CS 127
MEMO AI-85

ASPECTS OF SPEECH RECOGNITION BY COMPUTER

BY

PIERRE VICENS

SPONSORED BY
ADVANCED RESEARCH PROJECTS AGENCY
ARPA ORDER NO. 457

APRIL 1969

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Aspects of Speech Recognition by Computer

A DISSERTATION

SUBMITTED TO THE COMPUTER SCIENCE DEPARTMENT

AND THE COMMITTEE ON THE GRADUATE DIVISION

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Pierre Vicens

April 1969

STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AI-85
CS 127

April 1969

ASPECTS OF SPEECH RECOGNITION BY COMPUTER

by

Pierre Vicens

ABSTRACT: [illegible] ~~This thesis~~ describes techniques and methodology which are useful in achieving close to real-time recognition of speech by a computer. To analyze connected speech utterances, any speech recognition system must perform the following processes: preprocessing, segmentation, segment classification, recognition of words, recognition of sentences. [illegible] ~~We~~ present implemented solutions to each of these problems which achieved accurate recognition in all the trial cases.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Department of Defense (SD-183).

ACKNOWLEDGEMENTS

I would like to acknowledge my personal indebtedness to Professor Raj Reddy, my thesis advisor. Some of the ideas presented in this thesis were proposed by him or are an extension of his work. In addition, I am greatly indebted to Professor John McCarthy whose interest and strong recommendation to the "Commissariat a l'Energie Atomique" made this work possible. I am also grateful to Professors William Miller, Edward Feigenbaum and James Beauchamp for their many valuable comments on the manuscript.

A special debt is owed to the Artificial Intelligence Project personnel for its support and interest in the research and for providing a stimulating atmosphere which makes research a pleasure. Amongst these, special mention should be made of Messrs. L. Earnest for his active interest and help at times of crises, R. Paul for his help in the design and implementation of the preprocessing hardware, K. Pingle and J. Singer who programmed the HAND-EYE System, L. Erman and C. Goodman for reading the initial version of the dissertation, R. Neely for providing many useful utility programs and all those whose voices were experimented on.

I wish to express my thanks to Mrs. Queenette Baur for her diligence and excellence in the typing of the manuscript.

This work is dedicated to my wife, Louise and my daughter, Cecile.

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense, (SD-183).

ASPECTS OF SPEECH RECOGNITION BY COMPUTER

ABSTRACT

This thesis describes techniques and methodology which are useful in achieving close to real-time recognition of speech by a computer. To analyze connected speech utterances, any speech recognition system must perform the following processes: preprocessing, segmentation, segment classification, recognition of words, recognition of sentences. We present implemented solutions to each of these problems which achieved accurate recognition in all the trial cases.

The preprocessing process involves the division of the speech spectrum into convenient frequency bands, and calculation of amplitude and zero-crossing parameters in each of these bands every 10 milliseconds. In the software simulation, two smoothing functions divide the speech spectrum into two frequency bands (above and below 1000 Hz). In the hardware implementation, the spectrum is divided into three bands using bandpass filters (i.e. 150-900 Hz, 900-2200 Hz, 2200-5000 Hz).

Utilizing the parameters generated by the preprocessing procedure, the segmentation process determines whether the characteristics of the sound are changing in time or are similar. Portions that possess similar parameters are grouped together to form sustained segments and portions that possess changing parameters form transitional segments, resulting in the segmentation of connected speech into parts approximately corresponding to phonemes.

The classification process assigns a phoneme-group label to each segment by looking at segment characteristics which are obtained by averaging the preprocessing parameters over the entire segment.

In learning mode, the sound description so generated is stored in a lexicon in a form suitable for fast retrieval. In recognition mode, heuristic procedures search the lexicon and build a list of probable candidates by considering rough features of the utterances. Then each candidate description is compared with the incoming message description and the candidate of best-match is selected. The comparisons are performed first by determining correspondences between segmental descriptions and then by evaluating similarity scores on the basis of the closeness of parameters for the corresponding segments.

The sentences of limited languages which are defined by a grammar are decoded first by obtaining a segmental description of the sentence and then by scanning the sentence description forward or backward looking for "known" (previously learned) words. At any step feedback from the grammar is used to eliminate from the matching process the syntactically incorrect word representations.

Some significant results presented in this disertation are:

- 98% correct recognition for a single-speaker list of 54 words in 2-3 seconds per word, after 4 training-rounds, above 92% correct recognition being already achieved after 1 training-round (tested for 2 speakers).
- 85% - 90% correct recognition for a list of 54 words recorded by 10 speakers in 9-12 seconds per word, after 9 training-rounds.
- 97% correct recognition for a single-speaker list of 70 French words in 2-3 seconds per word.

- 92% correct recognition for a single-speaker list of 561 words and short sentences in 16-17 seconds after 3 training-rounds. Decoding of 3-4 seconds long, syntactically structured sentences in 10-15 seconds.

The research described above leads us to the following conclusions:

- The fact that, using crude parameters, we were able to obtain satisfactory results indicates that it is not the type of preprocessing which matters, but rather the power of the subsequent algorithms.
- The present controversy about the best elementary unit to be used in the analysis of speech (phoneme, syllable, word, etc...) seems unwarranted. In this investigation we used all of them at various stages of the analysis.
- Accurate recognition of limited languages can be achieved even though an accurate phoneme-like classification is not available.
- Techniques of Artificial Intelligence, such as the reduction of search space by means of heuristics, appear to hold great promise for speech recognition.
- Attempts at building more powerful syntax-directed sentence analyzers are likely to be more fruitful than a great amount of effort spent in devising preprocessing techniques.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ALGORITHMS

LIST OF ILLUSTRATIONS

Chapter I
INTRODUCTION

Speech is, perhaps, the most extensively investigated of all the human perceptual and motor processes. For a long period of time, research in this area was aimed at speech synthesis and speech transmission. Recently, advances in technology and the availability of new machines able to deal with large amounts of data have made attempts at efficient speech analysis possible, and, as an extension, automatic speech recognition. First attempts at speech recognition by computer were restricted to the recognition of simple sounds, like vowels and digits, just as preliminary attempts at picture processing were restricted to the recognition of characters. However, approaches developed for the recognition of characters, such as the use of a metric in a multidimensional space partitioned by hyperplanes, could not be easily extended to the analysis of the complex sequence of sounds which are part of a spoken message. Here, the structure of the message and the interrelationships among the sounds of the message are the important factors.

Our approach to the speech recognition problem can be summarized as follows:

1. Development of procedures for the extraction of relevant parameters from the speech wave (preprocessing procedures).

2. Formulation of heuristic procedures to segment spoken messages, represented by the speech parameters previously extracted,into discrete parts,to classify those parts, and thereby create a description of the messages.

3. Formulation of heuristic procedures to match the message descriptions generated by the segmentation process with prestored (or "learned")representations.

4. Development of artificial languages specifically designed to simplify the problem of determining word boundaries and to resolve phonetic ambiguities in the analysis of long connected-speech utterances.

In this chapter some of the main problems associated with computer speech recognition are discussed and a model of a general purpose speech recognizer is presented. Then, the efforts of other researchers in this area of Artificial Intelligence are reviewed. Finally, the last section outlines the aim and scope of the present work and the methods and material used.

I-1. WHAT IS SOUND? HOW CAN ONE SOUND BE DISTINGUISHED FROM ANOTHER?

These questions must be answered before we can effectively recognize speech. Phoneticians have provided us with several terms for describing speech sounds: morphemes, syllables, and phonemes. Unfortunately their idealized classifications, based on articulatory, acoustical or perceptual properties of sounds are more qualitative than quantitative and are meant for use by humans rather than by machines. To further complicate the problem, a phoneme, considered to be the smallest perceptual unit of a language, may have different allophones which do not necessarily present similar acoustic properties, nor are the acoustic characteristics always invariant within a given phoneme. Connected speech is created by a continuous motion of the vocal apparatus from sound to sound, so that the vocal tract dwells only momentarily in a state appropriate to a given phoneme. Furthermore, a phoneme is a relative concept dependent on the language (e.g., in Japanese, the words <u>raw</u> and <u>law</u> would be treated as the same word because /r/ and /l/ are different allophones of the same phoneme. In Hawaiian, <u>pack</u> and <u>back</u> would be considered the same word because /p/ and /b/ are different allophones of the same phoneme). These difficulties with the definition of a phoneme suggest that it might be desirable to consider a different unit of sound more amenable for machine recognition than the phoneme.

One can define a <u>sound</u> on a purely acoustic basis:

..As a sustained segment in which the acoustic characteristics of the sound remain relatively constant (i.e., vowels, nasals, fricatives, silences, etc.).

..As a transitional segment in which the acoustic characteristics vary with time (any segment which is not a sustained segment).

Note that, in such a scheme, an ideal phoneme may be spread over several segments, or two phonemes may be grouped into one segment.

Likewise, the conventional definition of a "word" is not appropriate to computer recognition of speech. In all the languages of some interest words are determined by the written form of the language and not by the acoustical properties of the spoken form. In order to break up an utterance into independently recognized words, we must provide a means of defining acoustical boundaries between them. For example, on a purely acoustic basis we can define a syllable to be that part of the speech signal which lies between two silences or between a fricative and a silence. A word then is formed by concatenating one or several syllables. Under this hypothesis, /HOW ARE YOU/ is considered a word of one syllable and /RESCAN/ a word of two syllables.

Based on these definitions an efficient speech recognizer should have the following characteristics:

..It must be able to determine the boundaries of sounds and to classify those sounds as belonging to some categories.

..It must be able to determine the boundaries of words composed of several sounds and to recognize those words.

In this model, we are not limited to specific sounds or words. A sound may correspond to a phoneme or several phonemes, and a word may correspond to several conventional words or only to a syllable.

I-2. WHY ARE SPEECH RECOGNITION SYSTEMS INTERESTING?

There are several motivating factors for attempting to provide speech input to computers. Although many people are intuitively aware of the advantages of such facility, they deserve to be explicitly stated:

1. Universality: speech is the most universal and natural mode of communication among men.

2. Fast Data Transfer Rate: Statistics have shown that in normal speech an average of 4 to 6 words (12.5 phonemes) are uttered each second. Therefore, if man-machine communication is our main concern, this medium is faster and better than a teletype.

3. No need to be close to the Computer: The fact that one does not need to be close to the computer to operate it, and the simplicity of the required remote station (e.g., a telephone handset) give more convenience to such a system.

4. Versatile Motor Process: Adequately programmed, it may provide its user with additional motor processes and additional effectors besides the usual hands and feet. For example, in space exploration, a voice-controlled guidance system could help the pilot in the execution of all the simultaneous tasks he must perform.

Unfortunately, at present some disadvantages tend to inhibit the development of practical speech recognition systems:

1. Prohibitive Cost: It is the main disadvantage of such a system. For example, to recognize the users' commands in a time-sharing environment, a large scale computer would have to be used to perform the necessary analysis. However, as research continues, more problems are being solved, and the price of hardware is constantly decreasing, so that a $50,000

"Speech Reader" station is conceivable in the future. For many situations, such a station would be more convenient than a card-reader, and faster than a teletype.

2. Handling of Different Voices: To be usable, a "Speech Reader" should be able to deal with different accents and different voices. An obvious solution to this problem is to train the machine with several speakers. However, the limited memory available prevents from using a large number of different voices. A better approach seems to be the use of transformations on the speech input which normalize it with regard to the speaker characteristics before the recognition process. Some of these transformations are being studied, and hopefully, solutions will be found in the next few years.

3. Handling of Natural Spoken Languages: The well known difficulties encountered are those which prevent the use of natural languages when dealing with computers. The problem is even more severe for speech recognition, since the spoken form of a language is, in general, less structured grammatically than its written form. This area represents an active field of research in the theory of grammar and semantics. Although researchers usually deal with the written form of natural languages, many of the results obtained will be directly usable by speech recognizers. Today, in the absence of good solutions this problem can be circumvented by the use of rigorous syntaxes, in the same manner that highly-structured programming languages were adopted to conveniently program computers.

This thesis describes working systems covering various aspects of speech recognition. We do not pretend to have solved all the problems involved in the recognition of connected speech. We have solved some of them in restricted environments. Hopefully, in the near future, our solutions will be improved and more problems will be solved, thus increasing the field of application of speech recognition systems.

1-3. PREVIOUS ATTEMPTS AT SPEECH RECOGNITION BY MACHINES.

Many attempts have been made to recognize speech. In this investigation, we shall confine ourselves to the recent research in the areas which are directly relevant to our work; namely the areas of recognition of isolated words (or messages), and utilization of linguistic constraints to decode connected speech utterances defined as a sequence of segments.

We shall review the attempts made by Davis, Biddulph and Balashek (1952); Reddy (1967); Gold (1966); and more recently Bobrow and Klatt (1968); who attacked the problem of recognizing speech by the analysis of real data as recorded by a microphone (or a tape recorder). We shall also discuss the research performed by Reddy and Robinson (1968) and Alter (1968) who attempted to recognize connected speech by applying linguistic constraints to the analysis of hypothetic string of phonemes which might be produced by an acoustic recognizer.

David, Biddulph and Balashek (1952) attempted to recognize telephone quality digits spoken at normal speech rates. The speech spectrum was divided into two frequency bands, one below and one above 900 Hz. Axis-crossing (zero-crossings) counts were then made on each band energy to determine the frequency of the maximum syllabic rate energy within each band. A two dimensional frequency protrayal was built from the preceding analysis; following this, a comparison was performed with each of ten standard digit patterns and the digit of best match selected. Such a procedure cannot be extended to the recognition of large vocabularies or to the analysis of long utterances. Nevertheless, the technique of separating the speech spectrum into frequency bands and counting the zero-crossings in each band can be extended for use as a parameter extraction method in a more sophisticated system.

The specific aim of Reddy (1967) was to produce a phonemic transcription of a connected speech utterance which was readable and bore a satisfactory resemblance to what was said. The original data to his phoneme recognition system was the waveform digitized by an analog-to-digital converter sampled every 50 μs (20,000 Hz). This speech wave was divided into a succession of 10 millisecond segments. These minimal segments were then grouped together to form larger segments approximately corresponding to phonemes. Once the segmentation and classification into phoneme groups were performed, Fourier spectrum analysis was utilized to further classify the segments. The immediate goal of this work was to obtain a phoneme string from a connected speech utterance and was based on single speaker data. The present work extends the heuristics utilized so that they become usable in a multispeaker environment and associates the segment string classified into phoneme groups with words and sentences of limited languages.

Gold (1966) investigated the problem of recognizing words spoken by different speakers. Each word was analyzed by a spectrum analyzer, a pitch detector and a voicing detector. Fifteen features were extracted by segmenting the sound, detecting the stressed vowel, and making measurements on the stressed vowel and its neighboring segments. These measurements were filed for 540 words uttered by ten speakers. A decision algorithm was devised while analysing and storing these data. Then, during a second pass, all of the 540 words were passed through this algorithm and the results tabulated, each speaker's word being compared to the words said by all of the other speakers. The results obtained were about the same level of accuracy as our results are, however, since he used all of the other speaker data for the recognition of one, we are not able to effectively compare the two systems. We suspect that because he used only a part of the utterances (stressed vowels and neighboring segments) to determine the similarity between words, his system is less effective for vocabularies which are not well-balanced (i.e., when they contain words with the same stressed vowel).

Bobrow and Klatt (1968) have based their limited speech recognition system (LISPER) on the comparisons of distinctive features extracted directly from the outputs of 19 bandpass filters. Their original data was composed of several word lists recorded by two speakers in a very quiet room (S/N ratio >35 db). Although LISPER uses a different approach than ours, it is of interest because it permits us to directly compare the performances of the two systems. We could run under our system the two word lists (recorded by Ken Stevens and Carl Williams, which Dr. Bobrow graciously provided us with), which Bobrow and Klatt used to obtain their statistical results. Comparison of the results indicates that our approach is slightly better for single speaker lists when the vocabulary is phonetically well-balanced, and much better when the number of possible confusions is increased, this being true even in a noisy environment (S/N ratio $\approx$ 15 db). The main shortcoming of their model appears to be that since the utterances are not segmented, their work cannot easily be extended to the analysis of long sentences in which a division into small components (phonemes or words) is necessary. Lacking timing information, we are unable to say how effective their search and classification strategy was and whether or not the recognition was done in close-to-real time.

As far as we can determine, both Gold and Bobrow and Klatt use a maximum-likelihood type of classification system which calculates similarity measures for all the candidates in the lexicon. Any algorithm which does not effectively eliminate most of the candidates before computing similarity measures cannot be used in dealing with large vocabularies. By contrast most of our time and effort has been spent in devising efficient heuristics for the reduction of the candidate space.

Reddy and Robinson (1968) and Alter (1968) have investigated the problem of recognition of long sentences represented by sequences of phoneme-like segments. In both cases, the input to the programs was a sequence of typed phonemes. To solve ambiguities arising from similar words or from errors in the input string, both used a dictionary of allowed symbols and linguistic information (Fortran grammar in Backus-Naur form for Alter, a simplified English grammar for Reddy and Robinson). While they were useful ideas, they were not tested with actual speech input. Our experience suggests that these researchers may have had to modify their model substantially before they will be able to handle word boundary problems. If they had, we believe that they would have discovered that neither English nor Fortran are well suited for man-machine voice communication.

I-4. AIM AND SCOPE OF THE INVESTIGATION:

The specific goal of this research was to build a high accuracy, limited vocabulary, recognition system working in real-time, or close to real-time, and to use it in the analysis of connected speech utterances of highly-restricted languages. The vocabulary of the recognizer, limited only by the memory size of the computer, can be as large as 1000 messages of up to 1.5 seconds duration on our machine.

This dissertation, describes a message recognizer and its use in a voice-controlled visual feedback manipulator and a voice-controlled desk calculator. For each of the two examples a finite-stage grammar (manipulator) or a linear grammar (desk calculator)is used to assist in the recognition of connected speech utterances and to resolve phonetic ambiguities. Although the system is not restricted to any particular speaker, it gives better results for the speakers with whose voices it is trained. The statistical results given in the last chapters are based on data obtained using several different speakers. In all the phases of the research, no attempt was made to artificially reduce the noise level of the room since that would not be the normal mode of man-machine communication.

Methods and Material

Most of the research has been done on the PDP-6 computer available at the Stanford Artificial Intelligence Project. A general diagram of the machine can be seen in Figure I-1. Recently, a PDP-10 processor was added to the existing system and both machines were used to implement the desk calculator. Since real-time operation was the main goal of this last application, a large amount of computation power was required. As Fortran IV

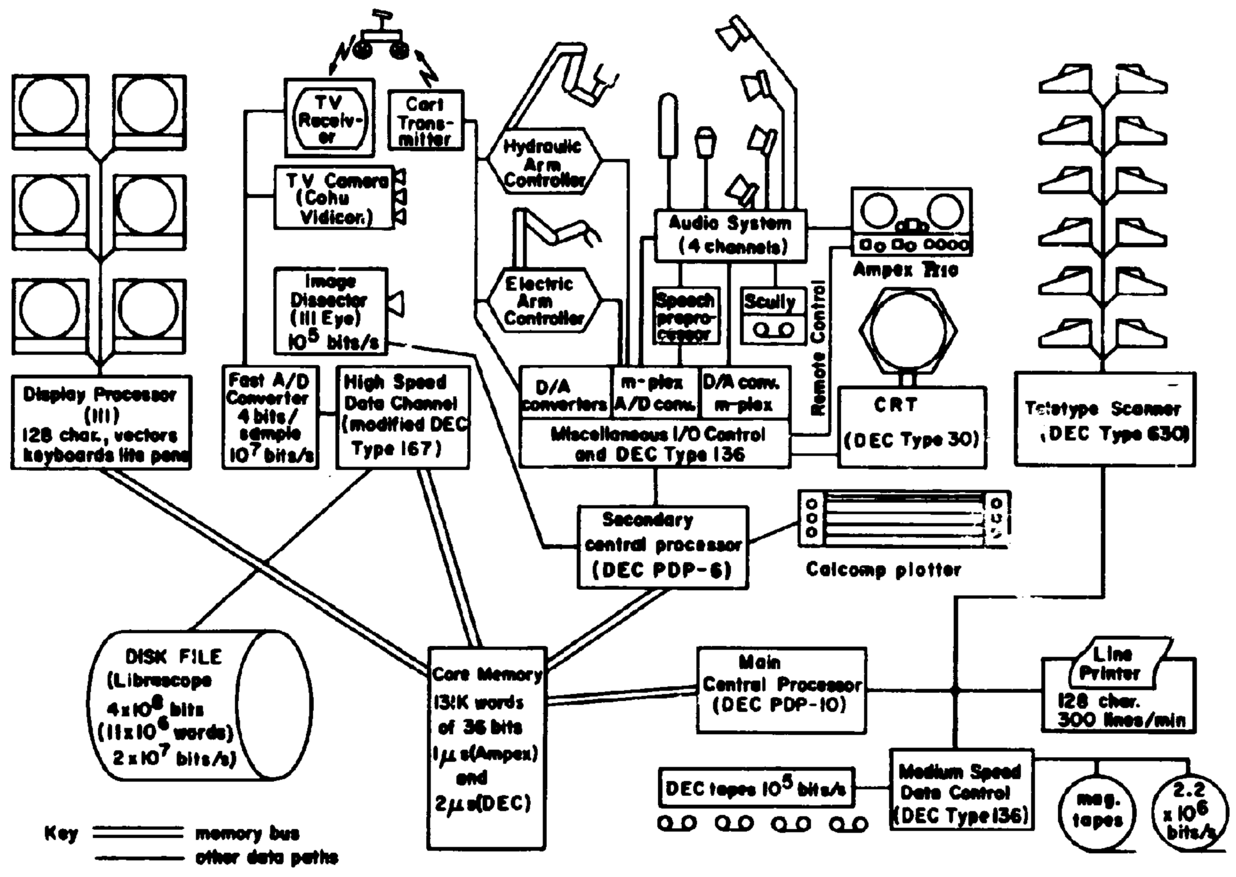

TV Receiver
Cart Transmitter
TV Camera (Cohu Vidicon)
Image Dissector (III Eye) 10^5 bits/s
Hydraulic Arm Controller
Electric Arm Controller
Audio System (4 channels)
Ampex Trio
Speech preprocessor
Scully
Remote Control
CRT (DEC Type 30)
Display Processor (III) 128 char., vectors keyboards lite pens
Fast A/D Converter 4 bits/sample 10^7 bits/s
High Speed Data Channel (modified DEC Type 167)
D/A converters
m-plex A/D conv.
D/A conv. m-plex
Miscellaneous I/O Control and DEC Type 136
Teletype Scanner (DEC Type 630)
Secondary central processor (DEC PDP-6)
Calcomp plotter
DISK FILE (Librascope 4×10^8 bits (11×10^6 words) 2×10^7 bits/s)
Core Memory 131K words of 36 bits 1μs (Ampex) and 2μs (DEC)
Main Central Processor (DEC PDP-10)
Line Printer 128 char. 300 lines/min
DEC tapes 10^5 bits/s
Medium Speed Data Control (DEC Type 136)
mag. tapes
2.2×10^6 bits/s
Key
memory bus
other data paths

FIGURE I-1 THE STANFORD ARTIFICIAL INTELLIGENCE PROJECT COMPUTER SYSTEM

was the only algebraic language originally available on the PDP-6 computer, this language was chosen to program the large amounts of arithmetic computation involved in all the procedures. The stored word lexicon, requiring a complicated list structure with two independent sets of pointers for each stored message representation, is manipulated by means of Fortran compatible machine language subroutines. Likewise, the necessary Input/Output operations and the display package were implemented in the more flexible machine language. Later, the often utilized parts of the Fortran portion of the system were recoded in assembly language. Finally, for the desk calculator in which speed was one of the main concerns, we simplified some of the algorithms and coded then entirely in machine language. Nevertheless, in the present dissertation, all of the described algorithms are presented in ALGOL notation.

Organization

This dissertation is organized in the order a speech recognition system implementation should follow. Except for the Desk-Calculator (Chapter VI), which exhibits a real-time application similar to the Hand-Eye-Ear program (Chapter V), each chapter is the logical continuation of the previous ones.

In Chapter II we describe a procedure and its hardware implementation for the extraction of significant parameters from speech. Such a preprocessing procedure is necessary to reduce the large amount of data contained in a speech utterance.

In Chapter III we present a segmentation procedure of the speech waveform using the parameters obtained through the preprocessing procedure. Each created segment is then classified as belonging to one of the broad categories VOWEL, FRICS, BURST, STOP, NASAL and CONST. The segmentation procedure is general and can be used to recognize acoustic sound boundaries with good accuracy. The simple classification algorithm presented was found to be sufficient for our purpose, since average parameters are kept along with the label in order to characterize a segment.

In Chapter IV, we describe the system EARS (Effective Analyzer and Recognizer of Speech) which is able to learn and recognize as many as 1000 words and messages. Several heuristics to reduce the candidate space and a solution to the segment synchronization problem are given. To match the segmental parameters of an input utterance against known parameters of the same phrase, one must determine correspondence between the segments of the two utterances. The synchronization procedure first maps vowel to vowel and fricative to fricative. The few unmapped segments between any two pairs are then mapped on the basis of similarity of segmental parameters. A global similarity evaluation is performed utilizing the mapped segments and a best-match type comparison chooses the response. Statistical results are given along with some evaluation of the principal heuristics used.

In Chapter V, we describe the utilization of the word recognizer in the analysis of connected speech utterances: the HAND-EYE-EAR program. Unlike the previous example, in which the utterances were recognized as a single unit, here the commands are analyzed by recognizing individual words within the sentences. At any step, feedback from a finite state grammar is utilized to eliminate syntactically incorrect word representations from the search process. Statistical results for several sentences uttered by different speakers are given.

In Chapter VI we describe a real-time sentence analyzer. The sentences are desk-calculator statements, the vocabulary consisting of some 35 words. The segmentation of the uttered commands is executed by the PDP-6 computer while the experimenter is talking. The decoding of the command and its execution is done in the PDP-10 processor using a left-to-right parsing of the statement. The grammar used is a simple linear grammar which, by look-ahead, reduces the search while recognizing the words and interprets the statements while executing the commands.

Chapter II
PREPROCESSING FOR SPEECH ANALYSIS

II-1. INTRODUCTION

The average information rate of the human voice signal has been estimated to be as high as 300,000 bits per second. Various preprocessing techniques have been proposed to reduce this huge mass of information to a more manageable level, e.g., spectrum analysis, approximation by orthogonal functions, zero-crossing analysis, etc... This chapter describes two procedures for preprocessing speech which are extensions of the zero-crossing analysis technique.

One of the earliest attempts at speech analysis using the frequency spectrum of the voice was made by Dudley (1939) with his invention of the vocoder. The fundamental frequency amplitude and the short time amplitude spectrum for ten discrete frequency bands were extracted from the speech signal. The circuit consisted of a frequency discriminator to obtain the fundamental frequency, bandpass filters, rectifiers and low pass filters for the other frequencies. Since the original development of the vocoder, many different versions and variations of this scheme have been constructed. Flanagan (1965) has thoroughly reviewed most of the techniques which have stemmed from the original vocoder.

Two general methods for representing signal waveforms by orthogonal functions have been described in the literature. Mathews, Miller and David (1961) used Fourier series expansion in a "pitch-synchronous analysis of voiced sounds". Dolansky (1960) performed a similar analysis, but instead used orthogonalized, exponential functions. These methods are useful for digital processing of the signals, but they can be time-consuming.

Peterson (1951) was one of the first investigators to use zero-crossing information to analyze speech. His idea was to take the average density of zero-crossings of the speech wave and of its time derivative as approximations to the first and second formants, respectively. A number of refinements of this zero-crossing technique have been made. Munson and Montgomery (1950), David, Biddulph and Balashek (1952) pre-filtered the speech signal into frequency ranges appropriate to individual formants. The zero-crossing rate and the amplitude were then measured in each of the bands. At Stanford University, Reddy (1966), in an attempt to recognize speech by computer, used the amplitudes and zero-crossings of digitized speech waves to segment speech utterances and to classify segments into phoneme groups. He primarily used the amplitude information to group acoustically similar 10 ms segments. Because of their high variability, zero-crossings were used only as a secondary parameter.

In the first parameter extraction procedure presented, the variability of the parameters is reduced by two computer-coded smoothing functions. These functions are equivalent to low pass and high pass filters with approximately 1000 Hz cut-off frequencies. Amplitude and zero-crossing parameters are then based on the output of each of these equivalent filters. The parameters extracted were not completely satisfactory, due to the low reliability of correlating high frequency components obtained by differencing techniques with the appropriate speech sounds. However, the results obtained from the subsequent processes were acceptable, e.g., 95 percent correct for the segmentation procedure (Reddy and Vicens, 1968).

In order to obtain a more accurate representation of the speech signal, it was decided to divide the speech spectrum into three frequency bands (150 Hz-900 Hz, 900Hz-2200 Hz, 2200 Hz-5000 Hz) and to determine amplitude and zero crossing parameters in these bands.

Since close to real-time recognition was desired, it was also decided to realize this new preprocessor in hardware form. This resulted in a low data rate device (3600 bits/sec) using bandpass filters and analog circuitry which provides the input for an analog to digital converter.

These two preprocessing procedures were developed on our time-shared PDP-6 Computer. Audio input for the first procedure consists of a microphone connected to an A-D converter via an amplifier. The speech signal is sampled at 20,000 samples per second and digitized to 9 bits (40 db dynamic range). With the second method, six parameters, which are accumulated by the analog speech preprocessor, are digitized to 6 bits accuracy every 10 ms. (600 samples per second).

II-2. THE SOFTWARE SPEECH PREPROCESSOR

Since significant changes do not usually occur within any 10 ms of speech, we use a 10 ms interval as our basic unit, and call it a minimal segment. Let an arbitrary wave within a minimal segment be represented by a discrete function f_i, whose values are the ordinates of this wave at n equidistant points. The amplitude of the wave on the minimal segment is then defined to be $\underset{i=1,n}{\mathrm{Max}}\, f_i - \underset{i=1,n}{\mathrm{Min}}\, f_i$. The zero-crossings of the wave on the minimal segment is the number of sign changes of f_i.

After investigating several possible parameters, we found the zero-crossing and amplitude parameters of a smoothed speech wave to be less variable than the original wave. In addition, we found it desirable to have a measure of the high frequency components present in the speech wave. Therefore, two other parameters were obtained by subtracting the smooth wave from the original and measuring the amplitude and zero-crossings of the residual wave.

II-2-1. The Smoothing Function

The amplitude and zero-crossing parameters just defined for the smooth and residual waves are very sensitive to the choice of the smoothing function.

The simplest function one can use is the regular averaging function defined by $y_p = \frac{1}{n} \sum_{j=\alpha+1}^{j=\alpha+n} x_j$. This function was tried and was found not to be sufficiently accurate with respect to the residual wave computation. After a mathematical study of the problem and a new try, the function $y_p = \frac{(1+\epsilon)}{2q+1} \sum_{j=p-q}^{j=p+q} x_j$ was finally chosen. It will be shown that ϵ depends only on q and on the frequency of the wave, but not on p . The following paragraph describes the mathematical approach taken to solve this problem and evaluates the value of ϵ.

Let us now suppose that the original wave is a sine wave. Then the wave obtained from the regular averaging process, $\left(\frac{1}{n}\sum_{j=\alpha+1}^{j=\alpha+n} x_j\right)$, is always smaller in amplitude than the original wave, and the higher the frequency of the original wave is, the smaller the resultant output wave becomes. We shall show that for a sine wave, the original wave: x_p, and the smoothed wave: $y_p = \frac{1}{2q+1}\sum_{j=p-q}^{j=p+q} x_j$, are related by $y_p = (1-\epsilon)\, x_p$ where ϵ is independent of p and has a significant value.

Let $x_n = A \sin(\omega t_o n)$ be our original wave. ω represents the radian frequency of the sine wave, and t_o the sampling period of the analog-to-digital converter ($t_o = 100\ \mu s$).

Let the sequence y_p be defined by the averaging process:

$$y_p = \frac{1}{2q+1}\sum_{j=p-q}^{j=p+q} x_j = \frac{1}{2q+1}\sum_{j=p-q}^{j=p+q} A \sin(\omega t_o j).$$

In order to evaluate this term, let us introduce the sequence $z_p = \frac{1}{2q+1}\sum_{j=p-q}^{j=p+q} A \cos(\omega t_o j)$ and the complex sequence

$U_p = z_p + i\, y_p = \frac{A}{2q+1}\sum_{j=p-q}^{j=p+q} e^{i\omega t_o j}$ which is a geometric series.

Then $U_p = \frac{A}{2q+1}\ \frac{e^{i\omega t_o (p-q)} - e^{i\omega t_o (p+q+1)}}{1 - e^{i\omega t_o}}$,

and $U_p = \frac{A}{2q+1}\ e^{i\omega t_o p}\ \frac{e^{-i\omega t_o q} - e^{i\omega t_o (q+1)}}{1 - e^{i\omega t_o}}$,

rationalizing $U_p = \frac{A}{2q+1}\ e^{i\omega t_o p}\ \frac{(e^{-i\omega t_o q} - e^{i\omega t_o (q+1)})\ (1 - e^{-i\omega t_o})}{(1-e^{i\omega t_o})\ (1 - e^{-i\omega t_o})}$

so $U_p = \frac{A}{2q+1} e^{i\omega t_o p} \frac{\left(e^{-i\omega t_o q} + e^{i\omega t_o q} - e^{i\omega t_o (q+1)} - e^{i\omega t_o (q+1)}\right)}{2 - e^{i\omega t_o} - e^{-i\omega t_o}}$

Applying now the formulas: $e^{i\beta} + e^{-i\beta} = 2\cos\beta$, $e^{i\beta} - e^{-i\beta} = 2i\sin\beta$,

we obtain:

$$U_p = \frac{A}{2q+1} e^{i\omega t_o p} \frac{\cos\omega t_o q - \cos\omega t_o (q+1)}{1 - \cos\omega t_o} = \frac{A}{2q+1} e^{i\omega t_o p} \underline{\frac{\sin \omega t_o \frac{2q+1}{2}}{\sin \frac{\omega t_o}{2}}}$$

In the preceding equation, the underlined quantity is real, therefore we can immediately deduce the value of y_p, the imaginary part of the complex sequence U_p:

$$y_p = \frac{A}{2q+1} \frac{\sin \omega t_o \frac{2q+1}{2}}{\sin \frac{\omega t_o}{2}} \sin \omega t_o p.$$

If the term $\omega t_o q$ is small, we can expand the sines in a taylor series:

$$\sin\beta = \beta - \frac{\beta^3}{3!} + O(\beta^5)$$

$$y_p = \frac{A}{2q+1} \frac{(2q+1)\frac{\omega to}{2} - \frac{(2q+1)^3}{3!}\left(\frac{\omega to}{2}\right)^3 + O\left(\left(\frac{\omega toq}{2}\right)^5\right)}{\frac{\omega t_o}{2} - \frac{1}{3!}\left(\frac{\omega t_o}{2}\right)^3 + O\left(\left(\frac{\omega t_o}{2}\right)^5\right)} \sin \omega t_o p$$

$$y_p = A \frac{1 - \frac{(2q+1)^2}{3!}\left(\frac{\omega t_o}{2}\right)^2 + O\left(\left(\frac{\omega t_o q}{2}\right)^4\right)}{1 - \frac{1}{3!}\left(\frac{\omega t_o}{2}\right)^2 + O\left(\left(\frac{\omega t_o}{2}\right)^4\right)} \sin \omega t_o p$$

$$y_p = A\left(1 - \frac{q^2+q}{6}\omega^2 t_o^2 + O\left(\left(\omega t_o\right)^4\right)\right) \sin \omega t_o p$$

$$y_p \approx \left(1 - \frac{q^2+q}{6}\omega^2 t_o^2\right) x_p$$

As noted previously, the term $\epsilon = \frac{q^2+q}{6} \omega^2 t_o^2$ depends on the frequency of the wave (ω) and on the A-D sampling period t_o, but not on p (i.e., not on the position of the points on the original wave).

Let us now substitute some numerical values in the formula to obtain an approximation to ϵ. Assuming that q = 2 (we average over 5 points), t_o = 100 μs (corresponding to 10,000 samples per second), ω = 1000π (the frequency of the sine wave is 500 Hz), we obtain:

$\omega t_o = 1000\pi \times 10^{-4} = 0.1\pi$, which is small enough to justify the truncated Taylor expansion of the sine function.

Then $\epsilon = \frac{4+2}{6} (0.1\pi)^2 \approx 0.1$

The preceding remarks on the smoothing of a sine wave led us to use, on the speech wave, the modified averaging function

$$y_p = \frac{k}{5} \sum_{j=p-2}^{j=p+2} x_j \quad \text{where } k = \frac{1}{1-\epsilon} \approx 1+\epsilon .$$

ϵ is computed for each minimal segment using the smoothed value of the zero-crossings of the preceding 10 ms minimal segment: N . The value of ϵ with respect to N is given by

$$\epsilon = \frac{4+2}{6} \pi^2 N^2 10^{-4} \approx 10^{-3} N^2$$

From the engineering point of view, this corrective factor shifts the 0 db level of the filter as a function of the fundamental frequency of the wave. In other words, our smoothing function is equivalent to a low pass filter with variable gain depending on the fundamental frequency of the wave.

II-2-2. The Algorithm

Given the data rate of the analog-to-digital converter (20,000 samples/second), the maximum frequency of the digitized signal is restricted to 10,000 Hz. A primary averaging over two points diminishes the maximum frequency from 10,000 Hz to 5,000 Hz. A secondary averaging over five points with the use of the corrective factor just described results in the smooth wave on the minimal segment. Subtracting the smooth wave from the wave obtained from the primary averaging yields the residual wave on the minimal segment; the actual averaging computation is carried out using five "ring" registers to store the current and the four preceding values of the original wave and one register for their sum. The smooth wave is obtained by one addition, one subtraction, one multiplication (corrective term) and one division. The residual wave is obtained by one subtraction: the central point x_p (stored in the central of the five ring registers) minus the value obtained for the smooth wave. Zero-crossing and amplitude parameters are then computed for each of the two waves and stored for future utilization.

As speed was one of the main goals of this algorithm, it was written in machine language. The Algol version given next is only an equivalent and was never used on the machine (Algorithm 1).

II-2-3. Conclusions

Displays of these four parameters, for different messages can be seen in Figure II-1. The zero-crossings of the smooth waveform provide an estimate for the dominant frequency under 1000 Hz, which is usually the Formant 1 frequency. The zero-crossings of the residual wave provide an estimate for Formant 2, except for fricatives where it represents the dominant frequency over 1000 Hz.

```
PROCEDURE PREPROCESS ;

COMMENT SOFTWARE  SPEECH PREPROCESSOR ;


BEGIN
          INTEGER AMPLRESIDUALOLD,AMPLSMOOTHOLD,AMPLMAX,
                  AMPLSMOOTHMAX,AMPLSMOOTHMIN,AMPLRESIDUALMAX,
                  AMPLRESIDUALMIN,ZRXSMOOTHNB,ZRXRESIDUALNB,
                  WAVEFORM,STOREAMP[0:4],SUM,EPSILON,
                  AMPLSMOOTHNEW,AMPLRESIDUALNEW,
                  INDEX1,INDEX2,INDEX3,INDEX4;
          COMMENT
                  ARRAYS TO STORE THE COMPUTED RESULTS
                  DEFINED AS GLOBAL ARRAYS IN THE MAIN PROGRAM

          INTEGER ARRAY STORAMPSMOOTH[1:150],STORAMPRESO[1:150],
                        STORZRXSMOOTH[1:150],STORZRXRESO[1:150];


          INTEGER PROCEDURE NEWSAMPLE ;
          BEGIN   COMMENT WILL GIVE THE NEXT SAMPLE FROM THE
                           RAW SPEECH WAVEFORM ;
          END;

          INTEGER PROCEDURE SIGNCHANGE(VAL1,VAL2);
                  INTEGER VAL1,VAL2;
                  IF VAL1*VAL2 < 0 THEN  2
                                   ELSE IF  VAL1*VAL2 = 0 THEN 1
                                                          ELSE 0;

          INTEGER PROCEDURE MIN(VAL1,VAL2) ;
                  INTEGER VAL1,VAL2 ;
                  IF  VAL1 ≤ VAL2 THEN VAL1
                                  ELSE VAL2 ;

          INTEGER PROCEDURE MAX(VAL1,VAL2) ;
                  INTEGER VAL1,VAL2 ;
                  IF  VAL1 ≥ VAL2 THEN VAL1
                                  ELSE VAL2 ;


  COMMENT THE FOLLOWING PROGRAM WILL EXTRACT FOUR PARAMETERS FROM
          THE SOUND WAVE : ZERO-CROSSINGS AND AMPLITUDES OF A SMOOTH
          WAVE AND OF A RESIDUAL WAVE FOR 150 10 MS MINIMAL SEGMENTS ;


          SUM := AMPLRESIDUALOLD := AMPLSMOOTHOLD := AMPMAX := 0;
          FOR INDEX3 := 0 STEP 1 UNTIL 4 DO STOREAMP[INDEX3] := 0;
          INDEX3 := EPSILON := 0 ; INDEX4 := 0 ;
```

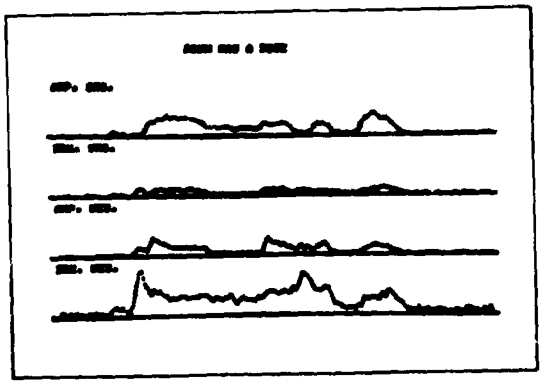
Algorithm 1. Preprocessing Procedure .

```
    FOR INDEX1 := 1 STEP 1 UNTIL 150 DO
        BEGIN

COMMENT   SMOOTH  WAVE  COMPUTATION ;

        AMPLSMOOTHMAX := AMPLRESIDUALMAX := 0 ;
        AMPLSMOOTHMIN := AMPLRESIDUALMIN := 1000;
        ZRXSMOOTHNB := ZRXRESIDUALNB := 0 ;
        FOR INDEX2 := 1 STEP 1 UNTIL 100 DO
            BEGIN
            WAVEFORM := (NEWSAMPLE+NEWSAMPLE)/2 ;
            SUM := SUM+WAVEFORM-STOREAMP[INDEX3] ;
            STOREAMP[INDEX3] := WAVEFORM ;
            IF INDEX3 < 4 THEN INDEX3 := INDEX3+1 ;
                          ELSE INDEX3 := 0 ;
            AMPLSMOOTHNEW := SUM*(1000+EPSILON)/5000 ;
            ZRXSMOOTHNB := ZRXSMOOTHNB+SIGNCHANGE(
                                 AMPLSMOOTHNEW,AMPLSMOOTHOLD) ;
            AMPLSMOOTHMAX := MAX(AMPLSMOOTHMAX,AMPLSMOOTHNEW) ;
            AMPLSMOOTHMIN := MIN(AMPLSMOOTHMIN,AMPLSMOOTHNEW) ;

COMMENT  NOW RESIDUAL WAVE COMPUTATION ;

            AMPLRESIDUALNEW := STOREAMP[INDEX4]-AMPLSMOOTHNEW ;
            IF INDEX4 < 4 THEN INDEX4 := INDEX4+1
                          ELSE INDEX4 := 0 ;
            ZRXRESIDUALNB := ZRXRESIDUALNB+SIGNCHANGE(
                              AMPLRESIDUALNEW,AMPLRESIDUALOLD) ;
            AMPLRESIDUAL := MAX(AMPLRESIDUALMAX,AMPLRESIDUALNEW) ;
            AMPLRESIDUALMIN := MIN(AMPLRESIDUALMIN,AMPLRESIDUALNEW);
            END;

COMMENT STORE ALL THE RESULTS IN THE CORRESPONDING ARRAYS ;

        ZRXSMOOTHNB := ZRXSMOOTHNB/2 ;
        STORZRXSMOOTH[INDEX1] := ZRXSMOOTHNB ;
        STORZRXRESD[INDEX1] := ZRXRESIDUALNB/2 ;
        STORAMPSMOOTH[INDEX1] := AMPLSMOOTHMAX-AMPLSMOOTHMIN ;
        STORAMPRESD[INDEX1] := AMPLRESIDUALMAX-AMPLRESIDUALMIN ;
        AMPMAX := MAX(AMPMAX,STORAMPSMOOTH[INDEX1]) ;

COMMENT COMPUTE THE CORRECTIVE TERM FOR THE NEXT SEGMENT ;

        EPSILON := ZRXSMOOTHNB*2 ;
        END;

COMMENT NORMALIZE THE AMPLITUDE PARAMETERS ;

    FOR INDEX1 := 1 STEP 1 UNTIL 150 DO
    BEGIN    STORAMPSMOOTH[INDEX1] := 32*STORAMPSMOOTH[INDEX1]/AMPMAX;
        STORAMPRESD[INDEX1] := 32*STORAMPRESD[INDEX1]/AMPMAX ;
    END
END PREPROCESS ;
```

Algorithm 1 (continued). **Preprocessing Procedure** .

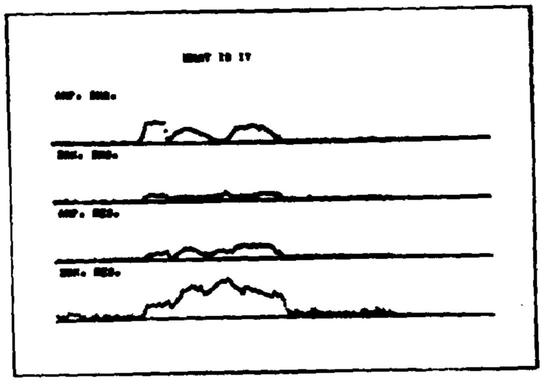
Results of the preprocessing program for the sound

"JOHN HAS A BOOK"



Results of the preprocessing program for the sound

"WHAT IS IT"



Results of the preprocessing program for the sound

"SPLIT IN TWO"

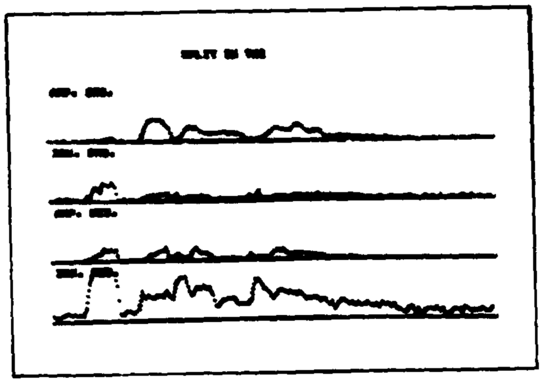


Figure II-1. Sofware Preprocessor Results .

As simple, unweighted averaging functions do not have good filter characteristics, the parameters we obtained were still not completely satisfactory. In particular, the zero-crossings of the residual wave obtained by differencing presented discontinuities which resulted from using a single estimator to characterize several dominant frequencies above 1000 Hz. The segmentation achieved using the four parameters was about 95 percent correct (Reddy and Vicens 1968) and, thus, the recognition process, based on this segmentation, was only 90 to 95 percent correct.

II-3. THE HARDWARE SPEECH PREPROCESSOR

Minimal segment, amplitude and zero-crossing are defined as in the preceding section. The last two are now determined using analog circuitry sampled every 10 ms.

To separate the high frequency components, it was decided to divide the speech frequency spectrum into three frequency bands, roughly corresponding to Formant 1, Formant 2 and higher frequencies. As vowels contain, in general, more reliable information than other phonemes, the choice of the cut-off values of the filters was dictated by known parameter values for the vowels (Peterson and Barney (1952)), see Figure II-2.

The complete circuit as represented on Figure II-4, is a hybrid circuit, partly analog (e.g., peak-to-peak detectors, zero-crossing counters) and partly digital (e.g., channel multiplexer clock). After sampling by the A-D, it provides the recognizer system with 6 parameters: amplitude and zero-crossing parameters for each filter output. The digital portions of the circuit were built using DEC R series modules, the analog portions were "home made" on compatible flip-chip modules. The filters are bandpass types with 600 ohm input and output and were manufactured by TT Electronics, Inc. For each filter the cut-off ratio on both sides is $F/F_c = 0.65$ at a 40 db attenuation.

II-3-1. The Analog Circuitry

The analog circuitry is built to supply voltages to one channel of an analog-to-digital converter through the internal channel multiplexer of the device. The parameter values are accumulated in capacitors which are sampled and reset every 10 ms by the digital clock. The time

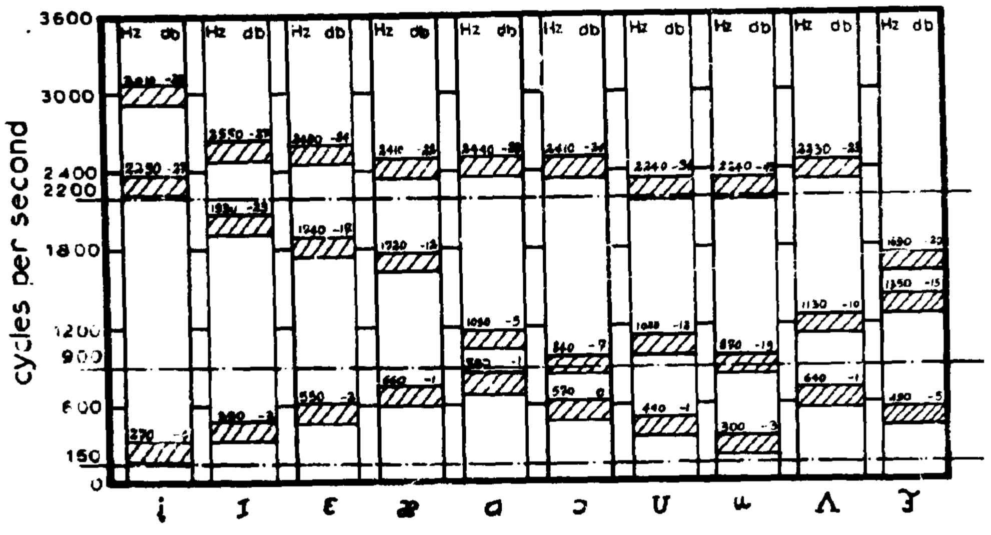


Figure II-2. Mean Formant Frequencies and Relative Amplitudes for Male Speakers Uttering the English Vowels in an /h-d/ Environment - (after Peterson and Barney).

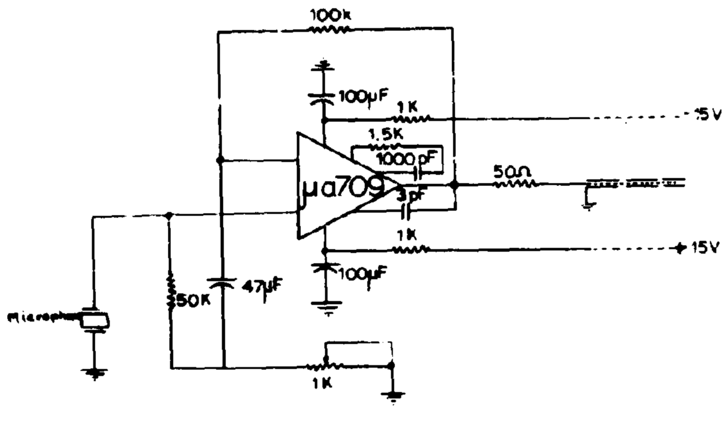


Figure II-3. The Microphone Amplifier

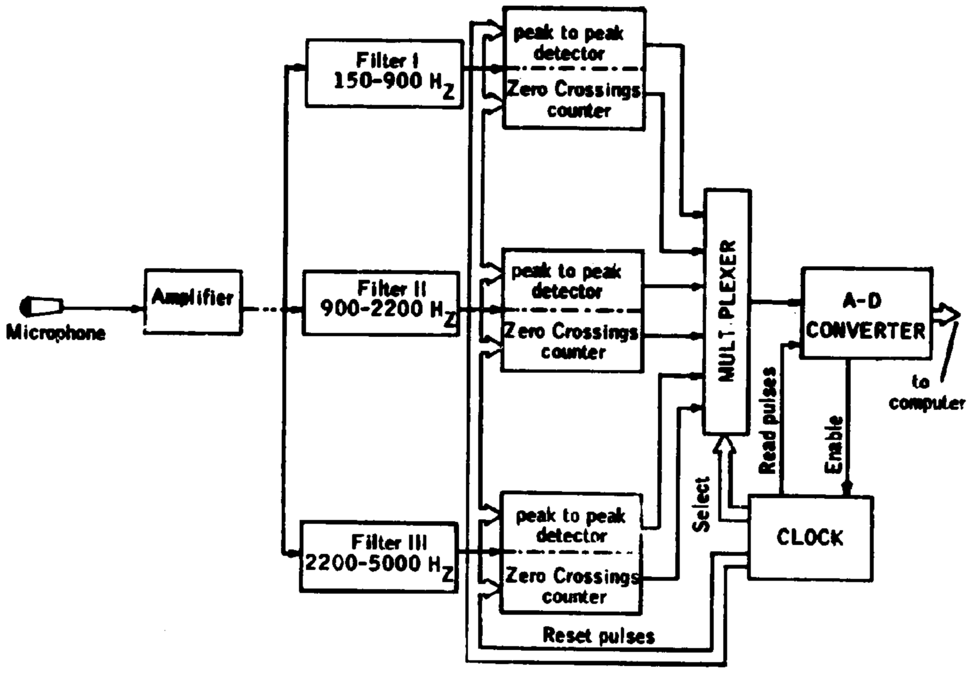


Figure II-4. The Hardware Speech Preprocessor .

necessary to sample and reset one parameter is approximately 90 microseconds; this results in an error of less than 1 percent over a 10 millisecond interval.

A peak-to-peak detector and a zero-crossing counter are mounted on the same double standard board and accept a 1 volt analog signal as input.

The reset pulses are DEC positive-going pulses (-3 to 0 V) of 40 μs duration created by the clock.

The Peak-to-Peak Detector

The peak-to-peak detector is represented on Figure II-5, is composed of three distinct parts: a positive peak detector, a negative peak detector and a differencing amplifier.

The two peak detectors have an identical design, only the polarity of the active circuitry is reversed. Both of them use an operational amplifier mounted on a unity gain feedback amplifier. The 0.1 μF capacitor in the feedback loop is charged to the peak value previously detected and is maintained at this charge by the operational amplifier. The field-effect transistor present in the feedback loop prevents the charge leakage from the 0.1 μF capacitor. The other circuitry shown serves to compensate the frequency response of the operational amplifier and to discharge the 0.1 μF capacitor when reseting the circuit.

The differencing amplifier subtracts the two voltages present on each capacitor and amplifies the result with a gain of 5, giving an output between 0 and -10 V, thus using the full scale of the A-D converter. The output waveform of the peak-to-peak detector, superimposed on the output waveform for two different frequencies and levels, may be seen on

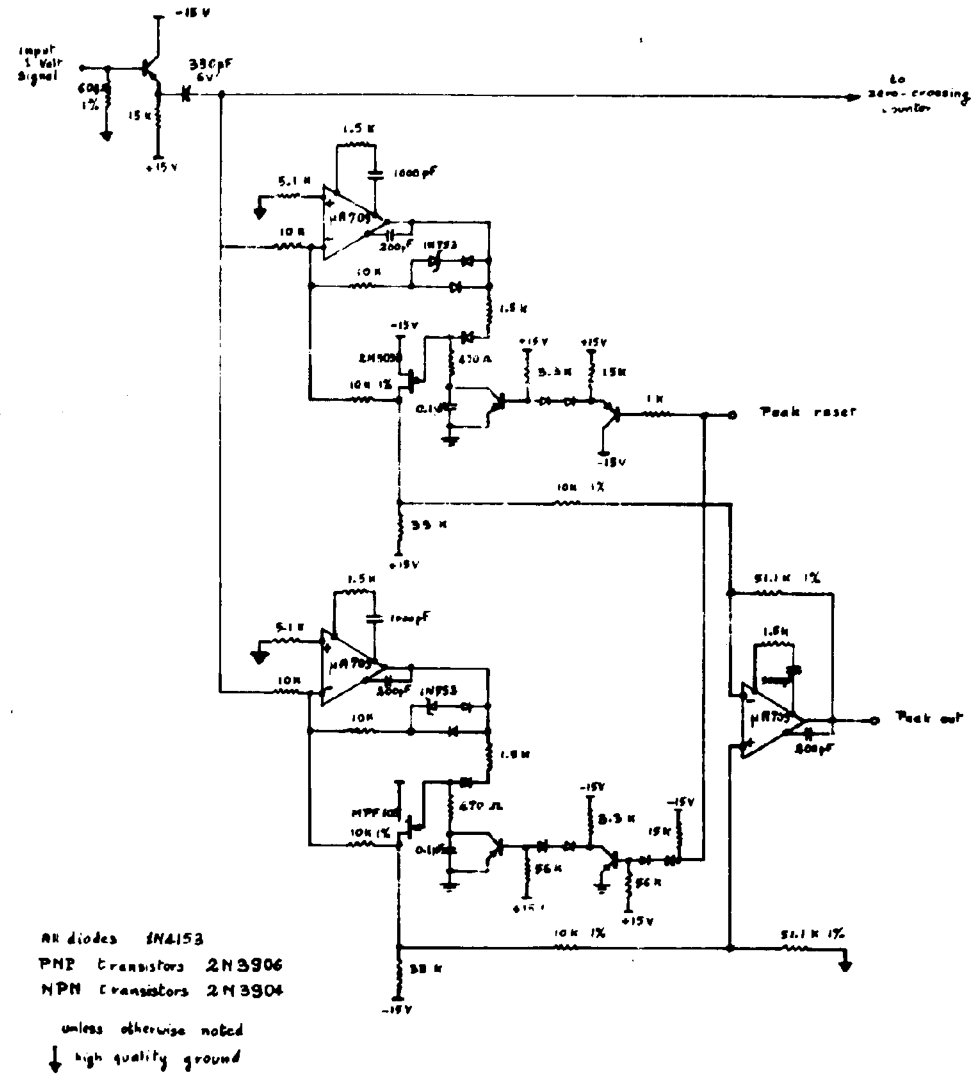


Figure II-5. The Peak-to-Peak Detector .

Figure II-8.

The Zero-Crossing Counter

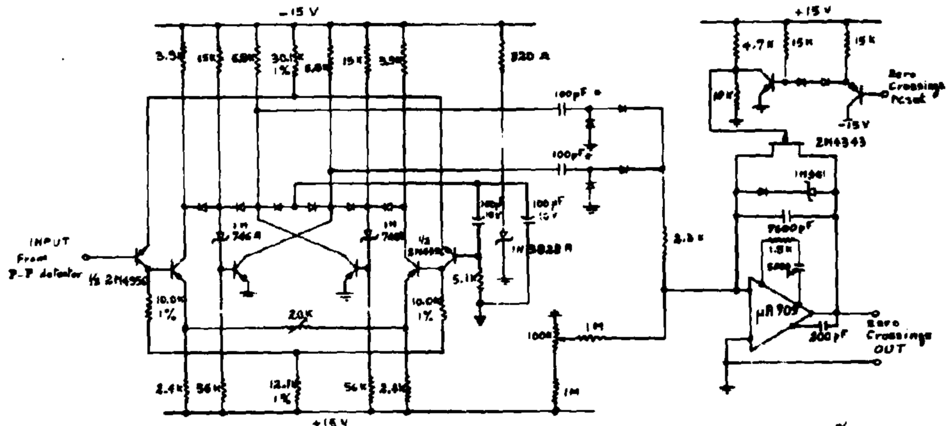
The zero-crossing counter, as represented in Figure II-6, is also composed of three distinct parts: a <u>differential amplifier</u>, a <u>flip-flop</u> and an <u>integrator</u>.

The differential amplifier, built with a double transistor (two identical transistors in the same package) amplifies the input signal. If the signal level is high enough, each sign change of this signal changes the state of the flip-flop. This acceptance level is adjusted by means of the 20K potentiometer represented on the drawing. After several tests, this potentiometer was adjusted for an acceptance level of 0.03 V on the original signal, i.e., the zero-crossings are counted only if the amplitude of the original signal is higher than 0.03 V.

Each time the flip-flop changes its state, it charges up the integrator by a small amount through the two matched 100 pF capacitors. The necessary fixed reference voltage is obtained by means of a zener diode (1N3828A) clamping the output of the flip-flop.

The integrator uses an operational amplifier to charge up the 7600 pF capacitor. This capacitor is discharged by a field-effect transistor during the reset period.

The values of the 7600 pF capacitor, 100 pF capacitors and the reference zener diode are such that -10 V, which is the full scale level for the A-D converter, represents 100 zero-crossings. The offset of the integrator is compensated by a resistance network connected to the inverting input of the operational amplifier. Again the operational amplifier frequency compensation circuitry and the reset circuitry are

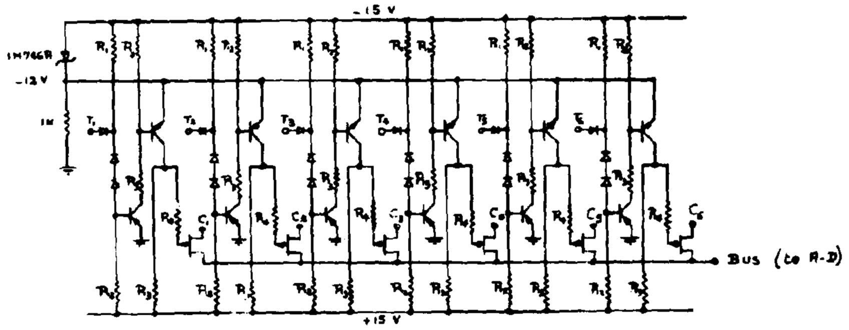


All diodes 1N4153
PNP transistors 2N3906
NPN transistors 2N3904

100 pF ∘ Both matched to 0.5 %

↓ High quality ground

Figure II-6. The Zero-Crossing Counter



All diodes	1N4153	R_1 = 33 K
PNP transistors	2N3906	R_2 = 10 K
NPN transistors	2N3904	R_3 = 27 K
FET transistors	2N4343	R_4 = 22 K

$T_1, T_2, T_3, T_4, T_5, T_6$ triggers

$C_1, C_2, C_3, C_4, C_5, C_6$ Analog inputs

Figure II-7. The Channel Multiplexer

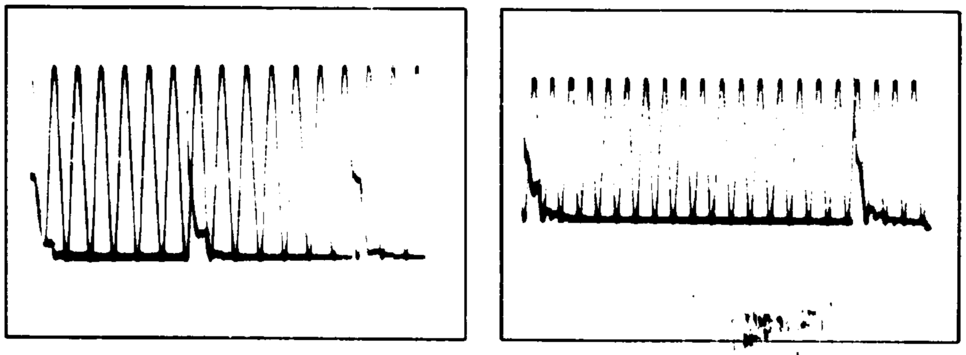


Figure II-8. Peak-to-Peak Detector Outputs .

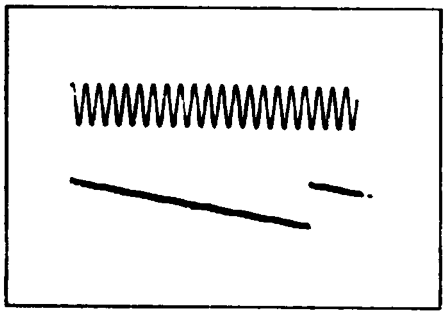


Figure II-9. Zero-C ssing Counter Output .

presented in the drawing . Figure II-9 shows the output of the integrator along with the input to the circuit for a given frequency.

The Channel Multiplexer

The purpose of the channel multiplexer is to connect each circuit to be read to the A-D converter via a common bus. Figure II-7 represents a schematic of the entire multiplexer which is composed of six analog multiplex switch circuits. Each has a separate control or trigger input, and a separate analog input. All the analog outputs of the switches are tied together to a common bus going to the A-D. The control inputs are compatible with DEC logic, i.e., the switch is ON when the trigger is in the TRUE state (-3 V), OFF when the trigger is in the FALSE state (0 V).

II-3-2. The Digital Circuitry - The Clock

A logical diagram of the clock, the only digital circuit present in the device, is shown on Figure II-10. This clock is a pulse generator which "manages" all the other circuit components. Three kinds of pulses are generated:

- The setup pulses sent to the selected multiplexer trigger.
- The read pulses sent to the analog-to-digital converter.
- The reset pulses sent to the circuit previously read.

The setup pulses are negative going pulses (0 V to -3) of 40 μs duration. As long as the trigger is in the TRUE state (-3 V), the corresponding switch stays ON and the selected analog circuit component is connected to the A-D converter.

The read pulses are standard 100 nanosecond pulses (-3 V to 0 V) sent to the external clock input of the A-D converter (a provision is

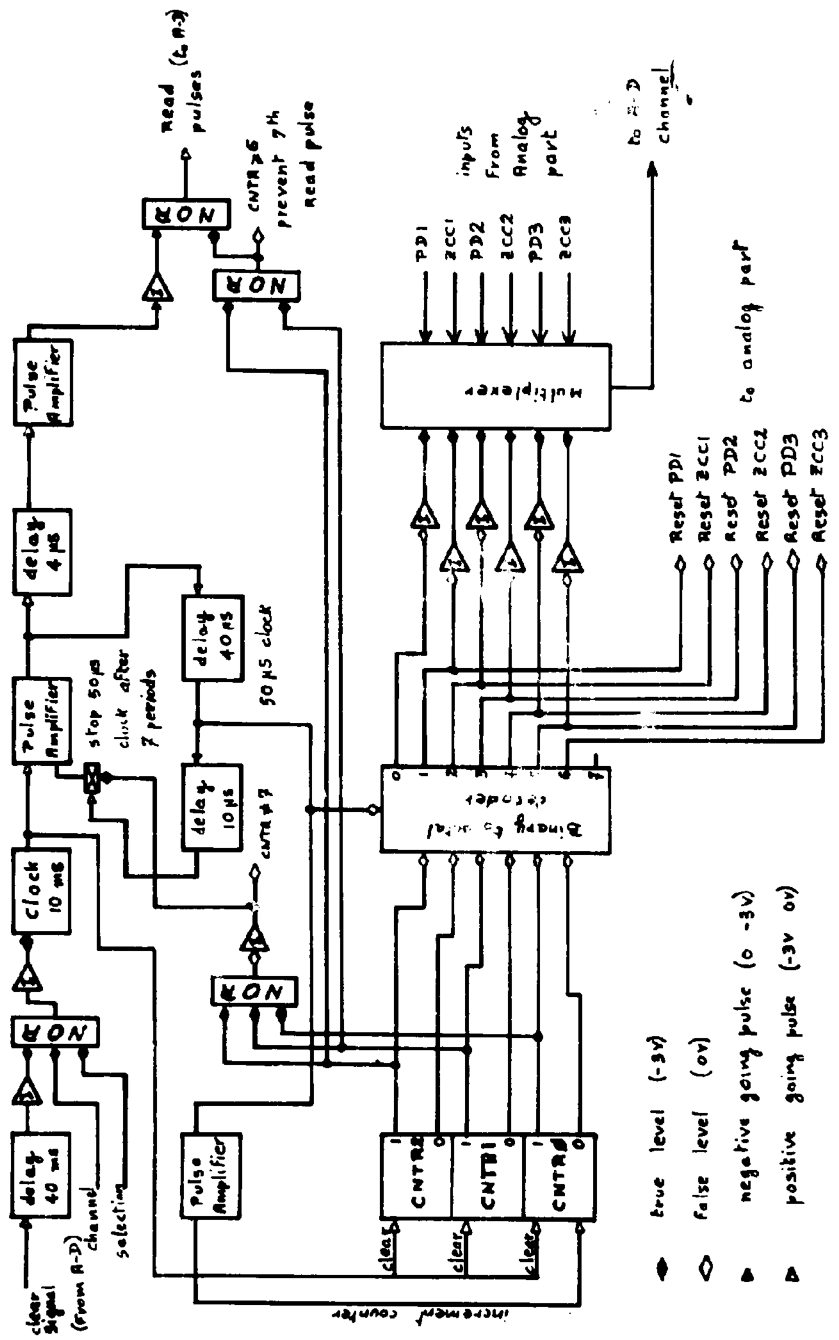


Figure II-10. The Digital Clock .

made on our converter, such that each time a standard pulse is sent to this specific input, a read operation is initiated.) The read pulses are 4 μs late with respect to the beginning of the setup pulses, so that the selected circuit is switched into the bus when the reading cycle is initiated.

The reset pulses are positive going pulses (-3 V to 0 V) of 40 μs duration. As long as the voltage is 0 V, the corresponding capacitor in the analog circuitry is discharged. A detailed pulse timing chart for one 10 ms read cycle is shown on Figure II-11.

Since 10 ms is the basic sampling period (minimal segment), a read cycle of the six parameters must be initiated every 10 ms. Two independent clocks are necessary: a 10 ms master clock which initiates two one-shot multivibrators which in turn define the 50 μs intervals. A counter stops this 50 μs clock after seven periods and the device waits for the next 10 ms pulse coming from the master clock.

Setup and reset pulses are obtained from the output of the 40 μs one-shot multivibrator through a binary-to-octal decoder driven by a counter. The setup pulse of one circuit is obtained by inverting the reset pulse of the previously read circuit.

The read pulses are obtained from the output of the 10 μs one-shot multivibrator through another one-shot delay and a standardizing pulse amplifier. The delay was adjusted so that these pulses are 4 μs late with respect to the setup pulses.

The clock must fulfill some requirements imposed by the A-D converter or by our sampling scheme:

- Pulses must not be sent to the external clock input of the

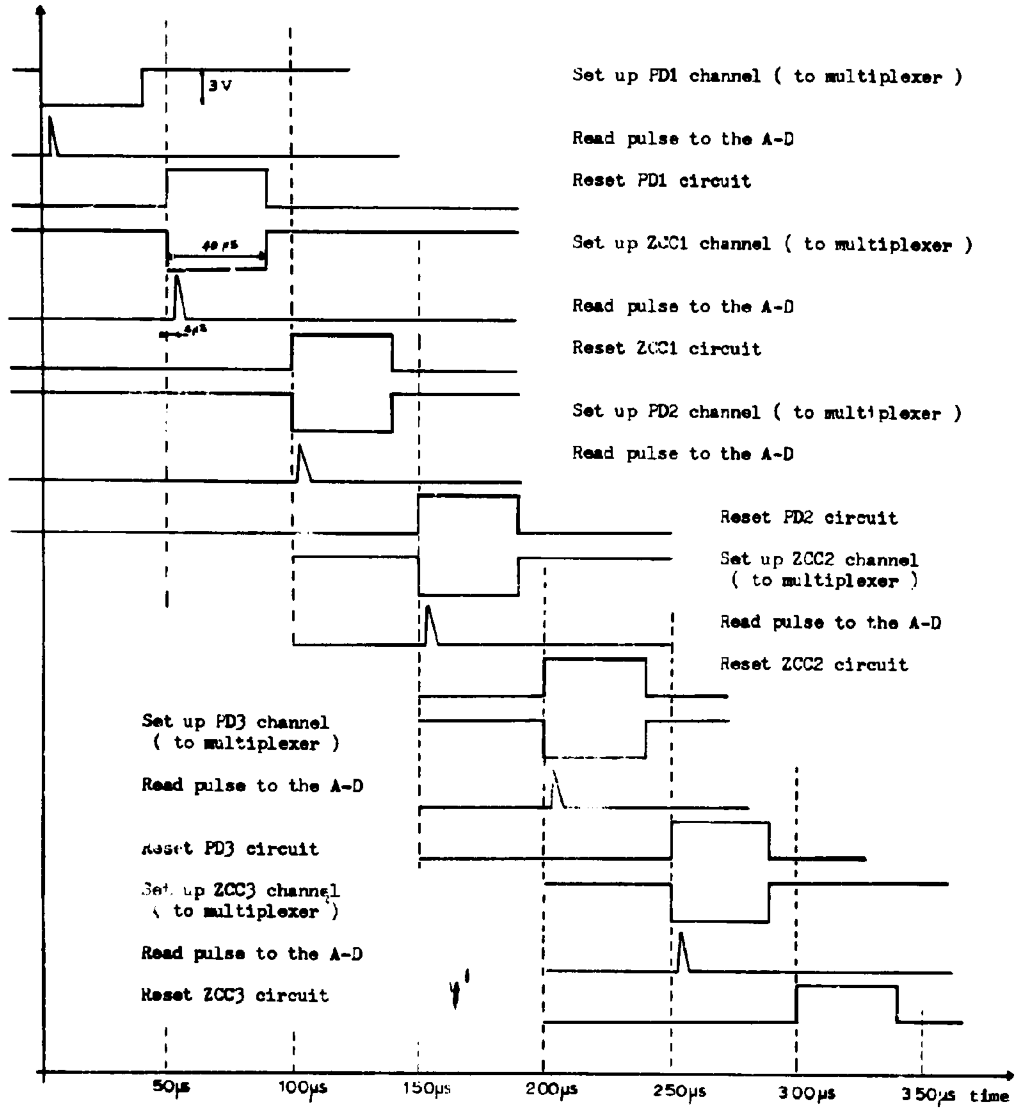


Figure II-11. Digital Pulses Timing Chart .

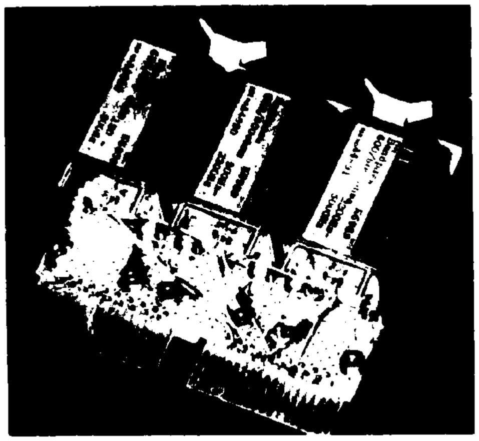
converter when somebody else is using it. A very simple solution to this problem was implemented: the clock is only allowed to run when the corresponding A-D converter channel is selected by the PDP-6 system.

- When initiating a read operation for a series of samples, we have to be sure that the first parameter we get is the right one (or all of them would be interchanged). This was done by restarting the master clock each time a clear signal is sent by the system to the converter, indicating that a new operation is initiated. Another one-shot multivibrator was necessary to allow the clock to finish the preceding cycle.

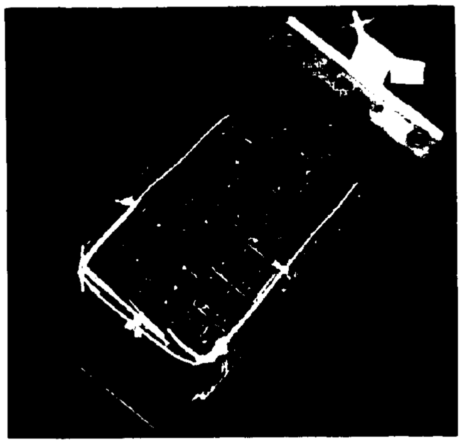
Figure II-12 shows all the circuit components in their actual form. All of them were mounted on plugable standard flip-chip boards.

II-3-3. The Hardware Preprocessor Service Routine

As described earlier, the device is connected to an analog to digital converter. The latter communicates with the central processor and the core memory through a medium-speed Data Control DEC type 136 and an I/O bus (Figure I-1). The hardware preprocessor is initiated by an input operation from the central processor, and the six parameters which are extracted every 10 milliseconds are packed by the DEC type 136 into two 36 bit words and stored in core memory. In order to make available to the segmentation procedure some convenient numbers, it is necessary to unpack and to normalize the original parameter data. The hardware preprocessor service routine is a real-time user's program which starts the I/O operation (and the tape recorder if this device is used), unpacks and normalizes the parameter data after detection of the sound beginning, detects the end of the sound and stops the I/O operation (and the tape

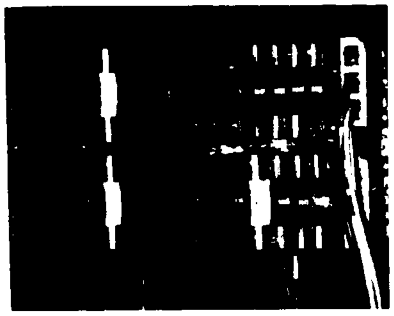


The Filters



The Channel Multiplexer

A Peak to Peak Detector
and a Zero Crossing Counter



The Hardware Speech Preprocessor

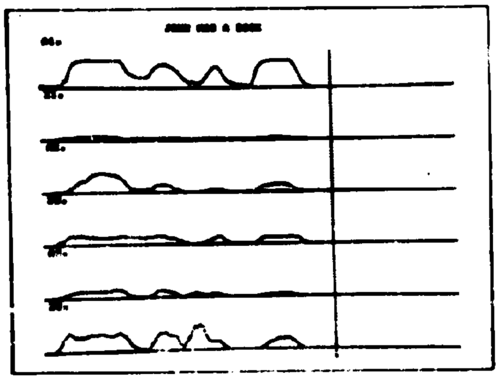
Figure II-12. Pictures of the Circuit Components .

recorder). A real-time user's program is treated as a special case by the time-sharing system: it is restarted every 16.7 millisecond (60 Hz), and is executed in parallel with the regular user's program. Furthermore, it runs in supervisor mode, thus allowing for all kinds of special Input/Output operations and system manipulations. The hardware preprocessor service routine processes the microphone input buffer while the Data Control DEC type 136 is filling up the same buffer. The two processes (i.e., data control and real-time program) are not loosely connected since, in a given period of time, the real-time program treats more samples than the Data Control can create (it fills up two 36 bit words every 10 ms). When necessary, the real-time program waits for the other process. This program checks the audio input every 16.7 milliseconds. If relevant information is coming in, this information is unpacked, normalized, transferred to our input buffer and checked for silence. When a long silence has been detected, the program, assuming that the speech utterance is finished, gives the estimated size of the input buffer, stops the I/O operation in the supervisor, turns the tape recorder off if this device is used, and turns itself off, thus returning the control to the regular user's program.

II-3-4. <u>Conclusions</u>

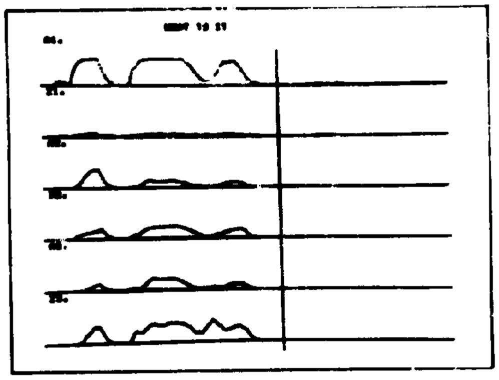
Displays of the six parameters obtained for the messages previously analyzed with the software preprocessor can be seen on Figure II-13. By comparing the two sets of pictures (Figures II-1 and II-13), one can easily see that the new sets of parameters are smoother than the previous ones. Furthermore, the noise of the room is reduced and the service routine gives us the duration of the sound (represented by the vertical line), Results of the preprocessing hardware for the sound

"JOHN HAS A BOOK"



Results of the preprocessing hardware for the sound

"WHAT IS IT"



Results of the preprocessing hardware for the sound

"SPLIT IN TWO"

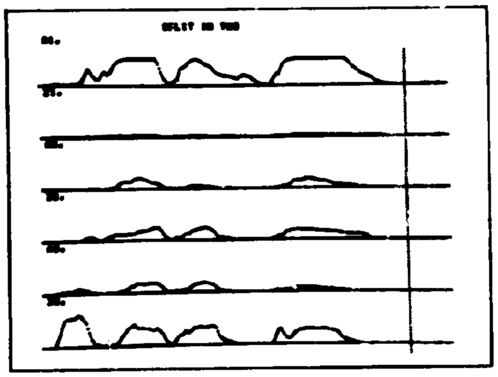


Figure II-13. Hardware Preprocessor Results .

the sample always starting at the beginning of the input buffer. The parameters derived by this process appear to result in better segmentation and recognition by the other stages of the system.

II-4. CONCLUSIONS

The procedures described above for the extraction of significant parameters of speech form the first step in a more elaborate speech recognition system presented in the subsequent chapters. Their validity is proved only because the complete system gives satisfactory recognition scores. The fact that we have obtained good results is due in part to the judicious choice of parameters but is mainly due to the power of the subsequent recognition algorithms. In fact, we think that any other significant parameters may be used in place of the ones used here without degradation of the results. The main reason for the choice of parameters is to provide a reasonable compromise with respect to simplicity of the system and completeness of their representation of the speech signal. On the other hand, they do not always give a complete representation, and occasional confusions result.

Chapter III

SEGMENTATION AND DESCRIPTION OF CONNECTED SPEECH UTTERANCES

III-1. INTRODUCTION

If we plot the changes in air pressure produced by a speech utterance as a time -vs- pressure graph we obtain a speech wave such as the one given in Figure III-1 Note that neither the words of the utterance nor the sounds within words are separated as in the case of the written form of the human language, and yet we are able to associate discrete written forms with continuous spoken forms. To be able to make similar associations, a machine must be capable of dividing connected speech utterance into discrete parts. This problem is known as the problem of segmentation of connected speech.

Segmentation of speech is of interest in many different areas of speech research. In speech recognition one must match the incoming signal with the known linguistic elements It is unrealistic to attempt to do one-for-one pattern matching at the waveform level or by using the output of any preprocessing procedure similar to those described in the first chapter What is needed is a transformation which will reduce the parameters to be matched to a manageable level. Segmentation, as described in this chapter, is one such transformation, which generates a description of the incoming signal using the parameters produced by the preprocessing procedure. For example, the representation of the word six might be as follows: "Fricative, followed by a transition, followed by a vowel, followed by a transition, followed by a stop, followed by a fricative, each with the following parameters:. . . " For the recognition of a limited set of messages (even as many as 1000) such a description is usually adequate (Chapter IV). For

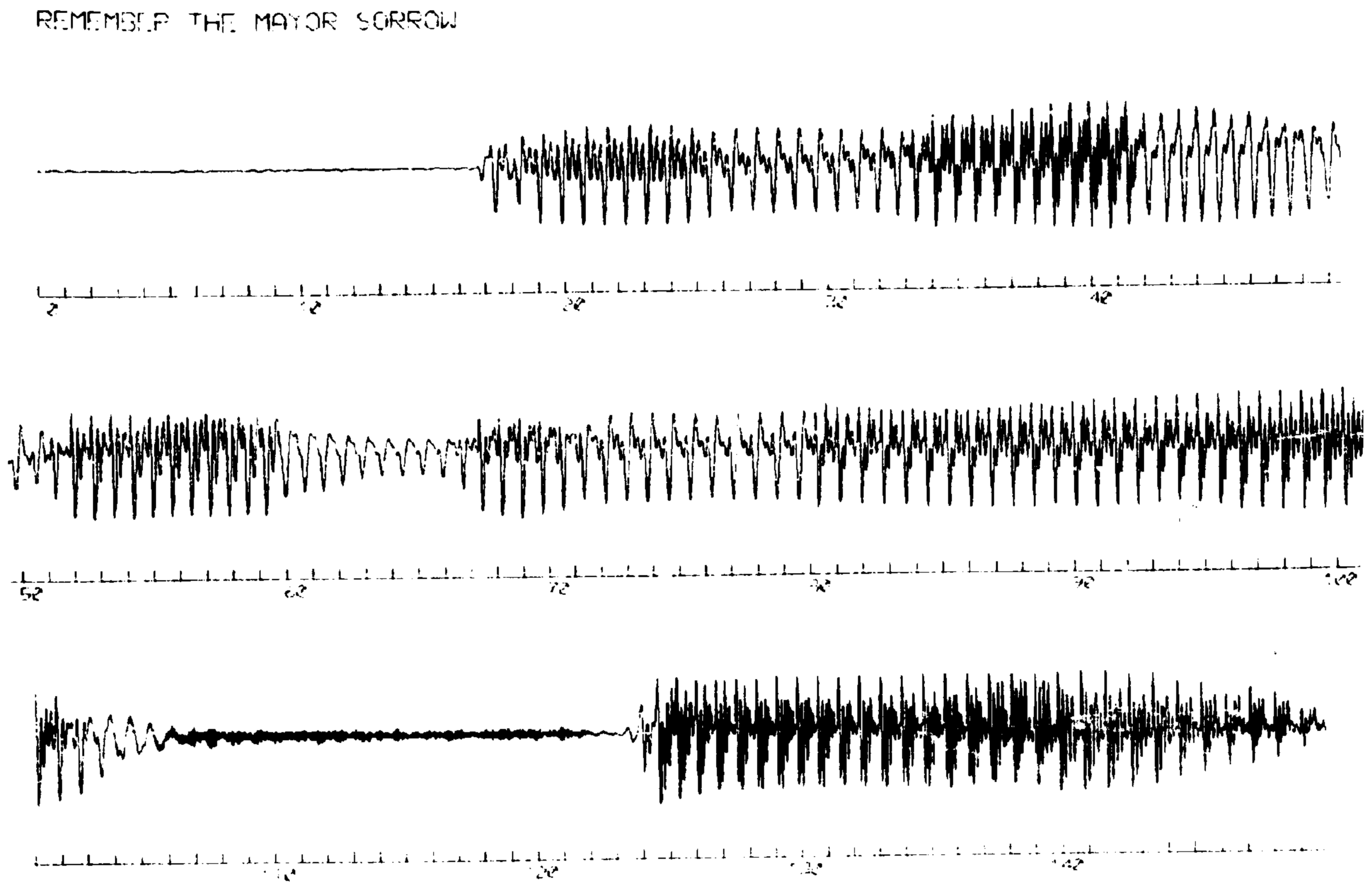


Figure III-1. Original Waveform for the Sound : REMEMBER THE MAYOR SORROW .

an automatic phonetic typewriter system, one might need a further investigation of the transitional segments to determine whether the word was six or slits.

At present, parameters for speech synthesis systems are obtained by the tedious manual measurement process which might take several days or even months to generate a single sentence. An automated parameter measuring process based on segmentation should reduce the time to minutes or even seconds. In speech compression systems, a segmentation program coupled with a pitch period determination program can be used to replace a periodic sustained segment by a single pitch period and a repetition factor.

Fry and Denes (1955), Sakai and Doshita (1963), Hughes and Hemdal (1965), Gold (1966) and Reddy (1966) have all had to develop segmentation procedures in connection with their speech recognition systems. The first two had to build special purpose hardware to segment the sounds.

Two computer coded segmentation procedures were implemented, corresponding to the software preprocessor and the hardware preprocessor. As they show the same basic ideas, only the latest, intended to process the output from the hardware, will be described in some detail. In both cases, we attempted not to restrict the algorithm to a single cooperative speaker. Also, no attempt was made to artificially reduce the noise level of the room since that would not be the normal mode of man-machine communication.

The segmentation program is written in Fortran IV and uses 4K of 36 bit words of core memory. Intermediate and final results were displayed on a CRT display to determine the goodness of segmentation and to trace down all the possible errors.

A detailed flow chart of the various stages of the segmentation program is given in Figure III-2. The input to the procedure is a matrix built by

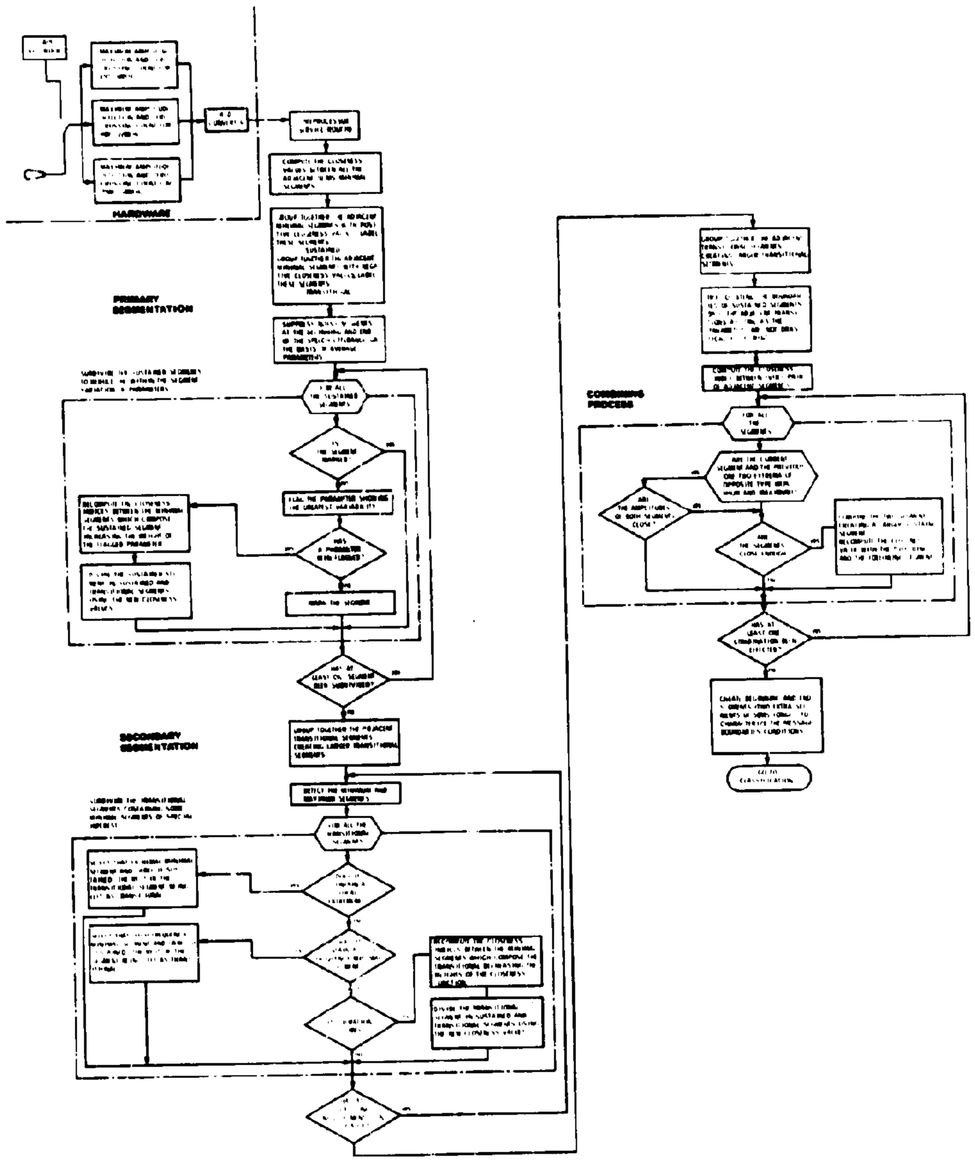


Figure III-2. Flowchart of the Segmentation Process .

the hardware preprocessor service routine which contains, for each 10 ms period, the six normalized parameters extracted by the analog device. Closeness indices are computed between adjacent minimal segments characterized by these parameters. The primary segmentation procedure groups together adjacent minimal segments that may be regarded as being similar, forming primary segments. The secondary segmentation procedure divides these primary segments into smaller segments if the within-segment variation of parameters is too high. The closeness indices are then recomputed between the secondary segments using the average parameters and weaker weights. If two adjacent secondary segments are sufficiently close, they are combined to form lerger segments. A classification procedure labels all the sustained segments as possibly belonging to one of the phoneme groups: fricative, vowel, stop, nasal, consonant or burst. On the basis of the labeled segments, some additional combining is performed, e.g., of adjacent fricatives or stops. A feature matrix, containing some general information on the speech utterance, along with the average parameters for each segment is then built. This matrix is the internal representation of the speech utterance used in all the subsequent processes: storing, retrieving, matching.

III-2. PRIMARY SEGMENTATION

The purpose of the primary segmentation procedure is to group together similar adjacent minimal segments which are produced by the preprocessing procedure. The segments created are labelled <u>sustained</u> or <u>transitional</u> and stored for the next procedure.

In order to perform this first grouping, we must provide a criterion which will define the similarity or closeness between two segments. To allow some compactness in our formulas, let us use the vectorial notation. A <u>minimal segment</u> is then represented by a n - component vector $\underline{V}$, and a speech utterance by a matrix $U_{m \times n}$, m being the number of minimal segments (i.e., the duration of the utterance in 10 ms unit). This representation, adequate to carry out the computation involved in this segmentation procedure, is in fact our initial representation of the sound (Figure III-4 displays these 6 parameters).

One can define the closeness of two segments in terms of Euclidian distance between the two points of the n-dimensional space; however, such a simple metric proves to be unsatisfactory in this case. To be effective, the metric should ignore the intra-phoneme variability and be sensitive to inter-phoneme variability. Our experiments have shown that the closeness index function should obey the following heuristics:

H1. Since certain parameters have a large range of variation, the closeness function should provide for appropriate weighting of parameters. Let us assume that this is specified by means of a weight vector $\underline{W}$ of dimension n (same as $\underline{V}$).

H2. Unvoiced fricatives, mainly /s/, have to be treated separately because of the great variation possible in all the parameters. However, these are easily recognized because of the large value of the zero-crossing parameter in the third frequency band and the relatively small amplitude in the first band.

H3. When a zero crossing parameter is less than a minimum, this means that the amplitude of the signal in this frequency band was too small to exceed our 0.03 volt acceptance level during the 10 ms period (the zero-crossing counter, Chapter II). Therefore we consider this parameter erroneous and decrease the corresponding weight accordingly.

H4. Although most of the parameters may be similar, a drastic change in one parameter should result in a 'not-similar' indication. Let the drastic change threshold be defined by a limit vector $\underline{L}$.

H5. If the difference between corresponding parameters is less than a minimum, then the two parameters should be considered as identical. Let the minimum difference threshold be defined by a vector $\underline{M}$.

H6. The larger the parameter value, the greater should be the difference that we are willing to accept. This suggest the use of a relative error function such as $\frac{\Delta y}{y}$.

H7. When the parameters are close to zero the relative error function $\frac{\Delta y}{y}$ can take abnormally large values even though the difference between the parameters is not correspondingly large. In order to correct this defect, we use a modified relative error expression: $\frac{\Delta y}{\sqrt{y}}$. This choice may seem arbitrary, but it arises simply from the replacement of an initial scale function by a second degree approximation.

Our first attempt to represent the closeness index between two numbers was to multiply the value obtained for $\frac{\Delta y}{y}$ by a factor $f(y)$ defined by:

$f(y) = 1$ if $12 < y$

$f(y) = 1/2$ if $6 < y \leq 12$

$f(y) = 1/4$ if $0 \leq y \leq 6$

as shown in Figure III-3

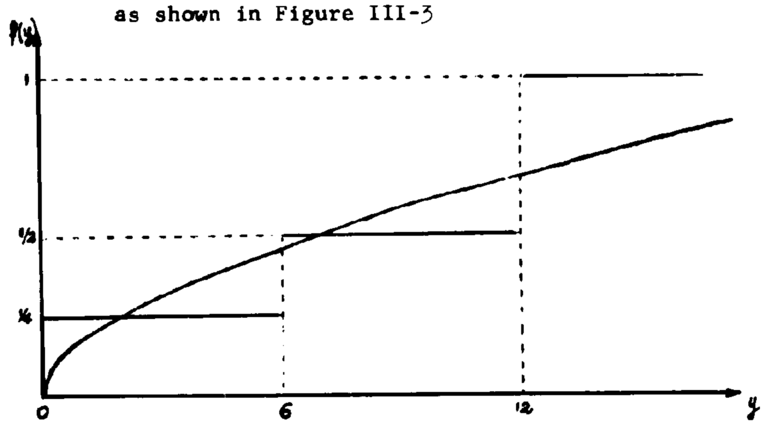


Figure III-3 Rationale for the Choice of $\frac{\Delta y}{\sqrt{y}}$ as Basic Closeness Measure

Of course such a representation was very inconvenient because of the discontinuities presented at $y = 6$ and $y = 12$. A cure was to replace this scale function by the parabola $\frac{1}{4\sqrt{2}}\sqrt{y}$ (shown in Figure III-3). The weight vector W can take into account the constant factor $\frac{1}{4\sqrt{2}}$, thus leaving $\frac{\Delta y}{\sqrt{y}}$ our basic closeness measure for all y.

Now we can precisely define the closeness index function c. Let V1 and V2 be the parameter vectors representing two adjacent minimal segments. Then the elements of the relative difference vector R are given by:

$$R_i = \frac{(|V1_i - V2_i|)}{\sqrt{V1_i + V2_i}}$$

Let C represent a closeness vector, whose elements are given by:

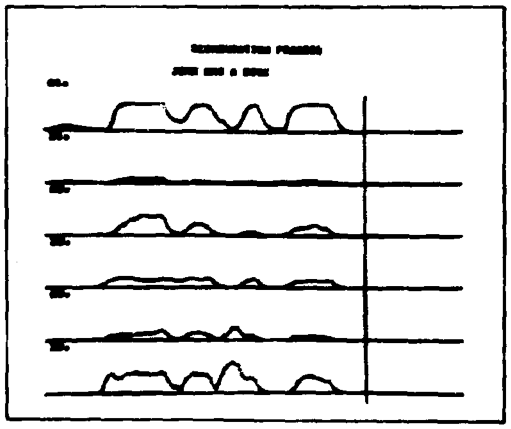


Figure III-4. Result of the preprocessing procedure for the sound: "JOHN HAS A BOOK"

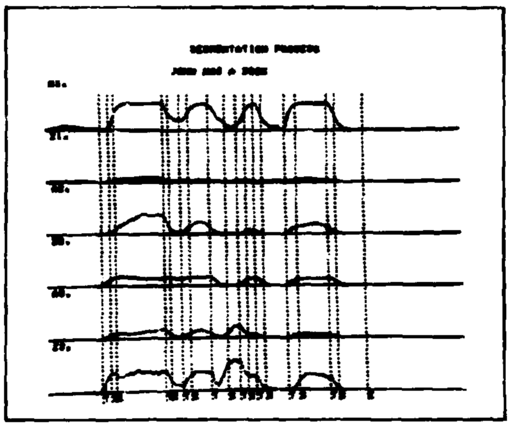


Figure III-5. Segments after primary segmentation

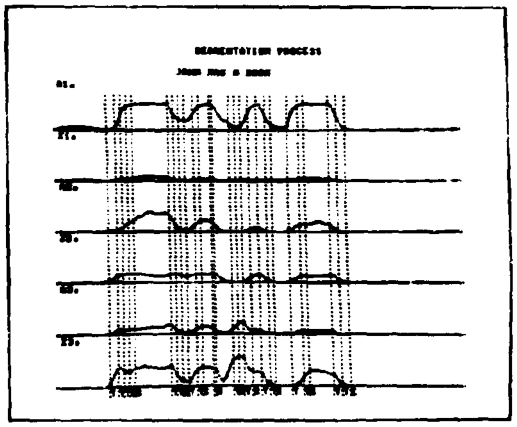


Figure III-6. Segments after secondary segmentation

$$C_i = 2, \text{ if } |V1_i - V2_i| \leq M_i;$$
$$= 2.5 - W_i \times R_i, \text{ otherwise.}$$

Now closeness index between V1 and V2 may be defined as follows:

$$c = \min\left(-4, \sum_{i=1}^{n} C_i\right), \text{ if } \bigvee_{i=1,n} (R_i > L_i);$$
$$= \sum_{i=1}^{n} C_i, \text{ otherwise.}$$

the constants of the closeness index computation are chosen so that the closeness index will be positive if the minimal segments are similar and negative otherwise.

An Algol equivalent of the similarity computation procedure is given next. (Algorithm 2). Two parameters defined in the procedure are not yet defined: MORVARPARAM and SPECIALWEIGHT, their use will be explained in the secondary segmentation description.

The primary segmentation consists mainly of creating larger segments by combining together all the adjacent minimal segments having closeness indices greater than or equal to zero. All segments whose indices are less than zero are not combined with any segment and therefore form transitional segments. A display of the primary segmentation for the sound "JOHN HAS A BOOK" can be seen in Figure III-5. For most of the sounds, this primary segmentation already provides a satisfactory division. However, where the parameter transitions are so gradual that there is a little noticeable change from segment to segment, this procedure could result in grouping together two acoustically different parts of the sound. The detection of such segments and the consequent error recovery is left to the secondary segmentation.

```
INTEGER PROCEDURE CLOSENESS(SEGSTORG,SEGNB1,SEGNB2,WEIGHTSET,
                 MORVARPARAM,SIZSEGSTORG) ;

    INTEGER SEGNB1,SEGNB2,WEIGHTSET,MORVARPARAM,SIZSEGSTORG ;
    INTEGER ARRAY SEGSTORG[1:SIZSEGSTORG,1:7] ;


COMMENT
        THIS PROCEDURE COMPUTES THE CLOSENESS VALUE BETWEEN THE
    TWO SEGMENTS STORED IN THE ROWS SEGNB1 AND SEGNB2 OF THE ARRAY
    SEGSTORG

        WEIGHTSET INDICATES WHICH SET OF CONSTANTS IS TO BE USED
    IN THIS CLOSENESS INDEX COMPUTATION .

        MORVARPARAM FLAGS THE MORE VARIABLE PARAMETER AS DETECTED
    BY THE PROCEDURE CHECKVARIATION . THE WEIGHT CORRESPONDING TO
    THIS PARAMETER WILL BE SLIGHTLY INCREASED .


        THE REAL ARRAYS WEIGHT[1:2,1:6] AND RATIOLIM[1:2,1:6]
    AND THE INTEGER ARRAYS LIM[1:6] AND ZRXLIM[1:6] ARE DEFINED
    AS GLOBAL ARRAYS IN THE MAIN PROGRAM AND FILLED WITH CONSTANTS
    AT COMPILE TIME (IF POSSIBLE, OR WHEN STARTING THE PROGRAM).


        WEIGHT[1,1] := 4,0 , WEIGHT[2,1] := 4,0 ,
        WEIGHT[1,2] := 7,5 , WEIGHT[2,2] := 6,0 ,
        WEIGHT[1,3] := 4,0 , WEIGHT[2,3] := 4,0 ,
        WEIGHT[1,4] := 7,5 , WEIGHT[2,4] := 5,0 ,
        WEIGHT[1,5] := 4,0 , WEIGHT[2,5] := 4,0 ,
        WEIGHT[1,6] := 7,5 , WEIGHT[2,6] := 5,0 ,


        RATIOLIM[1,1] := 2,5 , RATIOLIM[2,1] := 2,0 ,
        RATIOLIM[1,2] := 1,2 , RATIOLIM[2,2] := 1,0 ,
        RATIOLIM[1,3] := 2,5 , RATIOLIM[2,3] := 2,0 ,
        RATIOLIM[1,4] := 1,2 , RATIOLIM[2,4] := 1,0 ,
        RATIOLIM[1,5] := 2,5 , RATIOLIM[2,5] := 2,0 ,
        RATIOLIM[1,6] := 2,0 , RATIOLIM[2,6] := 1,6 ,


        LIM[1] := 4 ,LIM[2] := 2 , LIM[3] := 4 , LIM[4] := 4 ,
        LIM[5] := 4 , LIM[6] := 10 ,


        ZRXLIM[1] := 0 , ZRXLIM[2] := 2 , ZRXLIM[3] := 0 ,
        ZRXLIM[4] := 14 , ZRXLIM[5] := 0 , ZRXLIM[6] := 30;
```

Algorithm 2. Closeness Evaluation Procedure .

```
BEGIN   REAL    REALCLOSE ;
        BOOLEAN NONSIMILAR;
        INTEGER PARAMNB ;
        LABEL ENDLOO ;


COMMENT
        FRICATIVE TYPE "S" SPECIAL CASE : THOSE SEGMENTS ARE
    DETECTED AND A POSITIVE VALUE IS GIVEN TO CLOSENESS IN ORDER
    TO HAVE THEM COMBINED ,


    IN THE NEXT LINES, AMP1[,] STANDS FOR SEGSTORG[,,1]
              AMP3[,]       "       SEGSTORG[,,3]
              ZEROX3[,]     "       SEGSTORG[,,6] ;


    IF AMP3[SEGNB1] ≥ AMP1[SEGNB1] AND AMP3[SEGNB2] ≥ AMP1[SEGNB2]
        THEN IF ZEROX3[SEGNB1] ≥ 60 AND ZEROX3[SEGNB2] ≥ 60
            THEN CLOSENESS := 8
            ELSE IF ZEROX3[SEGNB1] ≥ 45 AND ZEROX3[SEGNB2] ≥ 45
                AND AMP1[SEGNB1] ≤ 6 AND AMP1[SEGNB2] ≤ 6
                THEN CLOSENESS := 8

    ELSE BEGIN


COMMENT
    GENERAL CASE ;

    REALCLOSE := 0,0 ;
    NONSIMILAR := FALSE ;
    FOR PARAMNB := 1 STEP 1 UNTIL 6 DO
        BEGIN  TEMP1 := MAX(SEGSTORG[SEGNB1,PARAMNB],LIM[PARAMNB]) ;
               TEMP2 := MAX(SEGSTORG[SEGNB2,PARAMNB],LIM[PARAMNB]) ;
               SPECIALWEIGHT :=
               IF PARAMNB = MORVARPARAM THEN 1,25
                                        ELSE 1,0 ;


COMMENT
        SPECIALWEIGHT IS USED TO INCREASE THE TESTS CORRESPONDING
    TO THE PARAMETER FLAGED BY THE PROCEDURE CHECKVARIATION ;


            DIFF := ABS(TEMP1-TEMP2) ;
            IF DIFF ≤ LIM[PARAMNB] THEN BEGIN
                                        REALCLOSE := REALCLOSE+2,0 ;
                                        GO TO ENDLOO
                                        END ;
            RATIO := DIFF/SQRT(TEMP1+TEMP2) ;
```

Algorithm 2 (continued). Closeness Evaluation Procedure

```
COMMENT
        THE NEXT STATEMENT REFLECTS THAT IF THE ZERO-CROSSINGS ARE
  . UNDER A CERTAIN THRESHOLD, THEY ARE CONSIDERED ERRANEOUS AND
    THEREFORE WE DECREASE THE CORRESPONDING WEIGHT ;

            IF TEMP1 ≤ ZRXLIM[PARAMNB] AND
               TEMP2 ≤ ZRXLIM[PARAMNB] THEN RATIO := RATIO*0.7 ;
            IF RATIO > RATIOLIM[WEIGHTSET,PARAMNB]
                              THEN NONSIMILAR := TRUE ;
            REALCLOSE := REALCLOSE+(2.5-
            SPECIALWEIGHT*WEIGHT[WEIGHTSET,PARAMNB]*RATIO) ;
    ENDLOO:    END ;
    IF NONSIMILAR THEN CLOSENESS := MIN(-4,REALCLOSE)
                  ELSE CLOSENESS := REALCLOSE ;

    END

END CLOSENESS ;
```

Algorithm 2 (continued). Closeness Evaluation Procedure .

III-3 SECONDARY SEGMENTATION

The purpose of the secondary segmentation procedure is to correct the possible errors of the primary segmentation by looking at the variation of parameters in the sustained segments and at the local maxima and minima of the amplitude parameters in the transitional segments.

Every time a segment is created, the total variation for each parameter is kept. If, in any sustained segment, this variation exceeds a certain limit, the segment is divided into smaller segments. The limit computation, based on heuristics similar to those of the primary segmentation, depends also on the duration of the segment.

H1. The limit computation should provide for appropriate weighting of parameters. Let $\underline{W}$ be the weighting vector.

H2. Unvoiced fricatives, /s/ have to be detected and treated separately as in the primary segmentation.

H3. If the difference between corresponding parameters is less than a minimum, then the two parameters should be considered as identical. Let the vector $\underline{M}$ be the minimum difference threshold.

H4. The larger the parameter value, the greater should be the difference that we are willing to accept. This suggests the use of a relative error function such as $\frac{\Delta y}{y}$.

H5. The limit should depend on the duration. But two contradictory heuristics seem to direct this duration dependency:

-The longer the segment, the more likely that its parameters vary significantly. In other words, the longer the segment, the weaker our tests should be.

-The longer the segment, the more likely it represents two different phonemes with similar parameters. In other words, the longer the segment, the stronger our tests should be to detect this case.

To reflect such contradictory heuristics, a discontinuous function of the duration is used. A graphic representation of this function is given on Figure III-7. As indicated by the drawing, the tests remain constant, with respect to the duration, from 0 to 60 milliseconds. Then they decrease in intensity from 60 to 120 milliseconds, and for segments longer than 120 milliseconds, they again increase.

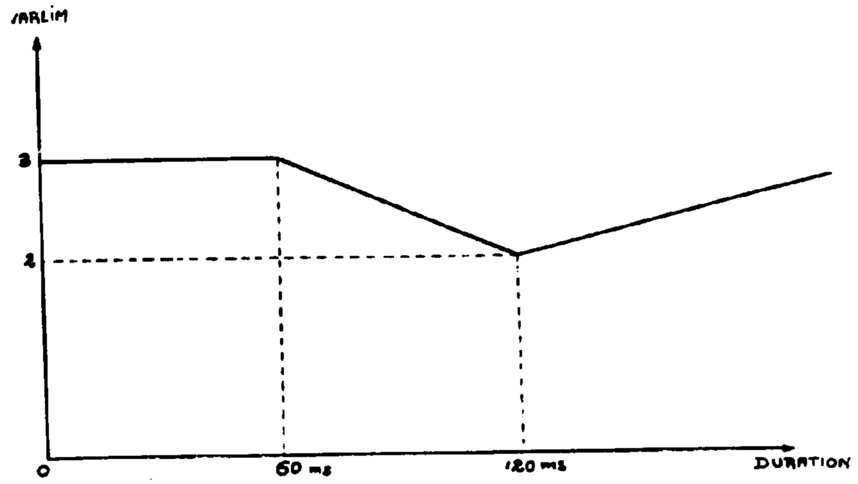


Figure III-7 Graphic Representation of the Factor Varlim

Therefore VARLIM is defined by:

$$
\begin{aligned}
\text{VARLIM} &= 3, \text{ if DURATION} \leq 60 \text{ ms} \\
&= 4 - \text{DURATION}/60, \text{ if } 60 < \text{DURATION} \leq 120 \text{ ms} \\
&= 2+(\text{DURATION}-120)/100, \text{ if } 120\text{ms} < \text{DURATION}
\end{aligned}
$$

Using again our vectorial notation of the primary segmentation, and calling <u>V1MIN</u> and <u>V1MAX</u> the minimum and maximum vectors for the segment whose average vector is $\underline{V1}$ we can now precisely define the variation checking function.

For all i do steps 1-3:

1. $T1 = V1MIN_i$, if $V1MIN_i > M_i$

   $= M_i$, otherwise.

2. $T2 = V1MAX_i$, if $V1MAX_i > M_i$

   $= M_i$, otherwise.

3. If $|T1-T2| > M_i$ DO STEPS 3a, 3b, 3c

   3a. $R_i = \dfrac{|T1-T2|}{T1+T2}$

   3b. $V_i = VARLIM \times W_i \times \left(\text{if } T1+T2 \leq 10.0 \text{ then } 0.75 \text{ else } 1.0 \right)$

   3.c. if $R_i < V_i$ then keep V_i/R_i and i

Let j be defined by $V_j/R_j = \text{Max}\ (V_i/R_i)$

This j flags the most variable parameter according to our tests and is the value of the variation checking function. If there is no i such that $R_i < V_i$, the value of the procedure CHECKVARIATION is 0, indicating that all the parameters were accepted as being not variable. Algorithm 3 is an Algol representation of this procedure.

When a parameter is found to have too much variation within a sustained segment the subdivision is achieved by recomputing the closeness index between adjacent minimal segments with a modified weight vector $\underline{W}$. The feedback from the variation checking procedure to the closeness computation procedure is done using the parameter MORVARPARAM. As one can

```
INTEGER PROCEDURE CHECKVARIATION(SEGNB) ;
        INTEGER SEGNB ;

COMMENT
                THIS PROCEDURE CHECKS THE PARAMETER VARIATION IN EACH
        SUSTAINED SEGMENT AND FLAGS THE MOST VARIABLE PARAMETER .


                THE BUILT SEGMENTS ARE STORED IN SEGSTORG ARRAY, THE
        MINIMAL SEGMENTS BEING STORED IN THE ARRAY SEGIN .
                ALL THESE ARRAYS ARE DEFINED IN THE MAIN PROGRAM AS
        GLOBAL ARRAYS SEGSTORG[1:60,1:25],SEGIN[1:150,1:7] .


        IN THE FOLLOWINGS LINES ,

                AMP1MAX[,] STANDS FOR SEGSTORG[,,6] ,
                AMP3MAX[,]      "     SEGSTORG[,,18] ,
                ZEROX3[,]       "     SEGSTORG[,,20] ,
                DURATION[,]     "     SEGSTORG[,,3] ,
                ZEROX3MIN[,]    "     SEGSTORG[,,19] ,
                AMP1[,]         "     SEGSTORG[,,5] ,

                THE REAL ARRAYS WEIGHTVAR[1:6] AND LOWERLIM[1:6] ARE
        DEFINED AS GLOPAL ARRAYS IN THE MAIN PROGRAM AND FILLED WITH
        CONSTANTS AT COMPILE TIME (IF POSSIBLE, OR WHEN STARTING THE
        PROGRAM ),

                LOWERLIM[1] := 6.0 , LOWERLIM[2] := 2.0 ,
                LOWERLIM[3] := 4.0 , LOWERLIM[4] := 4.0 ,
                LOWERLIM[5] := 4.0 , LOWERLIM[6] := 10.0,

                WEIGHTVAR[1] := 1.75 , WEIGHTVAR[2] := 2.0 ,
                WEIGHTVAR[3] := 1.75 , WEIGHTVAR[4] := 2.0 ,
                WEIGHTVAR[5] := 2.0  , WEIGHTVAR[6] := 1.25;
```

Algorithm 3. Variation Checking Procedure .

```
BEGIN
        INTEGER ARRAY KEEPLARGVARINDEX[1:6] ;
        REAL ARRAY KEEPLARGVAR[1:6] ;
        REAL VARMAX,VARLIM,TEMP1,TEMP2 ;
        INTEGER KEEPINDEX,J ;
        LABEL RETURN ;


COMMENT
                FRICATIVE TYPE "S" SPECIAL CASE : IF THE SEGMENT IS A
        FRICATIVE TYPE "S", THE FOLLOWING SPECIFIC VARIATION TESTS ARE
        PERFORMED ;


        IF ZEROX3[SEGNB] ≥ 40 AND AMP1MAX[SEGNB] ≤ 7
                AND AMP3MAX[SEGNB] ≥ AMP1MAX[SEGNB]
                        THEN BEGIN IF ZEROX3MIN[SEGNB] < 30
                                        THEN CHECKVARIATION := 6
                                        ELSE CHECKVARIATION := 0 ;
                                  GO TO RETURN
                             END ;

COMMENT
    GENERAL CASE ,
        FIRST STEP :
        THE LIMITING FACTOR FUNCTION OF THE SEGMENT DURATION IS COMPUTED;

    IF DURATION[SEGNB] < 12 THEN
        IF DURATION[SEGNB] < 6 THEN VARLIM := 3.0
                    ELSE VARLIM := 4.0-DURATION[SEGNB]/6.0
                ELSE VARLIM := 2.0+(DURATION[SEGNB]-12.0)/10.0 ;

COMMENT
    SECOND STEP : THE VARIATION CORRESPONDING TO EACH PARAMETER
    OF THIS SEGMENT IS COMPUTED .
        THE 6 AVERAGE PARAMETERS ARE KEPT IN THE SEGSTORG ARRAY
    IN THE COLUMNS 5,8,11,14,17,20 ALONG WITH THEIR MINIMUM AND
    MAXIMUM VALUE WITHIN THE SEGMENT IN THE ADJACENTS COLUMNS .
        FOR EXAMPLE: AMP1 FIRST AVERAGE PARAMETER IS KEPT IN COLUMN 5
            WITH AMP1MIN IN COLUMN 4
            AND  AMP1MAX IN COLUMN 6 ;


    FOR J := 4 STEP 3 UNTIL 19 DO
        BEGIN   TEMP1 := SEGSTORG[SEGNB,J] ;
                TEMP2 := SEGSTORG[SEGNB,J+2] ;
                IF TEMP1 < LOWERLIM[J/3] THEN TEMP1 := LOWERLIM[J/3];
                IF TEMP2 < LOWERLIM[J/3] THEN TEMP2 := LOWERLIM[J/3];
                SUM := TEMP1+TEMP2 ;
                DIFF := ABS(TEMP1-TEMP2) ;
```

Algorithm 3 (continued). Variation Checking Procedure .

```
                IF DIFF ≥ LOWERLIM[J/3] THEN
                              BEGIN RATIO := DIFF/SUM ;


COMMENT
        NOW WE WILL COMPUTE THE VARIABILITY THRESHOLD CORRESPONDING
    TO THIS SEGMENT AND THIS PARAMETER, IF THE PARAMETER IS SMALL, WE
    WILL LOWER THIS THRESHOLD TO PERFORM A WEAKER TEST ;


    VARTHRESHOLD := VARLIM*WEIGHTVAR[J/3]*IF SUM ≤ 10.0 THEN 0.75
                                                       ELSE 1.0 ;


COMMENT
                PERFORM THE VARIABILITY CHECKING ;

        IF RATIO < VARTHRESHOLD THEN BEGIN
                                          KEEPINDEX := KEEPINDEX+1 ;
                                          KEEPLARGVAR[KEEPINDEX] :=
                                              VARTHRESHOLD/RATIO ;
                                          KEEPLARGVARINDEX[KEEPINDEX] := J/3 ;
                                        END
                                   END
                END JLOOP ;


COMMENT
                THIRD STEP : THE MORE VARIABLE PARAMETER FOR THIS SEGMENT
        WILL NOW BE CHOOSEN BY LOOKING IN KEEPLARGVAR ARRAY ;


        IF KEEPINDEX  0 THEN    BEGIN
                                    VARMAX := KEEPLARGVARINDEX[1] ;
                                    FOR J := 2 STEP 1 UNTIL KEEPINDEX DO
                                      IF VARMAX < KEEPLARGVAR[J] THEN
                                         VARMAX := KEEPLARGVAR[J] ;
                                    CHECKVARIATION := VARMAX ;
                                END
                        ELSE    CHECKVARIATION := 0 ;
        END;
RETURN: END CHECKVARIATION ;
```

Algorithm 3 (continued). Variation Checking Procedure .

see in the closeness procedure, the modification gives greater weight to the parameter found the most variable, which is a natural idea. On the basis of these new closeness values, the original sustained segment is replaced by two or more segments. This process is recursively repeated using the smaller segments until the variability in the segments is within acceptable limits.

Local maxima and minima of the amplitude of the waveform are phonemically significant (they usually represent significant vowels and consonants). When a phoneme is articulated for a very short period of time, it has a rapidly varying on-glide and off-glide. When closeness indices are computed for this portion of the sound, one may find that no two adjacent segments satisfy our definition of being close. Thus they may end up being part of a longer transitional segment. A special effort is made to detect and recover such extrema by searching the transitional segments. In this case, the original transitional segment is replaced by two or more segments, the local extremum being a 10 ms sustained segment. Certain very short burst segments are also recognized in the same manner. Again this process is recursively repeated until there is no longer a transitional segment containing a local extremum or a short burst. A display of the secondary segmentation can be seen in Figure III-6. At this point, the beginning and ending of the speech utterance are scanned to suppress the segments which may be considered as silence on the basis of average parameters.

III-4. COMBINING

The purpose of the combining process is to group together acoustically similar secondary segments. This task is performed in two distinct passes:

..The first one treats the transitional segments.

..The second combines similar adjacent segments.

In general, the transitional segments are null-segments (Reddy 1967b), therefore, they do not contain any pertinent information, and a special effort is made to reduce those segments as much as possible. This is done by extending the sustained segments onto the transitional if the parameters are not too different.

In order to combine secondary segments, we determine whether or not two segments are similar by using the same closeness function defined in primary segmentation. The parameters used in the closeness indice computation are now the average parameters for the secondary segments. As we are dealing with average parameters, the weights are decreased to make the procedure less sensitive to smaller variations.

Ideally we would like to combine any two adjacent segments which have similar parameters. However, it is not uncommon in speech to have two phonetically similar sounds adjacent to each other. Thus, one must be very careful in deciding whether two secondary segments can be combined. The following heuristics are useful in making that decision.

. If two segments are very close (say $c>0$) we combine them. Otherwise, we never combine a local maximum and a local minimum which are adjacent.

. If a segment is not an extremum, then it is a candidate for combining with an adjacent segment. By looking at the closeness value, one can determine whether it is closer to the preceding or the following segment. The two segments are then combined if the closeness value between the candidate and the chosen neighbor exceeds a threshold. This threshold is independent on the duration of the segment, i.e., if the segments are long, we should be reluctant to combine them.

Although it may seem easy to determine extrema in a given speech utterance, we found it a non-trivial problem because of the high accuracy required. This extrema detection procedure is used at several levels of the segmentation program. Its first use was described in the preceding section: secondary segmentation. It is also used in the combining process to prevent the combination of adjacent minimal and maximal segments. The classification procedure uses it again to label the vowels defined as local maxima of the amplitude. If this procedure is unable to detect any extremum present in a given utterance, the resulting sound description will certainly be wrong. At best, a vowel will be classified as consonant, but if the average parameters of this vowel are close to those of an adjacent segment, both will be combined and the segmentation will be totally erroneous. For example in the word "accumulate" (Figure III-8) the two vowels /u/ and the consonant /m/ have the same average amplitude and almost the same average parameters. Therefore if the first /u/ is not found to be a local maximum, the /m/ is not identified as a local minimum and the two phonemes are combined. The problem is further complicated by the fact that the extrema detection procedure should detect only the relevant extrema, i.e., be insensitive to the intra-phoneme variation of amplitude.

For the purpose of this procedure, let us define A_i, "amplitude" of

Figure III-8. Segmentation error in "ACCUMULATE". /u/ and /m/ are combined together to form one segment .

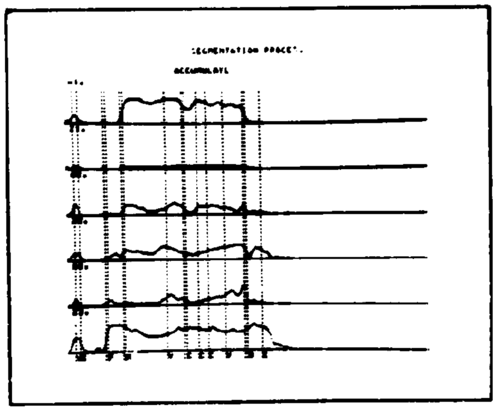


Figure III-9. Classification error in "DIRECTIVE". The first /i/ has not been marked as a significant maximum and has been classified CONST .

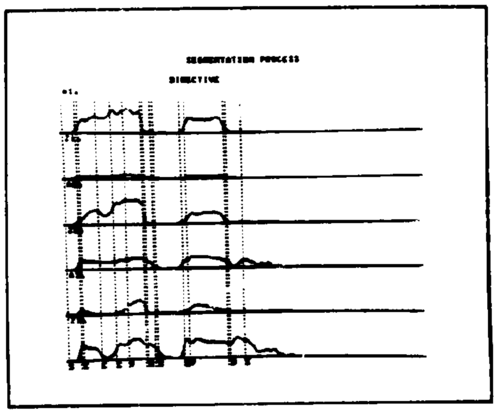
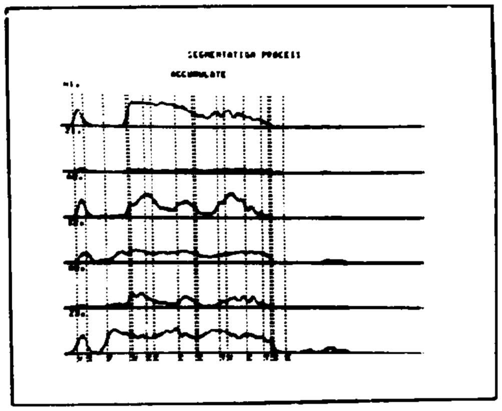


Figure III-10. Classification error in "ACCUMULATE". The second /u/ has been classified CONST .



the i^{th} segment as:

$$A_i = 2*A1_i + A2_i + A3_i/2 + DUR_i/30 \qquad \begin{array}{l} 0 \leq A1_i \leq 63 \\ 0 \leq A2_i \leq 63 \quad DUR_i \text{ in milliseconds} \\ 0 \leq A3_i \leq 63 \end{array}$$

$A1_i$, $A2_i$, $A3_i$ being the three average amplitude parameters for the i^{th} segment and DUR_i its duration. Let $\{A_i\}$ be the set of all the A_i's for the speech utterance. The j^{th} segment will be a local extremum if and only if A_j is a significant extremum of the set $\{A_i\}$. A significant extremum is defined by the following heuristics:

-A_j is significant maximum of $\{A_i\}$ if $A_{j-1} + 10 \leq A_j \geq A_{j+1} + 10$

-A_j is significant minimum of $\{A_i\}$ if $A_{j-1} - 10 \geq A_j \leq A_{j+1} - 10$

-If we have a "plateau" where the extremum is spread over several segments, the segment of longest duration will be the significant extremum.

-At the beginning and end of the speech utterance, where the sound is, in general, limited by silence, we will only look for local maxima.

This significant extremum definition was found to be effective in almost all the speech utterances we segmented. However, if the local extremum is very mild, or does not exist, the procedure may fail to detect it. The Figures III-9 and III-10 show such occasional errors.

The combining process consists of detecting and marking the local minima and maxima. Then the closeness indices between adjacent segments are computed. If two adjacent segments are found to be "close", they are combined into one segment. The average parameters and the closeness indices with the preceding and following segments are recomputed. This process is repeated until none of the adjacent segments are "close". Figure III-11

Figure III-11. Segments after combining process

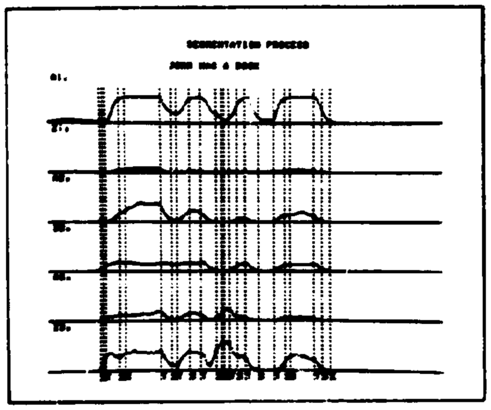


Figure III-12. Result of the segmentation procedure

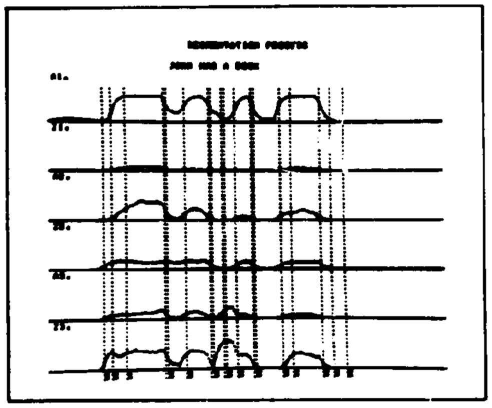
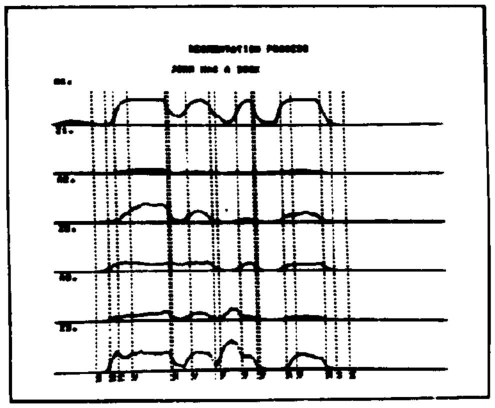


Figure III-13. Classification of segments into phoneme groups



shows a display of the segmented sound after the combining process. At this point two small segments are added at the beginning and the end of the utterance, in order to characterize the silences limiting the sound. They may be irrelevant if the message starts with a vowel, but in some cases, they represent the only chance of solving ambiguities. For example, the initial segment is the only difference between the acoustical forms of the two words: "core" and "four". In general, the preceding processes have deleted those segments as being noisy and it is necessary to add them now that the message boundaries are precisely known.

III-5. CLASSIFICATION INTO PHONEME GROUPS

The segmentation procedure consists of primary segmentation, secondary segmentation and combining. This section describes a method of assigning linguistic labels to the segments. The sustained segments are classified as belonging to a phoneme group such as fricative, vowel, stop, nasal or burst. These phoneme groups are similar to the conventional linguistic grouping of phonemes, but without the requirement that the groups be mutually exclusive. The rationale for the choice of such grouping, and a classification procedure are given by Reddy (1967a). The procedure used in the present system is simpler and oriented more towards word recognition than phoneme recognition. Since each segment is represented by its label as well as the average parameters computed during the segmentation, we do not need a precise phoneme recognizer. Another difference with Reddy's procedure is that we do not detect the null segment, i.e., segments representing a phoneme boundary which cannot be associated with any linguistic phoneme. They are labeled according to their parameter values.

If a segment is noiselike, then it is labeled FRICS. Otherwise, if the segment is a local maximum of amplitude and satisfies some specific tests, then it is labeled VOWEL. Otherwise, it is labeled STOP, NASAL or CONSONANT depending on the average parameters of the segment.

A detailed flowchart of the classification procedure is presented on Figure III-14. Since we have attempted to make this flowchart meaningful, certain tests of secondary nature have been left out so as to avoid cluttering up the drawing with details. Since a flowchart is self-explanatory, individual tests will not be described, only the vowel

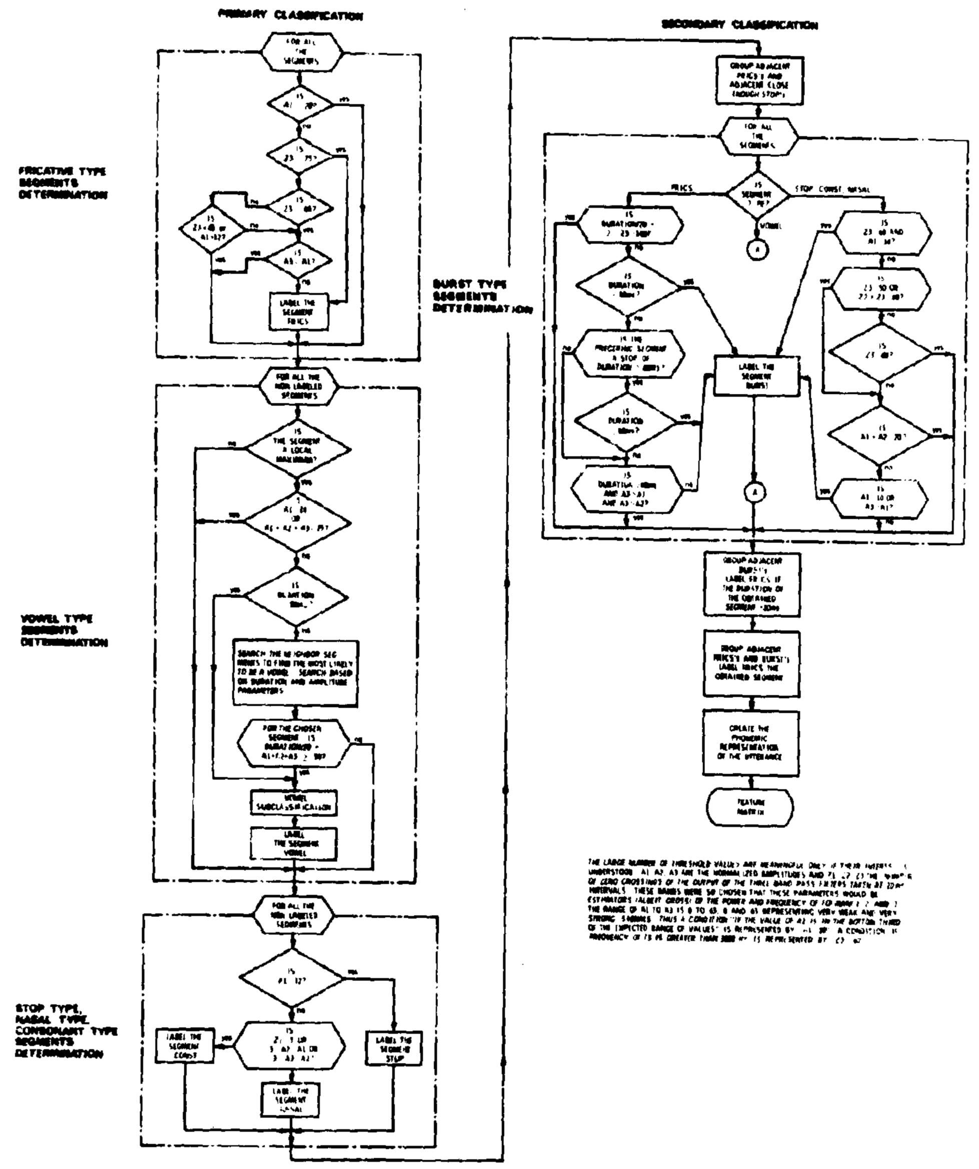

PRIMARY CLASSIFICATION
SECONDARY CLASSIFICATION
FRICATIVE TYPE SEGMENTS DETERMINATION
VOWEL TYPE SEGMENTS DETERMINATION
STOP TYPE, NASAL TYPE, CONSONANT TYPE SEGMENTS DETERMINATION
BURST TYPE SEGMENTS DETERMINATION
FOR ALL THE SEGMENTS
LABEL THE SEGMENT FRICS
FOR ALL THE NON LABELED SEGMENTS
IS THE SEGMENT A LOCAL MAXIMUM?
SEARCH THE NEIGHBOR SEGMENTS TO FIND THE MOST LIKELY TO BE A VOWEL. SEARCH BASED ON DURATION AND AMPLITUDE PARAMETERS
VOWEL SUBCLASSIFICATION
LABEL THE SEGMENT VOWEL
LABEL THE SEGMENT CONST
LABEL THE SEGMENT STOP
LABEL THE SEGMENT NASAL
GROUP ADJACENT FRICS'S AND ADJACENT CLOSE (ROUGH STOP)
LABEL THE SEGMENT BURST
GROUP ADJACENT BURST'S
GROUP ADJACENT FRICS'S AND BURST'S LABEL ONCE THE OBTAINED SEGMENT
CREATE THE PHONEMIC REPRESENTATION OF THE UTTERANCE
FEATURE MATRIX
THE LARGE NUMBER OF THRESHOLD VALUES ARE MEANINGFUL ONLY IF THEIR INTERPRETATION IS UNDERSTOOD. A1, A2, A3 ARE THE NORMALIZED AMPLITUDES AND Z1, Z2, Z3 THE NUMBER OF ZERO CROSSINGS OF THE OUTPUT OF THE THREE BAND PASS FILTERS TAKEN AT 10ms INTERVALS. THESE BANDS WERE SO CHOSEN THAT THESE PARAMETERS WOULD BE ESTIMATORS (ALBEIT GROSS) OF THE POWER AND FREQUENCY OF FORMANT 1, 2, AND 3. THE RANGE OF A1 TO A3 IS 0 TO 63, 0 AND 63 REPRESENTING VERY WEAK AND VERY STRONG SOUNDS. THUS A CONDITION "IF THE VALUE OF A1 IS IN THE BOTTOM THIRD OF THE EXPECTED RANGE OF VALUES" IS REPRESENTED BY A1 < 20. A CONDITION IF FREQUENCY OF F3 IS GREATER THAN 3000 Hz IS REPRESENTED BY Z3 > 67

Figure III-14. Flowchart of the classification Process .

subclassification procedure will be further described.

In the following chapter (the EARS system), we shall discuss several heuristics which reduce the candidate space of the lexicon search process. The conceptual ideas behind all of these is that rough features extracted from the incoming message and the stored candidates can be used to discard from the list of candidates all those with drastically different rough characteristics. The vowel categories, as intended in this and the following chapters are such rough features. The vowel segments are subclassified into nine subcategories with respect to their zero-crossing parameters: Z1 and Z2. As shown on Figure III-15, the plane (Z1, Z2) is divided into 9 regions bounded by straight lines. Each of these regions defines a vowel subclass.

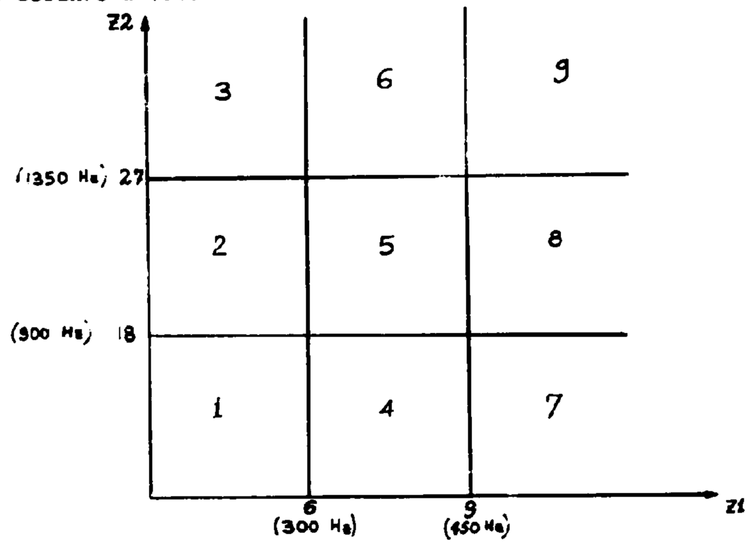


Figure III-15. Vowel Subclassification

Although the phonemes labeled during the primary classification (i.e., FRICS, VOWEL, NASAL, CONST, STOP) are similar to the conventional linguistic phonemes, the segments labeled BURST do not present the characteristics of conventional BURST (i.e., short fricative segment generally found after a STOP segment and characterizing a plosive: p, t or k). In our case, a segment is labeled BURST if it presents some of the characteristics of a FRICS but not all of them (i.e., the segment is too short, or the power of the first formant is too high, etc...). In the following chapter, we shall describe a lexicon search procedure based on the number of vowels and the number of FRICS segments of the utterance representation. In order to increase the chances of finding the correct answer, two levels of search are performed when attempting to recognize a given utterance. The first level searches the candidates with the same representation (BURST not being considered as FRICS). If no candidate matching the incoming utterance is found during this first pass, the second level of search, in attempting to find a satisfactory match, replaces these BURST labels by FRICS, along with some other label modifications on the fricative segments and the vowels.

Despite the relative simplicity of this classification algorithm, we obtained satisfactory results since we characterized the segments by their average parameters when precise information was needed. It is indeed possible to subdivide the sounds into smaller groups. However, the tendency towards erroneous grouping seems to increase in proportion of the number of groups.

III-6 REPRESENTATION OF THE UTTERANCE THE FEATURE MATRIX

The results of the previous processes are summarized in an array that we called the <u>feature matrix</u>. This feature matrix which is used for all the storing, retrieving and matching processes forms the internal representation of the speech utterance.

The first row which is utilized for a fast retrieval of the possible candidates and for the reduction of the candidate space contains general information on the utterance, namely:

-The number of the vowels.

-The number of unvoiced Fricatives.

-The number of rows (number of segments +1).

-Pointers to the segments representing the vowels.

-A rough image of the message which gives the relative position of vowels and S's (i.e., a vowel is represented by the octal number 1, a FRICS by the octal number 2).

Each subsequent row which represents a segment using label, duration and average parameters, is utilized in similarity computation.

Examples of such feature matrices for two utterances of the sound "JOHN HAS A BOOK" are shown on Figure III-16.

```
VOWNB = 4      FRICSNB = 1      ROWNB = 13

    VOWEL ROW POINTERS = 5    8   1Ø   12

      TYPE     DURATION                   SET OF PARAMETERS
                                  A1      Z1      A2      Z2      A3      Z3

      STOP       5Ø MS             3       2       Ø       Ø       1       Ø
      BURST      5Ø MS             2       Ø       2      25      11      54
      TRANS      3Ø MS            44       6      24      22       9      46
      VOWL8      7Ø MS            61      1Ø      46      19      19      51
      CONST      5Ø MS            4Ø       7      34      2Ø      21      53
      NASAL      6Ø MS            13       4       2       4       1      35
      VOWL5      8Ø MS            32       7      25      21      2Ø      56
      FRICS     1ØØ MS             1       Ø       Ø       Ø      11      77
      VOWL2      5Ø MS            26       4       7      18       4      45
      STOP       9Ø MS             4       3       Ø       Ø       1       1
      VOWL4     1ØØ MS            48       7      13      17       1      39
      STOP       5Ø MS             1       Ø       Ø       Ø       1       Ø


    CRUDE REPRESENTATION OF THE MESSAGE = 11211ØØØØØØØ
```

```
VOWNB = 4      FRICSNB = 1      ROWNB = 11

    VOWEL ROW POINTERS = 4    6    8   1Ø

      TYPE     DURATION                   SET OF PARAMETERS
                                  A1      Z1      A2      Z2      A3      Z3

      STOP       5Ø MS             3       2       Ø       Ø       1       1
      BURST      4Ø MS             2       Ø       Ø      16      13      57
      VOWL8     16Ø MS            6Ø      1Ø      57      21      28      48
      NASAL      6Ø MS            15       4       1       3       1      39
      VOWL5     11Ø MS            37       8      28      24      17      52
      FRICS      9Ø MS             2       1       Ø       Ø       8      83
      VOWL2      5Ø MS            31       4       7      19       4      45
      STOP       9Ø MS             3       3       Ø       Ø       1       Ø
      VOWL4     1ØØ MS            57       7      18      17       2      5Ø
      STOP       5Ø MS             Ø       Ø       Ø       Ø       1       1


    CRUDE REPRESENTATION OF THE MESSAGE = 11211ØØØØØØØ
```

Figure III-16. Feature Matrices for the Sound : JOHN HAS A BOOK .

III-7. RESULTS AND CONCLUSIONS

We illustrate the segmentation achieved by a number of pictures (Figures III-17 to III-23) and some timing information performed on a wide variety of speech utterances with the latest version of the program.

The figures are direct photographs of the CRT display attached to the computer and were obtained with the early program (software preprocessor). Today, it would be impossible to obtain such displays, since the original wave is no longer read into the computer. Each figure has a title, usually of the uttered sentence, the envelope of the entire utterance (on the first line), and the speech waveform (on the lines 2 - 4). The speech waveform is displayed on three lines, the third line being the continuation of the second and the fourth being the continuation of the third. The captions underneath each line indicate the segmentation obtained. Each segment begins with the first character of the caption and ends with the first character of the next caption. If the segment is a sustained segment, then the caption indicates the phoneme group to which it might belong. The beginning of a transitional segment is indicated by a single character 'T'.

Figures III-17 through III-22 show that the speech wave is segmented into parts approximately corresponding to phonemes. They also indicate the places where a segmentation scheme based primarily on the acoustic information can be expected to differ from an idealized phonemic segmentation.

-A single phoneme might be divided into two segments This is especially true at the beginning and ending of the sound (see Figures III-20 and III-22).

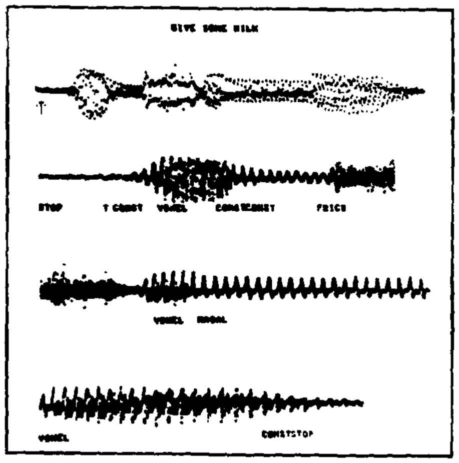


Figure III-17. Segmentation of the sound : "GIVE SOME MILK"

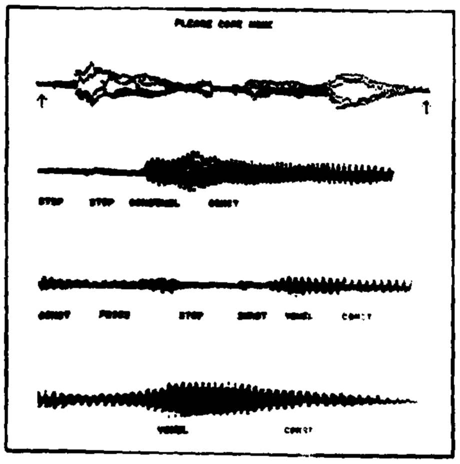


Figure III-18. Segmentation of the Sound : "PLEASE COME HOME"

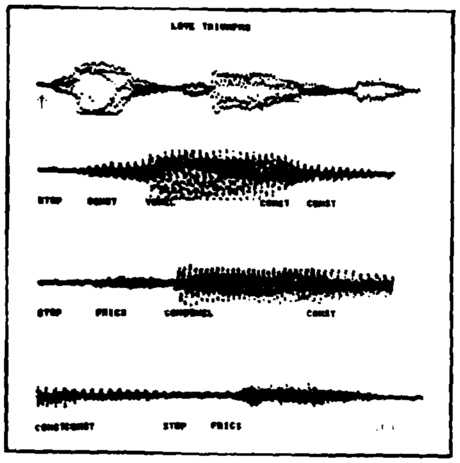


Figure III-19. Segmentation of the Sound : "LOVE TRIUMPHS"

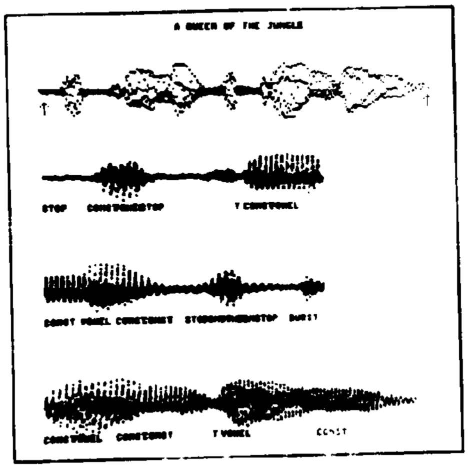


Figure III-20. Segmentation of the Sound : "A QUEEN OF THE JUNGLE"

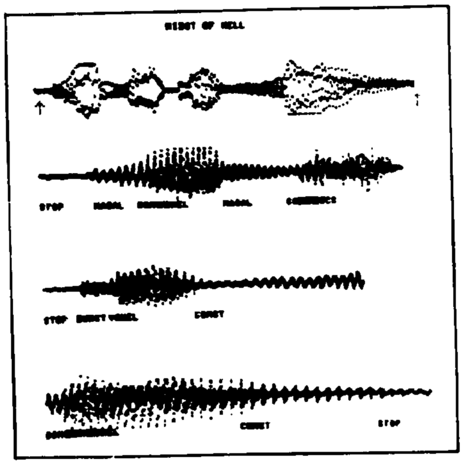


Figure III-21. Segmentation of the Sound : "MIDST OF HELL"

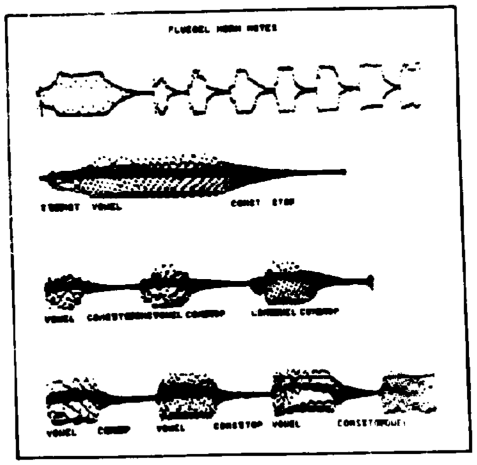


Figure III-22. Segmentation of a Set of notes using a Fluegel Horn .

-Occasionally two phonemes which are acoustically similar might be grouped into one segment. For example /r/ and /i/ in TRIUMPHS (Figure III-19).

-Phoneme group classification may not agree with conventional grouping, e.g., the syllabic /l/ in JUNGLE (Figure III-20) is best classified as a VOWEL, and /l/ and /i/ of PLEASE (Figure III-18) have opposite classification from what we would expect. These are primarily questions of classification and not segmentation, the reader is referred to Reddy (1967a) for further discussions.

The program has been tested mainly using male speakers in the noisy environment of the machine room. Good segmentation has been obtained for few utterances by female speakers. Figure III-23 shows the segmentation of a sequence of notes using a Fluegel Horn. Although it is meaningless to assign phonemic labels to musical sounds, the figure illustrates the segmentation achieved. This segmentation procedure has been used for several months as a basic tool for sound analysis, thus processing several thousands of speech utterances. The few errors that occur are usually due to speaker sloppiness or due to poor response characteristics of the microphone. Some of these errors could be eliminated, if desired, by tuning the program for a given speaker and microphone combination. As a general purpose procedure was desired (not limited to a few people or specific equipment) this tuning has been performed.

To close this chapter, let us now give some timing results performed with the latest version of the program (hardware preprocessor). Figure III-24 represents the results of a series of tests performed by the author speaking directly into the computer. The utterance durations varied

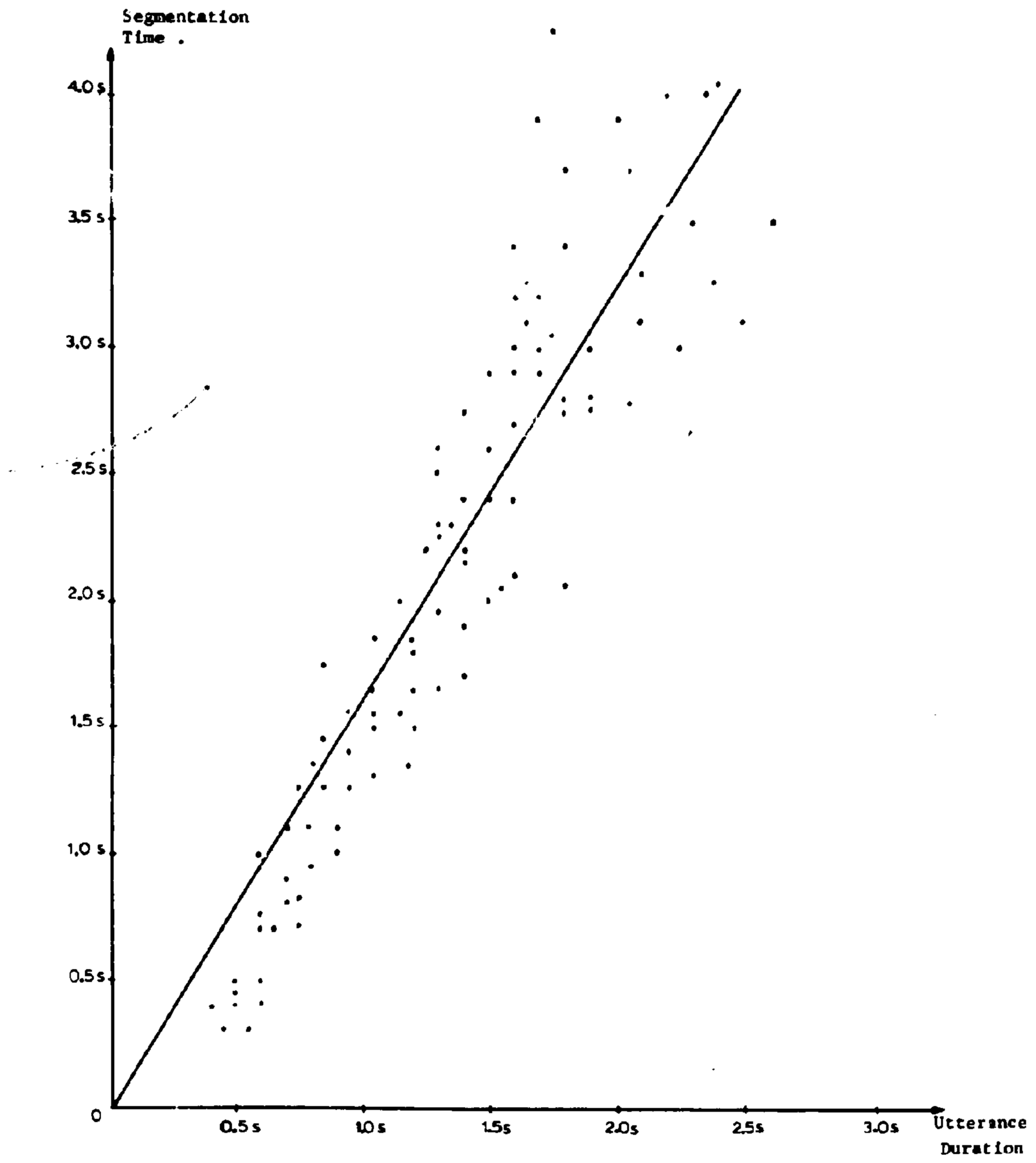


Figure III-23. Segmentation Time-vs-Utterance Duration.

from 400 ms (YES) to 3.5 seconds (PICK UP EVERY MEDIUM BLOCK STARTING AT THE BOTTOM RIGHT CORNER). By looking at the Figure, one can see that the segmentation is done in near-to-real-time (exactly 1.5 times real time). If we recall that this program is coded in FORTRAN IV, and therefore can probably be speeded up by at least a factor of 2 if we translate it in machine language, the segmentation can be performed in less than real time. This realization leads to the possibility of a real time segmentation procedure to be executed while a speaker is talking. Such a procedures was in fact programmed and is described in Chapter VI of the present dissertation.

Chapter IV

RECOGNITION OF WORDS

IV-1. INTRODUCTION

In this chapter, we will discuss the recognition of words (or phrases treated as a whole) which might be isolated or which might be part of a longer utterance. Such a system is of interest because description of connected speech utterances in terms of the words contained in them can be performed only if the system is capable of recognizing their elementary parts. In the following chapters, we shall show how the word recognition system is used in the decoding of connected speech utterances of limited languages. That is, the system is utilized to identify the terminal symbols of sentences of languages defined by two specific grammars.

The main problem to be solved in a message recognition system is the generation of a sufficiently compact representation of the messages so that retrieval and comparison of utterances can be performed with minimal effort. Representations of speech by digitized waveform or by spectral data are hardly suitable for either of the above functions. In Chapters II and III, we have seen how a sequence of transformations (i.e., pre-processing, segmentation, and sound classification) reduces the raw data into a sequence of labeled segments characterized by average parameters. This condensed representation which adequately represents an utterance by a small number of relevant parameters, was found accurate enough to allow the recognition of messages by similarity computation.

The problem of storage and data representation within a lexicon requires the consideration of the following factors:

-1. Given the structure of the message that we wish to recognize, it should be possible to associatively look up the lexicon to determine the appropriate list of candidates.

-2. The data structure should provide appropriate linkages between different acoustic descriptions of the same message, although these acoustic descriptions may have been entered in the system at different times.

-3. The data representation should be sufficiently compact so that at least a thousand word vocabulary can be conveniently handled on presently existing conventional computer systems.

The data structure presented in this chapter attempts to satisfy these requirements.

Before one can effectively recognize any message of a given language, one has to generate a lexicon of words and their acoustic descriptions which may be used for comparison purposes. Such generation may be automatic when the problems associated with sound synthesis from a phoneme string are solved. However, present attempts at synthesis of speech are inadequate for use in an effective speech recognition system. The other possibility is to let the computer generate its own descriptions based on actual human speech. In this work we use the latter method to generate the entries in the lexicon, i.e., we train the system with examples of the words and messages of the language uttered by different speakers. This permits us to essentially postpone solving the problems of speaker variability and the problems associated with the effect of context on the acoustic characteristics of a given phoneme.

Another problem to be solved is the minimization of computation so that recognition may be achieved in close to real-time. For this we need effective and efficient heuristics to perform the following functions:

-1. Selection of the probable candidates.

-2. Segment synchronization.

-3. Message similarity determination.

Selection of probable candidates takes the form of a series of procedures capable of extracting from a large lexicon a small list of highly probable candidates. The candidates are initially selected on the basis of the structure of the incoming message. Several heuristics acting on rough features of the utterances are then used to further reduce this list, and to order it so that the most probable responses are considered first. The rough features which were found useful in this candidate space reduction problem depend on vowel spectral characteristics determined for the incoming utterance and for the candidates in the candidate list.

The problem of segment synchronization is related to the fact that two utterances of the same phrase, even by the same speaker, may result in quite different acoustic descriptions. In order to evaluate the similarity between the two utterances, one must specify correspondences between their segments. Since vowels and unvoiced fricatives are more reliably detected than other segments, the synchronization procedure maps VOWEL for VOWEL and FRICS for FRICS, the remaining unmapped segments between any two pairs of mapped segments being then linked on the basis of similarity of parameters.

Given that we have to evaluate the similarity between two segments, the similarity function to be used is not unique for every segment pair. It has to be adaptive with respect to the type of the segments to be compared. For example, one must use different weighting factors when comparing two fricative segments as opposed to two vowels.

The above discussion illustrates the problems which have to be solved in the implementation of a successful word recognition system, namely: representation, learning, and recognition. Sections IV-2, IV-3, and IV-4 give the details of solutions to these problems and their implementation. Section IV-5 presents a global view of the EARS system (Effective Analyzer and Recognizer of Speech) which is an isolated word recognition system. Section IV-6 exhibits some results obtained using the system. They were obtained through the processing of several word lists, namely:

-1. List of 54 words from Gold (1966) recorded by Stevens and Williams at Bolt, Beranek and Newman, Inc. (S/N ratio > 35 db).

-2. List of 54 words from Gold recorded by ten different speakers in a noisy environment (S/N ratio ≈ 15 db).

-3. List of 70 French words and short sentences recorded by the author (S/N ratio ≈ 25 db).

-4. List of 501 English words and short sentences recorded by Singer (S/N ratio ≈ 15 db).

For ease of reading, in the following sections, we shall designate by <u>incoming message</u> the message to be recognized by the procedure.

IV-2. ORGANIZATION OF THE LEXICON

To satisfy the requirements mentioned in the introduction, the lexicon is provided with two independent list structures, namely:

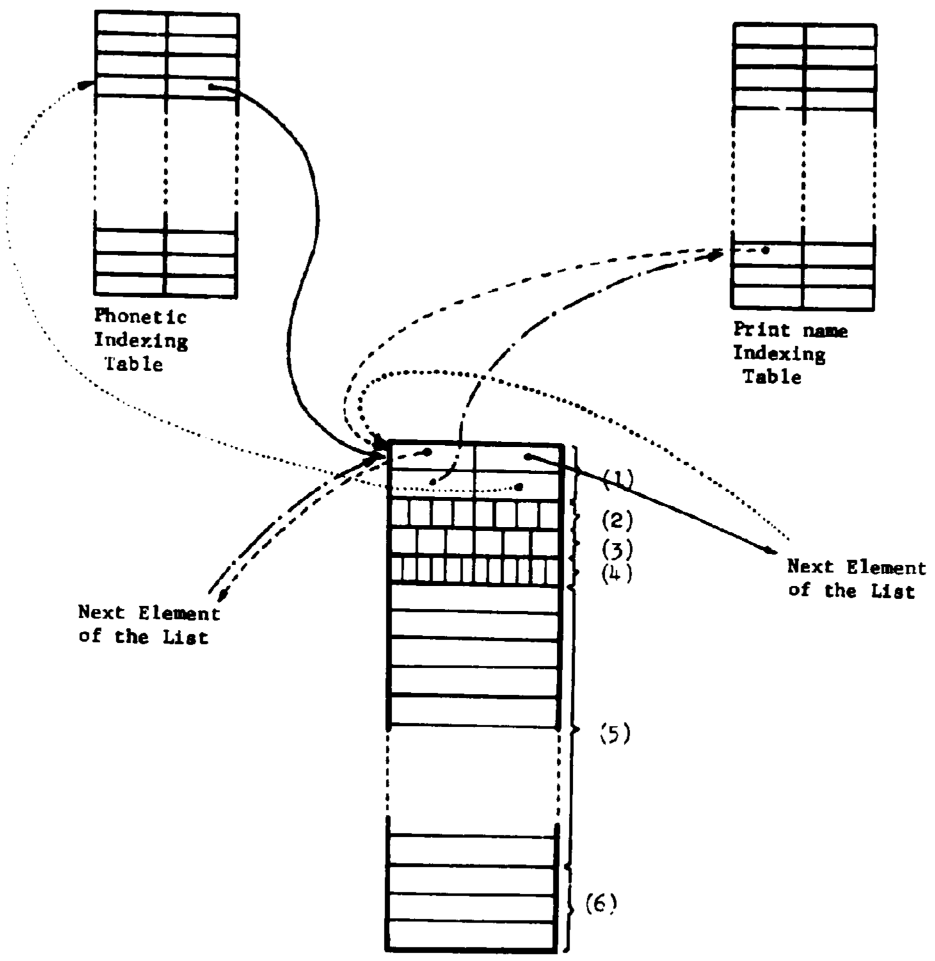
-A list structure which depends on the phonetic representation of the message (i.e., the number of vowels and the number of unvoiced fricatives).

-A list structure which depends on the print name of the message (i.e., the first character and the last character of the message print-name).

Each "learned" message forms a block of contiguous storage which is linked by forward and backward pointers to the preceding and following elements in each of the two independent lists. A typical sample of a "learned" message is shown on Figure IV-1. The block is simply formed by condensing the feature matrix of the message (Section III-7) and by appending to this packed form a header composed of four pointers and a trailer which is the message print name in its ASCII form (7 bits per character).

The first row of the feature matrix, which contains the number of vowels, the number of S's, the number of rows, pointers to the vowels and the crude message representation is packed into the storage words marked (2), (3), (4) on the drawing. The subsequent matrix rows are packed 7 bits per parameter (i.e., two 36 bit memory words per row) and stored in the area marked (5). The memory word (2) also contains information needed by the lexicon handler subroutines, namely:

-An active or inactive flag. A block marked inactive will be deleted from the lexicon when a garbage collection will be executed and



(1) Forward and backward pointers .(half memory word per pointer).
(2) Flag bits and parameters used by the lexicon handler .
(3) Vowel pointers (5 maximum) .
(4) Crude representation of the message .(i.e. relative position of vowels and S's)
(5) Compacted values list of the message phonetic representation .
(6) Print name of the message .

Backward pointer in the phonetic list .
Forward pointer in the phonetic list .
Backward pointer in the print name list .
Forward pointer in the print name list .

Figure IV-1. Typical Entry in the Lexicon .

its storage block returned to the available storage.

-The total size of the block in memory words.

Both quantities were found useful for the implementation of an efficient garbage collector.

The first element of a given list is determined by a table look-up, and the last element is signaled by a null pointer. The initial table look-up in the phonetic indexing table is done on the basis of the number of vowels and fricatives present in the pointed-to representations, namely, the index value is given by the relation:

8*VOWELNB + FRICSNB

The initial table look-up in the print-name indexing table is done on the basis of the first and last alphanumeric characters of the message (special characters and blanks are skipped during this index computation process).

Since it was decided to use Fortran IV as basic language for coding the system, the lexicon handler was implemented by a series of small Fortran compatible machine language subroutines. For maximum flexibility, one would like most of the system to be coded in the easily modifiable Fortran language. However, a certain degree of sophistication from the lexicon handler is needed to avoid wasted computation time and tedious Fortran programming.

The compromise we chose was to implement a few basic subroutines, namely:

-1. INITIAL which initializes a lexicon free-storage, i.e., resets the available storage pointer and the two indexing tables to zero.

-2. INSERT which, given the feature matrix and print-name of a message to be "learned", creates a block in the lexicon and thereby modifies both list structures.

-3. DELETE which, given the storage address of a block in the lexicon, marks this block inactive, and modifies both list structures so as to isolate this block.

-4. LISCAN which, given the feature matrix of a message to be recognized, returns a complete list of candidates with the same phonetic structure (i.e., same number and relative positions of vowels and fricatives, and vowels similar to those of the incoming message).

-5. LISSPN which, given the written representation of a message, returns a list of all stored messages having the same print-name.

-6. EXPAND which, given the storage address of a block in the lexicon, returns an unpacked feature matrix representing the stored element so that the matching process deals only with identically structured feature matrices.

Added to this Fortran compatible package are a garbage collector (automatically initiated when INSERT needs more storage) and several second order subroutines (i.e., storage comprehensive printouts, packing subroutines, etc...). The garbage collector collects all the storage blocks previously deleted and restructures the lexicon in contiguous storage, thus updating the pointers in each block. Since each block is provided with forward and backward pointers, this task is performed in one pass through the lexicon storage. When the lexicon storage area is filled up with active blocks and no garbage collection is possible, insertion requests are no longer honored; in other words, the system stops learning.

To conclude this section, let us show that this storage organization satisfies the precited requirements:

-The two independent list structures described insure a fast retrieval of previously-learned messages having a phonetic structure or print-name similar to that of the incoming message.

-The condensed form of a learned candidate block satisfies the last requirement. For example, an utterance like JOHN HAS A BOOK (Figure III-17) consists of 11 segments, and occupies thirty 36 bits words of memory. Therefore, 1000 such messages can be stored in a 30,000 memory word lexicon (on our machine the maximum possible size of the lexicon is 90,000 memory words).

IV-3. CANDIDATE LIST BUILDING PROCESS

In the following sections, we shall describe a candidate selection process which, given an utterance representation to be recognized and a list of possible candidates, chooses the candidate of best-match. As far as we can determine, all word recognizer programs previously described in the literature use such a candidate selection process for selecting the best-match candidate, but the process is applied to the entire lexicon. This method, though it has proved efficient for small vocabularies (say 50 words) becomes very inefficient when the number of messages is increased to a thousand, or more. This is because the implementation of sophisticated candidate selection procedures is paid for by large amounts of time-consuming computation. We shall first describe how the list of possible candidates can be reduced to a few entries by a small number of relevant tests applied to the acoustic structure and the vowel characteristics of the messages.

The lexicon organization previously described provides the first phase in candidate space reduction. The procedure, given the initial list from the lexicon, acts in three stages:

-1. Elimination of all candidates whose overall structures (i.e., relative positions of vowels and S's) are different than that of the incoming signal.

-2. Elimination of all the candidates with drastically-different vowel zero-crossing characteristics.

-3. Elimination of all the candidates having low vowel-similarity scores obtained by comparison to that of the incoming message.

The initial list of candidates, consists of all representations having the same number of vowels and S's as the incoming message and is obtained directly from the lexicon. However, the procedure LISCAN which performs this selection task, skips over the stored candidates having crude representations different than the incoming message one and does not enter them in the candidate list. This operation is realized by a direct comparison of the crude representation of each stored candidate (see word (2) Figure IV-1) and the crude representation of the coming message present in its feature matrix (see Figure III-16).

The second stage eliminates from the candidate list all those having drastically-different vowel characteristics and orders the list so that the most similar candidates are placed first. The necessary decisions are made on the basis of zero-crossing parameters for the vowels. In Section III-5, a procedure was described which classifies the vowels into nine subclasses according to the values of the parameters Z1 and Z2 (estimators of the Formant 1 and Formant 2 frequencies). Figure IV-2 reminds this vowel subclassification procedure. The procedure described here uses this information to modify the candidate list. To do so, it utilizes a table (Figure IV-3) which defines crude dissimilarity values between each pair of vowels on the basis of their subclass values. For example, a vowel with a subclass value of 3 and a vowel with a subclass value of 5 have a crude dissimilarity of 4. Each ● entry in the table indicates a <u>prohibited correspondence</u>, i.e., if a candidate vowel and the corresponding incoming message vowel are in prohibited correspondence, the candidate is eliminated from the initial candidate list.

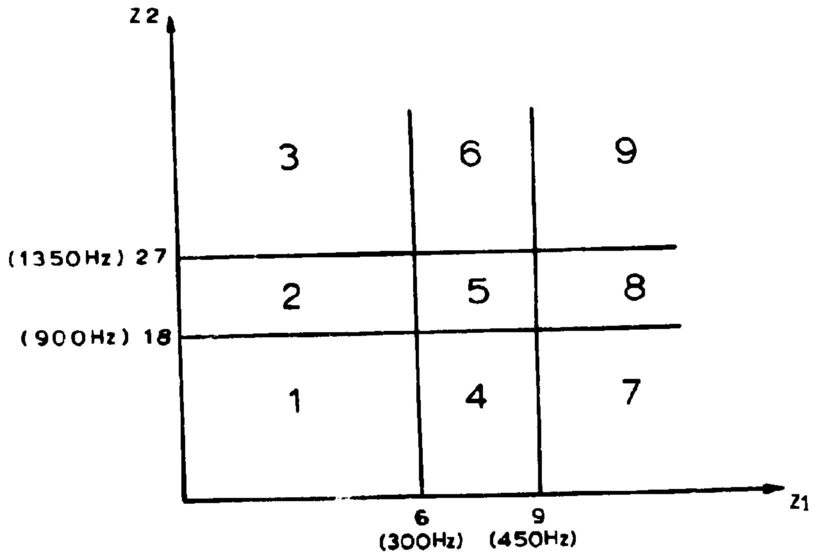


Figure IV-2. Vowel Subclasses .

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

Figure IV-3. Table defining a rough dissimilarity between vowels .

The procedure simply looks up in the table the dissimilarity values for all the incoming message vowels and the vowels of the corresponding candidate. If a prohibited correspondence is detected, the candidate is eliminated. If this does not happen, the dissimilarity values are added and form an overall dissimilarity value characterizing the candidate. This process is repeated for all the candidates in the candidate list and the list is reordered by increasing order of dissimilarity values.

For reasons of efficiency, the third attempt at the reduction of the candidate space is implemented as part of the segment synchronization procedure. It is described in detail in subsection IV-4-2. Basically, the procedure computes a similarity score between the vowels of the incoming message and the vowels of each candidate in the list, using the segment-similarity evaluation function which is described in the following sections. If this score is below a threshold, the entry is eliminated from the candidate list.

The three stages of reduction of the candidate space described above, have different strengths depending on the incoming message and on the learned vocabulary. The first stage, while very effective when the message contains several vowels and S's, is useless when the message contains no fricative. The second stage is not very effective, from the standpoint of candidate space reduction (on the average only 20 percent of the candidates are eliminated). However, the simplicity of the algorithm and the fact that it orders the candidate list by decreasing similarity makes it effective in reducing the average computer time needed to recognize an utterance. The third stage is extremely effective and eliminates an average of 60 percent of the selected candidates.

If the candidate list becomes empty at any step, the procedure returns a failure.

IV-4. CANDIDATE SELECTION PROCESS

The previous sections have described how a small list of acceptable candidates can be obtained from a large, carefully organized lexicon by efficient heuristic procedures. This section discusses how a unique identification of an incoming message can be derived from this list of acceptable candidates by similarity computation.

Several problems which have not been clearly explored in the previously mentioned literature have to be solved before one can implement an efficient selection procedure based on similarity computation:

-1. Given two utterances to be matched, one has to determine correspondences between the elements of the two utterance representations. Of course, segment synchronization is not a problem if the vocabulary merely consists of short monosyllabic or dissyllabic words, such as the ten digits used for many previous experiments in speech recognition. On the other hand, if a system capable of efficiently recognizing utterances containing several syllables is desired, it becomes crucial to attempt to solve this problem. The strategy we use employs a "similarity evaluation function" to link segments produced and classified by previous stages in the recognition process.

-2. Given two utterances to be matched and correspondences between their components, one has to determine a similarity measure to evaluate the closeness between the corresponding elements.

-3. Since the lexicon is organized on the basis of the number of vowel and fricative segments, a classification error in the incoming utterance results in an erroneous list of possible candidates (i.e., the "correct" utterance is not included) and consequently in a failure of the recognition process. To attain good recognition, one has to attempt to recover from such a situation. The procedure we have implemented checks the incoming message representation to detect borderline cases of vowels and unvoiced fricatives, modifies the incoming message representation with respect to these borderline cases and initiates new searches of the lexicon.

The present section, which is the most important of the chapter, is divided into four subsections:

-1. Overall description of the candidate selection process.

-2. Segment synchronization procedure.

-3. Similarity evaluation procedure.

-4. Error recovery procedure.

IV-4-1. Overall description of the candidate selection process

A detailed flowchart of the candidate selection process is shown on Figure IV-4. The procedure BUILDLIST searches the lexicon, and computes a "similarity" between the incoming message and all the entries in the acceptable candidate list. This similarity computation is performed first by calling the segment synchronization procedure which creates linkages between the segments of the two representations to be matched and then by averaging the similarity values obtained for each pair of linked segments. The results of this computation are stored for the selection process which chooses the best-match candidate. If one of the candidates obtains a score greater than or equal to 95 percent, the process immediately stops and returns the candidate print-name (excellent-match-candidate heuristic). Since the initial list of candidates is ordered on the basis of the similarity of vowel parameters

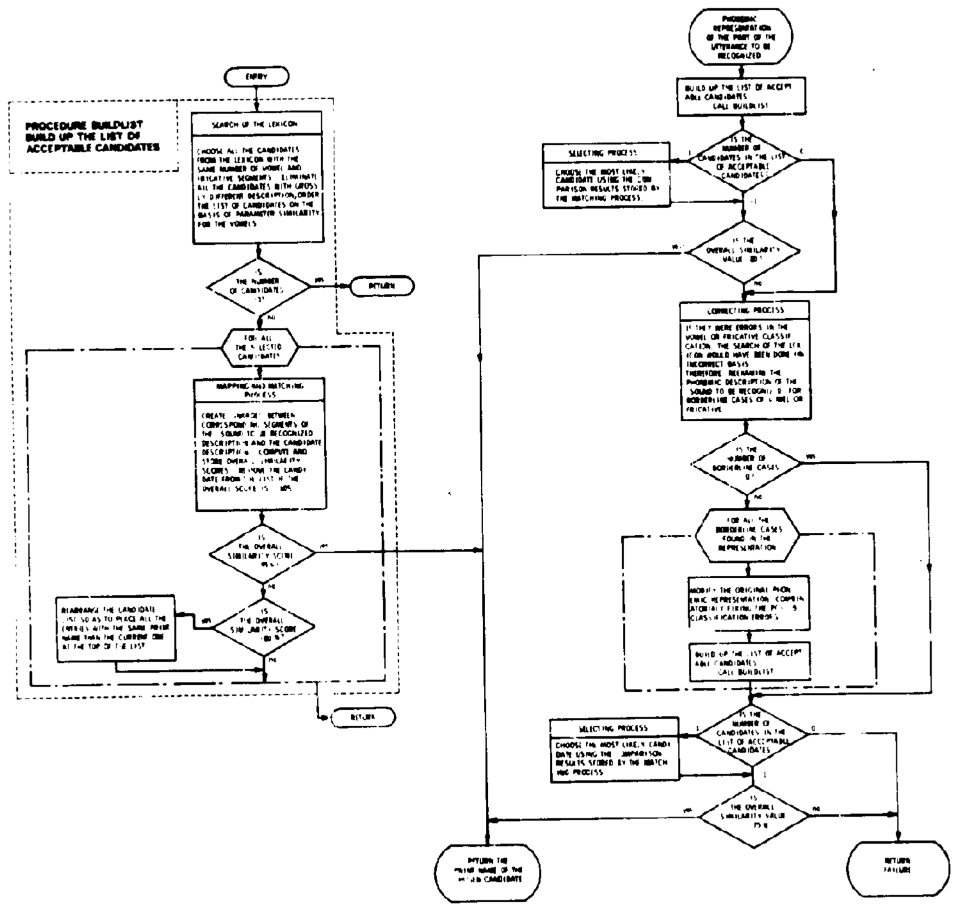

ENTRY
PROCEDURE BUILDLIST
BUILD UP THE LIST OF ACCEPTABLE CANDIDATES
SEARCH UP THE LEXICON
CHOOSE ALL THE CANDIDATES FROM THE LEXICON WITH THE SAME NUMBER OF VOWEL AND FRICATIVE SEGMENTS. ELIMINATE ALL THE CANDIDATES WITH GROSSLY DIFFERENT DESCRIPTION, ORDER THE LIST OF CANDIDATES ON THE BASIS OF PARAMETER SIMILARITY FOR THE VOWELS
YES
RETURN
NO
FOR ALL THE SELECTED CANDIDATES
MAPPING AND MATCHING PROCESS
YES
NO
REARRANGE THE CANDIDATE LIST SO AS TO PLACE ALL THE ENTRIES WITH THE SAME PRINT NAME THAN THE CURRENT ONE AT THE TOP OF THE LIST
RETURN
PHONEMIC REPRESENTATION OF THE PART OF THE UTTERANCE TO BE RECOGNIZED
BUILD UP THE LIST OF ACCEPTABLE CANDIDATES CALL BUILDLIST
IS THE NUMBER OF CANDIDATES IN THE LIST OF ACCEPTABLE CANDIDATES?
SELECTING PROCESS
CHOOSE THE MOST LIKELY CANDIDATE USING THE COMPARISON RESULTS STORED BY THE MATCHING PROCESS.
CORRECTING PROCESS
IF THEY WERE ERRORS IN THE VOWEL OR FRICATIVE CLASSIFICATION, THE SEARCH OF THE LEXICON WOULD HAVE BEEN DONE ON INCORRECT BASIS. THEREFORE REEXAMINE THE PHONEMIC DESCRIPTION OF THE SOUND TO BE RECOGNIZED FOR BORDERLINE CASES OF VOWEL OR FRICATIVE
FOR ALL THE BORDERLINE CASES FOUND IN THE REPRESENTATION
BUILD UP THE LIST OF ACCEPTABLE CANDIDATES CALL BUILDLIST
SELECTING PROCESS
CHOOSE THE MOST LIKELY CANDIDATE USING THE COMPARISON RESULTS STORED BY THE MATCHING PROCESS
RETURN FAILURE

Figure IV-4. Flowchart of the Candidate Selection Process .

(i.e., vowel subclasses, Section IV-3), this high score, if it occurs at all, is likely to happen early in the search, thus saving a large amount of computation time. As the process continues, modifications of the initial candidate list take place: each time a quite good similarity score is obtained (≥80%), the list is rearranged so as to place next-in-order all entries having the same print-name. In doing so, we assume that if a candidate obtains a score of 80%, it is likely that one of the candidates having the same print-name will obtain 95% or more.

In normal message identification, this procedure is called each time a new representation of the utterance is built. The initial representation is, of course, the representation determined through the segmentation process. However, since the error recovery procedure changes the original representation of the utterance, new calls of this BUILDLIST procedure might be necessary to investigate other parts of the lexicon.

The selecting process, which is utilized when no candidate with a very high similarity score is found, is a simple algorithm acting on the scores stored by the matching process. Each candidate left in the candidate list at this point, is characterized by three scores:

-1. Similarity score for vowels.

-2. Similarity score for non-vowels.

-3. Overall similarity score.

On the basis of these three numerical values, the selecting process chooses the best-match candidate. The first decision is made on the basis of the overall similarity scores. If the overall scores of several candidates are close, a second decision is made based on the vowel scores. If several candidates are still left after this second stage, the algorithm considers the non-vowel scores. If there is more than one candidate still present in the set of possible responses, the candidate with the best overall score is finally chosen. Of course, the process terminates at any stage when the number of considered candidates reduces to 1. The print-name of the chosen candidate is returned as the recognition response, provided it satisfies the acceptability criterion (overall score ≥75%).

IV-4-2. Segment synchronization procedure

A detailed flowchart of the segment synchronization procedure is given on Figure IV-5. Moreover, the process is illustrated by a set of CRT photographs which shows its different stages (Figure IV-6 and Figures IV-12 to IV-16).

Since unvoiced fricative segments and vowels can usually be more reliably detected than other phoneme classes, the synchronization procedure first maps vowel to vowel and fricative to fricative. However, it is not uncommon that if the vowel is preceded or followed by a high consonant sound like /r/ or /l/, this consonant is incorrectly classified VOWEL, thus creating a mislinkage at this early state of the mapping. The same mislinkage may occur if the vowel is a diphthong, in which case, either part of it can be classified VOWEL. To correct this defect, a procedure redefines the links on the basis of similarity of parameters between the linked vowel segments and the consonant segments adjacent to them. The similarity of parameters between segments is defined by the similarity function which is described in the following subsection. Figure IV-6 illustrates this vowel mapping correction procedure. The messages to be matched are two different utterances of: A QUEEN OF THE

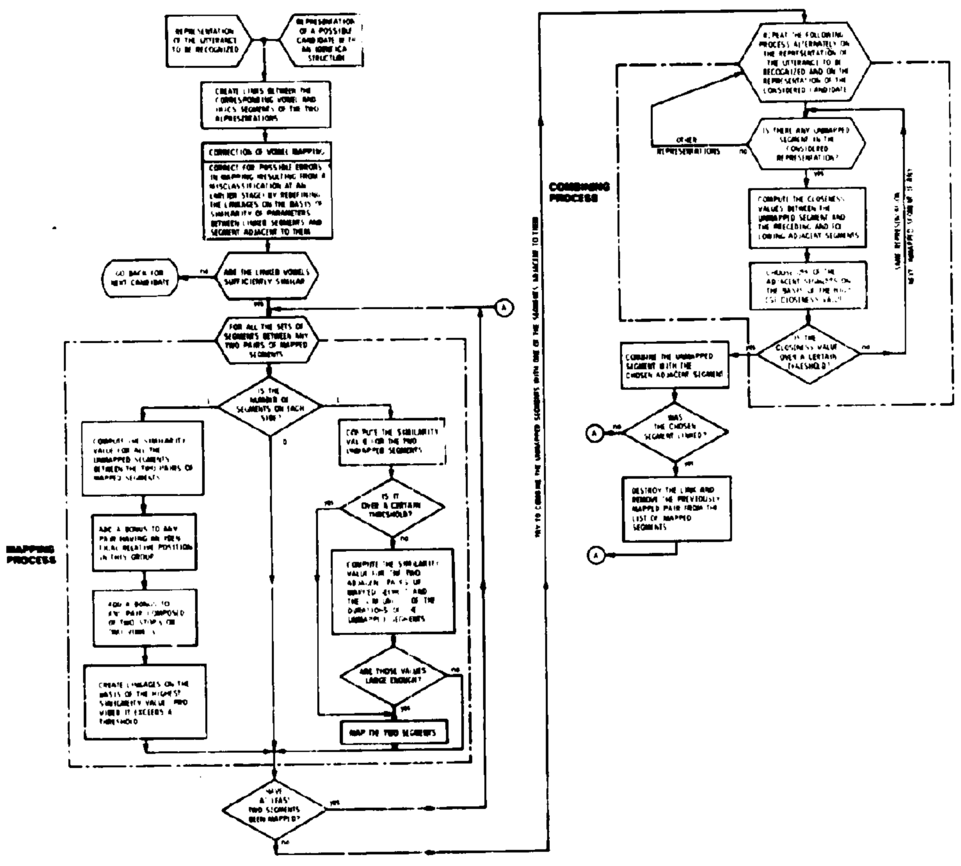
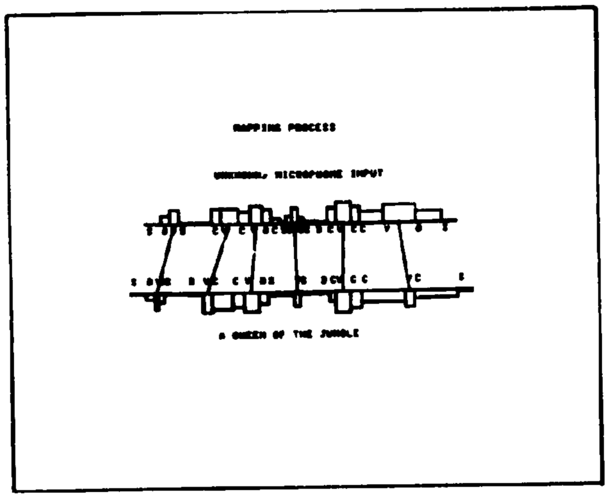

REPRESENTATION OF THE UTTERANCE TO BE RECOGNIZED
REPRESENTATION OF A POSSIBLE CANDIDATE WITH AN IDENTICAL STRUCTURE
CREATE LINKS BETWEEN THE CORRESPONDING VOWEL AND HICS SEGMENTS OF THE TWO REPRESENTATIONS
CORRECTION OF VOWEL MAPPING
CORRECT FOR POSSIBLE ERRORS IN MAPPING RESULTING FROM A MISCLASSIFICATION AT AN EARLIER STAGE BY REDEFINING THE LINKAGES ON THE BASIS OF SIMILARITY OF PARAMETERS BETWEEN LINKED SEGMENTS AND SEGMENT ADJACENT TO THEM
ARE THE LINKED VOWELS SUFFICIENTLY SIMILAR
no
GO BACK FOR NEXT CANDIDATE
yes
A
FOR ALL THE SETS OF SEGMENTS BETWEEN ANY TWO PAIRS OF MAPPED SEGMENTS
IS THE NUMBER OF SEGMENTS ON EACH SIDE?
1
1
0
COMPUTE THE SIMILARITY VALUE FOR ALL THE UNMAPPED SEGMENTS BETWEEN THE TWO PAIRS OF MAPPED SEGMENTS
ADD A BONUS TO ANY PAIR HAVING AN IDENTICAL RELATIVE POSITION IN THIS GROUP
ADD A BONUS TO ANY PAIR COMPOSED OF TWO STOPS OR TWO VOWELS
CREATE LINKAGES ON THE BASIS OF THE HIGHEST SIMILARITY VALUE PROVIDED IT EXCEEDS A THRESHOLD
COMPUTE THE SIMILARITY VALUE FOR THE TWO UNMAPPED SEGMENTS
IS IT OVER A CERTAIN THRESHOLD?
yes
no
COMPUTE THE SIMILARITY VALUE FOR THE TWO ADJACENT PAIRS OF MAPPED SEGMENTS AND THE DURATIONS OF THE UNMAPPED SEGMENTS
ARE THOSE VALUES LARGE ENOUGH?
no
yes
MAP THE TWO SEGMENTS
MAPPING PROCESS
HAVE AT LEAST TWO SEGMENTS BEEN MAPPED?
yes
no
TRY TO COMBINE THE UNMAPPED SEGMENTS WITH ONE OF THE SEGMENTS ADJACENT TO THEM
COMBINING PROCESS
REPEAT THE FOLLOWING PROCESS ALTERNATELY ON THE REPRESENTATION OF THE UTTERANCE TO BE RECOGNIZED AND ON THE REPRESENTATION OF THE CONSIDERED CANDIDATE
IS THERE ANY UNMAPPED SEGMENT IN THE CONSIDERED REPRESENTATION?
OTHER REPRESENTATIONS
no
yes
COMPUTE THE CLOSENESS VALUES BETWEEN THE UNMAPPED SEGMENT AND THE PRECEDING AND FOLLOWING ADJACENT SEGMENTS
CHOOSE ONE OF THE ADJACENT SEGMENTS ON THE BASIS OF THE HIGHEST CLOSENESS VALUE
SAME REPRESENTATION NEXT UNMAPPED SEGMENT IF ANY
IS THE CLOSENESS VALUE OVER A CERTAIN THRESHOLD?
yes
no
COMBINE THE UNMAPPED SEGMENT WITH THE CHOSEN ADJACENT SEGMENT
WAS THE CHOSEN SEGMENT LINKED?
no
A
yes
DESTROY THE LINK AND REMOVE THE PREVIOUSLY MAPPED PAIR FROM THE LIST OF MAPPED SEGMENTS
A

Figure IV-5. Flowchart of the Segment Synchronization Process .



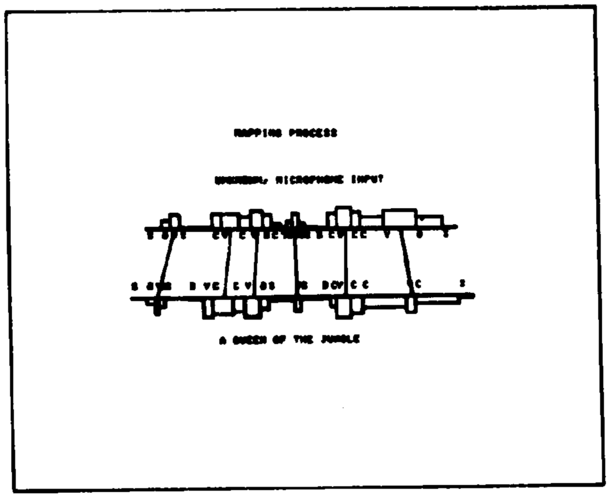


Figure IV-6. The Vowel Mapping Correction Procedure .

JUNGLE. The first photograph presents the status of the mapping links just before the correcting procedure. As one can easily see, the second link, which is built on the basis of the segment labels, is incorrect. It corresponds to the diphthong /wi/; in one utterance /w/ was labeled VOWEL, in the other /i/. The second photograph shows the status of the mapping links after the correcting process. On the basis of similarity of parameters, the second link has been modified, and is now correct. A crude evaluation of the overall contribution of this heuristic to the quality of the recognition will be given in the section IV-7.

The mapping procedure continues by computing a similarity score between all the now correctly mapped vowels. If the obtained score is below a certain threshold (i.e., score <0), the candidate is eliminated from the candidate list. This is the third computation-time saving candidate space reduction procedure previously mentioned (Section IV-3)

The program proceeds by mapping the segments between any two pairs of mapped segments on the basis of parameter similarity as shown on the flowchart. This process is recursively repeated until the program cannot effect any more mapping. The few remaining unmapped segments are then candidates for combining with their preceding or following segments. Since these second-order combinations are in general degrading both representations to be matched, one must be very careful when applying them. The closeness index between segments is computed using the closeness function defined in the segmentation procedure (Section III-2). On the basis of the closeness values between the unmapped segment and its adjacent segments, the closer one is chosen; if the closeness value between the unmapped segment and the chosen segment is high enough, then a combination occurs. These combinations are done one at a time and in a parallel manner on both representations; that is, each representation is alternatively considered. Each time a combination occurs, the mapping process is reentered in an attempt to map the segment result of the combining. This mapping - combining process is recursively repeated until no more combining or mapping can be performed. The overall similarity evaluation procedure is then executed.

IV-4-3 Similarity evaluation procedure

The similarity evaluation procedure computes several similarity scores on the two representations to be compared, namely:

- Vowel similarity score.
- Non-vowel similarity score.
- Overall similarity score.

The overall similarity score is computed from the two preceding scores, weighted by the relative duration of vowels and non-vowels in the two utterance representations. Likewise, the contribution of each segment to the corresponding score is proportional to its duration. To achieve this task, the similarity evaluation procedure utilizes an elementary similarity function which calculates the similarity between a pair of segments, one belonging to the incoming message representation, and the other to the stored candidate representation. This function is also used by the mapping process of the segment synchronization procedure in deciding which segments have to be linked between any two pairs of already linked segments. As in the segmentation procedure, in which we had to define a comparable closeness function, a simple-minded metric proves to be unsatisfactory. The requirements of this new similarity function are the same as those of the segmentation closeness function. Therefore, since the basic closeness function $k \frac{\Delta y}{\sqrt{y}}$ has given satisfactory results in defining the closeness between adjacent segments of the same utterance, we decided to use it here. Moreover, this similarity function has a supplementary requirement: It cannot be unique for all the pairs of linked segments. Since the segments to be compared have different characteristics, this similarity evaluation function has to be adaptive with respect to the type of segments to be matched. For example, one must use different weights when comparing two vowels in which the amplitude parameters, Z1 and Z2 (estimates a formant 1 and formant 2 frequencies) are the important factors, as opposed to the matching of two fricatives in which A3 and Z3 (estimates for formant 3) are the important factors.

The weights were so chosen that the value of the segment similarity procedure is between -1000 and 100. An Algol equivalent to this similarity evaluation procedure is given (Algorithm 4) The various weights present in the procedure were heuristically determined through the consideration of a large set of specific examples.

IV-4-4 Error recovery procedure

Since the search of the lexicon is executed on the basis of the number of vowels and number of fricative segments (FRICS), a classification error, at an earlier stage, would have caused an incorrect lexicon search. In order to correct this defect, we implemented an error recovery procedure. This procedure takes the form of a series of secondary searches which are only initiated if no satisfact ry candidate is found during the primary speech. To do so, the incoming message representation is examined for borderline cases of vowels (a sonorant

```
INTEGER PROCEDURE SIMILAREVAL(SEGNB1,SEGNB2);
        INTEGER SEGNB1,SEGNB2;

COMMENT
        THIS PROCEDURE EVALUATES THE SIMILARITY BETWEEN THE TWO
        SEGMENTS:
        SEGNB1 BELONGING TO THE COMING MESSAGE REPRESENTATION AND
        SEGNB2 BELONGING TO THE STORED CANDIDATE REPRESENTATION ;

        BEGIN
        INTEGER PROCEDURE SCORE(PARAM1,PARAM2,WEIGHT,INFLIM,RATIOLIM);
                INTEGER PARAM1,PARAM2,INFLIM;
                REAL WEIGHT,RATIOLIM;

COMMENT
        THIS PROCEDURE EVALUATES THE SIMILARITY BETWEEN THE TWO NUMBERS
        PARAM1 AND PARAM2, THE WEIGHTING FACTORS AND THE LIMITS ARE
        DEFINED BY WEIGHT,RATIOLIM AND INFLIM ;

        BEGIN   INTEGER TEMP1,TEMP2,DIFF;
                REAL RATIO;

                TEMP1 ::= IF PARAM1 > INFLIM THEN PARAM1
                                             ELSE INFLIM ;
                TEMP2 ::= IF PARAM2 > INFLIM THEN PARAM2
                                             ELSE INFLIM ;
                DIFF ::= IABS(TEMP1-TEMP2) ;
                IF DIFF ≤ INFLIM THEN SCORE ::= 100
                                 ELSE
                BEGIN RATIO ::= WEIGHT*DIFF/SQRT(TEMP1+TEMP2) ;
                      IF RATIO > RATIOLIM THEN SCORE ::= -200
                                          ELSE
                        BEGIN SCORE ::= 110.0*(1.0-RATIO) ;
                              IF SCORE > 100 THEN SCORE ::= 100 ;
                              IF SCORE < -200 THEN SCORE ::= -200 ;
                        END
                END
        END SCORE ;

COMMENT
        IN THE FOLLOWING LINES, THE COMING MESSAGE PARAMETERS ARE
        CHARACTERIZED BY A SUBSCRIPT 1  (I.E. A31,TYPE1,DUR1)
        THE STORED CANDIDATE PARAMETERS BY A SUBSCRIPT 2
        (I,E, A32,TYPE2) , SINCE EACH SEGMENT IS CHARACTERIZED BY
        8 PARAMETERS, THE PROCEDURE DEALS WITH 16 PARAMETERS :

        TYPE1,TYPE2      LABELS OF THE CONSIDERED SEGMENTS

                         0 DEFINES A  TRANSITION
                         1      "     CONST
                         2      "     NASAL
                         3      "     STOP
                         4      "     BURST
                         5      "     FFICS
                         6-14   "     VOWEL  ( 9 SUBCLASSES )
```

Algorithm 4. Similarity Evaluation Procedure

```
        DUR1,DUR2          SEGMENT DURATIONS
        A11,A12
        A21,A22            AVERAGE AMPLITUDES FOR THE THREE FREQUENCY
                                                          BANDS .
        A31,A32
        Z11,Z12
        Z21,Z22            AVERAGE ZERO-CROSSINGS FOR THE THREE FREQUENCY
                                                          BANDS .
        Z31,Z32

        IN THIS PROCEDURE, WE ASSUME THAT THESE PARAMETERS ARE DEFINED
        AS GLOBAL PARAMETERS ( ARRAYS ) IN THE MAIN PROGRAM .
        THE SWITCH DECMAT, WHICH IS DESCRIBED NEXT MAKES THE FUNCTION
        ADAPTIVE WITH RESPECT TO THE SEGMENT LABELS ;

        LABEL    CONSTCONST,CONSTNASAL,CONSTSTOP,CONSTBURST,PROHIBITED,
                 NASALCONST,NASALNASAL,NASALSTOP,NASALBURST,NASALVOWEL,
                 STOPCONST,STOPNASAL,STOPSTOP,STOPBURST,STOPFRICS,
                 BURSTCONST,BURSTNASAL,BURSTSTOP,BURSTBURST,BURSTFRICS,
                 FRICSSTOP,FRICSBURST,FRICSFRICS,CONSTVOWEL,BURSTVOWEL,
                 VOWELCONST,VOWELNASAL,VOWELBURST,VOWELVOWEL ,
                 FINSIMILAREVAL,SONORANTSONORANT ;


         SWITCH DECMAT ::=
                 CONSTCONST,CONSTNASAL,CONSTSTOP,CONSTBURST,PROHIBITED,
                 CONSTVOWEL,NASALCONST,NASALNASAL,NASALSTOP,NASALBURST,
                 PROHIBITED,NASALVOWEL,STOPCONST,STOPNASAL,STOPSTOP,
                 STOPBURST,STOPFRICS,PROHIBITED,BURSTCONST,BURSTNASAL,
                 BURSTSTOP,BURSTBURST,BURSTFRICS,BURSTVOWEL,PROHIBITED,
                 PROHIBITED,FRICSSTOP,FRICSBURST,FRICSFRICS,PROHIBITED,
                 VOWELCONST,VOWELNASAL,PROHIBITED,VOWELBURST,PROHIBITED,
                 VOWELVOWEL ;


          INTEGER LIMIT,INFLIM1,INFLIM2;
          REAL    WEIGHT1,WEIGHT2,RATIOLIM1,RATIOLIM2,FACT;


COMMENT
          STARTING POINT OF THE PROCEDURE
                  INITIAL SWITCHING   ;

          GO TO DECMAT [6*MIN(6,MAX(1,TYPE1[SEGNB1]))+
                         MIN(6,MAX(1,TYPE2[SEGNB2]))-7] ;


STOPSTOP:STOPBURST:BURSTSTOP:
       SIMILAREVAL:=SCORE(DUR1[SEGNB1],DUR2[SEGNB2],0.625,2,1.5)
                   +SCORE(A11[SEGNB1],A12[SEGNB2],0.625,4,1.5)
                   +SCORE(A21[SEGNB1],A22[SEGNB2],0.625,4,1.5) ;
```

Algorithm 4 (continued). Similarity Evaluation Procedure

```
COMMENT
        IF FORMANT3 AMPLITUDE (AVERAGE AMPLITUDE IN THE THIRD
        FREQUENCY BAND) IS TOO SMALL, THE CORRESPONDING ZERO-CROSSING
        IS ERRATIC AND CONSEQUENTLY MUST NOT BE TAKEN AS ONE OF THE
        FACTORS DEFINING THE SIMILARITY BETWEEN THE TWO SEGMENTS ;

        IF A31[SEGNB1]+A32[SEGNB2] ≤ 6    THEN
                BEGIN   SIMILAREVAL ::= SIMILAREVAL/3 ;
                        GO TO FINSIMILAREVAL
                END
                                          ELSE
                BEGIN   SIMILAREVAL::=(SIMILAREVAL+
                        SCORE(Z31[SEGNB1],Z32[SEGNB2],0,625,4,1,5))/4 ;
                        GO TO FINSIMILAREVA
                END ;


BURSTBURST:FRICSSTOP:STOPFRICS:

        SIMILAREVAL ::=
                MAX(40,(SCORE(DUR1[SEGNB1],DUR2[SEGNB2],0,25,2,1,5)
               +MAX(25,(SCORE(Z11[SEGNB1],Z12[SEGNB2],0,625,2,1,5))
               + SCORE(A11[SEGNB1],A12[SEGNB2],0.625,2,1,5)
               + SCORF(Z31[SEGNB1]/2,Z32[SEGNB2]/2,0.625,4,1,5)
               + SCORF(A31[SEGNB1],A32[SEGNB2],0.625,2,1,5))/5);
        GO TO FINSIMILAREVAL ;
PROHIBITED:

        SIMILAREVAL ::= -1000 ;
        GO TO FINSIMILAREVAL ;


FRICSBURST:BURSTFRICS:FRICSFRICS:

        SIMILAREVAL ::= SCORE(DUR1[SEGNB1],DUR2[SEGNB2],0.625,2,1,5);
        IF SIMILAREVAL < 50 THEN LIM ::= 20
                              ELSE LIM ::= 50 ;
        SIMILAREVAL ::= MAX(LIM,(SIMILAREVAL
               +2*SCORE(Z31[SEGNB1]/2,Z32[SEGNB2]/2,0,625,4 4,0)
               +SCORE(A11[SEGNB1],A12[SEGNB2],0,625,4,4,0]
               +SCORE(A31[SEGNB1]/2,A32[SEGNB2]/2,0 625,2,4,0))/5) ;
        GO TO FINSIMILAREVAL ;

COMMENT
        IF THE AMPLITUDES A11 AND A12 ARE VERY DISSIMILAR, THE TWO
        MATCHED SEGMENTS SHOULG GET A LOW SIMILARITY SCORE, EVENTHOUGH
        THE OTHER PARAMETERS ARE CLOSE .
        NASAL AND STOP SEGMENTS PRESENT THESE CHARACTERISTICS WHEN
        MATCHED, I.E. ONLY THE PARAMETER A1 IS DIFFERENT ;

STOPNASAL:STOPCONST:CONSTSTOP:NASALSTOP:

        SIMILAREVAL ::= SCORE(A11[SEGNB1],A12[SEGNB2],0,625,10,2,[illegible] ;
        IF SIMILAREVAL < 0  THEN GO TO FINSIMILAREVAL
                             ELSE GO TO STOPSTOP ;
```

Algorithm 4 continued). Similarity Evaluation Procedure

```
COMMENT
        IF A11 AND A12 ARE VERY DISSIMILAR THE SIMILARITY SCORE SHOULD
        BE LOW, INDEPENDENTLY OF THE OTHER PARAMETERS ,
        IF THEY ARE AT THE SAME LEVEL, THEN IF THEY ARE SMALL
        THEN THE COMBINATION IS LIKELY TO BE A BURST-BURST TYPE
        IF THE AMPLITUDE PARAMETERS ARE HIGH THEN IT IS A CONST-CONST ;

CONSTBURST:NASALBURST:BURSTNASAL:BURSTCONST:VOWELBURST:BURSTVOWEL:

        SIMILAREVAL ::= SCORE (A11[SEGNB1],A12[SEGNB2],0.650,2,1.5) ;
        IF SIMILAREVAL < 0  THEN GO TO FINSIMILAREVAL
                            ELSE
        IF MAX(A11[SEGNB1],A12[SEGNB2]) ≥ 4 THEN GO TO CONSTCONST
                                            ELSE GO TO BURSTBURST ;


NASALNASAL:

        IF DUR1[SEGNB1] < 5 OR DUR2[SEGNB2] < 5
                THEN BEGIN
         SIMILAREVAL ::= SCORE(A11[SEGNB1],A12[SEGNB2],0.425,3,2.0);
                   IF SIMILAREVAL < 0 THEN GO TO FINSIMILAREVAL
                                      ELSE GO TO CONSTCONST
                     END
                ELSE GO TO CONSTCONST ;


CONSTCONST:

        RATIOLIM1 ::= 0.3 ; RATIOLIM2 ::= 0.5 ;
        WEIGHT1 ::= WEIGHT2 ::= 0.0 ;
        GO TO SONORANTSONORANT ;


VOWELVOWEL:VOWELCONST:VOWELNASAL:CONSTVOWEL:NASALVOWEL:

        RATIOLIM1 ::= RATIOLIM2 ::= 0.0 ;
        WEIGHT1 ::= WEIGHT2 ::= 0.1 ;

COMMENT
        IF THE TWO MATCHED SEGMENTS HAVE SIMILAR AMPLITUDES, THEN THE
        AVERAGE OF A1 AND A2 OVER THE TWO SEGMENTS IS USED TO COMPUTE
        THE WEIGHTS WHICH APPEAR IN THE SCORE CALLING SEQUENCE ,
            IF THEY ARE VERY DISSIMILAR, FACTOR WILL BE COMPUTED ONLY
        WITH RESPECT TO THE SEGMENT OF LARGEST AMPLITUDE ,
            THE CONCEPTUAL IDEA BEHIND THIS WEIGHT DEPENDENCY ON THE
        AMPLITUDE OF THE SEGMENTS TO BE MATCHED IS THAT :
                THE SMALLER THE POWER OF THE SOUND (REPRESENTED BY
        THE SEGMENT AMPLITUDE), THE MORE LIKELY ITS PARAMETERS WILL
        VARY FROM ONE UTTERANCE TO ANOTHER EVEN THOUGH THEY ARE SPOKEN
        BY THE SAME SPEAKER ,
            FOR EXAMPLE, WE KNOW THAT STRESSED VOWELS HAVE CONSISTENT
        PARAMETERS, BUT PARAMETERS CORRESPONDING TO UNSTRESSED SOUNDS
        HAVE A LARGE RANGE OF VARIATION ;
```

Algorithm 4 (continued). Similarity Evaluation Procedure

```
SONORANTSONORANT:

   IF IABS(A11[SEGNB1]+A21[SEGNB1]-A12[SEGNB2]-A22[SEGNB2])/2 ≤ 12
       THEN  FACTOR ::=
             (A11[SEGNB1]+A21[SEGNB1]+A12[SEGNB2]+A22[SEGNB2])/4
       ELSE  FACTOR ::=
             (MAX(A11[SEGNB1]+A21[SEGNB1],A12[SEGNB2]+A22[SEGNB2])-10)/2 ;

   RATIOLIM1 ::= RATIOLIM1+(280.0-2.0*FACTOR)/100.0 ;
   RATIOLIM2 ::= RATIOLIM2+(300.0-FACTOR)/80.0 ;
   WEIGHT1 ::= WEIGHT1+(FACTOR+40.0)/140.0 ;
   WEIGHT2 ::= WEIGHT2+(2*FACTOR+40.0)/240.0 ;


   SIMILAREVAL ::=
       SCORE(DUR1[SEGNB1],DUR2[SEGNB2],0.625,2,1,5)
          +3*SCORE(Z11[SEGNB1],Z12[SEGNB2],WEIGHT1,1,RATIOLIM1)
          +SCORE(A11[SEGNB1],A12[SEGNB2],WEIGHT2,2,RATIOLIM2)/2
          +SCORE(32*A21[SEGNB1]/A11[SEGNB1],32*A22[SEGNB2]/A12[SEGNB2],
                                  WEIGHT2,2,RATIOLIM2)
          +SCORE(32*A31[SEGNB1]/A11[SEGNB1],32*A32[SEGNB2]/A12[SEGNB2],
                                  WEIGHT2,4,RATIOLIM2)/2 ;

   DIVISOR ::= 6 ;

COMMENT
     IF A2 PARAMETER IS SMALL, THEN Z2 IS ERRATIC ;

   IF A21[SEGNB1] ≥ 8 OR A22[SEGNB2] ≥ 8 THEN
      BEGIN  SIMILAREVAL ::= SIMILAREVAL+
                +2*SCORE(Z21[SEGNB1],Z22[SEGNB2],WEIGHT1,2,RATIOLIM1) ;
             DIVISOR ::= DIVISOR+2
      END;


COMMENT
     IF A3 PARAMETER IS SMALL THEN Z3 IS ERRATIC ;

   IF A31[SEGNB1] ≥ 8 OR A32[SEGNB2] ≥ 8 THEN
      BEGIN    SIMILAREVAL ::= SIMILAREVAL
          +SCORE(Z31[SEGNB1],Z32[SEGNB2],WEIGHT1,4,RATIOLIM1) ;
          DIVISOR ::= DIVISOR+1
      END ;

   SIMILAREVAL ::= SIMILAREVAL/DIVISOR ;
   GO TO FINSIMILAREVAL ;
```

Algorithm 4 (continued). Similarity Evaluation Procedure

```
CONSTNASAL:

    IF DUR1[SEGNB1] < 5 OR Z11[SEGNB1] ≥ 5
        OR A11[SEGNB1] ≤ 10*A21[SEGNB1]
        OR A11[SEGNB1] ≤ 12*A31[SEGNB1] THEN GO TO CONSTCONST
                        ELSE
        BEGIN SIMILAREVAL ::= SCORE(A11[SEGNB1],A12[SEGNB2],0.35,4,2.0);
              IF SIMILAREVAL < 0 THEN GO TO FINSIMILAREVAL
                                 ELSE GO TO CONSTCONST

        END ;


 NASALCONST:

         IF DUR2[SEGNB2] < 5 OR Z12[SEGNB2] ≥ 5
                OR A12[SEGNB2] ≤ 10*A22[SEGNB2]
                OR A12[SEGNB2] ≤ 12*A32[SEGNB2] THEN GO TO CONSTCONST
                                                ELSE

                BEGIN SIMILAREVAL ::=
                        SCORE(A11[SEGNB1],A12[SEGNB2],0.35,4,2.0) ;
                      IF SIMILAREVAL < 0 THEN GO TO FINSIMILAREVAL
                                         ELSE GO TO CONSTCONST

                END

 FINSIMILAREVAL:END  SIMILAREVAL ;
```

Algorithm 4 (continued). Similarity Evaluation Procedure consonant might have been classified VOWEL, or a weak vowel might have been classified CONST or NASAL) and borderline cases of fricatives (an unvoiced fricative might have been classified BURST or a voiced fricative might have been classified FRICS). A feasibility value is assigned to each borderline case and the borderline case list is arranged in decreasing order of feasibility.

The following heuristics were found useful in defining the borderline cases:

-If a NASAL or CONST segment is a mild local maximum (a strong local maximum would have been classified vowel), it is candidate for becoming a VOWEL segment and its feasibility value is

$$A1 + A2 + A3 + DURATION/20 - 90 \qquad \begin{array}{l} 0 \leq A1 \leq 63 \\ 0 \leq A2 \leq 63 \text{ DURATION in ms} \\ 0 \leq A3 \leq 63 \end{array}$$

The heuristic concept behind this statement can be expressed as follows: The larger the segment amplitude and duration are, the more likely that it represents a vowel.

-If a non-stressed vowel is short, or has a low amplitude, it is candidate for becoming a CONST segment. Its feasibility value is:

$$90 - A1 - A2 - A3 - DURATION/20$$

(The smaller the amplitudes and duration are, the more likely that the segment is not a vowel.)

-If a FRICS segment is short or does not have excellent unvoiced fricative characteristics, it is candidate for becoming a BURST.

Its feasibility value is given by:

$(90 - \mathrm{DURATION}/20 - Z3)/3 + (A3-A1)/2 \qquad 0 \leq Z3 \leq 100$

(the shorter and less noisy the segment is, the more likely that it is not a FRICS.)

-If a BURST segment has a duration greater than 40ms it is candidate for becoming FRICS. Its feasibility value is:

$(\mathrm{DURATION}/20 + Z3 - 90)/3 + (A1-A3)/2$

(The longer and noisier the segment is, the more likely that it is FRICS.)

After this preliminary detection of borderline cases, the feasibility computation and the reordering, the first elements in the borderline case list are the most likely to be incorrectly classified.

Then each probable "classification error" is combinatorially modified and several new candidate lists are built by calling the procedure BUILDLIST(Flowchart - Figure IV-4). The meaning of the term "combinatorially" which is used in the preceding sentence, is best explained by an example:

Assuming that three borderline segments ① ② ③ have been detected in the incoming message representation, the following actions are performed by the procedure:

-Modify ① , store the acceptable-match candidates by calling BUILDLIST.

-Modify ② , call BUILDLIST.

-Modify ③ , call BUILDLIST.

-Modify ① and ② , call BUILDLIST.

-Modify ② and ③ , call BUILDLIST.

-Modify ③ and ① , call BUILDLIST.

-Modify ① , ② and ③ , call BUILDLIST.

On the basis of this new set of stored acceptable candidates, the selecting process chooses the best-match candidate which is given as the response if its overall similarity score is higher than the acceptability threshold ($>.5\%$).

To avoid too much computation time, the number of "corrected" classification borderline cases is limited to three (the three most likely to be in error, of course) and the procedure is allowed to run for no more than 15 seconds (30 seconds for the 561 word list). When this time has elapsed, the selecting procedure is called to choose between the candidates stored so far. A crude estimation of the effectiveness of this heuristic is given in section IV-7.

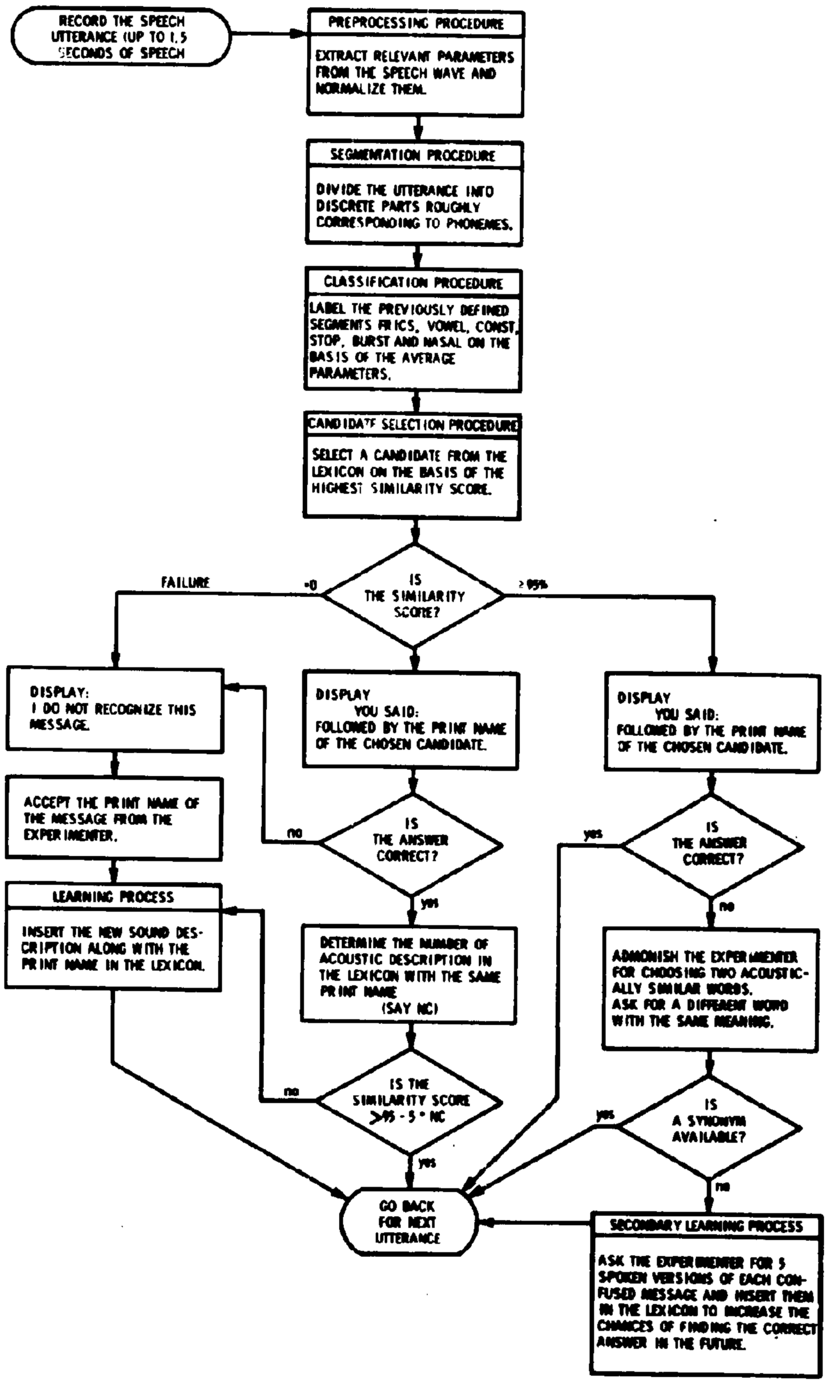
IV-5. ISOLATED WORD RECOGNITION - EARS (Effective Analyzer and Recognizer of Speech)

In order to evaluate the effectiveness of the storing, retrieving and matching processes, they were utilized as central elements of an isolated word recognizer: the EARS system. A flowchart of the system is shown on Figure IV-7.

The speech utterance coming from a microphone or a computer-controlled tape recorder (AMPEX PR10) is digitized by the hardware preprocessor. It is then analyzed by the segmentation procedure which builds the utterance representation (i.e., feature matrix described at the end of Chapter III). This feature matrix finally enters the candidate selection process which attempts to identify it and gives the written-form of the utterance if an acceptable-match has been found. The similarity scores between the incoming representation and the selected candidate (if any) are used to selectively learn new utterance representations. Namely, if the score is below a threshold, which depends on the number of candidates with the same print-name already learned, then the new representation is entered in the lexicon and will be used in the following tries. The system is interactive with an experimenter at a teletype or display console: The program types the answer when it recognizes a message and request the message print-name when it does not recognize it.

To train a system, one simply speaks in the microphone and types in the associated print-names when the system requests them.

Added to this basic pattern recognition system are several input/output facilities and debugging aids.



RECORD THE SPEECH UTTERANCE (UP TO 1.5 SECONDS OF SPEECH
PREPROCESSING PROCEDURE
EXTRACT RELEVANT PARAMETERS FROM THE SPEECH WAVE AND NORMALIZE THEM.
SEGMENTATION PROCEDURE
DIVIDE THE UTTERANCE INTO DISCRETE PARTS ROUGHLY CORRESPONDING TO PHONEMES.
CLASSIFICATION PROCEDURE
LABEL THE PREVIOUSLY DEFINED SEGMENTS FRICS, VOWEL, CONST, STOP, BURST AND NASAL ON THE BASIS OF THE AVERAGE PARAMETERS.
CANDIDATE SELECTION PROCEDURE
SELECT A CANDIDATE FROM THE LEXICON ON THE BASIS OF THE HIGHEST SIMILARITY SCORE.
FAILURE
=0
IS THE SIMILARITY SCORE?
≥95%
DISPLAY: I DO NOT RECOGNIZE THIS MESSAGE.
DISPLAY YOU SAID: FOLLOWED BY THE PRINT NAME OF THE CHOSEN CANDIDATE.
DISPLAY YOU SAID: FOLLOWED BY THE PRINT NAME OF THE CHOSEN CANDIDATE.
ACCEPT THE PRINT NAME OF THE MESSAGE FROM THE EXPERIMENTER.
no
IS THE ANSWER CORRECT?
yes
IS THE ANSWER CORRECT?
LEARNING PROCESS
INSERT THE NEW SOUND DESCRIPTION ALONG WITH THE PRINT NAME IN THE LEXICON.
yes
no
DETERMINE THE NUMBER OF ACOUSTIC DESCRIPTION IN THE LEXICON WITH THE SAME PRINT NAME (SAY NC)
ADMONISH THE EXPERIMENTER FOR CHOOSING TWO ACOUSTICALLY SIMILAR WORDS. ASK FOR A DIFFERENT WORD WITH THE SAME MEANING.
no
IS THE SIMILARITY SCORE >95 - 5 * NC
yes
IS A SYNONYM AVAILABLE?
yes
no
GO BACK FOR NEXT UTTERANCE
SECONDARY LEARNING PROCESS
ASK THE EXPERIMENTER FOR 5 SPOKEN VERSIONS OF EACH CONFUSED MESSAGE AND INSERT THEM IN THE LEXICON TO INCREASE THE CHANCES OF FINDING THE CORRECT ANSWER IN THE FUTURE.

Figure IV-7. Overall Flowchart of the EARS System .

The input facilities allow the experimenter to read a lexicon constituted during a preceding session or to read predigitized utterances from several input devices of the PDP-6 - PDP-10 dual processor system (disk, magnetic tapes, dectapes). This feature has been heavily used for the statistical tests performed on the program.

The corresponding output facilities allow the experimenter to store a lexicon and digitized utterances on external permanent memory.

The debugging aid is provided by intermediate printouts which can be selectively initiated and by a set of display routines capable of displaying the "thinking process". The display package and man-machine interaction were found to be the most powerful tools for debugging such a large system in which the overall changes caused by a procedure modification could not be predicted.

The statistical tests exhibited in Section IV-6 and the CRT photographs shown were obtained while using EARS.

The collection of subroutines which compose the system EARS occupies 35,000 words of memory. This number includes the display package, but not the lexicon, whose size depends on the size of the vocabulary. For the word list we processed, the lexicon space provided was, on the average, 90 memory words for each utterance to be recognized (for a single-speaker list), i.e., 5,000 for the 54 word lists recorded five times by one speaker, 10,000 words for the 54 word lists recorded by 10 different speakers, 50,000 for the list of 561 words.

The program in operation is illustrated by a portion of the typed dialog between the experimenter and the system as appearing at a teletype console, (Figure IV-8). To render this conversation more understandable,

```
          **
          SAY A MESSAGE PLEASE

          I DID NOT RECOGNIZE THIS MESSAGE.
          WOULD YOU TYPE IT PLEASE

JOHN HAS A BOOK
          **
          SAY A MESSAGE PLEASE

          YOU SAID : JOHN HAS A BOOK

YES
          **
          SAY A MESSAGE PLEASE

          I DID NOT RECOGNIZE THIS MESSAGE.
          WOULD YOU TYPE IT PLEASE

A QUEEN OF THE JUNGLE
          **
          SAY A MESSAGE PLEASE

          I DID NOT RECOGNIZE THIS MESSAGE.
          WOULD YOU TYPE IT PLEASE

PLEASE COME HOME
          **
          SAY A MESSAGE PLEASE

          I DID NOT RECOGNIZE THIS MESSAGE.
          WOULD YOU TYPE IT PLEASE

HOW ARE YOU TODAY
          **
          SAY A MESSAGE PLEASE

          YOU SAID : A QUEEN OF THE JUNGLE

YES
          **
          SAY A MESSAGE PLEASE

          YOU SAID : PLEASE COME HOME

YES
          **
          SAY A MESSAGE PLEASE

          YOU SAID : JOHN HAS A BOOK

YES
          **
```

Figure IV-8. Typical dialog between the system and an experimenter .

the responses from the system are indented and the experimenter's are not. Moreover, the internal decisions of the various procedures are illustrated by a set of photographs (Figures IV-9 through IV-23). Since the initial data were read from a previously created magnetic tape file, the utterance name appears at the top of the display, under the name of the procedure currently in use. When the system is processing data coming directly from the microphone, the inscription UNKNOWN, MICROPHONE INPUT replaces this name (e.g., Figure IV-6). These photographs were taken during one of the statistical tests, and the system had already examined 129 word representations (middle of the third word list from Gold recorded by K. Stevens), thus storing more than a hundred of them.

For this case, the effectiveness of the candidate space reduction heuristics is exhibited by the small number of comparisons ultimately performed (only 3 candidates went through the whole process).

Figure IV-9. The six parameters extracted from the speech wave by the hardware preprocessor are being processed by the segmentation procedure .

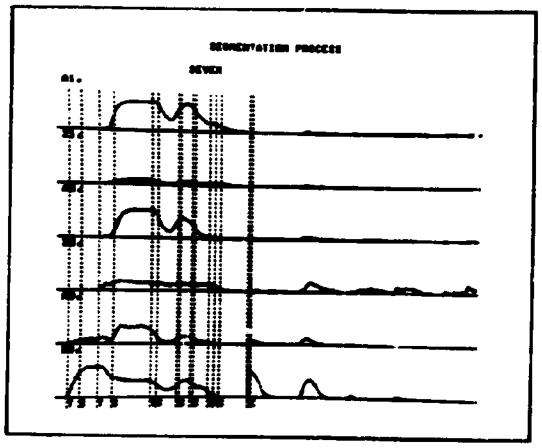


Figure IV-10. The utterance SEVEN has been segmented and the segments have been classified in phoneme groups . The resulting description of SEVEN is :

FRICS, followed by VOWEL, followed by NASAL, followed by VOWEL, followed by two NASAL's and one STOP .

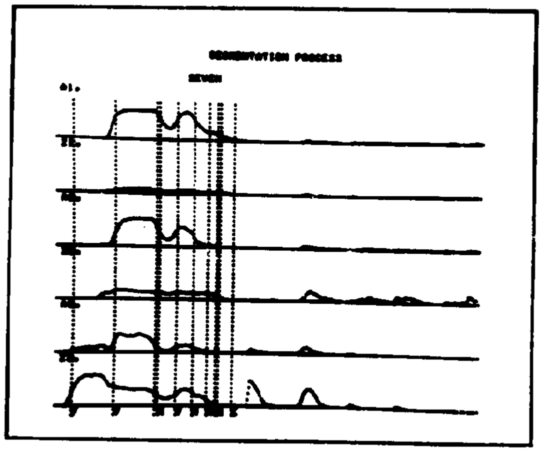
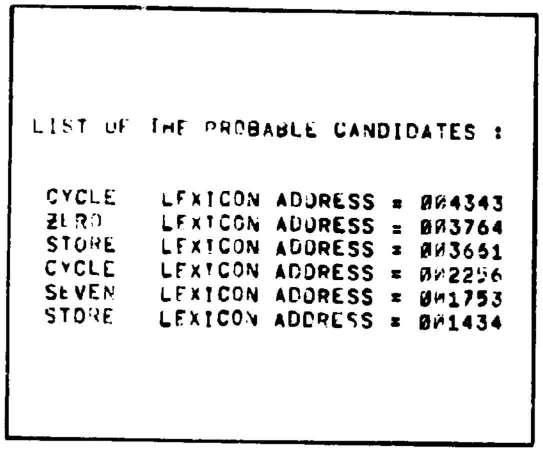


Figure IV-11. The lexicon search procedures have built the list of acceptable candidates on the basis of rough features of the utterance. Six candidates have been selected . All of them exhibit one FRICS followed by two VOWEL's .



```
LIST OF THE PROBABLE CANDIDATES :

 CYCLE    LEXICON ADDRESS = ØØ4343
 ZERO     LEXICON ADDRESS = ØØ3764
 STORE    LEXICON ADDRESS = ØØ3651
 CYCLE    LEXICON ADDRESS = ØØ2256
 SEVEN    LEXICON ADDRESS = ØØ1753
 STORE    LEXICON ADDRESS = ØØ1434
```

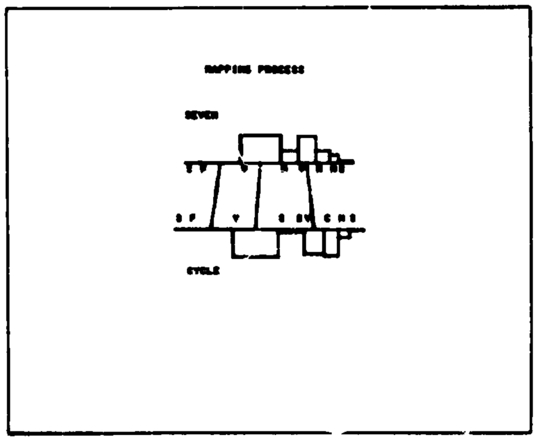


Figure IV-12. The first candidate is investigated by the similarity evaluation procedure . The procedure establishes links between corresponding segments of the two representations; VOWEL's and FRICS's are mapped .

(the utterance segments are represented by the amplitude in the first frequency band : A1).

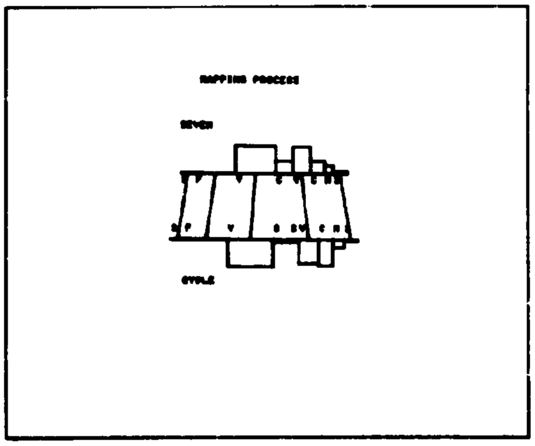


Figure IV-13. On the basis of similarity of parameters, more linkages are built between any two pairs of previously mapped segments . The procedure refused to map the segments between the two VOWEL's since their parameters were found too dissimilar .

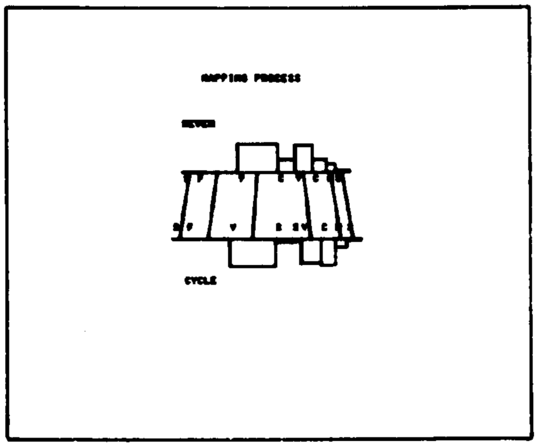


Figure IV-14. The initial mapping process terminates here. Each segment left unmapped will now be combined, if possible, with one of its adjacent segments .

Figure IV-15. The combining process has combined two of the remaining segments with one of the segments adjacent to them . The corresponding links were destroyed .

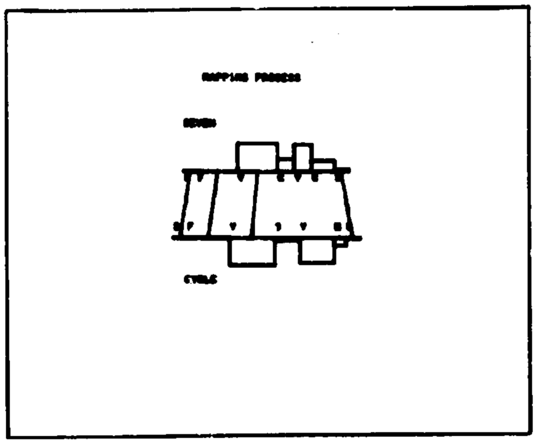


Figure IV-16. The links are recreated on the basis of similarity of parameters . The segment synchronization procedure terminates . The two central segments which correspond to the /v/ of SEVEN and the /k/ of CYCLE are not linked since they were found too dissimilar .

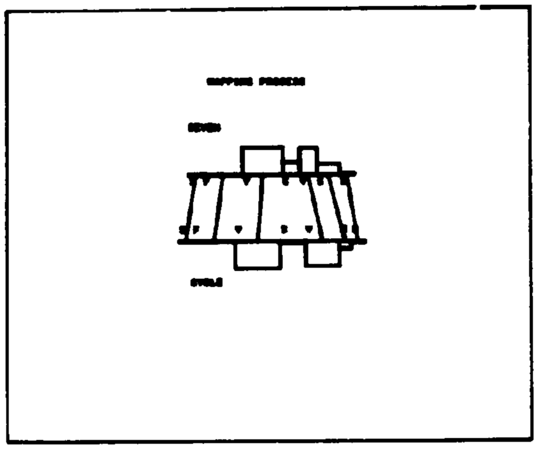


Figure IV-17. Three similarity scores have been computed and stored :

- vowel similarity score : 88%
- non-vowel similarity score : 49%
- overall similarity score : 63%

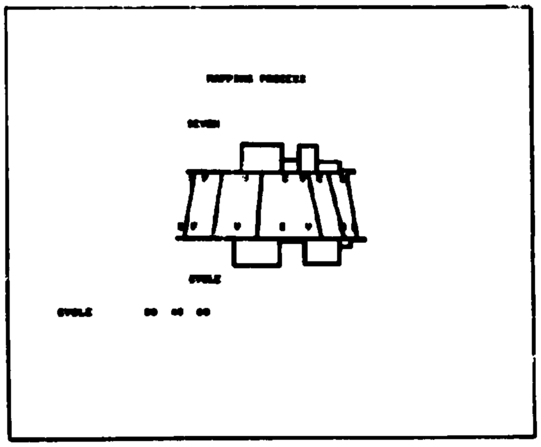


Figure IV-18. The next candidate is immediately eliminated since the vowel parameters are too dissimilar.

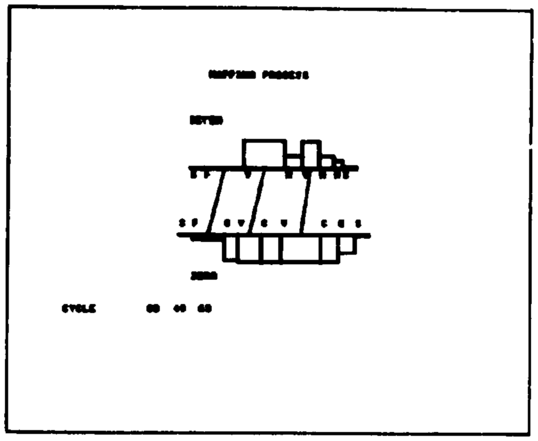


Figure IV-19 . The next candidate is also eliminated for the same reason .

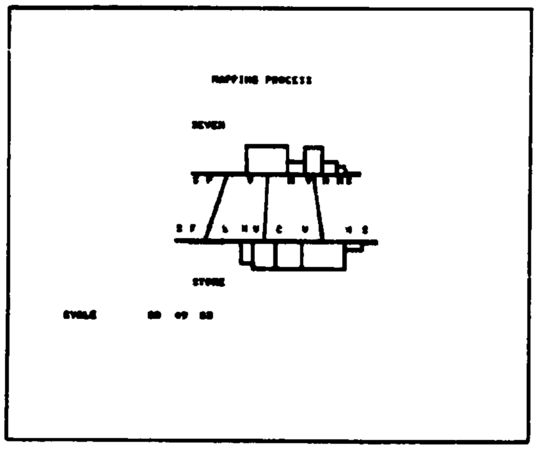


Figure IV-20. A different utterance of CYCLE has been accepted with the following scores :

- vowel similarity score : 90%

- non-vowel similarity score : 59%

- overall similarity score : 70%

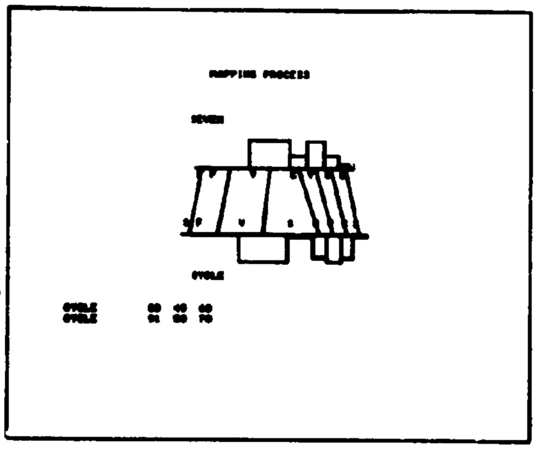


Figure IV-21. The candidate SEVEN present at the fifth place in the candidate list has been accepted with the scores :

- vowel similarity score : 92%
- non-vowel similarity score : 94%
- overall similarity score : 93%

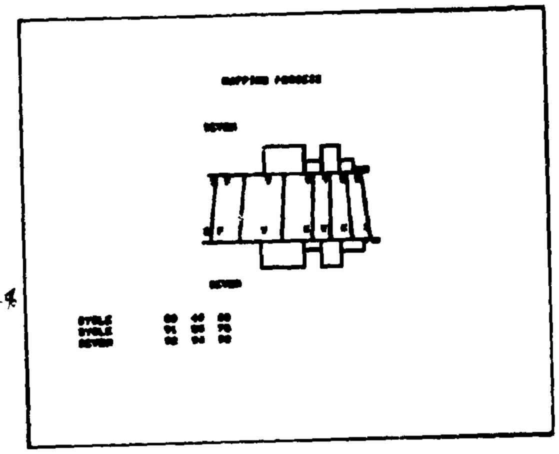


Figure IV-22. Another representation of STORE is eliminated .

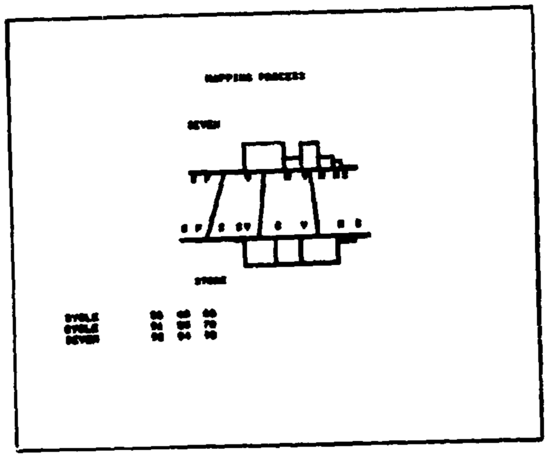
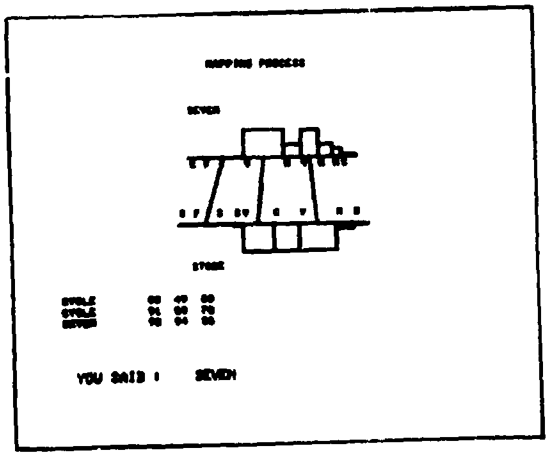


Figure IV-23. On the basis of the stored similarity scores, the candidate SEVEN is chosen . Since its overall similarity score is higher than 80%, its print name is returned to the main program which displays the program's decision .



IV-6. RESULTS

The EARS system was utilized in several direct experiments in which it had to recognize messages said by various speakers. In order to obtain statistical results, we tested the program with "canned" data recorded on audio tape. Several message lists were processed, namely:

-1. A word list from Gold (1966) recorded by Dr. K. Stevens and C. Williams. These lists recorded at Cambridge (Massachusetts) were graciously provided by Dr. D. Bobrow. The messages were recorded on high quality magnetic tape in a very quiet room (S/N ratio > 35 db). Figure IV-24 summarizes the results obtained, i.e., statistical results on accuracy and time-taken, and the confusions which arose. In this and the following tables an EH? entry means that the program rejected the utterance and an OK characterizes a correct recognition. The confusions are signaled by the name of the confused-with utterance.

-2. Word list recorded by J. Singer. To exhibit the large learning ability of the program, we made it process a large list of words and short sentences. The original list was 600 words long, but for technical difficulties (i.e., read errors on the computer tape holding the digitized data, samples exceeding the buffer size of the system, etc...), we had to remove 39 entries from the list. Moreover, since the computer time involved in such experiments is quite large (about 10 hours), only 4 sets of words were recorded and the first training round was only learned without identifying the utterances. The results are summarized in Figure IV-25. The word set was extracted from "A spoken word count" by Jones and Wepman and the sentence set from the play "Box and Cox" by J. Morton. The noise of the room was quite high

WORD LIST RECORDED BY KEN STEVENS					LIST OF MESSAGES	WORD LIST RECORDED BY CARL WILLIAMS				
PASS 0	PASS 1	PASS 2	PASS 3	PASS 4		PASS 0	PASS 1	PASS 2	PASS 3	PASS 4
EM?	OK	OK	OK	OK	INSERT	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	DELETE	EM?	EIGHT	OK	OK	OK
EM?	OK	OK	OK	OK	REPLACE	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	MOVE	EM?	TWO	LOAD	OK	OK
EM?	OK	OK	OK	OK	READ	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	BINARY	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	SAVE	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	CORE	EM?	FOUR	OK	FOUR	OK
EM?	OK	OK	OK	OK	DIRECTIVE	EM?	OK	EM?	OK	OK
EM?	OK	OK	OK	OK	LIST	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	LOAD	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	STORE	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	ADD	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	SUBTRACT	EM?	OK	OK	OK	OK
STORE	OK	OK	OK	OK	ZERO	EM?	EM?	OK	OK	OK
LOAD	OK	OK	OK	OK	ONE	EM?	OK	OK	OK	OK
SAVE	OK	OK	OK	OK	TWO	ZERO	OK	OK	OK	OK
EM?	OK	OK	OK	OK	THREE	EM?	OK	OK	READ	OK
LOAD	OK	OK	OK	OK	FOUR	CORE	OK	OK	OK	CORE
EM?	OK	OK	OK	OK	FIVE	ONE	NINE	OK	OK	OK
EM?	OK	OK	OK	OK	SIX	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	SEVEN	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	EIGHT	EM?	OK	OK	OK	OK
FIVE	OK	OK	OK	OK	NINE	ONE	OK	OK	FIVE	OK
EM?	OK	OK	OK	OK	MULTIPLY	EM?	OK	OK	OK	OK
NINE	OK	OK	OK	OK	DIVIDE	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	NUMBER	EM?	OK	OK	OK	OK
EM?	OK	OK	NINE	OK	NAME	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	END	EM?	OK	OK	OK	OK
TWO	OK	OK	OK	OK	SCALE	EM?	OK	OK	OK	OK
SEVEN	OK	OK	OK	OK	CYCLE	SEVEN	OK	OK	OK	OK
EM?	OK	OK	OK	OK	SKIP	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	JUMP	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	ADDRESS	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	OVERFLOW	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	POINT	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	CONTROL	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	REGISTER	EM?	OK	OK	OK	OK
LOAD	OK	OK	OK	OK	WORD	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	EXCHANGE	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	INPUT	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	OUTPUT	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	MAKE	EIGHT	OK	OK	OK	OK
INSER	OK	OK	OK	OK	INTERSECT	INSER	OK	OK	OK	OK
DIVID	OK	OK	OK	OK	COMPARE	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	ACCUMULATE	EM?	OK	OK	OK	OK
NUMBE	NAME	BINAR	OK	OK	MEMORY	EM?	OK	OK	OK	OK
NAME	OK	OK	OK	OK	BYTE	EM?	OK	OK	OK	OK
CORE	OK	OK	OK	OK	QUARTER	EM?	OK	OK	OK	OK
EM?	OK	OK	OK	OK	HALF	BYTE	OK	OK	OK	OK
FOUR	OK	OK	OK	OK	WHOLE	EM?	OK	FOUR	OK	OK
NAME	OK	OK	OK	OK	UNITE	DIVID	OK	MEMOR	OK	OK
EM?	OK	OK	OK	OK	DECIMAL	EM?	OK	OK	OK	OK
OUTPU	OK	OK	OK	OK	OCTAL	OUTPU	OK	OK	OK	OK
54	54	54	54	54	NUMBER OF MESSAGES	54	54	54	54	54
0 0.0 %	53 98.1 %	53 98.1 %	53 98.1 %	54 100.0 %	RECOGNIZED CORRECTLY	0 0.0 %	49 90.7 %	50 92.6 %	51 94.4 %	53 98.1 %
37 68.5 %	0 0.0 %	0 0.0 %	0 0.0 %	0 0.0 %	UNKNOWN OR REJECTED	44 81.5 %	1 1.9 %	1 1.9 %	0 0.0 %	0 0.0 %
17 31.5 %	1 1.9 %	1 1.9 %	1 1.9 %	0 0.0 %	ERRORS	10 18.5 %	4 7.4 %	3 5.6 %	3 5.6 %	1 1.9 %
1.9 S	2.0 S	2.8 S	2.0 S	2.9 S	AVER. COMP. TIME /MESSAGE	1.8 S	2.2 S	3.4 S	2.7 S	3.1 S

Figure IV-24. Results obtained for the word lists from B.B.N. .

LIST OF MESSAGES	WORD LIST RECORDED BY JEFF SINGER			
	PASS 0	PASS 1	PASS 2	PASS 3
AND	EH?	MEAN	ENGIN	OK
WELL	EH?	OK	OK	OK
BUT	EH?	OK	MUCH	OK
GO	EH?	BODY	BODY	OK
ONE	EH?	BEGIN	OK	OK
FOR	EH?	OK	OK	OK
ALL	EH?	OK	OK	OK
HERE	EH?	OK	OK	OK
FROM	EH?	OK	STONE	OK
ME	EH?	OK	OK	OK
TRY	EH?	COLLE	COLLE	OK
COME	EH?	ROOM	THEN	OK
MEAN	EH?	ME	ME	OK
MUCH	EH?	OK	MEDDE	OK
MAKE	EH?	OK	OK	OK
WHO	EH?	FOR	OK	OK
THEN	EH?	OK	OK	OK
RIGHT	EH?	NOT	OK	OK
OLD	EH?	OK	OK	OK
BOY	EH?	ONLY	OK	OK
MORE	EH?	OK	OK	OK
BACK	EH?	OK	OK	OK
SOMEONE	EH?	OK	OK	OK
EXPRESSION	EH?	EH?	OK	OK
FIGURE	EH?	OK	OK	OK
ANOTHER	EH?	MARRY	MOTHE	OK
APPEAR	EH?	OK	OK	OK
WHATEVER	EH?	OK	MATTE	OK
TROUBLE	EH?	OK	OK	OK
HIMSELF	EH?	OK	HERSE	OK
POSSIBLY	EH?	HOSPI	POSSI	OK
SERIOUS	EH?	SERIO	OK	SERIO
EVERYTHING	EH?	IS	INDIC	OK
ONLY	EH?	LONEL	LONEL	OK
PRETTY	EH?	OK	OK	OK
BELIEVE	EH?	OK	OK	OK
DOCTOR	EH?	OK	OK	OK
APPARENTLY	EH?	OK	OK	OK
TABLE	EH?	OK	OK	OK
ANYWAY	EH?	GRAND	SAVOR	OK
NEVER	EH?	MATTE	[illegible]	OK
BACKGROUND	EH?	OK	OK	OK
STUDY	EH?	OK	OK	OK
BEHIND	EH?	FINAL	OK	OK
PROBLEM	EH?	SMOKE	OK	OK
EUROPE	EH?	OK	OK	OK
BELOW	EH?	OK	ALONE	OK
FOREGROUND	EH?	EH?	OK	OK
PARTY	EH?	MARRI	OK	OK
STUDENT	EH?	SIGHT	SIGHT	OK
IS	EH?	PASS	OK	OK
SHE	EH?	OK	OK	OK
JUST	EH?	OK	DRESS	OK
SOME	EH?	SO	OK	SUN
SO	EH?	OK	OK	OK
SAY	EH?	OK	SAD	OK
USED	EH?	[illegible]	OK	OK
FACE	EH?	OK	OK	OK
MUST	EH?	GUESS	OK	OK
SORT	EH?	OK	OK	OK
GUESS	EH?	OK	OK	PASS
ELSE	EH?	OK	OK	OK
SCENE	EH?	SAME	SAME	OK
SCHOOL	EH?	OK	OK	OK
SIDE	EH?	OK	OK	OK
US	EH?	BEST	OK	OK
SHOW	EH?	OK	OK	OK
[illegible]	EH?	OK	OK	OK
SAD	EH?	SAY	SAY	OK
FAMILY	EH?	OK	OK	OK
OUTSIDE	EH?	UPSET	OK	OK
AGAIN	EH?	OK	OK	OK
ALWAYS	EH?	OK	OK	OK
BETTER	EH?	OK	MARRI	OK
ANIMAL	EH?	PAINT	OK	OK
FEELING	EH?	SCENE	FAMIL	OK
SURPRISED	EH?	OK	OK	OK
ALTHOUGH	EH?	EH?	OK	BELOW
TOGETHER	EH?	AGAIN	OK	OK
HERSELF	EH?	OK	OK	OK
CONCERT	EH?	OK	SAY	OK
GEORGE	EH?	OK	OK	OK
COUPLE	EH?	OK	[illegible]	OK
BRIDGE	EH?	MAKE	MIDE	OK
DIFFERENT	EH?	DIFFE	DIFFE	OK
EMIGRATE	EH?	UNDER	OK	OK
READY	EH?	HAVE	MARRY	OK
[illegible]	EH?	OK	OK	OK
OF COURSE	EH?	OK	OK	OK
ALMOST	EH?	OK	OK	OK
AGAINST	EH?	ADVIC	OK	OK
INTERESTED	EH?	INTER	INTER	OK
DETERMINED	EH?	[illegible]	[illegible]	OK
DISTANCE	EH?	OK	OK	OK
ABOVE	EH?	ACROS	BODY	OK
COMFORT	EH?	OK	OK	OK
HYPNOTIZE	EH?	OK	OK	OK
SISTER	EH?	SENSE	OK	OK
INSIDE	EH?	OK	OK	OK
CHANCE	EH?	OK	OK	OK
CROSS	EH?	OK	MOST	OK
ENJOY	EH?	OK	OK	OK
EXCEPT	EH?	EXACT	EXPEC	OK
DISTURBED	EH?	RESUL	OK	OK
UNDERSTAND	EH?	OK	OK	OK
BODY	EH?	OK	OK	OK
DIFFICULT	EH?	OK	OK	OK
MODERN	EH?	OK	OK	OK
EASY	EH?	OK	OK	OK
ACTUALLY	EH?	OK	OK	NATUR
EVERY	EH?	OK	OK	OK
INSTRUMENT	EH?	OK	OK	OK
MUSIC	EH?	PEASC	OK	OK
UNCLE	EH?	OK	OK	MOST
RETURN	EH?	DETER	OK	OK
GRANDMOTHER	EH?	ANYWA	OK	OK
HESITATE	EH?	OK	OK	OK
PERSUADE	EH?	OK	OK	OK
FORWARD	EH?	OK	OK	OK
HOWEVER	EH?	OK	WHATE	OK
HARDLY	EH?	OK	OK	OK
AFTERNOON	EH?	OK	OK	OK
ASSUME	EH?	OK	OK	OK
EVERYBODY	EH?	OK	I MAY	OK
INSTEAD	EH?	OK	OK	OK
MATTER	EH?	OK	OK	BETTE
ANXIOUS	EH?	OK	OK	CHRIS
CONNECTION	EH?	TRAGE	OK	OK
CONVERSATION	EH?	OK	OK	OK
COVERED	EH?	OK	TROUB	OK
WONDERFUL	EH?	MEDIC	OK	OK
ATTITUDE	EH?	OK	ACTUA	OK
CONVINCE	EH?	OK	OK	OK
INDIVIDUAL	EH?	AMBIT	OK	OK
OBVIOUSLY	EH?	OK	OK	OK
PARTICULARLY	EH?	EH?	OK	OK
MODERNISTIC	EH?	RISE	OK	OK
AUTOMOBILE	EH?	OK	EH?	OK
MINISTER	EH?	OK	OK	OK
BESIDE	EH?	DECID	OK	OK
DURING	EH?	OK	OK	DAWN
PAINTING	EH?	OK	TALE	OK
SHAKE	EH?	OK	OK	OK
RELAX	EH?	OK	LAST	OK
REPRESENT	EH?	OK	OK	OK
SUGGEST	EH?	OK	OK	OK
SURPRISE	EH?	SURPR	SURPR	SURPR
ASLEEP	EH?	OK	OK	OK
RIVER	EH?	BETTE	WINTE	OK
AMERICAN	EH?	MATTE	OK	OK
MUSICIAN	EH?	AFFEC	OK	OK
SATISFIED	EH?	EH?	OK	OK
USUALLY	EH?	REASO	OK	OK
QUESTION	EH?	OK	TRAGE	OK
ATTENTION	EH?	OK	OK	OK
CONTEMPLATE	EH?	FINAL	OK	OK
PARENT	EH?	THEM	OK	OK
ACCOUNT	EH?	OK	OK	OK
OTHERS	EH?	OK	ARTIS	OK
PRESUME	EH?	ASSUM	OK	PERFO
SERVICE	EH?	SERIO	OK	OK
ELDERLY	EH?	OK	OK	OK
FINISH	EH?	SHE	OK	OK
HOSPITAL	EH?	PLEAS	OK	OK
REST	EH?	REST	REST	BEST
PERFORM	EH?	OK	PRESU	OK
STRIKE	EH?	OK	SIGHT	START
AMBITION	EH?	OK	OK	OK
ARRIVE	EH?	AT AL	OK	OK
ENDING	EH?	PAINT	ANYWA	OK
LIKELY	EH?	OK	NOTHI	OK
JUNGLE	EH?	SHADO	SHADO	OK
RECEIVE	EH?	ESCAP	OK	OK
RELAXED	EH?	RELAX	RELAX	OK
SPELL	EH?	SPOT	OK	OK
SUGGESTION	EH?	OK	OK	OK
MORNING	EH?	OK	OK	OK
ATTEMPT	EH?	OK	OK	OK
NATURE	EH?	OK	COMME	OK
PERSONALITY	EH?	PRESU	OK	OK
SICKNESS	EH?	SENSE	OK	OK
EXTREMELY	EH?	FAMIL	EXTEN	OK
CHRISTMAS	EH?	PERSO	BUSIN	OK
ACCIDENT	EH?	NOTHI	OK	OK
FURNITURE	EH?	FIRST	OK	OK
AUGUST	EH?	OK	ELDER	OK
PRESENT	EH?	OK	OK	OK
EXPERIENCE	EH?	OK	OK	OK
LONELY	EH?	OK	OK	OK
IMPRESSION	EH?	OK	OK	OK
REALIZE	EH?	ENORA	WINDO	OK
SLIDE	EH?	OK	SIDE	OK
EDUCATION	EH?	OK	OK	ATTEN
INTERESTING	EH?	OK	OK	INTER
BROKEN	EH?	OK	OK	OK
LATER	EH?	OK	LABOR	LABOR

Figure IV-25. Results obtained for a list of 561 English words .

```
PAPER           EN?  OK     DAYDR  OK        PLEASED         EN?  IS     OK     OK
TEACHER         EN?  OK     PITCH  OK        REPAIR          EN?  OK     OK     OK
ARGUMENT        EN?  OPERA  EVERY  OK        SHOULDER        EN?  OK     OK     OK
SERIOUSLY       EN?  SERIO  SERIO  SERIO     STONE           EN?  SHOW   OK     OK
DISGUSTED       EN?  OK     OK     DISTR     ANYBODY         EN?  OK     OK     OK
PLEASURE        EN?  OK     CONNE  OK        CONSOLE         EN?  OK     OK     OK
POLICE          EN?  OK     OK     OK        USE             EN?  IS     OK     OK
SPACE           EN?  OK     OK     OK        SURE            EN?  OK     OK     OK
UPSET           EN?  EXCEP  OK     OK        SICK            EN?  SIN    OK     OK
SOMETIMES       EN?  SOMEP  OK     OK        BUST            EN?  OK     DRESS  OK
ACCOMPLISH      EN?  EXACT  OK     OK        FAST            EN?  OK     GUESS  OK
BADLY           EN?  OF     OK     OK        SAME            EN?  SCENE  SCENE  OK
BUSINESS        EN?  OK     OK     OK        LAST            EN?  OK     OK     OK
DAYDREAM        EN?  PATIE  OK     OK        FIRST           EN?  OK     OK     OK
DISTRESS        EN?  DISTR  OK     OK        CLOSE           EN?  CAUSE  OK     OK
GENTLEMAN       EN?  JUNGL  TRAGE  OK        SLEEP           EN?  OK     OK     OK
HAPPINESS       EN?  OK     HAPPY  OK        SUCH            EN?  OK     OK     OK
TELEVISION      EN?  CHILD  OK     OK        STAY            EN?  OK     OK     OK
INTEREST        EN?  OK     OK     OK        MUST            EN?  OK     OK     OK
ALREADY         EN?  WITHO  OK     OK        START           EN?  SORT   SIGHT  OK
EMBRACE         EN?  OK     OK     OK        REST            EN?  BEST   OK     BEST
FRIGHTENED      EN?  OK     OK     OK        CASE            EN?  OK     OK     OK
CHARACTER       EN?  OK     OK     OK        SUN             EN?  SOME   SOME   OK
CONTENDED       EN?  OK     OK     OK        HORSE           EN?  OK     OK     OK
EXTEND          EN?  OK     OK     OK        SOON            EN?  OK     SOUND  OK
UNDOUBTEDLY     EN?  OK     OK     OK        CAUSE           EN?  OK     OK     OK
ANSWER          EN?  HANDS  OK     OK        SPEAK           EN?  OK     OK     OK
NATURALLY       EN?  ACCID  OK     LIKEL     SPOT            EN?  SPELL  OK     OK
SMOKE           EN?  SORT   OK     OK        BUILDING        EN?  OK     OK     OK
DISCUSS         EN?  DISTR  OK     DISTR     TOMBSTONE       EN?  OK     OK     OK
SORROWFUL       EN?  START  OK     SORT      LESSON          EN?  OK     OK     OK
STUDIOUS        EN?  OK     OK     OK        FINALLY         EN?  OK     READY  OK
OPERATE         EN?  OK     OK     OK        SOMEWHAT        EN?  SOMED  SOMEW  OK
BIBLE           EN?  OK     OK     OK        AT ALL          EN?  BELOW  OK     OK
MOMENT          EN?  OK     BODY   OK        DRESSED         EN?  REST   JUST   OK
SUBJECT         EN?  OK     OK     OK        ADVICE          EN?  OK     OK     OK
SURGEON         EN?  OK     OK     OK        DECIDE          EN?  OK     BESID  OK
SADNESS         EN?  OK     OK     OK        FLOWER          EN?  SHOW   FATHE  OK
CONCERN         EN?  DISTU  OK     OK        CHILDREN        EN?  OK     OK     OK
APPEARANCE      EN?  OK     OK     OK        REMIND          EN?  BEHIN  OK     OK
CONDITION       EN?  CONNE  OK     OK        EXPECT          EN?  EN?    OK     OK
DECISION        EN?  OK     OK     POSIT     BEAUTIFUL       EN?  OK     OK     OK
EVENING         EN?  OK     OK     DAYDR     PRACTICE        EN?  PURPO  OK     OK
SOMEWAY         EN?  OK     SOMEW  OK        EXACTLY         EN?  SUFFE  OK     OK
BOTTOM          EN?  OK     OK     OK        POSITION        EN?  OK     MUSIC  OK
COMPLETELY      EN?  IMPRE  OK     OK        FOREST          EN?  OK     HORSE  OK
UNHAPPY         EN?  OK     OK     OK        IMAGINATION     EN?  OK     OK     AMBIT
VIOLINIST       EN?  OK     ALWAY  OK        LISTEN          EN?  LESSO  OK     LESSO
ACTIVITY        EN?  OK     EN?    OK        SORRY           EN?  OK     SUN    OK
ANTICIPATE      EN?  OK     OK     OK        DEPRESSED       EN?  OK     APPEA  OK
DISAPPOINTED    EN?  OK     OK     OK        SEVERE          EN?  SLEEV  OK     OK
RESULT          EN?  OK     MERSE  OK        PATIENT         EN?  OK     OK     OK
SENSE           EN?  OK     OK     OK        PREHISTORIC     EN?  OK     OK     OK
SUFFER          EN?  OK     OK     EXACT     ACROSS          EN?  BECAU  OK     OK
SLEEVE          EN?  OK     OK     OK        THEMSELVES      EN?  OK     OK     OK
CIRCUMSTANCE    EN?  OK     OK     OK        TRAGEDY         EN?  OK     OK     OK
DISTRESSED      EN?  DISTR  DISTR  DISTR     WINTER          EN?  OK     OK     OK
FASHIONED       EN?  OK     OK     OK        MAYBE           EN?  OK     OK     OK
STARTLED        EN?  TROUB  OK     OK        MOTHER          EN?  ANOTH  OK     OK
SWEETHEART      EN?  OK     OK     OK        BECAUSE         EN?  OK     ACROS  OK
AFTERWARDS      EN?  OK     OK     OK        PERHAPS         EN?  OK     OK     OK
DETERMINATION   EN?  OK     OK     OK        EITHER          EN?  OK     OK     OK
DISCOVER        EN?  OK     OK     SORRY     PERSON          EN?  OK     OK     OK
RELAXATION      EN?  OUTSI  OK     OK        PEOPLE          EN?  OK     OK     OK
RELIGIOUS       EN?  OK     OK     OK        WINDOW          EN?  OK     WINTE  OK
SEVERAL         EN?  OK     OK     OK        REALLY          EN?  MARRY  OK     OK
ASHAMED         EN?  SHE    OK     OK        HAPPEN          EN?  OK     OK     OK
DESPAIR         EN?  STAY   OK     OK        STORY           EN?  OK     OK     OK
UNCONSCIOUS     EN?  OK     OK     OK        QUITE           EN?  OK     WIDE   OK
LAMPOST         EN?  OK     OK     OK        SUPPOSE         EN?  SURPR  OK     OK
STATION         EN?  STAY   SENSE  OK        FATHER          EN?  FLOWE  OK     OK
DARKNESS        EN?  BADLY  HAPPI  OK        NOTHING         EN?  LESSO  OK     OK
DIFFERENCE      EN?  OK     DIFFE  OK        AFTER           EN?  OK     AWFUL  OK
NECESSARILY     EN?  OK     EN?    OK        VIOLIN          EN?  OK     OK     OK
SUCCESFUL       EN?  OK     OK     OK        HAPPY           EN?  HAPPE  OK     OK
ESPECIALLY      EN?  OK     EXPRE  OK        BEFORE          EN?  FOR    FOR    OK
SUNLIGHT        EN?  SOMEW  OK     OK        MOUNTAIN        EN?  WRITE  OK     OK
INTERPRET       EN?  OK     OK     OK        PLEASANT        EN?  FASHI  OK     OK
AFFECTION       EN?  EFFEC  OK     OK        GRAVEYARD       EN?  BACKE  OK     OK
ENCOURAGE       EN?  OK     OK     OK        SITUATION       EN?  OK     OK     OK
ENTER           EN?  OK     OK     OK        LOOKING         EN?  OK     WORKI  OK
INJURED         EN?  OK     MEMBE  OK        OPERATION       EN?  EN?    OK     OK
MONSTER         EN?  ONCE   OK     OK        SOMEPLACE       EN?  OK     OK     SOMET
PECULIAR        EN?  OK     SHOE   OK        CORNER          EN?  OK     HANDL  OK
RISE            EN?  OK     OK     OK        MARRIED         EN?  OK     MARRY  OK
SHOE            EN?  OK     OK     OK        PARTICULAR      EN?  OK     PARTI  OK
SINISTER        EN?  OK     OK     OK        SOMEWHERE       EN?  OK     OK     OK
TALE            EN?  WERE   OK     OK        BETWEEN         EN?  AGAIN  OK     OK
THEATRE         EN?  OK     OK     OK        EVENTUALLY      EN?  OK     OK     OK
TWENTY          EN?  OK     FREE   OK        NUMBER          EN?  LATER  OK     ANOTH
WAKE            EN?  OK     OK     OK        ARTIST          EN?  REST   OK     OK
WHITE           EN?  OK     OK     BRIDG     WITHOUT         EN?  OK     OK     OK
AMOUNT          EN?  ONE    OK     OK        MORNING         EN?  OK     OK     OK
SOUND           EN?  SOON   OK     OK        UNLESS          EN?  OK     LAST   OK
BANNED          EN?  OK     OK     OK        ALONE           EN?  OK     BELOW  GO ON
BUSY            EN?  OK     OK     OK        SMALL           EN?  OK     OK     OK
CHOKE           EN?  OK     OK     OK        CERTAINLY       EN?  SURGE  OK     OK
CONTEMPLATION   EN?  OK     OK     OK        REASON          EN?  OK     OK     OK
DEFINITE        EN?  MOUNT  OK     GENTL     DRAGON          EN?  NEVER  OK     OK
DISCOURAGED     EN?  OK     OK     DISCO     MEDICAL         EN?  MATTE  MOUNT  OK
DRUNK           EN?  MEMBE  AMOUN  OK        TERRIBLE        EN?  PAREN  OK     OK
EVENT           EN?  AMOUN  OK     OK        COLLEGE         EN?  OK     OK     OK
HIDE            EN?  GO     OK     OK        POSSIBLE        EN?  OK     OK     OK
```

Figure IV-25.(continued). List of 561 English Words .

HUT	EH?	CHOKE	WRITE	OK
INVOLVE	EH?	OK	OK	OK
SAY	EH?	SHOW	OK	OK
SHADOW	EH?	FINAL	SHOW	SHOW
SIX	EH?	OK	OK	OK
STIFF	EH?	OK	OK	OK
THIRTY	EH?	SOUND	OK	OK
TOY	EH?	CORNE	FORMA	OK
UNFORTUNATE	EH?	OF CO	OK	OK
LAWN	EH?	AND	OK	OK
WITCH	EH?	OK	OK	WAKE
EXERCISE	EH?	OK	OK	OK
HANDLE	EH?	OK	OK	OK
ALONGSIDE	EH?	OK	OK	OK
APPARENT	EH?	OK	OK	TWENT
ARCH	EH?	OK	OK	OK
AWFULLY	EH?	MOTHE	ELDER	OK
BULLET	EH?	OK	OK	OK
BUY	EH?	OK	OK	OK
CANVAS	EH?	OK	OK	OK
CONSIDER	EH?	CONCE	OK	OK
EFFECT	EH?	PROFE	STRIK	OK
EFFORT	EH?	OK	RIVER	OK
FEMALE	EH?	FEELI	OK	OK
FOURTEEN	EH?	EH?	OK	OK
HANDSOME	EH?	OK	OK	OK
INTEND	EH?	OK	OK	OK
KINDLY	EH?	OK	COMPA	OK
LABOR	EH?	MEMBE	OK	OK
PEPPER	EH?	OK	LABOR	OK
MILITARY	EH?	OK	OK	OK
OFFICER	EH?	OK	OK	OK
OPENING	EH?	OK	OK	OK
PHYSICAL	EH?	EH?	OK	OK
PITCHER	EH?	CONFO	OK	OK
PROFESSION	EH?	OK	TRAGE	OK
SAVE	EH?	SAY	OK	OK
SCHOOLING	EH?	OK	OK	OK
TALL	EH?	OK	TWENT	OK
WAREHOUSE	EH?	OK	OK	OK
WATERFALL	EH?	MILIT	OK	OK
WEDDED	EH?	WITCH	OK	OK
YARD	EH?	OK	BARGE	OK
CREEP	EH?	OK	TOY	OK
ABSOLUTELY	EH?	OK	OK	OK
ACCORD	EH?	TOY	TOY	OK
AFFECTIONATE	EH?	ESPEC	PROFE	OK
ABSCOND	EH?	OK	OK	OK
BROOK	EH?	OK	OK	OK
COMPANY	EH?	OK	OK	OK
COMPARE	EH?	OK	OK	OK
CANON	EH?	BUM	BUM	OK
CLOSION	EH?	OK	OK	GOOD
EIGHT	EH?	EH?	OK	OK
ELEPHANT	EH?	OK	BULLE	OK
FAST	EH?	OK	DRESS	OK
FREE	EH?	TRUE	OK	OK
GREY	EH?	OK	BRIDG	OK
HUGE	EH?	OK	OK	OK
HYPNOSIS	EH?	OK	OK	OK
IGLOO	EH?	OK	OK	OK
MEANING	EH?	OK	OK	OK
NINE	EH?	OK	OK	OK
PURPOSE	EH?	OK	OK	OK
RAISE	EH?	OK	OK	OK
RESPECT	EH?	OK	OK	OK
RIFLE	EH?	WRITE	WEDGE	OK
ROD	EH?	OK	OK	OK
SIGHT	EH?	START	OK	OK
BACKGROUND	EH?	OK	OK	OK
TERRIBLY	EH?	OK	OK	TOY
THICK	EH?	OK	OK	SICK
GOOD MORNING	EH?	OK	NINE	OK
HOW ARE YOU	EH?	BUM	NINE	GO ON
JOHN HAS A BOOK	EH?	EH?	OK	OK
THANK YOU	EH?	KINDL	KINDL	OK
NOT I, INDEED	EH?	OK	OK	OK
HOW AM I	EH?	OK	OK	OK
FIRST OF WELL	EH?	OK	OK	OK
HE IS GONE	EH?	OK	OK	OK
WHAT NEXT	EH?	OK	OK	OK
HERE IT IS	EH?	OK	OK	OK
IN THE FIRST PLACE	EH?	EH?	OK	OK
COME IN	EH?	OK	OK	OK
IF YOU COME TO THAT	EH?	EH?	OK	OK
PICK ME UP	EH?	EH?	OK	OK
WHAT DO YOU WANT	EH?	OK	OK	OK
I CANNOT STAND THIS	EH?	IN TH	OK	OK
WELL BUT GENTLEMEN	EH?	OK	OK	OK
BOTH OF US	EH?	WAREH	OK	OK
I WILL TAKE IT	EH?	OK	OK	OK
SO WILL I	EH?	OK	OK	OK
I SPOKE FIRST	EH?	OK	OK	OK
VERY WELL, SIR	EH?	OK	NOW.	OK
WHAT ARE YOU ABOUT	EH?	EH?	OK	OK
PUT YOUR PIPE OUT	EH?	I CON	OK	OK
SUPPOSE WE CHANGE	EH?	EH?	OK	OK
I HAVE GOT IT	EH?	HALF	OK	DO AS
IT IS THE SAME TO ME	EH?	OK	OK	OK
HEADS, I WIN	EH?	THAT'	THAT'	OK
TAILS, YOU LOSE	EH?	OK	OK	OK
WILL YOU PERMIT ME	EH?	OK	OK	OK
YOU WILL FORGIVE ME	EH?	OK	OK	OK
THEN READ IT	EH?	OK	OK	OK
BREAK THE SEAL	EH?	EH?	DROWN	OK
GOODNESS GRACIOUS	EH?	OK	OK	OK
GRACIOUS GOODNESS	EH?	OK	OK	OK
HOW CAN THAT BE	EH?	ILL-F	OK	THAT
GO ON	EH?	OK	OK	OK
POOR WOMAN	EH?	OK	OK	OK
ILL-FATED BEING	EH?	OK	EH?	OK
I AM SURE OF IT	EH?	OK	OK	OK
I DENY IT	EH?	OK	I AM	OK
I SAY YOU HAVE	EH?	OK	OK	HOW S
I WILL HAVE IT	EH?	OK	WHAT	EH?
I WILL GO TO THE LAW	EH?	OK	OK	OK
HALF AND HALF	EH?	OK	OK	OK
THEN COME ON	EH?	OK	OK	OK
NOT AT ALL	EH?	EH?	OK	OK
NO SUCH THING, SIR	EH?	OK	OK	OK
ARE YOU MARRIED	EH?	OK	OK	OK
THAT MAY BE	EH?	OK	OK	OK
YOU WILL EXCUSE ME	EH?	OK	OK	OK
WILL YOU BE QUIET	EH?	OK	OK	OK
MY VERY DEAR SIR	EH?	OK	OK	OK
WELL, NOT EXACTLY	EH?	EH?	I NOW	OK
SO AS I DID	EH?	I WIL	OK	OK
WHAT IS IT	EH?	OK	MORNI	OK
WHAT IS YOUR NAME	EH?	OK	OK	ARE Y
DROWN YOURSELF	EH?	OK	OK	OK
THIS IS MY WORK	EH?	OK	OK	OK
THEN YOU ARE LOST	EH?	I DON	OK	OK
I HEARD OF IT	EH?	OK	OK	OK
NO, SIR - YOURS	EH?	OK	OK	OK
I WON'T HAVE HER	EH?	I WIL	I WIL	OK
NO MORE WILL I	EH?	NO NO	OK	OK
PISTOLS FOR TWO	EH?	OK	OK	OK
DON'T THINK OF ME	EH?	OK	OK	OK
THAT'S FAIR ENOUGH	EH?	OK	OK	OK
NOW, THEN, SIR	EH?	OK	OK	OK
I AM READY	EH?	ARE Y	OK	OK
OF COURSE, SIR	EH?	TOMOS	OK	OK
I CONGRATULATE YOU	EH?	EH?	OK	OK
POSTMAN YESTERDAY	EH?	OK	EH?	OK
I AM REALLY SORRY	EH?	OK	OK	OK
SHUT THE DOOR	EH?	OK	OK	OK
I DON'T WANT IT	EH?	OK	OK	OK
NO MORE DO I	EH?	NO NO	NO NO	OK
BETWEEN YOU AND ME	EH?	OK	EH?	OK
JOINT BUT YOURS	EH?	OK	OK	OK
HE SOON REPLIED	EH?	OK	EH?	OK
YOU DON'T SUSPECT ME	EH?	YOU S	OK	OK
YOU SMOKE TOO MUCH	EH?	YOU D	YOU D	OK
FINE, THANK YOU	EH?	EH?	OK	TERRI
SO IT SEEMS	EH?	OK	OK	OK
A PENNY ROLL	EH?	OK	OK	OK
LET US GO	EH?	HE IS	OK	OK
HOW SLEEPY I AM	EH?	OK	OK	OK
I MUST LIE	EH?	OK	ENJOY	OK

STATISTICAL	WORD LIST RECORDED BY JEFF SINGER			
RESULTS	PASS 0	PASS 1	PASS 2	PASS 3
NUMBER OF MESSAGES	561	561	561	561
RECOGNIZED CORRECTLY	0 0.0 %	353 62.9 %	417 74.3 %	513 91.4 %
UNKNOWN OR REJECTED	561 100.0 %	21 3.7 %	7 1.2 %	1 0.2 %
ERRORS	0 0.0 %	187 33.3 %	137 24.4 %	47 8.4 %
AVER. COMP. TIME /MESSAGE	0.9 S	17.4 S	17.0 S	16.3 S

Figure IV-25.(continued). List of 561 English Words .

(S/N $\approx$ 15 db). Since this vocabulary is less well-balanced than the previous one, the results presented are poorer as far as the second and third passes are concerned (the reader should note that some word pairs which are likely to be confused: "relax-relaxed", "serious-seriously", "possible-possibly", "listen-lesson", "thick-sick" are in the vocabulary). Finally, the results obtained for the fourth pass are at the expected level of accuracy. A fifth pass would have yield 94-95% accuracy.

-3. A French word list recorded by the author. To show that the program is independent of any specific language, a test was performed on a French word vocabulary. Figure IV-26 displays the obtained results.

-4. A word list recorded by 10 different speakers, (one each). Two figures: IV-27 and IV-28 summarize this program run. The speakers were randomly chosen amongst the Stanford Artificial Intelligence Laboratory Staff. Figure IV-27 illustrates the results obtained for this random order list. In the second run (Figure IV-28), the order of the speaker was modified so as to place at the beginning of the list, people judged to have dissimilar voices.

The difficulties encountered when dealing with lists recorded by several speakers are numerous, the next paragraph will summarize some of them.

As far as we could determine, the stressed vowel of any word is pronounced quite consistently by different speakers, i.e., the parameters corresponding to Formant1 and Formant2 do not vary very much from one speaker to another. But the remaining part of the words is subject to large variations: These variations affect the stress and the relative position of the sounds within the word. Consonants in unstressed

LIST OF MESSAGES	WORD LIST RECORDED BY PIERRE VICENS				
	PASS 0	PASS 1	PASS 2	PASS 3	PASS 4
FEUILLE	EH?	VIE	OK	OK	OK
ARBRE	EH?	OK	OK	OK	OK
SATISFACTION	EH?	OK	OK	OK	OK
BEAT	EH?	OK	OK	OK	OK
DIZAINE	EH?	EH?	EH?	OK	OK
COUSSIN	EH?	OK	OK	OK	OK
CENDRIER	EH?	OK	OK	OK	OK
PARACAS	EH?	OK	OK	OK	OK
BOITE	EH?	OK	OK	OK	OK
TOTEM	EH?	OK	OK	OK	OK
LAMPE	EH?	OK	OK	OK	OK
LIVRE	EH?	OK	LIRE	OK	OK
DISQUE	EH?	OK	OK	OK	OK
CHEMISE	EH?	OK	OK	OK	OK
CHIMIQUE	EH?	OK	OK	OK	OK
LOI	EH?	OK	OK	OK	OK
VALISE	EH?	OK	OK	OK	OK
APPAREIL	EH?	OCTAL	OK	OK	OK
UN	EH?	OK	OK	OK	OK
DEUX	EH?	EH?	OK	OK	OK
TROIS	EH?	OK	OK	OK	OK
QUATRE	EH?	OK	ARBRE	OK	OK
CINQ	EH?	OK	OK	OK	OK
SIX	EH?	OK	OK	EH?	OK
SEPT	EH?	EH?	OK	OK	OK
HUIT	EH?	OK	OK	OK	OK
NEUF	EH?	OK	OK	OK	OK
MEMOIRE	EH?	EH?	OK	OK	OK
ZERO	EH?	OK	OK	OK	OK
NUIT	EH?	OK	OK	OK	OK
VIE	EH?	HUIT	HUIT	OK	OK
ADDITIONNER	EH?	OK	OK	OK	EH?
SOUSTRAIRE	EH?	OK	OK	OK	OK
DIVISER	EH?	OK	OK	OK	OK
MULTIPLIER	EH?	OK	OK	OK	OK
BUREAU	EH?	OK	OK	OK	OK
DETRUIRE	EH?	OK	ECRIR	OK	OK
DETERIORER	EH?	OK	OK	OK	OK
ACHETER	EH?	OK	OK	EH?	OK
VERMOL	EH?	OK	OK	OK	OK
VIOLET	EH?	OK	OK	OK	OK
INDIGO	EH?	OK	OK	OK	OK
BLEU	EH?	OK	OK	OK	OK
VERT	EH?	UN	EH?	OK	OK
JAUNE	EH?	FEUIL	OK	OK	OK
ORANGE	EH?	EH?	EH?	OK	OK
ROUGE	EH?	LE	OK	OK	OK
LIRE	EH?	OK	OK	LIVRE	OK
ECRIRE	EH?	OK	OK	OK	OK
COMPTER	EH?	OK	OK	OK	OK
DISTRIBUER	EH?	OK	OK	OK	OK
CALCULER	EH?	OK	OK	OK	OK
MOITIE	EH?	OK	OK	OK	OK
QUART	EH?	OK	OK	OK	OK
ENTIER	EH?	MOITI	OK	OK	OK
UNITE	EH?	OK	OK	OK	OK
INSERER	EH?	OK	OK	OK	OK
DECIMAL	EH?	OK	OK	OK	OK
OCTAL	EH?	OK	OK	OK	OK
OCTET	EH?	OK	OK	OK	OK
MACHINE A ECRIRE	EH?	EH?	EH?	OK	OK
LE FILS PRODIGUE	EH?	OK	EH?	EH?	OK
LES PONTS DE LA SEIN	EH?	OK	OK	OK	OK
COMMENT ALLEZ VOUS	EH?	OK	OK	OK	OK
LES PETITS LITS BLAN	EH?	OK	OK	OK	EH?
LE VENT SOUFFLE	EH?	EH?	OK	OK	OK
LA TEMPETE FAIT RAGE	EH?	OK	OK	OK	OK
LES BATEAUX A VOILE	EH?	OK	OK	OK	OK
NOTRE-DAME DE PARIS	EH?	OK	OK	EH?	OK
LES CHAMPS ELYSEES	EH?	OK	OK	OK	OK

NUMBER OF MESSAGES	70	70	70	70	70
RECOGNIZED CORRECTLY	0 0.0 %	57 81.4 %	61 87.1 %	65 92.9 %	68 97.1 %
UNKNOWN OR REJECTED	70 100,0 %	7 10,0 %	5 7.1 %	4 5.7 %	2 2.9 %
ERRORS	0 0.0 %	6 8.6 %	4 5.7 %	1 1.4 %	0 0.0 %
AVER. COMP. TIME /MESSAGE	0.9 S	2.1 S	3.2 S	3.4 S	2.0 S

Figure IV-26. Results obtained for a list of 70 French words .

LIST OF MESSAGES	WORD LIST RECORDED BY 10 DIFFERENT SPEAKERS FROM THE STANFORD A.I. LABORATORY									
	PASS 0	PASS 1	PASS 2	PASS 3	PASS 4	PASS 5	PASS 6	PASS 7	PASS 8	PASS 9
INSERT	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
DELETE	EH?	OK	OK	MAKE	OK	MOVE	OK	OK	OK	OK
REPLACE	EH?	OK	OK	OK	OK	OK	OK	OK	OK	COMPA
MOVE	EH?	OK	OK	OK	WHOLE	OK	ONE	OK	ONE	OK
READ	EH?	OK	OK	OK	OK	NAME	OK	OK	OK	OK
BINARY	EH?	OK	UNITE	OK	BYTE	OK	OK	OK	OK	EH?
SAVE	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
CORE	EH?	FOUR	FOUR	OK	FOUR	OK	OK	OK	OK	FOUR
DIRECTIVE	EH?	OK	OK	OUTPU	OK	OK	OK	OK	OK	OK
LIST	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
LOAD	EH?	OK	OK	OK	CORE	OK	OK	OK	OK	OK
STORE	EH?	OK	OK	CORE	OK	OK	OK	OK	OK	OK
ADD	EH?	OK	OK	OK	OK	ONE	NINE	EH?	HALF	OK
SUBTRACT	EH?	OK	OK	EH?	OK	OK	OK	OK	OK	OK
ZERO	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
ONE	EH?	NINE	OK	UNITE	CORE	EH?	WORD	OK	NINE	OK
TWO	EH?	OK	OK	OK	OK	OK	OK	OK	OK	MOVE
THREE	EH?	OK	READ	OK	OK	OK	OK	OK	OK	OK
FOUR	EH?	STORE	CORE	OK	OK	CORE	OK	CORE	OK	CORE
FIVE	EH?	EH?	ADD	OK	OK	ADD	OK	OK	OK	OK
SIX	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
SEVEN	EH?	OK	OK	OK	SAVE	OK	SAVE	OK	SAVE	OK
EIGHT	EH?	OK	OK	OK	OK	OK	OK	NAME	OK	MAKE
NINE	EH?	OK	OK	ONE	ADD	ADD	OK	OK	OK	OK
MULTIPLY	EH?	OK	OK	OK	OK	OK	INPUT	OK	OK	OK
DIVIDE	EH?	OK	OK	NINE	OK	OK	NINE	OK	EH?	OK
NUMBER	EH?	EH?	OK	OK	OK	OK	OK	OK	OK	OK
NAME	EH?	READ	OK	OK	OK	EIGHT	END	MEMOR	OK	READ
END	EH?	EH?	NAME	NAME	MAKE	OK	OK	OK	OK	OK
SCALE	EH?	OK	OK	SAVE	SKIP	STORE	SAVE	OK	OK	OK
CYCLE	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
SKIP	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
JUMP	EH?	OK	BYTE	OK	OK	OK	EH?	OK	OK	EH?
ADDRESS	EH?	OK	OK	OK	OK	OK	OK	REPLA	OK	OK
OVERFLOW	EH?	MULTI	OK	OK	OK	OK	OK	OK	OK	OK
POINT	EH?	OK	OK	OK	FOUR	BYTE	OK	OK	EIGHT	OK
CONTROL	EH?	EH?	OK	OK	OK	OK	OK	OK	OK	OK
REGISTER	EH?	EH?	OK	OK	EH?	OK	EH?	OK	OK	OK
WORD	EH?	LOAD	CORE	OK	OK	OK	ONE	OK	OK	OK
EXCHANGE	EH?	OK	OK	OK	SCALE	OK	OK	OK	OK	SCALE
INPUT	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
OUTPUT	EH?	OK	OK	OK	OK	OCTAL	OCTAL	OK	OK	OCTAL
MAKE	EH?	POINT	EIGHT	POINT	READ	OK	EIGHT	OK	OK	OK
INTERSECT	EH?	INSER	OK	OK	OK	OK	OK	OK	OK	OK
COMPARE	EH?	DIVID	OK	MEMOR	OK	OK	OK	OK	OK	OK
ACCUMULATE	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
MEMORY	EH?	NUMBE	UNITE	OK	OK	NUMBE	NUMBE	OK	NINE	OK
BYTE	EH?	OK	FIVE	OK	OK	OK	OK	OK	HALF	OK
QUARTER	EH?	EH?	WORD	OK	OK	WORD	OK	OK	OK	OK
HALF	EH?	EH?	ADD	JUMP	JUMP	NINE	EH?	BYTE	OK	OK
WHOLE	EH?	FOUR	OK	OK	OK	OK	CORE	FOUR	OK	OK
UNITE	EH?	OK	OK	DELET	BYTE	OK	MAKE	OK	EH?	MAKE
DECIMAL	EH?	OK	EH?	OK	OK	OK	OK	OK	OK	OK
OCTAL	EH?	OK	OUTPU	OK	OK	OK	OUTPU	OUTPU	OK	OK
NUMBER OF MESSAGES	54	54	54	54	54	54	54	54	54	54
RECOGNIZED CORRECTLY	0 0.0 %	36 66.7 %	39 72.2 %	41 75.9 %	39 72.2 %	40 74.1 %	36 66.7 %	46 85.2 %	45 83.3 %	43 79.6 %
UNKNOWN OR REJECTED	54 100.0 %	7 13.0 %	1 1.9 %	1 1.9 %	1 1.9 %	1 1.9 %	3 5.6 %	1 1.9 %	2 3.7 %	2 3.7 %
ERRORS	0 0.0 %	11 20.4 %	14 25.9 %	12 22.2 %	14 25.9 %	13 24.1 %	15 27.8 %	7 13.0 %	7 13.0 %	9 16.7 %
AVER. COMP. TIME /MESSAGE	0.9 S	3.5 S	4.3 S	7.1 S	7.2 S	6.3 S	7.3 S	9.0 S	11.4 S	12.0 S

Figure IV-27. Results obtained for a list of 54 words recorded by 10 speakers .

LIST OF MESSAGES	WORD LIST RECORDED BY 10 DIFFERENT SPEAKERS FROM THE STANFORD A.I. LABORATORY									
	PASS 0	PASS 1	PASS 2	PASS 3	PASS 4	PASS 5	PASS 6	PASS 7	PASS 8	PASS 9
INSERT	EH?	OK	EH?	OK	OK	OK	OK	OK	OK	OK
DELETE	EH?	OK	MOVE	NAME	POINT	OK	OK	OK	OK	OK
REPLACE	EH?	EH?	OK	EH?	OK	OK	OK	OK	OK	OK
MOVE	EH?	WHOLE	OK	OK	OK	LOAD	OK	OK	OK	LOAD
READ	EH?	DELET	OK	OK	OK	NAME	OK	OK	OK	OK
BINARY	EH?	DIVID	EH?	EH?	OK	OK	OK	MEMOR	OK	OK
SAVE	EH?	OK	THREE	EXCHA	OK	OK	OK	OK	OK	OK
CORE	EH?	WHOLE	OK	FOUR	FOUR	OK	FOUR	FOUR	OK	OK
DIRECTIVE	EH?	EH?	OCTAL	OK	OK	OK	OK	OK	EH?	OK
LIST	EH?	EH?	OK	EH?	OK	OK	OK	OK	OK	OK
LOAD	EH?	OK	OK	OK	OK	OK	MOVE	OK	OK	OK
STORE	EH?	FOUR	OK	OK	OK	EH?	CORE	FOUR	OK	OK
ADD	EH?	EH?	ONE	EH?	NINE	HALF	OK	OK	EH?	OK
SUBTRACT	EH?	EH?	OK	OK	OK	OK	OK	OK	OK	OK
ZERO	EH?	EH?	ONE	EH?	OK	OK	TWO	OK	OK	OK
ONE	EH?	WORD	OK	EH?	LOAD	EH?	MOVE	OK	OK	OK
TWO	EH?	OK	OK	LOAD	OK	OK	OK	OK	OK	OK
THREE	EH?	EH?	QUART	OK	TWO	OK	OK	READ	OK	OK
FOUR	EH?	CORE	POINT	CORE	CORE	OK	OK	CORE	OK	OK
FIVE	EH?	ONE	ADD	OK	NINE	OK	OK	DIVID	OK	OK
SIX	EH?	EH?	OK	OK	OK	OK	OK	OK	OK	OK
SEVEN	EH?	OK	OK	OK	OK	SAVE	OK	OK	CYCLE	OK
EIGHT	EH?	LIST	NAME	NAME	NAME	NAME	READ	OK	NAME	OK
NINE	EH?	ADD	ADD	EH?	OK	EH?	ONE	OK	OK	OK
MULTIPLY	EH?	OK	OK	EH?	OK	OK	EH?	OK	OK	OK
DIVIDE	EH?	OK	OK	EH?	OK	EH?	OK	OK	FIVE	OK
NUMBER	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
NAME	EH?	READ	READ	READ	READ	OK	OK	OK	OK	READ
END	EH?	EH?	OK	NINE	NAME	OK	NAME	NAME	OK	OK
SCALE	EH?	SAVE	EH?	SAVE	OK	OK	SAVE	OK	OK	OK
CYCLE	EH?	OK	SEVEN	EH?	OK	OK	OK	OK	SEVEN	OK
SKIP	EH?	OK	EH?	OK	OK	OK	OK	OK	OK	OK
JUMP	EH?	END	OK	EH?	OK	OK	POINT	BYTE	OK	OK
ADDRESS	EH?	EH?	WRITE	OK	OK	OK	OK	OK	REPLA	OK
OVERFLOW	EH?	EH?	MULTI	MULTI	OK	OK	OK	OK	OK	OK
POINT	EH?	OK	WORD	OK	OK	EIGHT	OK	OK	OK	OK
CONTROL	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
REGISTER	EH?	EH?	OUTPU	OK	OK	OK	OK	OK	OK	EH?
WORD	EH?	ONE	ONE	OK	EH?	OK	OK	OK	OK	ONE
EXCHANGE	EH?	SAVE	EH?	SCALE	SCALE	SCALE	EH?	OK	OK	OK
INPUT	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
OUTPUT	EH?	OK	OK	OK	OK	OK	OK	OK	OK	OK
PARK	EH?	DELET	EIGHT	EIGHT	READ	OK	POINT	EIGHT	OK	OK
INTERSECT	EH?	OK	OK	EH?	OK	EH?	INSER	OK	OK	OK
COMPARE	EH?	EH?	END	EH?	OK	EH?	OK	OK	OK	OK
ACCUMULATE	EH?	OK	ZERO	EH?	OK	OK	OK	EH?	OK	OK
MEMORY	EH?	EH?	NUMBE	OK	NUMBE	NINE	OK	OK	OK	OK
BYTE	EH?	JUMP	JUMP	JUMP	OK	OK	OK	OK	OK	OK
QUARTER	EH?	EH?	WORD	OCTAL	NUMBE	OK	OK	WORD	OK	OK
HALF	EH?	JUMP	NINE	EH?	BINAR	OK	OK	WRITE	OK	EH?
WHOLE	EH?	POINT	MOVE	OK	OK	LOAD	MOVE	OK	FOUR	OK
WRITE	EH?	DIVID	NINE	EH?	POINT	EH?	OK	OK	OK	OK
DECIMAL	EH?	EH?	OK	EH?	OK	OK	OK	OK	OK	OK
OCTAL	EH?	OUTPU	OK	OUTPU	OUTPU	OUTPU	OK	OK	OUTPU	OK
NUMBER OF MESSAGES	54	54	54	54	54	54	54	54	54	54
RECOGNIZED CORRECTLY	0	17	23	22	36	37	39	42	45	49
	0.0 %	31.5 %	42.6 %	40.7 %	66.7 %	68.5 %	72.2 %	77.8 %	83.3 %	90.7 %
UNKNOWN OR REJECTED	54	16	5	17	1	7	2	1	2	2
	100.0 %	29.6 %	9.3 %	31.5 %	1.9 %	13.0 %	3.7 %	1.9 %	3.7 %	3.7 %
ERRORS	0	21	26	15	17	10	13	11	7	3
	0.0 %	38.9 %	48.1 %	27.8 %	31.5 %	18.5 %	24.1 %	20.4 %	13.0 %	5.6 %
AVER. COMP. TIME /MESSAGE	6.6 S	3.5 S	4.1 S	6.3 S	6.6 S	9.3 S	10.2 S	8.0 S	9.2 S	9.4 S

Figure IV-28. Results obtained for a list reordered on the basis of voice dissimilarity between speakers .

syllables may contain less fricative noise, a weaker stop burst release, or an incomplete stop closure as compared to the same consonants in stressed syllables . The substitution of an incomplete gesture for a consonant cluster is also common in unstressed syllables of natural speech. None of these effects would necessarily produce word recognition difficulties if they appeared consistently in the data. Unfortunately, they do not, and their range of variation is quite large. Although we cannot provide the original sound samples with this dissertation, the variability between speakers is illustrated by a set of photographs: Figures IV-29 to IV-34.

Figure IV-29 represents an utterance of MULTIPLY provided with three distinct vowels. Figure IV-30 and IV-31 show a different utterance of MULTIPLY. This last sample is characterized by a complete absence of the /t/, in fact the sound heard is more like MULIPLY than MULTIPLY. The program's incorrect identification given for this utterance was OUTPUT. Figure IV-32 through IV-34 show two utterances in which the first unstressed vowel is very mild and was incorrectly classified by the segmentation program. Both resulted in incorrect identifications, i.e., EXCHANGE was identified as SCALE and DIVIDE as FIVE. It is probable that for this latter case, the error recovery procedure would have corrected the classification error, but since the similarity score for FIVE was quite high (83%) it was returned as the response and no correction was attempted.

Also, we found that the variable subject-to-microphone distance was addding a non-negligible variation factor to the experiment. A microphone held too close to the lips records the expiration after the

Figure IV-29. Utterance of MULTIPLY showing three distinct vowels .

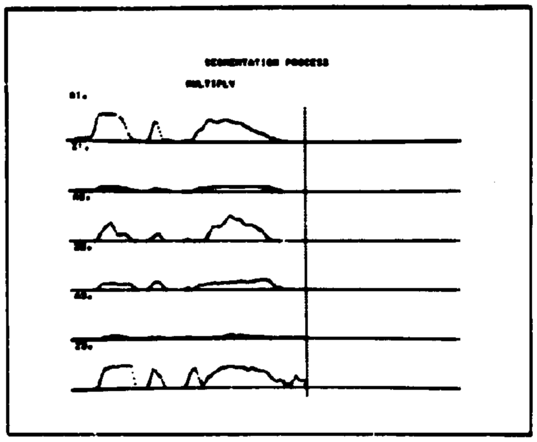


Figure IV-30. Different utterance of MULTIPLY in which the /t/ was not pronounced .

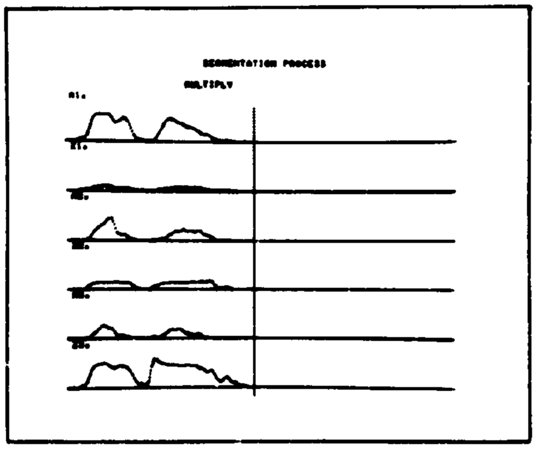


Figure IV-31. Resulting classification error, one vowel has not been detected .

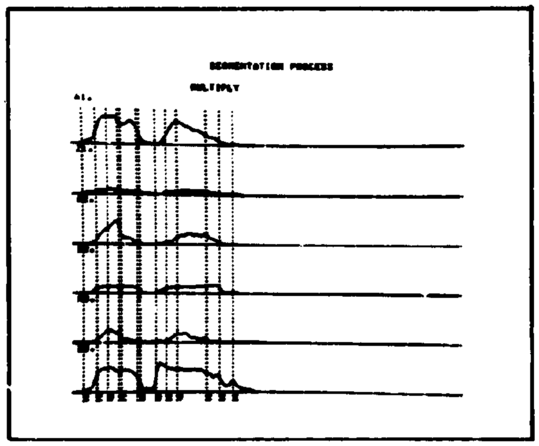


Figure IV-32. Utterance of EXCHANGE in which the first vowel /e/ was not clearly pronounced .

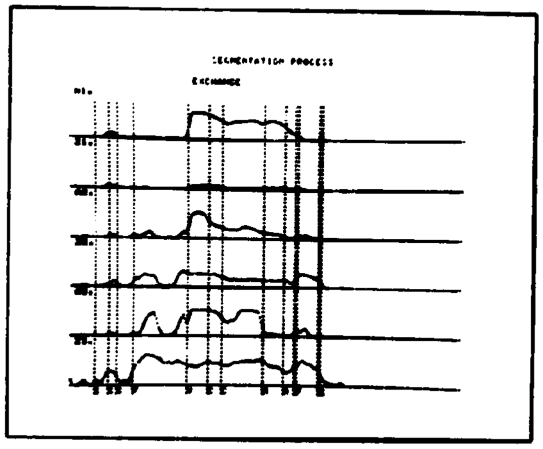
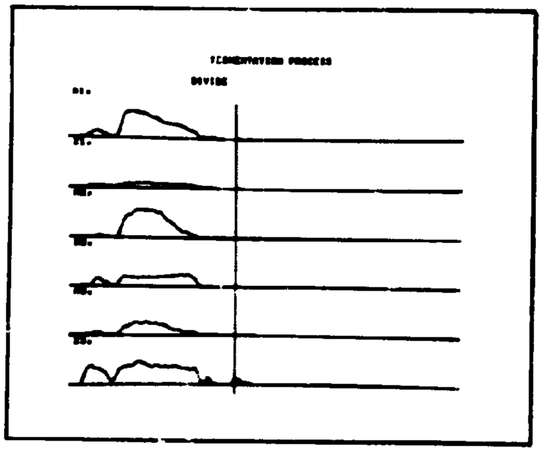


Figure IV-33. Utterance of DIVIDE in which the first vowel /i/ is almost inexistant .



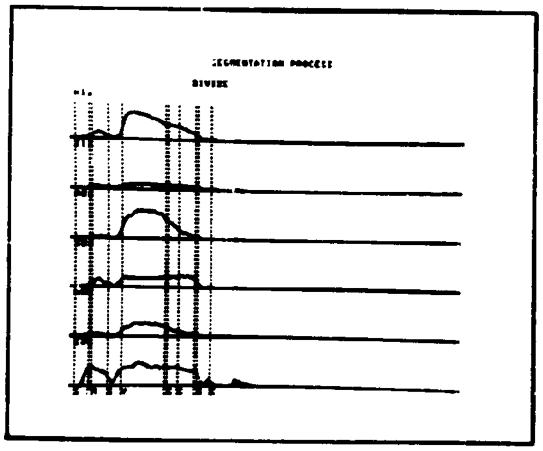


Figure IV-34. Resulting classification error .

utterance, giving the illusion of an extra sound. This difficulty is usually corrected by the use of a wind screen or by special heuristics within the program. More work is needed to reliably discriminate between speech and expirations (note that /h/ sounds are special cases of expiration). On the other hand, a microphone held too far away from the mouth results in loss of resolution due to the ensuing lower signal-to-noise ratio. Some of the samples were in fact so noisy that the preprocessor service routine had trouble detecting the utterance when reading the original audio tape.

IV-7. EVALUATION OF SOME OF THE PRINCIPAL HEURISTICS

Since the heuristics interact, it is in general very hard, if not impossible, to evaluate the contribution of one heuristic to the overall performance of a given program. In our case, for example, it would be impossible to predict the overall change in the program's quality resulting from a heuristic modification in the segmentation procedure or from a different lexicon organization. Fortunately, the few important heuristics of the candidate selection process are not likely to interact with the others. To give a crude evaluation of these, one can simply suppress them from the program and show the resulting degradation of quality. In this section we shall illustrate the effectiveness of the following heuristics:

-1. The vowel mapping correction heuristic which redefines the links between corresponding vowels on the basis of similarity of parameters. (subsection IV-4-2)

-2. The error recovery heuristic (or multisearch heuristic) which initiates a second level of searches if no satisfactory candidate is found during a primary search (subsection IV-4-4).

-3. The excellent-match-candidate heuristic which stops the process immediately if a candidate with a similarity score higher than 95% if found.

Figure IV-35 summarizes the results obtained with each of these heuristics being turned off.

-Table (A) presents the results obtained with the full program for the list recorded by Dr. K. Stevens which was experimented on.

-Table (B) presents the results obtained with a program not

STATISTICAL RESULTS	PASS 0	PASS 1	PASS 2	PASS 3	PASS 4
NUMBER OF MESSAGES	54	54	54	54	54
RECOGNIZED CORRECTLY	0 0.0 %	53 98.1 %	53 98.1 %	53 98.1 %	54 100.0 %
UNKNOWN OR REJECTED	37 68.5 %	0 0.0 %	0 0.0 %	0 0.0 %	0 0.0 %
ERRORS	17 31.5 %	1 1.9 %	1 1.9 %	1 1.9 %	0 0.0 %
AVER. COMP. TIME /MESSAGE	1.9 S	2.0 S	2.0 S	2.0 S	2.5 S

(A)

STATISTICAL RESULTS	PASS 0	PASS 1	PASS 2	PASS 3	PASS 4
NUMBER OF MESSAGES	54	54	54	54	54
RECOGNIZED CORRECTLY	0 0.0 %	46 85.2 %	47 87.0 %	47 87.0 %	51 94.4 %
UNKNOWN OR REJECTED	40 74.1 %	6 11.1 %	6 11.1 %	5 9.3 %	2 3.7 %
ERRORS	14 25.9 %	2 3.7 %	1 1.9 %	2 3.7 %	1 1.9 %
AVER. COMP. TIME /MESSAGE	1.2 S	1.4 S	2.0 S	2.2 S	2.1 S

(B)

STATISTICAL RESULTS	PASS 0	PASS 1	PASS 2	PASS 3	PASS 4
NUMBER OF MESSAGES	54	54	54	54	54
RECOGNIZED CORRECTLY	0 0.0 %	51 94.4 %	52 96.3 %	52 96.3 %	53 98.1 %
UNKNOWN OR REJECTED	37 68.5 %	2 3.7 %	1 1.9 %	0 0.0 %	0 0.0 %
ERRORS	17 31.5 %	1 1.9 %	1 1.9 %	2 3.7 %	1 1.9 %
AVER. COMP. TIME /MESSAGE	1.8 S	2.6 S	2.7 S	3.0 S	2.9 S

(C)

STATISTICAL RESULTS	PASS 0	PASS 1	PASS 2	PASS 3	PASS 4
NUMBER OF MESSAGES	54	54	54	54	54
RECOGNIZED CORRECTLY	0 0.0 %	53 98.1 %	53 98.1 %	53 98.1 %	54 100.0 %
UNKNOWN OR REJECTED	37 68.5 %	0 0.0 %	0 0.0 %	0 0.0 %	0 0.0 %
ERRORS	17 31.5 %	1 1.9 %	1 1.9 %	1 1.9 %	0 0.0 %
AVER. COMP. TIME /MESSAGE	1.8 S	2.3 S	3.1 S	3.0 S	3.4 S

(D)

Figure IV-35. Evaluation of some of the Candidate Selection Heuristics .

provided with the vowel mapping correction process. The overall quality of the program, in all the passes, has decreased by a factor of 5%.

-Table (C) presents the results obtained with a program not provided with the error recovery procedure. The overall quality has again dropped by 5%, and the results exhibit the fact that this heuristic is more effective at the beginning when there are fewer samples in the lexicon. On the other hand, this heuristic is time consumming, and could be suppressed if one is more interested in speed than in performance, (for example, in real-life use of the program, the speaker can occasionally be requested to repeat his last utterance if the program failed to recognize it).

-Table (D) presents the results obtained with a program not provided with the excellent-match-candidate heuristic. The quality is at the same level but the time taken to identify a word is increased by 30%. The effectiveness of this heuristic, which increases with the number of stored samples for a given utterance, is also exhibited in the preceding results. In all the runs previously presented, the average computation time per message drops by an appreciable amount for the last lists to be processed. This fact, which may seem abnormal since the size of the lexicon is increasing, results from an increase in the number of successful applications of this heuristic.

IV-8. CONCLUSIONS

The results presented show that recognition of speech does not depend so much on the accuracy of the utilized parameters since high scores were achieved with very crude and questionable parameters. Any other techniques: spectrum analysis, formant trackers, polynomial expansions, etc... can be expected to work reasonably well provided subsequent algorithms are carefully designed. In the same manner, an accurate phoneme-like classification is unnecessary, although such features might be useful to reduce the search, it remains to be seen whether the reduction resulting from accurate classification will exceed the amount of effort required to perform the necessary classification error recovery task.

As far as we can determine, the problem of devising heuristic procedures to reduce the search space in speech recognition is investigated for the first time. These procedures are effective and it is clear that they have to be further developed if a speech recognition system having close to human abilities is desired.

Chapter V

THE HAND-EYE-EAR SYSTEM

V-1. INTRODUCTION

Shannon, Minsky, McCarthy, and others have considered the possibility of a computer with hands, eyes, and ears at one period or another during the latter part of the last decade. The main obstacles to the realization of the idea were the unavailability of suitable computers and Input/Output devices, and the prohibitive cost of such a system. Ernst (1961) and Roberts (1963) were among the first few who used a computer to realize these objectives. Glaser, McCarthy and Minsky (1964) proposed that the first major attempt at the biological exploration of Mars would be made by a computer controlled automatic laboratory, containing a wide variety of perceptual input devices and mechanical manipulators which can perform, under computer control, many of the tasks of bio-chemical laboratory, requiring only a limited supervision by the experimenter on earth.

At Stanford Artificial Intelligence Project an integrated HAND-EYE system had been implemented under the PDP-6 time-sharing system. This initial system, able to perform simple sorting and stacking operations on cubical blocks, has been described in detail in some recent publications by Wichman (1967); Pingle (1966); Pingle, Singer and Wichman (1968); and Pieper (1968). As an illustration of the existing capabilities of the speech recognizer program, it was decided to provide this HAND-EYE system with a speech analysis program able to "understand" spoken commands, thus building an integrated HAND-EYE-EAR system that obeys the experimenter's voice.

In this chapter, we shall describe the overall structure of the system with some emphasis on the speech analysis program, since it is more within the scope of the present dissertation. In the rest of the chapter we will describe the following aspects of the HAND-EYE-EAR system:

..Presentation of the system configuration with a few details on the "eye" and "arm" subsystems. A complete description is given in the references previously mentioned. (Although this section is not the result of research made by the author, it was added for the sake of completeness.)

..Analysis of the different calibration and training operations one must perform in order to run the system.

..Description of the "ear" subsystem. The use of a grammar (whose BNF is given) with a convenient vocabulary (terminal symbols) allow the decoding of long sentences (up to 5 seconds long) in 5 to 10 seconds.

..Presentation of the results obtained and conclusions. A sequence of snapshots of the program in operation as depicted on the CRT display attached to the computer, and some statistical results obtained from the data generated by four different speakers will appear at the end of the chapter.

V-2. SYSTEM DESCRIPTION

V-2-1. Task Description

Several blocks sre scattered on a large table provided with one TV camera, an electric manipulator and a TV monitor (Figure V-2). Nearby, an experimenter, holding a microphone, watches the screen of a CRT display (Figure V-1). The blocks in the field of view of the TV camera are traced on the display screen. Then the computer, using the CRT to express itself, requests a command from the experimenter. A sentence like "PICK UP THE SMALL BLOCK STANDING ON THE TOP RIGHT CORNER" is spoken into the microphone. The computer "thinks" for 4 to 6 seconds, the arm moves, picks up the specified block and places it at the top of a stack. A new scene representation which omits the block just taken away appears on the screen of the CRT, and the process continues. The previous scenario is usually executed in about 30 seconds. For a computer to be able to perform such relatively complicated tasks requiring interaction with its environment without human intervention, it must be provided with two distinct parts:

-Certain special hardware items such as a microphone, a TV camera, an artificial arm manipulator, etc...

-A program residing in the computer memory which specifies the behavior of the system.

V-2-2. Hardware Configuration

The current HAND-EYE-EAR system consists of a vidicon television camera, an electrically powered arm, and a common microphone, all connected to the PDP-6 - PDP-10 computer system (Figure I-1).

Figure V-1. The HAND-EYE-EAR system .

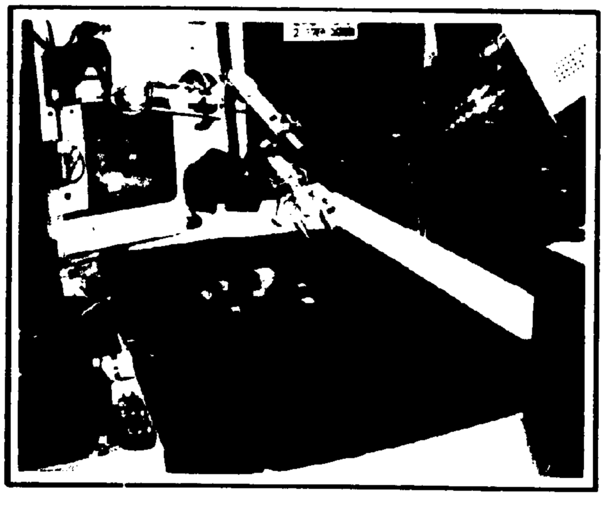


Figure V-2. Details of the HAND-EYE part .

Visual input to the system is provided by the vidicon television camera operating in accordance with EIA standards. The video signal is digitized to 4 bits (16 levels of light intensity) and sampled at an instantaneous rate of 6.5 million samples per second. Making use of interleaving, any rectangular portion of the image, up to 666 x 500 points for the full field of view, may be read into memory under program control in two video frame times (1/15 seconds).

The electric arm was originally designed as a device to be strapped to a paralyzed human arm. Six degrees of freedom permit it to place its "hand" in arbitrary positions and orientations within its reach, plus a finger-closing motion. It is powered by small permanent magnet gear-head motors mounted on the arm, giving joint velocities of 4 to 6 r.p.m. with small loads. Position feedback is provided by potentiometers mounted at each of the six joints. The hand is a two finger parallel grip device and is approximately the size of a human hand. The maximum reach of the arm is about 68 centimeters (27 inches) and its weight is about 7 kilograms (15 pounds). Power to the actuator motors is supplied by a series of constant-width 16 volt pulses, whose repetition rate is determined by the controlling program.

Audio input to the system is provided with a crystal microphone connected to the hardware preprocessor which is described in the Chapter II of the present dissertation. Sentences up to 5 seconds long are read and processed into the computer memory, each time the experimenter is requested by the computer to speak a command into the microphone.

Other peripheral equipment used includes a point-plotting CRT display (DEC model 30), a standard TV monitor, and a display (or teletype) console. The arm and camera are mounted on the top of a large table. The other equipment is distributed around the table at convenient locations. A complete view of the system is given on Figure V-1.

V-2-3. Software Configuration

All of the programs of the HAND-EYE-EAR system are run under the PDP-6 - PDP-10 dual processor time-sharing system. The present set of programs consists of:

-1. An "eye" section which is capable of reading the camera and which generates a scene description. If we digitize the light intensity at every point in the whole field of view of the television camera, the computer will receive 666 x 500 or 333,000 samples, or 1,332,000 bits of information per frame. The problem of scene description is the formulation of routines which will abstract meaningful descriptions of objects of interest in the scene and their positions.

The existing eye program locates cubical blocks of various sizes scattered at random on a contrasting background. Depth determination depends on the assumption that all objects rest on a known planar surface ("The support hypothesis", Roberts, (1963)). Mathematically, this is simply a mapping between two fixed planes which, within the HAND-EYE-EAR system are the top of the table and the image plane of the camera. The edge tracing program in use does not reliably detect subtle differences in brightness between adjacent surfaces of the same or similar objects, so that only the outlines (or exterior edges) of the objects are traced. (Figure V-9).

-2. An "arm" section which, given the position of a cube, can pick it up and move it to a desired position. In order for a manipulator to grasp objects with arbitrary positions and orientations, it must have at least seven degrees of freedom: three for position, three for orientation and one for grasping. The geometric configuration of the arm is shown on Figure V-3. 1,2,........6,7 represent the actuators; S1, S2, S3 are the links of the arm and P0, P1, P2, are the shoulder, elbow and wrist joints respectively.

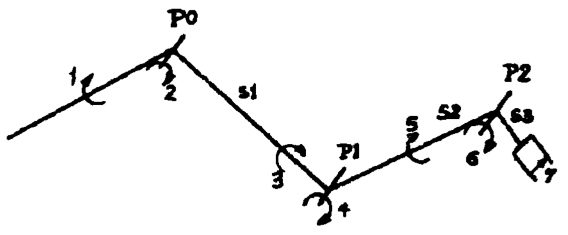


Figure V-3. Schematic Representation of the Electric Arm

To solve the arm positioning problem one must compute the deflection angles required to achieve a desired position and orientation. In general, there are multiple solutions to any given positioning problem (in our case 16 solutions). However, not all of these are realizable since the arm has mechanical stops which limit the deflections to certain ranges. Several methods of solving the positioning problems with or without constraints (e.g., existence of obstacles) have been devised by Singer, Pingle, Wichman (1968) and Pieper (1968). The existing control system used with the arm consists of an analog-to-digital converter for reading potentiometers on joints, an output register for motor pulsing, and a servo-control program. Since this program must operate in real time within the time-sharing system, it is treated as a special case by the system, and is given control every 16.7 milliseconds (or 60 times a second). The program acts as a simple proportional servo which calculates the pulse rate for each motor. Velocity damping is unnecessary since the joints have a great deal of internal friction.

-3. An "ear" section which is capable of reading the microphone and which generates a sound description. The speech signal, as reflected by the changes in voltage generated by the microphone, results in a data rate of about 180,000 bits per second of speech. Typically, a one second interval of a normal utterance consists of between 5 to 10 different sounds, which usually require less than 50 bits to represent in the written form. The preceding chapters presented a way of reducing a speech sample to its written representation. All the procedures previously described are used here to analyze the spoken command and reduce it to a command word of 36 bits of information. No major changes were made in the preprocessing, segmentation and classification procedures. The speech analysis control program was modified to determine the boundaries of words before recognizing them separately, and the word recognition procedure was altered to permit the removal from the list of candidates of all of those which are not wanted. (To allow feedback from a specified grammar.)

-4. A control program which sequences the other programs. To perform the block-stacking task described about, the control program first initiates the "eye" routines to obtain a scene description, then the "ear" routines which request a command from the experimenter, analyze it and return a command word. Using the scene description, the control program chooses a block with respect to the experimenter's directive, the block position is computed and the "arm" program is called to pick it up.

-5. A set of calibration and training routines which relate the camera and arm coordinate systems to the work-space and allow the user to train the word recognizer of the "ear" program.

This collection of programs occupies approximately 75,000 words of storage including data areas. The languages chosen were machine language and FORTRAN IV and no major interfacing problems were found when loading the entire system in the same load-module. Figure V-4 represents a diagram of the system showing the flow of data from one process to another.

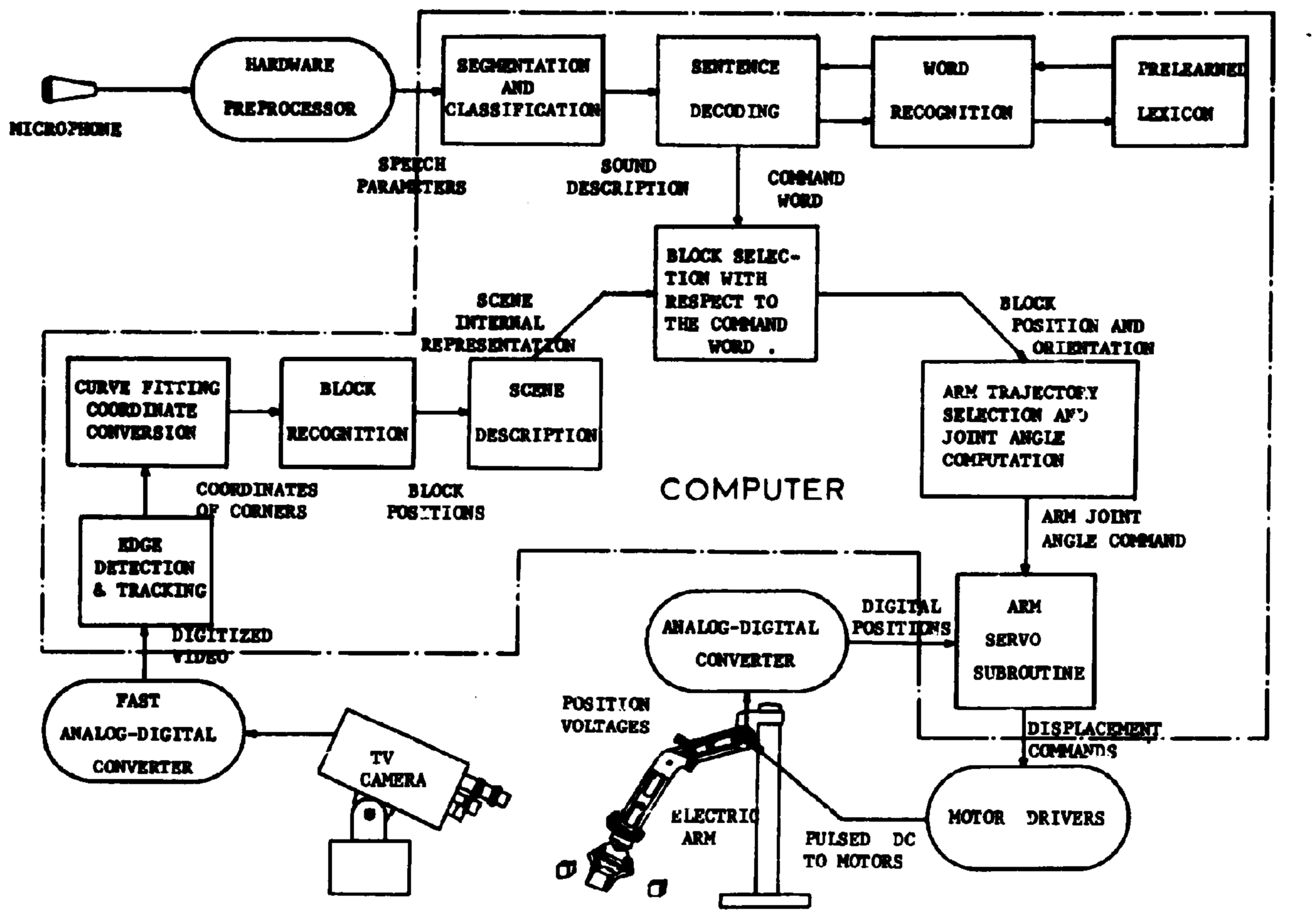


Figure V-4. Diagram of the HAND-EYE-EAR system .

V-3. CALIBRATION AND TRAINING OPERATIONS

Before running the program, several calibrations of the "eye" and the "arm" subprograms have to be performed. Likewise, the "ear" part must be trained to build up the <u>lexicon</u> needed by the word recognizer (candidate selection process).

During the initialization phase, the "eye" routines execute two distinct calibrations without human intervention:

-The first one determines the clipping levels of the TV camera based on the range of variations of light intensities of the scene. These two voltage levels define the "black" (4 bit value 0) and the "white" (4 bit value 15) of the digitized TV image. Between these two extremes, the light intensity is digitized to 4 bits which give 16 levels of brightness. Since the illumination of the scene is likely to change from one experiment to the other, one has to adjust these levels each time a new experiment is initiated. The calibration is achieved by scanning the video output of the TV camera trying several voltage values (8 clipping levels are provided, thus making 28 possible combinations) until the darkest parts of the scene are "black" and the brightest are "white", thereby giving the largest possible resolution.

-The second one relates four points on the table to their positions in the TV image. The only visual information available to the computer is the TV image. A theory, described by Roberts (1963), shows that the knowledge of the coordinates of 4 points of a plane, and the coordinates of their images on the image plane of a camera, provides enough information to define a geometric transformation mapping the points of the plane to their images in the image plane. Of course, this transformation is incomplete since only the points of the plane are correctly transformed (6 points are necessary for a full spacial transformation). The "eye" calibrating routine traces around four rightangled triangles permanently placed on the table and locates their right angles, whose fixed position on the table is known by the routine. Figures V-5, V-6, V-7 illustrate this calibration step. The routine uses this information to compute a matrix used for transforming any point in the camera coordinate system into the coordinate system on the table top.

The only information available to the "arm" routines is the resistance values of the potentiometers placed at each joint. These resistances transformed into voltages by an external power supply, are measured by the analog-to-digital converter. The converter output levels must be related to the corresponding joint angles. To perform this operation the arm is placed in a standard position, in which the relative position of each joint and the position of the hand in the coordinate system on the table top and the vertical axis, is fixed. In this arm position each joint angle value is known by the program and the corresponding resistance value is read by the A-D converter, thus mapping the arm angular coordinate system to the table-top coordinate system.

Two training facilities are provided by the "ear" routines. The simplest one allows the program to read in and use a lexicon built during a preceding session. The other one provides for feeding the program with new word representations, the written form of these words being typed by the experimenter. Of course, both methods can be used concurrently, i.e., reading an old lexicon and completing it with new word representations. Training the word recognizer using isolated words

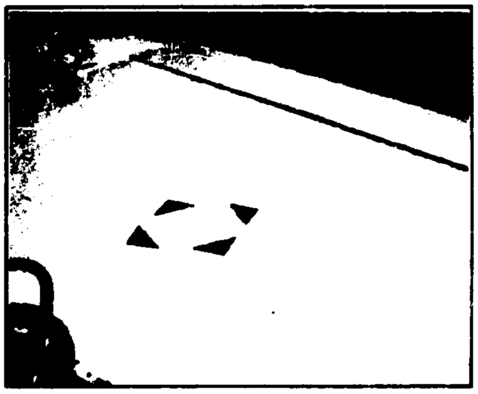


Figure V-5. View of the table top showing the four triangles used to compute the transformation matrix .

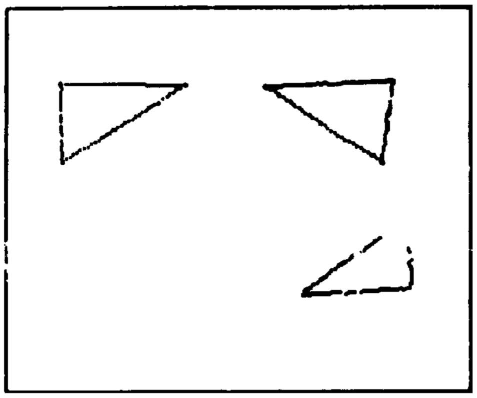


Figure V-6. The edge follower is tracing the sides of the rightangled triangles . The results of its analysis appears on the CRT display .

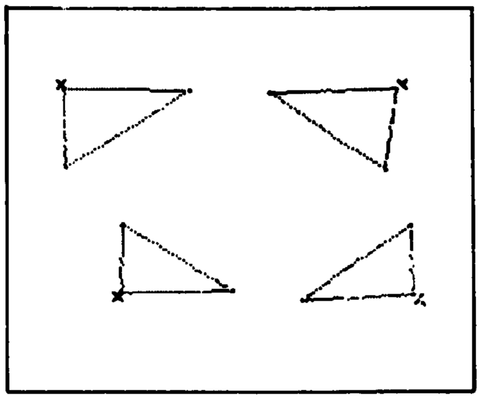


Figure V-7. The triangles have been traced, and the four corners used by the calibration routine are marked .

was found to be unsatisfactory for two principal reasons:

-1. The first problem is that vowels are stressed differently for isolated words than for the same words used in a long sentence.

-2. The second is the problem of word ending. When a word is part of a connected utterance, it ends quite rapidly when the next word begins. If this word is at the end of a sentence or isolated, the sound dies down slowly, thus giving a completely different acoustic description.

Therefore, the lexicon was built using fragments of short sentences containing the desired words. The process is as follows:

-1. A sentence is read into the computer through the microphone and the hardware preprocessor.

-2. The sentence is segmented and the segments are classified into phoneme groups.

-3. The display of the sentence representation appears at the top of the CRT screen.

-4. On the basis of what was said and the segmentation results, the experimenter selects a fragment of the sentence, and passes it to the candidate selection process. If this portion is recognized correctly, the corresponding representation is not learned (not introduced in the lexicon). If the routine fails to recognize it, the experimenter is requested to type the equivalent written-form. The representation is then introduced in the lexicon with its written equivalent. In the section <u>results and conclusions</u> (section V-5) we exhibit the set of sentences used to train the program for obtaining statistical results.

For a given speaker, the program needs an average of two or three representations for each <u>word</u> to be recognized. This number increases slightly if the number of speakers increases (for 4 speakers, an average of 5 to 7 representations is necessary).

V-4 THE "EAR" PART OF THE "HAND-EYE-EAR" PROGRAM.

The problem, simply stated, is to recognize commands to the system in quasi natural english. Since the system is only able to look at a given scene and pick up blocks, the vocabulary is very limited. However, the sentences used to define size and position of a block are typically 2 to 4 seconds long and contain between 5 and 15 words. Furthermore, the number of different valid sentences is large and all of them have to be interpreted correctly. These constraints (duration and number of the valid sentences) prevented us from using the scheme previously described (Chapter IV) in which an utterance is recognized as a whole. Therefore, we decided to break up the connected speech utterances into several words composed of syllables, each word being recognized separately. To accomplish this we had to solve the problem of locating word boundaries in connected speech.

Like many other aspects of English, the problem of locating word boundaries in connected speech can be ambiguous. For example, sound description /AISKRIM/ could have resulted from the words "I scream" or "ice cream". One obvious solution to this problem is to require the speaker to pause for a few milliseconds between words or phrases. But this gets to be annoying after a while. To render this problem unambiguous, a better solution is to use a grammar acting on a restricted vocabulary. For example, let us consider connected speech utterances of the form.

(See next page)

```
<command>::=<function name> <argument list>
<argument list>::=<argument>|<argument list> <preposition> <argument>
<function name>::=PICK UP|STACK|ASSIGN|ADD|SUBTRACT|....
<argument>::= BIG|BLOCK|LEFT|SIDE|....
<prepositions>::= AT|TO|OF|FROM|....
```

By carefully choosing the function names, the possible arguments, and the associated prepositions, it is possible to determine the word phrase boundaries. Certain keywords play an important part in this determination. A good example of such a word is BLOCK, which starts a silence (B) and ends with a silence (K). The syllable in between these two is not likely to be modified by the adjacent sounds. In other words, BLOCK can be recognized in any context with a low percentage of error. As a result, such a heuristic as "scan until you find BLOCK" may be used quite safely. Use of restricted special purpose command languages for communication to the computer such as the one above is not unreasonable in view of the fact that we have had to make a similar compromise for programming languages. How interesting the spoken language can become in the future will depend on how realiably and precisely future programs can generate sound descriptions.

V-4-1. <u>Vocabulary and Grammar of the HAND-EYE-EAR System</u>

The vocabulary and the grammar chosen for the HAND-EYE-EAR program is as follows:

(See next page)

SYNTAX:

```
    <command>::=<command 1> | <command 2>

    <command 1>::= <order 1> EMPTY

    <order 1>::= RESCAN|STOP

    <command 2>::=PICK UP <argument list>

    <argument list>::= <every> <size indicator> EMPTY <position indicator>

    <every>::= EVERY|EMPTY

    <size indicator>::= EMPTY | <size> BLOCK

    <size>::= SMALL|MEDIUM|BIG|EMPTY

    <position indicator>::=<position 1> | <position 2> | EMPTY

    <position 1>::=<position'> SIDE| <position"> SIDE

    <position 2>::= <position'> <position"> CORNER|

                    <position"> <position'> CORNER

    <position'>::= LEFT|RIGHT|EMPTY

    <position">::= TOP|BOTTOM|EMPTY
```

SEMANTICS:

The meanings of some of the terminal symbols is obvious, but some others like RESCAN and EMPTY need explanation.

The command "rescan" is used to indicate that the scene might be disturbed and that the vision program should generate a new scene description.

The terminal symbol EMPTY means no speech utterance at all or sounds not recognized by the word recognizer. If any of the non-terminal symbols is finally reduced to EMPTY the middle value is assumed (i.e., if <size indicator>=EMPTY, a medium size block will be assumed).

Sentences like "pick up the small block standing at the top right corner", "rescan the scene", "pick up every block" are syntactically correct.

An overall flow chart of the sentence decoder is shown on Figure V-8. Given a command, the speech analysis program segments the whole utterance and generates a sound description. The syllable boundaries are located by utilizing the FRICS or BURST segments and the minimum aplitude segments. The scanner groups together one or several syllables to form a word candidate and calls the word recognizer to obtain the written representation of this part of the sentence. Feedback from the grammar takes the form of a list of allowed words, and only these words will be used for matching against the incoming utterance by the word recognizer.

V-4-2. Syllable Boundaries Determination

In this paragraph, a <u>syllable</u> will be defined to consist of <u>one vowel</u> segment and other adjoining segments. The boundaries between two syllables will be heuristically determined by the following rules:

-1. If there is no fricative segment between two vowels, the boundary between the two <u>syllables</u> defined by these vowels is at the segment of lowest amplitude. This lowest amplitude segment is, in turn, part of both syllables, i.e., when the <u>syllable</u> at its lefthand side is considered, this segment is part of it, and when the <u>syllable</u> at its righthand side is considered, it is also part of it.

-2. If there is a fricative segment between the two vowels, the boundary between the two <u>syllables</u> defined by these vowels is the boundary between the fricative segment and the preceding segment.

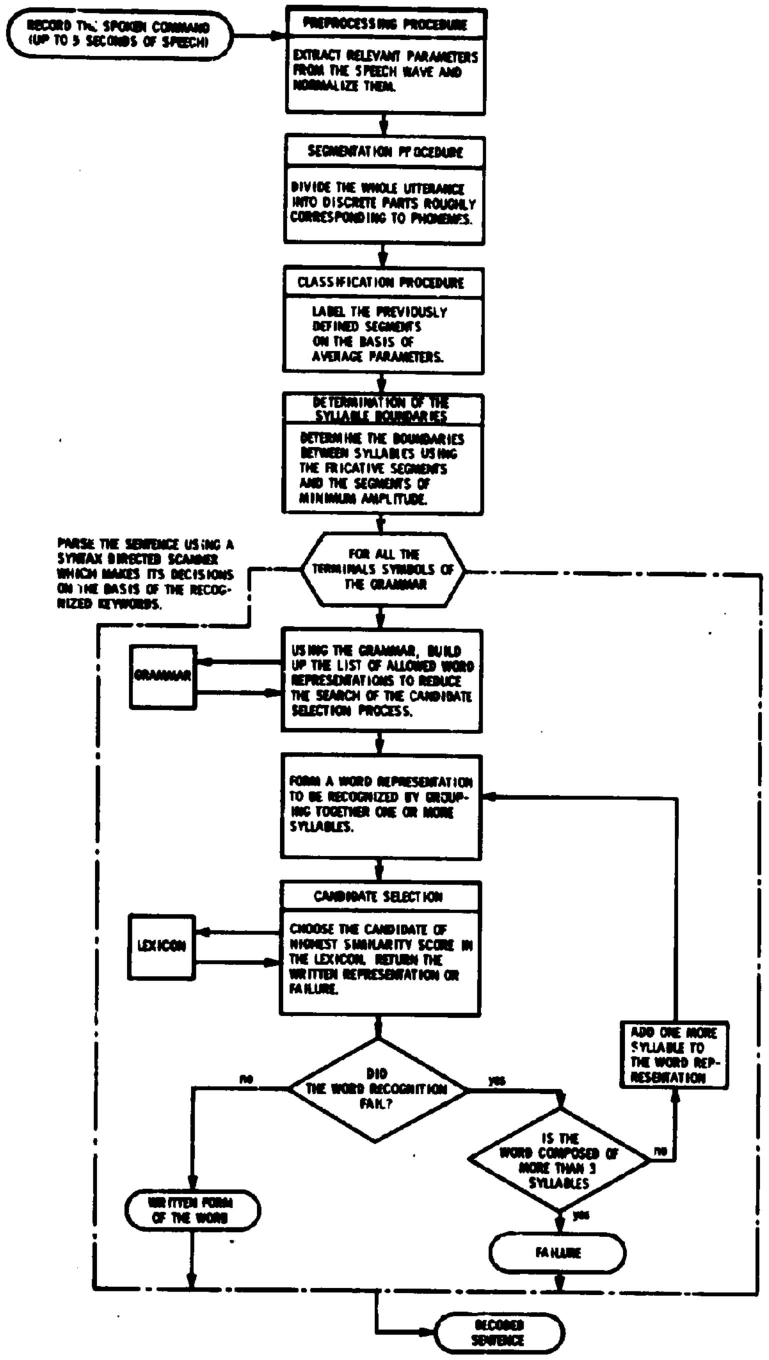

RECORD THE SPOKEN COMMAND (UP TO 5 SECONDS OF SPEECH)
PREPROCESSING PROCEDURE
EXTRACT RELEVANT PARAMETERS FROM THE SPEECH WAVE AND NORMALIZE THEM.
SEGMENTATION PROCEDURE
DIVIDE THE WHOLE UTTERANCE INTO DISCRETE PARTS ROUGHLY CORRESPONDING TO PHONEMES.
CLASSIFICATION PROCEDURE
LABEL THE PREVIOUSLY DEFINED SEGMENTS ON THE BASIS OF AVERAGE PARAMETERS.
DETERMINATION OF THE SYLLABLE BOUNDARIES
DETERMINE THE BOUNDARIES BETWEEN SYLLABLES USING THE FRICATIVE SEGMENTS AND THE SEGMENTS OF MINIMUM AMPLITUDE.
PARSE THE SENTENCE USING A SYNTAX DIRECTED SCANNER WHICH MAKES ITS DECISIONS ON THE BASIS OF THE RECOGNIZED KEYWORDS.
FOR ALL THE TERMINALS SYMBOLS OF THE GRAMMAR
GRAMMAR
USING THE GRAMMAR, BUILD UP THE LIST OF ALLOWED WORD REPRESENTATIONS TO REDUCE THE SEARCH OF THE CANDIDATE SELECTION PROCESS.
FORM A WORD REPRESENTATION TO BE RECOGNIZED BY GROUPING TOGETHER ONE OR MORE SYLLABLES.
CANDIDATE SELECTION
LEXICON
CHOOSE THE CANDIDATE OF HIGHEST SIMILARITY SCORE IN THE LEXICON. RETURN THE WRITTEN REPRESENTATION OR FAILURE.
ADD ONE MORE SYLLABLE TO THE WORD REPRESENTATION
no
DID THE WORD RECOGNITION FAIL?
yes
IS THE WORD COMPOSED OF MORE THAN 3 SYLLABLES
no
yes
WRITTEN FORM OF THE WORD
FAILURE
DECODED SENTENCE

Figure V-8. Overall flowchart of the sentence decoder .

According to these rules, the word RESCAN is composed of two syllables: .RE. and .SCAN.

These heuristics are only valid in our restricted vocabulary, mainly because no word ends with /S/ so that if an /S/ is found, it is certainly at the beginning of, or inside, a word and therefore at the beginning of a _syllable_ part of that word. Furthermore, these heuristics are relatively easy to implement since the fricative segments in the utterance representation are labeled FRICS or BURST and the local minima of amplitude are detected and marked by the segmentation procedure.

The process simply consists of scanning each utterance representation and marking the beginning and end of each _syllable_. A _syllable_ is then composed of all the segments between a beginning and an end marker.

Almost all the common _syllables_ defined in our vocabulary correspond to these "computed" syllables, words like PICK, UP, BLOCK, SMALL form _syllables_. The only exceptions are created by THE LEFT AND THE RIGHT. These are often grouped into one syllable by the program, since the power of the /ə/ in THE does not usually exceed the power of /l/ in LEFT or the power of /r/ in RIGHT. These difficulties can readily be overcome if we train the word recognizer with the representation of THE LEFT and THE RIGHT in place of LEFT and RIGHT.

V-4-3. Word Recognition

In this paragraph a _word_ is _defined_ as a group of one or more adjacent syllables. Furthermore, only the words included in the vocabulary (terminal symbols of the grammar) are considered. In all the the cases the recognition is attempted from short _words_, (one syllable) to long _words_ (up to three syllables); i.e., the program first attempts to recognize a monosyllabic word, if no acceptable representation is found which matches this word; a second adjacent syllable is added and recognition of the resulting two syllable word is attempted; if the program fails to give the written equivalent of this group of segments, it tries a three syllable word, if again no possible match is found, the program notes the failure and makes decisions on the basis of the sentence parts already recognized. The word boundaries are determined only when a word is recognized; i.e., the word begins at the beginning of its first syllable and ends at the end of its last syllable.

V-4-4. Sentence Recognition

The recognition of sentences beginning with STOP and RESCAN presents no major difficulties since only the first word of the sentence is to be recognized. However, the decoding of PICK UP commands is not a trivial problem. Assuming that PICK UP has been recognized as a two syllable word, two anchor points at which the program is supposed to find relevant information are provided in the sentence. The first of these is the end of the utterance which, according to the grammar, must contain the necessary position information. The second is the word BLOCK somewhere in the middle of the utterance. This word was chosen because it is an easy-to-recognize monosyllabic word. According to the grammar, this word must be preceded by the size information and/or the word EVERY.

The decoding of these commands is then done as follows:

-1. Recognize PICK UP

-2. Scan the sentence from the left to the right comparing all the syllables not starting with a fricative segment with the stored representations of BLOCK.

-3. When BLOCK is located, scan from right to left to recognize the adjectives preceding BLOCK.

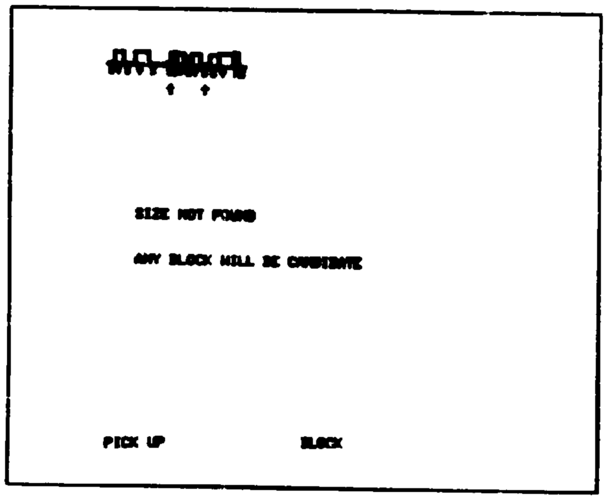
-4. Then recognize the last word of the sentence (SIDE or CORNER) and recognize the preceding adjectives.

At each step, feedback from the grammar takes the form of a list of allowed words; i.e., when the program is reducing <size> to a terminal symbol, the only word representations considered by the candidate selection process are those of SMALL, MEDIUM, BIG and EVERY, thus reducing the search time by a factor of 3 and eliminating most of the ambiguities.

Figures V-13 to V-21 illustrate the decoding of the sentence "PICK UP EVERY SMALL BLOCK STARTING AT THE BOTTOM RIGHT CORNER". To exhibit the details of the "thinking" process, intermediate displays are shown. In normal operation, only a simplified representation of the utterance (at the top of the screen) and the results of the recognition (at the bottom of the screen) are displayed. The 90 seconds mentioned for the pass is the time taken with these intermediate displays turned off.

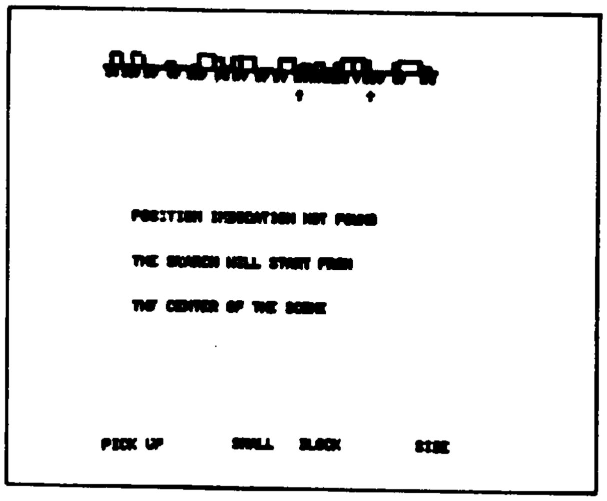
Figures V-9 and V-10 illustrates some decisions taken by the program which could not reduce a non-terminal symbol. In one case, the sentence was PICK UP ANY BLOCK. ANY was not recognized because it is not in the vocabulary. The program noted the failure and proceeded. In the second case, the word RIGHT was not recognized. The two ARROWS at the top of the screen delimit the three-syllable word considered in the last attempt to reduce the non-terminal <position>. In normal operation, the program proceeds taking the decision displayed on the screen.

In hybrid mode, (i.e., decoding and training) it will stop at the end of the decoding process and allow the experimenter to train it with new representations of the words not recognized.



```
SIZE NOT FOUND

ANY BLOCK WILL BE CANDIDATE


PICK UP              BLOCK
```

Figure V-9. The program did not recognize the adjective before BLOCK . (this word was ANY not in the vocabulary).



```
POSITION INDICATION NOT FOUND

THE SEARCH WILL START FROM

THE CENTER OF THE SCENE


PICK UP        SMALL   BLOCK        SIZE
```

Figure V-10. The program did not recognize the positional adjective RIGHT . The two arrows under the utterance description limit the three syllables word considered in the last attempt to reduce the non-terminal ⟨position⟩ .

V-5. RESULTS AND CONCLUSIONS

V-5-1. Results of the Sentence Analyzer

We illustrate the results obtained with the sentence-analyzer by a series of pictures (Figures V-11 to V-25) and statistical results performed with recordings made by several speakers (Figures V-26 to V-28).

The pictures are a direct photograph of the CRT display taken during a program run. They clearly illustrate the different phases of the process. The detailed comments written by the side of each picture relate it to the operations executed by the computer at the same moment. As we previously stated, most of the sentence analysis displays presented in these Figures are intermediate displays which are not shown in the normal operation of the program. Only the top portion which represents the utterance description and the bottom portion where the result of the recognition process appears are ordinarily displayed. The statistical results were obtained using sentences recorded by four speakers. Figure V-27 gives the sentences used to build the lexicon of the word recognizer and the sentences used to test the sentence analyzer. The voices of the first two speakers were used to train the "ear" routine. To do so they had to speak the training set five times and the test set once, while the next two speakers spoke only the test set. Figure V-28 and V-29 show the results obtained. The noise was quite high (S/N ratio $\approx$15db), since the recording was made in the machine room. Moreover the speakers were talking at normal speed, as shown by the timings given for each sentence. These timings (given in seconds) include (1) the duration of the spoken command, (2) the time taken by the segmentation process to segment the utterance and classify the resulting segments into phoneme

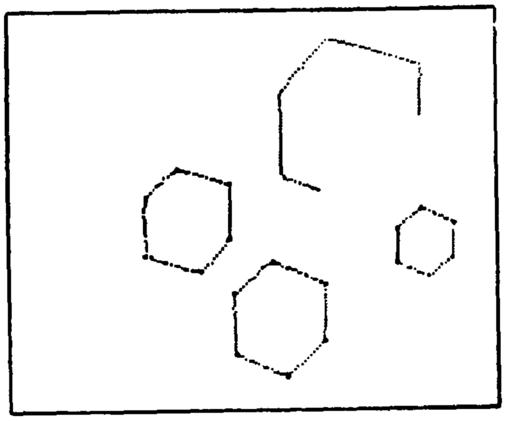


Figure V-11. The "EYE" routines are locating cubes in the field of view of the camera . Only the outlines of the objects are seen . They are tested in various ways to determine whether or not a detected object is a cube . Three cubes have already been found ; the "edge follower" is tracing the edges of a fourth one .

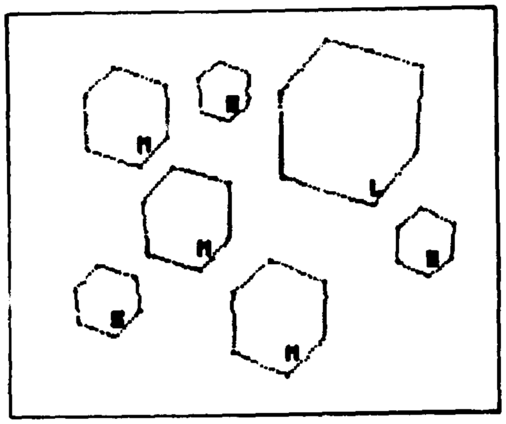
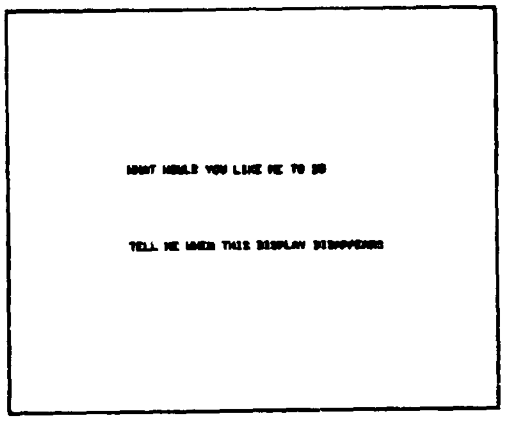


Figure V-12. The entire scene has been determined, the scene description has been generated, and sizes have been attributed to each cube . the "EYE" routines terminate here, the control will now be shared by the "ARM" routines and the "EAR" routines until it will be necessary to generate a new scene description .


WHAT WOULD YOU LIKE ME TO DO

TELL ME WHEN THIS DISPLAY DISAPPEARS

Figure V-13. The "EAR" routines are requesting a command from the experimenter . He can start speaking as soon as the display disappears, but the microphone service routine will wait for him for 30 seconds before complaining . The recorded speech utterance has a duration varying from 0.3 second to 5 seconds, the sound input being stopped when a long silence is detected .

In the present case, the uttered sentence was :"PICK UP EVERY SMALL BLOCK STARTING AT THE BOTTOM RIGHT CORNER" .

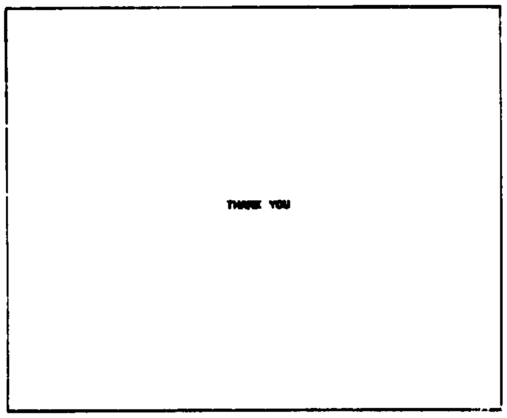


Figure V-14. The spoken command has been recorded correctly and the "EAR" program is analyzing it . Segmentation and classification into phoneme groups are being performed on the entire utterance, thus creating a sound description . This process takes from 0.5 to 6 seconds depending on the duration of the uttered sentence .
(about 4 seconds in this case)

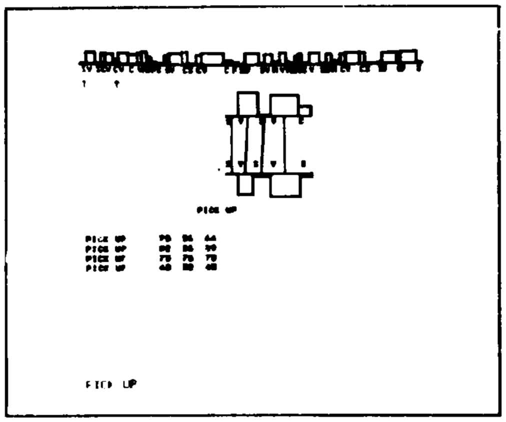


Figure V-15. The segmented utterance appears at the top of the CRT screen, (more precisely, a display of the parameter A1, defined in the chapter II). The two arrows indicate the portion of the utterance description currently being analyzed . Several stored representations of PICK UP were found to match this part of the sentence . On the basis of similarity values, PICK UP was accepted as the first word of the command .

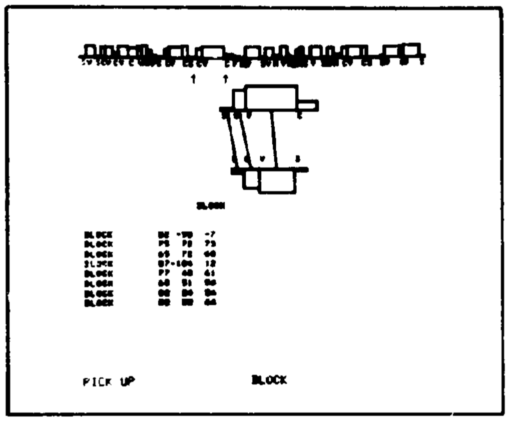


Figure V-16. The sentence analyzer scans the utterance description to find BLOCK. A match has been found for the group of segments indicated by the arrows . The program was trained with several different speakers, which explains the low similarity scores obtained by some of the stored representations .

Figure V-17. The program backtraces from BLOCK to find the qualifying adjective . An acceptable match has been found for one syllable before BLOCK . Two stored representations of SMALL have matched this part of the utterance .

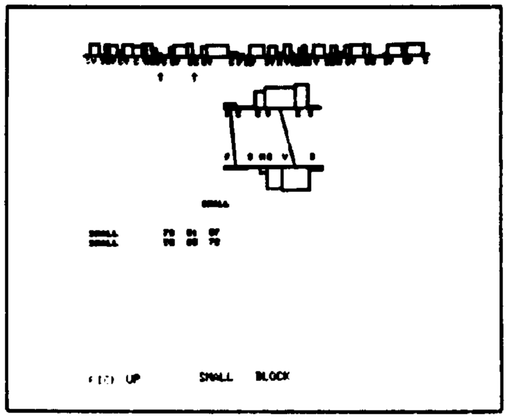


Figure V-18. The program backtraces again from the beginning of SMALL, looking for EVERY, several stored representations of EVERY have matched the part of the command delimited by the arrows .

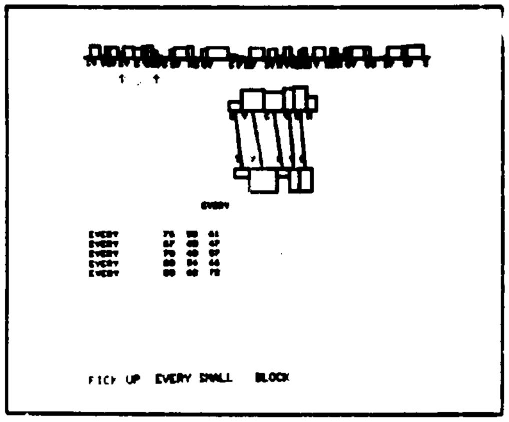
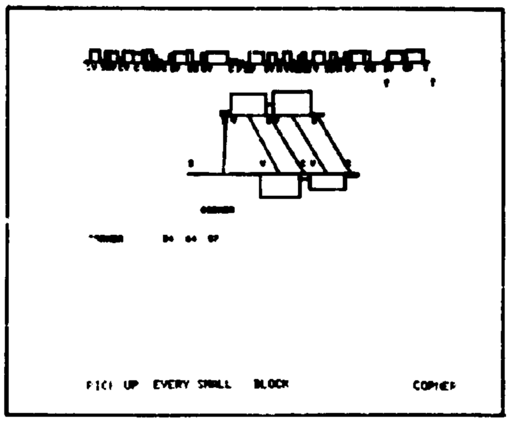


Figure V-19. The sentence is now examined from the end . According to the grammar, the position information is contained in this part of the sentence . Only two words are possible here : SIDE or CORNER . In this case, CORNER is recognized .



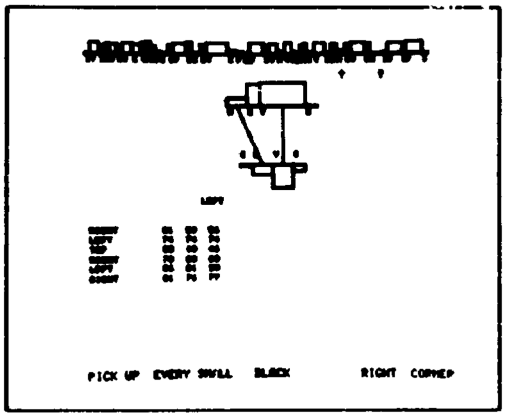


Figure V-20. The program then backtraces from the beginning of CORNER to recognise the adjectives before it (CORNER expects two positional adjectives). Several syntactically valid stored representations were accepted. On the basis of the highest similarity value, RIGHT was chosen .

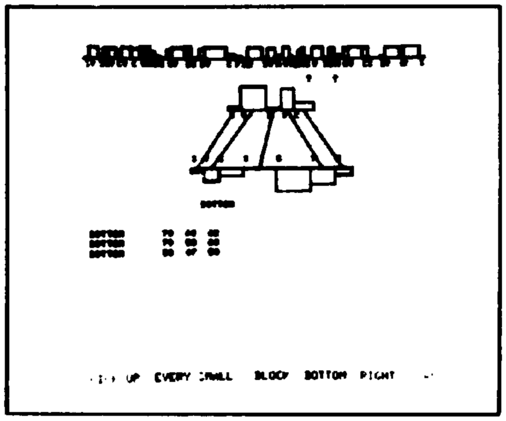


Figure V-21. Backtracing from RIGHT, the program recognises BOTTOM . The sentence is now entirely decoded ; a command word containing the useful information is built and passed to the control program which will now resume this task by activating the "ARM" routines .

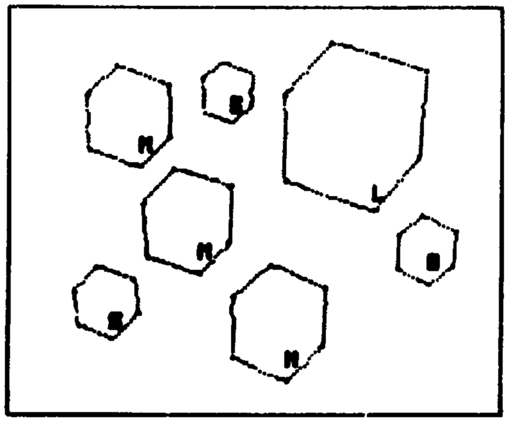


Figure V-22. The block chosen by the control program (SMALL block nearest to the BOTTOM RIGHT CORNER) is marked on the CRT screen, and the coordinates of the corners of this block are passed to the "ARM" routines . The arm displacements will be computed, and the arm pick up and stack the block .

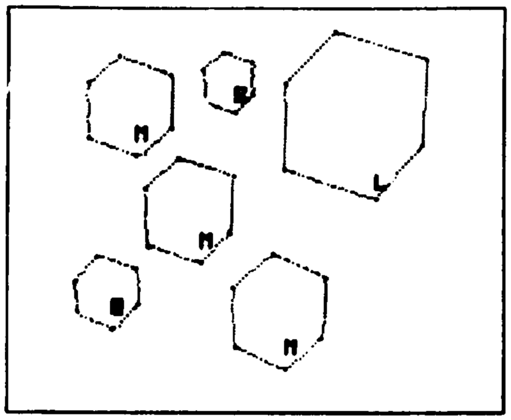


Figure V-23. The block previously taken away disappears from the screen . As the system was ordered to pick up all the small blocks, a second block is marked as the next to be picked up .

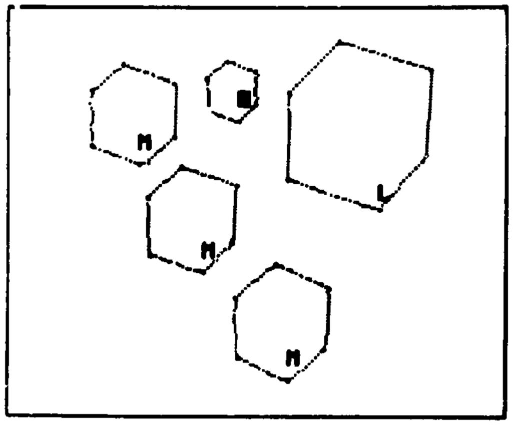


Figure V-24. The last remaining block is marked and will readily disappear .

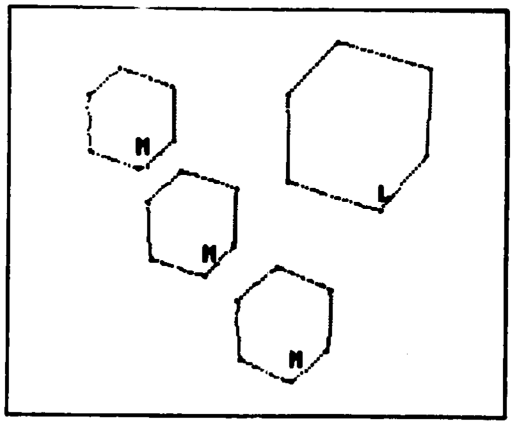


Figure V-25. The command has been executed, and the final scene description appears on the screen . In few seconds the program will request a new command from the experimenter .

The time taken to perform this pass through the entire system was about 90 seconds, most of it spent in the arm displacements .
(this does not include the time taken to produce the displays corresponding to the "EAR" routines).

```
TRAINING SET :

RESCAN
STOP
PICK UP THE SMALL BLOCK STANDING ON THE RIGHT SIDE
PICK UP THE BIG BLOCK STANDING ON THE LEFT SIDE
PICK UP THE MEDIUM BLOCK STANDING ON THE TOP SIDE
STANDING ON THE BOTTOM SIDE
STANDING ON THE TOP LEFT CORNER
STANDING ON THE BOTTOM LEFT CORNER
STANDING ON THE TOP RIGHT CORNER
STANDING ON THE BOTTOM RIGHT CORNER
PICK UP EVERY SMALL BLOCK
PICK UP EVERY MEDIUM BLOCK
PICK UP EVERY BIG BLOCK
YES


TEST SET :

RESCAN
PICK UP THE SMALL BLOCK STANDING ON THE RIGHT SIDE
PICK UP THE BIG BLOCK STANDING ON THE TOP RIGHT CORNER
PICK UP THE MEDIUM BLOCK STANDING ON THE BOTTOM LEFT CORNER
PICK UP EVERY SMALL BLOCK STARTING AT THE RIGHT SIDE
PICK UP ANY BLOCK
PICK UP THE SMALL BLOCK STANDING ON THE TOP LEFT CORNER
RESCAN
PICK UP THE BIG BLOCK STANDING ON THE BOTTOM SIDE
PICK UP THE BLOCK ON THE TOP LEFT CORNER
PICK UP EVERY BLOCK
PICK UP EVERY MEDIUM BLOCK STARTING AT THE BOTTOM RIGHT CORNER
RESCAN
PICK UP EVERY BIG BLOCK STARTING AT THE TOP LEFT CORNER
PICK UP THE MEDIUM BLOCK ON THE TOP SIDE
PICK UP THE SMALL BLOCK ON THE LEFT SIDE
PICK UP THE MEDIUM BLOCK ON THE CENTER
PICK UP THE BIG BLOCK STANDING ON THE BOTTOM LEFT CORNER
PICK UP THE BLOCK ON THE RIGHT TOP CORNER
RESCAN
PICK UP THE SMALL BLOCK STANDING ON THE LEFT BOTTOM CORNER
PICK UP THE MEDIUM BLOCK STANDING ON THE RIGHT SIDE
STOP
YES
```

Figure V-26. Training set and test set used to obtain statistical results on the sentence analyzer .

UTTERED SENTENCES	UTTERANCE DURATION	SEGMEN. TIME	DECODING TIME
RESCAN	0,990 S	0,950 S	0,683 S
PICK UP SMALL BLOCK RIGHT SIDE	2,590 S	3,934 S	4,450 S
PICK UP BIG BLOCK TOP RIGHT CORNER	2,780 S	4,166 S	4.050 S
PICK UP BIG BLOCK BOTTOM LEFT CORNER	3,020 S	5,700 S	5,984 S
PICK UP EVERY SMALL BLOCK RIGHT SIDE	3,010 S	5,417 S	5,016 S
PICK UP EVERY BLOCK	0,030 S	1,033 S	2,433 S
PICK UP SMALL BLOCK TOP LEFT CORNER	3,040 S	5,600 S	1,934 S
RESCAN	1,000 S	0,867 S	3,017 S
PICK UP BIG BLOCK BOTTOM SIDE	2,750 S	3,500 S	5,400 S
PICK UP BLOCK TOP LEFT CORNER	2,100 S	3,684 S	3,483 S
PICK UP EVERY BLOCK	1,150 S	1,116 S	1,617 S
PICK UP EVERY MEDIUM BLOCK BOTTOM RIGHT CORNER	3,310 S	7,000 S	6,984 S
RESCAN	1,040 S	0,934 S	2,450 S
PICK UP EVERY BIG BLOCK TOP LEFT CORNER	3,340 S	6,183 S	3,700 S
PICK UP MEDIUM BLOCK TOP SIDE	2,180 S	3,700 S	8,133 S
PICK UP SMALL BLOCK LEFT SIDE	2,360 S	3,284 S	2,750 S
PICK UP MEDIUM BLOCK	2,010 S	4,250 S	5,700 S
PICK UP BIG BLOCK BOTTOM LEFT CORNER	2,990 S	4,150 S	4.133 S
PICK UP BIG BLOCK RIGHT TOP CORNER	2,360 S	3,250 S	3,467 S
RESCAN	1,020 S	0,933 S	0,917 S
PICK UP SMALL BLOCK LEFT BOTTOM CORNER	3,260 S	5,450 S	4,533 S
PICK UP MEDIUM BLOCK LEFT SIDE	2,650 S	4,250 S	5.816 S
STOP	0,690 S	0,383 S	0,167 S
YES	0,690 S	0,533 S	0,183 S

RESCAN	0,780 S	0,600 S	0,484 S
PICK UP SMALL BLOCK RIGHT SIDE	2,740 S	4,367 S	3,617 S
PICK UP BIG BLOCK TOP RIGHT CORNER	3,050 S	5,733 S	4,900 S
PICK UP MEDIUM BLOCK BOTTOM RIGHT CORNER	3,130 S	5,600 S	5,134 S
PICK UP ***** SMALL BLOCK RIGHT SIDE	2,880 S	5,167 S	2,733 S
PICK UP BLOCK	1,320 S	2,083 S	2,800 S
PICK UP SMALL BLOCK TOP LEFT CORNER	3,290 S	5,733 S	3,800 S
RESCAN	0,880 S	0,666 S	0,600 S
PICK UP BIG BLOCK BOTTOM SIDE	2,620 S	4,416 S	3,784 S
PICK UP BIG BLOCK TOP LEFT CORNER	2,300 S	3,350 S	5,233 S
PICK UP EVERY BLOCK	1,160 S	1,733 S	2,017 S
PICK UP EVERY MEDIUM BLOCK BOTTOM RIGHT CORNER	3,050 S	8,200 S	2,516 S
RESCAN	0,840 S	0,867 S	0,656 S
PICK UP ***** BIG BLOCK TOP LEFT CORNER	2,980 S	4,983 S	3,834 S
PICK UP MEDIUM BLOCK TOP SIDE	2,890 S	5,534 S	5,300 S
PICK UP SMALL BLOCK LEFT SIDE	2,290 S	3,033 S	2,834 S
PICK UP MEDIUM BLOCK	2,110 S	4,100 S	6,633 S
PICK UP BIG BLOCK BOTTOM LEFT CORNER	2,890 S	4,350 S	4.767 S
PICK UP BLOCK RIGHT TOP CORNER	2,320 S	3,350 S	3,467 S
RESCAN	0,810 S	0,600 S	0,300 S
PICK UP SMALL BLOCK LEFT BOTTOM CORNER	2,850 S	5,966 S	10,284 S
PICK UP MEDIUM BLOCK RIGHT SIDE	2,750 S	7,334 S	3,800 S
STOP	0,740 S	0,467 S	0,217 S
YES	0,720 S	0,584 S	0,233 S

Figure V-27. Sentence analysis results for 2 speakers whose voices were used to train the computer .

```
        UTTERED SENTENCES                                 UTTERANCE  SEGMEN.   DECODING
                                                          DURATION    TIME       TIME

RESCAN                                                    0,960 S   1,100 S   0,784 S
PICK UP SMALL BLOCK RIGHT SIDE                            2,820 S   3,583 S   4,750 S
PICK UP BIG BLOCK TOP RIGHT CORNER                        2,920 S   4,550 S   3,900 S
PICK UP EVERY BIG BLOCK BOTTOM LEFT CORNER                2,960 S   6,283 S   5,684 S
PICK UP EVERY SMALL BLOCK RIGHT SIDE                      2,950 S   3,483 S   3,183 S
PICK UP MEDIUM BLOCK                                      1,330 S   1,433 S   2,767 S
PICK UP SMALL BLOCK TOP LEFT CORNER                       2,870 S   3,534 S   4,916 S
******                                                    0,700 S   0,817 S   0,850 S
PICK UP BIG BLOCK RIGHT SIDE                              2,710 S   4,916 S   3,500 S
PICK UP SMALL BLOCK TOP LEFT CORNER                       2,000 S   2,950 S   2,517 S
PICK UP ****** BLOCK                                      1,070 S   1,366 S   2,484 S
PICK UP EVERY MEDIUM BLOCK BOTTOM RIGHT CORNER            3,090 S   5,934 S   4,383 S
RESCAN                                                    0,900 S   0,884 S   0,683 S
PICK UP EVERY BIG BLOCK *** **** ******                   2,700 S   5,250 S   6,850 S
PICK UP BIG BLOCK TOP SIDE                                2,360 S   2,650 S   3,850 S
PICK UP SMALL BLOCK LEFT SIDE                             2,210 S   2,900 S   3,000 S
PICK UP MEDIUM BLOCK                                      1,890 S   2,166 S   4,767 S
PICK UP BIG BLOCK BOTTOM LEFT CORNER                      2,810 S   3,684 S   4,783 S
PICK UP BLOCK RIGHT TOP CORNER                            1,880 S   2,650 S   3,650 S
RESCAN                                                    0,860 S   0,816 S   0,750 S
PICK UP SMALL BLOCK RIGHT BOTTOM CORNER                   2,980 S   5,250 S   5,533 S
PICK UP EVERY BIG BLOCK RIGHT SIDE                        2,700 S   3,383 S   5,350 S
STOP                                                      0,720 S   0,550 S   0,217 S
YES                                                       0,460 S   0,483 S   0,201 S


RESCAN                                                    0,960 S   0,666 S   1,050 S
PICK UP SMALL BLOCK LEFT SIDE                             2,570 S   4,900 S   3,900 S
PICK UP BIG BLOCK TOP LEFT CORNER                         2,860 S   3,784 S   4,033 S
PICK UP MEDIUM BLOCK BOTTOM LEFT CORNER                   2,670 S   5,050 S   5,883 S
PICK UP EVERY SMALL BLOCK RIGHT SIDE                      2,750 S   5,300 S   3,707 S
PICK UP EVERY BLOCK                                       1.240 S   1,367 S   2,933 S
PICK UP ***** BLOCK TOP LEFT CORNER                       2,730 S   4,250 S   3,500 S
RESCAN                                                    0,980 S   0,783 S   1,050 S
PICK UP BIG BLOCK BOTTOM SIDE                             2,630 S   3,750 S   4,166 S
PICK UP BLOCK TOP RIGHT CORNER                            2,020 S   3,984 S   3,633 S
PICK UP EVERY BLOCK                                       1,350 S   1,350 S   2,267 S
PICK UP ***** BIG BLOCK BOTTOM RIGHT CORNER               3,330 S   6,183 S   7,250 S
RESCAN                                                    0,920 S   0,767 S   0,566 S
PICK UP EVERY BIG BLOCK TOP LEFT CORNER                   2,940 S   6,433 S   3,817 S
PICK UP MEDIUM BLOCK TOP SIDE                             2,120 S   2,966 S   3,100 S
PICK UP SMALL BLOCK LEFT SIDE                             2,200 S   2,717 S   3,166 S
PICK UP MEDIUM BLOCK                                      2,000 S   2,633 S   6,733 S
PICK UP BIG BLOCK BOTTOM RIGHT CORNER                     2,820 S   5,316 S   4,067 S
PICK UP BLOCK RIGHT TOP CORNER                            2,300 S   3,734 S   4,350 S
RESCAN                                                    0,900 S   0,717 S   0,917 S
PICK UP SMALL BLOCK RIGHT BOTTOM CORNER                   3,110 S   4,600 S   5,100 S
PICK UP MEDIUM BLOCK RIGHT SIDE                           2,600 S   4,316 S   5,350 S
STOP                                                      0,740 S   0,516 S   0,200 S
YES                                                       0,470 S   0,400 S   0,193 S
```

Figure V-28. Sentence analysis results for 2 speakers whose voices were unknown to the computer .

groups, and (3) the time taken to decode the sentence. The character strings at the left of the Figures are the sentence analyzer output. They were slightly modified to render them more comprehensible; i.e., each non-recognized word was replaced by a series of "*" and each incorrectly recognized word was underlined.

Although the percentage of correctly recognized sentences is not exceptionally high 85.5% correctly recognized for the two speakers utilized to train the program, and 66.6% for the other two, the number of correctly recognized words is more impressive (96% and 90% respectively). Furthermore, the program is capable of detecting some of its own failures, that is, those which occur when one of the nonterminal symbols cannot be reduced, and for these cases it can be modified to request the experimenter to repeat the sentence. Likewise, to insure a correct execution of a given task, it can be provided with execute and reject commands; i.e., if a sentence has been correctly decoded, the experimenter requests its execution by saying "execute", but if it has been incorrectly decoded, he says "reject" and repeats the same command again.

An interesting point of the exhibited results is the program's failure to recognize the 14th command of the third speaker (Figure IV-28). The program did not recognize <u>corner</u> at the end of the sentence and therefore could not recognize the two preceding adjectives since it could not determine the boundaries of these two words. This illustrates the well known problem of error recovery of syntax-directed compilers. Of course, in our simple case, an ad hoc technique could have been devised to recover from such a blunder. But in a more general case, the problem of error recovery, when the program fails to detect the boundaries of one word is one of the most difficult sub-problems in the general problem of connected speech recognition.

Although the recognition times presented are already short (an average of 9 seconds was taken for the recognition of the longest sentences), one could effectively eliminate the time required for segmentation by using the real-time segmentation program described in the next chapter (desk-calculator) which divides the utterance into discrete parts while the experimenter is talking. Likewise, the replacement of the actual grammar by a stricter one (without the terminal symbol EMPTY added to allow the decoding of quasi natural english sentence) in which a left-to-right parsing algorithm can be applied could slightly reduce the decoding time. Under these conditions, a 4 second long sentence could be recognized in 3 to 4 seconds, that is about 3 times faster than a trained typist can type it.

V-5-2. Conclusions

It will be probably a long time before a computer can equal the perception and dexterity of a human being. This will require not only advances in the area of computer architecture and in the quality of the external devices, but also a better understanding of perceptual and motor processes.

Even the limited progress achieved so far can result in computer hand-eye-ear systems that are better suited for some purposes than human beings. For example, they may see things and hear sounds that a person cannot and they can work in areas prohibited to humans (nuclear energy laboratories for example). They may be faster, stronger, more economical or more expandable than men.

The fact that a computer may not be able to "understand" all the things it can see or carry on fluent conversation should not be a cause for extra concern. Let us consider the case of programming languages. Although we have not been able to communicate with computers in a natural language, a great deal is achieved using structured ad hoc languages like Fortran or Algol. We believe that this will be the case with visual and voice input to the computers or with computer control of manipulators.

We foresee several practical applications that can profitably use the techniques described in this chapter. One that is most often mentioned is the possible bandwith reduction in picture and speech transmission systems. Computer controlled carts which can navigate themselves, automated factories, where computer controlled manipulators with visual feedback can handle many situations which cannot be presently handled by fixed sequence manipulators, voice controlled data retrieval systems are within the range of the present state of the art.

Chapter VI

DESCAL

A VOICE CONTROLLED DESK-CALCULATOR

VI-1. INTRODUCTION

The speech analysis subsystem of the HAND-EYE-EAR system (i.e., the EAR part) does not fully utilize the power of the word recognizer which is available to it. This word recognizer, which is able to accurately differentiate between the words of large vocabularies, is only required in this case to decode sentences constructed from a 16 word vocabulary. Syntactical contraints further reduce the list of candidates to at most 4 entries each time a recognition is attempted. In this especially simple case, a much simpler ad-hoc algorithm could have been devised to resolve any remaining ambiguities. In this chapter, we shall describe a more exacting application of the speech recognition system: a voice controlled desk-calculator DESCAL.

The sentences are structured by a linear grammer acting on a vocabulary (i.e., set of terminal symbols of the grammar) composed of 46 words (or terms like RUB OUT). While processing spoken commands, the grammar is utilized twice, namely:

-1. To reduce the search and eliminate phonetic ambiguities by look-ahead during the decoding of the sentence.

-2. To direct the left-to-right parsing algorithm while executing the command.

Desirable characteristics of a voice controlled desk-calculator are:

-1. The uttered words should automatically appear as you speak,

BLANK PAGE

without waiting for the completion of the whole sentence (unlike the HAND-EYE-EAR system in which the command was given all at the same time in a connected speech utterance).

-2. Should the computer make a mistake at any given point, it should be possible to alter by voice command the word which was erroneously recognized. This requirement is in contradiction with the preceding one if the computer system utilized is unable to recognize words much faster than a person can utter them. If an error is noticed after several other words have been uttered, it becomes necessary to "erase" the symbols following the error before attempting to fix it and to repeat all of them afterwards.

-3. The language used for controlling the calculator should be natural so that users will concentrate on their computation and not on the form of the statements.

To satisfy the first requirement we had to reprogram the utterance description generating process and the utterance recognition in order to increase their speed. Since the previous chapters have shown that the speech analysis processes (segmentation and classification into phoneme-like-groups) were almost performed in real-time, we decided to execute them while the experimenter is talking. To perform this transformation, the corresponding programs were simplified, coded in machine language and incorporated into the hardware preprocessor service routine, the only program capable of processing the microphone input buffer while the input operation is executed (in a parallel fashion).

Because of system limitations which are mainly due to the slowness of the available computer (1.0 μs or 2.0 μs memory cycle time, 2.5 μs for a fixed point addition), it was impossible to recognize a given terminal symbol in real-time, even with the search space reduction resulting from the syntactical constraints. The amount of computation time necessary to recognize a word is typically 0.5 to 1 second, but since this program is run under the time-sharing system, this basic time can be largely increased if there are several users working on the machine and if the program is swapped in and out of memory.

To overcome these limitations, we could have implemented a speech-input similar to the HAND-EYE-EAR system speech-input (Chapter V); that is, a full desk-calculator statement is read-in through the microphone and then analyzed. In this hypothesis, the decoding of the statement would have been done by first determining the syllable boundaries and then by recognizing each word in the same manner the EAR part does. Of course, the word recognizer would have to be trained with actual words taken from few typical spoken sentences. If this had been done, and if a recognition error occurred, the only solution would be to reject the whole statement and to repeat it, which is annoying if only one of the symbols is erroneous.

To satisfy the second requirement, we decided to use a word at a time input because it eliminates the erasing problem and it makes the calculator independent of the speed of the machine. At present, the experimenter has to wait one second before saying the next word. Of course, this is not fully satisfactory, but if a faster computer was available (like CDC-6600 or IBM 360/91), a short silence between words would be sufficient to perform the recognition of the previously uttered word. In fact, since the recognition is performed in 0.5 to 1 second, a 5 times faster computer would be sufficient to implement a real-time desk-calculator (such a computer is already common since an IBM 360/91 is 30 times faster than the PDP-10 we used).

The language described in the next section attempts to satisfy the third requirement.

The chapter will present the structure of DESCAL along with the modifications effected on the recognition system in order to increase its speed which is one of the main goals of this last application. The chapter can be summarized as follows:

-1. Presentation of a language adapted to the specific voice controlled desk-calculator task.

-2. Description of efficient speech analysis and recognition procedures capable of decoding and executing syntactical correct commands in real-time (or almost real-time. The words are read in one at a time and are recognized in 0.5 second to 1 second).

-3. Description of the results obtained and conclusions. A set of photographs exhibiting the execution of a simple computation as displayed on the CRT console used as physical support of DESCAL terminates the chapter.

The collection of subroutines which compose DESCAL occupies 16K words of memory: 7 K words of buffer and lexicon storage area and 9K of machine code.

VI-2. THE LANGUAGE OF DESCAL

The usefulness of any desk-calculator will depend on its convenience. It has to be at least as good as usual desk-calculators as far as the generality of the operations which can be performed on such a device is concerned. This was achieved by implementing a 10 register machine provided with 7 basic operations and 10 arithmetic functions. All these basic components are easily modifiable and expandable by a few modifications performed on the tables which implement the grammar.

Languages for speaking to machines would in general have a different structure than their written counterparts. Desirable characteristics of a spoken command language are:

-1. The sentences of the language have to be easy to speak and natural sounding. For example, sentences like: "ALPHA EQUALS BETA PLUS GAMA SEMI-COLON" are unsatisfactory for speaking.

-2. The terminal symbols (i.e., words of the vocabulary) have to be chosen so that the phonetic ambiguities and word boundary ambiguities are minimized.

The language described in this section attempts to satisfy these requirements. Whenever these requirements are in contradiction, the convenience has precedence over the choice of the terminal symbols (e.g., the digits were left in their normal spoken forms even though ambiguities may arise from the acoustic similarity between FIVE and NINE).

For reason of convenience, and to allow interesting arithmetic functions, all the numbers of DESCAL are real. Of course, integer numbers are permitted in input, but they are immediately converted to their real representation.

VI-2-1. <u>Syntax of the Language</u>

The language of DESCAL is defined by the linear grammar which is represented by the following set of Backus-Naur productions:

```
<command>   ::=  <statement> EXECUTE | <statement> ERASE

<statement>   ::=  <store statement> | <compute statement> |
                   <arithmetic statement>

<store statement>  ::=  STORE <real number> INTO  <register>
<real number>      ::=  <fnumber> |<fnumber> @ <snumber>
<fnumber>          ::=  <snumber> |<snumber> . <number>
<snumber>          ::=  <number>| - <number>
<number>           ::=  <digit> |<digit> <number>
<digit>            ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<compute statement>  ::=  COMPUTE <function> OF <register>
<function>           ::=  SQUAROOT | SINE | COSINE | TANGENT | LOGARITHM |
                          NATURALOG | EXPONENT | ARCSINE | ARCOSINE | ARCTANGENT

<arithmetic statement>  ::=  <operator> <register> <preposition>
                             <register>
<operator>  ::=  MOVE | ADD | SUBTRACT | MULTIPLY | DIVIDE
<preposition> ::=  BY | TO | FROM
<register>  ::=  ZULU | STIK | STAR | KILO | OSCAR |
                 PAPA | WEST | SISTER | WHISKY | FOX
```

VI-2-2. Semantics of the Language

EXECUTE	When a desk-calculator statement has been correctly recognized by the sentence recognizer, its execution is requested by saying "EXECUTE". Two different actions take place at that point - The command is executed by an interpretive program. - The representations of the utterances marginally recognized are introduced into the lexicon, so that the system is continuously improving its performances.
ERASE	If an error has occured, or if the experimenter changes his mind, he rejects the entire statement by saying "ERASE". The statement is erased and ignored (not executed).
RUB OUT	(not in the grammar) This terminal symbol is allowed at any moment in a statement. It provokes the deletion of the previously uttered terminal symbol and places the syntax-directed sentence analyzer in the corresponding state.

To explain the other statements we shall use a convenient algol-like notation.

STORE real number INTO register	⇒	register ← realnumber
COMPUTE function OF register	⇒	register ← function (register)
MULTIPLY register1 BY register2	⇒	register1 ← register1 x register2
DIVIDE register1 BY register 2	⇒	register1 ← register1 / register2
ADD register TO register2	⇒	register2 ← register1 + register2
MOVE register1 TO register2	⇒	register2 ← register
SUBTRACT register1 FROM register2	⇒	register2 ← register2 - register1

VI-2-3. Implementation of the Syntax-Directed Statement Analyzer

Syntax-directed recognition is implemented by means of tables of syntactically correct terminal symbols at the current stage of the analysis. A flowchart of the sentence analyzer is given on Figure VI-1. Each box RECORD and RECOGNIZE ....contains, between parenthesis, the list of syntactically correct terminal symbols. Only these will be considered as possible candidates by the recognition process. In the implementation, a table of allowed terminal symbols is attached to each of these boxes. Any probable candidate, which was selected by the lexicon search procedures (Chapter IV), and whose print-name is not present in the current table of allowed symbols is eliminated from the list of probable candidates. This process considerably reduces the search space of the word recognition process and eliminates all ambiguities mainly because the vocabulary is carefully chosen. When a symbol is recognized, its print-name is placed in a storage area which contains the print-names of the statement parts already recognized. We will refer to this storage area which is used by the statement interpreter during the execution of the command as the written-form of the statement. Then, the syntax-directed analyzer assumes the next state (i.e., set up for the next table of allowed symbols) depending on the name of the last recognized terminal symbol For example, if STORE has been recognized as operator, the program gets ready to recognize a number and will stay in this state until INTO is recognized, then it will get ready to recognize a register and so on.......

As long as a terminal symbol is not recognized, the process stays in the same state (i.e., utilizes the same table of allowed symbols) and requests new utterances of this word by displaying question-marks.

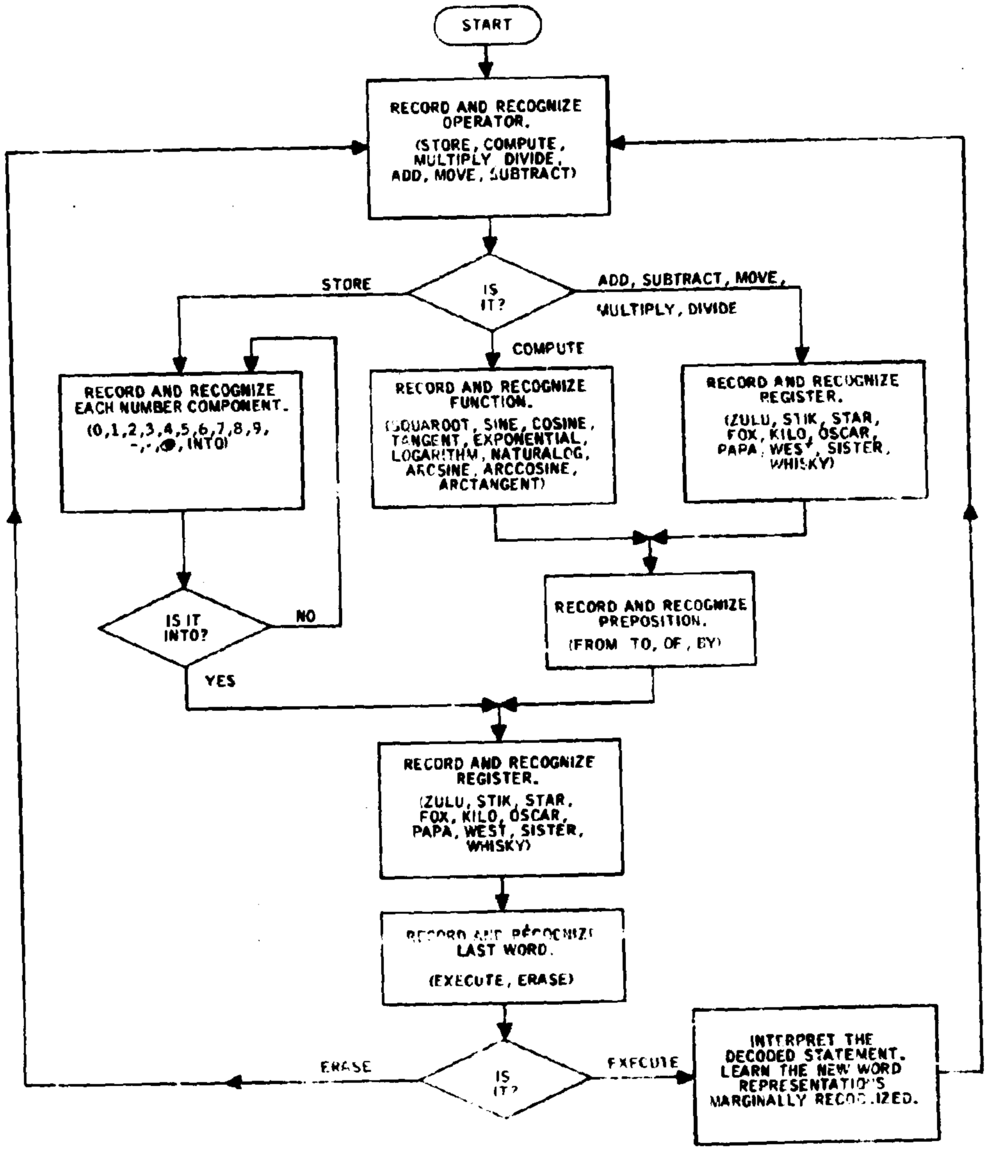

START
RECORD AND RECOGNIZE OPERATOR. (STORE, COMPUTE, MULTIPLY, DIVIDE, ADD, MOVE, SUBTRACT)
IS IT?
STORE
COMPUTE
ADD, SUBTRACT, MOVE, MULTIPLY, DIVIDE
RECORD AND RECOGNIZE EACH NUMBER COMPONENT. (0,1,2,3,4,5,6,7,8,9,-,.,INTO)
RECORD AND RECOGNIZE FUNCTION. (SQUAROOT, SINE, COSINE, TANGENT, EXPONENTIAL, LOGARITHM, NATURALOG, ARCSINE, ARCCOSINE, ARCTANGENT)
RECORD AND RECOGNIZE REGISTER. (ZULU, STIK, STAR, FOX, KILO, OSCAR, PAPA, WEST, SISTER, WHISKY)
IS IT INTO?
NO
YES
RECORD AND RECOGNIZE PREPOSITION. (FROM, TO, OF, BY)
RECORD AND RECOGNIZE REGISTER. (ZULU, STIK, STAR, FOX, KILO, OSCAR, PAPA, WEST, SISTER, WHISKY)
RECORD AND RECOGNIZE LAST WORD. (EXECUTE, ERASE)
IS IT?
ERASE
EXECUTE
INTERPRET THE DECODED STATEMENT. LEARN THE NEW WORD REPRESENTATIONS MARGINALLY RECOGNIZED.

Figure VI-1. The Syntax-Directed Sentence Analyzer

When the word-recognizer makes an identification error, the experimenter may use the RUB OUT feature. The word RUB OUT is allowed anywhere within a statement (i.e., its print-name is part of all the tables of allowed symbols). Whenever this word is recognized, the process suppress the last recognized symbol from the statement written-form and replaces the syntax-directed sentence analyzer in its previous state by utilizing the remaining parts in the written-form of the statement. The process which returns to the previous state utilizes the grammar in the same manner as the interpreter uses it to analyze the stored written-form during execution.

VI-2-4. Comments

The sentences of the language defined by this grammar are short and quasi-natural. People can speak and remember these statements easily:

```
ADD   STIK  TO  ZULU        EXECUTE
COMPUTE  LOGARITHM  OF FOX  EXECUTE
DIVIDE  STAR  BY  SISTER    EXECUTE
```

Acoustic ambiguities arise only during the recognition of the digits (in particular 5 and 9). But for the sake of convenience, we decided not to change their names.

Although the DESCAL vocabulary may be globaly ambiguous (e.g., TO and TWO, FOR and FROM, BY and NINE are very likely to be confused), these ambiguities do not create any problem as a result of the use of the grammar.

VI-3. THE WORD RECOGNITION SYSTEM OF DESCAL

The previous section has described the language of DESCAL and how a grammar can be used to decode sentences made of several words. The present section explains the process which records and recognizes a spoken terminal symbol, each time this action is requested by the syntax-directed analyzer. As in the previous chapters, the recognition of a given utterance is performed in two distinct steps, namely: generation of a description of the utterance and selection of a best-match candidate in a lexicon which contains the descriptions of previously "learned" utterances. The procedures which are described in the next three subsections are conceptually identical to those described in the Chapters III and IV of the present dissertation. However, since speed is one of the main goals of this second application, several important modifications had to be made to the system in order to reduce as much as possible the time necessary to recognize a given utterance.

VI-3-1. The Utterance Description Generating Process

Utilizing the heuristics presented in Chapter III, a completely new process which analyzes the utterance and generates a compact description while the experimenter is talking was programmed. A detailed flowchart of this fast segmentation-classification-into-phoneme-groups is presented on Figure VI-2. This program is a real-time user's program, i.e., it is restarted every 16.7 ms by the supervisor and runs in parallel with the regular user's program. In fact, the latter is not running at that time since it is waiting for the A/D converter input operation to be finished (I/O which is terminated by the real-time user's program as shown on the flowchart). The sequence of operations is as follows: the regular user's

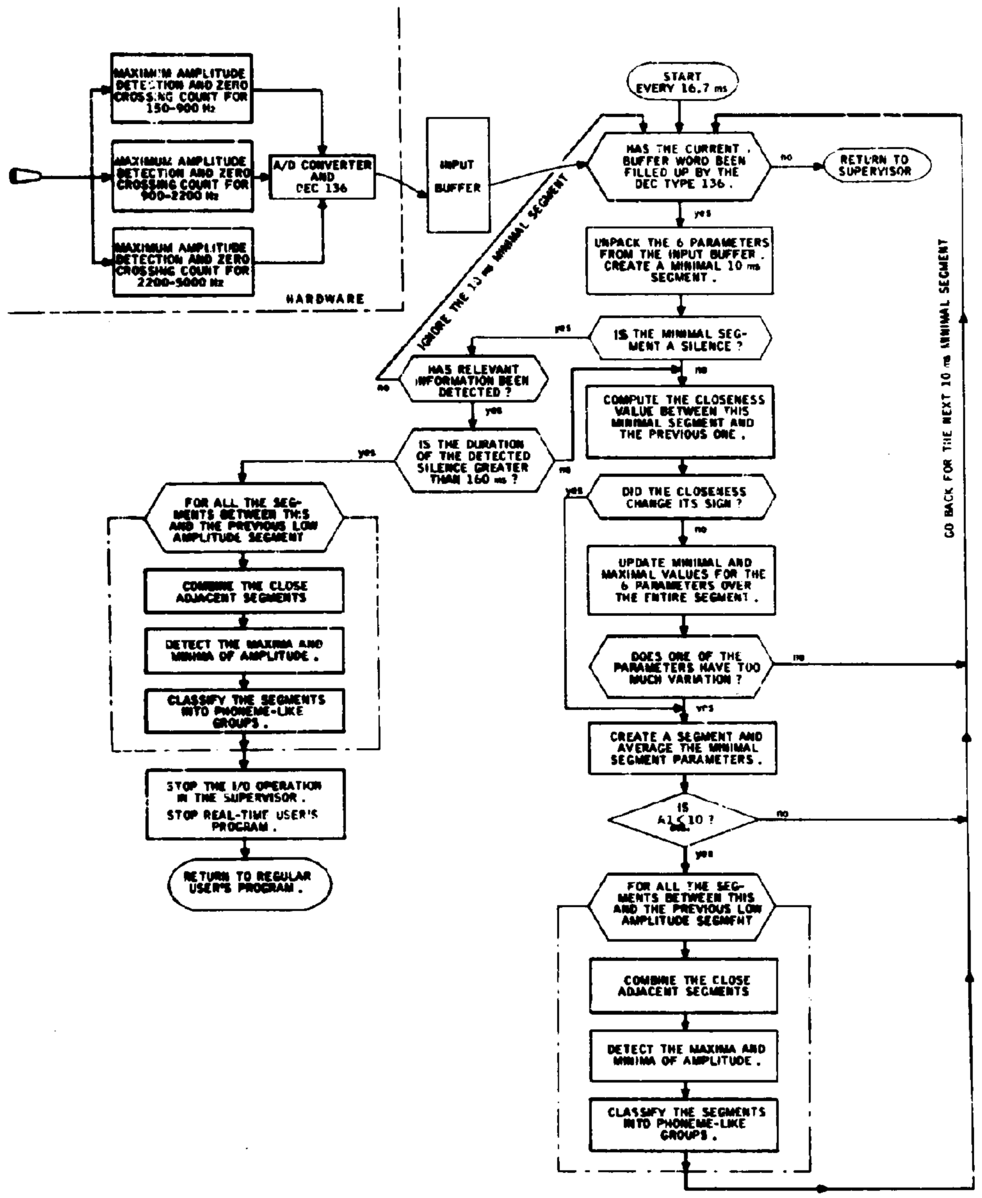

MAXIMUM AMPLITUDE DETECTION AND ZERO CROSSING COUNT FOR 150-900 Hz
MAXIMUM AMPLITUDE DETECTION AND ZERO CROSSING COUNT FOR 900-2200 Hz
MAXIMUM AMPLITUDE DETECTION AND ZERO CROSSING COUNT FOR 2200-5000 Hz
A/D CONVERTER AND DEC 136
HARDWARE
INPUT BUFFER
START EVERY 16.7 ms
HAS THE CURRENT BUFFER WORD BEEN FILLED UP BY THE DEC TYPE 136.
no
RETURN TO SUPERVISOR
yes
UNPACK THE 6 PARAMETERS FROM THE INPUT BUFFER. CREATE A MINIMAL 10 ms SEGMENT.
IS THE MINIMAL SEGMENT A SILENCE ?
yes
no
HAS RELEVANT INFORMATION BEEN DETECTED ?
no
IGNORE THE 10 ms MINIMAL SEGMENT
yes
IS THE DURATION OF THE DETECTED SILENCE GREATER THAN 160 ms ?
yes
no
FOR ALL THE SEGMENTS BETWEEN THIS AND THE PREVIOUS LOW AMPLITUDE SEGMENT
COMBINE THE CLOSE ADJACENT SEGMENTS
DETECT THE MAXIMA AND MINIMA OF AMPLITUDE.
CLASSIFY THE SEGMENTS INTO PHONEME-LIKE GROUPS.
STOP THE I/O OPERATION IN THE SUPERVISOR.
STOP REAL-TIME USER'S PROGRAM.
RETURN TO REGULAR USER'S PROGRAM.
COMPUTE THE CLOSENESS VALUE BETWEEN THIS MINIMAL SEGMENT AND THE PREVIOUS ONE.
DID THE CLOSENESS CHANGE ITS SIGN ?
yes
no
UPDATE MINIMAL AND MAXIMAL VALUES FOR THE 6 PARAMETERS OVER THE ENTIRE SEGMENT.
DOES ONE OF THE PARAMETERS HAVE TOO MUCH VARIATION ?
no
yes
CREATE A SEGMENT AND AVERAGE THE MINIMAL SEGMENT PARAMETERS.
IS A1 ≤ 10 ?
no
yes
FOR ALL THE SEGMENTS BETWEEN THIS AND THE PREVIOUS LOW AMPLITUDE SEGMENT
COMBINE THE CLOSE ADJACENT SEGMENTS
DETECT THE MAXIMA AND MINIMA OF AMPLITUDE.
CLASSIFY THE SEGMENTS INTO PHONEME-LIKE GROUPS.
GO BACK FOR THE NEXT 10 ms MINIMAL SEGMENT

Figure VI-2. The Real-Time Utterance Description Generating Program

program initiates the real-time program and the A/D converter input, these two processes are executed in parallel on the microphone input buffer. The control is returned to the regular program when a long enough silence has been detected in the input speech data.

By examining carefully the new segmentation process, one will be able to isolate the primary segmentation, the most important part of the secondary segmentation (parameters with too much variation in a sustained segment), the combining process, and the classification process. All of them are described in Chapter III. The main difference is that instead of a sequential execution, all the processes are now organized in a coroutine fashion. The primary and secondary segmentations are performed directly on the minimal segments. The combining, detection of extrema and classification are performed each time a low amplitude segment, which is necessarily a FRICS, STOP or BURST is found. The conceptual idea behind this classification scheme is that in general, one needs contextual information to decide the type of a given sonorant segment (VOWEL, CONST, NASAL). On the other hand low amplitude segments can be classified without referring to their adjacent segments. Therefore, they furnish anchor points between which a classification can be performed.

The function which computes the closeness between any two segments (closeness function) is described in Section III-2. The classification algorithm is kept in its original form (Section III-5), and so are the combining process and the extrema detection procedure (Section III-4). The complete real-time program returns to the regular user's program a standard feature matrix as presented on Section III-6 which enters directly the recognition process.

VI-3-2. <u>The Recognition Process</u>

The recognition process is almost identical to the process described in Chapter IV. The only two modifications which were performed are a translation of all the procedures into machine language and the removal of the error recovery procedure (Section IV-4-4) which has proven to be time consuming for a little improvement of performance. Since the experimenter is able to repeat his last utterance, in case of failure, we felt that the saving in time realized was worth the few additional errors.

VI-3-3. <u>The Learning Process</u>

The organization of the lexicon, which is described in the Section IV-2, is not altered for use in DESCAL. Consequently, the lexicon handler subroutines are left in the same form. Only the learning scheme was slightly modified so as to make the program self-improving. Two distinct learning phases are executed by the program. The first one takes place when the program is started. The experimenter is requested to speak three times the words of the vocabulary. These utterances are analyzed, the descriptions are generated and stored in the lexicon along with the corresponding character strings. Then the program accepts desk-calculator statements and recognizes each utterance. While this process is going on the utterance descriptions which represents the words of the sentence are stored along with the recognition scores they obtained and their corresponding character strings. When the word EXECUTE is recognized, the program, assuming that it correctly identified all the previous utterances, stores in the lexicon all the utterances which obtained low similarity scores, thus constantly improving its performances by a continuous learning. This process continues until the lexicon is full of learned data. The utterance description storage is, of course, altered by the RUB OUT command which deletes the last uttered word which is supposed incorrectly recognized.

VI-3-4. Comments

Since no statistical tests were performed on this word recognition process, we cannot compare it precisely to the process presented in Chapter IV. On the utterances it had to recognize, its performances were satisfactory (about 90 percent were correctly recognized), but these sentences were easy to segment (utterances composed of vowels, silences and fricatives) and the grammar was helping a great deal the recognition process. We believe that this program might not have performed as well if we had to segment several adjacent sonorant sounds or if we had to distinguish phonemically ambiguous words.

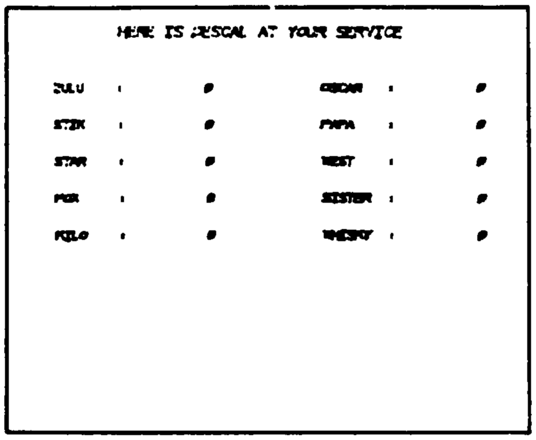
VI-4 RESULTS AND CONCLUSIONS

As in many of the preceding chapters, a typical computation executed on DESCAL will be explained by means of CRT pictures (Figures VI-3 through VI-14). They exhibit the computation of the hypotenuse length of a rightangled triangle when the lengths of the other sides are known. When the computation ends, the lengths of the sides are stored in the registers STIK and STAR, and the hypotenuse length is in PAPA. Besides the syntax-directed sentence decoder and the real-time utterance description generating process, DESCAL is provided with conversion routines for numbers (character string to floating-point representation and vice-versa) and a small interpreter which executes the desk-calculator statements. Since these procedures have not special interest, they will not be described here.

Figures VI-15 and VI-16 exhibit two error messages issued by DESCAL. SYNTAX ERROR means that the input sentence is not syntactically correct. Since the only symbols accepted by the input routine are the syntactically valid terminal symbols, this error message occurs very unfrequently, (unless there is an error in the coding of the algorithm which can then be traced and fixed). ARITHMETIC OVERFLOW means that one of the register value is too large (for example when dividing by 0). In this case, the command is not executed.

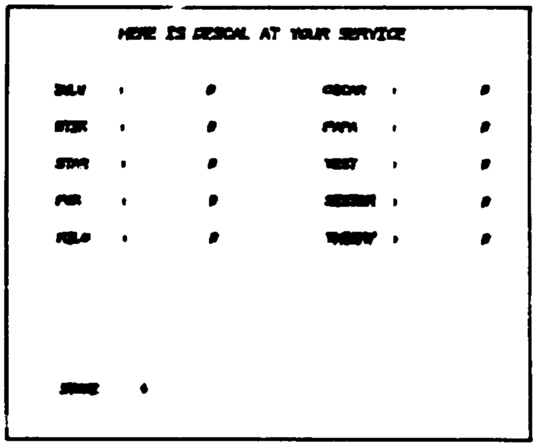
DESCAL is a working system which demonstrates that a reasonable desk-calculator responding in real-time to the experimenter's voice can be built on today's existing computers (about 5 times faster than a PDP-10). If we had a 30 times faster computer (like CDC 6600 or IBM 360/91), it may be possible to handle 20 people in time-sharing, which would perhaps be of interest, not so much for a desk-calculator type application, but as a data-retrieval system.

Figure VI-3. Initial state of DESCAL . The program is waiting for the microphone input .



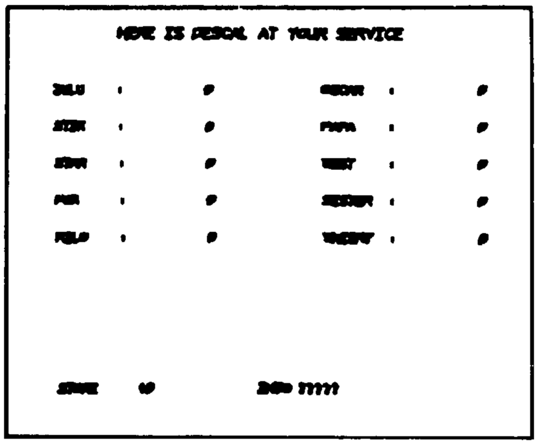
```
          HERE IS DESCAL AT YOUR SERVICE

ZULU  :      0          OSCAR  :      0
STIX  :      0          PAPA   :      0
STAR  :      0          WEST   :      0
FOX   :      0          SISTER :      0
KILO  :      0          THIRTY :      0
```

Figure VI-4. The operator STORE and the first digit 4 have been recognized, the program is waiting for the next number component .



```
          HERE IS DESCAL AT YOUR SERVICE

ZULU  :      0          OSCAR  :      0
STIX  :      0          PAPA   :      0
STAR  :      0          WEST   :      0
FOX   :      0          SISTER :      0
KILO  :      0          THIRTY :      0



STORE   4
```

Figure VI-5. The register which follows INTO has not been recognized . The program requests a new utterance of the same word by displaying ????? .



```
          HERE IS DESCAL AT YOUR SERVICE

ZULU  :      0          OSCAR  :      0
STIX  :      0          PAPA   :      0
STAR  :      0          WEST   :      0
FOX   :      0          SISTER :      0
KILO  :      0          THIRTY :      0



STORE   40          INTO ?????
```

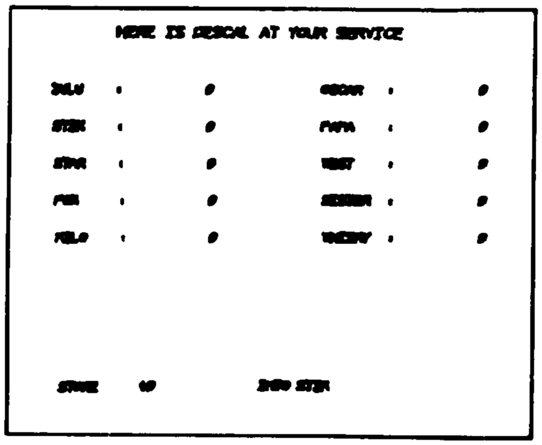

```
HERE IS DESCAL AT YOUR SERVICE

ZULU  :      0          OSCAR  :      0
STIK  :      0          PAPA   :      0
STAR  :      0          VEST   :      0
FUR   :      0          SISTER :      0
KILO  :      0          VICTORY :     0

STORE   10        INTO STIK
```

Figure VI-6. The desk-calculator statement has been entirely decoded and appears at the bottom of the CRT display .

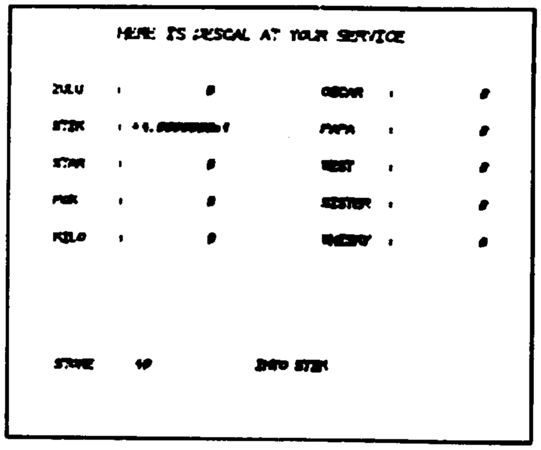

```
HERE IS DESCAL AT YOUR SERVICE

ZULU  :      0          OSCAR  :      0
STIK  : +1.[illegible]  PAPA   :      0
STAR  :      0          VEST   :      0
FUR   :      0          SISTER :      0
KILO  :      0          VICTORY :     0

STORE   10        INTO STIK
```

Figure VI-7. When the experimenter says EXECUTE , the command is executed . The value appears in the register STIK .

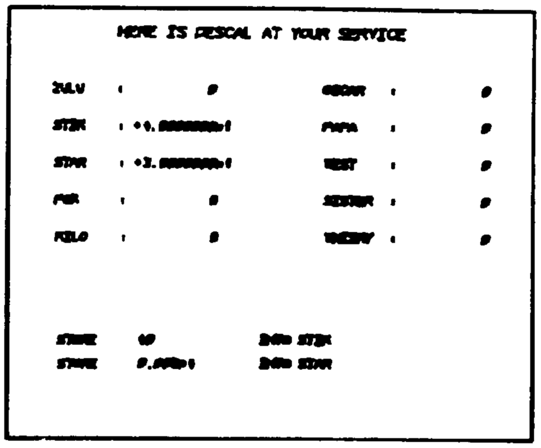
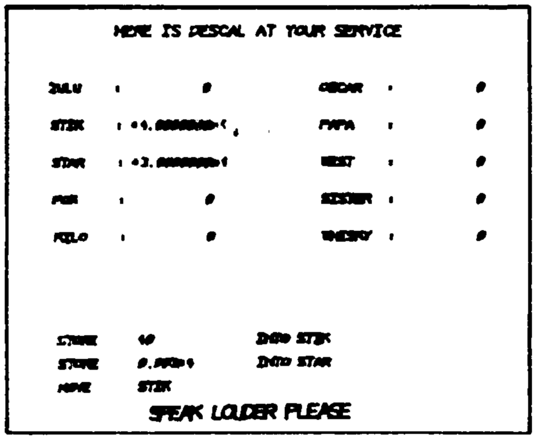
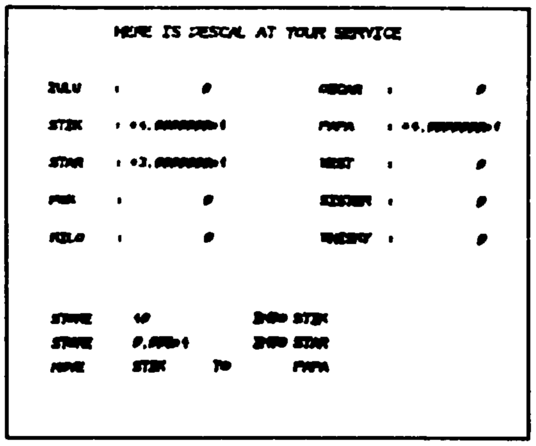

```
HERE IS DESCAL AT YOUR SERVICE

ZULU  :      0          OSCAR  :      0
STIK  : +1.[illegible]  PAPA   :      0
STAR  : +3.[illegible]  VEST   :      0
FUR   :      0          SISTER :      0
KILO  :      0          VICTORY :     0

STORE   10        INTO STIK
STORE   [illegible]   INTO STAR
```

Figure VI-8. The next statement has been decoded and executed .



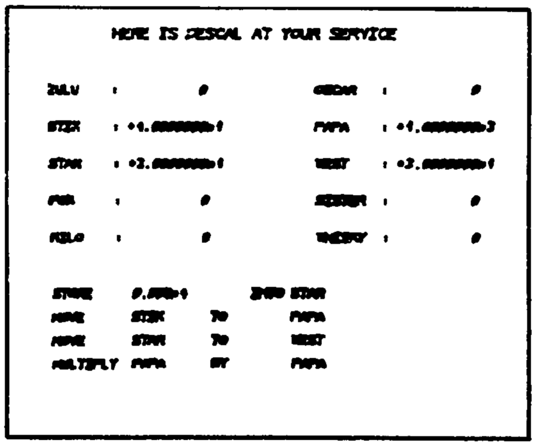
```
               HERE IS DESCAL AT YOUR SERVICE

  ZULU    :         Ø              OSCAR   :          Ø
  STIK    : +1.0000000+1           PAPA    :          Ø
  STAR    : +2.0000000+1           WEST    :          Ø
  FOX     :         Ø              SIERRA  :          Ø
  KILO    :         Ø              WHISKY  :          Ø



  STORE      10            INTO STIK
  STORE      0.200+1       INTO STAR
  MOVE       STIK
              SPEAK LOUDER PLEASE
```

Figure VI-9. The preposition TO was found too weak (amplitude parameters were too small) . The program requests a louder version of it before attempting the recognition .



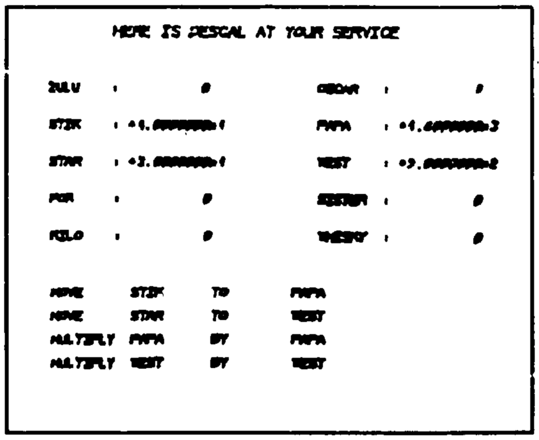
```
               HERE IS DESCAL AT YOUR SERVICE

  ZULU    :         Ø              OSCAR   :          Ø
  STIK    : +1.0000000+1           PAPA    : +1.0000000+1
  STAR    : +2.0000000+1           WEST    :          Ø
  FOX     :         Ø              SIERRA  :          Ø
  KILO    :         Ø              WHISKY  :          Ø


  STORE      10            INTO STIK
  STORE      0.200+1       INTO STAR
  MOVE       STIK      TO      PAPA
```

Figure VI-10. The computation executed on DESCAL proceeds .



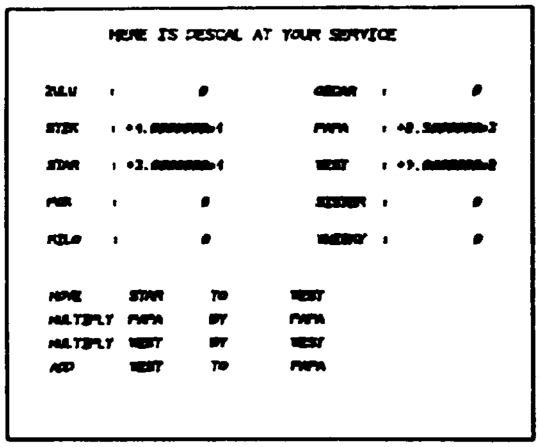
```
               HERE IS DESCAL AT YOUR SERVICE

  ZULU    :         Ø              OSCAR   :          Ø
  STIK    : +1.0000000+1           PAPA    : +1.0000000+3
  STAR    : +2.0000000+1           WEST    : +2.0000000+1
  FOX     :         Ø              SIERRA  :          Ø
  KILO    :         Ø              WHISKY  :          Ø

  STORE      0.200+1       INTO STAR
  MOVE       STIK      TO      PAPA
  MOVE       STAR      TO      WEST
  MULTIPLY   PAPA      BY      PAPA
```

Figure VI-11. One can easily follow the computation on the 10 registers of the machine . Only the 4 most recent statements are displayed on the CRT screen .



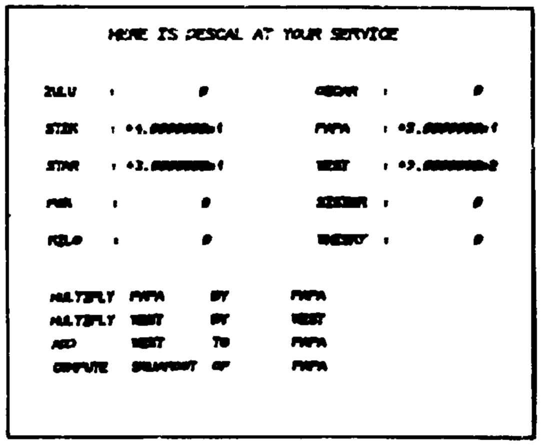
```
        HERE IS DESCAL AT YOUR SERVICE

ZULU   :         Ø          OSCAR   :         Ø
STIK   : +4.0000000E+1      PAPA    : +1.6000000E+3
STAR   : +3.0000000E+1      WEST    : +9.0000000E+2
FOX    :         Ø          SIERRA  :         Ø
KILO   :         Ø          WHISKY  :         Ø

MOVE      STIK    TO     PAPA
MOVE      STAR    TO     WEST
MULTIPLY  PAPA    BY     PAPA
MULTIPLY  WEST    BY     WEST
```

Figure VI-12. The program computed the squares of the two sides of the rightangled triangle in the registers PAPA and WEST .



```
        HERE IS DESCAL AT YOUR SERVICE

ZULU   :         Ø          OSCAR   :         Ø
STIK   : +4.0000000E+1      PAPA    : +2.5000000E+3
STAR   : +3.0000000E+1      WEST    : +9.0000000E+2
FOX    :         Ø          SIERRA  :         Ø
KILO   :         Ø          WHISKY  :         Ø

MOVE      STAR    TO     WEST
MULTIPLY  PAPA    BY     PAPA
MULTIPLY  WEST    BY     WEST
ADD       WEST    TO     PAPA
```

Figure VI-13. PAPA now contains the square of the hypotenuse length .



```
        HERE IS DESCAL AT YOUR SERVICE

ZULU   :         Ø          OSCAR   :         Ø
STIK   : +4.0000000E+1      PAPA    : +5.0000000E+1
STAR   : +3.0000000E+1      WEST    : +9.0000000E+2
FOX    :         Ø          SIERRA  :         Ø
KILO   :         Ø          WHISKY  :         Ø

MULTIPLY  PAPA        BY     PAPA
MULTIPLY  WEST        BY     WEST
ADD       WEST        TO     PAPA
COMPUTE   SQUAREROOT  OF     PAPA
```

Figure VI-14. The computation terminates here . The lengths of the sides are in STIK and STAR . The hypotenuse length is in PAPA.

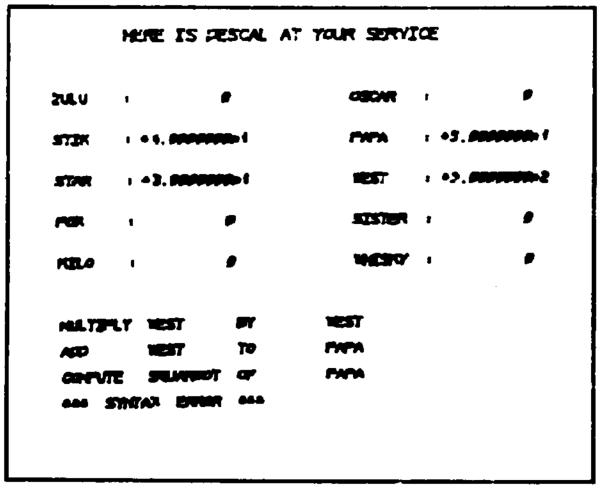

```
        HERE IS DESCAL AT YOUR SERVICE

ZULU    :          0          OSCAR  :           0

STIK    : +4.0000000E+1       PAPA   : +5.0000000E+1

STAR    : +3.0000000E+1       WEST   : +9.0000000E+2

FOX     :          0          SISTER :           0

KILO    :          0          WHISKY :           0


MULTIPLY  WEST       BY       WEST
ADD       WEST       TO       PAPA
COMPUTE   SQUAREROOT OF       PAPA
*** SYNTAX ERROR ***
```

Figure VI-15. The decoded statement was not syntactically valid . The statement was erased, replaced by the inscription :
*** SYNTAX ERROR ***, and not executed .

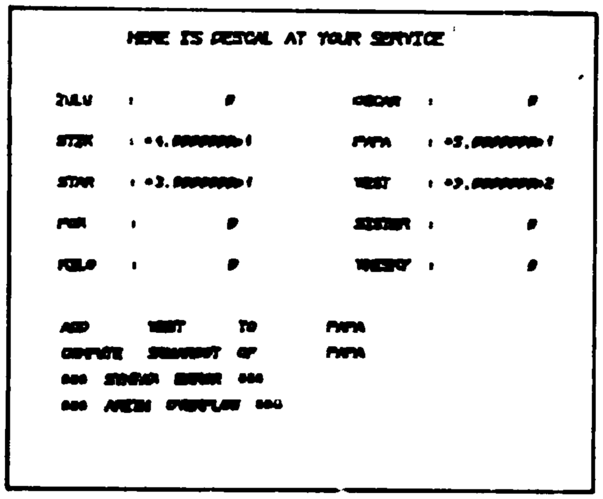

```
        HERE IS DESCAL AT YOUR SERVICE

ZULU    :          0          OSCAR  :           0

STIK    : +4.0000000E+1       PAPA   : +5.0000000E+1

STAR    : +3.0000000E+1       WEST   : +9.0000000E+2

FOX     :          0          SISTER :           0

KILO    :          0          WHISKY :           0


ADD       WEST       TO       PAPA
COMPUTE   SQUAREROOT OF       PAPA
*** SYNTAX ERROR ***
*** ARITH OVERFLOW ***
```

Figure VI-16. A division by 0.0 has been ordered . The desk-calculator complains .

Chapter VII

CONCLUSIONS

The principal goal of this work has been the efficient recognition of speech by a computer. To achieve this, several hardware and software techniques had to be developed. New approaches to problems already solved, and new solutions to problems not investigated before have been described Amongst these are:

-1. Implementation of a preprocessor, which extracts amplitude and zero-crossing parameters in adequate frequency bands.

-2. Use of a closeness function to segment the utterances on the basis of their acoustical properties.

-3. Investigation of the segment synchronization problem and design of a working solution which uses the segments obtained through the segmentation process.

-4. Reduction of the candidate space by a carefully organized lexicon and by tests performed on rough global features characterizing the utterances to be matched.

-5. Investigation of the problem of recognition of connected speech utterances and implementation of two working systems able to decode sentences of limited languages.

The research described in the preceding chapters leads us to the following conclusions about the nature and methodology of speech analysis and recognition by a computer.

-1. The fact that, using very simple-minded and crude parameters, we have been able to achieve very high recognition scores implies that the present preoccupation with spectrum, formant trackers, polynomial expansions, etc... seems unwarranted, and any one of these schemes will work reasonably well provided the subsequent algorithms are carefully designed. The reasons we used the zero-crossing technique are that it is simple and direct to obtain. One should be able to replace it by any other technique and still obtain good, it not better, results. If the prime interest of the reader is speech recognition, we suggest that he get out of the preprocessing loop and investigate the subsequent and intellectually stimulating problems.

-2. The present controversy about the impossibility of phonemic segmentation also seems unwarranted. In our investigation, we do not ask whether phoneme, syllable, or word are the units to be dealt with. We use every one of these concepts ranging from subphonemic level to word level at various stages of the analysis.

-3. In the recognition of limited languages (even with large vocabularies), it is unnecessary to have very accurate phoneme-like classification, although such feature might be useful to reduce the search, it remains to be seen whether the reduction resulting from highly accurate classification will exceed the amount of work needed to perform the necessary classification and associated error recovery task.

-4. Almost any reasonable classification technique will work in speech recognition, however if one is to hope to achieve recognition close to human abilities, then one has to rely heavily on the techniques developed in Artificial Intelligence research, namely: reduction of search space by means of heuristics based on the problem characteristics.

-5. It is not clear whether one has to ever try to distinguish between unvoiced stops /p/, /t/, /k/, (or other such similar groups) if he is dealing with limited languages, even if the language is some form of simplified English It appears that the syntactic constraints provided by the grammar are sufficiently powerful to resolve any such ambiguity. In the light of our experiments, it seems reasonable to spend effort in devising techniques to build more and more powerful grammar rather than in deciding what preprocessing techniques to use.

Because of the immensity of the task, it was not possible to consider some other aspects of speech recognition which have to be solved before we can have any semblance of a sophisticated speech recognition system. Future work on the subject should include:

-1 Implementation of a learning system.

The success of this speech recognition system and its accuracy are based on a large number of weights and thresholds. At present, these quantities are adjusted by hand through long and tedious debug-modify cycles. The implementation of a learning program which will replace these "handcrafted" values by accurate numbers deduced from the examination of thousands of data should considerably improve the system.

-2. Elimination of the variation of parameters from one speaker to another. At present, the system must be trained with the experimenter's voices in order to have a good recognition rate. This defect may partly disappear if the system is provided with "learned" coefficients obtained through the processing of speech data uttered by a large number of different speakers. A more promising approach which is under investigation at the moment is the normalization of the preprocessor output on the basis of voice characteristics for a given speaker. The data necessary to the normalization program could be obtained by the analysis of a standard set of words uttered by the speaker before any experimentation

-3. Study and design of languages well suited for speech recognition.

The main shortcoming of the two languages presented in this thesis is their simplicity. A careful choice of the vocabularies and more sophisticated grammars should allow the implementation of what we can call: spoken programming languages. For these, not only should the sentences be syntactically unambiguous, but it should be possible to determine word boundaries and to recognize words unambiguously. This extension of the present work is also under investigation, and interesting solutions should be obtained in the next few years.

The list of possible extensions of this thesis could be easily expanded. It is clear that a large number of problems remain to be solved in the area of automatic speech recognition. The computers are not yet capable of carrying on fluent conversations, but for the first time, a computer program was able to "understand" a simple English sentence made of several words and to "execute" the corresponding task.

BIBLIOGRAPHY

1. Alter, R. (1968), "Utilization of Contextual Constraints in Automatic Speech Recognition", IEEE Trans., AU16: 1, 6-11.

2. Bobrow, D. G. and Klatt, D. H. (1968), "A Limited Speech Recognition System", Proceedings of the AFIPS Fall Joint Computer Conference, Thompson, Washington, D. C., 33: 305-318.

3. David, K. H., Biddulph, R. and Balashek, S. (1952), "Automatic Recognition of Spoken Digits", J. Acoust. Soc. Am., 24, 637-642.

4. Dolansky, L. O. (1960), "Choice of Base Signals in Speech Signal Analysis", IRE Trans. on Audio, 8, 221-229.

5. Dudley, H. (1939), "The Vocoder", Bell Labs., Record 17, 122-126.

6. Ernst, H. A. (1961), "MH-1 a Computer Operated Mechanical Hand", Ph.D. Thesis, MIT, Cambridge.

7. Flanagan, J. L. (1965), Speech Analysis, Synthesis and Perception, Academic Press, Inc., New York.

8. Fry, D. B. and Denes, P. (1955), "Experiments in Mechanical Speech Recognition", Information Theory, Butterworth and Co., London 206-212.

9. Glaser, D., McCarthy, J. and Minsky, M. (1964), "The Automated Biological Laboratory", Report of the Subgroup on ABL Summer Study in Exobiology Sponsored by the Space Science Board of the National Academy of Sciences.

10. Gold, B. (1966), "Word Recognition Computer Program", Technical Report 452, Lincoln Laboratories, MIT. Cambridge.

11. Hughes, F. W. and Hemdal, J. F. (1965), "Speech Analysis", TR-EE65-7, Purdue Research Foundation, Lafayette, Indiana.

12. Mathews, M W., Miller, J. E. and David, E. E. (1961), "Pitch Synchronous Analysis of Voiced Sounds", J. Acoust. Soc. Am., 33, 179-186.

13. McCarthy, J., Earnest, L. D., Reddy, D. R., and Vicens, P. J. (1968), "A Computer with Hands, Eyes and Ears", Proc. of FJCC 1968, Thompson, Washington, D. D., 33: 329-338.

14. Munson, W. A. and Montgomery, H. C. (1950), "A Speech Analyzer and Synthesizer", J. Acoust. Am., 22, 678(A).

15. Peterson, E. (1951), "Frequency Detection and Speech Formants", J. Acoust. Am., 23, 668-674.

16. Peterson, G. E. and Barney, H. L. (1952), "Control Methods Used in a Study of the Vowels", J. Acoust. Am., 24, 175-184.

17. Pieper, D. (1968), "Kinematics of Manipulators under Computer Control", Ph.D Thesis, Stanford University, A.I. Memo No. 72, Stanford Artificial Intelligence Project, Stanford.

18. Pingle, K. K. (1966), "A Proposal For a Visual Input Routine", A.I. Memo No. 42, Stanford Artificial Intelligence Project, Stanford.

19. Pingle, K. K., Singer, J. A. and Wichman, W. M. (1968), "Computer Control of Mechanical Arm Through Visual Input", Proc. of IFIP 1968, H140-146.

20. Reddy, D. R. (1966), "Segmentation of Speech Sounds", J. Acoust. Soc. Am., 40, 307-312.

21. Reddy, D. R. (1966), "Phoneme Grouping for Speech Recognition", J. Acoust. Soc. Am., 41, 1235.

22. Reddy, D. R. (1967), "Computer Recognition of Connected Speech", J. Acoust. Soc. Am., 42, 329-347.

23. Reddy, D. R. and Robinson, A. E. (1968), "Phoneme-to-Grapheme Translation of English", IEEE Trans., AU16: 2, 240-246.

24. Reddy, D. R and Vicens, P. J. (1968), "A Procedure for Segmentation of Connected Speech", J. Audio Engr. Soc., 16: 4.

25. Reddy, D. R. and Vicens, P. J. (1969), "Spoken Message Recognition by a Computer", to be published.

26. Roberts, L. G. (1963), "Machine Perception of Three-Dimensional Solids", Optical and Electro-Optical Processing of Information, MIT Press, Cambridge.

27. Sakai, T. and Doshita, S. (1963), "The Automatic Speech Recognition System for Conversational Sound", IEEE Trans. on Electronic Computers, 12, 835.

28. Vicens, P. J. (1968), "A Speech Preprocessing Device" A.I. Memo No. 71, Stanford Artificial Intelligence Project, Stanford.

29. Wichman, W. M. (1967), "Use of Optical Feedback in the Computer Control of an Arm", A.I. Memo No. 56, Stanford Artificial Intelligence Project.