To the Graduate Council:

I am submitting herewith a thesis written by Stephen M. Killough entitled "Simplified Speech Recognition Techniques for Microcomputers." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Engineering, with a major in Electrical Engineering.

Robert W. Rochelle
Robert W. Rochelle, Major Professor

We have read this thesis and recommend its acceptance:

Donald W. Bouldin

Ernest L. Hall

Accepted for the Council:

Carmichael
The Graduate School

SIMPLIFIED SPEECH RECOGNITION TECHNIQUES FOR MICROCOMPUTERS

A Thesis

Presented for the

Master of Engineering

Degree

The University of Tennessee, Knoxville

Stephen M. Killough

December 1983

ACKNOWLEDGMENTS

The author wishes to thank the members of his committee, Drs. E. L. Hall, D. W. Bouldin, and R. W. Rochelle. Special thanks are due Mr. Donald Lokerson for his speech analysis concepts which were crucial to the completion of this thesis. Extra thanks are due Dr. Rochelle for introducing the author to Mr. Lokerson, providing guidance for the thesis, and for being a very nice person.

ABSTRACT

Speech recognition systems are becoming popular devices for improving communication between man and machines. The task of speech recognition is very difficult, resulting in poor accuracy, slow response time, limited vocabulary, and expensive hardware. The intent of this study was to produce a simplified method of speech recognition which would have a moderate level of performance yet would be inexpensive to implement. In particular, a goal was set to have the algorithm run in real time on a microcomputer.

The resultant method is a software adaptation of a machine designed and patented by Donald Lokerson.[1] The program runs on an Apple II computer equipped with an Intel 8088 microprocessor and with an analog-to-digital converter. The speaker-dependent program can distinguish among ten words at a 90% success rate while requiring minimal delays for processing.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
II.	THEORETICAL ASPECTS AND PREVIOUS EFFORTS	3
	Characteristics of Speech	3
	Vowels and Formant Frequencies	3
	Consonants	4
	Linear Predictive Coding	7
	Bank of Filters	13
	Lokerson Algorithm	17
	Quantizer	17
	Normalization of Formant Frequencies	21
	Common Issues	22
III.	IMPLEMENTATION	24
	Adaptation of the Lokerson Algorithm	24
	Pattern Matching	37
	Proposed Finite State Machine	39
	Template Method	41
	System Summary	43
IV.	RESULTS	49
	Accuracy	49
	Speed	51
	Sample Data	52
V.	CONCLUSIONS	63
	Possible Improvements	63
LIST OF REFERENCES		64
APPENDIXES		66
APPENDIX A.	TEMPLATE GENERATION PROGRAM	67
APPENDIX B.	TEMPLATE MATCHING PROGRAM	71
VITA		74

LIST OF FIGURES

FIGURE		PAGE
1.	Spectrum of the vowel "e"	5
2.	First and second formant frequencies for several speakers.	6
3.	Spectrum of the "s" sound	8
4.	Spectrum of the "sh" sound.	9
5.	Receiving end of an LPC communications system	10
6.	Frequency response of H(z) superimposed on the original spectrum.	12
7.	Diagram of the bank-of-filters method	14
8.	Typical output of a voiceprint machine.	15
9.	Diagram of the Lokerson machine	18
10.	Operation of the quantizer.	20
11.	System hardware	25
12.	Large dynamic range spectrum.	27
13.	Formant filter and quantizer system	28
14.	Frequency response of the lower formant filter.	29
15.	Frequency response of the upper formant filter.	30
16.	Spectrum of the "e" vowel before filtering.	31
17.	Spectrum of the "e" vowel after lower formant filtering	32
18.	Spectrum of the "e" vowel after upper formant filtering	33
19.	Modified version of the quantizer	35
20.	Hypothetical input signal	38
21.	Finite state machine transition example	40

FIGURE PAGE

22. Formant frequency versus time for the word "sea". 42

23. Data from Figure 22 superimposed on another utterance of the word "sea" 44

24. System block diagram. 45

25. Formant plot of the word "zero" 53

26. Formant plot of the word "one". 54

27. Formant plot of the word "two". 55

28. Formant plot of the word "three". 56

29. Formant plot of the word "four" 57

30. Formant plot of the word "five" 58

31. Formant plot of the word "six". 59

32. Formant plot of the word "seven". 60

33. Formant plot of the word "eight". 61

34. Formant plot of the word "nine" 62

CHAPTER I

INTRODUCTION

Speech recognition machines have recently become a reality. Since speech recognition is a very complicated task, the presently available machines are relatively complex and expensive. Present manufacturers are reducing the cost of these machines by developing custom integrated circuits and more cost-effective computers. The approach taken in this thesis was to simplify the speech recognition method rather than implement the method with more powerful hardware. From the onset a goal was established that the algorithm developed here should run on a standard microprocessor in real time.

The computer system available for this study was an Apple II equipped with a home-made analog-to-digital converter. It was soon discovered that the computer's 6502 microprocessor was not powerful enough for the task, so an Intel 8088 microprocessor was later added to the system. Several methods were tried and several false starts occurred, but in the end a method was developed which could generate a speech template in real time.

Three main techniques were considered before deciding on the final algorithm. The first method was the classical Linear Predictive Coding (LPC) technique, which is used by companies such as Texas Instruments[2] and Bell Laboratories. Although this is an elegant and accurate technique, the computation required is too much to be considered for

real time operation. The second method was the bank-of-filters approach used by Interstate Electronics and others. This approach uses an array of bandpass filters that can be implemented with analog or digital filters. As with the LPC technique, the computation time required for the bank-of-filters method was prohibitive. The third technique, which was the basis for this thesis, was based on a machine developed and patented by Donald Lokerson.[1] This device took the derivative of the input signal and measured the resulting amplitude, which resembled the output of a frequency modulation slope detector. This signal provided an indication of the formant frequencies and could thus be used to identify the vowels and words being spoken.

Chapter II discusses the speech recognition problem in general and mentions previous industry efforts in the speech recognition field. Chapter III details the specifics of the final algorithm used in this study and provides insight into how the program operates. The results of the program are described in Chapter IV, and Chapter V discusses possible improvements and concludes the thesis.

CHAPTER II

THEORETICAL ASPECTS AND PREVIOUS EFFORTS

Characteristics of Speech

Before describing how the various recognition techniques operate, some of the basic characteristics of speech should be discussed. For a complete discussion of how the vocal tract produces these characteristics, the reader should refer to Flanagan[3] chapters 2 and 3 or Rabiner and Schafer[4] chapter 3. Only the resultant characteristics as seen by the computer will be discussed here.

The two main categories of speech sounds are vowels and consonants. Of these two, the vowels are easier to characterize and analyze. This is because vowels consist of relatively long periodic signals which can be described well with Fourier transforms. On the other hand, consonants, more notably the "T," "P" and "K" sounds, are of relatively short duration. This makes the usual signal analysis techniques, such as Fourier transforms, more difficult since the short time windows degrade the frequency plots. For this reason most speech recognition systems can resolve different vowels much better than consonants. For example, a machine would have difficulty resolving the words "bog" and "dog" since the difference in their consonant sounds is very subtle.

Vowels and Formant Frequencies

Vowel sounds have consistent properties that can be easily viewed on a spectrum analyzer. Several distinctive features can be seen in

Figure 1, which is a Fast Fourier Transform (FFT) plot of the author pronouncing the vowel "e." The first feature is the discrete spectral lines caused by the periodic nature of the input signal. Since the periodic nature is caused by the vibration of the vocal cords, these sounds are also referred to as "voiced" speech. The spacing of the spectral lines represents the fundamental frequency, which will vary greatly depending on whether the speaker is a child, woman, or man. The next noteworthy characteristic is the major concentration of energy at the points F1 and F2. The frequency locations of these points are called the formant frequencies. These frequencies vary depending on which vowel is being spoken and, unlike the fundamental frequency, are only moderately dependent on whether the speaker is a man, woman, or child. Figure 2, reproduced from Rabiner and Schafer,[4] shows a diagram of the F1 and F2 formant frequencies for several speakers and several vowels. These F1 and F2 frequencies are either directly or indirectly the basis for most of the speech recognition machines available today. With carefully collected vowel sounds on spectrum analyzers with large dynamic ranges, a third formant frequency can be found in the 1500-Hz to 3000-Hz range. These third formants are of very low level suggesting that they are more a byproduct of speech rather than essential pieces of information.

Consonants

In general consonants do not provide properties as consistent as vowels. Two exceptions to this are the "s" and "sh" sounds because they

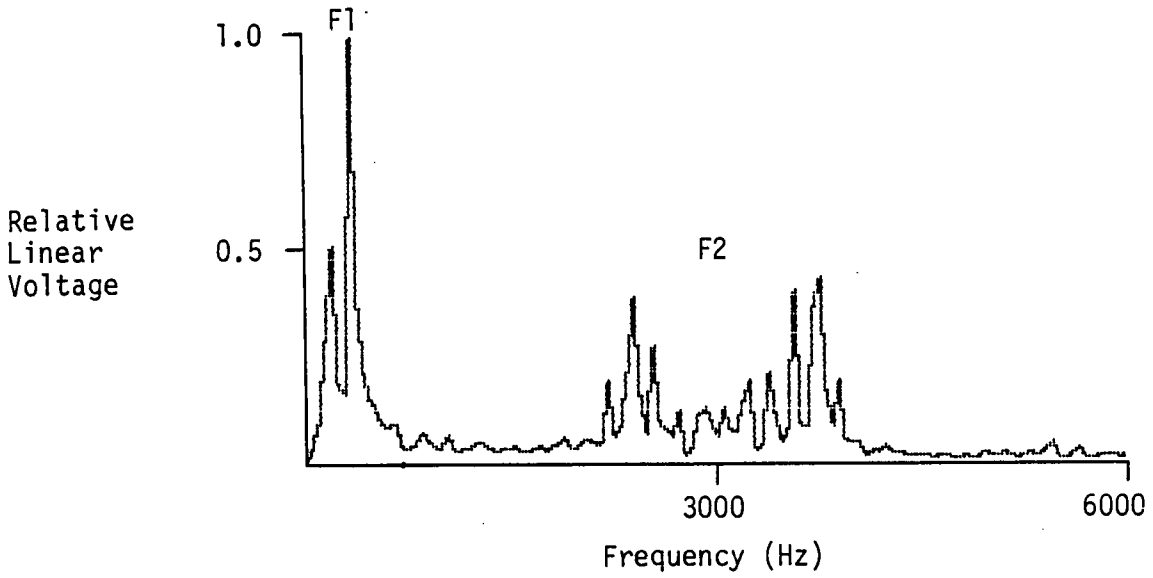


FIGURE 1. Spectrum of the vowel "e."

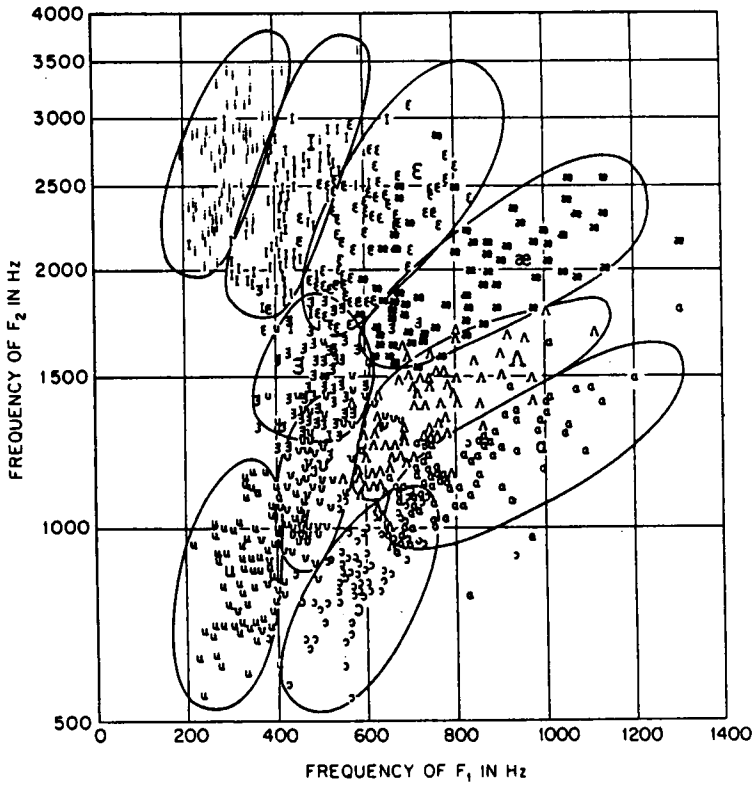


FIGURE 2. First and second formant frequencies for several speakers.

are of longer duration like the vowels. Figures 3 and 4 show the spectrum plots of the "s" and "sh" sounds respectively. From these figures major concentrations of energy can be seen around 4500-Hz for the "s" sound and 1000 to 4000-Hz for the "sh" sound. In contrast to the vowels, the "s" and "sh" sounds are not periodic but are characterized as random noise signals. Since the vocal cords do not take part in producing these sounds, these sounds are referred to as "unvoiced." Fortunately the formant locating technique can still be used to locate the energy concentrations of unvoiced as well as voiced sounds. Other consonants, such as the "p," are characterized by sudden amplitude changes rather than by certain frequency characteristics. Since the technique developed in this thesis does not utilize amplitude information, the other consonants will not be discussed. It is hoped that for limited vocabulary use, the different vowels will be sufficient to uniquely specify each of the words in the vocabulary.

Linear Predictive Coding

The first recognition technique investigated for this thesis, Linear Predictive Coding (LPC), is a technique originally used to reduce the bit rate required to transmit speech digitally. The technique has become refined enough such that ten-to-one reductions in bit rate are possible. Figure 5 shows a diagram of the system on the receiving end. H(z) is a transfer function which takes either of the two signal sources and produces a signal which resembles the original speech signal. Since the overall spectral responses of the two signal generators

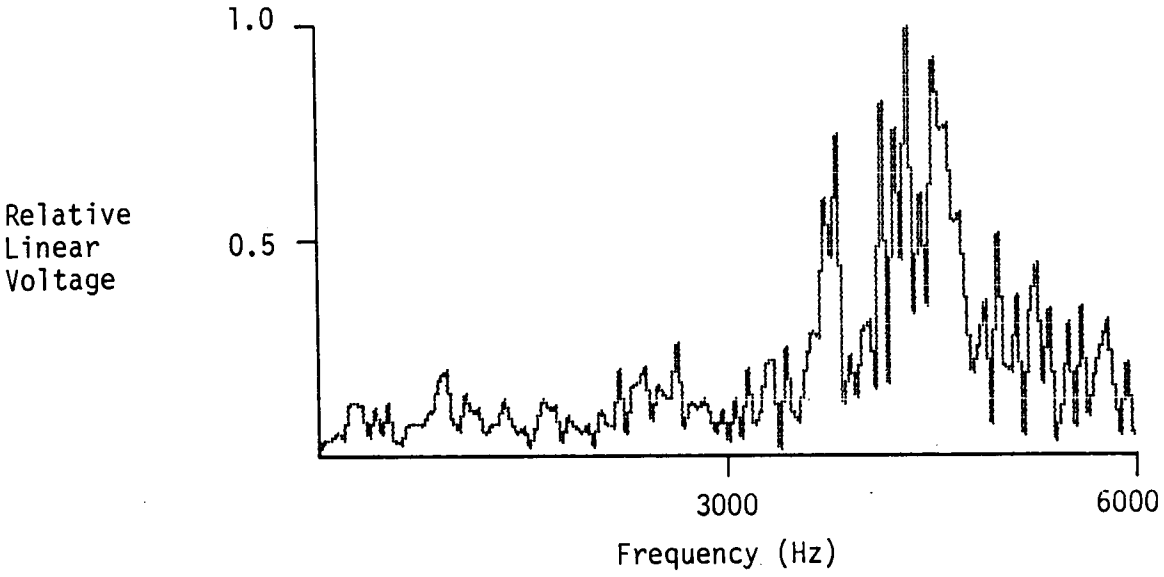


FIGURE 3. Spectrum of the "s" sound.

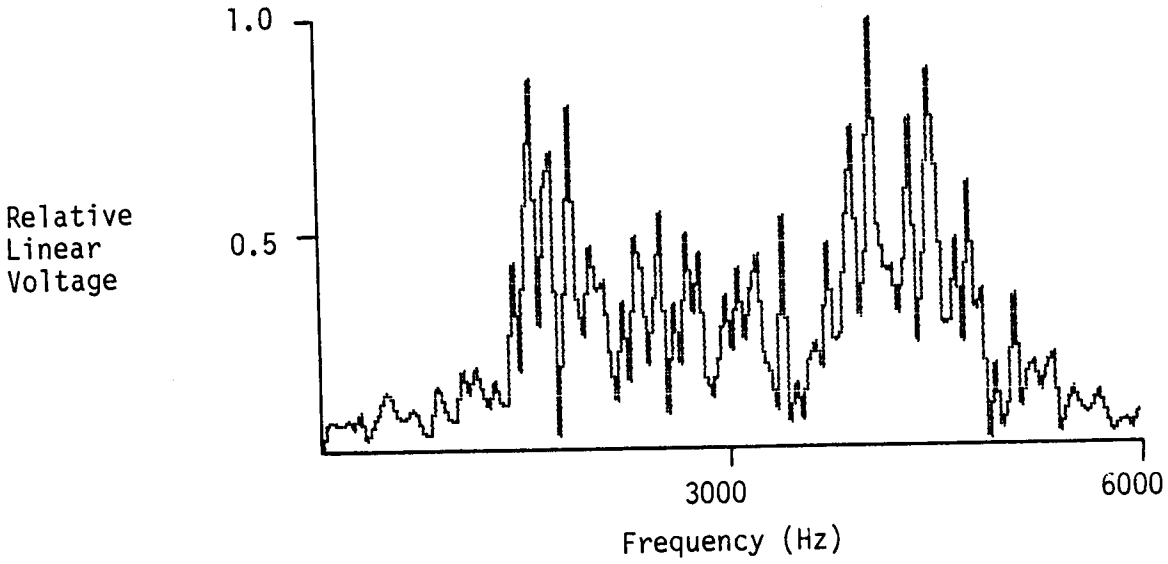


FIGURE 4. Spectrum of the "sh" sound.

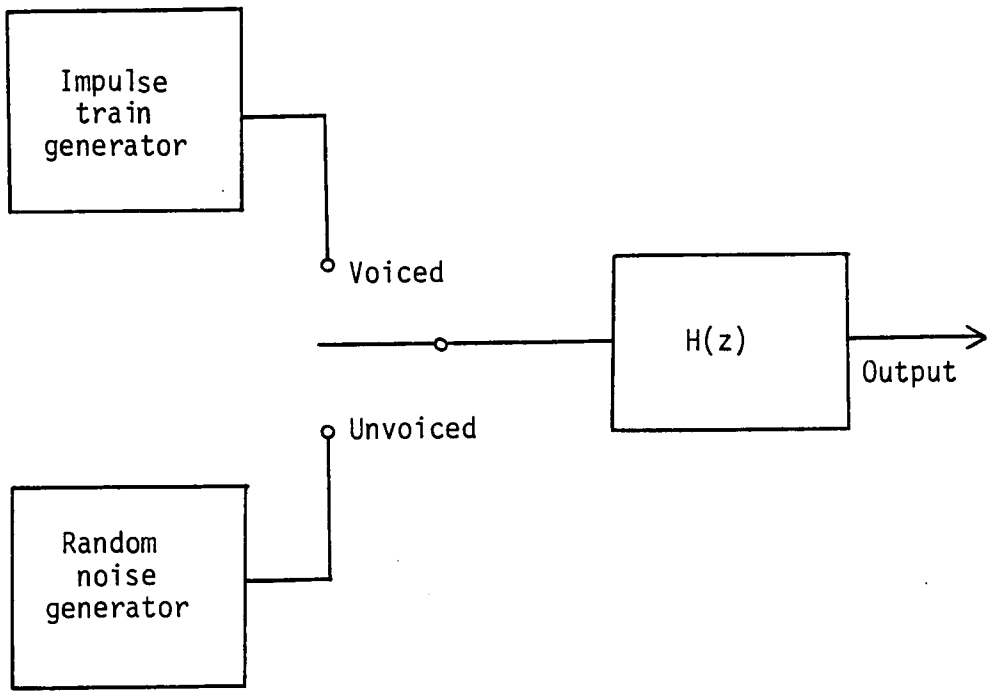


FIGURE 5. Receiving end of an LPC communications system.

are flat, the overall frequency characteristics for the speech are contained in the frequency response of H(z). Thus H(z) is a filter whose frequency response characteristics, such as peaks and resonances, correspond to the formant frequencies of the original speech. Figure 6, reproduced from Makhoul,[5] shows the frequency response of H(z) superimposed on the FFT of the original signal. When excited with the periodic pulse train, the output of the H(z) filter will also show the periodic nature, or discrete spectral lines, that the original signal shows.

Developing the H(z) transform from the speech signal is a very involved procedure (see Rabiner and Schafer,[4] chapter 8).and so will not be detailed here. One critical aspect of the procedure is that it produces H(z) in the form

$$H(z) = \frac{G}{1 - \alpha_1 z^{-1} - \alpha_2 z^{-2} \ldots - \alpha_p z^{-P}}$$

where P is the order of the predictor. In other words, H(z) is produced in coefficient form as opposed to partial fraction expansion or cascade form. Thus given H(z), to find the resonant points, such as those depicted in Figure 6, one must compute the pole frequencies of H(z). This entails using a numerical technique to find the roots of the polynomial in the H(z) denominator, which unfortunately is a procedure which takes much more time than generating H(z) itself. Note that for LPC's original use in communications and speech synthesis, the factorization problem does not apply since H(z) is used in its original form.

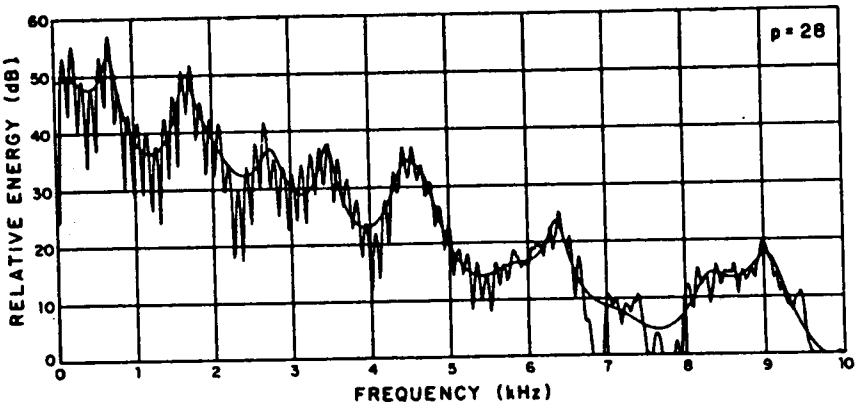


FIGURE 6. Frequency response of H(z) superimposed on the original spectrum.

A way around the factorization problem has been found by Itakwia[6] and is called the Minimum Prediction Residual principle. The minimum prediction residual is given by

$$S = aVa^T$$

where "a" is the vector of prediction coefficients and V is the correlation matrix of the input signal. The minimum prediction residual indicates a quality of fit, and can consequently be used to judge the match between previously generated correlation matrices and newly generated predictor coefficients. The actual formant frequencies are not calculated. Itakwia's program ran 22 times real time on a powerful DDP-516 computer, thus indicating that real time operation on an Apple II would be virtually impossible. Because the goal of real time operation could not be met, LPC was dropped from further consideration as a speech recognition method.

Bank of Filters

The second recognition technique investigated was the bank-of-filters method. This method, shown in Figure 7, consists of an array of band-pass filters each followed by a level detector. There are very many variations based on this basic block diagram, both analog and digital. A digital example is the Fast Fourier Transform (FFT), which has a number of equally spaced filters equal to the number of input samples. An analog example is the voiceprint machine, whose typical output is shown in Figure 8 (from Rabiner and Schafer,[4] page 311). The vertical

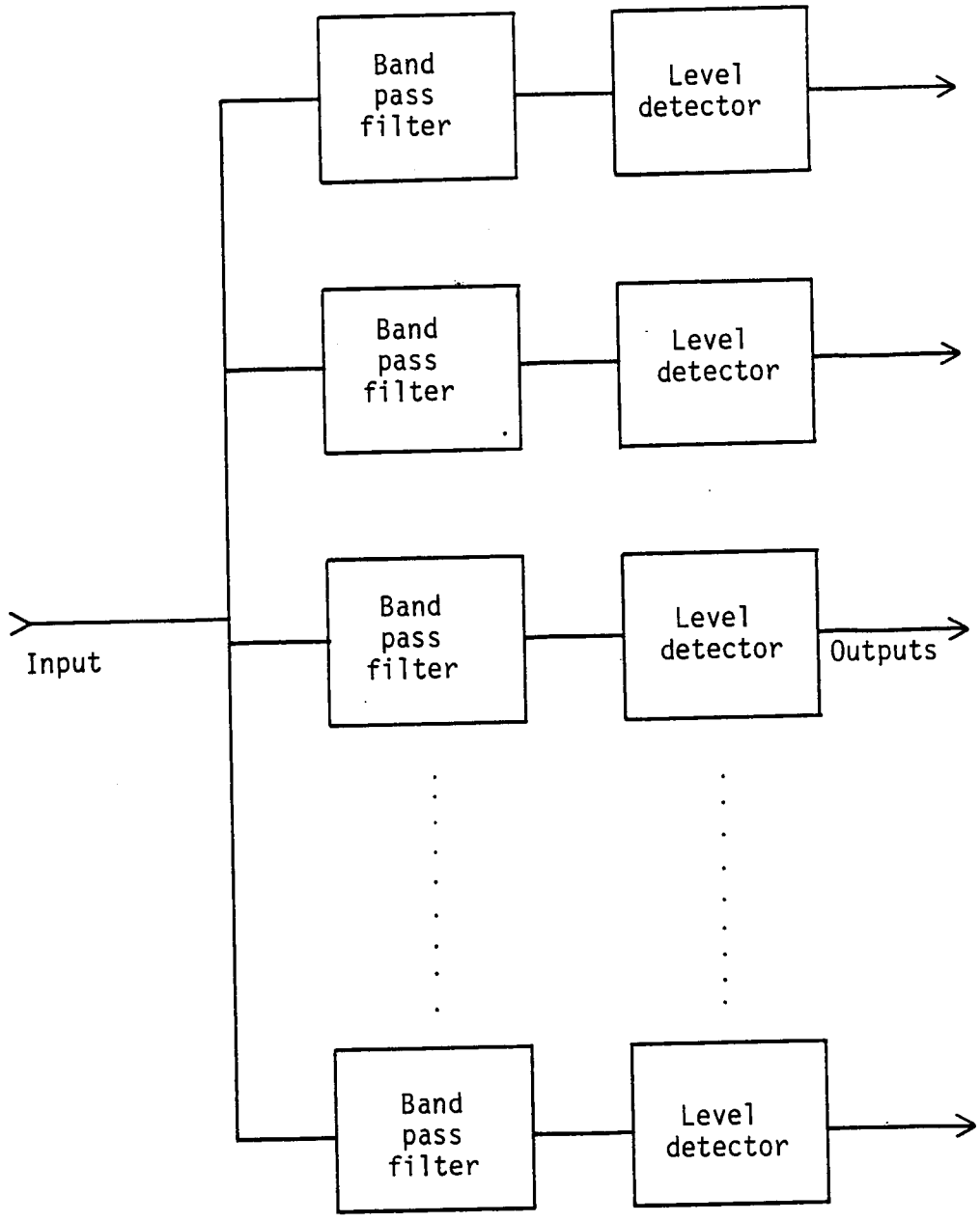


FIGURE 7. Diagram of the bank-of-filters method.

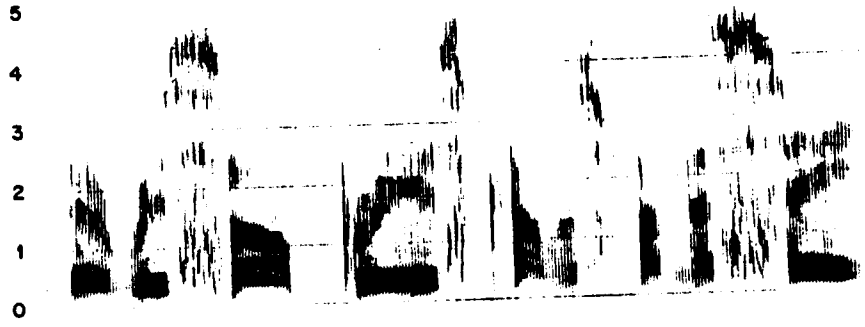


FIGURE 8. Typical output of a voiceprint machine.

scale represents the short-term spectral content of the signal, while the horizontal scale represents time. Higher intensity is shown by the dark areas. Most voiceprint machines generate their plots with an analog bandpass filter that has a varying center frequency.

The bandpass filters in a typical bank-of-filters system have bandwidths of a few hundred hertz because of the discrete line spectra of voiced signals. Since the recognition algorithm is looking for general formant frequencies instead of pitch information, the filters must be wide enough such that a change in fundamental frequency will not move spectral lines in or out of a filter's bandpass. This will enable the filters' output levels to remain the same with the same vowel even though the pitch period may have changed. Unfortunately the wider bandwidths limit the number of useful filters in a typical system to around twenty, and thus will limit the resolution for determining formant locations.

This method of speech analysis is currently the most popular for speech recognition. An example of a commercially available speech recognition unit is the Interstate Electronics[7] SYS-300, whose analysis is performed by an array of sixteen logarithmically spaced bandpass filters followed by envelope detectors. The outputs of the detectors are then multiplexed to an analog-to-digital converter to quantize the information and store it in a computer. The bandpass filters, envelope detectors, and multiplexer are all integrated onto one analog integrated circuit using switched-capacitor technology. Isolated word recognition for this system is very good; typical results are greater than 99% accuracy with a 100-word vocabulary.

In spite of the encouraging results from other companies, this method was not chosen for use in this thesis because the large number of multiplications required made real-time operation impractical.

Lokerson Algorithm

The third technique investigated was based on a machine designed by Donald Lokerson.[1] Lokerson originally developed this machine as an aid for deaf people, although speech analysis by computer has also been investigated.[8] The block diagram of the basic system is shown in Figure 9.

Quantizer

The Lokerson system has some similarities with the bank-of-filters approach except that the actual formant frequency determination is performed by an added device called the quantizer. The bank-of-filters approach would assume that the filter that had the highest output would be centered on the formant frequency, and thus the center frequency of the filter would be the formant frequency. Since the spectral spacings prescribe a minimum bandwidth and spacing frequency of the filters, the bank-of-filters method can produce only a rough estimate of the formant frequency. On the other hand, the Lokerson method only uses the filters to separate speech into its different formants and leaves the formant frequency determination to the quantizer, which can describe the formant frequency with very good resolution.

The quantizer, described by Lokerson as a multiple-crossing delta-modulator type, has characteristics similar to frequency modulation slope

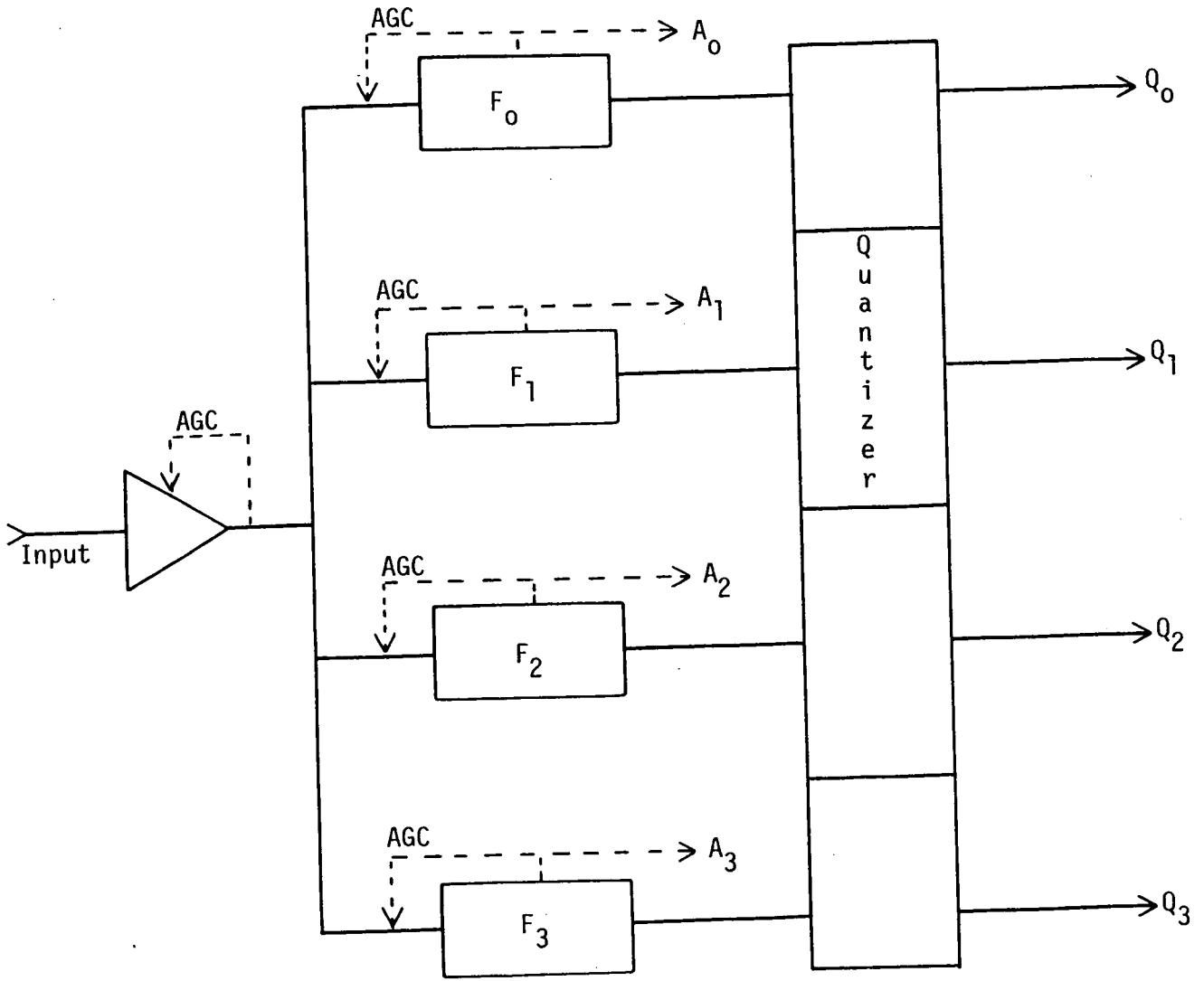


FIGURE 9. Diagram of the Lokerson machine.

detectors. As depicted in Figure 10, the quantizer output produces a pulse whenever the input voltage changes a prescribed amount. From the figure it can be seen that the average pulse rate is proportional to both the amplitude and the frequency of the input signal. Since the filter section preceding the quantizer contains an automatic gain control, the input is of constant amplitude, and thus the output pulse rate is proportional to only the formant frequency.

Another way of analyzing this device is to consider it as an ordinary delta modulator with all of its absolute-value outputs summed over a fixed time frame. This process can be described by the summation

$$Q = \sum_{n=0}^{N-1} ABS(S(n+1) - S(n))$$

where S(n) is the discrete - time version of the input signal, N is the number of samples per frame, and ABS is the absolute value operator. The S(n+1) - S(n) quantity can be considered as the slope, or derivative, of the signal; and the summation can be thought of as a voltage level detection. This is analogous to an FM slope detector in which the differentiator is the slope filter, and the summation is the amplitude detector.

The quantizer technique has a very important advantage over other conventional techniques, such as zero-crossing rate, in that it can take an input of several frequencies at once and determine an average frequency. When multiple frequencies are present, the output of the differentiator will contain amplitudes that are weighted by the input signal amplitudes and their corresponding frequencies. The summation process

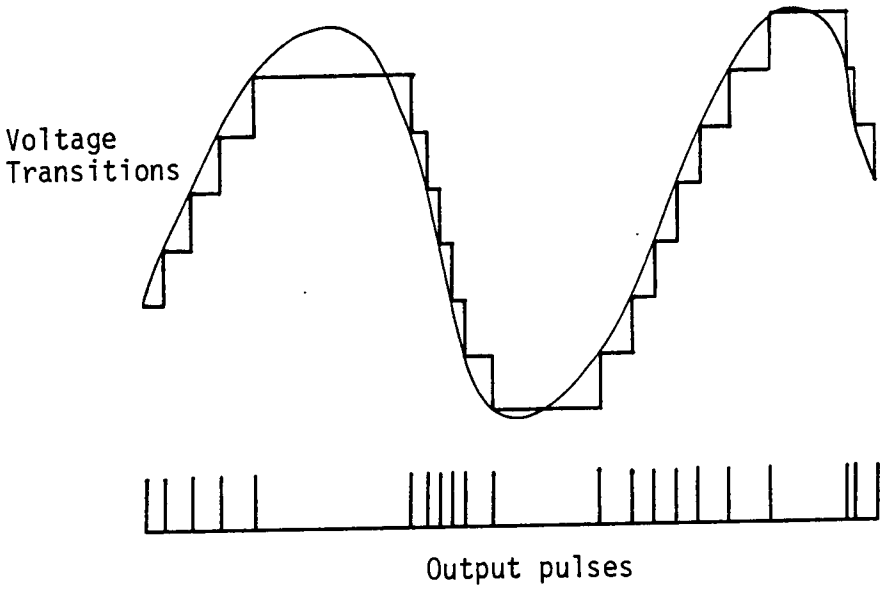


FIGURE 10. Operation of the quantizer.

will then take an average of these weighted amplitudes and produce an average frequency output. This concept will be explained mathematically in the next chapter.

Normalization of Formant Frequencies

One of the other unique features of Lokerson's machine is the normalization of the formant frequencies. Referring back to Figure 2, page 6, one notices that there is a significant variation of the formant frequencies for different people. If a speech recognition machine were used for several different people, this variation in formant frequencies could cause great difficulty. However the figure shows another interesting feature in that most of the oval figures have a definite slope. Lokerson exploited this feature by defining the new quantities

$$X = \frac{Q_1}{Q_2} = \frac{\text{formant frequency 1}}{\text{formant frequency 2}}$$

$$Y = \frac{Q_3}{Q_2} = \frac{\text{formant frequency 3}}{\text{formant frequency 2}}$$

Thus it is hoped that better speaker independence, even between adults and children, can result from using the X and Y quantities. As an example, consider the effect of playing back a tape recorder at twice the normal speed to produce a comical high-pitched voice. Even though the formant frequencies have been changed by a factor of two, the speech is still intelligible, and the X and Y quantities would be the same.

The chief advantage of the Lokerson method is its simplicity. Whereas the other speech recognition methods required large amounts of multiplication and division, the Lokerson method requires only a few multiplications by constants for the filters and only additions and subtractions in the quantizer. The simple algorithm allowed the goal of real time operation to be met, and thus the Lokerson method was chosen as the basis for the speech recognition technique in this thesis.

Common Issues

Despite the very different approaches in the three methods, there are several problems that all three techniques have. The first problem is pitch detection. None of the three methods produce a pitch signal directly, therefore the pitch detection method can be studied independently. The most common method is the autocorrelation method, which unfortunately requires over ten thousand multiplications for a typical frame. Rabiner and Schafer[4] discuss this method and other possible methods in chapter 4. For moderately accurate speech recognition, the pitch period is not actually required unless intonation differences are being sought, such as for detecting questions or sentence endings. Pitch detection is also essential if the speech is going to be synthesized.

Another problem common to all three methods is the manner in which the machine analyzes the pattern of the vowel movements to determine what word is being spoken. The most common method is the template method. This method records the parameters for several spoken words

and then compares them with parameters generated later on during the recognition process. This issue will be discussed later in the next chapter.

In summary, it has been shown that each of the three systems has its own advantages and disadvantages, and that each method has an area in which it is best. Linear Predictive Coding is the most accurate system and is the most complex, but is the clear winner if the speech is going to be synthesized. The bank-of-filters approach is best when analog hardware is available, and the Lokerson method is best when hardware resources are limited.

CHAPTER III

IMPLEMENTATION

Adaptation of the Lokerson Algorithm

The implementation of the method used in this thesis consists of three main parts: the formant filter, the quantizer, and the pattern match. The first two parts are performed by an Intel 8088 machine language program that operates in real time. Since the pattern matching algorithm is not a real-time calculation, it is instead programmed in BASIC and is run on the Apple II's 6502 microprocessor to ease the programming task. Although the Lokerson machine is the basis for the technique, the program does deviate from Lokerson's device in several respects.

The block diagram of the available hardware is shown in Figure 11. The automatic gain control (AGC) amplifier is a home-made circuit based on a commercially available noise-reduction compandor circuit. The output of the AGC amplifier goes to an 8-bit linear analog-to-digital converter, also home-made, which is capable of sampling every 8 microseconds (125 KHz), although this application uses the data at a much slower rate. This 8-bit data then interfaces with either of the microprocessors via the Apple II bus. Other experiments, such as Fast Fourier Transforms, digital filtering, zero crossing rate, and delta modulators can also be performed with this setup.

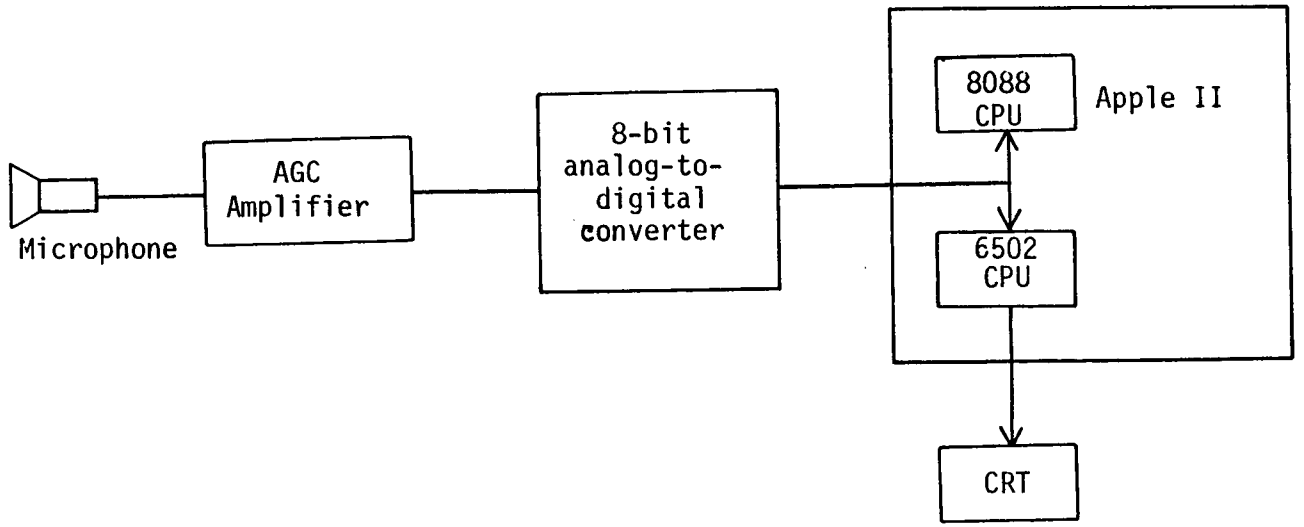


FIGURE 11. System hardware.

One of the differences between this program and the Lokerson algorithm arises because of the 8-bit converter resolution. In Figure 12, reprinted from Rabiner and Schafer,[4] page 255, a spectrum plot of a typical vowel is shown. Note that the third formant is 40 dB below the first formant, and note the 100-dB dynamic range capability of the measuring system used in this example. Since an 8-bit converter can provide only 48-dB dynamic range at best, the computer would have a very difficult time resolving third formants. It must be pointed out that speech transmitted over an 8-bit system is very intelligible to humans. This suggests that the third and higher formants are actually not necessary and are merely byproducts of speech rather than essential pieces of information. For this reason no attempt was made to analyze the third formant in this thesis.

The first two formants are resolved by lowpass and bandpass digital filters as shown in Figure 13. Plots of the frequency responses of the lower and upper filters are shown in Figures 14 and 15 respectively. As an example of how the filtering process separates the formants, consider applying the "e" vowel, whose FFT is shown in Figure 16, to the inputs of the filters. The resultant signals from the lower and upper filters are shown in Figures 17 and 18 respectively. It is evident from the figures that the separation of the formant frequencies is far from perfect, especially for the second formant. In spite of the poor separation characteristics, this method was still used because of its simplicity. The second formant filter is particularly simple since it requires only additions and subtractions and no multiplications.

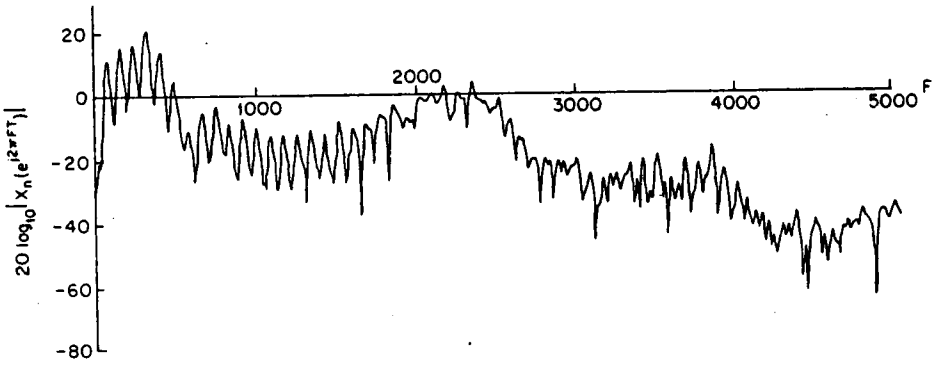


FIGURE 12. Large dynamic range spectrum.

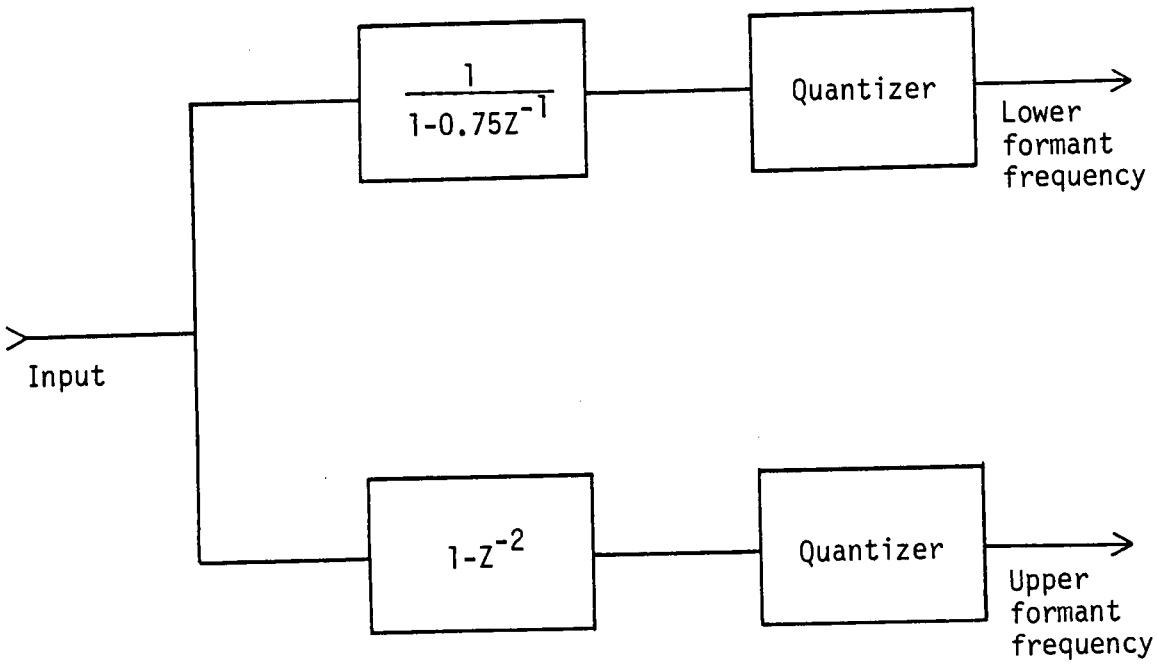


FIGURE 13. Formant filter and quantizer system.

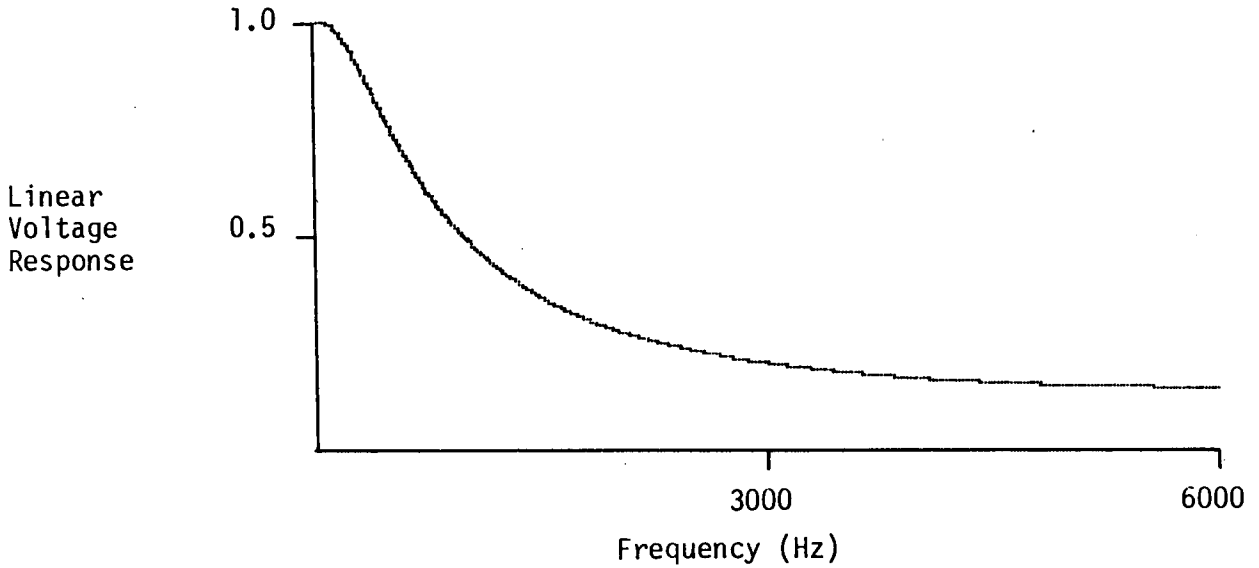


FIGURE 14. Frequency response of the lower formant filter.

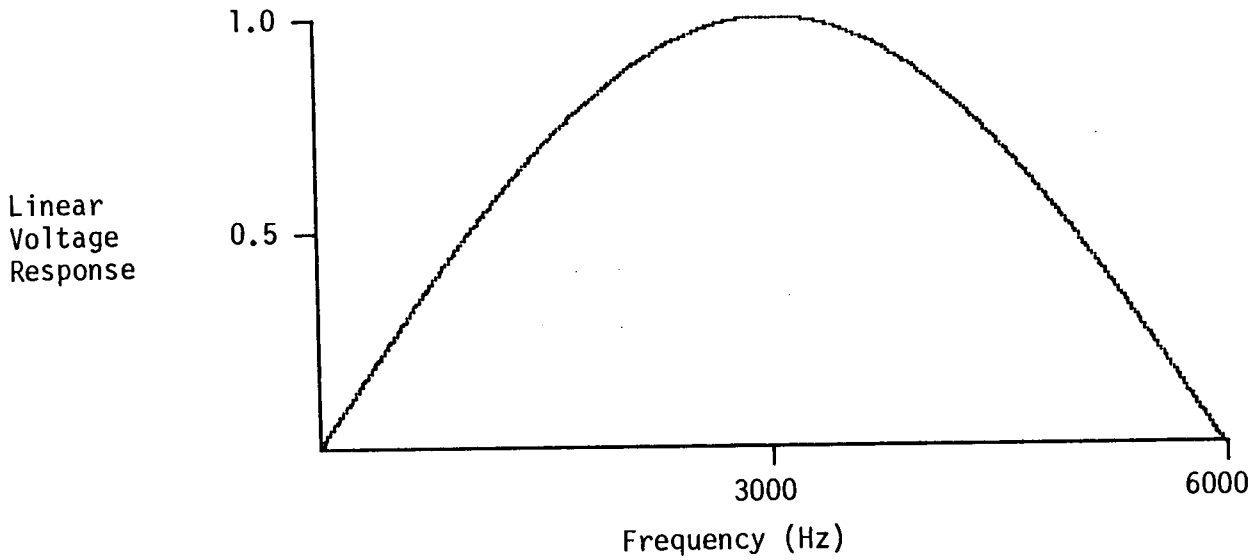


FIGURE 15. Frequency response of the upper formant filter.

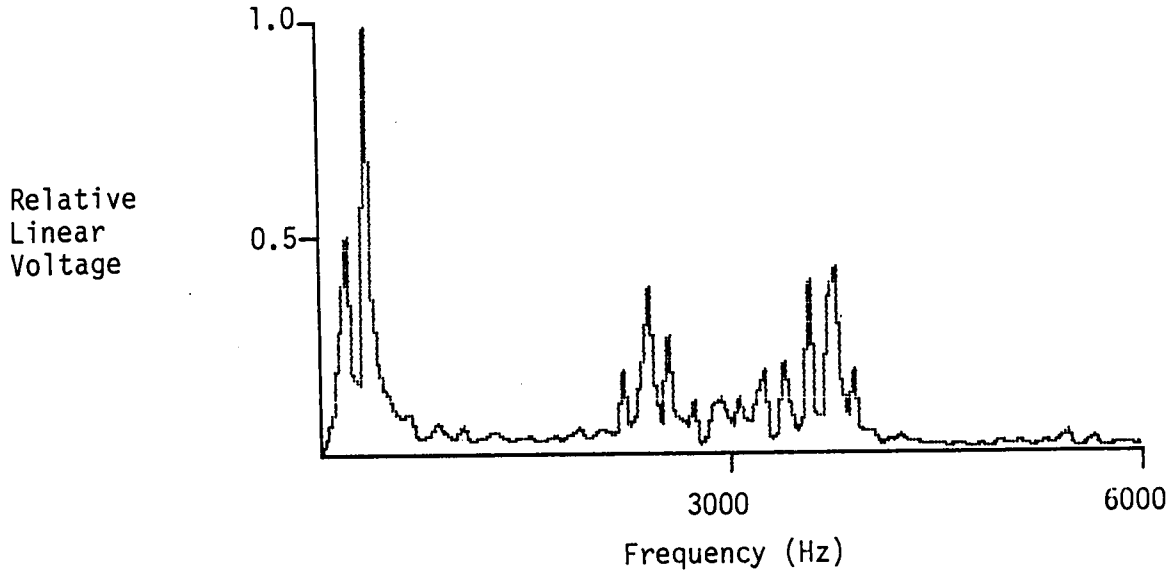


FIGURE 16. Spectrum of the "e" vowel before filtering.

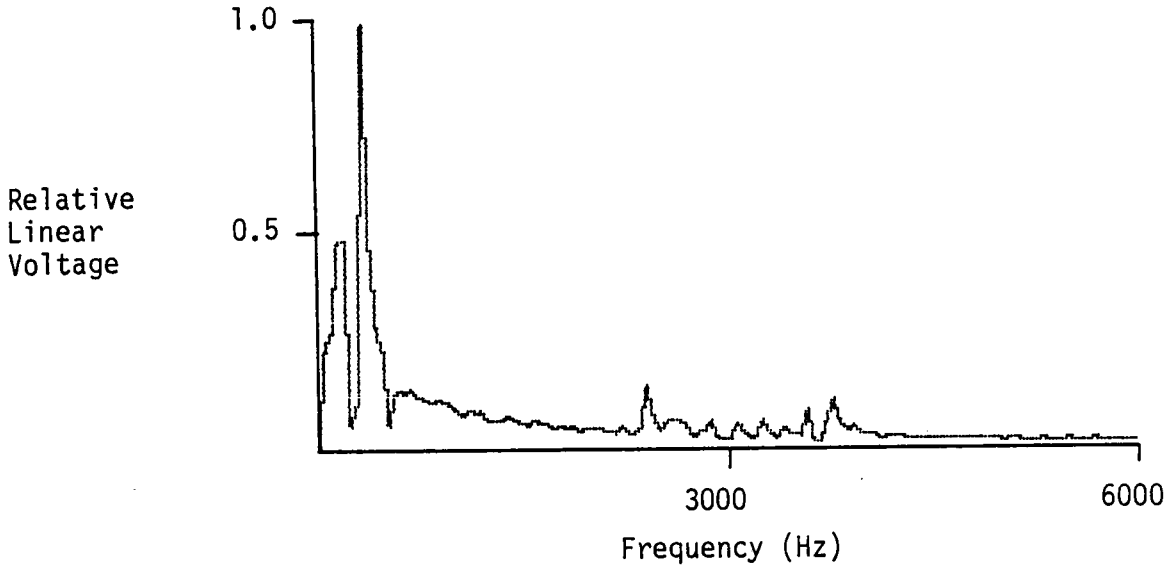


FIGURE 17. Spectrum of the "e" vowel after lower formant filtering.

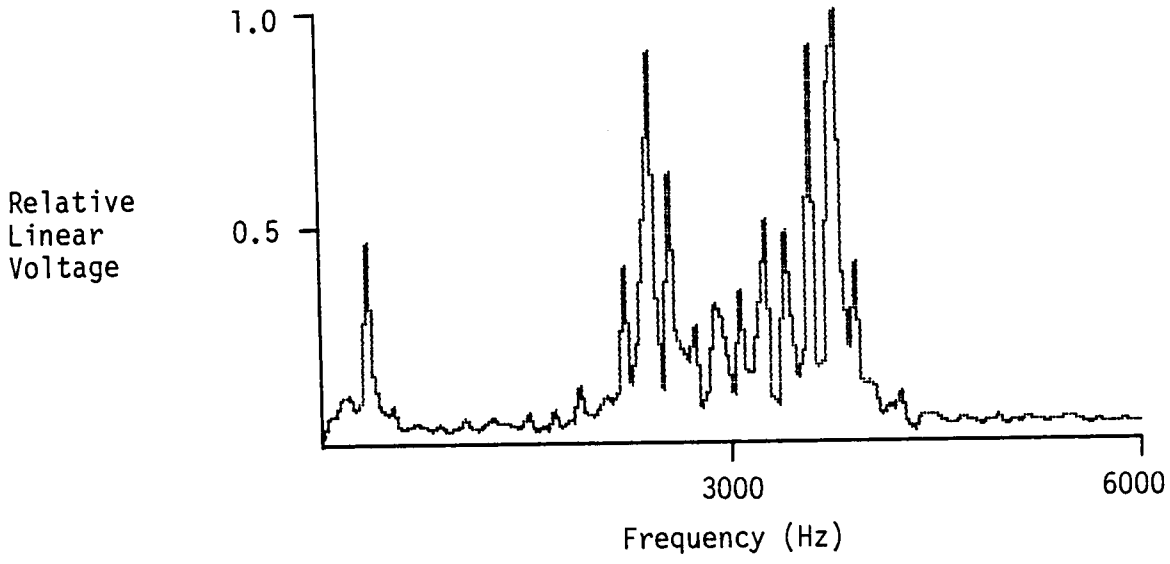


FIGURE 18. Spectrum of the "e" vowel after upper formant filtering.

Another deviation from the Lokerson method is the removal of the $X = Q_1/Q_2$ and $Y = Q_3/Q_2$ normalization steps. These changes were made because the third formant frequency is unavailable and because the second formant calculation is somewhat questionable because of the second formant filter. It was felt that one accurate number and one questionable number is better than one questionable ratio.

The third deviation involves the quantizer. Instead of the pulse method described in the previous chapter, the approach taken here was the accumulated delta modulator concept, also described in the previous chapter. Another modification was to remove the automatic gain control for each filter and to replace it with an amplitude normalization scheme. Figure 19 shows how the amplitude normalization is used in the modified quantizer. The mathematical equivalent of Figure 19 is

$$Q = \frac{\sum_{n=1}^{N} ABS(S(n) - S(n-1))}{\sum_{n=0}^{N} ABS(S(n))}$$

where S(n) is the input signal with the DC component removed (average value of zero), N is the number of samples per frame, and ABS is the absolute value operator. This normalization method has an advantage over using an AGC for each filter in that the AGC method would require a multiplication for each sample to bring the overall amplitude to the correct level. The normalization method requires only additions and one division per frame.

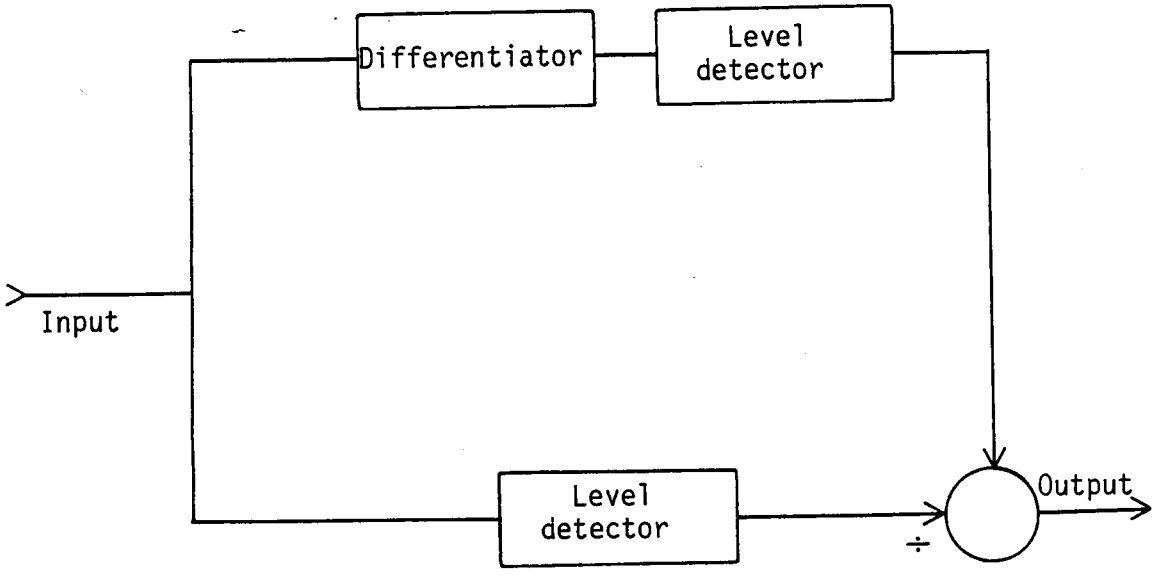


FIGURE 19. Modified version of the quantizer.

This new quantizer has interesting properties when input signals containing several frequencies at once are used. Suppose the input signals were

$$S(n) = A \sin(anT) + B \sin(bnT) + C \sin(cnT)$$

where T is the sampling period. The output of the differentiator would be

$$\frac{ds}{dn} = aAT \cos(anT) + bBT \cos(bnT) + cC \cos(cnT)$$

The output of the level detectors is difficult to define because of the absolute value operator. If the amplitudes of the sine waves are relatively equal, then the output of the detector would be approximately proportional to the sum of the sine amplitudes. Thus the final quantizer output would be

$$Q \cong \frac{T(aA + bB + cC)}{A + B + C}$$

The T would be absorbed by a general scaling factor.

This result has the interesting interpretation of "average frequency" since it is equivalent to the equation for the expected value, or average, of a discrete random variable.

$$E[F] = \sum_{i=1}^{N} f_i \, P(f_i)$$

The quantizer relates to the expected value by

$$P_1 = \frac{A}{A+B+C}, \quad f_1 = a$$

$$P_2 = \frac{B}{A+B+C}, \quad f_2 = b$$

$$P_3 = \frac{C}{A+B+C}, \quad f_3 = c$$

$$\sum_{i=1}^{3} f_i \, P(f_i) = \frac{aA}{A+B+C} + \frac{bB}{A+B+C} + \frac{cC}{A+B+C} = \frac{aA+bB+cC}{A+B+C}$$

As for the approximation, the same problem exists for both the numerator and the denominator of the quantity Q, and thus some of the approximating effects will cancel out. What is still an issue is how some of the frequencies discriminate against others. As an example of the effect of different frequencies with different amplitudes, consider the hypothetical waveform shown in Figure 20. Since the summation process is equivalent to an integration process, the level detector would treat the waveform as the sum of the integrals of the individual sine waves. Because the lower amplitude wave has an average value of zero, it will contribute little, if any, to the output of the level detector. This has an effect of discriminating against the lower level signals, or equivalently sharpening the peaks of the speech formants.

Pattern Matching

The final main block of the system is the pattern matching routine. Although the final method chosen was the common template approach,

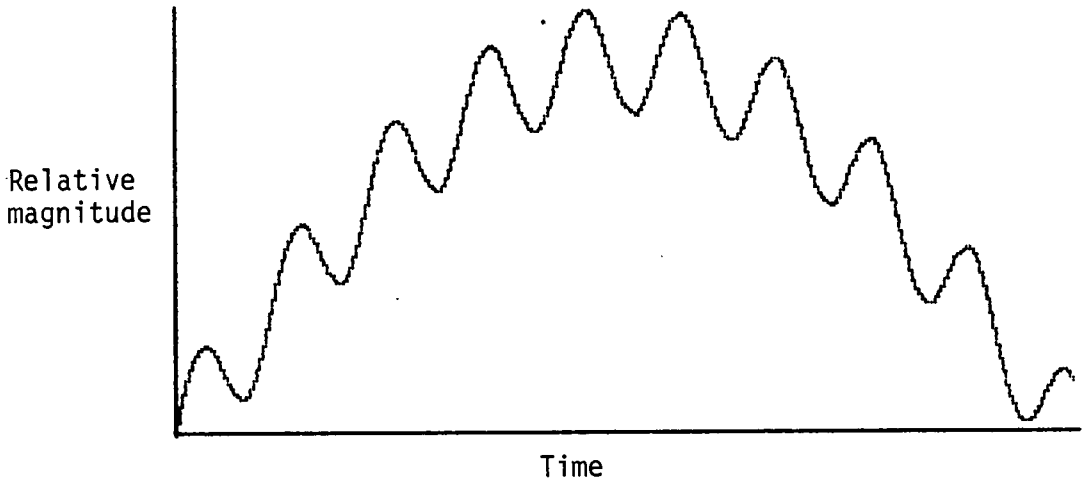


FIGURE 20. Hypothetical input signal.

considerable effort was spent on a novel finite state machine approach. A relative best-of voting system decides which template match is best among the templates stored in the vocabulary.

Proposed Finite State Machine

The finite state machine method involved defining regions similar to the ones shown in Figure 2, page 6. A series of level comparisons took the data from the quantizers and decided in which of the regions the current data fell. This region specified which "state" the machine was in. If the data fell in the same region on the next frame, the machine stayed in the same state and no new mode was generated. During the training phase a series of state transition, or nodes, were recorded and a linked-list table was created. If two different words started out the same, the first part of the linked list could be shared by the two words. An example of a transition pattern is shown in Figure 21 with the numbers inside the nodes corresponding to typical regions the data points may have passed through. When in recognition mode, the computer checked which path, if any, the word may have followed.

This system had several advantages but unfortunately had even more disadvantages. The main advantage was that the recognition process could proceed while the word was coming in rather than making the comparisons afterwards. If the traversed path reached a dead end, the computer could conclude that it had found the end of a word in a connected speech stream, thus isolating individual words and making connected speech recognition possible. Another advantage came from the

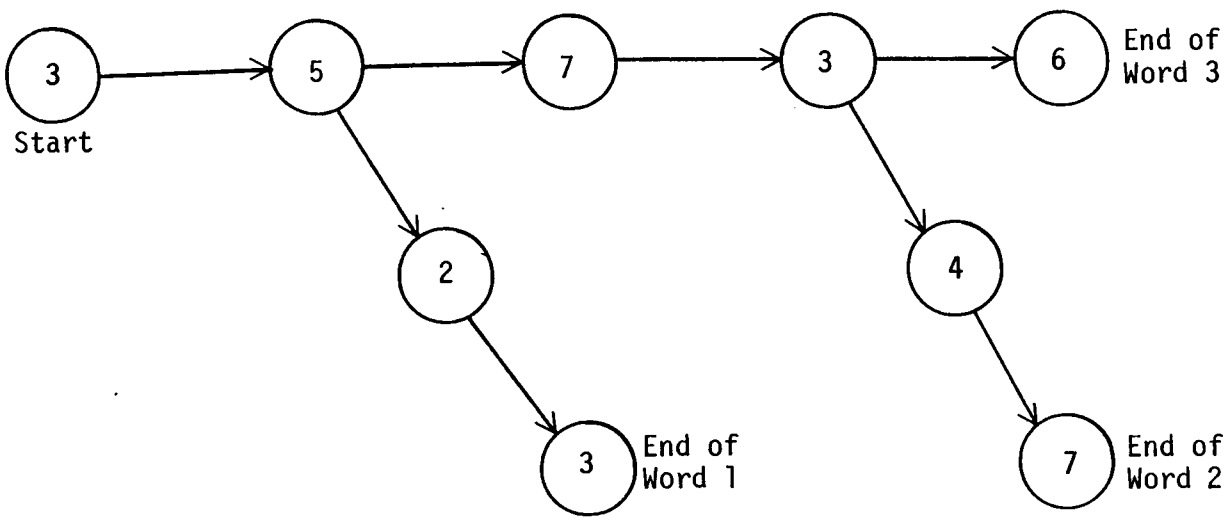


FIGURE 21. Finite state machine transition example.

repeated state removal in which the time spent on a certain vowel was removed. This eliminated problems caused by the subject talking at different speeds. A final advantage was that the information generated by this method was very compact.

The disadvantages, however, were so severe as to make this method worthless. The main disadvantage was that this method had no tolerance for errors. Any incorrect state would send the computer down the wrong branch and cause drastic misinterpretations. Since the input data was rather noisy and erratic, erroneous states were certain to occur and cause problems. If a more sophisticated and accurate formant determination scheme were available, the input data would be more consistent and could make the finite state machine approach feasible.

Template Method

Unlike the finite state machine approach, the template matching approach can tolerate an arbitrary amount of error. The matching algorithm actually calculates the amount of error and then selects the word with the least error. The equation

$$E = \frac{1}{M} \sum_{m=0}^{M} ABS(X(m) - Y(m))$$

performs this error calculation by subtracting the individual data points in the input template, represented by X(m), and the reference template, represented by Y(m). As an example, consider the template for the word "sea" (lower formant only) shown in Figure 22. This

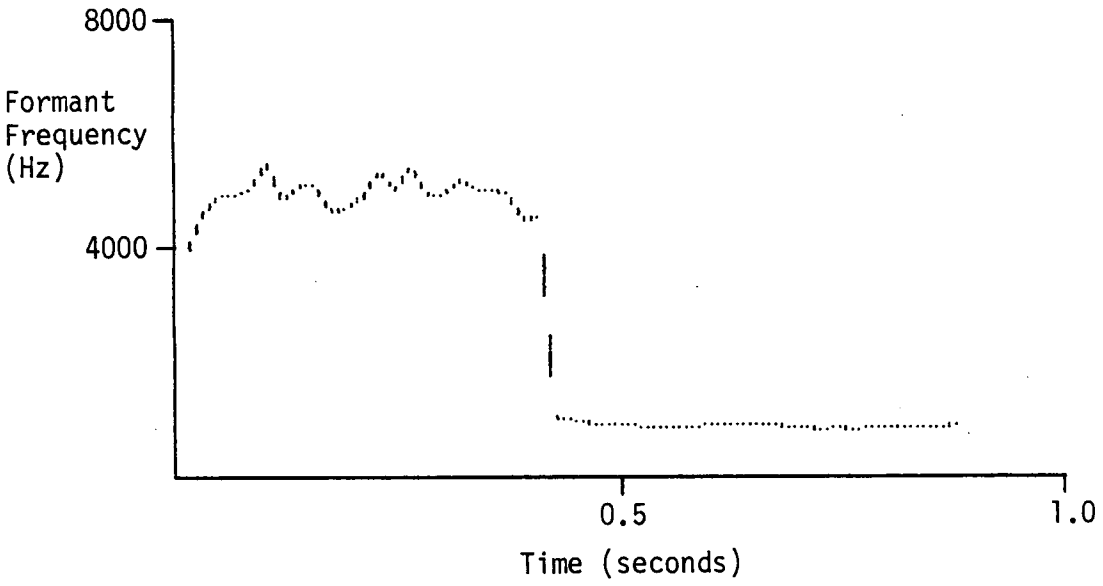


FIGURE 22. Formant frequency versus time for the word "sea."

template is a plot of the quantizer output for the lower formant plotted as a function of time. Figure 23 now shows another plot of the word "sea" spoken again superimposed on the data shown in Figure 22. The area between the two curves is highlighted. The matching algorithm takes this highlighted area, integrates it, and divides the integrated total by the time deviation of the template to find the average error.

The large error spot in Figure 23 points out one of the problems with the template method, namely the rate at which the words are spoken. Supposedly a time-warping algorithm could be used to cause major changes to occur at a prescribed rate. This idea has received much attention in the literature, for example see Christiansen and Rushforth,[9] but was not pursued in this thesis because adequate results of 90% accuracy were already being met.

System Summary

To summarize the operation of the system, first refer to the block diagram shown in Figure 24. In the block diagram and in the following equations, n refers to the sample number, N refers to the number of samples per frame, m refers to the frame number, and M refers to the number of frames per template. The samples, indexed by n, occur at a rate of 13,000 samples per second. N is always 256, and thus 20 mS worth of speech completes one frame and one Q_H and Q_L value. The number of frames, M, varies between 20 and 70 depending on the length of the utterance. The equations describing the individual blocks in the diagram are

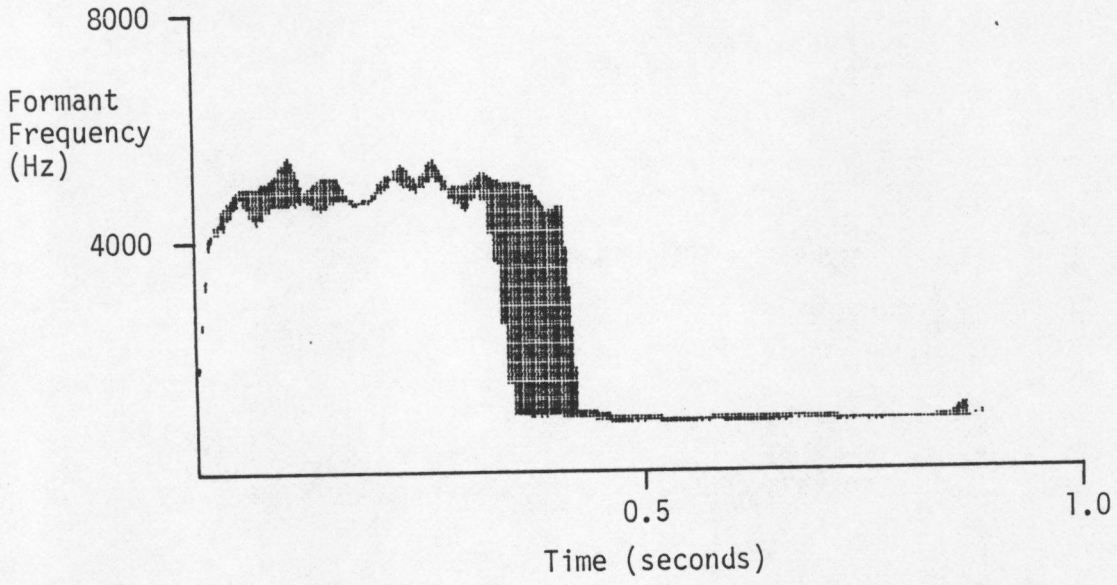


FIGURE 23. Data from Figure 22 superimposed on another utterance of the word "sea."

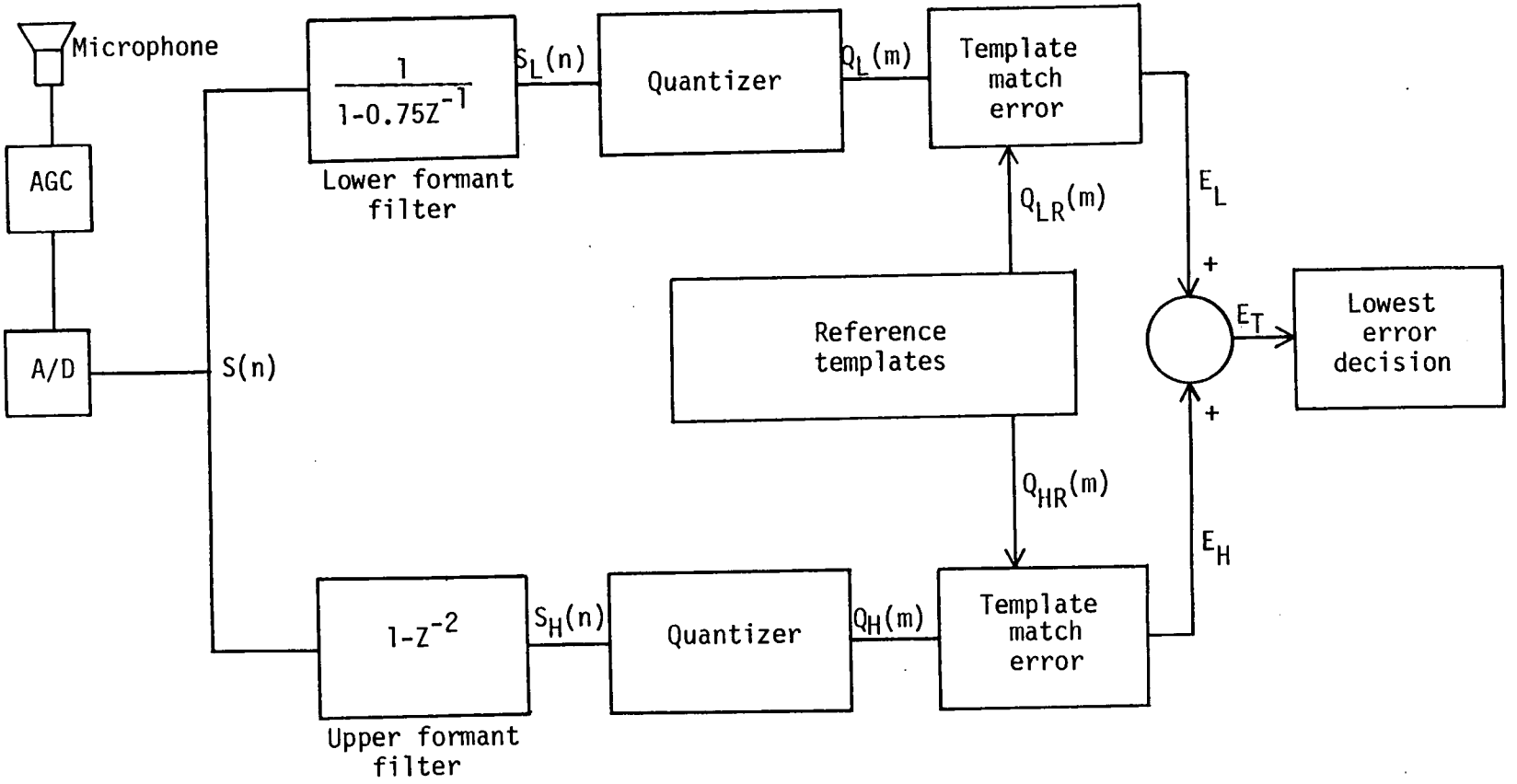


FIGURE 24. System block diagram.

$$S_L(n) = S(n) + 0.75\, S_L(n-1)$$

$$S_H(n) = S(n) - S(n-2)$$

for the formant filters, and

$$Q_L(m) = \frac{\sum_{n=1}^{N} ABS(S_L(n) - S_L(n-1))}{\sum_{n=0}^{N} ABS(S_L(n))}$$

$$Q_H(m) = \frac{\sum_{n=1}^{N} ABS(S_H(n) - S_H(n-1))}{\sum_{n=0}^{N} ABS(S_H(n))}$$

for the quantizers. Before each frame the computer waits for the voltage to reach a certain level to insure a more consistent waveform for each frame calculation. A time-out timer determines if the word is finished. The equations for the template matcher are

$$E_L = \frac{1}{M} \sum_{m=0}^{M} ABS(Q_L(m) - Q_{LR}(m))$$

$$E_H = \frac{1}{M} \sum_{m=0}^{M} ABS(Q_H(m) - Q_{HR}(m))$$

$$E_T = E_L + E_H$$

where the R subscript refers to the reference template. The final decision is made by picking the reference template that has the lowest E_T.

The program for implementing the block diagram is broken into two main parts. The formant filters and quantizers, which are the critical routines timewise, are performed by the Intel 8088 program shown in Appendix A. All other functions are performed by the BASIC routine shown in Appendix B.

A typical session with the program proceeds as follows. The BASIC routine asks the operator for the desired spelling for the utterance he is about to enter. The routine then calls the machine language routine to generate a template from the microphone input signal. Once a template is generated, the BASIC routine stores the template as a reference template in variable TM%. If desired, more than one template can be stored under the same spelling to provide multiple chances for the desired word to be recognized under the correct spelling. When the operator is finished generating reference templates, the program switches into the recognition mode.

In the recognition mode, the Intel processor generates a template as before. The BASIC routine then performs a template matching error calculation for each of the reference templates generated before. Since there is almost always a difference in time length between the reference template and the input template, the error routine uses the shorter of the two lengths in the error calculation. A compensation factor proportional to the difference in the two template lengths is then added to simulate mismatch caused by the longer word. For example, if the

word "seven" were compared to "seventeen" and the shorter word determined the length, the "-teen" would be clipped off and an erroneous match would occur. The compensation, shown in line 1170 of the BASIC program, prevents this. The BASIC routine then decides which reference template had the lowest error and displays the spelling for that template.

CHAPTER IV

RESULTS

Before any performance results are quoted, it must be pointed out that the results vary widely depending on the skill of the operator. In the course of developing the program, the author became proficient in its use. Trial tests with new operators have shown that unskilled operators produce significantly less accurate results when compared to the author's results. When a new operator uses the program, it is essential that a new set of reference templates be generated because the speaker-independent performance of the system is quite poor. All results shown here are with the author as the operator except where noted otherwise.

Accuracy

Although the speech recognition program will accept any utterance for generating templates, a particular vocabulary was chosen for experimentation. The vocabulary consisted of the spoken numbers "zero" through "nine" with each of the words entered twice. This particular vocabulary was chosen because of its reasonable size and because of its potential applications for entering data into computers.

Table 1 shows the results of an accuracy test on the test vocabulary. Three separately-trained operating sessions of 100 utterances each were used to demonstrate how the accuracy is dependent upon the

Digit Spoken \ Recognition Response	0	1	2	3	4	5	6	7	8	9
0	295									0 5 0
1		298								0 0 2
2			298	2 0 0						
3			1 5 1	293						
4		0 1 0	1 0 0		298					
5						296				0 1 3
6							294	0 3 3		
7							0 2 0	298		
8			1 0 0	1 0 0					298	
9	0 1 0	0 3 2				4 0 2				288

TABLE 1. Error performance for the test vocabulary.

quality of the training templates. The three separate sessions are denoted by the three separate numbers within the boxes, the top, middle, and bottom numbers referring to the first, second, and third sessions respectively. The blank boxes denote three zeros, and the diagonal boxes, which represent the correct responses, have their three numbers added together. From the table one can see that certain digits have problems with only one session, such as the "zero" in the second session. Other digits, such as the "three," seem to have consistently poorer performance. The overall accuracy for the three sessions is about 86%.

A potential accuracy problem that must be pointed out is background noise. The microphone gain is purposely set high to pick up subtle sounds, such as the "f" in "five," which unfortunately makes the system susceptible to extraneous noises. If the system were operated in a noisy industrial environment as opposed to the author's quiet home, a severe reduction in accuracy would occur.

Speed

The recognition time for the system is a more consistent parameter as opposed to the system accuracy. The actual template is generated in real time and thus the generation time is of no concern to the operator. Using a BASIC program that has been compiled, the template comparisons can be performed in 1.4 seconds on the twenty utterance test vocabulary. This is equivalent to 0.07 seconds per reference template. If desired, this template matching program could be rewritten in machine language to produce much faster results.

Sample Data

As an example of what the typical data generated internally by the program looks like, the reference templates for the test vocabulary have been printed in Figures 25 through 34. To prevent cluttering up the plots, only the lower formant frequency is plotted. A second utterance of the same word is superimposed on the first plot to demonstrate the typical mismatch between the templates and to show the highlighted mismatch that is integrated to perform error calculations.

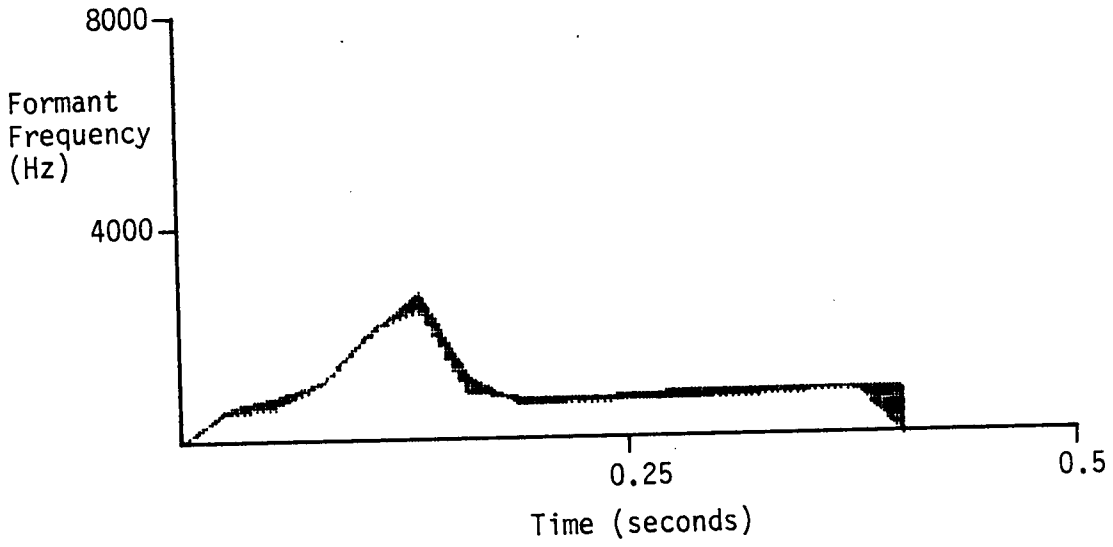


FIGURE 25. Formant plot of the word "zero."

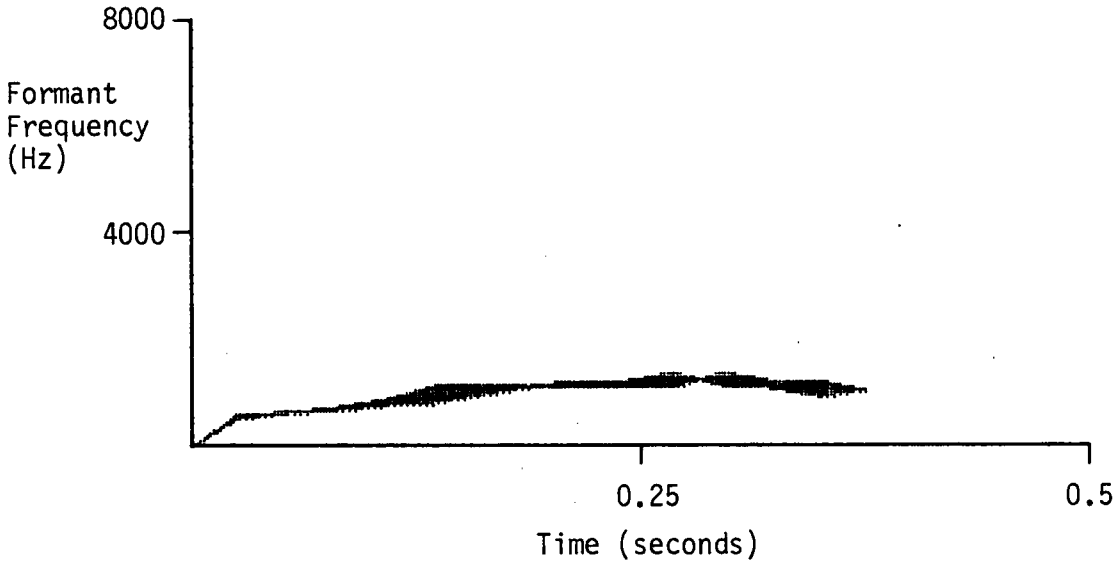


FIGURE 26. Formant plot of the word "one."

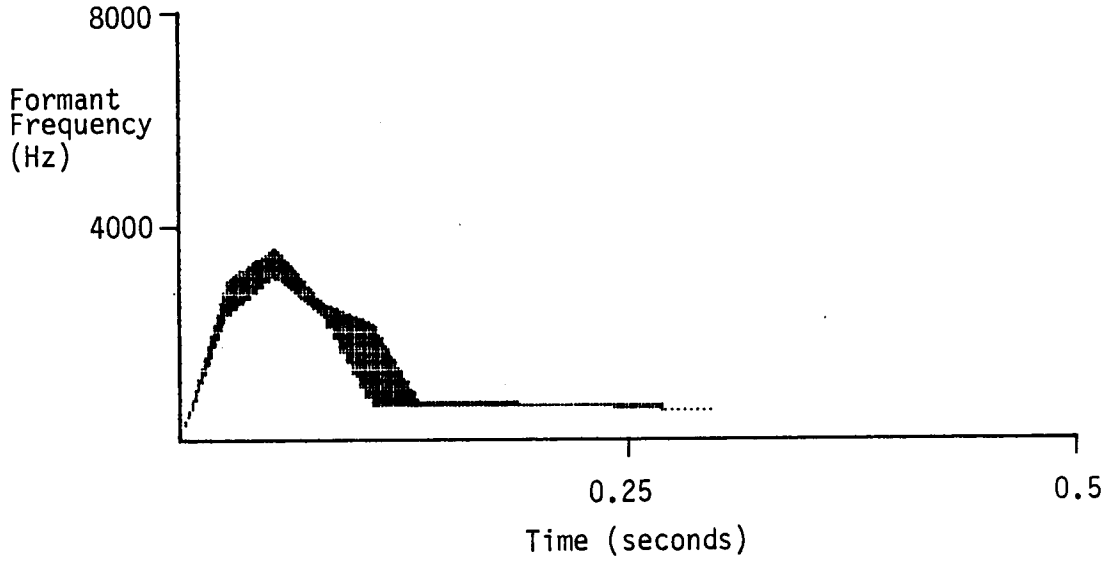


FIGURE 27. Formant plot of the word "two."

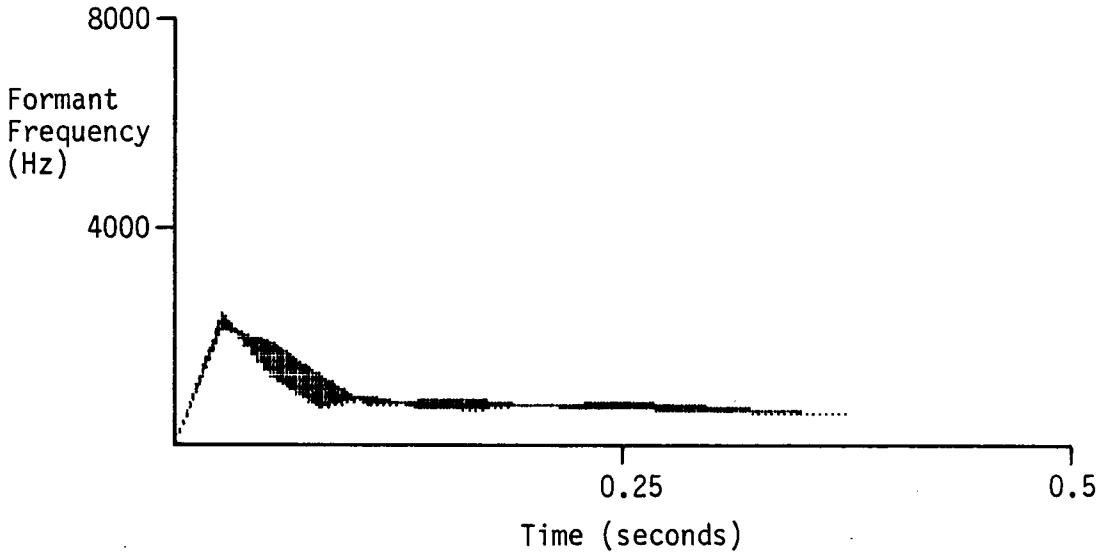


FIGURE 28. Formant plot of the word "three."

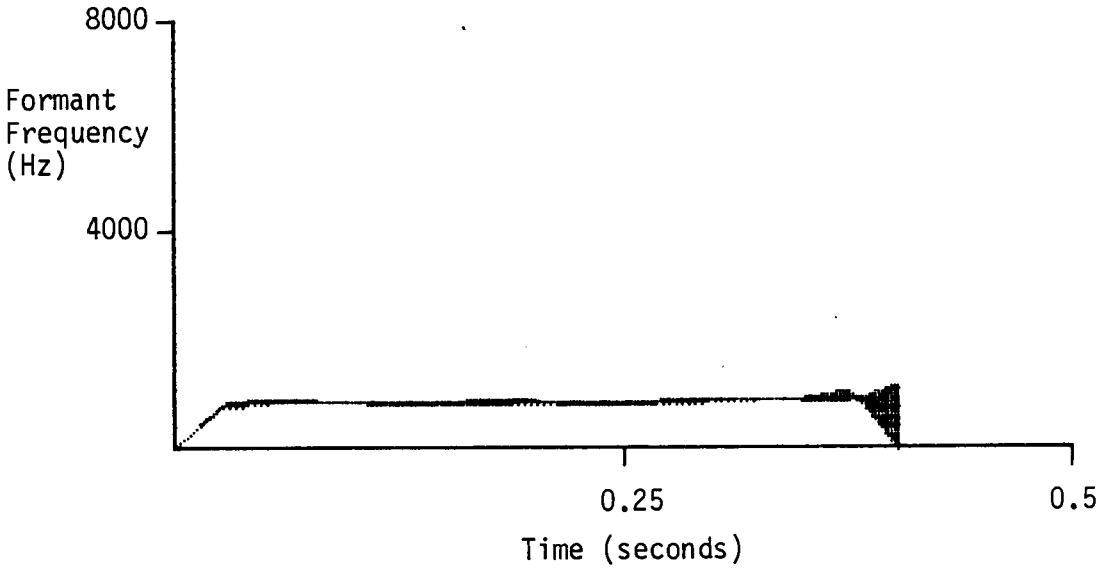


FIGURE 29. Formant plot of the word "four."

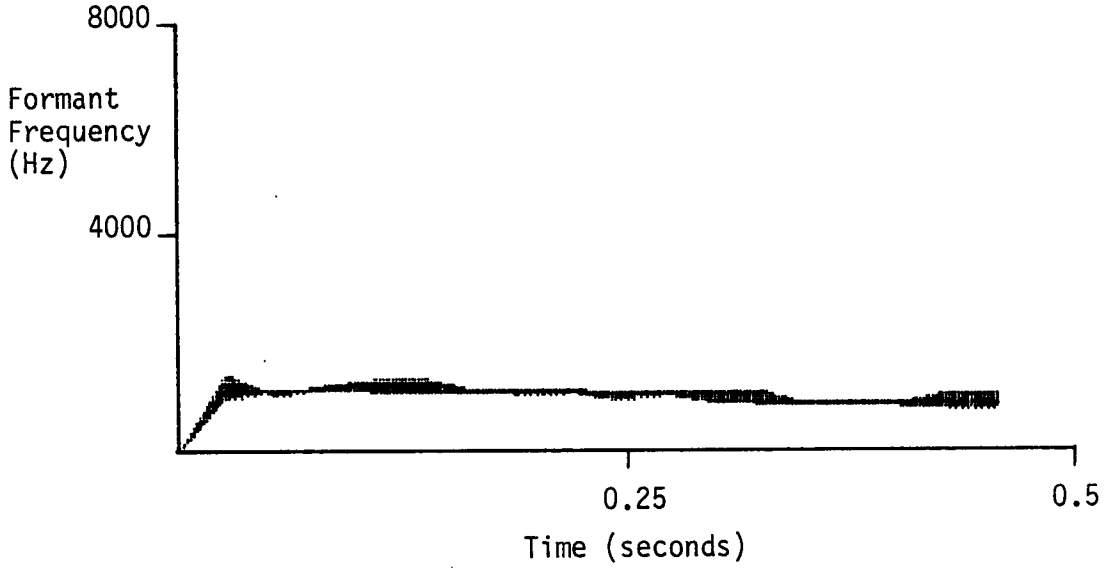


FIGURE 30. Formant plot of the word "five."

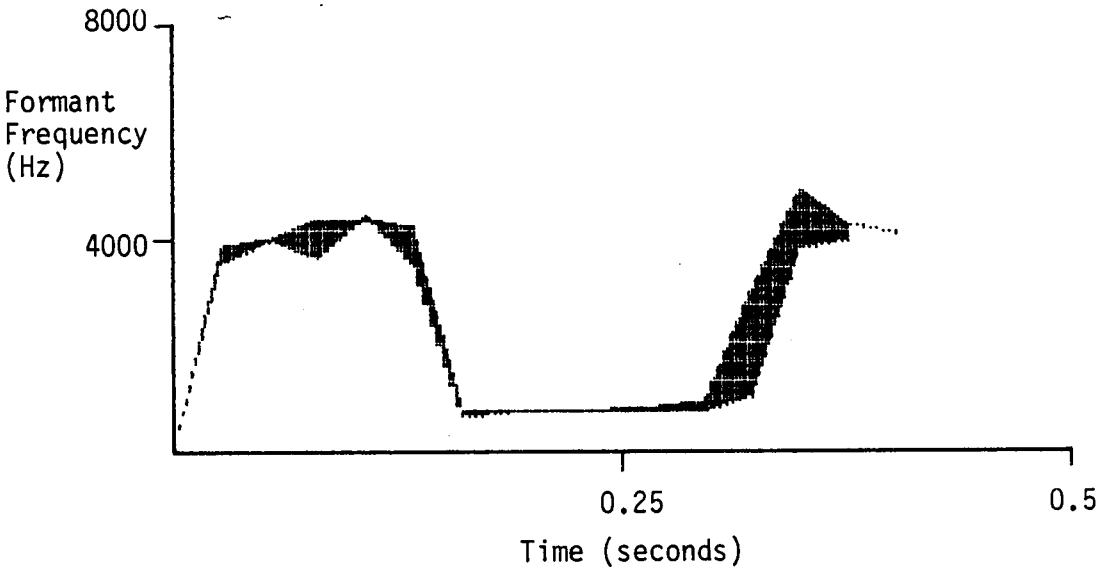


FIGURE 31. Formant plot of the word "six."

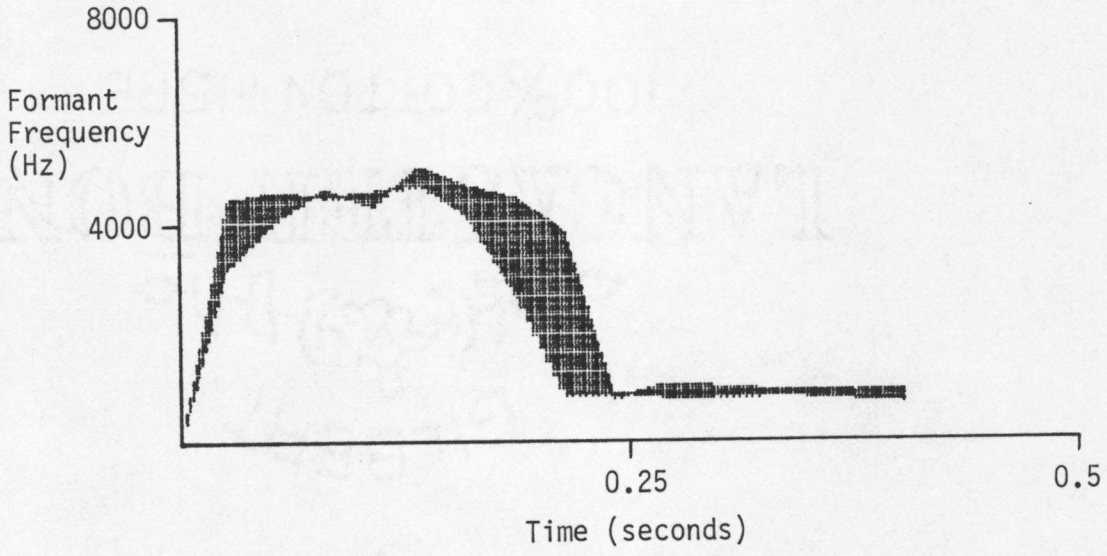


FIGURE 32. Formant plot of the word "seven."

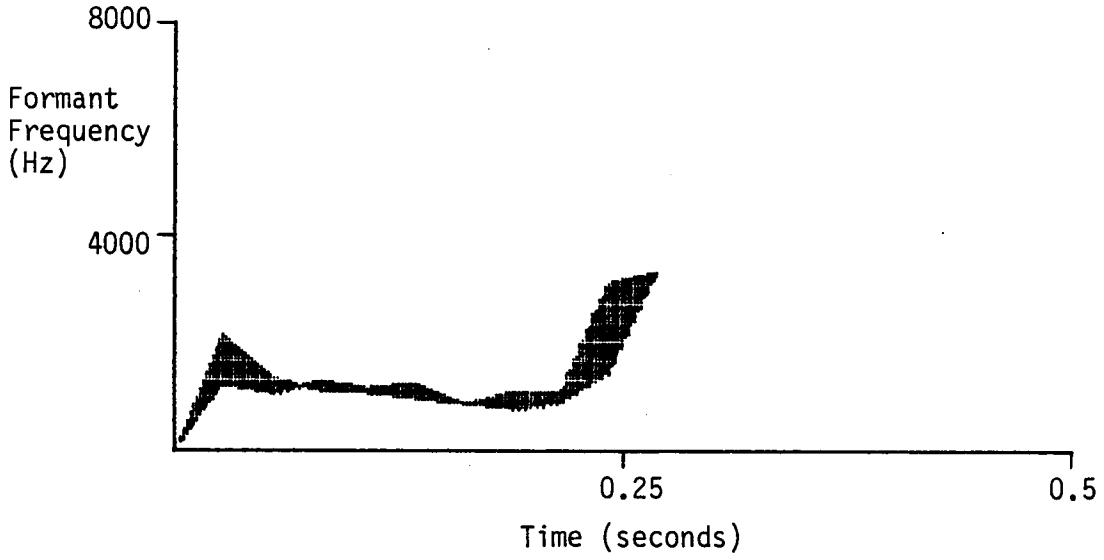


FIGURE 33. Formant plot of the word "eight."

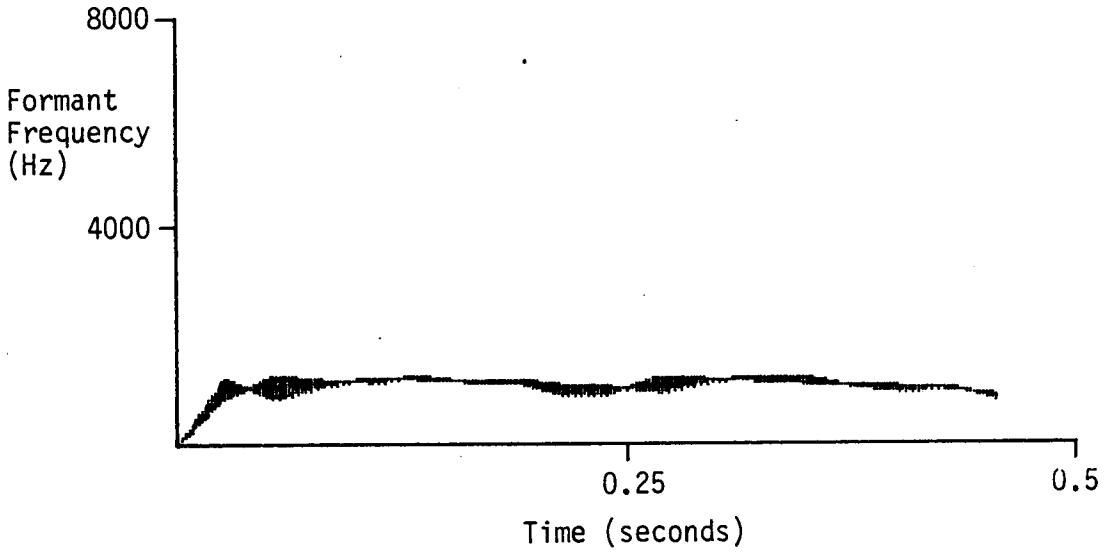


FIGURE 34. Formant plot of the word "nine."

CHAPTER V

CONCLUSIONS

In conclusion it can be said that the goals set forth in this thesis were met and that the program worked as desired. The algorithm developed in this thesis has been shown to be accurate and easily adaptable to microcomputers.

Possible Improvements

Although the system's performance can be considered adequate at the present time, there are several possible improvements that may interest the reader. The first improvement, time warping to compensate for different talking speeds, has already been mentioned in Chapter III. The second improvement is to logarithmically scale the formant frequencies before programming the template test, or similarly calculate percentage errors rather than difference errors. This would even up the sensitivities to errors between the upper and lower frequencies. The next improvement would be to require a maximum error level for the template matching to signal that the utterance quality was too low to make a good decision or that the utterance was not in the vocabulary. Other small quality improvements could be made, such as improving the starting trigger and increasing the resolution of the analog-to-digital converter.

LIST OF REFERENCES

LIST OF REFERENCES

1. Donald G. Lokerson; SPEECH ANALYZER United States Patent #4,039, 754:1977.

2. Larry Brantingham; "Speech Synthesis with Linear Predictive Coding," Interface Age, June 1979.

3. James L. Flanagan; Speech Analysis Synthesis and Perception, Springer-Verlag, New York:1972.

4. L. R. Rabiner and R. W. Schafer; Digital Processing of Speech Signals, Prentice-Hall, Englewood Cliffs, NJ, 1978.

5. J. Makhoul; "Spectral Linear Prediction: Properties and Applications," IEEE Trans. on Acoustics, Speech, and Signal Proc., Vol. ASSP-23, No. 3, pp. 283-296, June 1975.

6. F. Itakura; "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. on Acoustics, Speech, and Signal Proc., Vol. ASSP-23, No. 1, pp. 67-72, Feb. 1975.

7. Interstate Electronics SYS-300 Manual, company proprietary.

8. Donald G. Lokerson; "A Conceptually Unique Speech Training Aid System" and "A Unique Real-Time Speech Decoder that Operates from New Perspectives," 1979 IEEE International Conference on Acoustics, Speech, and Signal Processing, April 2-4, 1979.

9. Richard W. Christiansen, Craig K. Rushforth; "Detecting and Locating Key Words in Continuous Speech Using Linear Predictive Coding," IEEE Trans. on Acoustics, Speech, and Signal Proc., Vol. ASSP-25, No. 5, October 1977.

APPENDIXES

APPENDIX A

TEMPLATE GENERATION PROGRAM

```
  1000                    ESEG 1000H ; APPLE II ADDRESS
0000                      APPLEBYTE RB 1
  0000                    APPLEWORD EQU WORD PTR APPLEBYTE
  C400                    CARD    EQU APPLEBYTE+0C400H ; A/D CARD
  9000                    TABSRT EQU APPLEBYTE+09000H ; DATA
                          ;
  0080                    DSEG 0080H ; AD8088 ON BOARD RAM
0000 0000                 TSTACK  DW 0
0002 0000                 LASTU   DW 0
0004 0000                 NEXFREE DW 0
0006 0000                 STATEPT DW 0
                          ;
  0090                    CSEG 0090H ; AD8088 SCRATCH RAM
  0096                    LEVEL EQU 150D
0000 B80010               MOV AX,1000H
0003 8EC0                 MOV ES,AX ; SET EXTRA SEGMENT TO APPLE
0005 B88000               MOV AX,0080H
0008 8ED8                 MOV DS,AX ; DATA SEGMENT AD8088 SCRATCH
000A 89260000             MOV TSTACK,SP
000E BB0005               MOV BX,9500H-9000H
0011 4B                   CLEAR: DEC BX
0012 26C687009000         MOV TABSRT[BX],0
0018 75F7           0011  JNZ CLEAR
001A C706040000000        MOV NEXFREE,0
0020 B80010               ALTST: MOV AX,1000H
0023 8EC0                 MOV ES,AX
0025 B88000               MOV AX,0080H
0028 8ED8                 MOV DS,AX
002A 89260000             MOV TSTACK,SP
002E C706060000000        MOV STATEPT,0
                          ; AWAIT TRIGGER
0034 26A000C4             SKP40: MOV AL,CARD
0038 3C96                 CMP AL,LEVEL
003A 72F8           0034  JB SKP40
003C 26A000C4             SKP41: MOV AL,CARD
0040 3C96                 CMP AL,LEVEL
0042 73F8           003C  JAE SKP41
                          ;
                          ;        H        L
                          ; AX SCRATCH SCRATCH
                          ; BX   Z^-1     Z^-2
                          ; CX    ZCR     LOOPS
                          ; DX HFZ^-1    FZ^-1
                          ;
                          ; SP HIGH DIFF
                          ; BP HIGH MAG
                          ; SI LOW  DIFF
                          ; DI LOW  MAG
                          ;
```

```
0044 B1FF          NEXT: MOV CL,0FFH ; INIT LOOP COUNT
0046 BB0000        MOV BX,0 ; ZERO OUT REGISTERS
0049 B500          MOV CH,0
004B BA0000        MOV DX,0
004E BC0000        MOV SP,0
0051 BD0000        MOV BP,0
0054 BE0000        MOV SI,0
0057 BF0000        MOV DI,0
005A B400          CONT: MOV AH,0
005C 26A000C4      MOV AL,CARD
0060 0412          ADD AL,12H ; CORRECT FOR CARD OFFSET
0062 7302      0066 JNC SKP0
0064 B0FF          MOV AL,0FFH
0066 2AD8          SKP0: SUB BL,AL ; DO HIGH FIRST
                   ; ASSUME NO SLOPE OVERLOAD
0068 2AF3          SUB DH,BL ; GET DIFF SIGNED
006A 7D02      006E JGE SKP1 ; GET ABS
006C F6DE          NEG DH
006E 86F0          SKP1: XCHG DH,AL
0070 03E0          ADD SP,AX ; INCREMENT HIGH DIFF
0072 86F0          XCHG DH,AL ; PUT ORG BACK IN AL
0074 8AF3          MOV DH,BL
0076 8ADF          MOV BL,BH
0078 8AF8          MOV BH,AL ; SET B UP FOR NEXT TIME
                   ; NOW DO FILT
007A 02C2          ADD  AL,DL ; ADD LAST  FILT
007C D0D8          RCR AL,1
007E 12C2          ADC AL,DL ; 0.75 COEFF
0080 D0D8          RCR AL,1 ; COMPLETE FILT
0082 8AE0          MOV AH,AL ; KEEP COPY OF FILT
0084 32E2          XOR AH,DL ; SEE IF ZERO CROSSED
0086 7902      008A JNS SKP2
0088 FEC5          INC CH ; INCREMENT IF YES
008A 2AD0          SKP2: SUB DL,AL ; GET DIFF, UNSIGNED
008C 7302      0090 JAE SKP3 ; ABS
008E F6DA          NEG DL
0090 B400          SKP3: MOV AH,0
0092 86D0          XCHG DL,AL ; SAVE FILT FOR NEXT
0094 03F0          ADD SI,AX ; INCREMENT LOW DIFF
                   ; FIX MAGS
0096 8AC6          MOV AL,DH
0098 3C00          CMP AL,0
009A 7D02      009E JGE SKP4
009C F6D8          NEG AL ; ABS
009E 03E8          SKP4: ADD BP,AX ; INCREMENT HIGH MAG
00A0 8AC2          MOV AL,DL
00A2 2C80          SUB AL,80H; SUB DC COMP, UNSIGNED
00A4 7302      00A8 JAE SKP5
00A6 F6D8          NEG AL
00A8 03F8          SKP5: ADD DI,AX ; INCR LOW MAG
00AA FEC9          DEC CL
```

```
00AC 75AC              005A JNZ CONT
                            ;
00AE BA5000                 MOV DX,50H
00B1 8BC6                   MOV AX,SI
00B3 F7E2                   MUL DX ; EFFECT OF /256 FOR MAG AVG
00B5 F7F7                   DIV DI
00B7 8BF8                   MOV DI,AX ; LOW MAJ F IN DI
00B9 BA1000                 MOV DX,10H
00BC 8BC4                   MOV AX,SP
00BE F7E2                   MUL DX
00C0 F7F5                   DIV BP
00C2 8BE8                   MOV BP,AX ; HIGH MAJ F IN BP
                            ; SCALE PARMS
00C4 8BC5                   MOV AX,BP
00C6 8AE8                   SKP11: MOV CH,AL
00C8 8BC7                   MOV AX,DI
00CA 8AD0                   SKP12: MOV DL,AL
                            ;
00CC 8B1E0400               MOV BX,NEXFREE
00D0 81FB0005               CMP BX,0500H ; EXCEED TOP PAGE
00D4 7203              00D9 JB SAFE
00D6 E92E00            0107 JMP OVERFL
                            ; LOW MAJ F ON EVEN ADDRESS
00D9 2688970090             SAFE: MOV TABSRT[BX],DL
00DE 43                     INC BX
                            ; HIGH MAJ F ON ODD ADDRESS
00DF 2688AF0090             MOV TABSRT[BX],CH
00E4 43                     INC BX
00E5 891E0400               MOV NEXFREE,BX ; SAVE FOR NEXT ROUND
                            ; END IS SIGNALED BY A ZERO VALUE
                            ;
00E9 26A000C4               TITRIG: MOV AL,CARD
00ED 3C96                   CMP AL,LEVEL
00EF 73F8              00E9 JAE TITRIG
00F1 BBE02E                 MOV BX,12000D ; TIMEOUT TIMER
00F4 26A000C4               TRG1: MOV AL, CARD
00F8 3C96                   CMP AL,LEVEL
00FA 7203              00FF JB SKP72
00FC E945FF            0044 JMP NEXT
00FF 4B                     SKP72: DEC BX
0100 75F2              00F4 JNE TRG1
0102 8B260000               MOV SP,TSTACK
0106 CB                     RETF
0107 C70606000000           OVERFL: MOV STATEPT,0
010D 8B260000               MOV SP,TSTACK
0111 CB                     RETF
                            END
```

APPENDIX B

TEMPLATE MATCHING PROGRAM

```
5  REM ! INTEGER DF,SB,X,Y,N,M,QL
10  HIMEM: 36000
20 D$ =  CHR$ (4)
30  DIM Q%(70,1),TM%(30,70,1),W$(30),TL%(30),CR(30)
35  REM  CALL TO $0090:0000
40  POKE 36001,0: POKE 36002,0: POKE 36003,144: POKE 36004,0
100  HOME :N = 0:W$(0) = "?"
110  PRINT
115  PRINT "ENTER WORD TO TRAIN OR BLANK TO END"
120  INPUT A$
130  IF A$ = "" THEN  GOTO 1000
140  PRINT "PRESS SPACES TO TRAIN OR RETURN TO END"
150  GET Q$
160  IF  ASC (Q$) = 13 THEN  GOTO 110
170  GOSUB 5000: REM  LOAD WORD TO Q,QL
180 N = N + 1:W$(N) = A$
210  FOR X = 1 TO QL
220 TM%(N,X,0) = Q%(X,0)
225 TM%(N,X,1) = Q%(X,1)
230  NEXT X
240 TL%(N) = QL
250  GOTO 150
1000  REM  RECOGNIZE ROUTINE
1010  HOME
1020  INPUT "PRESS RETURN AND ENTER A WORD";R$
1030  GOSUB 5000: REM  GET DATA
1060  REM  DO TEMPLATE TEST
1070  FOR Y = 1 TO N
1080 M = QL
1085 LN = QL
1090  IF TL%(Y) > M THEN M = TL%(Y)
1095  IF TL%(Y) < LN THEN LN = TL%(Y)
1100 SB = 0
1110  FOR X = 1 TO LN
1115 LE =  ABS (TM%(Y,X,1) - Q%(X,1))
1120 DF =  ABS (TM%(Y,X,0) - Q%(X,0)) + LE
1150 SB = SB + DF
1160  NEXT X
1170 CR(Y) = (SB + 10 * (M - LN)) / LN
1180  NEXT Y
1190  REM  PICK THE BEST
1200 BT = 100:Z = 0
1210  FOR Y = 1 TO N
1220  IF CR(Y) < BT THEN BT = CR(Y):Z = Y
1230  NEXT Y
1240  PRINT W$(Z)
1250  GOTO 1020
5000  REM  CALL TO 8088
5010  POKE 49326,161: POKE 49327,140: POKE 49312,255
```

```
5020  IF  PEEK (49312) < 128 THEN  GOTO 5020
5030 QL = 0
5040  PRINT
5050  FOR I = 36864 TO 38144 STEP 2
5070  IF  PEEK (I) = 0 THEN  GOTO 5150
5080 QL = QL + 1
5090 Q%(QL,0) =  PEEK (I)
5100 Q%(QL,1) =  PEEK (I + 1)
5140  NEXT I
5150  RETURN
```

VITA

Stephen Malone Killough was born in Wynne, Arkansas, on March 2, 1955. Presently living in Tennessee, he has also lived in Georgia and Florida.

Before attending The University of Tennessee, Knoxville, the author graduated from Georgia Tech with a Bachelor of Electrical Engineering degree. He is now with Union Carbide of Oak Ridge, Tennessee.