

Copyright © 1983, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

AN INTEGRATED CIRCUIT BASED  
SPEECH-RECOGNITION SYSTEM

by

H. J. Murveit

Memorandum No. UCB/ERL M83/80

14 November 1983

(cover)

AN INTEGRATED CIRCUIT BASED  
SPEECH-RECOGNITION SYSTEM

by

H. J. Murveit

Memorandum No. UCB/ERL M83/80

14 November 1983

ELECTRONICS RESEARCH LABORATORY


College of Engineering  
University of California, Berkeley  
94720

## An Integrated-Circuit Based Speech-Recognition System

Ph.D.

Hy Murveit

E.E.C.S.

Signature 

Chairman of Committee

A high performance, flexible and potentially inexpensive speech recognition system is described in this thesis. The system is based on two special-purpose integrated circuits that perform the speech recognition algorithms very efficiently. One of these integrated circuits is the front-end processor. It computes spectral coefficients from incoming speech, normalizes these spectra and finds the start and end of words in the speech. It transmits these spectra to a second integrated circuit that compares them with spectra from a set of stored word templates. The system can compare an input word with one thousand word templates and respond to a user within one quarter of a second. The system normally responds to words spoken in isolation from a particular speaker, however it can be used with connected speech as well as in a speaker independent manner. Modifying speech recognition algorithms to work with specially designed integrated circuits is shown to permit even high performance algorithms to be performed inexpensively. Using techniques such as these speech recognition devices should have a large range of applications within the next few years.

## TABLE OF CONTENTS

Introduction . . . . .	v
0.1 An Overview of Speech Recognition Systems . . . . .	v
0.2 Speech Understanding Systems . . . . .	v
0.3 Statistical Approaches . . . . .	ix
0.4 Pattern Matching Approaches . . . . .	xii
0.5 Cost Effective Implementations . . . . .	xiii
0.6 Some Commercial Approaches . . . . .	xvi
0.7 Outline of the Thesis . . . . .	xviii
Chapter 1. The Isolated Word Recognition System . . . . .	1
1.1 Introduction . . . . .	1
1.2 Definitions and System Parameters . . . . .	2
1.3 Use of the System . . . . .	4
1.4 Speech Recognition Algorithms . . . . .	7
1.5 Front End . . . . .	8
1.5.1 Spectral Analysis . . . . .	8
1.5.1.1 Bandpass Filter Spectral Analyzer . . . . .	9
1.5.1.2 Linear Predictive Coding . . . . .	15
1.5.2 Logarithmic Conversion and Energy Normalization . . . . .	18
1.5.3 Endpoint Detection . . . . .	20
1.5.4 Selective Downsampling . . . . .	23
1.6 Word Comparator . . . . .	24
1.6.1 The Distance Measure . . . . .	24
1.6.2 Dynamic Time Warping . . . . .	25
1.7 Recognition or Rejection Criteria . . . . .	30
1.8 The Training Algorithm . . . . .	30
1.9 IC Based System . . . . .	33
1.9.1 A description of the Integrated Circuits . . . . .	33
1.9.2 Speech-Recognition System Timing . . . . .	35
1.9.3 System Memory Requirements . . . . .	38
1.9.3.1 Filter Quantization . . . . .	40
1.9.3.2 Word Widths for Column Memory . . . . .	42
1.9.4 Features of the IC-Based System . . . . .	42
1.10 Current System Implementation . . . . .	45
1.10.1 The Distribution of Functions . . . . .	46
1.10.2 Filter Hardware . . . . .	48
1.10.3 Dynamic Time Warp Hardware . . . . .	49
1.10.4 Character Input and Output . . . . .	50
1.10.5 Summary . . . . .	50
Chapter 2. System Design Decisions . . . . .	51

2.1	Introduction .....	51
2.2	Testing Data Base .....	51
2.3	Front End .....	55
2.3.1	Selective Downsampling .....	55
2.4	Comparator .....	58
2.4.1	Path Selection .....	58
2.4.2	Path Normalization .....	62
2.5	Training Algorithms .....	64
2.6	Comparison with Other Systems .....	66
2.7	Quantization and Filter Variance .....	67
Chapter 3. Connected-Speech Recognition .....		69
3.1	Introduction .....	69
3.2	Connected-Word Algorithm .....	70
3.3	Performance .....	75
3.4	Syntactic aids .....	78
3.5	Hardware Support .....	80
Chapter 4. Speech Recognition Systems with Difficult Vocabularies .....		82
4.1	Introduction .....	82
4.2	Previous Work in the Area .....	83
4.3	Data Base Used for the Experiments .....	84
4.4	Performance of the Standard System .....	84
4.5	Performance of Feature Weighted Systems .....	86
4.6	Optimization of Templates .....	88
4.7	A Future Approach .....	92
4.8	Strengths and Weaknesses of the Current System .....	92
4.9	Conclusion .....	94
Appendix 1. ....		99
5.1	Speech Research Facility .....	99
5.2	Speech Input and Output .....	99
5.3	UNIX Based Speech Analysis and Graphics .....	100
5.4	Real Time Processing .....	101

### ACKNOWLEDGEMENTS

I owe many thanks to my parents, friends, and "friends at the lab" for helping me, supporting me, and in general making life nice. In particular I would like to thank my thesis advisor Bob Brodersen for his many hours of help and his willingness to sit down, roll up his sleeves, and help sort out the bugs in programs or wild ideas.

---

Research sponsored by the Defense Advanced Research Projects Agency  
under contract no. N00039-84-C-0507.

## INTRODUCTION

### 0.1. An Overview of Speech Recognition Systems

Everyday voice communication with machines will soon become a reality. Although devices that recognize and respond to speech already appear in the market place, speech recognition systems are not yet household items. This is primarily due to their cost and some performance inadequacies resulting from their lack of computational power. We feel that integrated circuit technology when properly applied to speech recognition can remedy these faults and propel us into the age of the listening computer.

Although the ultimate solution to the problem of machine transcription of speech is still far away, researchers have taken several different approaches to building useful speech recognition devices. These approaches restrict the scope of the speech recognition problem. Vocabularies used with speech recognition systems are limited and the syntax and semantics of what can be discussed with the systems is a subset of what can be said in natural speech. In addition some systems limit the number of different speakers who may speak to them and some require periods of silence between spoken words.

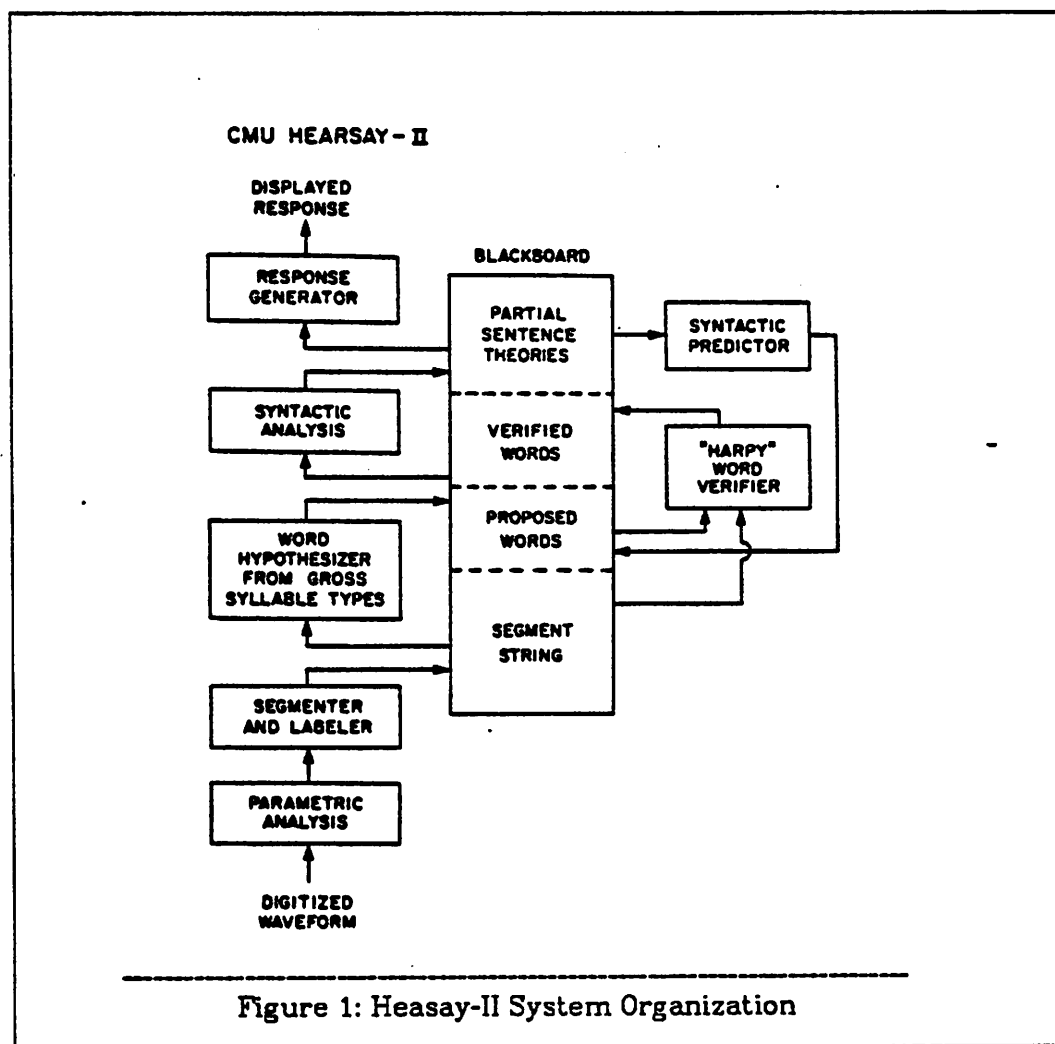
### 0.2. Speech Understanding Systems

One school of thought applies artificial intelligence techniques to the speech recognition problem.<sup>1</sup> They attempt to use all information about the speech recognition task, from acoustic analysis to general knowledge about the conversation, to decode the input signal. They restrict the input speech



to discuss a specific area, for instance a chess game, and to use a restricted vocabulary and syntax to discuss that area in order to simplify the problem.

As an example of this type of system consider the Hearsay-II<sup>2</sup> speech understanding system developed at Carnegie-Mellon University. It is a connected-speech recognition system with syntactic constraints.



This system had several "knowledge sources" that cooperated to make a recognition decision. The first knowledge source extracted linear predictive spectra, peak to peak amplitudes and zero crossing rates from input speech. The zero crossings and amplitude were used to divide the speech into short

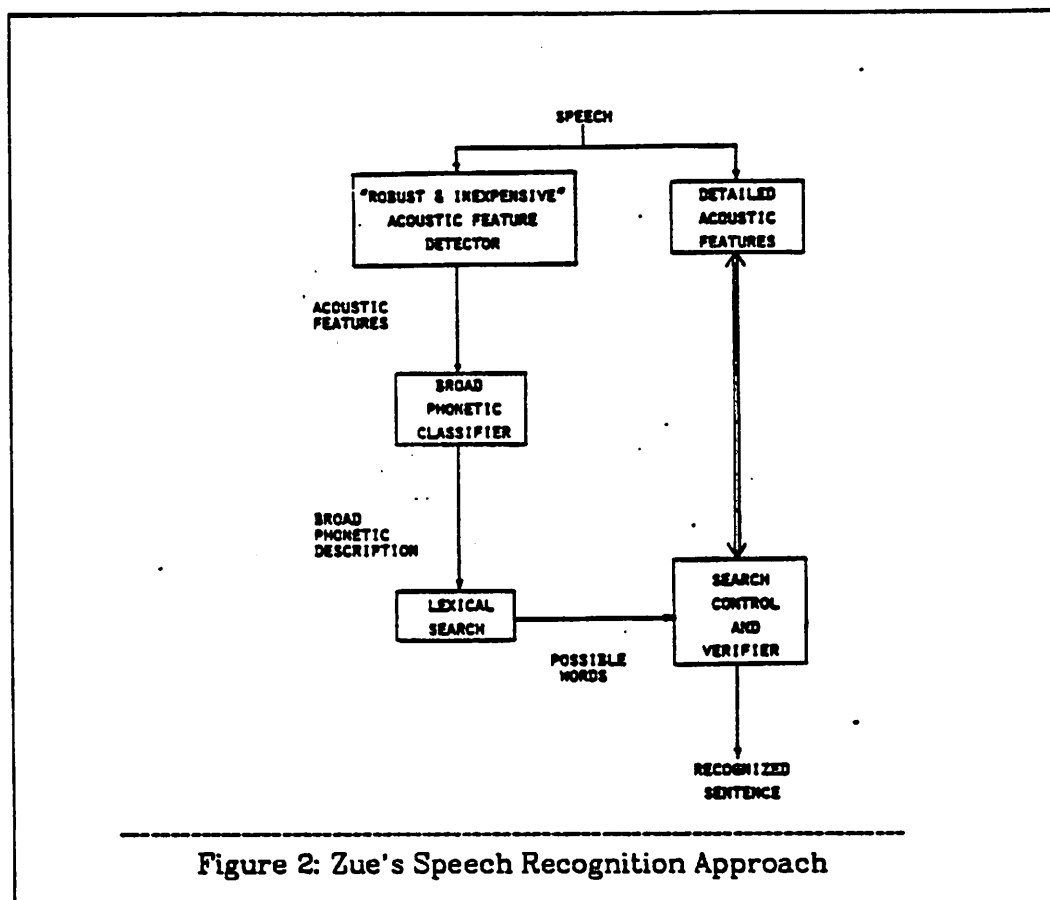
segments. These parameters were also used to roughly classify those segments (e.g. silence, fricative, vowel). A word hypothesizer then proposed words that matched a sequence of labels from the rough classifier. A word verifier then scored each hypothesis based on a linear predictive spectral matching scheme. An additional verifier checked for acoustic compatibility of adjacent words. A syntactic and semantic analyzer looked for high scoring words (or phrases) anywhere in the input and proposed additional words before and after these words that complied with the fairly rigid syntax and semantics of the Hearsay-II system. These sequences were then verified and the best scoring ones were treated as candidate partial sentences. They were examined by the syntactic knowledge source and additional words were hypothesized. This continued until the best scoring complete sentence was found.

The Hearsay-II system performed fairly well. Although only 77% of the test sentences had all their words recognized correctly, 91% were understood (the correct action was performed). However the system was slow. It executed 250 times real time using a PDP-KA10 computer (capable of executing 400K instructions per second).

Systems such as Hearsay-II attempt to "understand" the speech instead of recognizing individual words, and in this way deduce the meanings of ambiguous parts of the speech signal. In a sense this approach to speech recognition mimics the way people recognize speech. Systems built in this manner have tried to solve ambitious problems, but their performance has not been sufficient to make these systems as practical as those that have attacked simpler problems. The reasons for this are twofold. More research is required before we discover how people understand speech; thus these sys-

tems could use algorithmic improvements. Also enormous amounts of computational power are required by these systems to simulate machine intelligence. These computational resources and their economic implications make the limited achievements of the artificial intelligence based systems impractical to implement outside of the laboratory. However, continued further work in these fundamental areas is necessary towards the ultimate goal of a dictation machine.

Zue<sup>3</sup> and his students at MIT have recently begun a project similar to that discussed above but one that aims for reasonable computational complexity. Goals are to recognize a large vocabulary of isolated words or a small vocabulary of connected words. In this project the speech recognition system divides speech segments into "easy to recognize classes" (e.g. *strong turbulence, strong vowel, silence, voiced*). When there is insufficient information for making an informed decision then the decision is put off and thus some segments may initially be labeled as "don't cares". Once segments are classified then a list is made of all the words that could be constructed from these labeled segments. The system performs more detailed analysis on these words only. Zue stresses that this "late-binding" of decisions is important, both from a computational point of view and as a way of improving system accuracy. Should the initial labeling of word segments be accurate and the lists generated be small, this approach will be a successful one.



### 0.3. Statistical Approaches

Another approach to speech recognition is the statistical one.<sup>4,5,6</sup> Here parts of speech are modeled as random processes whose statistics can be estimated. Once the types of processes are decided upon and the statistics are gathered then recognition can begin. A set of words is decoded if the probability that those words generated the input speech (given the random process model of those words) is greater than the probability that any other set of words generated the input speech. This approach assumes less about our knowledge of the structure of speech than the previous one as it relies on measured statistics. However it is difficult to gather meaningful statistics about the input speech without long training sessions.

IBM<sup>4</sup> and Bell Laboratories<sup>5</sup> model words as being generated by hidden markov sources. That is, words are represented by finite state machines, where the transitions chosen to leave states are random variables, and the output (spectra) produced by the transitions are also random. Words are linked together by the syntactic structure of the system. The markov state transition random variables and output random variables are derived by using known words and sentences with the system. The random variables are chosen as ones that maximize the probability that the correct word models (those corresponding to the known input word) produced the known speech at the expense of incorrect word models. When unknown speech is input to the system, the system searches for the path through the markov chain that with highest probability produced the input sounds. The words corresponding to that path are recognized.

These approaches have shown promising results. In a 1980 experiment<sup>7</sup> IBM reported 2% sentence error on connected speech with a 250 word restricted syntax "Raleigh language". They also report a 3% error rate on a 1000 word vocabulary isolated speech recognition system. However the computational requirements are large for complex problems such as these. IBM reports 10-80 times real time processing on an IBM 370/168 computer. For smaller problems, such as isolated speech (the Bell Labs approach), less complex pattern matching techniques have performed slightly better than the statistical approaches. This is probably because it is not practical to gather the large amounts of data required to reliably estimate the statistics in Bell Lab's systems. In addition much ad hoc adjustment is needed to find the original structure of the markov model in these systems from which the transition probabilities are estimated. This is another source of error. A common criticism of the statistical approach is that these systems make

little use of what is known about the phonological and phonetic nature of speech. Rather they rely almost completely on the (usually insufficient) statistics they gather about speech.

An interesting recent approach in this area is the Carnegie-Mellon system FEATURE.<sup>8</sup> It tries to combine the advantages of the artificial intelligence and statistical approaches. It is a speaker independent system designed to recognize the alphabet, a difficult vocabulary. The system allows knowledgeable linguists to select quantitative cues or features that they consider important to distinguish words. Cues are also derived through statistical pattern clustering analysis. The speech recognition system evaluates these features and decides which to use and when to use them in the following manner. Statistics are gathered on a speech database about the distribution of these linguistically important features (e.g. bandpass energies during a vowel onset) in various words. The system estimates the distributions of these features and attempts to build reliable functions that discriminate between letters or sets of letters in the vocabulary. These features are combined in a tree format so that a complete recognition function is built. Difficult decisions are deferred while more obvious ones are made higher up in the tree (much like Zue's approach). Now given an input unknown letter the decision tree is traversed to produce the letter that most likely was just spoken.

The advantage of this system is that it examines only important cues for the decisions it is currently making. For instance if it is distinguishing between *b,d* and *p,t* it will only examine the cues important in that decision (the voicing or low frequency energy at the start of the word). This is opposed to pattern matching approaches (below) that consider all parts of a

word when making their distinctions, whether those parts of the word are important to the decision at hand or not. A disadvantage of this system is that it is ad hoc. The features are not automatically derived, and it is difficult to imagine doing so other for than the most basic features. Changing vocabularies would require major system redesign. However this system is reported<sup>6</sup> to perform well on a difficult vocabulary with only moderate computational requirements.

#### **0.4. Pattern Matching Approaches**

Pattern matching techniques have been used for commercial speech recognition systems<sup>8</sup> and have also been examined in research laboratories extensively.<sup>9,10,11,12</sup> These approaches keep at least one "sample" (or template) of each word they are to recognize. When an unknown word is spoken to them, it is compared to each of the sample words and the template that is most similar to the unknown word is recognized as the word just spoken. In order to recognize speech from any speaker many different templates of each word are required representing the different ways words can be pronounced. Although this has been done with promising results<sup>13,14</sup> most pattern recognition systems are designed as speaker dependent. That is, one must train the system to his voice before using the system by repeating each word in the vocabulary several times.

The advantages of the pattern matching approach include its relative simplicity and high performance for speaker-dependent isolated-word tasks. It is also very versatile; vocabularies and speakers can be changed easily. These points make this type of system the most useful of the approaches today.

A disadvantage is that computation requirements are high when sophisticated word comparison techniques are used. One such technique is the dynamic time-warping technique<sup>9,10</sup> described in this thesis. This problem is severe when many templates are used as in systems that recognize large vocabularies or that are speaker independent. However specially designed systems, such as the one to be presented in this thesis, can perform these algorithms in real time and for low cost. Another disadvantage of this approach is that it is difficult to extend this system for use with connected speech. Although, as will be discussed in the third chapter, connected speech is possible and useful with this system, the amount of word distortion tolerated is limited. This is because the template words used cannot take into account the coarticulation between words, and this is a major effect in connected speech. It is also very tedious to train such a system with large vocabularies (on the order of one thousand words) because all the words must be repeated several times. Further, comparisons between very similar words are unreliable. The pattern matcher does concentrate on the features that should distinguish close words. Rather it uses all the information in the words, whether that information is useful or not.

### **0.5. Cost Effective Implementations**

In order for a speech recognition system to be successful it must both perform well enough so that its users are satisfied and it must be economically feasible for its users. Because of the large amount of computation necessary for speech recognition systems to achieve high accuracies, past speech recognition systems have been based on large computers and have often had slower than real time response. However, in order to be acceptable to its users, a speech recognition system must have real time response.



Extra compute power has often been supplied by multiprocessor computer systems, special-purpose hardware and array processor technologies. All of these approaches are expensive and have forced the price of high performance speech recognition systems to be prohibitive. Doddington<sup>15</sup> in his evaluation of commercial speech recognition systems notes that the best performances obtained were those by systems made by Verbex Corporation and Nippon Electric Corporation both of whom used specially designed and expensive computers to recognize speech.

Recent progress in large scale integrated circuit (LSI) technology as well as progress in computer aided design (CAD) of LSI has introduced other possibilities for implementation of speech recognition systems. Now there exist powerful computer central processing units on single integrated circuits (IC's) such as the Motorola 68000.<sup>16</sup> There are also IC's designed specifically for signal processing that are even more promising for speech recognition than LSI microprocessors. An example of this is the Texas Instruments TMS320 digital signal processor.<sup>17</sup> With the advent of sophisticated CAD tools and LSI design techniques, it is also possible to design special-purpose IC's for applications such as speech recognition without high development costs or turn-around times.<sup>18,19</sup> Each of the above three techniques can be used to take advantage of the economics of integrated circuit technology to produce a system with high computational capability yet low component count and system cost.

The three applications of LSI to speech recognition systems discussed above range from a device that can be used for speech recognition as well as many other very different applications (68000), to a microprocessor that is intended for signal processing (TMS320), to a device built specifically to

implement a certain speech recognition algorithm. This distinction between generality and specific use is important. Advocates of generality claim that universal devices will be produced in larger volume and hence costs of these devices will be lower than devices built for specific problems. Further they claim that future modifications to systems can be accomplished with minor changes to software written for the microprocessors instead of major IC redesigns for special-purpose devices. However, general-purpose machines pay a price for their universality. Specific algorithms cannot be performed nearly as quickly on a general-purpose architecture. For instance in its product description Texas Instruments claims that the TMS 320 can be used to build a 40 word connected word recognition system. Using the same type of algorithms our special purpose IC set recognizes 1000 words (templates) and this limit is not as fundamental as theirs. By using an integrated circuit IC processes as sophisticated as Texas Instruments used for the TMS 320 (2.7  $\mu\text{m}$  versus 5  $\mu\text{m}$  channel lengths) our system could more than double its speed. Furthermore, the general-purpose integrated circuit based machines are much larger than IC's with specific designs such as ours. This will decrease the production yields of those machines and cause them to be more expensive. Finally with the advent of modular integrated circuit design techniques<sup>18</sup> and quicker integrated circuit fabrication turn around times modification of IC's is not a problem. Advances in algorithmic work can be quickly incorporated into special-purpose designs.

Not every speech-recognition algorithm can be implemented with special-purpose integrated circuits. Algorithms implemented with IC's need to be structured in a regular manner. The more complex the structure of an algorithm the more difficult the special-purpose approach and a point is reached when general purpose processors become more practical. For

instance pattern matching algorithms used in this thesis are more amenable to special purpose designs than the Hearsay-II algorithms that were described earlier.

### **0.6. Some Commercial Approaches**

Several other companies have taken one of the three approaches listed above to build high performance recognition systems at lower cost. Among them Intel corporation<sup>20</sup> has released a speech recognition board consisting of two Intel 2920 signal processor IC's to perform spectral analysis and an Intel 8086 microprocessor with much read-only and read-write memory to do the time-warp computations. They claim high accuracy recognizing up to two hundred words of isolated speech. A similar approach (perhaps using different algorithms) was taken by Votan Corporation<sup>21</sup> which built a speech recognition board based on bit slice computers. These two systems have the disadvantage of using components that are too general purpose for their needs. A system made of specially designed integrated circuits can have more performance in two to five components than these systems have with complete printed circuit boards. Thus it is likely that companies such as these will compose their systems from a small set of integrated circuits in the future.

Nippon Electric Corporation<sup>22</sup> (NEC) has developed a speech recognition pattern-matching integrated circuit that is functionally similar to the one we have made. The major difference is that theirs is based on a microprogrammed processor whereas ours is logic specifically designed for the task at hand. Although they use a processing technology superior to ours ( 2.5  $\mu\text{m}$  channel lengths) their computation rate is slower. They claim to be able to match 180 isolated word templates in real time and use a connected-word

recognition algorithm with forty templates. The 180 templates can be increased to 300 with pruning of unlikely candidates. Our IC's with similar algorithms can process 1000 template words with either the isolated or connected algorithms and without pruning. There are two major reasons for the disparity in speed. One is that ours is a special-purpose design with little overhead. The NEC microprocessor must "waste time" fetching instructions. Surprisingly, when implementing pattern matching algorithms used with speech recognition, the straight-forward design of the algorithm is actually smaller and simpler than the design of a microprocessor to perform the algorithm. Another difference in the approaches is that our circuitry begins to recognize speech as soon as it detects the onset of a word instead of waiting for the word's end as does NEC. This gives it more time for the recognition process. However due to this our system requires extra scratch memory (the column memory described in section 1.9.3) for the computation that is not required in the NEC approach.

Burr et. al. <sup>23</sup> at Bell Laboratories took a different approach to building IC's for speech recognition. They designed a circuit to compute a small part of the speech recognition system and replicated these IC's at all nodes in the algorithm. In this way an enormous amount of pipelining and parallelism was available with their approach. Their system is able to compare the input isolated speech with 10,000 word templates in real time. However they required hundreds of their IC's to perform this task. Thus their system is an expensive system for either a low end or a high end application. Either our system or the NEC system could be replicated several times to reach the throughput of the Bell Labs system with a much lower chip count. The effectiveness of the parallelism in this system is limited by the communication overhead between the processors.

## **0.7. Outline of the Thesis**

This thesis discusses a speech recognition system implemented with two special-purpose integrated circuits. In the first chapter the system is described in some detail. The algorithms used to recognize speech are spelled out as well as the details of the system implementation. The second chapter describes some experiments performed both to evaluate the system design and to choose from several algorithmic alternatives. These first two chapters discuss a speaker-dependent isolated-word speech recognition system. In the third chapter an extension to the system, connected speech, is introduced. It is shown that with only minor modification the system can recognize connected speech with reasonable performance. The final chapter discusses attempts to improve the system, why they were successful or unsuccessful. It comments about the effectiveness of the current system and suggests future directions for research in speech recognition.

## CHAPTER 1

### The Isolated Word Recognition System

#### 1.1. Introduction

Speech recognition has been a active area of research over the past few years. Several of the projects undertaken <sup>1,4</sup> have been very ambitious, both from the viewpoint of investigating new concepts in speech communications and in the amount of complexity in these systems. However high computational requirements caused by the complexity of these systems have caused them both to be very expensive to build and to have less than real-time response times. Thus these projects have been impractical to implement in environments outside of the research laboratory.

There have also been speech recognition systems introduced as products recently.<sup>15</sup> The designers of some of these systems, in order to make their products economically feasible, have been forced to scale down the performance of the recognition products. Other designers, while maintaining high performance rates, have had to sell their systems for very high prices. These factors combined have caused the speech recognition industry to not be as profitable a business as had been projected in the past.

We feel that enough is currently understood about the speech signal so that high performance devices can be built to recognize speech in meaningful tasks and that these devices can be economically feasible. Further we propose that these devices can aid in algorithm refinement and reevaluation for future research.

To demonstrate this we<sup>12, 19, 24, 25, 26</sup> have developed such a speech recognition system. It translates the words spoken to it into character strings by comparing those words with features of previously spoken words. In several ways this system is similar to other speech recognition systems described elsewhere.<sup>9, 10, 11</sup> However this system is unique in that it has the qualities listed below.

- **IC Implementation:** The system is based on two special-purpose integrated circuits that permit it to execute sophisticated speech recognition algorithms for large vocabularies in real time and with low system cost.
- **DTW Algorithm:** The system is based on Dynamic Time Warping (DTW) techniques which yield the highest recognition accuracy for most applications. In fact ways have been found to improve the performance of this type of system.
- **Large Vocabulary Size:** The speech recognition system can compare an input word with roughly one thousand template words and still maintain real-time response. This large vocabulary size can be expanded further by connecting several of the DTW integrated circuits in parallel.
- **Expandability:** Not only can the vocabulary size be expanded, but by a natural extension of the algorithms the system can be made speaker independent and can accept connected-speech input.

## 1.2. Definitions and System Parameters

Currently, in order to design machines that accurately recognize speech, restrictions must be made on the type of speech recognized by those systems. A common restriction is requiring the user to speak isolated words

to the system. In this case a user must separate his words with short pauses (typically 100-200 ms.). This restriction has two important effects. First of all it makes the jobs of finding word endpoints much easier. That is not to say that there is no problem finding word endpoints in isolated speech. In fact most of the mistakes made by isolated-word recognition systems are due to endpoint errors. Isolated speech also avoids the problem of coarticulation between words, which is the effect of having neighboring words influence the way a given word is pronounced. A system without an isolated-speech restriction is called a connected-speech recognition system.

Another restriction often placed on speech input is whether the system is meant to recognize speech from any speaker, a speaker independent system, or whether only one or a few different speakers can use the system, a speaker dependent system. In speaker-dependent systems, a user must train the system to his voice before he can use the system. Allowing only one speaker to use the system decreases the variability of the pronunciation of the expected input to the system. Of course even a single talker varies the way he speaks words a great deal.

Restricting the vocabulary size, the number of unique words that a speech recognition system can accept, and restricting the difficulty of the vocabulary are other ways to improve the performance of a speech recognition system. In fact the system's performance is more closely related to the difficulty of a vocabulary than to its actual size. For example the letter names of the English alphabet is a particularly difficult vocabulary for use with a recognition system, much more so than larger vocabularies with words that are more distinct as will be shown in the next chapter.



Some systems may also require that sequences of words spoken to them conform to certain syntactic conventions. Then syntactic information can be used to distinguish otherwise very similar words when one of those words is syntactically inappropriate.

Whatever the system type, in order for it to be useful it must be able to recognize speech in real time. In other words the delay between the time the user ends his word or phrase and the time the system acts on that speech should not be noticeable to the user. For interactive systems, delays should be less than about one quarter of a second.

There are several important ways one must evaluate speech recognition systems. The most obvious is the error rate, or the percentage of incorrect words in all the words recognized. Another important parameter is the rejection rate or the percentage of words spoken to the system that are ignored. A final evaluation criterion is the false alarm rate or the number of words that are recognized even though they were not spoken to the recognizer and do not appear in the recognition system's vocabulary. Such words may be in reality words that the system's user speaks to someone in the room (not meant to be recognized by the system) or even extraneous sounds such as a door slamming.

### 1.3. Use of the System

The speech recognition system has many of the restrictions listed above (mostly as options to improve performance). The following is a brief description of the way a user interacts with the system in an isolated-speech user-dependent mode.

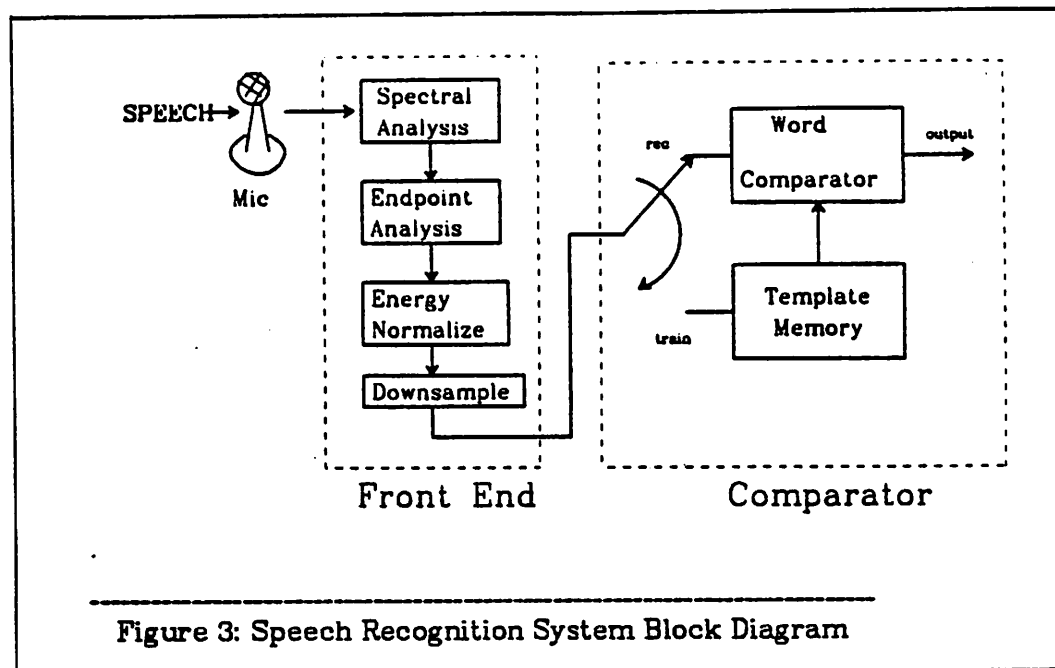
- **Choose a Task** The user picks the tasks he wishes to perform with the computer, such as integrated circuit (IC) layout or a data base

application. The tasks that are most useful with speech recognition systems are usually those where hands or eyes are already busy. IC layout is a good example because the IC designer uses his hands to point to objects on the screen of a graphics terminal while giving commands to the system by voice.

- **Pick a Vocabulary:** The user decides which words he wishes to use to converse with the application program he has chosen. He chooses words to fit the application such as *wire* or *John Smith*. The words are any strings that would ordinarily be typed. They can be simple words or phrases spoken as one word.
- **Check the Difficulty:** The user should check the vocabulary to see if there are words that are identical in sound (such as *right* and *write*) or that are unnecessarily close in sound (such as *nine* and *mine*). If such words are in the vocabulary, attempts should be made to replace them with other dissimilar sounding words. Of course sometimes similar words must remain in the vocabulary (such as when the task at hand is recognizing the alphabet). This may be done at this point or later during system use when the user notices that certain words are confused by the system. This can also be done automatically in the training phase described below.
- **Describe the Semantics:** The user must also indicate to the system what he wishes it to do after it recognizes a particular word. For instance he may tell it to transmit the ASCII erase character *control-H* every time it hears the word *erase*.
- **Train the System:** Once the vocabulary is chosen, the user must repeat several times each of the words he wishes to use to the speech recogni-

tion system. During this time the system searches for consistent pronunciations of each of the words in the vocabulary, and stores the features and the semantics associated with each of these words into a template word in a system dictionary that it maintains.

- **Use the System:** After training, one can use the system. Once started up it constantly "listens" for words spoken to it. That is the input energy level is examined to see if speech is in progress. An endpoint detection algorithm tries to extract words from background noise primarily by using these energy levels. When the system "hears" a word it compares that word to all words that are stored in its dictionary. When a template is found that is both more similar to the input than all other words and is also more similar to it than a predefined rejection threshold, that dictionary word is recognized and its associated semantic action is performed. If no words are found that satisfy these conditions, no action is performed and the input word is rejected.
- **Continued Use of the System:** Once the system has been trained it need not be retrained. The user can reuse the system whenever he likes. However, occasionally words are found that were originally mistrained and that are not recognized well by the system. These words can be selectively retrained. New words that were not part of the original vocabulary can be added. Adjustments can also be made to the rejection threshold so that a proper balance of rejected correct words versus accepted extraneous words is achieved.



#### 1.4. Speech Recognition Algorithms

The speech-recognition system is shown in the above figure. Roughly it can be divided up into a front-end subsystem and a word-comparison subsystem. The front-end subsystem extracts features from its input (the microphone) while the comparator compares those features to previously stored features.

The front-end samples its input time waveform regularly and computes frames or spectra of the time waveform every 10 milliseconds. While it is computing the spectra it performs endpoint analysis or decides whether a word is currently being spoken or not. It then energy normalizes the spectra in words by adjusting each one to have the same energy. Finally it downsamples the frames. That is it deletes frames that follow others without significant spectral change. The front-end outputs those normalized downsampled frames to the word comparator subsystem.

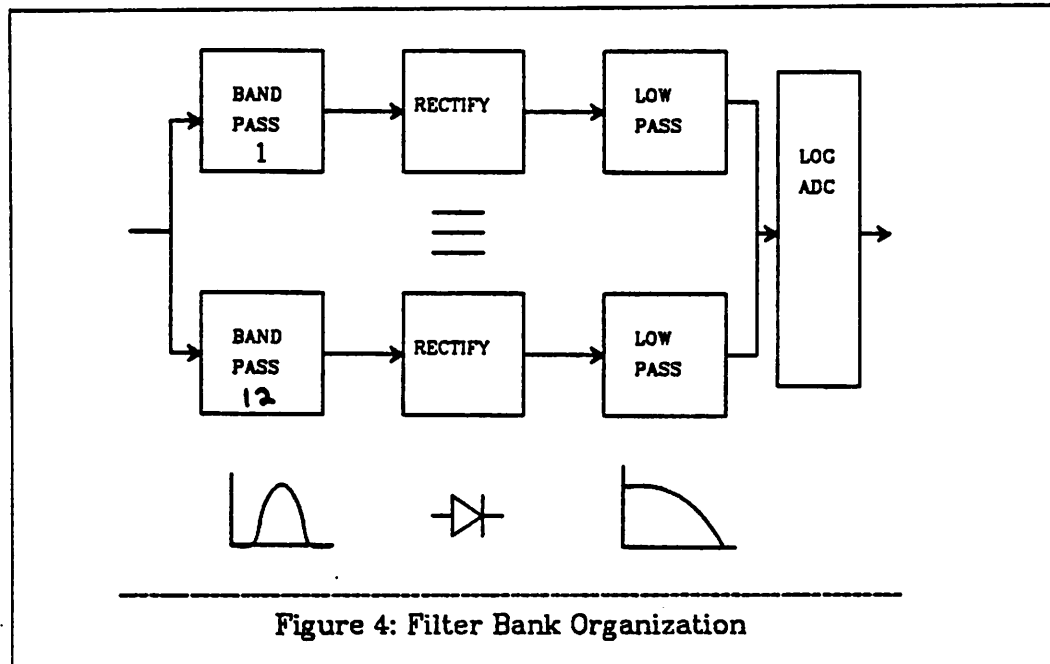
The comparator subsystem compares the frames input to it to the frames of the words in the system dictionary. A score is computed for the similarity between the input word and every template word in the system dictionary. The best score is found, and that value is compared to the rejection threshold. If the error score for the best word is less than that threshold, the semantic action associated with that word is performed. If not, no action is taken.

## **1.5. Front End**

The following sections describe the front-end portion of the speech recognition system. These portions are implemented together on the front-end integrated circuit.

### **1.5.1. Spectral Analysis**

In the ASR system, speech is modeled as a succession of spectra. The assumption of short-time steady state spectra is generally valid since speech is produced by vocal tracts as affected by articulators (lips, tongue, velum, vocal cords), and these articulators move slowly relative to the spectral sampling rate. The ASR system samples the speech spectrum one hundred times per second. A spectrum for the front end is the average voltage a signal has in several frequency bands.



#### 1.5.1.1. Bandpass Filter Spectral Analyzer

As can be seen from the figure, the front-end computes frames by bandpass filtering the input speech, rectifying the bandpass outputs and then lowpass filtering (averaging over a period of time) the rectified signal. It then takes the logarithm of these values.

As we await the completion of our front end IC, the spectral analysis portion of the system has been implemented with commercially available hardware.<sup>27</sup> It is made of third-octave switched-capacitor filters. The 3dB cutoff frequencies of the bands is shown in the following table.

Filter Bank 3 Db cutoff frequencies in Hz.					
filter	lower	upper	filter	lower	upper
1	200	400	7	1250	1600
2	400	500	8	1800	2000
3	500	630	9	2000	2500
4	630	800	10	2500	3150
5	800	1000	11	3150	4000
6	1000	1250	12	4000	5000

The original filter bank consisted of third octave filters from the octave 100-200 hz. to the octave 3200-6400 hz. It became obvious from the start that the low octave was an unreliable cue to use when differentiating words. The primary reason for this was that third octave filters at low frequencies are very narrow, much narrower than the spacing between pitch harmonics. Therefore pitch variations were harming recognition performance. However low frequency information is important for accurate speech recognition. For instance it would be difficult to differentiate between sounds that differ mainly in voicing, such as /b/ and /p/, without low filters. Therefore we substituted a full octave filter for the three filters in the range of 200-400hz.

The spectral analysis uses six pole bandpass filters, a half wave rectifier, and a three pole butterworth low pass averaging filter that cut off at 25 hz. The averaging filters approximate an analysis window that lasts about twenty milliseconds. Thus the short term spectrum of the input speech (averaged over twenty milliseconds) is sampled every ten milliseconds.

The averaging filter is a critical part of the design of the front-end. It restricts the amount of input speech over which the spectral analysis takes place. It assumes that the speech spectrum is stationary during this analysis interval. Unfortunately different parts of speech require different types of averaging filters for this assumption to be valid. Some sounds are stationary for long periods of time (such as long vowels) and others have spectra that

change rapidly (such as the burst in /b/). For example, figure 5 shows the output of lowpass averaging filters whose inputs are a steady state vowel that has been bandpassed (200-400 hz) and rectified. The filters have two poles and have cut-off frequencies of 100hz, 50hz, 30hz and 20hz. The ripple varies from 100% of the signal for the high frequency cut-off to 1.2% of the signal for the 20hz filter. Thus in a steady-state vowel, in order to eliminate ripple caused by the pitch periodicity a long averaging window is required. However an impulsive signal such as the burst of a stop consonant requires a short time window. We have compromised, choosing our twenty millisecond window implemented by the low pass filter.



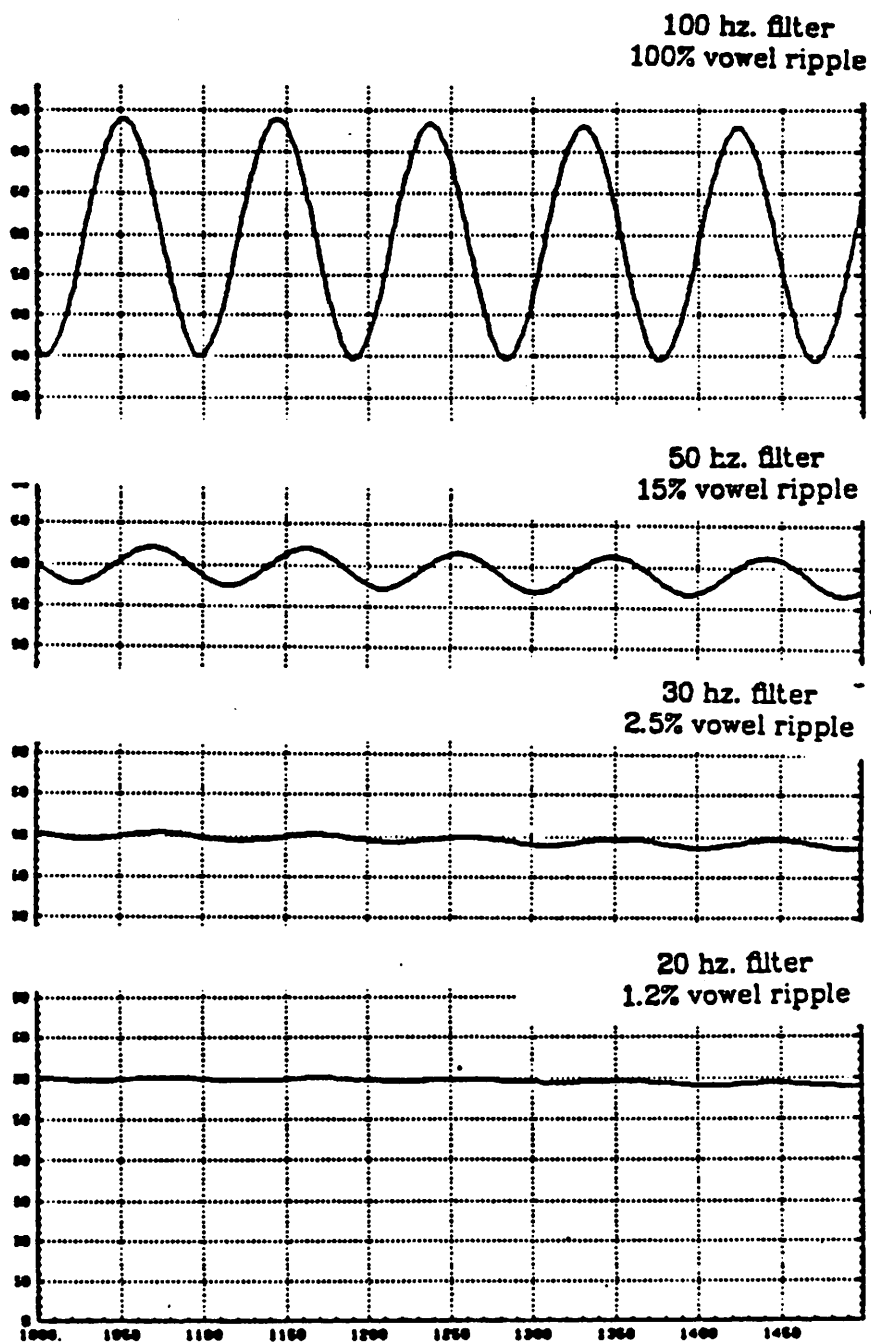
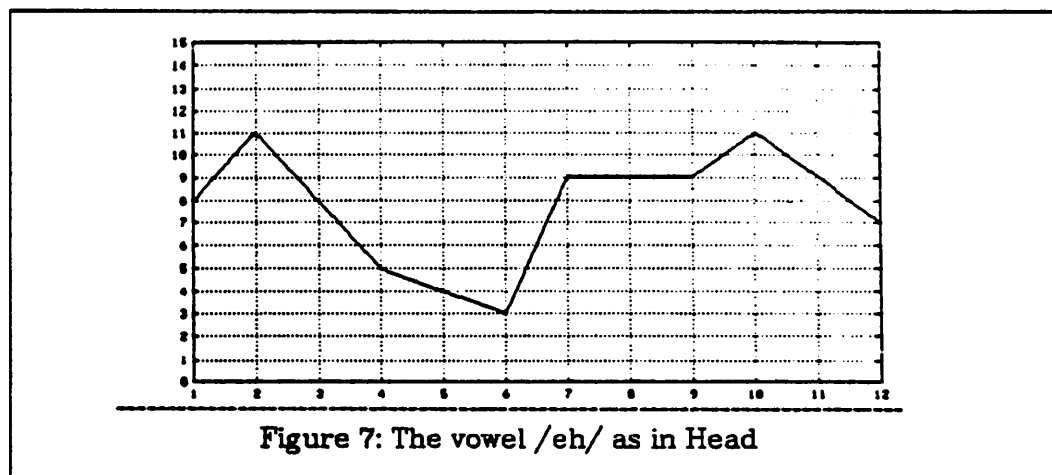
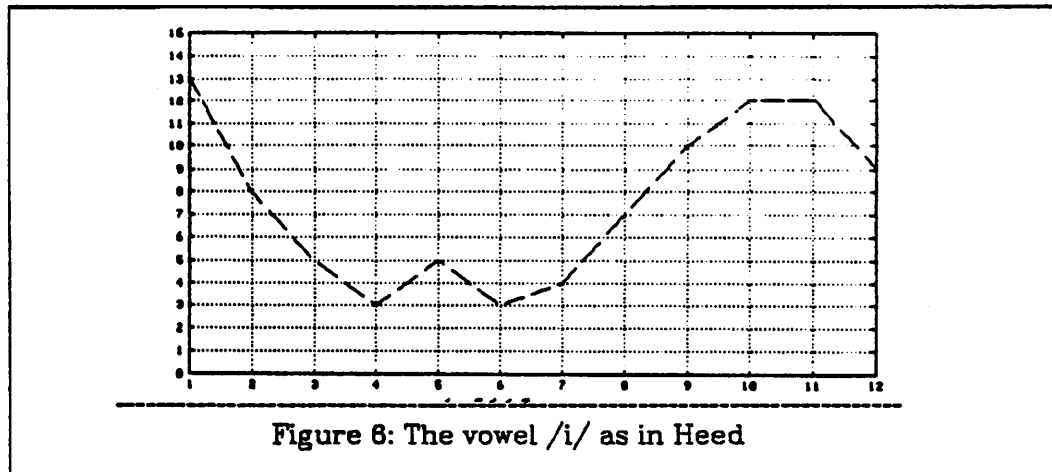
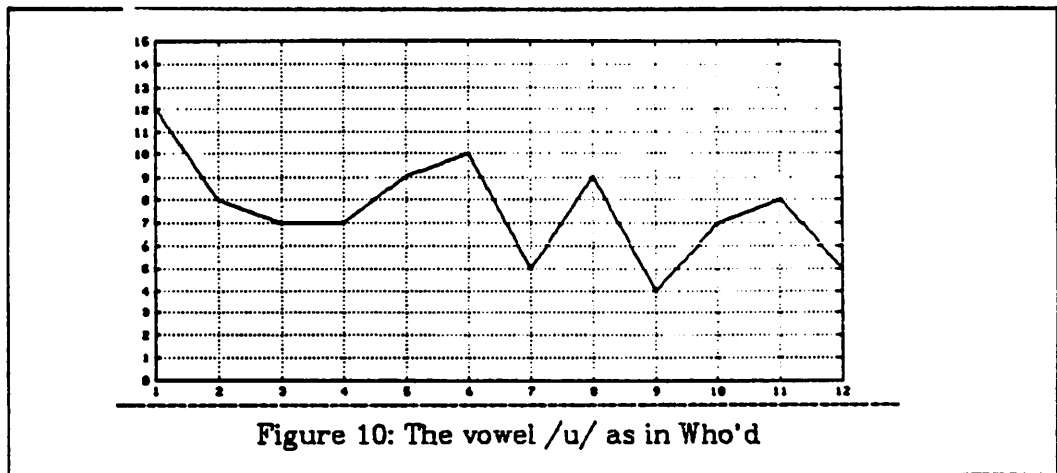
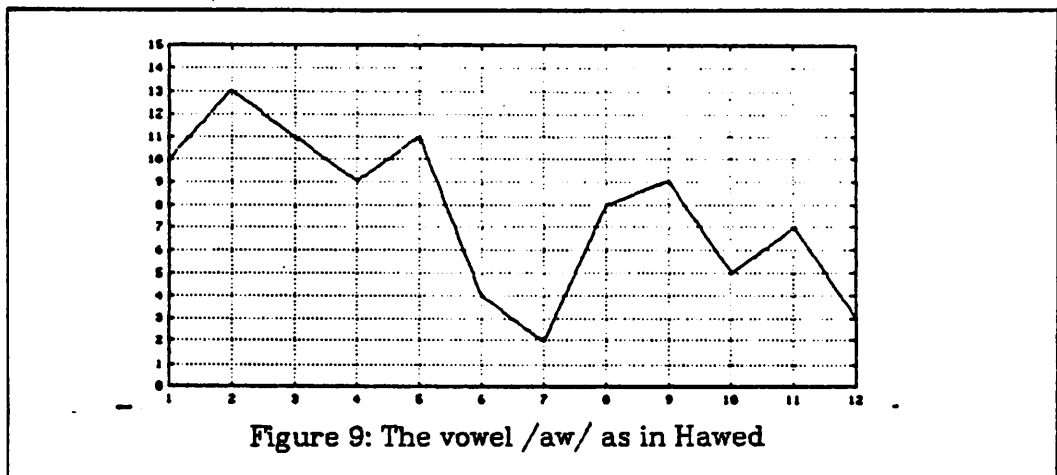
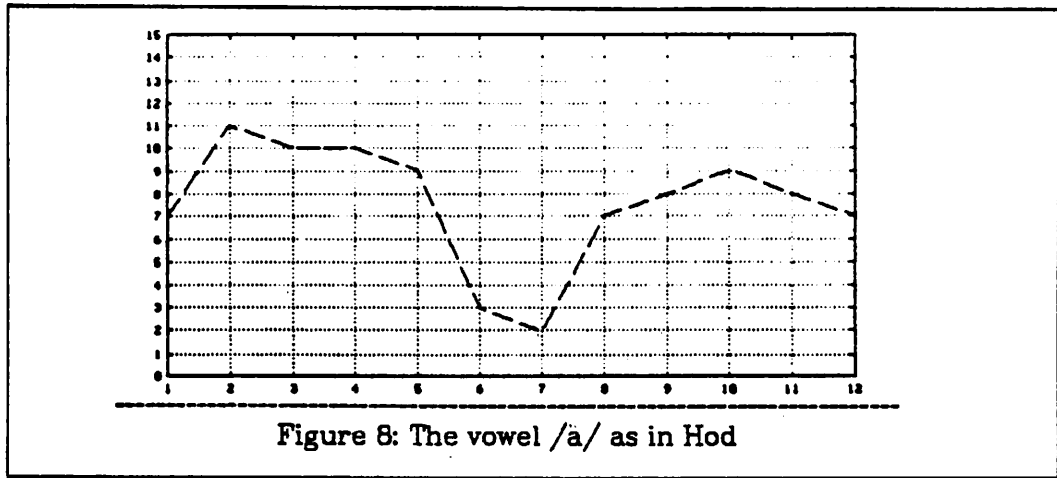
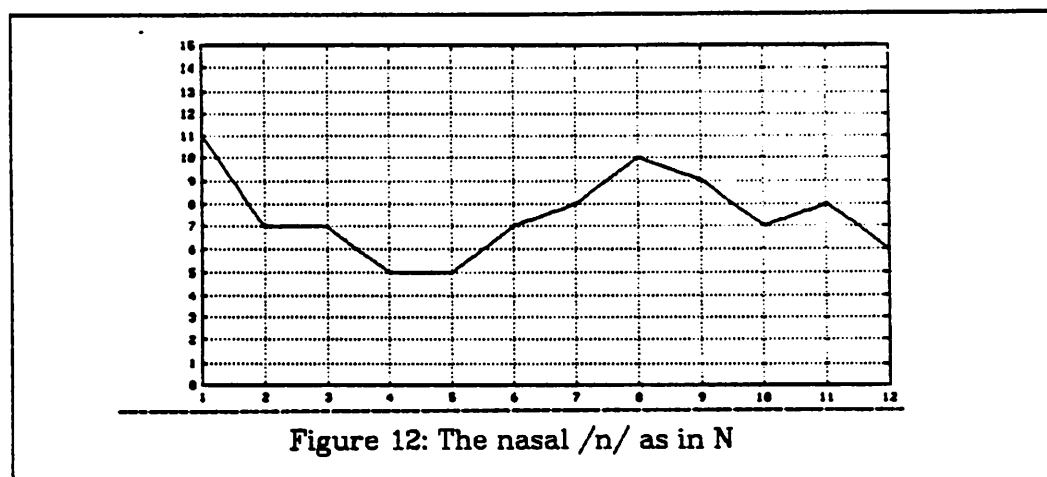
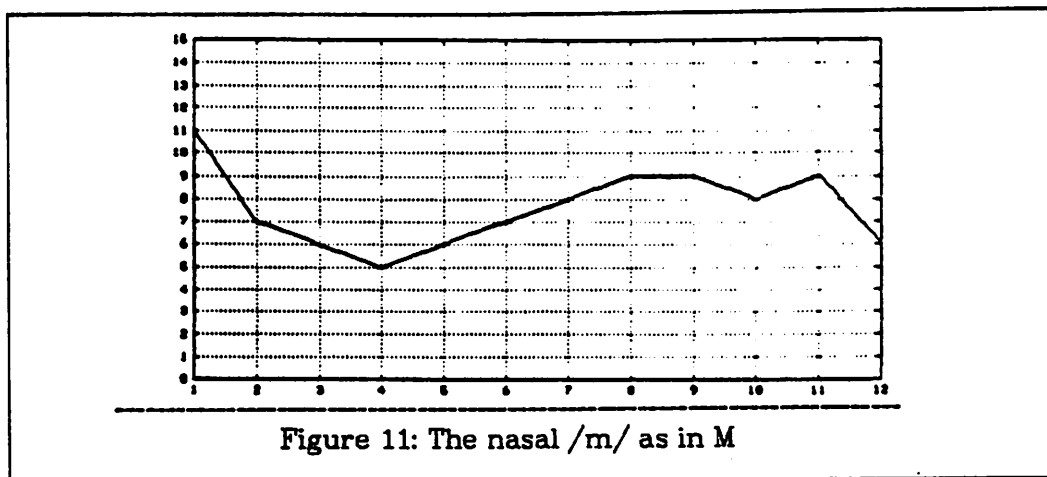


Figure 5: Vowels with an Averaging Filter

The figures below show a series of normalized spectra made by our front end for a male speaker repeating several vowel and nasal sounds (/i/ heed, /eh/ head, /a/ hod, /aw/ hawed, /u/ who'd, /m/ and /n/).







#### 1.5.1.2. Linear Predictive Coding

Linear Predictive Coding<sup>10</sup> (LPC) is another approach to spectral analysis for speech recognition. It is similar to bandpass filtering (BPF) in that speech is represented as a series of frequency-smoothed power spectra. LPC analysis represents speech as a series of all-pole spectra that best fit the input speech spectra. As the spectra have a limited number of poles, typically 8-20, fine detail of the spectra such as pitch harmonics are not represented. Thus the spectral envelopes are represented and the excitation (the fine detail in the spectra) are not. In bandpass filter banks, this spec-

tral smoothing is accomplished by using wide bandwidths filters to find the spectral envelopes.

One major difference between the two approaches is that BPF allows any sort of spectrum to be represented, whereas LPC uses spectra composed of poles alone. LPC has the advantage that it represents spectra by their resonances and thus may use less bits to represent them than bandpass filtering which explicitly represents each frequency band, not just the locations of the resonances.

The major difference between LPC and BPF approaches is the spectral normalization used and the resulting distance measure. As will be stated in the next section, BPF normalizes spectra by sampling the spectra in bands and scaling all the values so that the total spectrum has constant energy. LPC's normalization also requires constant energy, but the all pole spectra used are ones that match the peaks (harmonics) of the spectra more than the other portions. Specifically LPC analysis finds the spectrum which minimize the error

$$E_{lpc} = \frac{G^2}{2\pi} \int_{-\pi}^{\pi} \frac{P(\omega)}{\hat{P}(\omega)} d\omega .$$

where  $P(\omega)$  is a spectrum to be modeled,  $\hat{P}(\omega)$  is the LPC model desired, and  $G$  is a gain factor.<sup>28</sup> This error is used both to derive the LPC spectrum and also to compare two spectra during speech recognition. Note that this error measure weights the areas of the spectrum more where  $P(\omega)$  is greater than  $\hat{P}(\omega)$  than where  $P(\omega) < \hat{P}(\omega)$ . Hence the normalized LPC spectrum closely follows the harmonic envelope of the speech spectrum, whereas the BPF spectrum weights all parts of the speech spectrum equally in generating its smoothed spectrum.

The LPC error above can be interpreted as the energy in a signal obtained by passing the signal corresponding to  $P(\omega)$  through a filter defined by  $\frac{1}{\hat{P}(\omega)}$ . For comparison reasons the BPF distance measure is restated below as

$$\begin{aligned} E_{bpf} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[ \log P(\omega) - \log \hat{P}(\omega) \right]^2 d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[ \log \left( \frac{P(\omega)}{\hat{P}(\omega)} \right) \right]^2 d\omega. \end{aligned}$$

As Makhoul<sup>28</sup> notes, the two distance measures  $E_{lpc}$  and  $E_{bpf}$  are quite similar. The major difference are computational expenses in two areas, spectral analysis in the front-end and spectral distance measure used when comparing spectra.

One of the major reasons why we selected BPF analyzers at the start of this project was that there were integrated circuits<sup>27</sup> available off the shelf to implement them. LPC analyzers required more complex systems. However now there have been examples of several working LPC analyzer integrated circuits.<sup>29,30</sup> With digital implementation of front-end processors such as ours, either technique can be implemented.

For spectral comparisons, the distance measure used with LPC coefficients is more complex than the squared distance metric used with BPF. A common implementation of the LPC spectral distance is the Itakura distance measure<sup>10</sup>

$$Dist = c + \log \left[ (\mathbf{br}) / (\hat{\mathbf{a}}\mathbf{r}) \right].$$

Here  $\mathbf{a}, \hat{\mathbf{a}}, \mathbf{b}$  and  $\mathbf{r}$  are vectors of size  $p$  (typically 8-12) and  $(\mathbf{br})$  means the dot inner product of the two vectors  $\mathbf{b}$  and  $\mathbf{r}$ .  $c = \log(\mathbf{aa})$  and  $\mathbf{b}$  (where  $b_i = 2 \sum_{j=0}^{p-1} \frac{a_j a_{j+i}}{(\mathbf{aa})}$ ) are computed during spectral analysis and stored with the

reference pattern. The coefficients  $\hat{\mathbf{a}}$  are computed for the unknown pattern along with the unknown's autocorrelation function  $\mathbf{r}$  and the dot product  $(\hat{\mathbf{a}}\mathbf{r})$ . Computing  $(\mathbf{b}\mathbf{r})$  for each of the spectra in the system's dictionary is the the major work done to compute spectral distances for LPC. Though this scheme is more complex than the one for BPF, a special purpose design such as the approach we've taken for our comparator integrated circuit could implement this or a BPF approach equally as well.

There have been several studies<sup>9,31,32</sup> done comparing the two techniques for dynamic time warp based speech recognition systems with mixed results. We ran a pilot study comparing the techniques with a slight advantage for the band pass filter approach. It seems that the relative merits of LPC versus BPF are not completely known, however it seems clear that from a practical point of view the difference between the two approaches is small.

### 1.5.2. Logarithmic Conversion and Energy Normalization

After calculating the absolute inband spectral amplitudes, the front-end processor takes the logarithm of the spectrum and then energy normalizes it by subtracting from each log filter amplitude the average log filter amplitude for that frame.

$$\hat{f}_i = f_i - \frac{\sum_{i=1}^p f_i}{p} \quad (1)$$

The processor takes the logarithm of the data spectral amplitudes for several reasons. Comparisons between two log-spectra are more fair than between linear spectra since with log spectra the same percentage difference results in the same error for spectra with different amplitudes. A side advantage is that a logarithmic representation allows the spectral normalization to be performed with simple adds and subtracts instead of

multiplies and divides.

In the spectral analysis hardware described above, an off the shelf  $\mu 255$  Law analog to digital converter is used to sample the outputs of the low pass filters and to convert each low pass output to  $\mu 255$  Law every ten milliseconds. The microprocessor converts this eight bit  $\mu 255$  Law representation to a four bit true log representation in the following manner. The  $\mu 255$  Law samples (8 bits) are represented by one sign bit three bits indicating a chord ( $C$ ) and four other bits indicating an offset ( $O$ ). Rare negative numbers are represented as zeros. Positive numbers are represented as  $2^C[2^4+O]-2^4$ . The most significant three bits used to convert this to a four bit logarithmic format are those representing  $C$ . The least significant bit is derived from  $O$  by table lookup. If  $O$  is greater than 5 the least significant logarithmic bit is one, otherwise it is zero.

Normalization eliminates the overall spectral amplitude from consideration when comparing two frames. Thus the speech recognition system does not penalize a word when it is trained loudly and then spoken softly (or trained spoken closer to the microphone and then spoken further away from it). In order to do so two spectra are always compared with the same amplitude levels. Given the distance measure to be used (a squared euclidean metric) the normalization equation can be derived by the following argument.

No error due to a difference in overall spectral gain should be added to the spectral distance measure. Since the distance measure used is  $d(F_a, F_b) = \sum_{i=1}^p (f_i^a - f_i^b)^2$  the normalized distance of frame  $F_a$  from  $F_b$  should be



$$\hat{d}(F_a, F_b) = \text{MIN}_k \sum_{i=1}^p [f_i^a - f_i^b + k]^2 \quad (2)$$

$$= \text{MIN}_k \sum_{i=1}^p [k^2 + (f_i^a - f_i^b)^2 + 2k(f_i^a - f_i^b)]. \quad (3)$$

where  $f_i^a$  and  $f_i^b$  are the log-amplitude filter coefficients of the vectors  $F_a$  and  $F_b$ ,  $p$  is the number of filter coefficients and  $k$  is a variable gain factor for one of the frames that acts as a variable gain factor to eliminate differences in amplitude. Setting the derivative of (3) with respect to  $k$  equal to zero,

$$k = p^{-1} \sum_{i=1}^p f_i^a - p^{-1} \sum_{i=1}^p f_i^b. \quad (4)$$

and substituting  $k$  into (2),

$$\begin{aligned} \hat{d}(F_a, F_b) &= \sum_{i=1}^p \left[ \left( f_i^a - p^{-1} \sum_{i=1}^p f_i^a \right) - \left( f_i^b - p^{-1} \sum_{i=1}^p f_i^b \right) \right]^2 \\ &= \sum_{i=1}^p [\hat{f}_i^a - \hat{f}_i^b]^2, \end{aligned} \quad (5)$$

and the normalization discussed above is derived.

### 1.5.3. Endpoint Detection

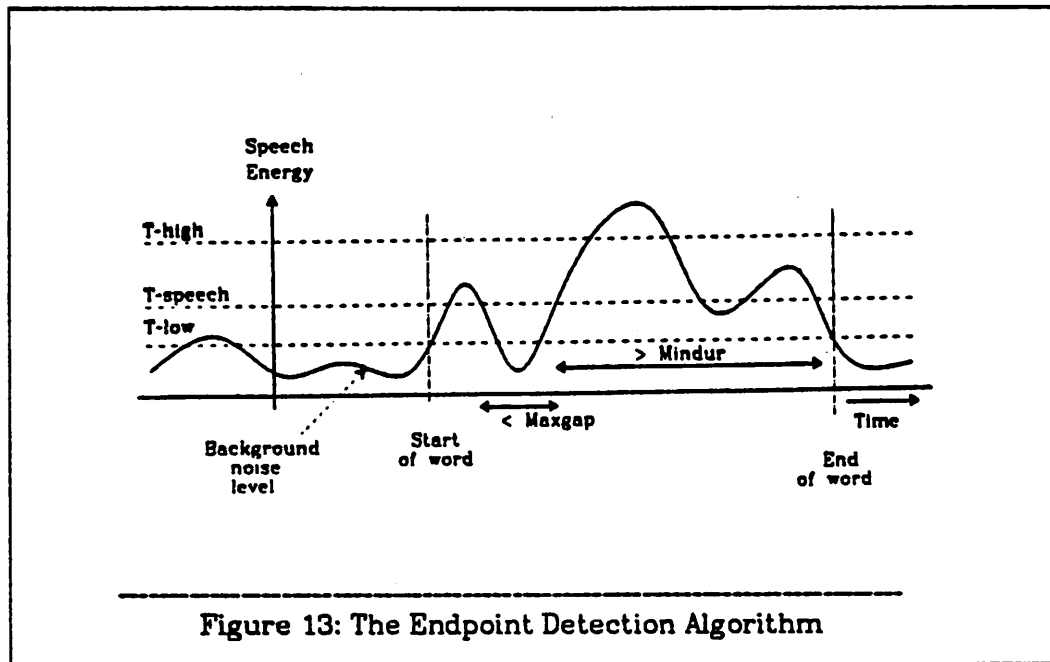
The speech recognition system represents words as a series of frames that are generated every 10 milliseconds by the front end. It is the job of the endpoint analyzer to determine which of these frames are part of a word or phrase being spoken and which are part of the background noise.

The endpoint analyzer is roughly based on one that was proposed by Rabiner and Sambur.<sup>35</sup> It was redesigned and implemented and has since been modified by Davies.<sup>25</sup> The endpoint decision function used is based on the level of the input signal and knowledge of typical durations of various speech sounds. The endpoint analyzer, in order to function, needs to know the current background noise level relative to the current speech level. This

level is called  $E_{bg}$ . With the background noise level known, the endpoint algorithm calculates three thresholds levels  $T_{low}$ ,  $T_{speech}$ , and  $T_{high}$ .  $T_{speech}$  is a level at which the algorithm is confident that "something is happening" to the input signal. It is 10 dB above the background noise level.  $T_{low}$ , 3 dB. above the background level, is a lower level which could indicate that some speech is in progress, or the level may simply be caused by background noise.  $T_{high}$  is a high level. The algorithm requires every word to reach the level  $T_{high}$  somewhere inside that word to eliminate moderate level room noises and to keep proper signal levels in the converter and analysis circuits.

The endpoint algorithm starts by searching for words to begin. It does so by examining the input speech level read from the filter bank. The level is the average of the unnormalized filter coefficients, the same term  $p^{-1} \sum_{i=1}^R f_i$  that is computed by the normalization routine. Should the input level reach  $T_{speech}$  and stay at least that high for MINDUR consecutive frames (the minimum duration of a word, 80 milliseconds), the endpoint analyzer announces that a word or phrase has begun. It considers the speech to have begun at the last crossing of the  $T_{low}$  level before the recent crossing of the  $T_{speech}$  level. However, in order to accommodate burst-like speech sounds at the start (and at the end) of words a change is made to this procedure. Define a burst of speech as a sequence of frames where the speech level crossed  $T_{speech}$  and then fell below it before MINDUR frames had passed. Then, if the time between when the input speech level of a burst falls below  $T_{speech}$  and when the level of a following word crosses  $T_{speech}$  is less than MAXGAP frames that burst is included in the word, from the time that burst originally crossed  $T_{low}$ . MAXGAP is 180 milliseconds which including the smearing of the burst from the low pass filter in the front end. In order to exclude

speech artifacts such as clicks or lip smacks from words, the burst must have a duration of PULSEWIDTH frames (40 ms.). An example of a word with a burst included at the start is shown in the figure below.



Once a word has begun the endpoint analyzer searches for the end of that word. This occurs when the input level falls and stays below  $T_{speech}$  for at least MAXGAP frames. MAXGAP is the maximum length of an inword silence. To find the end of the word the algorithm searches these MAXGAP frames for the last frame in that sequence that fell below  $T_{low}$  when the previous PULSEWIDTH frames had been above  $T_{low}$ . An example of an allowable inword silence (where the input level falls below  $T_{speech}$  for less than MAXGAP consecutive frames) is the gap between the /d/ and /p/ of the word *endpoint*.

If the speech has ended but the level has never crossed  $T_{high}$  the endpoint analyzer aborts the recognition process on that word, otherwise it declares the input a legitimate word or phrase.

MAXGAP affects the response time of the speech recognition system as no decision on recognition can be made until at least MAXGAP frames after the word has ended. Only at that time does the system know that the word is over.

To find the background noise level, the endpoint algorithm constantly monitors each input frame checking for changes to its guess of  $E_{bg}$ . It does this in the following manner. It checks if the current input frame is less than  $E_{bg}$ . If it is, the algorithm assumes that  $E_{bg}$  is too high and decrements it by a small amount  $\varepsilon$  and then recomputes the various thresholds. If the input frame is between  $E_{bg}$  and  $T_{speech}$  it assumes that  $E_{bg}$  is too low and increments it by  $2\varepsilon$ . If the input is between  $T_{speech}$  and  $T_{high}$  then  $E_{bg}$  is incremented by  $\varepsilon$ . If the input frame level is above the high threshold it is assumed that the frame is speech and not noise and  $E_{bg}$  is not changed.

#### 1.5.4. Selective Downsampling

Each normalized filter value that is part of an input word is stored in template memory (during training mode), the system's dictionary, or sent on to the word comparator (during recognition mode). However frames which are very similar to previously transmitted frames are not transmitted nor is a repeat bit sent. In this way the ASR system dynamically varies the spectral sampling rate. This both reduces the data rate through the system and reduces the amount of emphasis the ASR system would otherwise place on long steady state sounds in its word comparison algorithms. In so doing it improves the accuracy of the speech recognition system. This reduction of the frame rate also increases the size of the vocabulary that the system can process in real time. It also increases the number of words that can be stored in template memory. The technique of selectively downsampling has

been recently discussed by others.<sup>34, 35, 36</sup>

Selective downsampling is accomplished in the following way. The first frame of a word is always transmitted to the comparator. Succeeding frames are compared to the last frame transmitted. A new frame is only transmitted if the spectral distance between it and the previously transmitted frame is greater than a predefined threshold. With the threshold chosen (a distance of seven units -- see the tables in the next chapter), the effective spectral sampling rate after downsampling is between twenty and twenty-five milliseconds when averaged over long periods of speech. In the short term the spectral sampling rate can be very low, such as during a long steady state vowel when only one frame in ten is transmitted, or it can be as fast as the original sampling rate, as during transitional speech.

### **1.6. Word Comparator**

Ultimately a speech recognition system must compare features from its input word with features in its dictionary. These comparisons have traditionally been the most time consuming part of similar speech recognition systems.

The basic unit of comparison in the speech recognition system is the frame. As each word is composed of a series of frames, words are compared by summing the spectral distances between corresponding frames of the words. Two issues then are: how frame errors are computed and how corresponding frames are discovered.

#### **1.6.1. The Distance Measure**

The recognizer compares frames by taking the squared euclidean distance between them. That is the spectral error is defined as the sum of the

squared difference across all bands of the two spectra. Let two frames be  $F_a = (f_1^a, f_2^a, \dots, f_p^a)$  and  $F_b = (f_1^b, f_2^b, \dots, f_p^b)$ . Then

$$d(F_a, F_b) = \sum_{i=1}^p (f_i^a - f_i^b)^2 \quad (6)$$

Absolute value distances have also been used with band pass filter front ends for speech recognition, primarily as a way of avoiding the squaring operations in (6).<sup>9</sup> Such a distance would be

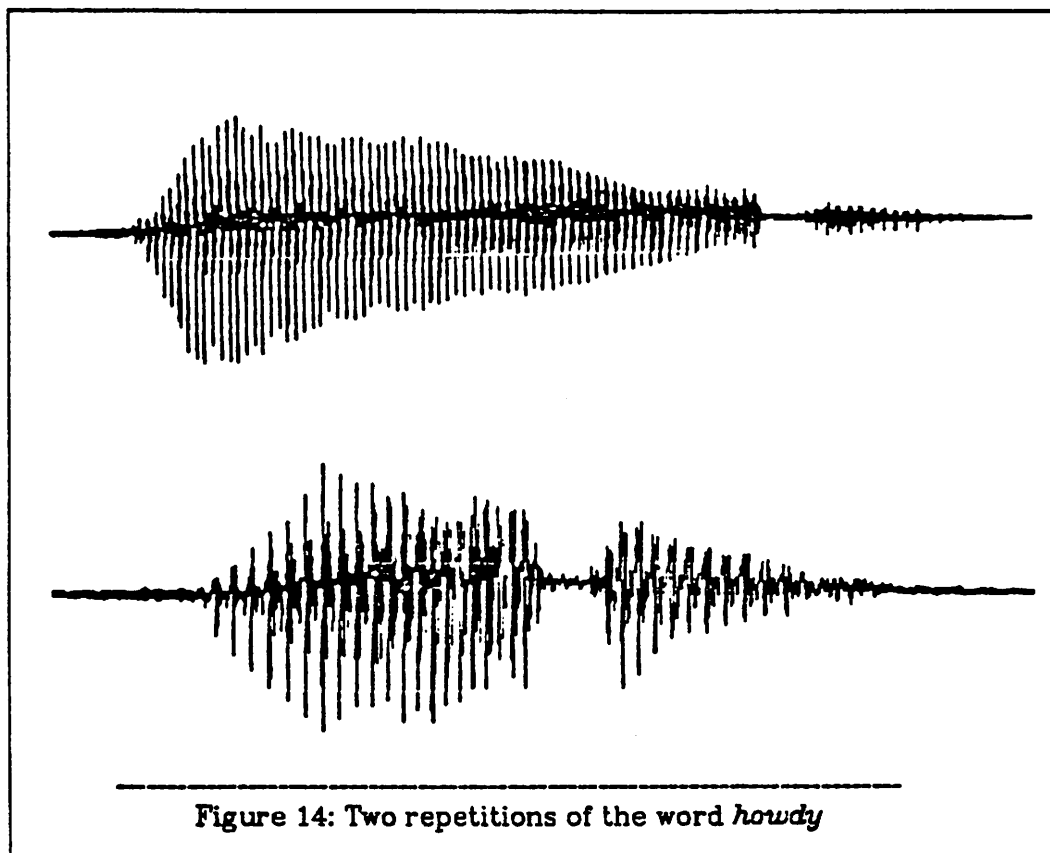
$$d(F_a, F_b) = \sum_{i=1}^p |f_i^a - f_i^b|. \quad (6a)$$

However when building special purpose hardware for a speech recognition system the squaring operation is not difficult to implement. The function  $(f_i^a - f_i^b)^2$  can be built as a four bit subtractor followed by a five bit by eight bit Read Only Memory (ROM), 256 bits of ROM, or a smaller Programmable Logic Array (PLA). As the subtractor would be needed to compute either equation (6) or (6a) the extra PLA is not a large price to pay for the squaring operation. Studies done earlier indicate that the absolute value distance metric degrades recognition performance.<sup>9</sup>

### 1.6.2. Dynamic Time Warping

Finding which frame of one word corresponds to which frame of the other is a complex issue. This is because repetitions of a given word may have different durations, and even with similar durations parts of one word may last longer than their corresponding parts in other repetitions. Thus when comparing two words that have different durations but are otherwise similar, if one of the words is uniformly expanded or contracted to be the length of the other, the  $j^{\text{th}}$  frame of the expanded or contracted word may correspond to a different speech event than the  $j^{\text{th}}$  frame of the other word. Figure 14 shows this using as an example the time waveform of two

repetitions of the word *howdy*. Note that the first repetition of *howdy* has a much longer first vowel, and though it has been uniformly compressed to about the same time scale as the second repetition, its first vowel now is a much larger portion of the word than in the second repetition.



The problem of nonlinearly warping words can be solved using the following comparison process. Assume that the corresponding frames of two words are known. Say that frames  $P_{i_n}(t)$  and  $P_{i_p}(t)$  of words  $W_{i_n}$  and  $W_{i_p}$  correspond for  $1 \leq t \leq L_p$ , where the two words have lengths  $L_{i_n}$  and  $L_{i_p}$ .  $L_p$  is the length of the comparison path between the two words. Then the distance between words  $W_{i_n}$  and  $W_{i_p}$  is the average of the spectral errors along the path of corresponding frames between the two words.

$$\text{dist}(W_{in}, W_{tp}) = \frac{1}{L_p} \sum_{t=1}^{L_p} d(P_{in}(t), P_{tp}(t)) \quad (7)$$

This technique is used in the speech recognition system with a minor modification to  $L_p$  as discussed in section 2.4.2.

The technique of finding these corresponding frames is known as dynamic time warping. To determine this correspondence or equivalently the functions  $P_{in}$  and  $P_{tp}$  note that the set of corresponding frames of two words should be the ones that minimize the distance defined above between those words. Then the distance between two words can be represented as  $\underset{P_{in}, P_{tp}}{\text{MIN}} \text{dist}(W_{in}, W_{tp})$  for all  $P_{in}$  and  $P_{tp}$ . The job of the word comparator is to solve this minimization.

The above minimum can be found without trying all possible  $P_{in}$ 's and  $P_{tp}$ 's if the following reasonable restrictions are placed on the  $P$  functions. First make the functions monotonic

$$P(t) \leq P(t+1), 1 \leq t \leq L_p - 1. \quad (8)$$

Second require that the functions be continuous

$$P(1)=1, P_{in}(L_p)=L_{in}, P_{tp}(L_p)=L_{tp}, \quad (9)$$

$$P(t+1) - P(t) \leq 1, 1 \leq t \leq L_p - 1.$$

These restrictions say that the order of the frames in a word may not be reversed in a comparison and that all parts of both words must be used in the comparison. Given the above assumptions, the above minimum can be computed by the following dynamic programming based algorithm.<sup>9,10</sup>

Let  $D_{i,j}^{in, tp}$  (abbreviated  $D_{i,j}$ ) be the minimum cost distance along some best path that compares the frames of  $W_{in}$  from frame 1 to  $i$  with the frames of  $W_{tp}$  from frame 1 to  $j$ . Clearly  $D_{1,1} = d(W_{in}(1), W_{tp}(1))$ . Since by equations (8) and (9) every path to  $i, j$  must pass through one of either  $i-1, j$ , or  $i, j-1$



or  $i-1, j-1$  then

$$D_{i,j} = d(W_{in}(i), W_{tp}(j)) + \text{MIN}(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}) \quad (10)$$

Here  $d(W_{in}(i), W_{tp}(j))$  is the cost of  $i, j$  and  $\text{MIN}(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1})$  is the cost of the best path to approach  $i, j$ .

To completely formulate this problem let  $D_{i,0} = D_{0,j} = \infty, i, j > 0$ . Then  $D_{L_{in}, L_{tp}}$ , the distance between  $W_{in}$  and  $W_{tp}$ , can be computed.

The recursion to solve for  $D_{i,j}$  is evaluated in the following order. First  $D_{1,j}$  is computed for  $j=1, L_{tp}$ . These computations are known as the first column of the algorithm. Then  $D_{2,j}$  is evaluated; the second column. This continues until the final column  $D_{L_{in}, j}$  is computed where the top of that column  $D_{L_{in}, L_{tp}}$  is the desired result. At this point derive a similarity measure between words which is independent of the length of the words being compared by dividing the sum by the path length. The final normalized error between an input word and a template is the average frame error computed when comparing those two words.

$$\text{dist}(W_{in}, W_{tp}) = \frac{D_{L_{in}, L_{tp}}}{L_p} \quad (11)$$

Though the above technique has the advantage of automatically time aligning the words it compares (an important consideration in speech recognition) it has the disadvantage of requiring computation of all the  $L_{in} \cdot L_{tp}$  spectral distances  $d(W_{in}(i), W_{tp}(j))$ ,  $i \leq L_{in}, j \leq L_{tp}$  even though only  $L_p$  distance measures actually are involved in the final distance computation. Consider the following example. If there are  $N$  words in the dictionary with an average duration of  $T_{tp}$  seconds and the frame period was  $T_f$ , then there are  $L_{tp} = \frac{T_{tp}}{T_f}$  frames per word.  $N \cdot L_{tp} \cdot L_{in}$  distance measures must be calculated

in comparing the input word to each dictionary word. For a dictionary size of 1000 words and an average word length of half a second ( $T_{tp} = .5sec$ ) and a 10 ms. frame period ( $T_f = 10ms.$ ), the system would need to compute  $1000 \cdot 50 \cdot 50 = 2,500,000$  distance measures to recognize one word!

The important points to remember about this algorithm are:

- **Frame by frame computation:** When comparing the  $i^{th}$  frame of an input word with a set of template words only that frame is used to compute the spectral distances between that frame and every frame in each of the template words. The speech recognition system no longer needs to keep track of previous input frames.
- **Template word independence:** As the distance from the input word to a template word is independent of the input's distance to other template words those computations may be done at the same time.
- **High computation rate:** Even though only  $L_p$  spectral distances eventually figure in a word error score, all  $L_{tp} \cdot L_{tn} \gg L_p$  distance measures must be calculated resulting in a high computation rate.

Systems which implement this dynamic programming recursion with a computer or microprocessor often reduce the number of comparisons needed using various techniques that restrict the time-warp path.<sup>51</sup> After pruning the paths they wind up having to do about  $k \cdot L_p \ll L_{tp} \cdot L_{tn}$  spectral distance measures where  $k = 5$  or  $10$ . Another time saver is to reject poorly scoring templates before they are completely compared to the input. Though the above authors claim that these systems do not suffer significant performance degradation due to these computation optimizations, the resulting computation rate is still very large for a 1000 template system. Further connected speech algorithms discussed in chapter 3 cannot take

advantage of a path-restricting approach unless they wait until the input phrase has ended before beginning to execute the dynamic time warp algorithm.

Other dynamic time warped systems may have different variations of the algorithms discussed above. For instance some have *slope constraint*.<sup>37</sup> A version of slope constraint would add the following equation to the list of constraints shown before.

$$\Delta(t) = P_{in}(t) - P_{tp}(t) \quad (12)$$

$$|\Delta(t+2) - \Delta(t)| < 2$$

We have found that this degrades recognition accuracy. See the section on testing slope constraints in chapter 2.

### 1.7. Recognition or Rejection Criteria

After the system computes the error scores for matching the input and every one of the system's templates, it finds the template with the minimum error. If that error is less than a threshold value REJ\_THRESH then the action associated with that word is executed (i.e., that word is recognized).

REJ\_THRESH affects the tradeoff in number of rejections versus number of false alarms and errors. Either extreme can be very bothersome and can be adjusted by the user if necessary.

### 1.8. The Training Algorithm

In order to recognize speech, the ASR system needs at least one template word ( or typical pronunciations of a word ) for each word in the system's vocabulary. These templates differ for each speaker in a speaker-dependent system. Gathering the templates is a very important part of the speech recognition system. The quality of the templates greatly influences

the ensuing speech recognition performance.

There are several ways to train a speech recognition system. All of the techniques below have been used at some point in the design of our system. In all of the approaches care should be taken to insure that the speech used to train the system is representative of the type of speech to be recognized by the system.

- **Quick and Dirty:** Have the user repeat each word in the vocabulary one time. That is the template word.
- **Repeatable Pronunciation:** Have the user repeat each vocabulary word several times until  $N$  (2 to 10) words are found that are very similar to each other. The distance computed by the speech recognition algorithm is used to judge similarity. Words are considered similar if their pair distance is less than a threshold somewhat below the rejection threshold. When such a group of similar words are found, the one whose distance to all the rest is minimum is chosen to be the template.
- **Cluster Centers:** Several repetitions of the vocabulary word is spoken and as above groups of similar words (or clusters) of size  $\geq N$  are desired. If more than one such cluster is found, more than one vocabulary word is used for that vocabulary entry. The words that are closest to the centers of the clusters are used for the templates in the system's dictionary. The main difference between this method and the one above is that this technique assumes that the number of words input at first is large and that  $N$  is larger than it was in the previous method. This allows the algorithm to find multiple templates that are different ways that the vocabulary word is pronounced by the speaker.

- **Clustering with Averaging:** This is the above method except that instead of picking the training words closest to the centers of the clusters to be the templates, new templates are constructed that are the average of the words in the cluster and hence are the real center of the cluster.

All of the above methods except the last are very straight forward. The first two are much easier on the user, in that he need not repeat each word in the vocabulary many times. The third technique is a simpler form of the fourth one. The technique of choice is the fourth as tests of the recognition system in the next chapter will show.

Averaging templates is tricky because the templates may be of different durations and the distance metric used is the dynamic time warping, a non-linear function. The following iterative technique works well.

- **Pick Initial Template:** Choose the word that is closest to the center of the cluster of words. This is the template that would have been chosen by the third algorithm above. Refer to this word as  $W_c$ .
- **Compute the Distance (saving the path):** The time-warped distance is computed between each word in the cluster  $W_i$  and  $W_c$ ,  $1 \leq i \leq N_c$  where  $N_c$  is the number of words in the cluster. However, in contrast to the normal recognition process, it is now important to store the paths  $P_i(t)$  as well as the interword distances.
- **Average the Corresponding Frames:** Create a new word  $W_{ip}$  whose features in its  $j^{th}$  frame are the average of the features in each word of the cluster that correspond by time warping to the  $j^{th}$  frame of the template. Another way to explain this follows. Let  $J$  be the set of all frames from words  $W_i$ ,  $1 \leq i \leq N_c$  such that  $P_i(t)=j$ . That is  $J$  is the set of all frames in any word  $W_i$ , such that when  $W_i$  is warped against  $W_c$  all

frames in  $\mathcal{W}_i$  that warp to the  $j^{\text{th}}$  frame of  $\mathcal{W}_c$  are in  $\mathbf{J}$ . Then the features of the  $j^{\text{th}}$  frame of the template created  $\mathcal{W}_{tp}$  are the averages of the features in  $\mathbf{J}$ .

$$\mathbf{J} = \left\{ \mathcal{W}_i(t) : P_i(t) = j \right\}$$

$$\mathcal{W}_{tp}(j)(i) = \frac{1}{|\mathbf{J}|} \sum \mathcal{J}_i \quad (13)$$

$$1 \leq j \leq L_{\mathcal{W}_c}$$

$1 \leq i \leq$  the number of features in a frame

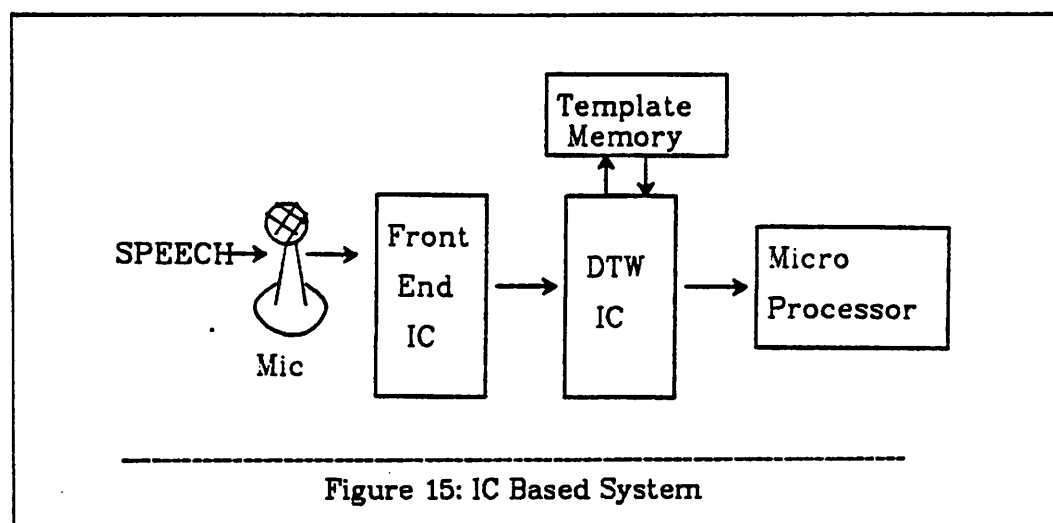
- **Check for Convergence:** The new template  $\mathcal{W}_{tp}$  is composed of all the average filter terms. Now compare  $\mathcal{W}_c$  and  $\mathcal{W}_{tp}$ . If the distance between them is small or if too many iterations of this algorithm have already been used then the algorithm is completed. Otherwise set  $\mathcal{W}_c = \mathcal{W}_{tp}$  and go back to the second step.

In practice this algorithm will iterate two to ten times even when requiring an exact match between  $\mathcal{W}_c$  and  $\mathcal{W}_{tp}$ .

## 1.9. IC Based System

### 1.9.1. A description of the Integrated Circuits

The speech recognition described above is being implemented with two special purpose integrated circuits. See Lowy,<sup>19</sup> Davies<sup>25</sup> and Ruetz<sup>38</sup> for a detailed description of those IC's.



The front-end IC computes short-time spectra of speech (similar to the filter-bank hardware currently in use) and also performs the functions of taking logarithms of, normalizing and downsampling the data and computes the endpoint algorithm. It both sends data directly to the comparator IC (in speech recognition mode) and also to the system microprocessor (in a training mode). It also transmits control information to the rest of the system (such as `START_OF_WORD` and `END_OF_WORD`). It has an on-chip pre-amplifier, an anti-alias filter and an analog to digital converter so that a microphone may be directly connected to it.

The comparator IC reads its input from the front-end IC in recognition mode and sends the results of its comparisons on to the system microprocessor. The microprocessor is not burdened during the recognition task. During speech recognition its only jobs are minimizing the scores to find the best match, checking a rejection threshold and performing the action desired when a particular word is recognized.

The microprocessor will usually perform some application system task as well as its recognition duties (such as being a terminal controller). The

microprocessor also must aid in training. It must compute the templates and must be able to store them and reload the ASR system dictionary when a user who has trained the system in the past starts using the system. This is not the case for applications where a speaker-independent vocabulary is used, pretrained and stored in a non-volatile ASR dictionary memory. In a connected speech or syntactic input mode there are more burdens placed on the microprocessor. This will be discussed chapter 3.

Currently the front end IC is being tested . Some of its subsections have already been tested and found to operate properly.<sup>3A,25</sup> Much of the comparator IC has already been fabricated and is currently working in the system. <sup>1A,25</sup>

### 1.9.2. Speech-Recognition System Timing

The speech recognition system runs in real time. That is, the system processes frames in the amount of actual time represented by a typical frame. This is roughly 20 milliseconds with selective downsampling. Thus selective downsampling impacts the performance of the speech recognizer greatly. Consider the example used above concerning a thousand word vocabulary. Each  $T_f$  seconds (the average frame rate) the system must compute  $D_{i,j}$  for  $j=1, L_{tp_n}$  for each of the  $N$  words in the dictionary. So with the assumptions used in the earlier section it would have to compute

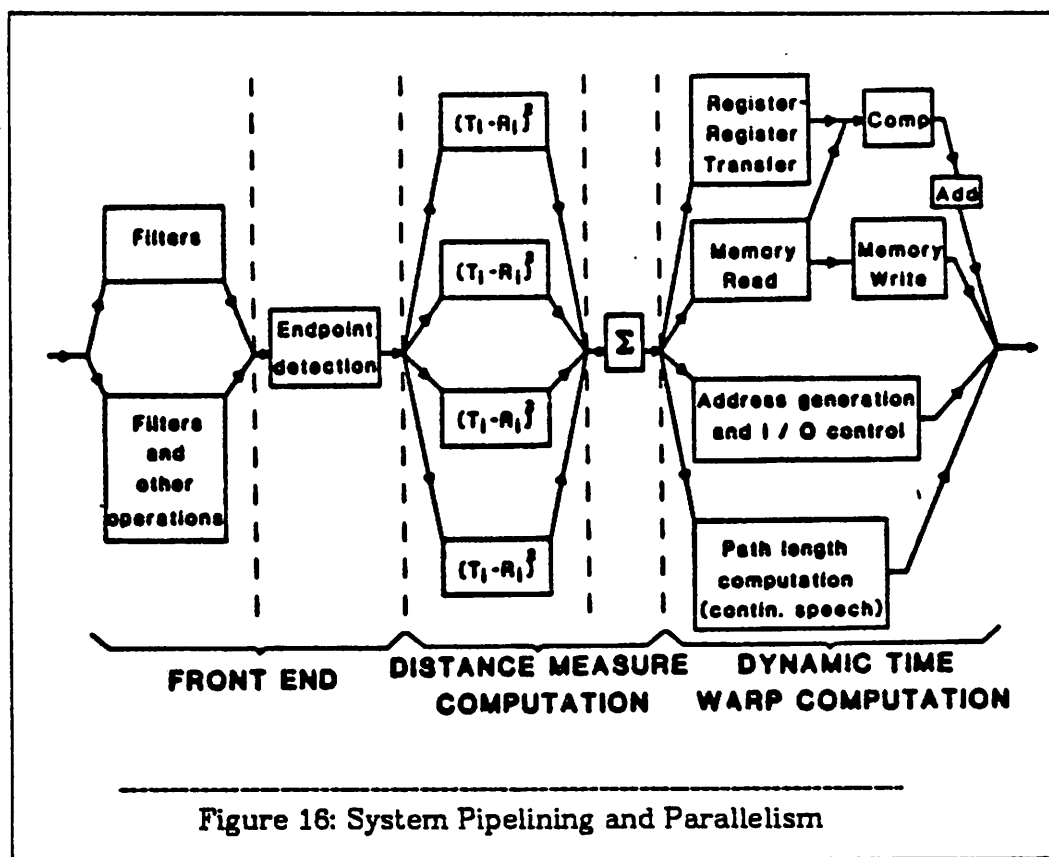
$\frac{1}{T_f} \sum_{n=1}^N T_{tp_n} \approx \frac{N \cdot T_{tp}}{T_f}$  squared euclidean distance measures every  $T_f$  milliseconds. With  $N=1000$  and  $T_{tp}=.5$  seconds and  $T_f=10$  ms. (no downsampling) it has  $\frac{1000 \cdot .5sec.}{10ms. \cdot 10ms.} = 200$  nanoseconds to compute each squared euclidean distance measure.

In addition to these distance measures, the system also must compute the dynamic programming recursion, perform



the endpoint analysis, derive the spectral data, normalize the final scores and so on.

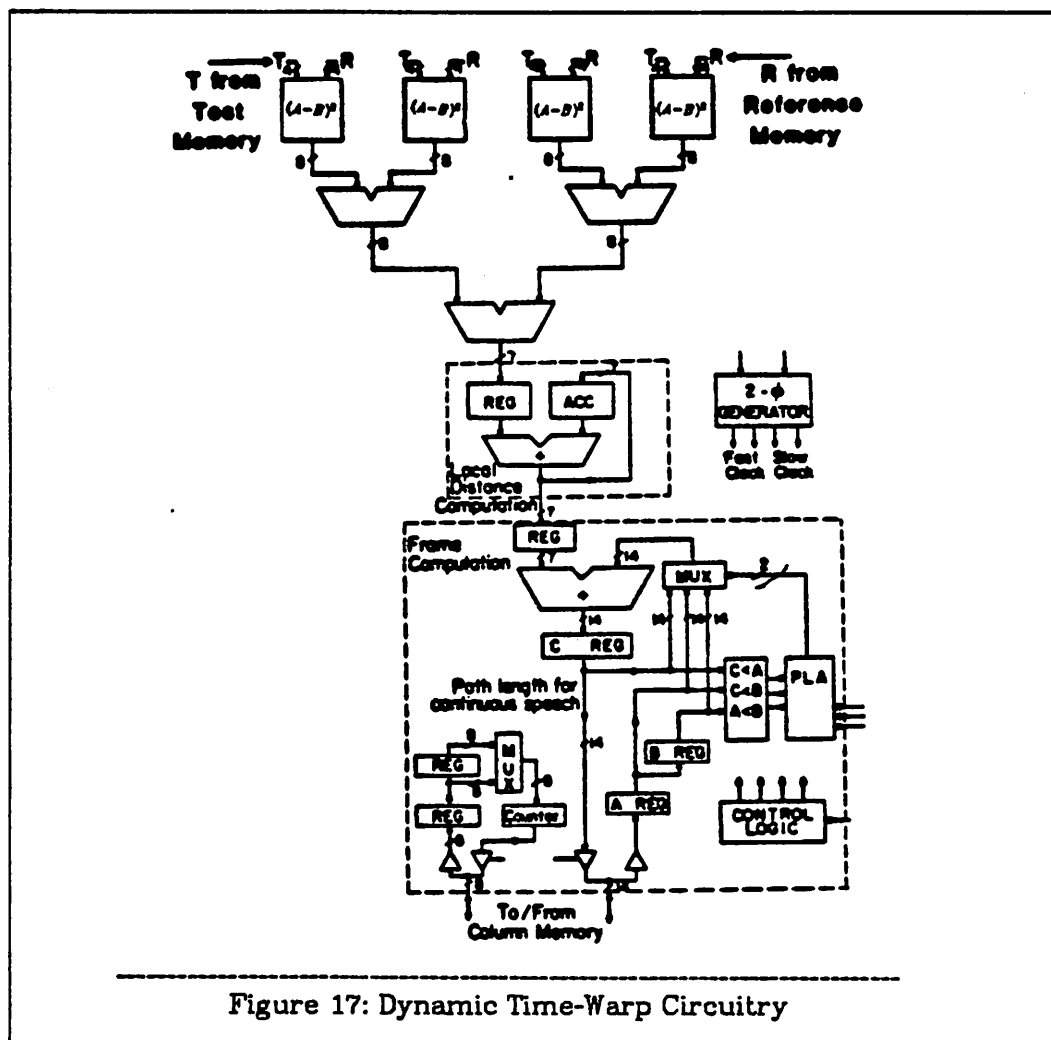
Lowering the effective frame rate reduces the high data rates above. If the effective frame period  $T_f$  falls to 20 ms. then the time the system has to compute each distance function becomes 800 nanoseconds, the current rate. The sections below explain how the circuitry maintains this still high computation rate.



The speech recognition system works in a pipelined and parallel manner as shown in figure 16. The filter bank analyzes the spectrum of the input speech and passes its information on to the endpoint analyzer. At the same time the endpoint analyzer decides when words begin and end. Once the endpoint process has determined that a word has begun it transmits what it

already has of that word to the word comparator despite the fact that it may not yet have determined the end of that word. This pipelining continues throughout the system.

Because of all this pipelining the total response time of the system is the endpoint analyzer's response time plus the time to process one frame. This is  $\text{MAXGAP} + 20\text{ms} \approx 200\text{ms}$  (see the above section on endpoints).



The above diagram shows how dynamic time warping is computed on the comparator IC. A set of programmable logic arrays compute the difference

squared of two features. These squared differences are summed up to compute the total distance function. In order to speed up the distance measure four differences are computed at the same time and thus four cycles through the accumulator loop are sufficient to compute the distance function. Hence with 200 nanosecond cycle times the hardware can compute a squared euclidean distance measure in four cycles or 800 nanoseconds; sufficient to process 1000 template words (500 seconds of speech) in real time. Another adder, comparator and control PLA are used to do the dynamic programming calculation.

### 1.9.3. System Memory Requirements

There are two large memories needed by the speech recognition system. They are the template memory and the column memory.

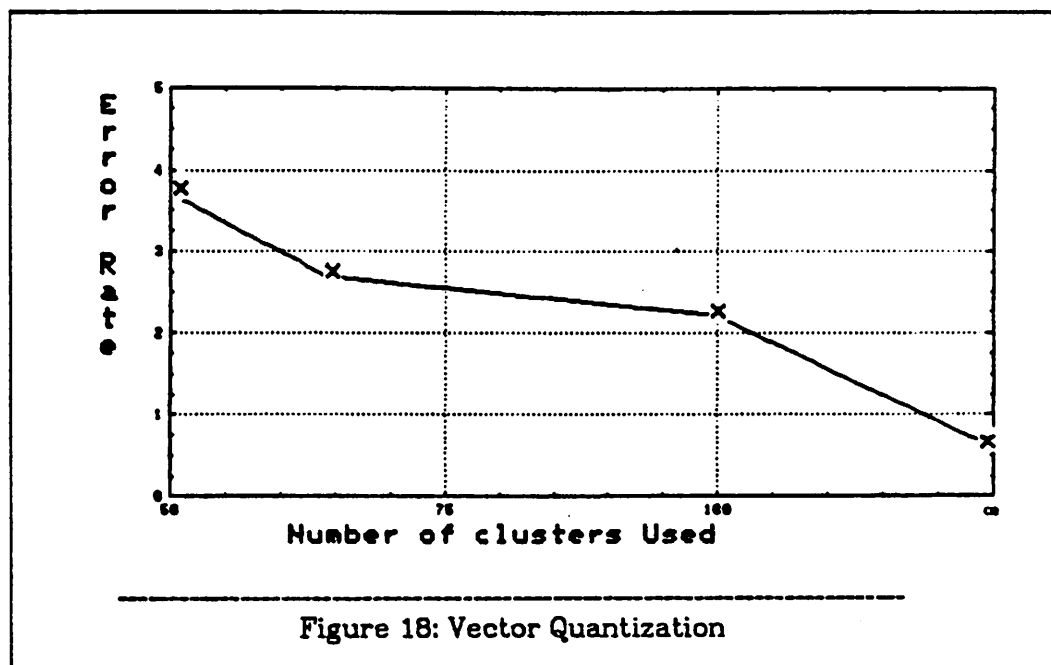
The system must store all of its templates locally and access this entire template memory during each  $T_f$  time interval when the recognizer is active. The memory required for the previous example is large. It is  $\frac{p \cdot b_f}{T_f} \sum_{n=1}^N T_{tp_n}$  where  $p$  is the number of features in each frame and  $b_f$  is the number of bits used to represent these features. In the previous example with  $p=12$  and  $b_f=4$  and  $T_f=20ms.$ , 1,200,000 bits of storage are needed to represent the 1000 words.

Another memory is needed as well. In order to run the comparison algorithm, some information must be passed on from frame to frame. Equation (5) shows that the values  $D_{i,j-1}$  for  $1 \leq i \leq L_{tp}$  and  $1 \leq tp \leq N$  are needed to perform the comparison algorithm on frame  $j$  of the input. Thus the system is required to remember  $\frac{b_D}{T_f} \sum_{n=1}^N T_{tp_n}$  bits for this column memory where  $b_D$  is the number of bits used to represent the  $D_{i,j}$ 's. With  $b_D=12$  and the above

parameters it needs a memory of 300,000 bits.

These memory requirements are relatively large and are the most costly portion of the system. Work is in progress to reduce this problem. As these memories are currently configured external to the comparison IC, one approach is to experiment with intelligent memories that take advantage of data characteristics to reduce the physical memory size. Robert Kavalier,<sup>24</sup> is currently experimenting with vector quantization of the template memory. It is expected that although there are  $p=12$  features per frame each quantized to  $b_f=4$  bits, very few of the possible  $2^{48}$  different possible frames ever occur from natural speech. In fact it is hoped that from  $2^8=64$  to  $2^8=256$  different frames could characterize the entire population of frames within some acceptable tolerance and still have good recognition accuracies. Should  $2^8$  frames or clusters suffice, the system would be able to represent each real frame with 8 bit pointers to cluster centers and the system would only require  $8 \cdot N \cdot \frac{T_{tp}}{T_f} + 256 \cdot 48 \approx 200,000$  bits for 1000 words of template memory. This is one sixth of the original memory size. Kavalier's results are shown below, with the number of vectors used and the percentage of memory used as compared with no vector quantization for a speech recognition experiment with four speakers.

Vector Quantization KIC vocabulary (errors of 2000)				
The Speaker	number of clusters / Percent Memory			
	no clustering / 100%	100 / 15%	64 / 12.8%	50 / 12.7%
RAK	9	31	34	67
RWB	6	50	53	53
ELD	2	8	26	27
HJM	36	89	104	147
Total	53 (0.6%)	178 (2.2%)	217 (2.7%)	294 (3.7%)



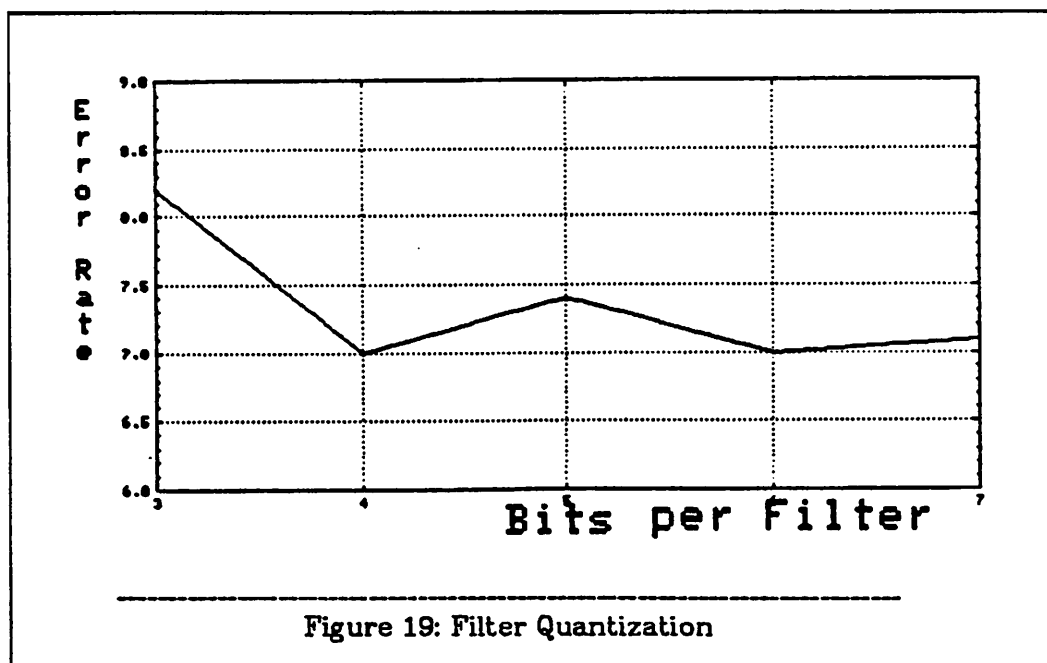
Rabiner et al<sup>5</sup> have done a similar experiment vector quantizing LPC spectra in a speech recognition system. Their results coincide with Kavalier's and show that vector quantization can be used to substantially reduce template memory requirements with still high recognition performance. However the vector quantization does significantly degrade the system performance.

#### 1.9.3.1. Filter Quantization

Since template memory requirements are the most costly part of the speech recognition system, it is advantageous to use as few bits as possible to encode the templates. However using few bits to quantize the filter values will result in quantization noise that can degrade the performance of the recognition system. At the onset of this project we ran an experiment to decide on an acceptable amount of quantization error in the recognition system. As the front end was composed of analog spectral circuitry followed by a  $\mu$ -law analog to digital converter, the effect of using the bits in the converter or just the six, five, four or three most significant bits to represent

data. The table and graph below shows the performance of the system. The vocabulary used in the test is the Alpha-Digits vocabulary discussed at the start of Chapter 2, with five repetitions used to train the system and five other repetitions used to test it. The results are given in percent of errors with no rejection.

Quantization for Filter Values					
Percent Incorrect, alpha-digit Vocabulary					
The Speaker	Bits used to quantize filters				
	7	6	5	4	3
BB	4.7	4.7	5.0	5.4	6.1
BE	11.2	10.1	11.9	11.5	12.2
GW	7.7	7.7	7.7	7.4	7.7
HM	3.7	4.0	4.4	4.0	6.2
JK	7.4	7.0	8.1	6.7	8.8
NL	6.4	6.4	6.1	5.7	5.7
NF	8.8	9.1	8.4	8.4	10.5
Average	7.1%	7.0%	7.4%	7.0%	8.2%



Thus quantizing the filter coefficients with four bits does not harm the performance of the system. All our further work was carried on with four bit

quantized features.

### 1.9.3.2. Word Widths for Column Memory

We represent the distance between two features  $(f_1 - f_2)^2$  with 8 bits, thus there is no error introduced here. We also represent the sum of the differences in our squared euclidean distance measure, equation (6), by 8 bits. Although this does clip certain values it does not affect the recognition performance. Good matches have average errors of 10-20 and our rejection threshold is usually set to 25. Therefore limiting frame error scores to 256 only clips values which will not be considered anyway.

The accumulated score  $D_{i,j}$  computed in equation (10) is limited to 12 bits, more than is necessary as well. This can be shown as follows. Words are rejected if their average scores are greater than 25. In order to reach a score of 4096 (12 bits) the path would have to be 4096/25 or 163 frames long. More than three seconds of speech at a 20ms. average frame rate. Therefore it is the number of bits in this accumulated score which limits the length of the input words for isolated and connected speech.

### 1.9.4. Features of the IC-Based System

Placing the functions of the speech recognition system on two special-purpose integrated circuits has several advantages.

- **High Performance:** The special-purpose design of the speech recognition system allows it to compute certain functions very quickly that would cause computational problems when executed on more general purpose machines. For instance the squared euclidean distance measure computation is done in four clock cycles of the comparator IC (less than a microsecond). This is more than an order of magnitude faster than what

could be done on a high performance microprocessor. Thus a special-purpose design approach allows the speech recognition system to execute algorithms that either could not be considered on general purpose machines, or that could only be used for small vocabularies. The system does not need to prune its searches. That is, it need not reject certain candidate recognition possibilities early (to save computation time), and perhaps degrade performance. With the IC architecture it is actually easier to perform the recognition task fully, rather than decide which words to reject early and which to search fully.

- **Low System Cost:** As a consequence of implementing the algorithms with integrated circuits, the IC-based system's cost will be very small. Certainly this approach is more cost-effective than one that uses a special-purpose breadboard to implement a system as the chip count is much smaller here. However, the IC based system is also more cost-effective than an approach based upon a general purpose microprocessor. This is due to the unnecessary generality of the of the microprocessor. Even though the microprocessor will be made in larger quantities than speech-recognition IC's, the size and technology required for these integrated circuits are smaller and lower in cost than for a microprocessor. Production yields of the smaller devices will be the major effect deciding production costs and eventual prices.

As an example consider the Texas Instrument TMS-320,<sup>17</sup> a family of modern complex digital signal processors. In fact the 320 has been considered the highest performance monolithic signal processor today.<sup>39</sup> Texas Instruments in its product description for the 320 "boasts" that the 320 can be used as a forty word speech recognition system. Contrast



this to the one thousand word capability of our special-purpose system which is built with less advanced integrated circuit processes and uses slower internal clocking speeds. Even with only moderate production volume the special purpose approach is sure to be a more cost effective solution.

- **Size:** The low chip count of the system makes it very small. All that will be needed for the speech recognition part of a system is a microphone two integrated circuits, memory and a host microprocessor.
- **Flexibility:** The speech recognition system can compare an input word with one thousand word templates in real time. This translates to recognizing a vocabulary of five hundred to a thousand words for a single speaker or fifty words with some speaker independence. This can be expanded further by placing in parallel replicas of the comparator IC's with memory. Also because of its computational speed and the fact that it does not need to prune its searches, the IC based speech recognition system can recognize connected as well as isolated speech. This will be discussed in chapter 3.
- **Further Experimentation:** Having a system that can compare the input fully with a thousand template words in real time allows us to experiment with systems of this size in real time. In this way research capabilities are enhanced. For instance we are able to have real time response for large vocabularies and in this way examine the special problems that such vocabularies have without long running computer simulations.

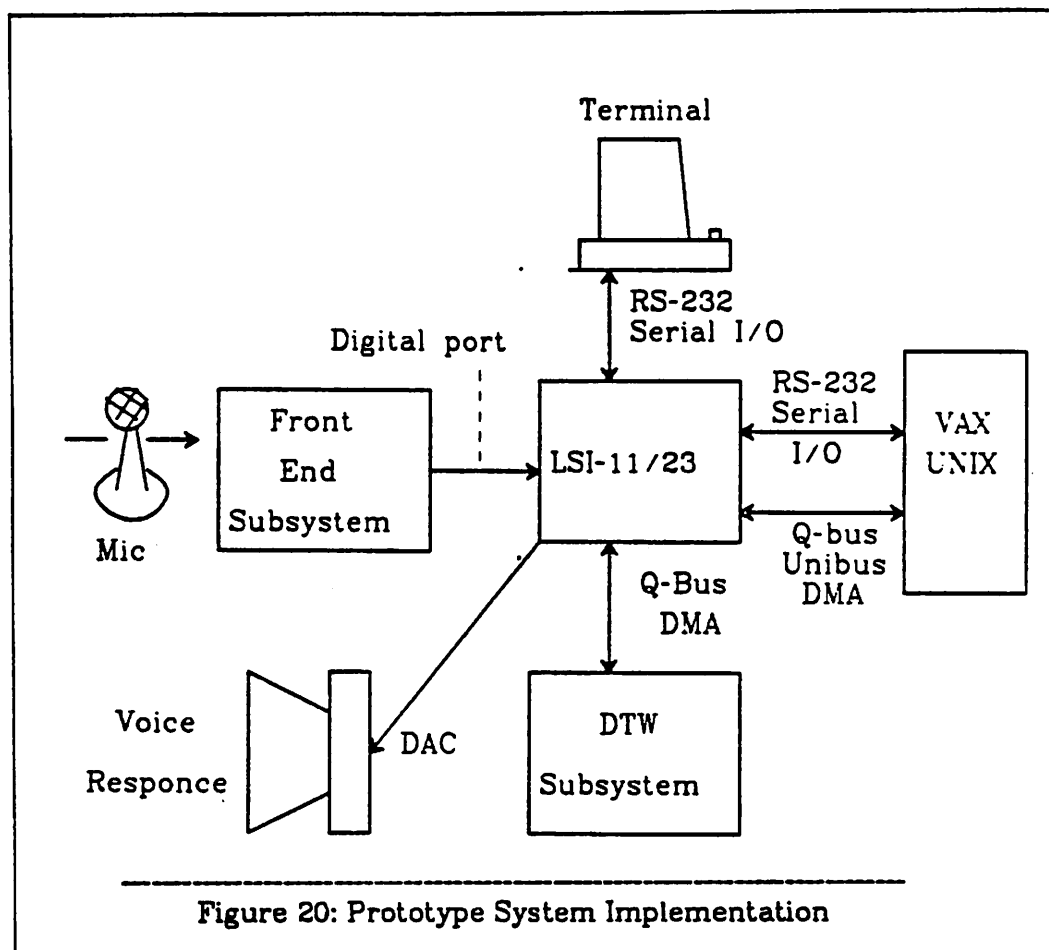
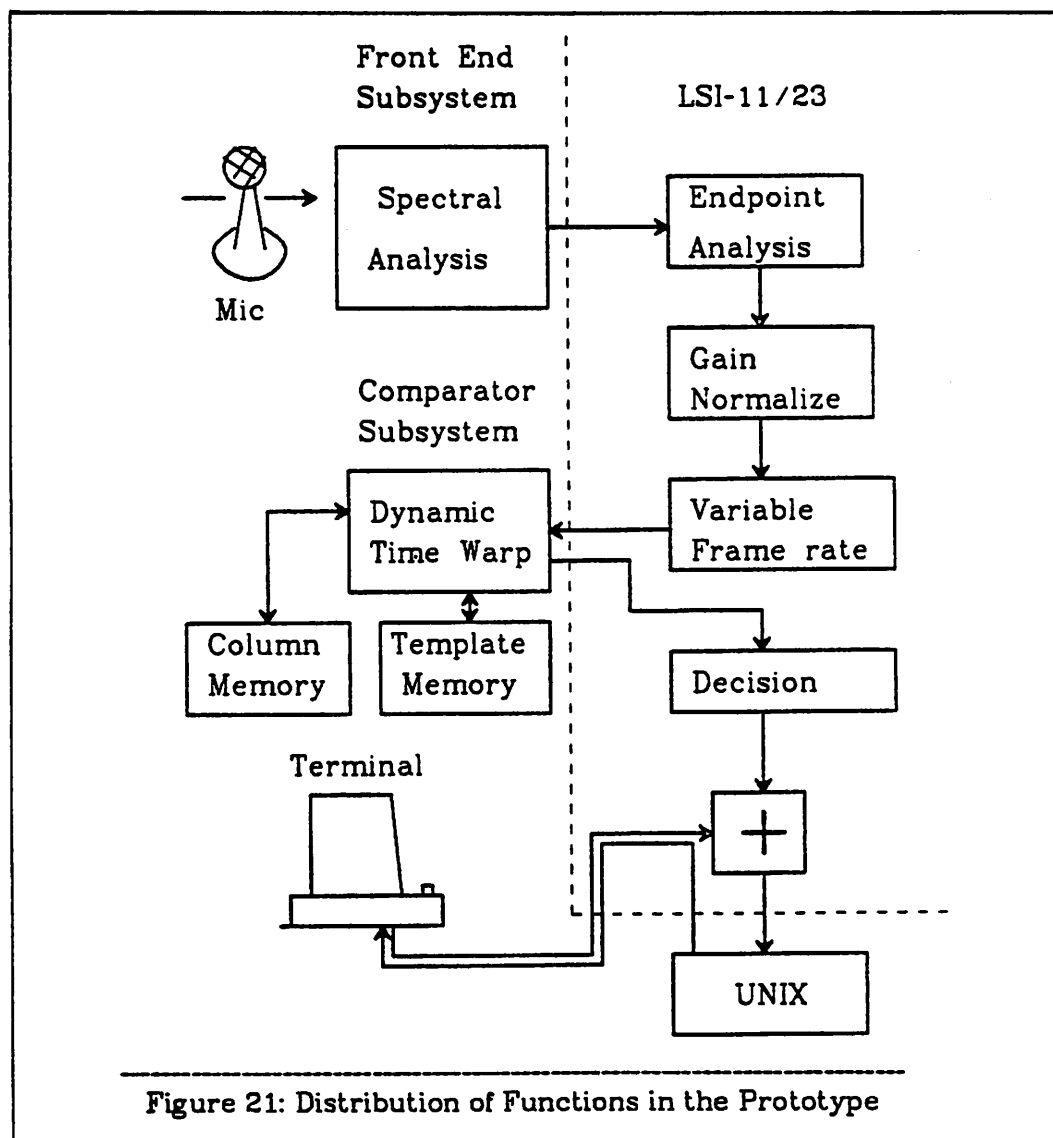


Figure 20: Prototype System Implementation

### 1.10. Current System Implementation

Our final goal is to develop a speech recognition system that is implemented by a set of special-purpose integrated circuits. In the process of doing this we have designed and have working a prototype system based on one special purpose IC together with TTL small scale and medium scale integrated circuits. The prototype speech recognition system shown in the figure above executes programs on a Digital Equipment Corporation LSI-11/23 microprocessor and has two associated custom made subsystems, one to perform spectral analysis of the spoken input,<sup>40</sup> and one to perform the comparison algorithms associated with the speech recognition system.<sup>28</sup>

These are analogous to the *front-end* and *comparator* integrated circuits in the IC based system. This system is currently being used primarily as speech input to integrated circuit layout programs. This section describes this implementation.

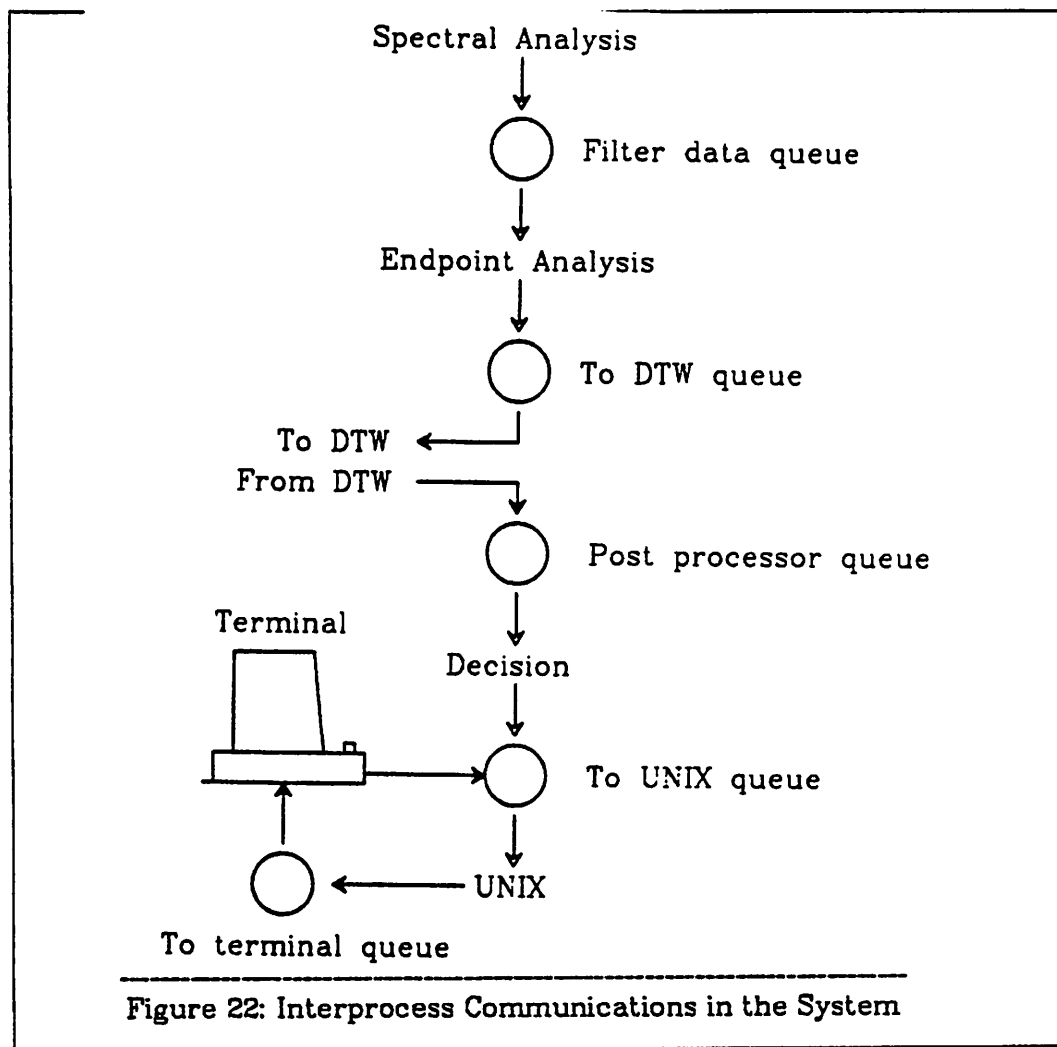


### 1.10.1. The Distribution of Functions

As can be seen on the above figure, the LSI-11/23 along with the filter-bank hardware performs the duties of the front-end IC. They compute the

spectra, take its logarithm, compute endpoints and downsample the data. The LSI-11/23 along with the pattern-matching circuitry perform the duties of word-comparison and recognition or rejection. The LSI-11/23 also performs the action associated with any recognition decision it makes.

The microprocessor and its speech recognition hardware and console are interfaced to a VAX-UNIX computer system. The LSI-11/23 receives characters from UNIX and sends them to the console's screen. It receives characters from the console's keyboard and sends them to UNIX. Usually the actions related to recognizing a word also involve sending a character string to UNIX. Any UNIX program that was originally written to accept keyboard input can be used with the speech recognition system either accepting typed input or spoken input translated to characters by the speech recognition system. Such programs include the KIC <sup>41</sup> and CAESAR <sup>42</sup> integrated circuit layout systems designed at Berkeley. In these cases the LSI-11/23's console becomes the terminal used with the layout system with its associated bitpad for pointing at items on the screen. Using the speech recognition system the user has the option of pointing, typing, or speaking to communicate with the IC layout programs.



### 1.10.2. Filter Hardware

When speech enters the ASR system its short-time spectrum is computed every 10ms. by the filter-bank hardware. The filter-bank runs continuously, not caring whether its data has been read or not. There is a 16-bit parallel digital port between the filter-bank and the 11/23.

Every ten milliseconds the filter hardware outputs a 16 bit word for each of the twelve filters it computes. The lower order bits are the  $\mu$ 255-law values of the filter bands and the upper byte is synchronization information. At sys-

tem start-up the 11/23 synchronizes itself with the filter-bank hardware by waiting for the filter bank to begin a frame (start from filter number 1). From then on whenever the filter bank computes a filter coefficient and sends that word to the LSI-11/23 port an LSI-11/23 interrupt is generated. The 11/23 responds to the interrupt by reading the port and queuing the filter data in the LSI-11/23's memory. This data is soon sent to the endpoint algorithm where frames that are not part of words are stopped. Frames that are in words are queued up to be sent to the comparator hardware.

### **1.10.3. Dynamic Time Warp Hardware**

The comparator hardware has a non-DMA parallel input port (from the 11/23) and a DMA output port (back to the 11/23). The input port accepts frame data that has passed through the endpoint algorithm as well as commands to alter its mode of operation. The commands choose between "training mode" or "recognition mode", and "isolated speech mode" or "connected speech mode."

At the start of word recognition the 11/23 sets the dynamic time warp hardware in the appropriate modes and raises the begin recognition flag. At this point, when the hardware is ready to accept data, it interrupts the 11/23 which then transmits the 12 features of the first unknown frame's data to the time warp hardware. The 11/23 also initiates a DMA cycle for the output of the comparator to that frame's data. When the DMA is complete, the hardware has completed the computations for the first column of the dynamic time warp algorithm, the output of the hardware is queued and processed. This cycle repeats until all frames of the unknown have been transmitted to the hardware and all columns have been computed. When an input word is complete the 11/23 (looking at the output of the comparator)

makes a recognition decision.

#### **1.10.4. Character Input and Output**

The 11/23 can also be interrupted by its two RS-232 ports, one communication with its console and the other communication with UNIX. It also has a DMA link to the UNIX system which is used to download the programs that run in the 11/23. It is also used to download templates stored on disk to the recognition system before recognition begins, and to upload templates from the recognizer to UNIX during the user training mode.

#### **1.10.5. Summary**

There are several processes running concurrently inside the speech recognition system.

The processes internal to the 11/23 are:

- **The filter-bank normalization and queuing operation**
- **The endpoint algorithm**
- **The process that transmits to the comparator**
- **The receiver from the comparator**
- **The recognition decision algorithm**
- **The character I/O routines to and from the console and UNIX.**

The other processes external to the 11/23 are:

- **The filter-bank hardware**
- **The comparator hardware**
- **The UNIX process with which the user is interacting**

The programs that run in the LSI-11/23 have all been almost completely written in the C programming language using a standard UNIX C compiler. The exceptions include low core tables needed to set up interrupt vectors and some routines that must run very quickly such as the interrupt routines that handle character I/O.

## CHAPTER 2

### System Design Decisions

#### 2.1. Introduction

The following sections describe some of the alternatives considered in the design of the speech recognition system. All of the studies use the system that was described in the previous chapter; an isolated-word speech recognition system that uses whole-word templates, all of which are compared to an input word via a dynamic time warped comparison measure based on a frame by frame spectral distance measure. The input and template words are represented as a series of frames. Frames are a set of band-pass power values time windowed in the input speech.

#### 2.2. Testing Data Base

The studies described in this chapter use the following test data base. The first vocabulary is an augmented version of the *Alpha-Digits vocabulary*. Its words are listed in the table below. Ten repetitions of these words were recorded from seven different speakers. There were three men (NF, HM, BB) and four women (MC, KH, GW, CW) recorded. Five of these repetitions were used to train the recognition system and five were used to test it.

The second vocabulary used is one that can be used with the KIC<sup>41</sup> integrated circuit layout system. In this case twenty repetitions of each word were recorded from four speakers, two men (HJM RAK) and two women (SUE LES). The first ten repetitions were recorded several days before the last ten and those first ten were used in training whereas the last were used



in testing.

All recordings were made in a low-noise environment with a close talking microphone (Shure sm-10).<sup>43</sup> The bandwidth of the input signal was greater than the analysis band of the front end, 200-8400hz. The speakers were prompted to say their words by a computer program. Spectra were then extracted from these words in real time and placed onto disk files for later experimentation. Data was also saved on analog reel to reel tapes. In all cases the test and training data bases were separate and the same test and training data was used in all of the experiments that follow.

1	b	g	newline	times
2	c	h	o	u
3	d	i	p	v
4	delete	j	plus	w
5	divide	k	point	x
6	e	kill	q	y
7	equals	l	r	zero
8	erase	m	s	
9	error	minus	space	
a	f	n	t	

0	basic	expand	menu	return
1	blink	fill	metal	right
180	bottom	flash	mirror_in_x	save
2	boxes	glass	mirror_in_y	select
270	buried	grid	more_blue	stretch_box
3	changelayer	hilight	more_green	subcell
4	child	implant	more_red	symbolic
45	cmos	instance	move	technology
5	colors	kic	nmos	top
6	copy	kill	no	undo
7	cut	label	outline	up
8	delete	last	pan	update
9	deselect	layer	parent	visible
90	diffusion	left	peek	width
abort	dimension	less_blue	poly	window
adder	directory	less_green	polygon	wires
addlayer	donut	less_red	pop	writeout
arc	down	lyra	push	yes
area	edit	maincell	redraw	zoom
attribute	erase	memory	removelayer	zoom_full

The KIC vocabulary is a vocabulary that is typical for applications of the speech recognition system. The speech recognizer performs very well with such vocabularies. The Alpha-Digits vocabulary is a much more difficult vocabulary for the speech-recognition system because of its many similar words. For example about half the errors of the 47 word Alpha-digit vocabulary typically come from the 10 word *E-set* of that vocabulary; the words *3, b, c, d, e, g, p, t, v* and *z*.

Below is a table showing the relative performance of the KIC and Alpha-Digits vocabularies as well as the performance of men versus women using the speech recognition system with no rejection of words. Note that as expected KIC performs much better than Alpha-Digits. Women seem to perform slightly better than men.

<b>Percentage of Errors</b>					
<b>KIC</b>			<b>Alpha-Digits</b>		
<b>men</b>	<b>women</b>	<b>both</b>	<b>men</b>	<b>women</b>	<b>both</b>
<b>0.9%</b>	<b>0.4%</b>	<b>0.6%</b>	<b>4.1%</b>	<b>3.9%</b>	<b>4.0%</b>

## 2.3. Front End

### 2.3.1. Selective Downsampling

The speech recognition system dynamically varies the spectral sampling rate of input speech. It does this by sampling the input speech spectra at a very high rate and downsampling or disregarding samples according to the following criteria. Let  $F_i$  correspond to an input frame and let  $F_r$  correspond to a reference frame.

If  $F_i$  is the first frame in a word

Or  $d(F_i, F_r) \geq DS\_THRESH$

Accept  $F_i$

$F_r = F_i$

Else Disregard  $F_i$ .

$DS\_THRESH$  is a value determined to optimize the performance of the system, both in data rate and recognition accuracy. Increasing  $DS\_THRESH$  will decrease the effective spectral sampling rate. The amount of reduction in data rate is a function of that threshold, the vocabulary in use and the current speaker. Recall that an effective sampling rate of twenty milliseconds was required in order for the system to process 500 seconds of speech in real time.

The Alpha-Digits and the KIC vocabularies were used to study the effect of downsampling. In the alpha-digits vocabulary, downsampling was expected to improve performance dramatically because the major distinctions between the similar words in that vocabulary often lie in the short and spectrally changing consonants and vowel transitions beginning the words. This is particularly the case in the previously mentioned *E-set*. These important word distinctions can be underemphasized by the recognition system

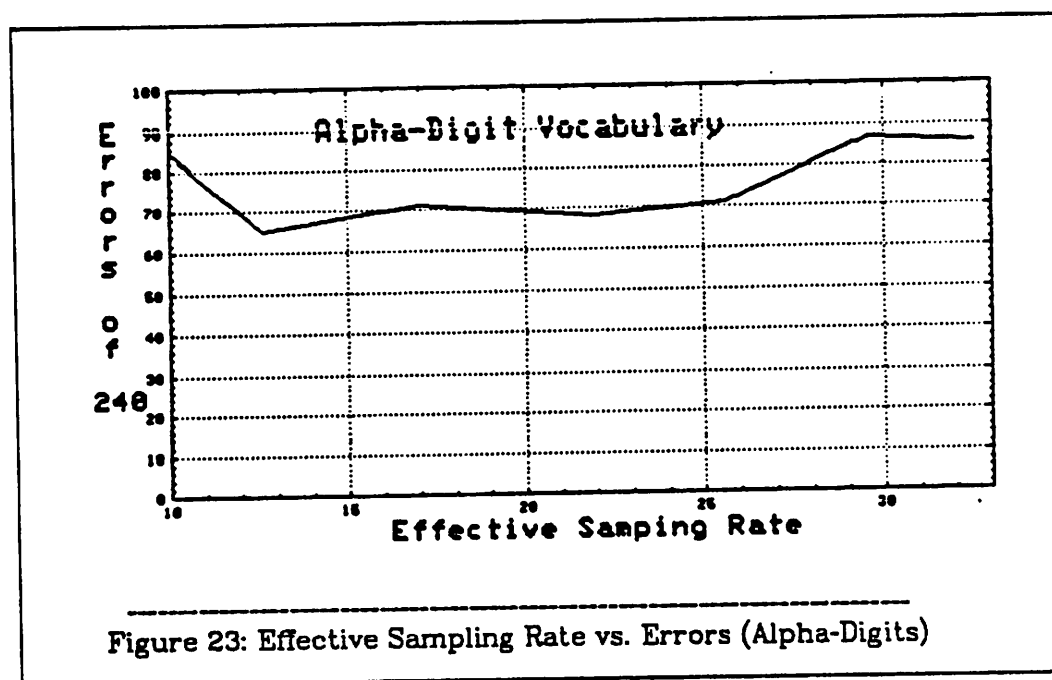
because of minor variations in the much longer vowel portion of the words. Downsampling eliminates large portions of the steady-state vowel sounds and reduces their influence on the recognition decision. However, there is little downsampling in the changing portions of words. Therefore downsampling places increased emphasis on the important changing portions of the words relative to the steady-state portions.

The study shows that this is true, though only a small amount of downsampling is required to attain improved performance levels.

The same advantages of downsampling described above apply to the KIC vocabulary as well. However here words are more distinct and thus the main source of error is not a basic difficulty in distinguishing words. Most errors are due to errors in the endpoint algorithm. For instance a substantial period of silence followed by a click may be appended to an input or template word. Downsampling decreases emphasis on such silences in words just as it does with steady-state speech sounds. For this reason downsampling substantially improves performance with the KIC vocabulary.

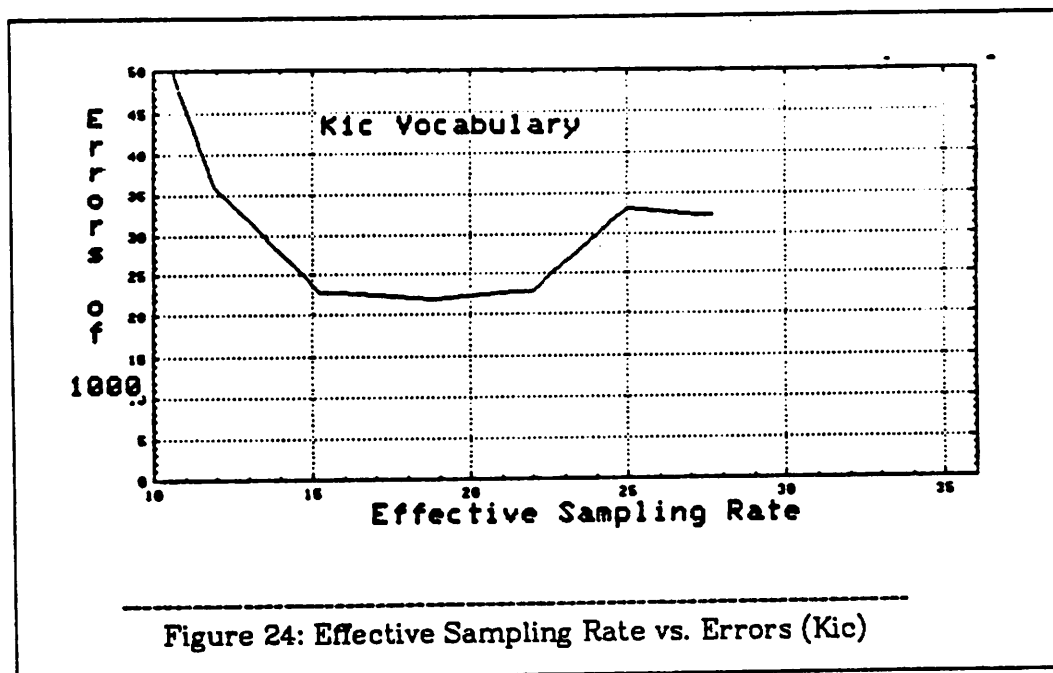
The table below shows results of an experiment where *DS\_THRESH* was systematically varied from no downsampling to a great deal of downsampling for the two vocabularies in question and the *E-set*. It also shows the effective spectral sampling periods as a function of *DS\_THRESH*. It shows that good performance is achieved with spectral sampling periods of more than twenty milliseconds.

Variable Bit rate experiment Alpha-Digits (words wrong of 240)							
The Speaker	Threshold / effective sampling period (ms.)						
	0/10.0	2/12.6	4/17.0	6/21.9	8/25.6	10/29.6	12/32.5
BB	8	7	8	6	12	10	11
CW	16	11	11	9	6	9	9
GW	8	8	10	12	10	12	11
HM	10	6	8	6	10	8	13
KH	17	19	19	16	18	24	23
MC	13	2	3	6	6	12	7
NF	13	12	12	13	9	12	12
Total	85	65	71	68	71	87	86



Variable Bit rate experiment E-set (words wrong of 350)							
	Threshold / effective sampling period (ms.)						
	0/10.0	2/13.5	4/19.0	6/24.3	8/29.2	10/33.8	12/40.3
Total	43	33	36	35	36	42	44

Variable Bit rate experiment KIC Vocabulary (words wrong of 1000)							
The Speaker	Threshold / effective sampling period (ms.)						
	0/10.0	2/11.9	4/15.2	6/18.8	8/22.0	10/25.0	12/27.7
HJM	24	15	7	7	10	16	14
LES	6	1	1	1	1	2	3
SUE	6	5	4	4	4	4	3
RAK	20	15	12	10	8	11	12
Total	58	36	23	22	23	33	32



## 2.4. Comparator

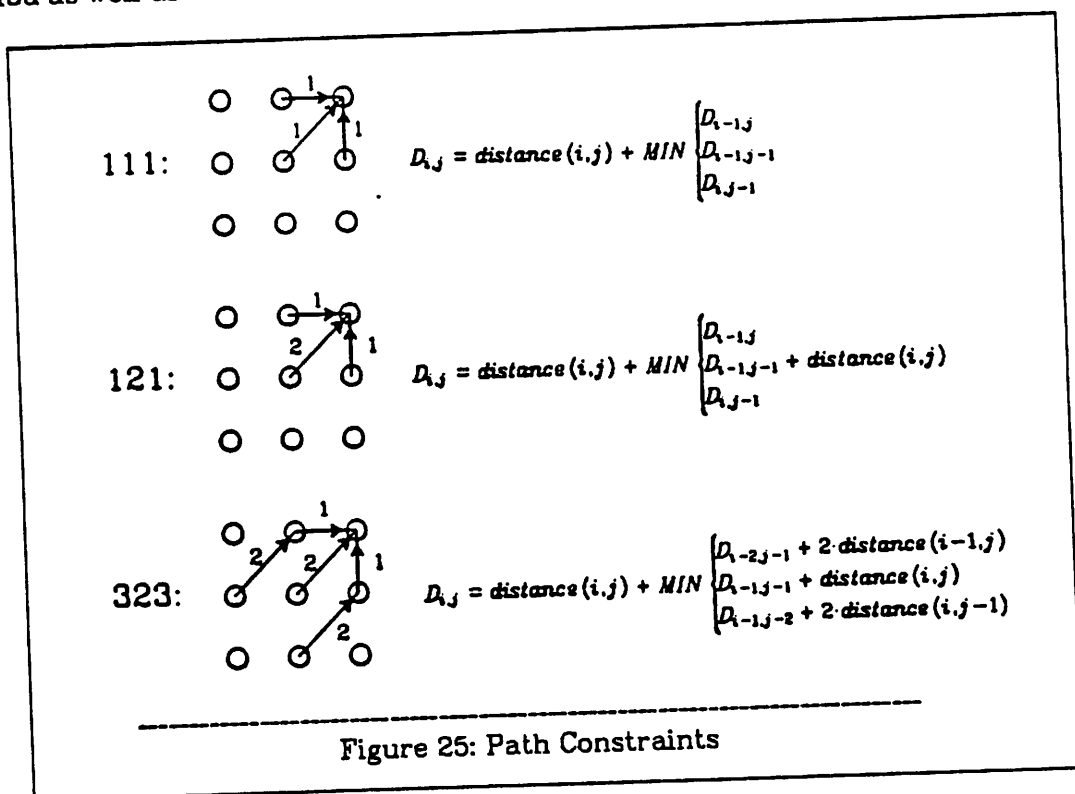
### 2.4.1. Path Selection

As I have discussed in chapter 1 equations 8 and 9, it is necessary to place some restrictions on the path chosen for the dynamic time-warp algorithm. There have been several proposed techniques in addition to ours. All involve changes to the equations that constrain the path functions  $P_{in}$  and  $P_{out}$  in chapter 1 equations 8, 9 and 10.

Sakoe and Chiba<sup>37</sup> have published a study where they describe several alternative schemes to dynamic time warping. They conclude that the best

time warping algorithms are those that are symmetrical (the constraints treat the template and the unknown identically) and those that impose slope constraints on the warp path. We have chosen a technique that they did not consider. It is symmetrical but does not directly impose slope constraints on the warped path. Our scheme (111 on the figure below) performs at least as well as the ones considered by Sakoe and Chiba and it is more convenient to implement in an IC based speech recognition system because of smaller memory requirements.

Below is a chart showing path constraints considered by Sakoe and Chiba as well as the one we have implemented.



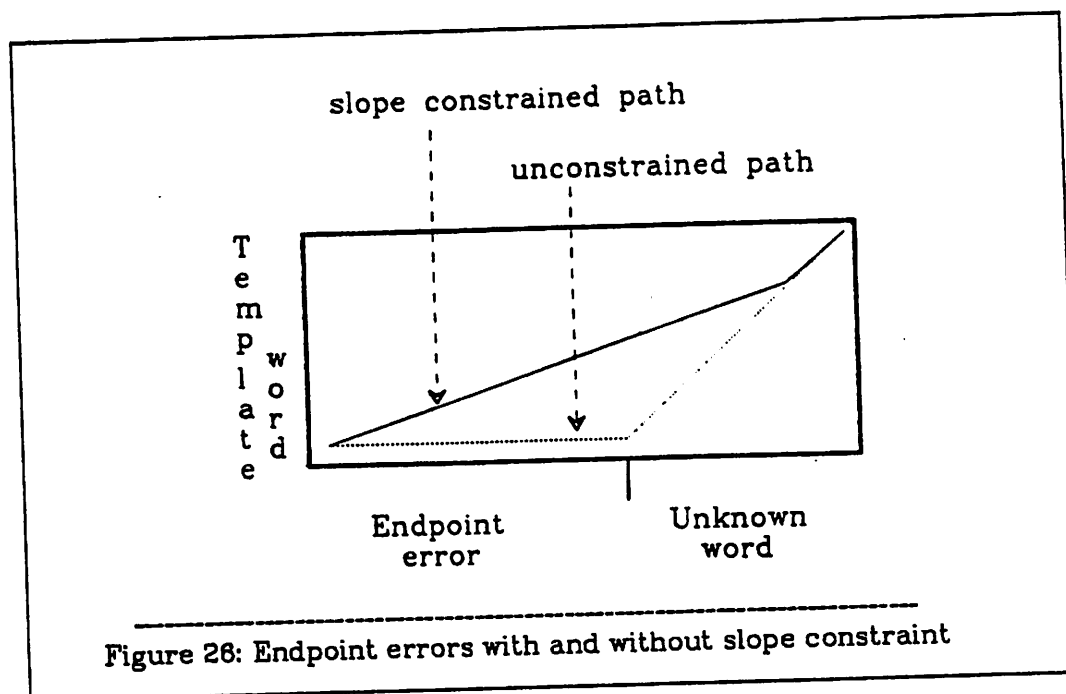
Note that technique 111 (the one we have chosen) requires one column of scratch memory for its implementation. That is, the DTW equations do not refer to  $D_{i,j}$  where  $\hat{j} < j-1$ . When computing the  $j^{\text{th}}$  column, the values of



$D_{i,j-1}$  are required and must be stored. However the values  $D_{i,j-2}$  are not needed and can be discarded. This is important since every column requires a great deal of memory.  $W \cdot F$  words of memory are required per column where  $W$  = the number of words and  $F$  = the average number of frames per word. This is about  $1000 \cdot 25 = 25,000$  12 bit values or 300,000 bits of memory in the speech recognition system. Method 323 shown above requires two columns of storage and therefore twice as much scratch memory.

Technique 111 as well as the other two shown above are symmetrical; either the template or the input word can be stretched or compressed relative to one and another.

Finally note that technique 111 does not have a slope constraint. Slope constraint restricts the warping paths. A slope constraint of (2,.5) such as in method 323 would not allow more than two frames of one word to be warped with only one frame of the other word. Although Sakoe and Chiba report that slope constraint aids recognition performance our tests show that just the opposite is true. These experiments point out that endpoint errors, which are one of the most significant sources of error in the system, are less severe without slope constraints. This is shown graphically in the figure.



In figure 26 the path without slope constraint is able to quickly find the correct path whereas the one with slope constraint never really finds the correct path. The constrained path also has almost no freedom to warp its path to avoid local high scores as in trying to reach the correct path it is confined to a minimal slope. All the advantages of dynamic programming are lost. Slope constraint is also difficult to implement with a variable frame rate scheme as the frame skipping distorts the time axis which slope constraint requires.

The advantage of slope constraint is that paths that are "not speech-like" are not allowed. That is, for example, paths that warp one frame of one word into 20 frames of the other are not allowed. In this example 10 ms. of one word is being compared to 200 ms. of the other. This is an expansion that does not normally occur in natural speech. However, as shown in the previous figure, what may be considered "not speech-like" may indeed occur due to endpoint errors. I believe that the positive results Sakoe and Chiba

achieved were due to the fact that endpoint errors did not figure in their experiments. However in a realistic system endpoint errors will be a major concern.

Below is a table showing the results of a speech recognition experiment where only the dynamic programming equations shown in figure 2 are varied. Recall that algorithms 121 and 323 are identical except that the latter is a slope constrained version of the former. Note that the algorithms not slope constrained greatly outperform the slope constrained algorithm. The major difference between algorithm 121 and the algorithm used in the speech recognition system (both are symmetrical and not slope constrained) is the type of path normalization used. This will be discussed in the next section.

The Speaker	Path algorithm		
	121	323	111
BB	14	13	12
CW	9	10	6
GW	10	15	10
HM	12	14	9
KH	17	22	18
MC	5	10	4
NF	11	9	9
Total	78	93	68

The Speaker	Path algorithm		
	121	323	111
HJM	13	17	8
SUE	2	8	3
LES	1	3	4
RAK	13	24	10
Total	29	52	25

#### 2.4.2. Path Normalization

In chapter one equation 7 the error in comparing one word to another was defined to be

$$\text{dist}(W_{in}, W_{tp}) = \frac{1}{L_p} \sum_{t=1}^{L_p} d(P_{in}(t), P_{tp}(t)). \quad (1)$$

In order to find the value of  $L_p$  in the above equation the following calculation would have to be performed on every iteration of the dynamic time warp calculation of  $D_{i,j}$

$$D_{i,j} = d(W_{in}(i), W_{tp}(j)) + \text{MIN}(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}), \quad (2)$$

$$L_{i,j} = \begin{cases} L_{i,j-1} + 1 & \text{if } \text{MIN}(D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}) = D_{i,j-1} \\ L_{i-1,j} + 1 & \text{if } \text{MIN}(D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}) = D_{i-1,j} \\ L_{i-1,j-1} + 1 & \text{if } \text{MIN}(D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}) = D_{i-1,j-1} \end{cases}$$

$$L_{0,0} = 0, \quad L_p = L_{L_{in}, L_{tp}}.$$

Although the computations for  $L_{i,j}$  are trivial since the MIN must be computed anyway for the DTW calculation, the memory required is not. The system would have to store one column of  $L_{i,j}$  or  $\frac{1}{T_f} \sum_{n=1}^N T_{tpn}$  at six or seven bits each. That would come to about 150K bits of memory. (These variables were explained in chapter 1 section 2.5.3.)

In the implementation of the speech recognition system the word distance equation is approximated by

$$\text{dist}(W_{in}, W_{tp}) = \frac{1}{\text{MAX}(L_{in}, L_{tp})} \sum_{t=1}^{L_p} d(W_{in}[P_{in}(t)], W_{tp}[P_{tp}(t)]). \quad (3)$$

$L_p$  is approximated by  $\text{MAX}(L_{in}, L_{tp})$  in the denominator of equation (11) in chapter 1. This approximation for  $L_p$  is the minimum possible value of  $L_p$  and corresponds to the most direct warping path. On the other extreme the maximum  $L_p$  possible is  $L_{in} + L_{tp}$  which is a highly warped and "not speech-like" path. This path would clearly indicate a mismatch. The approximation introduced encourages less warped paths. It does so because paths that are more roundabout are still only divided by the minimum length path and therefore their score is higher than the average spectral error along the

path. This is an implicit slope constraint that improves performance because it discourages "un-speech-like" paths while allowing for recovery to endpoint errors. Experimental results shown indicate that the system actually performs better with the approximation used than with the exact calculations which would have required significant extra memory requirements. Further, this is the reason that our system outperforms algorithm 121 from last section, since algorithm 121, similar to the exact normalization for algorithm 111 does not discourage "un-speech-like" paths.

The Speaker	The Normalization Used		
	Exact 121	Exact 111	Approximate 111
BB	14	12	12
CW	9	8	6
GW	10	9	10
HM	12	9	9
KH	17	20	18
MC	5	6	4
NF	11	11	9
Total	78	75	68

The Speaker	The Normalization Used		
	Exact 121	Exact 111	Approximate 111
HJM	13	15	8
SUE	2	6	3
LES	1	1	4
RAK	13	13	10
Total	29	35	25

## 2.5. Training Algorithms

A series of algorithms were introduced in the first chapter that train the speech recognition system based on a set of input templates for each vocabulary word. These algorithms range from very simple to complex and the burden on the user increases with the complexity.

A study of the performance of the recognition system follows comparing various ways to train it. There are five techniques considered.

- **Single:** This refers to using only a single word to train the system. This word was randomly chosen from the five words that were used to train the Alpha-Digit vocabulary.
- **KNN-1:** This algorithm uses all five training repetitions as templates in recognition mode. It picks as the word recognized the lowest scoring template of all templates. Each of the forty-seven Alpha-Digit vocabulary words has five templates thus there are two hundred and thirty-five templates used in these recognition runs.
- **KNN-3, KNN-5 :** This is the same as KNN-1 except that it allows the three (five) best scoring templates to vote for the recognized word.
- **Cluster:** This is the algorithm described in Chapter one section 2.4 as the *Cluster Centers* algorithm.
- **Average:** This was also described in the earlier section as *Clustering with Averaging*.

The following study shows that averaging is the best scheme from a consideration of performance given memory used. That is averaging uses many fewer templates than the KNN schemes and typically not many more than a single template scheme and yet it achieves substantially the performance of the best scheme using all the templates (KNN-1). The KNN scheme that performed best was the one that just used the top scoring template. This is true because there weren't enough training templates to make a KNN scheme viable. A study in chapter 4 shows that when using many copies of each word KNN schemes with  $k > 1$  perform very well.

The Speaker	training algorithm					average
	single	cluster	knn-1	knn-3	knn-5	
BB	23	20	8	11	10	12
CW	23	13	13	10	8	6
GW	29	17	8	9	15	10
HM	25	15	10	7	12	9
KH	43	24	10	22	19	18
MC	30	12	6	4	9	4
NF	28	17	10	11	14	9
Total	201	117	65	74	87	68

## 2.6. Comparison with Other Systems

George Doddington<sup>15</sup> published a comparison of commercially available speech recognition systems. We obtained the audio tapes he used to evaluate these systems and ran them through our recognizer.<sup>12</sup> The table shown below is taken from Doddington's article with an extra line inserted; the system labeled Berkeley is our prototype recognition system with clustering (but no averaging) used to train it, the system labeled cluster above. The averaging algorithm had not yet been developed at the time this study was done.

The system does do very well in this comparison, especially given the inferior training method. Though there is no cost associated with our system, with an integrated circuit implementation it should cost between ten and a few hundred dollars.

Manufacturer	Model	Nominal price	errors
Verbex	1800	\$65,000	10 (0.2%)
UC Berkeley	—	—	45 (0.9%)
Nippon Electric	DP-100	\$65,000	60 (1.2%)
Threshold Technology	T-500	\$12,000	73 (1.4%)
Interstate Electronics	VRM	\$2,400	147 (2.9%)
Heuristics	7000	\$3,300	300 (5.9%)
Centigram	MIKE 4725	\$3,500	366 (7.1%)
Scott Instruments	VET/1	\$500	646 (12.6%)

## 2.7. Quantization and Filter Variance

In order to make better use of the four bits that were chosen to represent the filter coefficients, the euclidean distance measure used was modified by taking into account the dimensional variances of the feature space. Instead of computing the distance measure as

$$d(F^a, F^b) = \sum_{i=1}^p [F_i^a - F_i^b]^2$$

define it as

$$d(F^a, F^b) = \sum_{i=1}^p \frac{[F_i^a - F_i^b]^2}{\sigma_i^2}$$

Here  $\sigma_i$  is an estimate of the standard deviation of the  $i^{\text{th}}$  filter coefficient. To reduce the quantization error for the filter data and to eliminate the divide from the distance measure encode the filter coefficients as

$$\hat{F}_i = \frac{(F_i - \bar{f}_i)}{\sigma_i} \cdot C1 + C2$$

where C1 and C2 adjust the range of the filter values so that they properly cover the range of 0-15 (for four bit quantization) and  $\bar{f}_i$  is an estimate of the mean value of the  $i^{\text{th}}$  filter coefficient.

The following experiments show the performance of the speech recognition system when  $\bar{f}_i$  and  $\sigma_i$  were estimated from the training data. In these cases C2 was set to 8, midway in the range of 0-15, and C1 was set so that the four bit filter coefficients can represent a  $\pm 3\sigma$  range for the filter data.



<b>Normalized means and Variances (Alpha Digits)</b>		
<b>The Speaker</b>	<b>Normalized Variance</b>	<b>Not Normalized</b>
BB	9	12
CW	5	6
GW	13	10
HM	6	9
KH	15	18
MC	4	4
NF	7	9
<b>Total</b>	<b>59</b>	<b>68</b>

A better weighting of the features improves recognition performance. This gain does not require any extra computation in the comparison process as all the filter values are modified in the front end. The extra front end processing is minimal. It is only a look up in a sixteen word table. This technique is now being used in our speech recognition system.

## CHAPTER 3

### Connected-Speech Recognition

#### 3.1. Introduction

Speech recognition systems that allow connected-speech input have several advantages over isolated-word systems. For one thing they are easier to use. The user need not pause before and after words when interacting with the system. Though this form of speech input is preferable, it is also more difficult to recognize automatically. Two major problems associated with connected-speech recognition are finding the beginnings and endings of individual words in a phrase and allowing for coarticulation between connected words.

There have been many attempts at connected-speech recognition over the past few years. These attempts can be roughly broken up into two approaches. Knowledge based attempts try to use high and low level information that has been discovered about speech to find word beginnings and endings in running speech and to classify those words in the presence of coarticulation.<sup>1</sup> These attempts typically require much computation time to search knowledge bases and to make inferences about the input speech. Though these approaches are very promising, the large system complexity necessary for systems such as these imply response times that are too slow to make those systems cost-effective devices today.

Another approach to connected-speech recognition is applying techniques that have had success with isolated speech to connected speech. This type of system can respond to speech in real time and is in use. It

algorithmically solves the problem of finding endpoints of individual words in a connected-speech phrase. However it ignores the problem of coarticulation between the input words. Therefore the performance of this type of system is related to the amount of coarticulation in the input speech. That is these systems are more sensitive to how carefully words are articulated.

We have taken the latter approach for our connected-speech recognition system. The isolated-word based speech-recognition system described earlier was adapted to recognize either isolated or connected speech. Further it can do so with no penalty in response time. It can also be implemented with the same integrated circuits that were described above.

Several other researchers have presented systems that are functionally similar to this one.<sup>44,45,46</sup> Their results and those in the following studies show that these systems work well for connected words spoken quite naturally.

### 3.2. Connected-Word Algorithm

The connected-word recognition system uses the same front-end to extract speech parameters as the isolated-word system. It also uses the same endpoint algorithm. In this case however phrase endpoints are computed instead of word endpoints. That is the endpoint algorithm detects where a series of words begins and ends. This phrase is passed through a dynamic time-warp algorithm similar to that used for isolated speech. The output of this algorithm is the string of template words that when concatenated match the input frames better than any other concatenation of templates. This series of templates become the recognized templates. Reject thresholds are again used before any actions are taken on the recognition results.

The word templates used here are the same ones as those that are used for isolated-speech recognition.† Therefore this algorithm is known as *isolated-word based connected-word recognition*.

This is the time warping algorithm used:

$$D_{i,j} = d(W_{in}(i), W_{tp}(j)) + \text{MIN} \left\{ D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1} \right\} \quad (1)$$

$$D_{0,0} = 0$$

$$D_{i,0} = \infty \quad i > 0$$

$$D_{0,j} = \text{init}(j), \quad j > 0$$

$D_{L_{in}, L_{tp}}$  is the accumulated distance between the best set of concatenated templates and the input.

The above equations (1) are identical to those used for isolated speech except for the initial column values  $D_{0,j}$  defined as  $\text{init}(j)$ .  $D_{0,j}$ , the initial condition to the time-warp algorithm at time  $j$ , is the error associated with the best concatenation of templates to match the input from frames 0 to  $j-1$ . One way to compute that value would be to set

$$\text{init}(j) = \text{MIN}_{tp} \left\{ D(tp)_{L_{tp}, j-1} \right\} \quad (2)$$

That is, pick as the bottom of column  $j$  (for each template) the top of column  $j-1$  for the template which had the minimum top value. This corresponds to the score associated with a sequence of templates best matching the input from the first frame to frame  $j-1$ . Call that minimum template  $Tp(j-1)$  which had a score  $Top(j-1)$ . Another value needed is the frame which began the time warp path for  $Tp(j-1)$  that ended at  $j-1$  with score  $Top(j-1)$ .

†Rabiner et. al. 47 have demonstrated that connected-speech systems perform better if templates used to train them are derived from connected-speech. That is asking users to input words as strings of digits and then cutting these strings apart manually or automatically to find the individual words used to train. Studies presented in this chapter do not take advantage of this technique.

Call it  $Link(f-1)$ . For consistency with algorithms to be presented in this chapter set  $Link(1) = -1$ . Then the path that best matches the input frames from frame  $Link(f)$  to frame  $f$  corresponds to template  $Tp(f)$  and has a score of  $Top(f)$ . Further the best way to match the input up to frame  $f$  with a concatenation of templates must include this path.

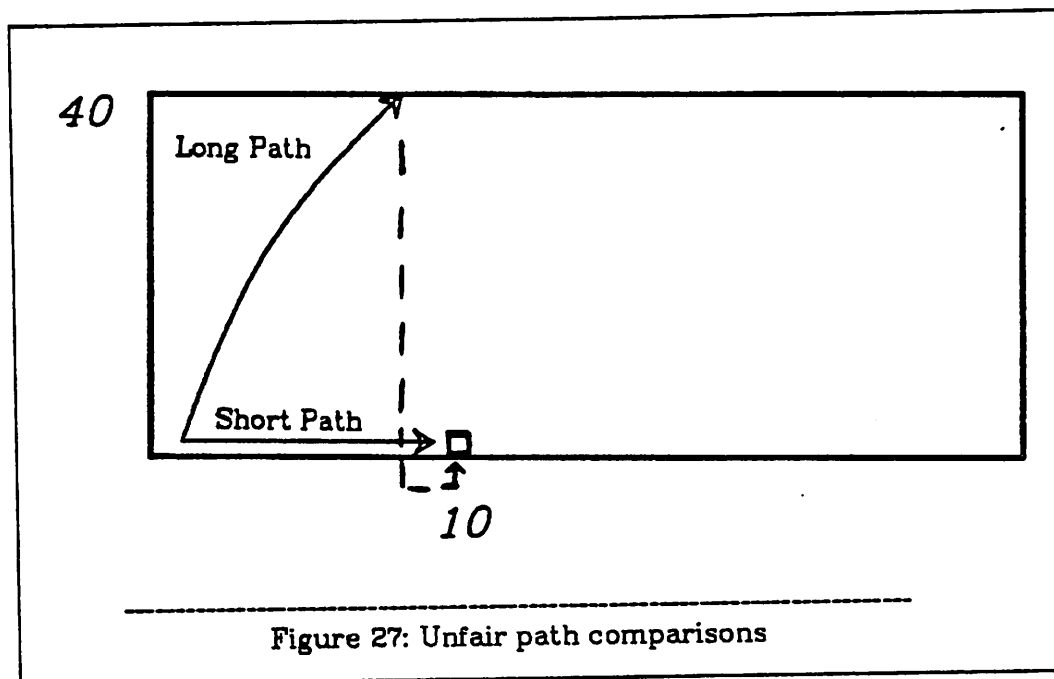
Once these functions are computed for all  $f$  the following algorithm can be used to find the words (in reverse order) that match a connected input string with  $L_{in}$  frames.

- **Initialize:** set  $f = L_{in}$
- **Iterate:** while  $f > 0$ , the recognized word is  $Tp(f)$ . Set  $f = Link(f)$ .

Though the above is a reasonable algorithm (it is the one published by Bridle<sup>46</sup>) it has a few flaws.

The above algorithm does not allow for short silences between words. This neglects the possibility that short pauses can occur between words even when accepting normally connected strings as input. Forcing the recognition algorithms to match these pauses would degrade performance of the algorithm.

The above algorithm is not completely fair when comparing alternative paths in its dynamic programming recursion. That is when comparing two paths, one longer than the other, the shorter one will have the advantage as there is no form of path normalization.



Consider the comparisons made at the start of the tenth column for a template with forty frames (shown in the figure). At that point the algorithm examines the sum of ten spectral distances, the path comparing the first frame of the template with each of the first ten frames of the unknown (the short path in the figure). This is compared with the sum of forty or more spectral distances (the long path). These are the sum of the spectral distances along the path that compares the entire template with the first nine frames of the input and is fed into  $init(10)$ . Even if the latter path has an average spectral error that is much less than the first path, it will rarely be chosen since it has many more distances summed up in it than the shorter path.

The first problem is easy to fix. Instead of defining  $init(f)$  as the best topscore of the previous frame  $Top(f-1)$ , extend it to be the best topscore of the previous few columns with penalties associated for skipping frames. This is done in the equations below.

$$init(f) = \underset{1 \leq g \leq f}{MIN} \left\{ Top(g) + penalty(g, f - 1) \right\} \quad (3)$$

$$penalty(a, b) = \sum_{f=a}^b skipcost(energy(frame f)) \quad (4)$$

$$skipcost(e) = \begin{cases} \text{Small Number} & \text{if } e \leq \text{low threshold} \\ \text{Moderate Number} & \text{if } e \leq \text{speech threshold} \\ \text{Large Number} & \text{if } e > \text{speech threshold} \end{cases} \quad (5)$$

*Small number* (15) used in the above equations corresponds to the score of a very good match between an unknown and a template. *Moderate number* is a fair match but above the isolated word rejection threshold. *Large number* is the score of a very poor match. These numbers are used to encourage skipping frames during low energy segments. They also allow skipping at high energy and no skipping at low energy, both at a higher cost.

Alternatively this problem could be solved by introducing new templates made from silence. These templates should match the silence to be eliminated.

The second problem discussed above is harder to solve. The dynamic time warp algorithm compares alternative paths fairly if the normalized spectral error per frame in the path is used to compare the paths. Thus in the above algorithm, where the total accumulated errors are used to start the columns, comparisons are unfair.

To remedy this, the topscores are normalized before they are fed into the time warp matrix again (as the bottom scores  $init(f)$ ) using the following technique:

$$init(f) = f \cdot MIN_{tp} \left\{ \frac{Top(f) - init(Link(f)) + sum(Link(f))}{\max(f, sum\_length(f))} \right\}, \quad (6)$$

$$sum\_length(f) = sum\_length(Link(f)) + length(Tp(f)), \quad (7)$$

$$sum\_length(-1) = 0. \quad (8)$$

In the above equations  $sum\_length(f_{frame})$  is the sum of the lengths of the templates that have matched the input from the first frame to the current frame. The accumulated spectral distances are normalized by being divided by the maximum of the number of frames matched in the input and the sum length of the templates. This produces a score that can be considered the error per input frame. This is multiplied by the number of input frames to allow fair comparisons between paths because they all are in terms of spectral distance per input frame. This is basically the same normalization technique as is used in isolated speech.

### 3.3. Performance

When recording the alpha-digits speech data base used in experiments described in Chapter 2 a series of phrases with connected digits spoken by the same people were also recorded. The phrases range in length from two to nine digits per string and are spoken naturally. They are shown in the table below.



<b>Strings used in the connected-word experiment</b>				
146638	15152	15210207	1547	170
187	19	20	254	255
2702145	29	31	3446570	3464
359345	38	4100	4157	43716
439	45	452	47625	486
499849	50245	51	5382151	55543
58861	617	62738	671	686
6864	7337	7357	74	755210
7564	78406	786	787700	7895518
81073	8353	88	91	9559206
98	9844	992642		

Several variations of the connected speech algorithms described above were evaluated using this data base and the results are shown below.

<b>Connected Speech - normalization (Strings wrong of 54)</b>		
<b>The Speaker</b>	<b>Normalized</b>	<b>No normalization</b>
BB	5	14
CW	3	9
GW	8	12
HM	8	21
KH	19	33
MC	5	10
NF	12	24
<b>Total</b>	<b>60</b>	<b>123</b>

For the above studies we conclude that normalization is extremely important for this connected-speech recognition system, as was anticipated by the unfair path argument. Without normalization there were many input sentences recognized with word deletion errors. Often words were deleted because one template would often match two input words. This would happen because the total number of distance measures summed along the incorrect path (one template) was many less than the number summed along the correct path (two templates). Normalizing corrected this problem.

<b>Connected Speech - silence elimination (Strings wrong of 54)</b>			
<b>The Speaker</b>	<b>No elimination</b>	<b>Penalty</b>	<b>Silence-tps</b>
BB	5	5	5
CW	3	3	3
GW	8	8	8
HM	8	8	6
KH	19	19	20
MC	5	5	5
NF	12	12	9
<b>Total</b>	<b>60</b>	<b>60</b>	<b>54</b>

Adding silence templates did a better job with in-phrase pauses than the algorithmic change that allowed skipping of frames. The latter approach tended to either skip too many frames and cause some errors or to not skip silences at all when the penalty parameters were set too high. Using silence templates is similar to using a penalty technique, but the penalties here are the euclidean distances between speech input and silence. This tends to be a better silence detector than the energy level technique used with penalties.

<b>Connected Speech - downsampling (Strings wrong of 54)</b>					
<b>The Speaker</b>	<b>Threshold / effective sampling period (ms.)</b>				
	<b>0/10.0</b>	<b>2/12.0</b>	<b>4/15.4</b>	<b>6/19.1</b>	<b>8/22.4</b>
BB	5	5	6	8	9
CW	3	3	4	5	5
GW	8	7	7	9	7
HM	8	10	10	7	12
KH	19	18	21	17	16
MC	5	5	8	5	6
NF	12	11	11	11	15
<b>Total</b>	<b>60</b>	<b>59</b>	<b>67</b>	<b>62</b>	<b>70</b>

Note in the above table that the positive effects of downsampling are not as visible with connected speech as they are with isolated speech. In fact performance is degraded substantially with some of the downsampling rates (15.4ms. and 22.4ms.) while remaining roughly constant for other rates (10.0, 12.0 and 19.1ms.). One reason for this decrease of the effectiveness of downsampling is that endpoint errors are not as much of an issue in the downsam-

pling algorithm's favor in connected speech. Penalties or silence templates exist in the connected speech recognition algorithm to correct these problems. However a data rate of close to a 20ms. effective sampling rate must be maintained in order to maintain the thousand word vocabulary.

There are a few possibilities to solve this problem. One is to eliminate downsampling for connected speech which will give acceptable performance, but only allow the system to compare the input with 250 template words in real time. Of course the full 1000 template comparison capability could be maintained with a possible decrease in performance. This decrease may be offset by other uses for the extra templates that can be compared. For instance these templates may be used to add syntactic analysis to the system as will be discussed in the next section. The other alternative is to re-examine the connected algorithm with downsampling to see why the system is so sensitive to variations in the downsampling parameter. This work is currently underway.

#### **3.4. Syntactic aids**

The connected speech algorithm has been modified to accept only syntactically correct connected-speech phrases, however the syntax used must be regular. Further, for convenience, phrases spoken to the machine should be complete sentences in the grammar. An example of this type of input would be a system designed to accept a fixed number of digits; for instance telephone numbers or credit card numbers. The syntactic information available to the program is the number of digits in the input string.

To modify the connected-speech algorithm to accept only grammatically correct strings the following changes must be made to that algorithm:

- **Replicate** each template for each state in the grammar it can appear in. In the case of a system that accepted seven digit telephone numbers that would mean making seven copies of each of the digits, one for each state in the grammar. Here the first state represents the first digit of the string, state number two represents the second, and state number seven represents the last.
- **Initialize** the start of a column as *init* ( $f$ ) as computed from only the topscores of a previous state. For instance initialize the templates representing the sixth digit as the best normalized score up to this point of the first five digits.
- **Choose** as the winning path the least cost path that terminates in a final state. For instance in the telephone number example, only accept paths that finish up at state seven (a phrase with seven digits).

Using syntactic constraints to aid recognition of the test data base obtained the following results.

<b>Syntactic aid to Connected Speech Recognition (Strings wrong of 54)</b>		
<b>The Speaker</b>	<b>Normalized Syntactic Aid</b>	<b>Normalized No syntax</b>
BB	2	5
CW	2	3
GW	5	8
HM	8	8
KH	13	19
MC	3	5
NF	7	12
<b>Total</b>	<b>38</b>	<b>60</b>

There is a definite performance advantage for the system with syntax, though there is a large computational cost. However a system with a moderate sized vocabulary and syntactic structure can fit in the 1000 templates provided by the IC based speech recognition system. Certainly a sys-

tem with a small syntax, such as a digit entry system, can be used with room to spare for other templates and states as well.

### 3.5. Hardware Support

Since this connected-speech system is very similar to the isolated-word system, very little had to be done to make them compatible. The major additions were a counter and memory to compute the value of  $Link(f)$ . A facility was also added to allow initialization of the bottom score of a column  $init(f)$ . In isolated speech the bottom score is always initialized to infinity except in the first column where it is zero.

A single input line on the comparator hardware defines isolated or connected mode to see whether  $Link(f)$  should be computed and whether the bottom score is input.

A column of memory seven bits wide is used to calculate  $link(f)$ . This translates to an extra  $\frac{7}{T_f} \sum_{n=1}^N T_{tp_n}$  or 175K bits with downsampling for a thousand word vocabulary.

The system microprocessor's role in recognition is increased in this system as it now must calculate the bottom scores for succeeding frames (equations 6,7 and 8). This is a burden on the microprocessor as the normalization requires a divide for each word in the vocabulary each input frame. Further the pipeline is stalled as the microprocessor cannot compute  $init(f)$  until the comparator IC is done with frame  $f-1$  and then it must sit idle until the microprocessor is done computing  $init(f)$ . As it is very desirable to overlap the microprocessors computations with those of the comparator, a scheme is used in which  $init(f+1)$  is approximated by  $init(f)+skipcost(f)$ . With this the comparator need not wait for the

microprocessor to complete its computations. We have simulated the performance of this approximated system and found it to be very similar to the performance of the exact system.

<b>Init Approximation (Strings wrong of 54)</b>		
<b>The Speaker</b>	<b>Approximate Init</b>	<b>Exact Init</b>
BB	4	5
CW	2	3
GW	8	8
HM	8	8
KH	20	19
MC	7	5
NF	11	12
<b>Total</b>	<b>60</b>	<b>60</b>

## CHAPTER 4

### Speech Recognition Systems with Difficult Vocabularies

#### 4.1. Introduction

The speech recognition system performs very well for tasks where the vocabulary used is comprised of distinct words. Because of this, when designing a vocabulary for the speech recognizer, very similar words should be avoided whenever possible. This is usually only a minor restriction. For instance the recognition system performed very well with the hundred word KIC vocabulary as was shown in chapter 2. Little was changed in that vocabulary from the original typed KIC input language. However there are important vocabularies that force the system to consistently make fine phonetic distinctions, and in these cases performance is degraded. Consider the comparison of KIC and Alpha-Digit vocabularies at the start of chapter 2 reproduced below.

Percentage of Errors					
KIC			Alpha-Digits		
men	women	both	men	women	both
0.9%	0.4%	0.6%	4.1%	3.9%	4.0%

One reason that the speech recognition system has problems with similar words such as *b, d, e, p, t* in the alphabet is that it weights, after downsampling, all parts of the word equally in its decision process. Further it weights all frequency bands equally in its distance measure. There are occasions, such as when choosing between *b* and *p*, where certain parameters are more important than others in deciding between words. In the previous example

low frequency bands during the start of the words are relatively more important to the comparison than bands in other parts of the word.

#### 4.2. Previous Work in the Area

Rabiner et. al.,<sup>48</sup> Bradshaw et. al.,<sup>49</sup> and more recently Cole et. al.<sup>5</sup> have published studies where they attempted to improve the performance of speech recognition systems that make fine phonetic distinctions. Rabiner used a two pass algorithm. He first found pairs of similar words in the vocabulary. Using training samples of those word pairs he found frames where the words differed the most and weighted those frames heavily in a second pass of time-warped comparisons. Cole's program FEATURE marked important times in words such as vowel onset and offset. It then used statistics gathered about certain features around these times, such as energy in a low frequency band, to build functions that best discriminate between the two words.

Rabiner's technique is attractive because it is easily automated and simple to implement. However, as every pair of similar words requires a second pass time weighting function, there could be an excessive storage requirements. Cole's technique is attractive because it uses higher level acoustic events to trigger the gathering of statistics. For instance, instead of Rabiner's algorithm implying that "the third, fourth and fifth frames are important", Cole's technique may imply "Low frequency in the 30 milliseconds before vowel onset is important". However it is likely that the time warping mechanism in Rabiner's algorithm will find the same important acoustic events as Cole's statistical search. In the above example the third, fourth and fifth frames would coincide with the thirty milliseconds before vowel onset.



Rabiner's technique, however, should be extended to weight parts of the spectrum as well as time. This is easily done in band pass filter front ends by weighting each band differently,

$$d(F^a, F^b) = \sum_{i=1}^p w_i \left[ F^a_i - F^b_i \right]^2 \quad (1)$$

where  $w_i$  are the band weights.

### 4.3. Data Base Used for the Experiments

The following studies have been run in order to examine ways to improve the performance of dynamic time warped systems with difficult vocabularies. The data base used with this study was one hundred repetitions of the E-set (3,b,c,d,e,g,p,t,v,z) vocabulary by one speaker. The speaker deliberately varied the way he pronounced the letters to make this a more difficult test. Eighty of the repetitions were used to train the system and twenty were used to evaluate the performance of various alternative approaches.

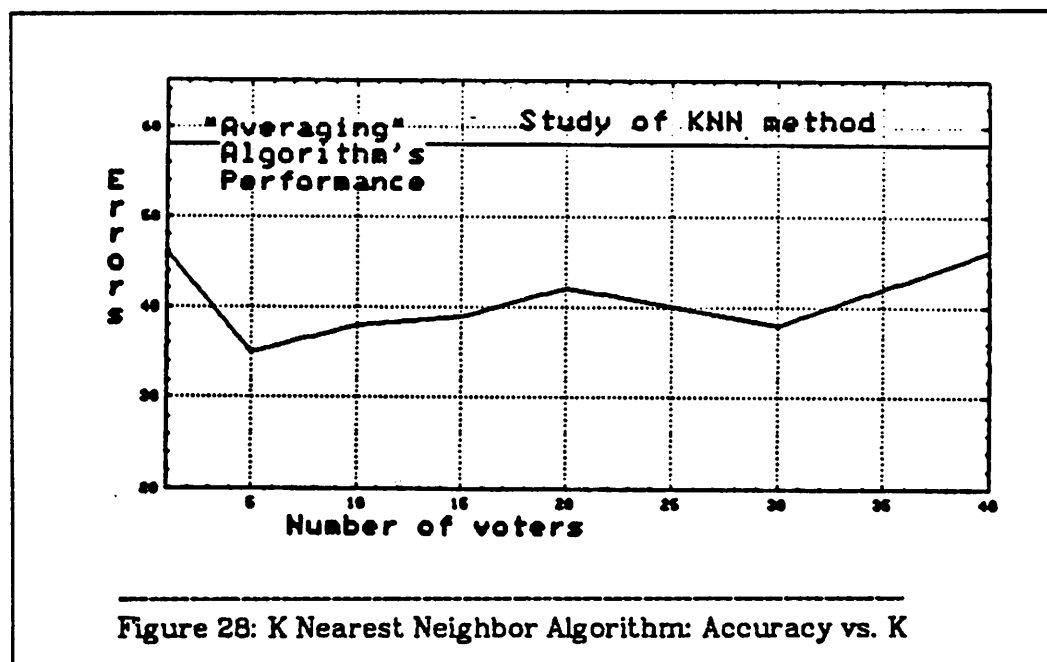
### 4.4. Performance of the Standard System

The standard speech recognition system presented in the first chapter was used as a control for this experiment. The "clustering with averaging" training procedure was presented with the eighty training samples each of the ten words. It came up with forty five templates for the ten words. The reason that many templates were generated instead of the usual one or two per word was that the speaker varied his speech greatly during the training and test sets. System performance was somewhat below the E-set study shown in chapter 2, primarily because of this wide variability.

The best performance of any system tested was achieved with the standard recognizer by using all eighty training words as templates and employing a k-nearest-neighbor (KNN) selection scheme. This used a total of 800

templates for the ten word vocabulary as opposed to the 45 used in the averaging template run. The top K scoring templates voted to find the recognized word. KNN was tried with several different values of K from KNN-1 (use only the best scoring template) to KNN-40 where the top forty templates voted. Note that the KNN schemes here performed better than those surveyed in chapter 2 and considerably better than the "averaging algorithm". This is due to the better sampling of templates (eighty per word as compared to five before). Also note that for the KNN schemes a low value for K relative to the number of templates per word was superior as before.

Template selection and decision criteria							
Errors (of 200) using standard system							
E-set, 20 test repetitions of 10 words							
average	KNN-1	KNN-5	KNN-10	KNN-15	KNN-20	KNN-30	KNN-40
58	46	35	38	39	42	38	46



#### 4.5. Performance of Feature Weighted Systems

We desired to obtain better system performance by weighting the features that are important to a given template more than other features. However, unlike the method suggested by Rabiner, we do not want a different distance measure for every difficult word pair, and we wish to consider filter weights as well as time weights. Equation (1) was chosen to implement the spectral distance measure and was implemented in several ways in the following experimental systems.

VARIANCE set all filter values to have uniform variance;  $\hat{f}_i = \frac{f_i}{\sigma_i}$  and  $w_i = 1$ . Recall that  $\sigma_i$  is the standard deviation of the  $i^{\text{th}}$  filter value across all frames of all eighty copies of each word. This is the feature variance normalization discussed in chapter 2.

For the QUALITY system the same variance normalization is used, but now

$$w_i = \frac{1 - \sigma_{mi}}{\sigma_{mi}} \quad (2)$$

$$\text{if } w_i < 0 \quad w_i = 0 \quad \text{if } w_i > 10 \quad w_i = 10$$

Here  $\sigma_{mi}$  is the standard deviation of the filter value when the template chosen is warped against eighty repetitions of the same word. This is the variance of the feature in a particular frame. Thus  $w_i$  for the second frame of a template will likely have a different value than  $w_i$  for the third frame. QUALITY uses as a measure of importance the variability of a feature across different repetitions of a word. Since features have been normalized to a standard deviation of one across all frames of all templates, if  $\sigma_m > 1$  then that feature has more variability among repetitions of a word than outside that word. In this case it would have a quality of zero. For a given frame the

$w_i$  are normalized to add to 1 thus the total distance measure is given as

$$d(F^a, F^b) = \sum_{i=1}^2 \frac{\frac{1-\sigma_{mi}}{\sigma_{mi}} (F_{ai} - F_{bi})^2}{\sigma_i^2 \sum_{j=1}^{12} w_j} \quad (3)$$

DISTANCE is similar to quality except that equation (2) is replaced by

$$w_i = \frac{(\bar{f}_{mi} - \bar{f}_i)^2}{\sigma_{mi}^2 + \sigma_i^2}$$

That is the squared differences in the in word and across word means divided by the sum of the variances.

Note that all three of the above methods weight filter bands but leave the time domain unweighted (the opposite of Rabiner's technique). The QUALITY and DISTANCE techniques require twice the template memory of the original system as all the  $w_i$  must be stored with the template, and there is a weight associated with each feature. The results of a test evaluating the system is shown below.

Frequency weighting technique						
Errors (of 200) using standard system						
E-set, 20 test repetitions of 10 words						
VARIANCE	QUALITY	TIME_QUALITY	DISTANCE	AVERAGE	KNN-5	
55	50	58	61	58	35	

The QUALITY technique improved the performance of the system although some of the improvement was due to the variance normalization. However the performance was not near the performance of the KNN-5 system.

TIME\_QUALITY is time weighting added to the QUALITY distance measure. The time weights were the sum of the qualities of the features in a given frame before the quality normalization,  $\sum_{j=1}^{12} w_j$  in equation (3). These time

weights were normalized for the entire word. Each frame of a template had such a time weight and it multiplied the distance measure computed with that frame. To avoid steering the dynamic time warped paths away from important parts of the word these weights are applied after the time warped path is found without time weights in the distance measure. These time weights slightly degraded the performance of the system. Although they seemed to help difficult word pairs, words that originally were not confused were confused with the time weighted system. That is because parts of the word that differed greatly were decreased in importance by the time weightings. It seems that such time weighting is a useful technique for one on one comparisons (for example just comparing *b* with *p* as used by Rabiner) but get muddled when used for comparisons against many different words.

#### 4.6. Optimization of Templates

The previous section discussed attempts to create templates that were composed of an averaged template used in conjunction with weights that described the importance of the individual features. In this section a system is described where all parts of the template, the weights as well as the features in the template itself are chosen for best possible performance. This system with one template per word achieved performance that is close to the performance of the KNN-5 system with eighty templates per word.

We desired to optimize the parameters of an entire template instead of just the frequency and time weights. In order to do so, one may consider the problem as one of optimizing parameters in a template to best solve a given problem: getting the best performance in a speech recognition task as predicted by performance in the training task. However if all those parameters were passed to an optimization routine an enormous amount of training

data and computation time would be required for meaningful results. For instance in an average template there are 25 frames each with 12 features, 12 frequency weights, and 1 time weight. That would be  $25 \cdot (12+12+1) = 625$  parameters to optimize for each template. Further to measure performance for any given set of values of those parameters would require performing a large number of speech recognition trials (eighty copies of each word compared against the new templates). Instead the following approach was taken.

Crude mappings were made between frames of each of the sample words in the training set. An example of such a mapping is that if word one were being compared with word ten, frame twelve of word one would be compared with frame fifteen of word ten. After the mappings were set the individual frames were optimized so that they best distinguished the words, just among the frames that they mapped to. Once all the frame weights were set this way, then time weights were optimized to best distinguish words. This algorithm was iterated with better and better frame to frame mappings. The algorithm is presented in more detail below.

- **Pick an initial template:** For each word  $W$  in the vocabulary a template  $T_w$  is chosen by the clustering with averaging training technique.
- **Define sets of frames:** All the training samples for  $W$  are time-warped against  $T_w$  as well as all of those for other words. This defines a mapping between the frames of  $T_w$  and the frames of all training words. Then all frames that mapped to the  $i^{\text{th}}$  frame of  $T_w$  are grouped into the set  $S_i$ . All the frames in  $S_i$  that come from training samples for  $W$  are placed in the set  $G_i$  and all the frames in  $S_i$  that don't correspond to  $W$  are placed in the set  $B_i$ . Then  $S_i = G_i \cup B_i$  and  $G_i \cap B_i = \phi$ .

- **Find the best first frame:** Find a new frame  $F_1$  that is composed of 12 artificial filter parameters and 12 frequency weights so that  $F_1$  best separates the sets  $G_1$  and  $B_1$  using the distance measure shown in equation 1. This is done with a hill climbing optimization algorithm for the 24 coefficients that minimized an error computed by the following algorithm.

- 1) Set  $error = 0$ .
- 2) For all frames  $g$  in  $G_1$ ,
  - if  $dist(g, F_1) > 80$  then
  - $error = error + K \cdot (dist(g, F_1) - 80)$ .
- 3) For all frames  $b$  in  $B_1$ ,
  - if  $dist(b, F_1) < 120$  then
  - $error = error + 120 - dist(b, F_1)$ .

The goal is to separate frames in  $B_1$  from  $G_1$  by having frames in  $G_1$  have errors of 80 or less and those in  $B_1$  have errors of 120 or more.  $K$  is chosen to increase the significance of  $G_1$  errors since there are more elements of  $B_1$  than  $G_1$ . Thus  $K = |B_1| / |G_1|$ .

- **Repeat for all other frames:** This technique is repeated for all the sets  $S_i$  creating the frames  $F_i$ .
- **Run the dynamic time warp algorithm:** Rewarp all the correct and incorrect training words to a template composed of the frames  $F_i$  and keep the warp paths.
- **Optimize the time weights:** Use the hill climbing optimization algorithm again to set the time weights for fixed warp paths, that best separate the scores of the training repetitions of  $W$  from all the rest of the words in the training data base.
- **Repeat for each word  $W$  in the vocabulary:**

This algorithm was tested on the above data base with the following results.

Optimized template test			
Errors (of 200) using standard system			
E-set, 20 test repetitions of 10 words			
Optimized	Opt without time	AVERAGE	KNN-5
39	48	58	35

The templates optimized for the words performed with improved accuracy. Time weights were applied as before, by defining the warp path using the template without time weighting and then adding up all the distances on the warp path weighted by the time weights. In this case the time weights improved the performance and the optimized templates (one per word) worked nearly as well as the eighty template per word k-nearest-neighbor technique.

The optimization algorithms require many training tokens, more than what would be practical in most user-dependent environments. This approach however can be incorporated in a larger system where only sets of similar words are processed through the optimization algorithm. The optimization technique may also be applied to speaker-independent systems where a large number of sample words are required anyway.<sup>13</sup>

The frequency and time weighting portions of the algorithm require hardware redesign, however they would not pose difficulties. The distance  $w_i(A-B)^2$  could be computed as a four bit subtract followed by an eight input ROM or PLA that computed  $w_i \cdot X^2$ . Currently the hardware is implemented with four bit subtractors and four input  $X^2$  PLA's. Template and column memories would be approximately doubled. The template memory doubles to accommodate the  $w_i$  coefficients. The column memory doubles so that both the time-weighted and not-time-weighted partial accumulated scores could be kept by the dynamic time warp circuitry. In this way it would



continue to operate in real time.

#### **4.7. A Future Approach**

I believe that the results given above represent the upper limit of performance to be achieved by simple extensions to our system. I believe that a more significant extension of the system is necessary in order to more reliably recognize difficult vocabularies such as very large vocabularies that include many similar words. One possibility for such a system would be to combine a dynamic time-warped approach with an approach similar to the FEATURE<sup>6</sup> system designed at Carnegie Mellon University as was described in the introduction. A high performance dynamic time warp based system such as ours could compare an input word with each word in a large vocabulary. Classes of words that couldn't be reliably distinguished by the time-warp system could be distinguished with the feature based system. This hybrid system would have the advantages of both systems. It would have flexibility in changing vocabularies or speakers (if it were speaker dependent). It would be very accurate for distinct vocabularies and it could differentiate similar words using significant features. It could perform most of its searching using our integrated circuits and therefore could be both quick and economical. The feature based subsystem would only need concern itself with small sets of similar words. We have begun a preliminary investigation into this area and believe it to be promising.

#### **4.8. Strengths and Weaknesses of the Current System**

The most significant conclusion to be drawn from our speech-recognition project is the power of the special-purpose integrated circuit. Circuits such as ours can be built to perform enormous amounts of computation with rela-

tively little effort. For example in our speech recognition system the distance measure circuitry computes a sixteen feature euclidean distance in less than a microsecond. That is the equivalent of sixteen million additions, multiplications and subtractions per second. Thus a relatively small integrated circuit executes at more than a 50 MIPS rate, two orders of magnitude higher than high performance and much larger microprocessors. Of course the special-purpose system's computations are narrow (some only four bits wide), and there is a high degree of pipelining and parallelism that is well suited to execute our algorithms. This is because the problem (dynamic time-warped speech recognition) was structured in a very regular way that allowed this high rate and was tolerant of low accuracy computations. The task awaiting knowledge based and statistical approaches to speech recognition is to find ways to structure those techniques so that they can also take advantage of integrated circuits. In this way they might achieve the high computation rates that they sorely need.

Another conclusion reached was that speech recognition devices are very useful. The system in our laboratory was a very helpful device when used with computer graphics programs. It seems more convenient and natural for users than a keyboard. In fact it often seemed that fewer errors are made by the speech recognizer than would be made by a user typing! We wish to continue in this area and integrate speech more fully into the computer environment.

The main weaknesses of the speech recognition system were training and endpoints. It is difficult for novice users to adequately train the system. Often the number of repetitions required is too high and a smaller number is used with occasional poor templates. This is usually cured "on the fly" by

retraining for only poorly recognized words. However more automatic ways of adapting the system to users should be considered. Speaker independent systems is an ideal way to solve this problem but one that still requires some investigation. Making systems speaker dependent is still quite reasonable for a large number of tasks.

More sophisticated endpoint detection techniques should be considered. Systems such as ours would rarely err on non-similar words if endpoint errors were not a problem.

Although our system does remarkably well distinguishing similar words (10% error in the E-set), new techniques should be considered. Hybrid pattern matching and feature based systems is one approach to this problem as described above.

#### **4.9. Conclusion**

A very powerful, flexible and potentially inexpensive speech recognition system has been described. It takes advantage of special-purpose integrated circuits achieve this high performance. With systems such as ours speech recognition should have a large range of applications within the next few years.

## References

1. Dennis H. Klatt, "Review of the ARPA Speech Understanding Project," *Journal of the Acoustic Society of America* Vol. 63 pp. 1345-1366 (Dec. 1977).
2. V. R. Lesser, R. D. Fennel, L. D. Erman, and D. R. Reddy, "Organization of Hearsay II Speech Understanding System," *IEEE Transactions on Acoustics Speech and Signal Processing ASSP-23* pp. 11-24 (Feb. 1975).
3. Victor Zue, *Private Communication*. 1982.
4. F. Jelinick, "Continuous Speech Recognition by Statistical Means," *Proceedings of the IEEE* Vol. 24 pp. 532-556 (April 1976).
5. L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition," *The Bell System Technical Journal* Vol. 62, No. 4 pp. 1075-1105 (April 1983).
6. R. A. Cole, R. M. Stern, M. S. Phillips, S. M. Brill, A. P. Pliant, and P. Specker, "Feature-Based Speaker-Independent Recognition of Isolated English Letters," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 731-734 (Spring 1983).
7. F. Jelinek, R. L. Mercer, and L. R. Bahl, *Continuous Speech Recognition: Statistical Means*, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
8. T. B. Martin, "Practical Applications of Voice Input to Machines," *Proceedings of the IEEE* 64 pp. 487-501 (April 1976).
9. George M. White and Richard B. Neely, "Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming," *IEEE Trans. Acoustics, Speech and Signal Processing* Vol. ASSP-24 pp. 183-188 (April 1976).
10. F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Trans. Acoustics, Speech and Signal Processing* Vol. ASSP-23 pp. 67-72 (Feb. 1975).
11. L. R. Rabiner and S. E. Levinson, "Isolated and Connected Word Recognition - Theory and Selected Applications," *IEEE Trans. on Communications* Vol. COM-29 pp. 621-659 (May 1981).
12. Hy Murveit, Menachem Lowy, and R. W. Brodersen, "A Thousand-Word Speech-Recognition System Using Special-Purpose MOS-LSI," *Proceedings of the USC Workshop on VLSI and Modern Signal Processing*, pp. 13-18 (November 1982).
13. L. R. Rabiner and J. G. Wilpon, "Considerations in Applying Clustering Techniques to Speaker-Independent Word Recognition," *Journal of the Acoustic Society of America* Vol. 66 pp. 663-673 (Sept. 1979).
14. Stephen L. Moshier, "Talker Independent Speech Recognition in Commercial Environments," *Fiftieth Meeting of the Acoustic Society of America*, (June 1979).
15. G. R. Doddington and T. B. Schalk, "Speech Recognition: Turning Theory to Practice," *IEEE Spectrum*, pp. 26-32 (Sept. 1981).
16. Motorola Inc., *Motorola Product Catalog*. 1983.

17. *Texas Instruments Digital Signal Processor, TMS320 Product Description*, Texas Instruments P.O. Box 1087, Richardson Texas, 75080(1983).
18. S. Pope and R. W. Brodersen, "Macrocell Design for Concurrent Signal Processing," *Caltech conference on VLSI*, pp. 395-411 (1983).
19. Menachem Lowy, *The Integration of a Speech-Recognition System*, University of California, Berkeley(June 1983). Ph.D. Thesis
20. C. F. Chan, M. E. Hoff, P. Nevard, and M. Lee, "Architecture and Applications of a Commercially Available Speech Recognition Board," *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 495-498 (Spring 1983).
21. Votan Corporation and Hayward California,
22. H. Ishizuka, M. Watari, H. Sakoe, S. Chiba, T. Iwata, T. Matsuki, and Y. Kawakami, "A Microprocessor for Speech Recognition," *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 503-508 (Spring 1983).
23. D. J. Burr, Bryan Ackland, and Neil Weste, "A High Speed Array Computer for Dynamic Time Warping," *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 471-473 (Spring 1981).
24. Robert Kavaler, *Applications of Clustering to Speech Recognition*, University of California, Berkeley(June 1983). M.S. Thesis
25. Eric Davies, *Endpoint Detection of Speech for Real Time Isolated-Word Recognition*, University of California, Berkeley(June 1983). M.S. Thesis
26. David M. Mintz, *An Implementation of a Speech Recognition System*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley(March 1983).
27. *Reticon R5604 Third octave filters*, Reticon, 345 Potrero Ave., Sunnyvale CA 94086
28. John Makhoul, "Linear Prediction in Automatic Speech Recognition," pp. 183-230 in *Speech Recognition*, ed. D. Raj Reddy, Academic Press, New York (1975).
29. Paul Hurst, University of California, Berkeley(1983). Ph.D. Thesis
30. Ronald Fellman, *An MOS-LSI Adaptive Linear Prediction Filter for Speech Processing*, University of California, Berkeley(1982). Ph.D. Thesis
31. A. Richard Smith, B. Patrick Landell, and George Vensko, "Solid State Audio/Speech Processor Analysis," Report F30602-78-C-0359 FR, ITT Defense Communications Div., Nutley, N.J. (1979).
32. B. A. Dautrich, L. R. Rabiner, and T. B. Martin, "On the Use of Filter Bank Features for Isolated Word Recognition," *Proceedings of the International Conference on Acoustics Speech and Signal Processing*, pp. 1061-1064 (Spring 1983).
33. M. R. Sambur and L. R. Rabiner, "An Algorithm for Determining the Endpoints of Isolated Utterances," *The Bell System Technical Journal* Vol. 54 pp. 297-315 (Feb. 1975).
34. S. K. Das, "Some Experiments in Discrete Utterance Recognition," *Proceedings IEEE International Conference on Acoustics, Speech and*

- Signal Processing*, pp. 178-181 (Spring 1980).
35. Chiu-Kuang Chuang and Stephan W. Chan, "Speech Recognition Using Variable Frame Rate Coding," *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1033-1036 (Spring 1983).
  36. R. Pieraccini and R. Billi, "Experimental Comparison Among Data Compression Techniques in Isolated-Word Recognition," *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1025-1028 (Spring 1983).
  37. Hiroaki Sakoe and Seibi Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoustics, Speech and Signal Processing* Vol. ASSP-26 pp. 43-49 (Feb. 1978).
  38. Peter Ruetz, University of California, Berkeley (June 1983). M.S. Thesis
  39. L. Robert Morris, "A Tale of Two Architectures: TI TMS 320 SPC vs. DEC Micro/J-11," *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1200-1203 (Spring 1983).
  40. Kenneth L. Carlock and Menachem Lowy, *Speech Processor, A 19 Channel VOCODER*, Electronics Research Laboratory, U.C. Berkeley (Oct. 1979). Technical Memorandum
  41. Ken Keller and Giles Billingsley, *KIC: A Graphics Editor for Integrated Circuits*, Electronics Research Laboratory, U.C. Berkeley Technical Memorandum
  42. John Ousterhout, *Caesar Users Manual*, Electronics Research Laboratory, U.C. Berkeley Technical Memorandum
  43. *The Shure SM-10 Microphone*, Shure Brothers Inc., 222 Hartley Ave., Evanston IL 60204
  44. H. Sakoe, "Two-level DP Matching - A Dynamic Programming Based Pattern Matching Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech and Signal Processing* Vol. ASSP-27 pp. 588-595 (December 1979).
  45. C. S. Myers and L. R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech and Signal Processing* Vol. ASSP-29 pp. 284-297 (April 1981).
  46. John S. Bridle, Micheal D. Brown, and Richard M Chamberlain, "An Algorithm for Connected Word Recognition," *Proc. Int. Conf. Acoustics, Speech and Signal Processing* Vol. 2 pp. 899-902 (May 1982).
  47. L. R. Rabiner, A. Bergh, and J. G. Wilpon, "An Embedded Word Training Procedure for Connected Digit Recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1621-1624 (Spring 1982).
  48. L. R. Rabiner and J. G. Wilpon, "Isolated Word Recognition Using a Two-Pass Pattern Recognition Approach," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 724-727 (Spring 1981).
  49. Gary L. Bradshaw, Ron Cole, and Zongge Li, "A Comparison of Learning Techniques in Speech Recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 554-557 (Spring 1982).

## **Appendix 1.**

### **5.1. Speech Research Facility**

A speech research facility has been developed over a period of years in our Laboratory at Berkeley. It enabled us to experiment with and build various speech recognition, synthesis and storage systems. The facility is based around a Digital Equipment Corporation (DEC) PDP-11/750 UNIX computer system and several LSI-11/02 and LSI-11/23 microprocessors.

Our philosophy has been to perform most of our speech processing tasks on the UNIX system in order to take advantage of the file system and easy programmability of that operating system. Tasks which required real-time response were performed on one of the LSI-11's acting as a satellite processor to the VAX, communicating with it through an RS-232 terminal line and a DMA based parallel link (a DR11W Unibus card attached to a DRV11b Q-Bus card).

### **5.2. Speech Input and Output**

There are two methods in the facility that convert analog signals to digital words in UNIX file system as well as that play back UNIX files as analog output. The major method uses an LPA-11 system supplied by DEC. This is a set of boards that plug into the Unibus of the VAX and has a set of analog inputs and outputs. They allow twelve bit digitization or playback of very long segments of speech (disk space is the only limit) via the UNIX commands Panin(audio) and Panout(audio). Sampling rates can be varied, however the LPA-11 cannot be programmed by the user to execute special tasks.

Another alternative for voice digitization and playback is the Speechlab system. This system uses an LSI-11 in addition to the UNIX system. One part of this system is a program that runs on the LSI-11 and that controls analog-to-digital and digital-to-analog converters on the LSI-11's Q-bus. It also uses a timer device and parallel digital ports on the Q-bus as well as a communication link to the UNIX system. The Speechlab system can digitize data and transmit it to UNIX, or playback data read from a UNIX file. It can plot or edit the data as well. Due to hardware limitations<sup>†</sup> the size of the conversions is limited to the memory size of the LSI-11, about 128K samples. The advantage of the LSI-11 based system however is that it is programmable (using the C programming language) and sophisticated operations can be performed during data acquisition.

There are variable frequency anti-alias and reconstructions filters that can be used with the speech sampling and playback devices. In addition high quality microphones, speakers, headphones, tape recorders, preamplifiers and amplifiers are all connected to an analog patch panel so that the different devices can be connected to each other and to the computer audio ports in a variety of ways.

### **5.3. UNIX Based Speech Analysis and Graphics**

Speech files are stored as a series of 16 bit "short-words" on the VAX. There are several programs that read these files and analyze the data. These include the fast fourier transform (FFT) program, the linear predictive analysis (LPC) program, the digital filter program (FILT) and a variety of other experimental computer programs. These latter programs perform

---

<sup>†</sup>The DRV11b DMA card on the LSI-11 that interfaces to the UNIX DR11W card does not allow overlapping DMA transfers with other LSI-11 bus operations such reading the analog to digital converter data port.



pitch tracking, various speech coding schemes and so on. In general a researcher can easily write a computer program that takes advantage of all the other facilities already developed for the lab and make his new program available to others.

There are also a variety of terminal independent graphics routines that have been written. These routines determine the type of terminal being used by examining UNIX environment variables and then call the appropriate driver routines for that type of graphics terminal. Terminals currently supported are Hewlett Packard 2648, Digital Equipment Corp. VT125, and Xerox 1700 as well as standard character oriented terminals with addressable cursors such as the ADM 3a or Zenith Z19. New terminals can be added to this package by including low level graphics routines pertinent to the new terminal to a system library. Some general purpose programs use these routines. They include the FFT program to compute fourier transforms, HPLT the general purpose plotter, HIST a histogram plotter, and SCATTER the scatter plot program. Users can also include the graphics library in their own programs. Hardcopy is obtained through a HP 2631g printer which is attached to one of our HP2648a graphics terminals. Plots can be submitted to the printer as to a lineprinter device. Thus one can work on a plot at home using crude graphics with a standard character oriented terminal and then submit it for higher quality graphics.

#### **5.4. Real Time Processing**

The LSI-11's which act as satellite processors to the VAX 11/750 can do much more than simply digitize and play back data. They can be programmed in the C Programming language to perform a variety of tasks. Programs are written on the UNIX system and compiled there. They are then

transmitted to the LSI-11 where they are executed. During execution these programs can request some services from the UNIX operating system. The LSI-11's have been used for testing of several devices built in the laboratory. In one instance the LSI-11 tested a band pass filter bank. It output analog signals to the filter bank through the LSI-11's digital-to-analog device and read in the filter's analyzed coefficient with its digital ports. In another case it supplied pitch and spectral parameters out to an experimental linear predictive coding speech synthesizer when those parameters were requested by that hardware. Most of these applications are straightforward applications of the Speechlab system, in some cases with small modifications. Every breadboarded hardware project built in the laboratory over the last several years has been serviced by the LSI-11.

The most sophisticated use of the LSI-11 was in the speech recognition project. There it controlled a variety of devices necessary for recognizing speech while maintaining a communications link with UNIX. This is discussed in more detail in Chapter one of this thesis.