ON THE FEASIBILITY OF VOICE
INPUT TO AN ON-LINE COMPUTER
PROCESSING SYSTEM

by

Howard A. Elder

Technical Report

No. 69-38

July 1969

Cornell University
Department of Computer Science
Ithaca, New York 14850

ON THE FEASIBILITY OF VOICE INPUT TO AN
ON-LINE COMPUTER PROCESSING SYSTEM

A Thesis

Presented to the Faculty of the Graduate School
of Cornell University for the Degree of
Doctor of Philosophy

by

Howard Anthony Elder

September, 1969

BIOGRAPHICAL SKETCH

The author was born in Flushing, New York, on January 13, 1940. He attended Public School 94 in Little Neck, New York, followed by Brooklyn Technical High School, where he graduated in June, 1957. In September of the same year he began his undergraduate education at Cornell University. He received the degree, Bachelor of Electrical Engineering (a five- year course), in September, 1961. He then began graduate study at Cornell in Engineering Physics, and received the degree, Master of Science, in June, 1963. He entered the United States Navy in March, 1963, and served fifteen months on the guided missile destroyer U.S.S. Harry E. Yarnell. He was then transferred to the United States Naval Academy at Annapolis, Maryland, where he served as an instructor of physics for one year, after which he received an Honorable Discharge from the Navy. He returned to Cornell University in June, 1966, for graduate work in Computer Science. His major financial support was from the CUPL project (Cornell University Programming Language) of which he was a member.

DEDICATION

To the memory of
Johann Sebastian Bach

ACKNOWLEDGEMENTS

The author wishes to acknowledge the guidance of his major advisor, Professor Christopher Pottle, and Professor William L. Maxwell; without the guidance, encouragement, and interest given by both Professor Pottle and Professor Maxwell it would not have been possible to complete this work. The author also wishes to acknowledge Professors C. J. Earle, S. H. Schanuel, N. H. Bryant, and Messrs. H. Moraff, and P. Hiltz of the Radiation Biology Laboratory at Cornell University.

TABLE OF CONTENTS

CHAPTER	PAGE
I. OBJECTIVES OF THIS RESEARCH	1
I.A. INTRODUCTION	1
I.B. OBJECTIVE	2
I.C. SKETCH OF THE PROPOSED SYSTEM	2
II. SOLVING THE INPUT PROBLEM- THE PATTERN MATCHING SCHEME	12
II.A. OVERALL DESCRIPTION OF THE PATTERN RECOGNITION SCHEME	12
II.B. FORMING THE ARRAY FOR A GIVEN UTTERANCE	17
II.B.1 DETERMINING THE PRESENCE OF AN UTTERANCE	17
II.B.2 ALIGNING THE UTTERANCE WITHIN ITS ONE-SECOND WINDOW	19
II.B.3 TIME SCALING THE UTTERANCE	20
II.B.4 VOLUME COMPRESSION AND ENERGY NORMALIZATION	23
II.C. FORMING THE REFERENCE ARRAY FOR A GIVEN WORD	25
II.D. STORAGE REQUIREMENTS FOR AN ARRAY	26
II.E. THE DETECTION PROBLEM	27
III. EXPERIMENTAL RESULTS	30
III.A. DESCRIPTION OF THE EXPERIMENTAL SETUP	30
III.B. DESCRIPTION OF THE WORD-RECOGNITION PROGRAM	33
III.B.1 PHASE A1- THE FOURIER ANALYSIS PROGRAM	37
III.B.2 PHASE A2- LOCATION OF THE PRINCIPAL PEAK AND COMPUTATION OF THE TIME SCALING FACTOR	47
III.B.3 PHASE B- FORM STANDARD ARRAYS	57
III.B.4 PHASE C- THE DETECTION PROCEDURE	60
III.C. EXPERIMENTAL RESULTS	65
III.D. TIMING CONSIDERATIONS	76
IV. CONCLUSIONS	78
APPENDIX A: LISTING OF SOURCE PROGRAMS	82
REFERENCES	113

LIST OF ILLUSTRATIONS

FIGURE	PAGE
1. SAMPLE CUPL PROGRAM	5
2. WORD LIST	7
3. SAMPLE AUDIO-CUPL INPUT	9
4. EQUIPMENT FOR SPEECH INPUT	15
5. SPEECH DATA FORMAT	16
6. GRAPHS OF TWO SIMILAR FUNCTIONS	22
7. INFORMATION FLOW IN EXPERIMENTAL SETUP	31
8. BLOCK DIAGRAM OF THE WORD-RECOGNITION PROGRAM	36
9. PHASE A1	44
9a. PERIODIC FUNCTION FOR FOURIER ANALYSIS	45
9b. PASS BANDS OF FOURIER FILTER	46
10. PHASE A2	50
11. SAMPLE POWER CURVE FOR "COMMA"	51
12. ALIGNMENT OF "LET" BY PHASE A2	55
13. PHASE B	58
14. PHASE C	62
15. PATTERN FOR "COMMA"	63
16. GRAPH OF EXPERIMENTAL RESULTS OF PATTERN MATCHING	71
17. CHOICES OF CROSSOVER FREQUENCIES FOR FILTERS	73
18. ERROR RATES FOR VARIOUS CHOICES OF KEY PARAMETERS	74

ABSTRACT

An on-line digital computer processing system is considered in which an ordinary telephone is the complete terminal device, input to the computer being provided as a sequence of spoken words, and output to the user being audio responses from the machine. The feasibility of implementing such a system with a FORTRAN-LIKE algebraic compiler as the object processor is considered. Complete details of a specific word recognition program are given. This technique depends on three simplifying restrictions, namely, a "small" vocabulary set, "known" speakers, and a "moment of silence" between each input word. Experimental results are presented giving error rates for different experimental conditions as well as the machine resources required to accommodate several users at a time. The results show that at this time it is both economically and logically feasible to handle at least 40 users at a time with an IBM 360/65 computer.

CHAPTER I

OBJECTIVES OF THIS RESEARCH

A. Introduction

Much work has been done in the area of speech recognition using a digital computer [1], but there has yet to appear an implementation using speech input to a computer processing system. In 1966 a system was proposed using push button input and audio output for communicating with an algebraic language processor over a Touch-Tone telephone [2], and, since that time, systems using this strategy have been implemented [3]. The appearance of words such as "LOAD", "STORE", "ADD", and "SUBTRACT" in one researcher's [4] vocabulary set suggests that he was interested in speech input to a computer language processor, but the speech recognition experts seem to be aiming strictly for further research results rather than applying those techniques available as current state of the art.

The general speech recognition problem is certainly formidable, and its solution probably many years away, but, in the special environment of "known" speakers and a "small" vocabulary set, the techniques presented in this research constitute a feasible solution to the problem of accepting as computer input a sequence of utterances spoken over an ordinary telephone.

B. Objective

The objective of this research is to show that it is currently feasible to "talk" or "converse" with a digital computer by communicating information as a series of utterances spoken over an ordinary telephone. Complete details of a specific speech-recognition technique are presented and the machine resources required by this technique are examined. Experimental results are presented giving the word recognition error rate for various choices of the key parameters of the word recognition program.

It should be noted that the speech recognition technique presented here could be applied to any on-line digital computer processing system that can function within the restrictions of "known" speakers, and a "small" vocabulary set (see Chapter II for a complete description of the restrictions imposed by this technique). For example, the on-line object system receiving the spoken information could be an algebraic programming language processor such as FORTRAN, a hotel accounting and reservation system, or a "desk calculator" system. In this research, however, attention is focused on the CUPL compiler as the object processor; this specific system is outlined in the following section.

C. Sketch of the proposed system

CUPL (Cornell University Programming Language) is a

scientific problem-oriented language used for all introductory instruction at Cornell [5]. The existing implementation for the IBM 360 is a fast (about 40000 statements/ minute compile, and 125000 statements/ minute execute on a model 65), core-resident, error-correcting, batch processing system. The syntax is simple and is designed to facilitate error-correcting. Each statement begins optionally with a label, followed by a "first word", which defines the statement type. Extensive matrix manipulating capabilities are built into the language. The statement types are:

- COMMENT statement
- LET, the assignment statement
- IF, the logic statement
- GO TO, the unconditional branch statement
- STOP, which terminates execution
- READ, the input statement
- WRITE, the output statement
- ALLOCATE, the dynamic array storage allocation statement
- WATCH, a debugging aid that causes the values of the WATCHed variables to be printed when the value changes.

Loops and subroutines are effected with a block structure. These facilities use the following statement types:

- BLOCK, which defines the beginning of a block
- END, which defines the end of a block
- PERFORM, which provides the linkage to a block, with

optional looping and/or indexing facilities

An example of a CUPL program for finding the roots of a quadratic equation is presented in Fig. 1.

```

      READ N
      PERFORM QUADSOL N TIMES
      STOP
QUADSOL  BLOCK
      READ A,B,C
      WRITE A,B,C
      LET D=B*B-4*A*C
      IF D LT 0 THEN GO TO OTHER
      LET E=SQRT(D)
      LET X1=(-B+E)/(2*A)
      LET X2=(-B-E)/(2*A)
      WRITE X1,X2
      GO TO QUADSOL END
OTHER    LET E=SQRT(-D)
      LET XREAL=-B/(2*A)
      LET XIMAG=E/(2*A)
      WRITE XREAL, XIMAG
QUADSOL  END
*DATA    N=4
      A=1, B=1, C=-6
      A=1,B=2, C=5
      A=-3, B=-.25, C=0
      A=3.14159,B=2.71828, C=1.4142

```

FIGURE 1

SAMPLE CUPL PROGRAM

The objective, then, is to show that it is feasible to construct a system, say "Audio-CUPL", with the property that one can speak his CUPL program into the computer over an ordinary telephone, the telephone also receiving the output from the system; that is, the telephone is the complete terminal device.

Since the proposed word recognition program requires a "small" vocabulary set, some restriction must be imposed on the syntax of CUPL when the CUPL program is intended for the Audio-CUPL system: variable names might be of the special form V_n , and statement labels might be of the form L_n , where n is an integer. Examples of legal variable names are V_1 , V_{13} , and V_5 , and some legal statement labels are L_5 , L_{21} , and L_{10} .

The vocabulary set made available to the user of Audio-CUPL is given in Fig. 2. The word recognition program operates on a whole word as a single entity: it does not segment the word into smaller units such as syllables or phonemes. In addition, a "moment of silence" is required between each spoken word (see Chapter II). Hence, "GO TO" is considered one "word" for recognition purposes.

ZERO	
ONE	READ
TWO	STOP
THREE	COMMA
FOUR	IF
FIVE	THEN
SIX	ELSE
SEVEN	AND
EIGHT	NOT
NINE	OR
"V"	ALLOCATE
"L"	PERFORM
PLUS	TIMES
MINUS	WATCH
SLASH	GREATER
STAR	LESS
LEFT	WHILE
RIGHT	EQUAL
LET	POINT
GOTO	HALT

FIGURE 2
WORD LIST

Note that the constant 131 would be spoken as "ONE" \$ "THREE" \$ "ONE", where the \$ is used to denote the moments of silence required between each utterance. Similarly, a variable name V13 would be spoken as "V" \$ "ONE" \$ "THREE". The same procedure, of course, applies to statement labels. The words LEFT and RIGHT represent the two parentheses, and RIGHT can serve the dual purpose of) and WRITE since syntax resolves the ambiguity. HALT is the end of transmission (or end of statement) delimiter, spoken at the end of each CUPL statement to signal the machine to process the statement.

The sample CUPL program of Fig. 1 can be translated to a form suitable for Audio-CUPL by using the vocabulary set of Fig. 2., and the rules specified above. Part of a translation is given in Fig. 3, where the "transmission" is ended after each CUPL statement. In order to use the CUPL functions, such as the square-root function, the word list of Fig. 2 would be supplemented with appropriate function names for ABS, ATAN, COS, EXP, FLOOR, LN, SIN, and SQRT.

READ \$ V \$ ONE \$ HALT \$
 PERFORM \$ L \$ ONE \$ V \$ ONE \$ TIMES \$ HALT \$
 STOP \$ HALT \$
 L \$ ONE \$ BLOCK \$ HALT \$
 READ \$ V \$ TWO \$ COMMA \$ V \$ THREE\$ COMMA \$ V \$ FOUR\$ HALT \$
 WRITE \$ V \$ TWO \$ COMMA \$ V \$ THREE \$ COMMA \$ V \$ FOUR \$
 HALT \$
 LET \$ V \$ FIVE \$ EQUAL \$ V \$ THREE \$ STAR \$ V \$ THREE \$
 MINUS \$ FOUR \$ STAR \$ V \$ TWO \$ STAR \$ V \$
 FOUR \$ HALT \$

FIGURE 3
 SAMPLE AUDIO-CUPL INPUT

Output from the Audio-CUPL system to the user would also be audio, furnished from a fixed vocabulary set, of which Fig. 2 is a subset, and "played back" to the user's telephone by some type of audio response unit under computer control. Several pieces of suitable hardware are currently available, some based on pre-recorded magnetic (audio) tape loops, and some using Digital-to-Analog (D/A) converters (for example, the IBM 7772 Audio Response Unit). After each CUPL statement has been spoken in (the HALT has been reached), the system would respond with a reading of the statement back to the user as it had been understood by the system. Any errors detected at this time by either the system or the user would be resolved by further instructions to the system. A "command" or "job control language" would be required to facilitate this man-machine interaction, and several command words would be added to the vocabulary set of Fig. 2. This interaction would be handled in a manner similar to that described for Tele-CUPL: see [2]. A facility would be provided for correcting a statement containing errors, and if the execution output becomes too lengthy, the user would direct the output to a high-speed printer. Since the word recognition program depends on "knowing" the user, a sign-on procedure is required to initially identify the user. The sign-on could be accomplished by using the dial (or push buttons) on the telephone after the computer has answered, with the user keying in an identifying number at this time. The computer

could identify the user by counting the clicks of the dial or the beeps of the Touch-Tone telephone. Alternatively, the computer could recite the digits in sequence and the user could speak any word after the digit identifying him; this would flag that digit to the system. This procedure could be repeated in order to pass an identifying number containing several digits.

CHAPTER II

SOLVING THE INPUT PROBLEM- THE PATTERN RECOGNITION SCHEME

A. Overall description of the pattern recognition scheme

As mentioned in the introduction, the current feasibility of a suitable word recognition program depends on imposing three simplifying restrictions on the system.

The first restriction is that the number of words in the vocabulary set to be recognized be "small"- say between 50 and 100. This restriction is imposed in order that the pattern-space is not so densely packed that the reference patterns are too "close" to one-another. Note that a careful choice of the words placed in the vocabulary set is also important in this regard; words that prove to be too "close" in the pattern-space (errors result in attempting to differentiate them) should be replaced by words that produce very distinct patterns. Clearly the larger the number of words in the vocabulary set, the more difficult it is to find a suitable new word which, when added to the set, will not be close to one or more existing words. OR and FOUR of Fig. 2 present such a problem.

The second basic restriction is that we require a "moment of silence" between each word uttered by the speaker. This is required in order to locate word boundaries on the incoming speech signal. In normal (connected) speech the sound level does not die down to

silence between many separate words, and the problem of determining word boundaries in this case is extremely difficult.

The third and most vital restriction is that the word-recognition program be required to recognize only "known" speakers. When a user approaches the system for the first time, he is required to undergo an "introductory phase" during which the system will calibrate the recognition process to the new user by having him repeat the words in a fixed order. The user-dependent information secured at this time is saved in mass storage (perhaps a disk) for subsequent occasions when this speaker will use the system. The information secured during the introductory phase consists of a "reference pattern" and certain other "features" for each word in the vocabulary set; this information is described in detail in section C.

The speech-recognition technique uses a time- and energy-normalized two-dimensional array as a representation of an utterance. This technique is a familiar one (for example, see [6]). One dimension of the array corresponds to time and the second dimension corresponds to a frequency analysis of the speech during the short time interval implied by the discrete time points. The equipment needed to obtain this information is shown in Fig. 4. The telephone transmission system is represented in this figure as an ideal microphone plus a noise source driving a band-pass filter to simulate the limited bandwidth of the

telephone channel. The telephone input is then fed into a bank of band-pass filters. The centers of the pass bands of these filters are arranged such that the filters collectively cover the audio frequency spectrum with little overlap. An envelope detector is applied to each filter's output, the output of the detector being proportional to the power in the specific frequency band averaged over a few milliseconds. The outputs of the detectors are then fed into a multiplexor that samples each detector in turn and passes the samples to an analog-to-digital (A/D) converter. The digitized samples are then fed into the memory of a digital computer through an Input-Output channel and appropriate interface equipment. The digital data secured by this arrangement is pictured in Fig. 5. The number $a(i,j)$ is proportional to the power output of filter i at the time instant j . A row of this array corresponds to a particular filter, and a column to a particular "time slot".

FIGURE 4

EQUIPMENT FOR SPEECH INPUT


```

filter 1      a(1,1) a(1,2) ..... a(1,m)
filter 2      a(2,1) a(2,2) ..... a(2,m)
...
...
...
filter n      a(n,1) a(n,2) ..... a(n,m)
               1       2           m ← time slot

```

FIGURE 5
SPEECH DATA FORMAT

B. Forming the array for a given utterance

1. Determining the presence of an utterance

Once a user is connected to the system, the problem arises of determining when he has begun an utterance. This problem is solved by applying a two-stage procedure. The first stage is to determine whether or not an utterance is present, and, once it has been determined that an utterance is present, an array of data (fig. 5) is collected for a one-second interval of the signal which includes the utterance. The second stage of the procedure is to locate the utterance within this one second interval; the first stage is described in this section, and the second stage is described in the following section.

The procedure for capturing an utterance in a one-second interval is as follows. The signal is streamed into memory, and after an interval of time t elapses (say 15 or 20 milliseconds) an integral is performed on this interval to determine if the average power during this interval exceeded q times (say $q=10$) the average noise power; the noise power may be determined shortly after connect occurs for each session with the computer. If the threshold q is exceeded, then it is assumed that an utterance is in progress. Note that two or three (this number can be fixed) of the intervals monitored before the interval that exceeds the threshold q will always be saved in a "wrap-around" memory arrangement so that the leading edge of the utterance

will not be lost should the threshold detector trigger part way into the utterance. A total of one second's data, including the intervals just preceding the interval that triggered the decision, are then saved in the form of an array (Fig. 5) and the rest of the word-recognition machinery is activated. The threshold integral may be performed numerically by the CPU, or an analog integration could be performed using a threshold detecting network that causes an interruption in the computer input channel when the threshold is exceeded. In this case the channel (of the IBM System/360, for example) could be programmed in a loop that fills the wrap-around core buffer until the trigger network causes the interruption. At this time the channel program would be changed in order to collect the balance of the one second interval. This arrangement makes good use of the channel's capabilities and at the same time saves the considerable CPU power that would be required by a running numerical integration. The presence of "noise utterances," such as a cough or "uh", will trigger the recognition process and will result in a spurious utterance in the recognized input data. For this reason the user will have to learn to control sounds not intended to be input data. Line noises and clicks on the telephone line will probably not, however, trigger the system because they appear, at worst, as large spikes which have a small average power over the interval t seconds long. The user will also have to be reasonably careful in choosing the room in which he speaks,

since loud conversation from bystanders or noisy machinery will certainly cause the number of errors to exceed the amount of useful data.

2. Aligning the utterance within its one-second window

The array of "raw data", captured by the process described in the above section, must be transformed into a "refined" array (again as in Fig. 5) in order for the recognition process to proceed. The refined array for a given utterance is referred to as the "standard array" or "standard pattern" for a given utterance. The first transformation required is to guarantee that a given word in the vocabulary set will always be aligned in time in the standard array to the central time slot (within a tolerance of about one time slot position). This is the second stage of the procedure for locating an utterance described in section 1. This further alignment of the utterance in time is required because the threshold procedure of section 1 will locate the beginning of the utterance only within an accuracy of several tens of milliseconds. This jitter is much larger than can be permitted by the pattern matching. The jitter arises from the fact that, for a given word, the relative maxima and slopes of leading edges do not bear a consistent relation to one another from one utterance of the word to another. The method used for this second stage

alignment is to locate, to a high degree of accuracy, the peak belonging to the absolute maximum in the raw data array. This peak will always be aligned to the central time slot position in the standard array. The algorithm for locating this peak is described in detail in the description of the programs given in Chapter III. It should be noted here, however, that this method depends on the user consistently placing the accent on the same syllable of a given word. For example, if, in the case of GOTO, GO is accented sometimes and TO at other times, the principal peak will correspond to different portions of the utterance in these two cases and problems will result.

3. Time scaling the utterance

In addition to aligning a characteristic point on the utterance (the principal peak) to a fixed position in the standard array, a "time scaling" is performed as well when the raw data array is transformed to the standard array. This time scaling corresponds to multiplying the real-time scale of the raw data by a constant slightly greater or slightly less than one. The time scaling is necessary because a given speaker will speak the same word at different rates on different occasions, these rates differing by more than 20% at times.

The time scaling is based on the fact that if two functions of time differ only by a constant multiple of the

independent variable, then the integrals of these two functions will differ by a factor equal to this constant. Consider two functions $f(x)$ and $g(x')$ such that $f(x)=g(ax)$, as in Fig. 6. Then, assuming that the integrals exist,

$$\int_0^{\infty} f(x) dx = \int_0^{\infty} g(ax) dx = \int_0^{\infty} g(ax) d(ax)/a = (1/a) \int_0^{\infty} g(x') dx'.$$

Hence, if $f(x)$ were one occurrence of an utterance's total power curve (obtained by summing columns in Fig. 5) as a function of real time, and $g(x')$ were another such real-time occurrence where the utterances have been spoken at a different rate, the ratio of the integrals of $f(x)$ and $g(x')$ give the time scaling factor a . The power curves must, of course, be normalized such that the ordinate scale is the same. This can be accomplished by performing a scaling such that the absolute maximum of the functions (points P and P' respectively) is one.

FIGURE 6

GRAPHS OF TWO SIMILAR FUNCTIONS

The success of this procedure depends on the power curves of different utterances of the same word having, at least to first order, the same shape. This hope seems to have been realized experimentally, and even where the relative magnitude of the major peak to a minor peak differs considerably from one utterance of a given word to another, a good first order time scaling correction will be provided by this method.

4. Volume compression, and energy normalization

The raw data array is transformed to a new array by aligning the characteristic point on the utterance to the central position in the new array, and at the same time applying the time scaling to the data as it is moved to the new array. In addition to these operations, volume compression and energy normalization are applied to the new array before it becomes a standard array representation of the utterance. These operations are described in this section.

Volume compression is based on the fact that the more energetic portions of an utterance do not carry an amount of distinguishing information proportional to the energy. For example, in the word SIT, the S-sound is of very small power compared to the IT sound, and yet the S-sound is the only distinguishing feature between SIT and IT. Volume compression is intended to provide a more equal weight for

all portions of the array, so that every point in the array can contribute significantly to the subsequent pattern matching calculations. Volume compression is applied using the exponential function

$$a'(i,j) = 1 - \exp\{-C*(a(i,j) - A_{\min})/A_{\max}\}$$

where $A_{\max}$ and $A_{\min}$ are the maximum and minimum (such that $A_{\min} \neq 0$) elements in the array, and C is a system parameter.

Finally, the "new array" becomes a "standard array" when the array is "energy normalized". The array may be thought of as a vector in a $n*m$ dimensional vector space by defining the components of the vector to be the rows of the array strung together:

$$a(11), a(12), \dots, a(1m), a(21), a(22), \dots, a(nm).$$

The energy normalization is performed to guarantee that the standard array is a "vector of unit length." This is accomplished by dividing each element of the new array $a(i,j)$ by a constant b such that the elements of the standard array, $a'(i,j) = a(i,j)/b$ satisfy

$$\sum_i \sum_j (a'(i,j))^2 = 1.$$

The purpose of this normalization is to minimize the effects of a speaker saying the same word with different loudnesses

on different occasions.

C. Forming the reference array for a given word.

Each utterance accepted by the system is formed into a standard array. During the "introductory phase" (that is, when a new user first approaches the system) the user is asked to recite the vocabulary set in a given order two or three times over. At this time the system "knows" what word of the vocabulary set an incoming utterance is supposed to represent, and the two or three occurrences of the same word may be used by the system to form a "reference array" for that word. The reference array for a given word is obtained by simply adding the corresponding points of the standard array representations of the several known utterances of that word, and then reapplying the energy normalization described above. Hence, the introductory process yields a reference array for each word in the vocabulary set, and in the subsequent recognition of an unknown utterance, the standard array produced from this unknown will be compared with the reference arrays. The unknown utterance will then be identified with the word corresponding to the reference array giving the best "match" with the unknown's standard array. Section E below gives more detail on the procedure used for matching the unknown to the reference arrays.

The reference arrays formed during the introductory phase are stored in some form of mass storage, such as a

disk pack. It is by means of these arrays that the word recognition program is calibrated separately to each user of the system: when an old user approaches the system his reference arrays are fetched from mass storage and used in the recognition of his utterances.

D. Storage requirements for an array

Since there must be enough main storage to hold a reference array for each word in the vocabulary set (perhaps 60 in number) the amount of storage used by a single array is of vital interest. This amount is determined by the total number of values stored in an array, and the number of bits used to represent a value. Hence, the storage for one reference array is $n*m*b$ bits, where b is the number of bits per point in the array. Note that if b is chosen so that the numbers stored in the array are not of a format that is standard on the computer being used, then additional processing overhead is incurred to perform the arithmetic called for by the word recognition program. The classic tradeoff between processing time and main storage size that occurs in many problems thus appears here also. It is probably well worth the few milliseconds or so of processing time required to expand the array into one whose points have a number format that is standard.

The "bit compression", namely the value b above, is imposed on the standard array during the process of volume

compression. If b is chosen as 4 bits per point, then only 16 volume level values are allowed in the volume compression function.

Note that n and m also affect the input/output channel loading, since the $n*m$ values in the array are transmitted to memory in a fixed time interval (one second).

E. The detection problem

The problem of deciding which word of the vocabulary set to identify with the standard array of an unknown utterance is called the "detection problem". Of interest is a "measure of closeness" between a standard array and a reference array. Several such measures might be imagined if one recalls that the arrays may be thought of as "unit vectors" in $n*m$ space (II.B.4). The scalar product (or dot product)

$$\text{COS} = \sum_i \sum_j a(i,j) * r(i,j)$$

gives the cosine of the angle between the two vectors, with $\text{COS}=1$ implying that the arrays are identical. The "distance between the tips" of the unit vectors is given by

$$\text{SDE} = \sum_i \sum_j (a(i,j) - r(i,j)) ** 2,$$

which is referred to as the square-difference error (SDE).

If $SDE=0$ then the arrays are identical. A simpler measure, the absolute difference error (ADE), could also be used:

$$ADE = \sum_i \sum_j |a(i,j) - r(i,j)|.$$

The SDE is used in this discussion.

Certain "features" of an utterance may be used to restrict the number of the reference arrays that are compared with the unknown's array. Only for those reference arrays with features "similar" to the unknown's array will an SDE be computed, and the unknown will tentatively be identified with the word corresponding to the reference array giving the smallest SDE. One such feature is the real-time length of the utterance, two words being considered "dissimilar" if their lengths differ by more than, say, 35- 40%. Another feature is the number of peaks that an utterance has in its power versus time curve. This feature is described in detail in Gold's paper [4], (see his Fig. 5). It will be seen in III.D that the use of features of this sort will be important in saving the central computer resource.

In the case of Audio-CUPL, the syntax of the language can be used as a check on the correctness of the identification of the unknown utterance. If the identification produces a syntax error then the unknown can be re-identified as corresponding to the reference array with the next smallest SDE. This procedure would correct a

word-recognition recognition error in the case where the intended word has the second best SDE, and the word with the smallest SDE produces a syntax error. The use of syntax in this way can help determine the choice of words placed in the vocabulary set by the system designer. In any case, the system designer must choose this word list carefully and strive for maximum mutual separation between the reference arrays.

The fact that the user is on-line during the recognition process also permits feedback from the user to be brought to bear in improving the accuracy of recognition. He may "recalibrate" (cause a new reference array to be created) words he has repeated trouble with, since he is probably pronouncing the word significantly different than at the time of his previous calibration. He may also correct errors immediately since the echo from the system will inform him of the error. However, the system would not "reject" an utterance (request the user to repeat) when the utterance does not match well to any reference array, since the requests for repeats would be greater than the number of errors that would otherwise occur (see III.C).

CHAPTER III

EXPERIMENTAL RESULTS

A. Description of the experimental setup.

The actual equipment used in the development and testing of the word-recognition program is pictured in Fig. 7. The utterances were recorded in an ordinary room using a dynamic microphone and an Ampex 601 audio tape recorder (Fig. 7a). Background noises were sometimes present when the recordings were made, such as a soft-drink vending machine, and passers by in the adjacent hallway; the room did not provide total sound isolation, and, on the other hand, the noise was not excessive (about 20 db below the signal most of the time). The word list of Fig. 2 (without POINT and EQUAL) was recited several times in the order given in the list. No special effort was made on the part of the speaker to say the words in a certain way to "help" the recognition process.

FIGURE 7a

FIGURE 7b

FIGURE 7c

FIGURE 7d

FIGURE 7

INFORMATION FLOW IN EXPERIMENTAL SETUP

The tape recorder was then carried to an IBM 1800 computer (Fig. 7b), and connected to the computer through an adjustable band-pass filter. The filter was set so that the low frequency cutoff was 60 cps and the high frequency cutoff was 2500 cps. These figures were chosen to simulate the fact that telephone transmission has a limited bandwidth- about 300 to 3400 cps for an ordinary voice-grade telephone channel. Note that a considerable amount of the distinguishing speech information is lost by not extending the bandwidth to at least 3500 cps [7,8], but for the purposes of experimentation it seemed prudent to choose the conservative figure of 2500 cps. The computer was set so that analog samples would be accepted for one second from the time a trigger signal was given. Samples were taken at the rate of 8000 samples per second, so that a total of 8000 samples were collected for each utterance fed into the machine. The A/D converter produced a signed 8 bit integer for each sample (values ranging from -128 to +127). The tape was positioned manually to a point just before an utterance, and a manual triggering signal was given about the time the tape machine was started. A storage oscilloscope was connected such that its sweep triggered at the same time the computer began accepting samples, and the vertical deflection recorded the voice signal presented to the A/D converter. This monitoring was necessary to guarantee that the whole utterance had been captured in the one second sample, and that the signal voltage was in a

range suitable for the A/D converter. After a number of utterances were placed in disk storage, they were punched out on cards in a format suitable for use with the IBM System/ 360; unfortunately a card deck was, at the time, the only data medium compatible with both the 1800 and the 360.

The cards were then carried to the 360 and transferred to a reel of digital tape suitable for use in subsequent runs on the 360 (Fig. 7c). All of the experimentation with the word-recognition program was then performed on the 360 using this digital tape (Fig. 7d).

The essential difference between the experimental configuration of Fig. 7 and the setup pictured in Fig. 4 is that no frequency analysis of the utterance is provided by the hardware of Fig. 7. During experimentation the frequency information was provided by numerical means (a Fourier analysis program); this approach is desirable since different choices of the filter parameters (different numbers of filters, and different choices of crossover frequencies) can be simulated by changing a few parameters in the Fourier analysis routine, with no hardware change or additional analog data conversion necessary.

B. Description of the word-recognition program.

The configuration of the word-recognition program used for this experimentation is shown in Fig. 8. The program is divided into four main sections, referred to as phases A1,

A2, B, and C. Phase A1 provides the frequency information for the process by performing a Fourier analysis of the digitized samples of each utterance supplied by the digital tape of the utterances (Fig. 7c). The power spectra produced by this analysis are passed to phase B via another tape. Phase A2 also operates on the digitized samples of the utterances in order to locate the principal peak of each utterance and determine its time scaling factor; this information is also passed to phase B. Phase B groups the harmonics supplied by A1 for a given utterance according to the choice of filter crossover frequencies, thereby forming the "raw data" array for the utterance (see II.B.2). This array is then transformed to the "standard array" for the utterance by aligning the principal peak to the center of the standard array, and applying the time scaling, volume compression, and energy normalization (see IIB, sections 2, 3, and 4). Two or three of the standard arrays corresponding to a given word are then averaged to form the "reference array" for that word (see IIC). The output from phase B is then a standard array for each utterance present on the original digital tape of the utterances (Fig. 7c), as well as a reference array for each word of the vocabulary set. This information is passed to phase C via a tape. Finally, phase C prints graphs of the reference arrays and performs the detection function on each of the standard arrays. Each of these phases is now described in detail: a block diagram is given and explained in this section, and

the actual source code for the program is supplied in
Appendix A.

FIGURE 8

BLOCK DIAGRAM OF THE WORD-RECOGNITION PROGRAM

1. Phase A1- The Fourier Analysis Routine.

Fig. 9 is a block diagram for the Fourier analysis routine, phase A1. The one second speech signal is input as 8000 sample points. The number of time slots into which the one second is to be divided is specified (NTM): this number is the same as m in Fig. 5, and the number of sample points to be used for each time slot (NSAMP) is chosen from NTM ($NSAMP = 8000/NTM$). The number of words in the vocabulary set is JNUMUT and the number of times each word in the vocabulary set is uttered is JSETS. Each group of NSAMP samples is to be Fourier analyzed using the Cooley-Tukey fast Fourier analysis routine; this routine makes use of a sine table which is computed at this point. Each set of utterances of the vocabulary set (of which there are a total of JSETS) on the digital tape of utterances consists of the words spoken in the order given in the word list (Fig. 2). This tape is first transferred to disk so that the utterances may be regrouped such that the JSETS occurrences of each word will appear together in the output from this phase. It is desirable for the speaker to go through the entire word list before repeating a given word so that a better sampling of his manner of speaking a given word will be obtained, but, on the other hand, processing is more convenient if the utterances corresponding to the same word are grouped together. Hence the utterances are regrouped in this program. For a given utterance in the input data, a Fourier analysis is made for each of the NTM time slots into

which this utterance is divided, and each analysis produces NSAMP/2 non-dc complex harmonic amplitudes. The power corresponding to each harmonic is obtained from the magnitude of its complex amplitude, and the phase information is discarded. The resulting power spectrum analysis for each time slot of each utterance is then written into the output data set.

A detailed discussion of the Fourier analysis is now presented: A function $f(x)$, $0 \leq x \leq L$, is to be analyzed into a finite Fourier series. The function is made to be periodic, with period L , and the domain of definition is thereby extended to $-\infty < x < +\infty$, as in Fig. 9a. The Fourier series for the infinite series case with continuous independent variable x is

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{\frac{-2\pi i n x}{L}} \quad (1)$$

$$c_n = \frac{1}{L} \int_0^L f(x) e^{\frac{-2\pi i n x}{L}} dx. \quad (2)$$

This becomes the finite sum with the independent variable ranging over the discrete points m by letting

$$x = \frac{mL}{J}, \quad 0 \leq m \leq J-1, \quad m \text{ an integer.} \quad (3)$$

The coefficients are now given by

$$C_n = \frac{1}{J} \sum_{k=0}^{J-1} f_k e^{\frac{-2\pi i n k}{J}}, \quad (4)$$

with the following relations holding:

$$C_n = C_{n+J} \quad (5)$$

$$C_n = C_{J-n}^*, \quad (6)$$

where the star denotes the complex conjugate. Hence there are only distinct coefficients for

$$0 \leq n \leq \frac{J}{2}. \quad (7)$$

It also may be shown that

$$C_n = C_{-n}^*. \quad (8)$$

The finite Fourier sum is then

$$f_m = \sum_{n=-J/2}^{J/2} C_n e^{\frac{2\pi i n m}{J}}. \quad (9)$$

The "response" of a "Fourier filter", as used in this experiment, is now considered. One can imagine testing an analog band-pass filter by applying a pure sine-wave source of variable frequency to the filter, and measuring the response to each frequency at the output of the filter. This scheme motivates the analysis of the response of the Fourier filter, where a certain wave is to be "filtered" by performing the discrete Fourier analysis over the sampled time interval, then adding up the power that corresponds to the harmonics that lie within the "pass band". The testing proceeds by letting $f(x)$ be a sine wave of period χ_0 (see Fig. 9a)

$$f(x) = \sin(2\pi x/\chi_0), \quad (10)$$

$$f_m = \sin\left(\frac{2\pi m L}{J\chi_0}\right). \quad (11)$$

The coefficients that result from Eq. (11) and Eq. (4) (noting that a geometric sum results) are

$$C_n = \frac{\sin \pi \delta}{-2J} \left[\frac{\cos 2\pi\left(\frac{N+\delta}{J} - \frac{\delta}{2}\right) - e^{\frac{2\pi i n}{J}} \cos \pi \delta}{\sin \pi\left(\frac{n+N+\delta}{J}\right) \sin \pi\left(\frac{n-N-\delta}{J}\right)} \right] \quad (12)$$

where N and δ are defined by

$$\frac{L}{x_0} = N + \delta \quad \text{and} \quad (13)$$

$$0 \leq \delta < 1, \quad N \text{ an integer.} \quad (14)$$

N gives the number of the harmonic whose frequency is "nearest" to the left of the frequency of the input signal $f(x)$, and δ gives the fraction that the input differs from the harmonic frequency. Note that the denominator of Eq. (12) is zero if and only if

$$n = N \quad \text{and} \quad \delta = 0. \quad (15)$$

For the case where $\delta = 0$ the numerator is zero, so that

$$C_n = 0, \quad 0 < n \leq \frac{J}{2}, \quad n \neq N, \quad \delta = 0. \quad (16)$$

When $n = N$ Eq. (4) and Eq. (11) may be solved directly to give

$$C_N = \frac{1}{2i}. \quad (17)$$

This result shows that if the input signal is exactly a harmonic frequency of the Fourier analysis then the response is precisely the input signal with no noise in the other harmonics. In the $\delta \neq 0$ case non zero C_n result for all harmonics. To find the response of the Fourier filter for a certain pass band, the pass band was divided into 100 frequencies, and Eq. (12) was used to compute the spectrum for each of the 100 frequencies. These 100 spectra were then summed by adding the 100 values for each harmonic:

$$|C_n|^2 = \frac{1}{100} \sum_{j=1}^{100} |C_n^j|^2 \quad (18)$$

where C_n^j is the response in the n^{th} harmonic for the j^{th} input sine wave. This procedure was applied to two pass bands, one from 100 to 250 cps, and the other from 2050 to 2300 cps. The results (for experimental values of $J=128$, fundamental frequency= 62.5 cps) are presented in Fig. 9b. It is seen that the pass bands resulting from this Fourier analysis technique are reasonable, with the noise (garbage produced by an input signal in the pass band outside of the pass band) about 30 db below the passed signal. It is interesting to compare this situation with that of an analog filter. In the latter case the filter is imperfect by passing some of the input signals not in the pass band, while in the former case the problem is mainly that the analysis of a signal within the pass band produces

non-zero harmonic coefficients outside the pass band. Note also that the analog filter has "memory" from one sampled time interval to the next because of the reactive components used to build the filter, while the Fourier filter has no such memory.

FIGURE 9

PHASE A1

FIGURE 9a

PERIODIC FUNCTION FOR FOURIER ANALYSIS

FIGURE 9b

PASS BANDS OF FOURIER FILTER

2. Phase A2- Location of the principal peak and computation of the time scaling factor.

Phase A2 is shown as a block diagram in Fig. 10, where each utterance on the digital tape of utterances is subjected to the indicated procedure. First a vector V of length m (where m is the number of time slots as before) is constructed where each value $V(j)$ in the vector represents the total power of the signal corresponding to the j th time slot. In the experimental arrangement the total power vector was available directly from the digital data tape; in the configuration of Fig. 4 the power curve is obtained from the raw data array of Fig. 5 by summing columns:

$$V(j) = \sum_i a(i, j).$$

V effectively gives the power as a function of time for the utterance; note that the $V(j) \geq 0$. The maximum and the minimum $V(j)$ (such that $V(j) \neq 0$) are found and these values are used to convert V to the function $\log(\text{power})$ versus time; the equation used is

$$V'(j) = \log(V(j)/V_{\max}) + C,$$

where $C = \log(V_{\max}/V_{\min})$.

The use of the logarithm is suggested by the fact that human hearing perceives loudness on a scale that is approximately

logarithmic. A graph of V (with the logarithmic scaling) for the word "COMMA" is shown in Fig. 11a. Note that this graph is obtained from V' by demanding that the $V'(j)$ be integers such that

$$0 \leq V'(j) \leq 20;$$

this integer-valued function is now used by the algorithm that determines the "position" of the principal peak. The first step in this procedure is to locate the left-most absolute maximum on the graph, which is position 9 on the time axis of Fig. 11a. In this discussion, let X_n denote the point X with label n , and let $X_n.X_m$ represent the line segment joining point X_n to point X_m . The next step is to locate the pairs of points (X_2, X_3) , (X_4, X_5) , ..., (X_{12}, X_{13}) . The midpoint of each line segment $X_2.X_3$, $X_4.X_5$, ..., $X_{12}.X_{13}$, is then found; these midpoints are at positions 9.5, 11.5, 11, 11.5, 11.5, and 11.5, respectively, on the time axis of Fig. 11. The average of these six numbers is then found, and the value furthest from the average (9.5 in this case) is discarded. A new average is computed based on the remaining five midpoint positions, and that value is defined to be the position on the time axis of the principal peak of the utterance. This point on the utterance will subsequently (by phase B) be aligned to the center of the standard array for the utterance. A comparison of Figs. 11a, 11b, and 11c gives a good idea of why this method

works. Regardless of whether the "top" of the peak is "rough" (Fig. 11a), skewed to the left (Fig. 11b), or skewed to the right (Fig. 11c), the averaging process inherent in this algorithm gives the same value for the position of the peak on the time axis.

FIGURE 10

PHASE A2

LOG (POWER)

FIGURE 11a

SAMPLE POWER CURVE FOR "COMMA"

LOG (POWER)

FIGURE 11b

LOG (POWER)

FIGURE 11c

The time scaling factor for an utterance is found in the following way: The integral of the power curve with respect to time is obtained (with the absolute maximum of this curve scaled to 1), giving an energy value for the utterance. During the introductory phase this integral is also obtained for each utterance used to make the reference array for a given word, and the values thus obtained are averaged to give the average energy for the given word. The time scaling factor is taken to be the ratio of the energy of the peak-energy scaled utterance to the average energy for the word.

In order to verify that the algorithms of phase A2 are operating successfully, graphs of the type presented in Fig. 12 are produced during the execution of the phase. The graphs in Fig. 12a and 12c show five utterances of a word superimposed upon one another by the algorithms; this superposition reflects both the determination of the principal peak by the program, as well as the time scaling, and gives a good idea of the success of these efforts. The graphs in Fig. 12b and 12d show the five utterances of the word separately (also lined up by their respective principal peaks, as well as time scaled), so that differences in their features are more apparent.

FIGURE 12b
ALIGNMENT OF "GREATER" BY PHASE A2

FIGURE 12c
ALIGNMENT OF "LET" BY PHASE A2

3. Phase B- Form standard array and reference arrays.

The first operation performed by phase B (see Fig. 13) is to form a raw data array of the form of Fig. 5 for an utterance from the power spectrum supplied by phase A1. In order to do this the dimension n (namely, the number of filters) must be chosen, and the crossover frequencies for the filters must also be chosen. The harmonics presented for a given time slot are simply added into the frequency band corresponding to the filter that would pass that harmonic. The process is repeated for each time slot, resulting in the raw data array of the utterance.

FIGURE 13

PHASE B

The values in the raw data array are now transferred to a new array (which is also of the form given in Fig. 5, and which will become the standard array) with the principal peak of the utterance centered (to the $m/2$ th time slot) and the time scaling factor applied during the transfer. The time scaling is a simple linear interpolation.

In order to apply volume compression the maximum element A_{max} and the minimum element A_{min} (such that $A_{min} \neq 0$) are found in the new array. An exponential volume compression function is then applied to the array according to the formula:

$$a'(i,j) = 1 - \exp(-C * (a(i,j) - A_{min}) / A_{max}),$$

for $i=1,2,\dots,n$, and $j=1,2,\dots,m$.

The constant C is referred to as the volume compression parameter.

The number of different values each point in the new array may assume depends on the number of bits used to represent each point. This is controlled by demanding that the volume compression function map onto the desired number of different values. For example, if each point in the new array is to be specified by 4 bits, then the volume compression function is arranged to map onto only $2^{*4}=16$ different levels.

Finally, the new array is energy normalized and thereby

becomes a standard array for the utterance by choosing b such that

$$\sum_i \sum_j a'(i,j)^2 = 1,$$

where $a'(i,j) = a(i,j)/b$.

Reference arrays are obtained from the standard arrays for a given word by adding two or three of the standard arrays pointwise and performing another energy normalization on the resulting reference array.

The reference arrays and a standard array for each utterance are written into the output data set of this phase, and this data set is the input data for phase C.

4. Phase C- The detection procedure.

Phase C begins by reading all the reference arrays (one for each word in the vocabulary set) into main storage, and graphing each array as it is read in (see Fig. 14); such a graph for the word "COMMA" is given in Fig. 15. The time slots of the array are numbered across the page, and the filters are numbered down the page, with the filter passing the lowest frequencies (filter number 1) being at the top. Each row of the array is graphed versus time, with a constant of proportionality being adjusted so that the maximum element in the entire array produces 10 X's. A

"total power curve" is obtained by summing the columns with the maximum of this curve being adjusted to 20 X's and is given at the bottom of the page (band 99). It should be noted that this curve differs from the power versus time curve of Fig. 11 since volume compression has been applied. The graph shown is for 16 filters and 64 time slots, with the filter crossover frequencies from the first column of Fig. 17 (see III.C). These parameters are, of course, adjustable.

FIGURE 14

PHASE C

FIGURE 15
PATTERN FOR "COMMA"

Each of the reference arrays is compared with each of the other reference arrays by applying the detection criterion (SDE). This information shows how "close together" the different reference arrays are, and gives a good indication of where possible errors might result in the subsequent detection of an unknown utterance.

The standard array for each utterance is then read and is compared with every member of the reference array set. The resulting SDE values are printed and the utterance is identified with the reference array giving the smallest SDE. The standard arrays may also be graphed (as in Fig. 15) if desired.

C. Experimental results.

The word recognition program developed in this research depends on several parameters that have a strong bearing on the practical usefulness of this speech recognition technique for a system such as Audio-CUPL. This usefulness is gauged by two key measures: the performance of the system, and the resources required by the system. The key parameters may be adjusted to "tune" the word recognition program to give the "best" performance for the "least" demand on the computer resources. The choice of certain of these parameters definitely involves a compromise between performance and the use of resources: better performance can be obtained at the expense of requiring more computer

resources. However, in certain other parameters it is found that the choice may be made to increase performance with either no change, or a decrease, in the resources required. In this section the performance criteria are considered, and the experimental results showing the relation between performance and the values of the key parameters are presented.

Several algorithms for handling the identification of an unknown utterance can be imagined. The simplest scheme is to find the reference array that is "closest" (smallest SDE) to the standard array for the utterance, and then simply identify the utterance with the word represented by that reference array. A more complicated procedure could be devised such that this identification will be made if and only if one or more other conditions are satisfied, and, if the additional conditions are not satisfied, then the utterance is rejected and the system requests a repeat from the user. The performance of the system should, in any case, be taken as inversely proportional to the inconvenience caused the user by errors in identification or requests for repetition: that system performs best which requires the least repetition from the user, either because the system incorrectly identified an utterance or because the system rejected the utterance as not well enough matched to any of the reference arrays. Two criteria were considered that would enable the system to reject "bad" utterances. One of these was to find the two reference

arrays closest to the unknown's array and then to require that

$$|SDE1 - SDE2| > D,$$

where SDE1 is the distance between the unknown and the closest reference array, D is a constant, and SDE2 is the distance between the unknown and the second closest reference array. The idea here is that, if the closest reference array wins by only a small margin, then a small perturbation to the system or the manner of speaking could have resulted in a different identification. It was found that using the simplest identification scheme resulted in 4 "errors" out of 111 unknown utterances with the "best" system tuning. Using the criterion to reject unknowns with close runners up resulted in no "errors" (identifications that were incorrect), but resulted in 40 rejected utterances! The inconvenience to the user would therefore be ten times as great. The fact that the user is on-line, and immediately hears how the system interpreted his utterances, makes an "error" by the system of relatively minor importance since the user can correct it before it can cause a chain reaction of bad effects. During the introductory phase, however, the main objective is to obtain reliable reference arrays, and at this time it is desirable to use a more stringent criterion, such as the one suggested above, even if the rejection rate is high. It is therefore

concluded that the number of errors with the simplest identification scheme is the best measure of system performance from the user's point of view. It should also be noted that the system tuned to lower the number of errors seems to also have the best separation of reference arrays. Data on system performance was provided in graphical form by phase C, and is shown in Fig. 16. These graphs were helpful in evaluating different choices of the key parameters as well as different identification criteria. Each of the 185 utterances in the data is represented by a horizontal line in the figure, giving a plot of the SDE's to the three reference arrays nearest to the utterance; the horizontal scale is .01 SDE units per print position. An error (the closest reference array does not correspond to the utterance) is plotted using the *, and an attempt to plot more than one character at a particular print position is indicated by a +. The integer at the extreme right of each line identifies the utterance; note that the five utterances of each word are grouped together. The three integers just to the right of the graph respectively identify the three plotted reference array SDE's to the utterance. An error is signaled by disagreement between the first of these and the integer identifying the utterance. The floating point number gives the difference in the SDE's between the closest reference array and the runner-up.

The key parameters of the word recognition program are the dimensions of the standard arrays and the reference

arrays n and m (Fig. 5), the number of bits b used to represent each value in an array, the choice of the crossover frequencies for the bandpass filters, and the choice of the volume compression parameter C (see III.B.3). The experimental tuning of the system through choices of these parameters is now considered. The runs described here were all made using the same original digital tape of utterances (III.A) which was made with the word list of Fig. 2, less the words POINT and EQUAL- a 37 word vocabulary set. Five utterances of each word were recorded, giving 185 utterances in all. The results did not seem to change significantly whether two or three utterances of a word were used to construct that word's reference array, so only two were used. Subsequent detection of "unknowns" therefore involved three completely fresh utterances of each word- 111 utterances in all.

Runs were made for the number of filters equal to 16 and 8 ($n=16$ and $n=8$), and each of these cases was run with the number of time slots equal to 64 and 32 ($m=64$ and $m=32$). The crossover frequencies for these two choices of n are given in Fig. 17; these choices were made from Fig. 18 of [9], where the "articulation index" is plotted against frequency. The bands of Fig. 17 are chosen so that each filter contributes an equal amount to the total articulation possible for a 3000 cps bandwidth. An arbitrary choice of crossover frequencies for the 16 filter case was also tested; these values are also shown in Fig. 17. The choice

based on the articulation index was found to be significantly better than the arbitrary choice (see Fig. 18).

FIGURE 16

GRAPH OF EXPERIMENTAL RESULTS OF PATTERN MATCHING

FILTER #	<u>16 FILTERS</u>		<u>8 FILTERS</u>
	FROM [9]	ARBITRARY	FROM [9]
1	100	100	250
2	250	200	550
3	380	300	900
4	510	400	1300
5	700	500	1750
6	850	600	2300
7	1100	700	3000
8	1250	800	4100
9	1500	900	
10	1700	1000	
11	1900	1100	
12	2050	1250	
13	2300	1400	
14	2500	2000	
15	3000	2500	
16	4100	4100	

FIGURE 17
CHOICES OF CROSSOVER FREQUENCIES FOR FILTERS

n	m	b	C	THEORETICAL CROSSOVER FREQUENCIES?	NUMBER OF ERRORS IN 111 UTTERANCES
16	64	16	100	NO	8
16	64	16	100	YES	4
16	64	16	500	YES	7
16	64	16	50	YES	5
16	64	4	100	YES	4
16	32	4	100	YES	12
8	64	4	100	YES	7
8	32	4	100	YES	16

FIGURE 18
ERROR RATES FOR VARIOUS CHOICES OF KEY PARAMETERS

The performance of the system depended less critically on the choice of the volume compression parameter C , with the best C being about 100 (Fig. 18). C is related to the dynamic range in the loudness of speech, with $C=100$ corresponding to a range of about 20 decibels between the softest (non-noise) portions of the utterance and the loudest portions. It should be mentioned that attempts were made to use a C dependent on i in the volume compression formula (see III.B.3)- that is, the compression applied was "weighted according to the filter". This produced no substantial improvement in system performance over the $C=100$ case. It is believed that this result is due to the fact that choosing the crossover frequencies from the articulation index has already "weighted" the filters in a good way.

The key consideration of the storage requirements for the reference arrays is determined by n , m , and b . In fact, the number of bits per reference array is $n*m*b$ bits. It was found that no degradation of system performance was experienced in going from 16 bits per point to 4 bits per point (see Fig. 18)! The number of time slots m was halved resulting in a substantial increase in the error rate (from $4/111 = 3.6\%$ to $12/111 = 10.8\%$). Similarly, the number of filters was cut to 8, increasing the error rate from 3.6% to 6.3%; note that the decrease in performance here is not nearly as great as that experienced by cutting the number of time slots. When both the number of filters and the number

of time slots are cut to 8 and 32 respectively, the error rate rose to 14.4%.

D. Timing considerations.

The amount of the central computer's instruction performing ability that is required by the word recognition program is an important consideration in evaluating the resources required by a system such as Audio-CUPL. The main uses of CPU power are considered in this section.

It was observed that the Fourier analysis routine required about 2 seconds per utterance in the experimental arrangement. While this figure could probably be trimmed down considerably by a careful hand coding of the program, it still must be concluded that a large resource (CPU power) is saved by using the analog filter arrangement proposed in II.A.

Computing the SDE between the standard array of an utterance and one of the reference arrays requires the execution of a loop $n*m$ times. If the arrays are half-word fixed point numbers, and the computer is an IBM 360/65, then the loop (written in 360 assembly language) could be:

	LA	R1,NUM	N*M=# TIMES LOOP EXECUTED
	LA	R2,PA	POINTER TO ONE ARRAY
	LA	R3,PB	AND TO OTHER ARRAY
	LA	R4,A	HEAD OF LOOP
	SR	R0,R0	ACCUMULATOR = SDE
A	LH	R7,0(R2)	

SH	R7,0(R3)
MR	R6,R7
AR	R0,R7
LA	R2,4(R2)
LA	R3,4(R3)
BCTR	R1,R4

This takes about $10.5 \cdot n \cdot m$ microseconds. If single precision floating point arithmetic is used the time is $12.5 \cdot n \cdot m$ microseconds, an increase of about 8.6%. Storing the reference array numbers in a 4 bit format will incur additional overhead in this loop, since the 4 bit numbers must first be expanded to a standard format for the 360 (see II.D). The times would be about $22 \cdot n \cdot m$ microseconds with floating point arithmetic, or about $20 \cdot n \cdot m$ microseconds with fixed point arithmetic. If $n=16$ and $m=64$, then the computation of an SDE would take about 1 millisecond if the array is stored in standard number format, or about 2 milliseconds in the 4 bit format. If $n=8$ and $m=64$ then these times would, of course, be cut in half.

If a given unknown is compared with NW different reference arrays then the time for an identification would be $NW \cdot 2$ milliseconds for the $n=8$, $m=64$ case. For example, if the vocabulary set contained 50 words, and an unknown were compared with the entire set, an identification would take 100 milliseconds. The use of features to restrict this search (II.E) would effect a substantial saving of CPU power.

CHAPTER IV

CONCLUSIONS

It is apparent from III.C and III.D that specifying n , m , and b for a production system presents the usual dilemma faced by a systems engineer in designing or tuning any large system. He can have the best performance (one error in about 30 utterances for $n=16$, $m=64$, $b=16$), or use $1/8$ of the core storage resource and the same CPU power by degrading the performance by a factor of two (for the $n=8$, $m=64$, $b=4$ case). Another factor of two may be gained in core use as well as use of CPU power at the expense of again degrading the performance by a factor of two ($n=8$, $m=32$, $b=4$); here there would be one error in about seven utterances! The best compromise if core storage is at a premium (and it always seems to be) might be the $n=8$, $m=64$, $b=4$ case, with about one error in fifteen. This arrangement would require 256 bytes of storage per reference array, or 12800 bytes for a vocabulary set of 50 words. Recognition time would be about 50 milliseconds per unknown if only half of the reference arrays are matched with the unknown. One IBM 2316 disk storage pack would hold the reference arrays for over two thousand users. Degrading performance to the point where an error occurred every seven utterances would seem to render the system fairly useless. However, it seems

reasonable to expect that the additional work that would go into the development of a production system would improve the figures quoted here considerably, probably halving the error rate for a given choice of m , n , and b .

The implications of these results in the hardware design of Fig. 4 will now be considered with the IBM System/360 in mind. Connection to the multiplexor channel could be made through the 2701 Data Adapter Unit, using the "parallel data adapter". This device would accept the 8 bit integers from the A/D converter and transmit them to core storage via the multiplexor channel. The 2701 can handle up to seven A/D converters, but in a configuration intended for time-sharing the system with several users it would be less expensive to use a single high-speed A/D converter and connect the several users through the multiplexor.

The multiplexor- A/D arrangement must handle $n*m$ samples per second (512 in the $n=8$, $m=64$ case), and the multiplexor should be able to handle m sweeps of the filter bank per second. There are many multiplexor- A/D components available commercially; for this discussion those available with the IBM 1800 Data Acquisition and Control System will be considered. The relay multiplexor has a maximum speed of 100 points per second, which is too slow for the n , m under consideration. The solid-state multiplexor and A/D together have a capacity of 20000 conversions per second, so that $20000/512 = 39$ users could be handled with one multiplexor-A/D combination. This equipment would cost roughly \$2500 to

\$5000, or roughly \$100 per channel per month, so that up to 39 users could be on-line at a given time. During program input (compile) the CUPL compiler uses about $60/40000 = 6.6$ milliseconds per statement. Supposing that about 10 seconds are required to input a statement, it is seen that the compiler use of CPU power is small compared to the time required by the word recognition program. The latter uses at worst about .1 second per utterance, which takes at least one second to input. At worst, then, 10 users could be handled by the model 65 CPU, but the use of features to limit the SDE calculations, and a slower speaking rate than one utterance per second, should make 40 users at a time a realistic figure. If the reference arrays for all users were kept in main storage 40 users would require 512K of core storage for the reference arrays; this however, would be a waste of money. The fact that at least one second is required to input each utterance for a given user would permit storing the user's reference arrays on a drum, and fetching them into core only when the actual detection is to be performed. Thus space in core would probably be required for only a few sets of reference arrays at a time, which is a modest main storage requirement. The use of drum storage in this way might make it more economical (in use of CPU power) to use a larger b simply to have the arrays in a standard number format for the SDE calculations.

It is concluded, therefore, that at this time a system such as the proposed Audio- CUPL system is feasible both

economically and logically. The outlay for the analog hardware is not large, so that a prototype system could be operated for a few hours per day without undue cost.

APPENDIX A

LISTING OF SOURCE PROGRAMS

(This appendix is available in the Original THESIS in the)

Cornell University Library

Ithaca, N. Y. 14850

REFERENCES

1. For an extensive bibliography, see IEEE Spectrum. March, 1965.
2. Conway, R. W., and Morgan, H. L., Tele- CUPL: A Telephone Time Sharing System. Comm. ACM 10, 9 (Sept., 1967).
3. The University of Michigan "Desk Calculator" System.
4. Gold, B., Word- Recognition Computer Program. Technical Report 452, Research Lab. of Electronics, M.I.T., June, 1966.
- Bobrow, D. G., and Klatt, D. H., A Limited Speech Recognition System. Proc. of the Fall Joint Computer Conference, AFIPS, 1968.
5. Walker, R. J., CUPL- The Cornell University Programming Language. Cornell University, Ithaca, N. Y., Dec., 1966.
6. Denes, P., and Mathews, M. V., Spoken Digit Recognition Using Time- Frequency Pattern Matching. Journal of the Acoustical Society of America, Vol 32, Nov., 1960.
7. Fletcher, H., Speech and Hearing. Van Nostrand, New York, 1929, p281.
8. Potter, R., Kopp, G. A., and Kopp, H. G., Visible Speech. Dover Publications, New York, 1966, p12, Fig. 6.
9. Stevens, S. S., editor, Handbook of Experimental Psychology. Wiley, New York, 1951, Chapter 26 by Licklider, J. C. R., and Miller, G. A., pp1040- 1074.