



The prototype (left) and engineered mark III Speech Input Devices

APPENDIX III

Phonetic Terms

VOICING

Any sound which uses the excitation of the larynx is said to be voiced. When we produce voiced sounds, the larynx provides the "kick" to cause certain cavities in the head to resonate. The periodicy of the larynx is quite independent of the phoneme we are producing. The mean value for a male speaker is 125 Hz, for a female 180 Hz. It can be as high as 250 Hz in young children.

When we raise or lower the pitch of our voice (intonation) we are varying the pitch of the larynx. Accurate control of the larynx is required for singing.

The duration of some phonemes is long compared to a larynx period, thus the information within the period changes very slowly between successive periods.

Detecting the repetitious nature of the voice will classify it as a voiced phoneme. Although it is not necessary to measure the actual period, this parameter can be useful subsidiary information.

NASAL/VOWEL SEPARATION

The nasal passages consist of a considerable number of small cavities of differing shapes and sizes. These make a very broad band, low resonance system (low Q) which does little more than enhance the fundamental (larynx) period.

The mouth cavities, on the other hand, are nominally two in number and their shapes and sizes are under complete control by the lips and tongue. These resonant cavities are relatively narrow band and of high resonance (high Q). These resonances influence the waveform shape considerably, depressing the fundamental component.

Both nasals and vowels are members of the voiced class.

VOWELS

We produce vowels simply by controlling the shape and size of the mouth cavities, primarily by positioning the tongue. Since one mechanism controls both resonant cavities, the frequencies are inter-dependent.

The two resonances are referred to as the first and second formants and their values provide a good measure of the vowel sound, ie position of the tongue.

DIPHTHONGS

A diphthong is produced by setting the tongue to start at an initial vowel sound and then progressively moving it to a new position whilst voicing. The movement is fairly slow, about 150 to 250 ms from start to finish. For example, if we set up an "AR" sound and progressively move to an "EE" sound we produce the diphthong in the words "NINE" "FIVE" "NIGHT" etc.

A quantity of information perceived to be a specific vowel in one context can easily be perceived to be one feature of another phoneme (ie diphthong) in a different context.

FRICATIVES

Examples of unvoiced fricatives are S in "Sit", SH in "SHore" and F in "Forget". They are set up by positioning the tongue and lips, but unlike voiced sounds, air is exhaled while leaving the larynx static and open.

The effect is to restrict the opening of the mouth which is backed up now with a cavity driven by an incoherent source. This imparts a bandwidth limitation on an otherwise random noise.

The H in Happy is produced by having very little impedance at all.

TRANSITIONAL PHONEMES

These involve a rapid change from one state to another. They manifest themselves by the rapid removal, or conversely, rapid addition of a constraint at the bounds of a neighbouring phoneme. For this reason it is not possible to produce a transitional phoneme in isolation. These phonemes are generally referred to as "Stops" and are further qualified as voiced or unvoiced.

For example, a rapid release of the lips leading into the vowel "EE" is classed as the voiced stop "B". The same release but with no voicing is classed as the unvoiced stop "P".

Therefore the subtle difference between "Pail" and "Bail" is that in the latter case, the onset of voicing is immediate, whereas in the former there is a slight delay before voicing starts.

"T" and "D" are also a complementary voiced/unvoiced pair. Furthermore, with this class of stop, when the voicing is started well before the release, producing a nasal sound, and continued through and after the release, a "nasal to vowel stop" (nasal stop) is produced.

Whereas lips and tongue can stop or constrain air from coming out of the mouth, there are other pieces of apparatus that can stop air entering the mouth cavities. These are the epiglottis and soft palate. For example they are used to produce the G in "Gate" and K in "Kettle".

When voicing is sustained through such a closure the "soft g" in "Sing" is produced. The soft g, like "n" and "m" are all members of the nasal class.

SILENCE (QUIET)

When gaps of silence occur within the bounds of a word it is usually as a result of mechanical constraints and is therefore informative. They also help considerably in segmenting at phoneme boundaries.

Gaps of silence can occur at word boundaries, but with natural speech, this only reliably occurs when mechanical constraints demand it. Whereas a gap of silence always demarks a phoneme, it will only co-incidentally, in continuous speech, demark a word.

PHONEME DURATION

Any phoneme that is not a stop must last for sufficiently long time to be perceived to be non-transitional. On the other hand a sustained phoneme is perceived to be yet a different phoneme. Such a pair of phonemes are the vowels in BUN and BARN where qualitatively they are the same sound.

The duration of a	short	phoneme is	60 - 100 ms
" "	long	" is	150 - 250 ms

For a duration of between 100 and 150 ms both are equally probable. There is considerable variation in duration of all phonemes, and once again the context resolves any discrepancy.

VOICED FRICATIVES

This class of phoneme ie "Z" in "Zero", "V" in "Very", "TH" in "THen", is characterised as being a member of both voiced and fricative classes. To produce a "Z" sound for example we position our lips and tongue for a "S" sound and whilst producing the turbulent "S" fricative the voicing is simultaneously activated. "S" and "Z" are therefore said to be "cognates", or a complementary pair.

APPENDIX IV

WISPA Recognition Experiments

WISPA Recognition Experiments

SPEECH DATABASE

The Speech Database consists of a set of RT-11 files containing either raw SID-II data or raw SID-III data. Each file contains the SID data from a single utterance. Filenames are of the form:

	nnnVs.iii	for SID-II data
and	nnnXs.iii	for SID-III data

where "nnn" is a 3-digit decimal key, "s" is a 1-digit decimal session identifier and "iii" is the initials (up to 3) of the speaker. The ASCII file "DBASE.DEF" is a table showing the utterance stored in each file.

Each speaker is represented by 200 utterances using either or both SID-II and SID-III. These contain 10 groups of 10 digits spoken in isolation and 100 groups of 3 digits spoken continuously.

VOCABULARIES

The base vocabulary for recognition experiments consists of a single admissable sequence (template) for each digit. There are six different training methods used to generate templates. These may be divided into two classes: (i) templates generated by individual speakers and (ii) templates generated by groups of speakers.

Individual templates:

These are stored in files with names of the type :

	VOCnVI.iii	for SID-II templates
and	VOCnXI.iii	for SID-III templates

where "n" is 1-digit decimal identifier and "iii" is the subject's initials.

The training methods are as follows:

1	VOC1VI.iii VOC1XI.iii	Each template made from a single isolated utterance of each digit.
2	VOC2VI.iii VOC2XI.iii	Each template made from 5 isolated utterances of each digit.
3	VOC3VI.iii VOC3XI.iii	Each template made from all 10 isolated utterances of each digit.

Group Templates:

These are stored in files with names of the type:

VOCnVG.ggg for SID-II templates
and VOCnXG.ggg for SID-III templates

where "n" is a 1-digit decimal identifier and "iii" is a group identifier (eg MAL= all males on database, FEM=all females on database).

The training methods are as follows:

4	VOC1VG.ggg VOC1XG.ggg	Each template made from the first single isolated utterance of each digit from each individual in the group.
5	VOC2VG.ggg VOC2XG.ggg	Each template made from the first 5 isolated utterances of each digit from each individual in the group.
6	VOC3VG.ggg VOC3XG.ggg	Each template made from all 10 isolated utterances of each digit from each individual in the group.

COLLECTION OF DATA

The collection of data is effected by the program "STASH". For the generation of the speech database, the running of STASH is controlled by a series of indirect command files (ICFs), to ensure that filenames adhere to the protocol adopted.

The 200 utterances from each speaker are collected in 5 separate sessions. The filenames of the indirect command files follow the convention:

COLnV for SID-II data
and COLnX for SID-III data

where "n" is a 1-digit session number.

EXPERIMENTS

There are two sets of experiments. 1 through 5 are for individual speakers, 6 through 10 are for groups of speakers.

Individual Experiments:

EXPT01:	Training data:	VOC1V.iii and VOC1X.iii
	Training speaker:	individual "iii"
	Recognition data:	The last 50 isolated digits
	Recognition speaker:	individual "iii"
EXPT02:	Training data:	VOC2V.iii and VOC2X.iii
	Training speaker:	individual "iii"
	Recognition data:	The last 50 isolated digits
	Recognition speaker:	individual "iii"
EXPT03:	Training data:	VOC3V.iii and VOC3X.iii
	Training speaker:	individual "iii"
	Recognition data:	The last 50 isolated digits
	Recognition speaker:	individual "iii"
EXPT04:	Training data:	VOC1V.iii and VOC1X.iii
	Training speaker:	individual "iii"
	Recognition data:	100 groups of 3 continuous digits
	Recognition speaker:	individual "iii"
EXPT05:	Training data:	VOC3V.iii and VOC3X.iii
	Training speaker:	individual "iii"
	Recognition data:	100 groups of 3 continuous digits
	Recognition speaker:	individual "iii"

Group Experiments:

Let G be number of individuals in group.

EXPT06:	Training data:	VOC1VG.ggg and VOC1XG.ggg
	Training speaker:	group "ggg"
	Recognition data:	The last 50 x G isolated digits
	Recognition speaker:	group "ggg"
EXPT07:	Training data:	VOC2VG.ggg and VOC2XG.ggg
	Training speaker:	group "ggg"
	Recognition data:	The last 50 x G isolated digits
	Recognition speaker:	group "ggg"
EXPT08:	Training data:	VOC3VG.ggg and VOC3XG.ggg
	Training speaker:	group "ggg"
	Recognition data:	The last 50 x G isolated digits
	Recognition speaker:	group "ggg"
EXPT09:	Training data:	VOC1VG.ggg and VOC1XG.ggg
	Training speaker:	group "ggg"
	Recognition data:	100 x G groups of 3 continuous digits
	Recognition speaker:	group "ggg"
EXPT10:	Training data:	VOC3VG.ggg and VOC3XG.ggg
	Training speaker:	group "ggg"
	Recognition data:	100 x G groups of 3 continuous digits
	Recognition speaker:	group "ggg"

RESULTS OF EXPERIMENTS

The results of each experiment are stored in an ASCII file with a filename of the form "EXPTnn.xxx", where "nn" is a 2-digit experiment identifier and "xxx" is the individual or group identifier. The file contains the filenames of each utterance used in recognition together with the recognition result as a key-string and associated penalty.

A FORTRAN program called "STAT" operates on each "EXPTnn.xxx" file to produce a complementary ASCII file named "STATnn.xxx". This contains a confusion matrix and the percentage of correct, incorrect and rejected KEYS. In the case of continuous groups of digits, information to construct the confusion matrix is obtained by executing a string-matching algorithm to isolate the recognition performance of keys within the string.

Note that the general filenames "EXPTnn.xxx" and "STATnn.xxx" do not reflect the source of data (ie SID-II or SID-III). They are only intented to store results from a single "run" and should be renamed if it is required to store results from several runs.

EXECUTION OF EXPERIMENTS

The pre-defined experiments are executed by running a series of indirect command files, as follows:

VOCnVI.COM	generates	VOCnVI.iii
VOCnXI.COM	generates	VOCnXI.iii
VOCnVG.COM	generates	VOCnVG.ggg
VOCnXG.COM	generates	VOCnXG.ggg
EXPTnn.COM	generates	EXPTnn.iii EXPTnn.ggg

These command files are built by a FORTRAN program "EXPTst", where s=V for SID-II, s=X for SID-III and t=I for individuals and t=G for a group. In general "EXPTsx" must be executed for each new run.

EXAMPLE RUN

```
        R EXPTVI        !Build ICF for an individual using SID-II
        JPY             !individual is "JPY"
        DL1             !database device is "DL1:"
        @VOC1VI         !Build VOV1VI.JPY
        @VOC2VI
        @VOC3VI
        @EXPT01         !Do Experiment 1
        @EXPT02
        @EXPT03
        @EXPT04
        @EXPT05
        R STAT          !Get results for Experiment 1 on JPY
        01
        JPY
        DL1
        R STAT          !Get results for Experiment 2 on JPY
        02
        JPY
        DL1
        R STAT          !Get results for Experiment 3 on JPY
        03
        JPY
        DL1
        R STAT          !Get results for Experiment 4 on JPY
        04
        JPY
        DL1
        R STAT          !Get results for Experiment 5 on JPY
        05
        JPY
        DL1
```

APPENDIX V

A Sample Run Information Form

Run number 4

Date completed 6/1/81

SID Mark 2

PARAM Version number 2C

Changes to PARAM ..... Insertion & Deletion Costs
changed to 7 for grade 1
5 for grade 2
& 3 for grade 3

Changes to SID .....

Changes to RECOG specifics .....

Changes to TRAIN specifics .....

Changes to shared code ..... function IRCOST

Comments .....

Experiments undertaken:

	Initials/ Group identifier						
Expt	LRM	BEP	JPY	MAL			Averages
01	90	96	98				94.67
02	98	96	96				96.67
03	96	94	94				94.67
04	57/71	45/65	35/69				46/75
05	52/79	80/92	50/78				61/83
06				86			
07				87			
08				86			
09				52/78			
10				56/80			
11							
12							

Results .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

APPENDIX VI

A Sample Parameter (System Variable) File

```
Parameter data file for Recognition/Training software
WARNING: This is a fixed format file - do not alter the data layout
.
Version Number
2A
.
VARRAY: Array for Voicing Map (OCTAL)
     1     5     4     4     4     4    14    14
     1     7     6     6     4    14    14    10
     1     3     2   406   414   414    10    10
     1     3     2   402   412   410    10    10
   100   142    42   242   220   230    30    30
   100   142    40   240   260   220    20    20
   100   140    40    40    60    20    20    20
   100   140    40    40    60    20    20    20
.
FARRAY: Array for Fricative Map (OCTAL)
     1     2     4
.
.
VCODE: Bitmap code for Voicing
1
FCODE: Bitmap code for Fricative
2
QCODE: Bitmap code for Quiet
4
VBITS: Number of Voicing Type Bits
10
FBITS: Number of Fricative Type Bits
4
QBITS: Number of Quiet Type Bits
1
QTOUT: Timeout for Quiet
49
INTVAL: Period (x 20mS) at which SID sample are extracted from circular buffer
2
SIGSIZ: Size of Smoothing Buffer
8
SIGLIM: Type Inclusion Limit
1
THRSH1: Grade 1 Threshold
4
THRSH2: Grade 2 Threshold
2
THRSH3: Grade 3 Threshold
0
CLPTHR: Clipping threshold for Similar STM elements
3
```

```
.
.
IC1: Cost of Grade 1 Insertion  (OCTAL)
7
IC2: Cost of Grade 2 Insertion (OCTAL)
3
IC3: Cost of Grade 3 Insertion (OCTAL)
1
DC1: Cost of Grade 1 Deletion (OCTAL)
7
DC2: Cost of Grade 2 Deletion (OCTAL)
3
DC3: Cost of Grade 3 Deletion (OCTAL)
1
C1: CHANGE Sub-Cost 1 (OCTAL)
0
C2: CHANGE Sub-Cost 2 (OCTAL)
1
C3: CHANGE Sub-Cost 3 (OCTAL)
2
.
.
CLASS CHANGE COSTS (OCTAL)
V to F
17
V to Q
17
F to V
17
F to Q
17
Q to V
17
Q to F
17
PARAMETERS FOR SEMANTIC SEGMENTATION
MAXVAL
10
LIMIT
155
THRESHOLD
62
OVERLAP
2
.
DEFAULT PROCESS MONITOR MAP AND OUTPUT FILE SPEC
TRAINING MAP
3
RECOGNITION MAP
40
TT:
```