Speech recognition with the Apple macintosh

James Ashton
University of Wollongong, uow@ashton.edu.au

Speech Recognition with the Apple Macintosh

A Report on an Honours Project
by

James Ashton

Department of Computing Science
University of Wollongong

November 1985

Abstract

In 1985 the Computing Science Department at Wollongong University adopted speech processing as a major area of research effort. Two honours projects: one involving speech production by computer; and this one involving speech recognition by computer were undertaken. This document describes the theory, goals, development and achievements of the project. Briefly, the project aims to produce a usable system by which spoken words can be automatically translated into written text. While this project does not attempt to be too ambitious due to the time constraints placed on it, it is hoped that it will provide useful experience in a new and increasingly important area of computer science.

0 Contents

1 Speech Recognition Techniques

Speech recognition by computer is a comparatively new concept and one which is yet to attain full workability. Many systems which perform the task to some limited extent exist today but in no case do they approach the human level of competence. The main reasons behind this are the immense amount of computing power required and a lack of understanding of the human brain's manipulation of language.

Recognition systems may be divided into four broad categories based on two criteria: whether they can recognise continuous speech or not; and whether or not they can recognise the speech of more than one speaker or not. Most systems are also limited in the vocabulary of words which they will recognise. Until recently systems of any kind were both rare and expensive, but advances in signal processing techniques and in V.L.S.I. integrated circuit technology have made them increasingly available and powerful.

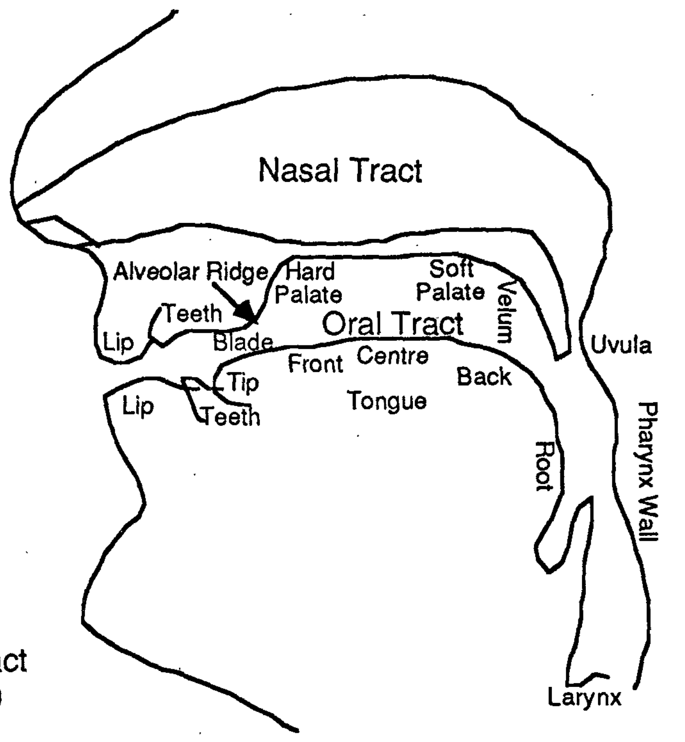
1.0 Speech and Sound

Speech is the sound produced by a complex arrangement of organs in the human body as shown in Diagram 1.0. Much of the sound is produced by the vocal chords. These produce a periodic waveform which, when it passes up through the mouth and nose is modified by the shape of those organs. These modifications cause several frequencies, known as *formant frequencies* to be amplified. Changing the shape of the mouth by moving the tongue, teeth and etc., and blocking the nasal passages using the *velum* causes the formant frequencies to change to give the different parts of speech their characteristic sounds as perceived by the ear. Sounds can also be produced away from the vocal chords, for example the sounds usually associated with the letters 's', 'f' and others produce sound as air rushes past parts of the mouth. In fact speech sounds can be broadly grouped into three classes: *vowels*, *nasals* and *fricatives* depending on their method of production.

Each of the different sounds made by the speech mechanism is classified as an *allophone*. Each language (and dialect) has it's own set of allophones which are used. Each allophone corresponds to a *phoneme*. Often several allophones correspond to the same phoneme but not vice versa. Phonemes are psychological entities used by the brain and all allophones corresponding to the same phoneme sound the same to the listener despite their differing sound features. The human ear can distinguish the various phonemes of a language by making use of the unique features of the waveforms of the allophones involved. Computer recognition of speech also must depend on detecting these same features.

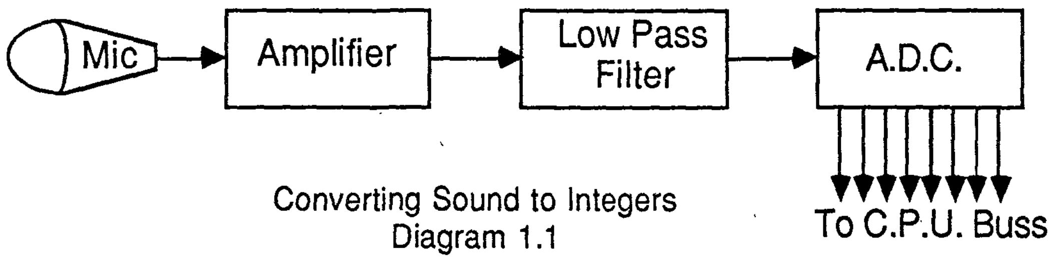
1.1 Signal Processing

A first step in computer speech recognition is to introduce sound into the computer in the form of numbers. This is done using a microphone connected to interface circuitry as illustrated in diagram 1.1.



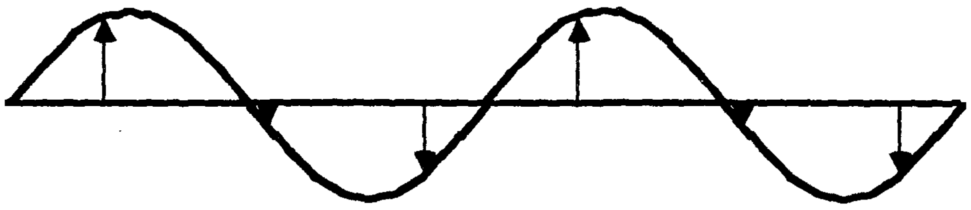
The Vocal Tract
Diagram 1.0

The signal from the microphone is amplified and filtered before being fed to an analogue to digital converter (or A.D.C.) which produces integers in proportion to the voltage applied and makes these integers available to digital devices for further processing. A new integer is produced at frequent intervals, capturing the shape of the waveform from the microphone. The low pass filter used is required due a sampling theorem requirement which specifies that the waveform of a signal be sampled at at least twice the highest frequency present in the signal to prevent *aliasing* and to accurately represent it. Diagram 1.2 illustrates sampling and aliasing.

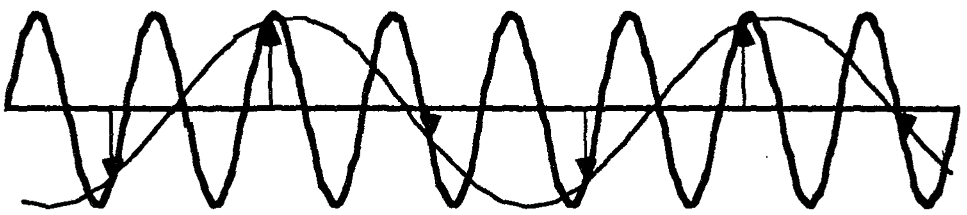


Converting Sound to Integers
Diagram 1.1

Once a speech waveform is available to the computer as numbers, it must be processed to extract features useful for speech recognition. A large number of techniques for achieving this are in use. Before discussing these it is useful to give the notation used when manipulating the stream of integers which arrive from the microphone at fixed intervals. An integer function x(n) is defined such that at instant n, the signal level is represented by x(n).



Discrete Signal Sampling



Aliasing Occuring Due to High Frequencies

Diagram 1.2

1.1.0 Zero Crossing Rate

This is a simplistic method which nevertheless has enjoyed some success in the past. All that is involved is counting the rate at which the signal (i.e. the stream of integers) changes from positive to negative. The number of zero crossings in a signal N samples long is given by:

$$\sum_{n=0}^{N} \left| \mathrm{sgn}[x(n)] - \mathrm{sgn}[x(n-1)] \right|$$

where

$$\mathrm{sgn}(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

Eq 1.0

1.1.1 Autocorrelation

The autocorrelation method utilises a vector R(n) for each N sample signal defined as follows:

$$R(n) = \sum_{k=0}^{N} x(k).x(k+n), \quad n = 0, 1, 2, \ldots N$$

Eq 1.1

By evaluating R(n) and finding it's maximum values, the fundamental frequency of a signal can be found, along with other useful information. The autocorrelation vector, while providing useful information in itself, is most useful as a preparatory step in other, more involved techniques, such as pitch synchronous analysis and L.P.C. analysis.

1.1.2 Homomorphic Analysis

This is a powerful technique which requires a great deal of computing power to implement. Through the use of *Fourier transforms*, signals can be converted from the *time domain* to the *frequency domain* and back again. The function x(n) is in the time domain because it's parameter n varies with time. Frequency domain functions vary with frequency and the frequency domain function corresponding to x(n) is denoted X(k). The discrete Fourier transform and it's inverse are given below.

$$X(k) = \sum_{n=0}^{N-1} x(n).e^{-2\pi jkn/N}$$

Eq 1.2

$$x(n) = \left\{\sum_{k=0}^{N-1} X(k).e^{2\pi jkn/N}\right\}/N$$

Eq 1.3

The steps involved in homomorphic analysis are shown in diagram 1.3. Conversion to the frequency domain (or *spectrum*) and thence to the log magnitude spectrum produces a graph with many peaks, the main peaks or formants often being obscured. To smooth the graph and to remove the signal resulting from the vocal chords (leaving only signal resulting from the influence of the vocal tract), the log magnitude spectrum is effectively low pass filtered. This is done using an inverse Fourier transform to produce what is known as a *cepstrum*, which then has it's high frequency component zeroed. When a Fourier transform is done again, the log magnitude is restored but smoothed so that only major features such as formants are visible. The positions and sizes of these features can be analysed to discover what allophones of speech resulted in the original signal.

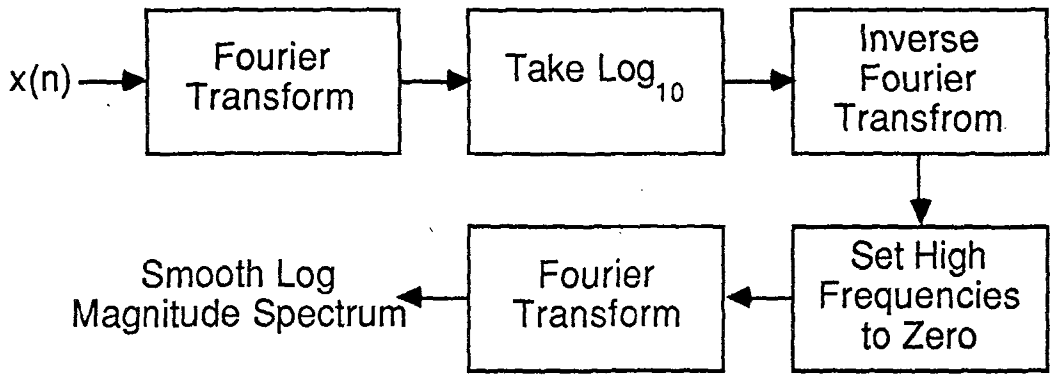

x(n)
Fourier Transform
Take Log10
Inverse Fourier Transfrom
Set High Frequencies to Zero
Fourier Transform
Smooth Log Magnitude Spectrum

Diagram 1.3

1.1.3 Linear Predictive Coding

Due to the large amount of computer time required to do homomorphic analysis, the L.P.C. model was developed which allows the computation of a smooth spectrum with fewer calculations. L.P.C. coding assumes that the frequency domain function of a signal can be represented by the reciprocal of a polynomial and uses further assumptions in calculating the polynomial's coefficients in order to reduce the computational expense. One method makes use of the correlation vector.

L.P.C. analysis begins by treating the vocal tract as a filter and then by assuming the *transfer function* (or spectrum) of the filter is represented by the reciprocal of the polynomial as mentioned. The input or *excitation* signal, the output signal and the filter function are denoted in the frequency domain by E(k), S(k) and A(k) respectively. The following equation gives the relationship between these functions.

$$E(k) = S(k).A(k) \qquad \text{Eq 1.4}$$

In the time domain these same three functions are denoted as e(n), s(n) and a(n) respectively and the time domain relationship between them follows.

$$e(n) = \sum_{i=0}^{M} a(i)s(n-i) \qquad \text{Eq 1.5}$$

To find the coefficients a(n) it is necessary to solve the following equation:

$$\begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(M-1) \\ R(1) & R(0) & R(1) & \cdots & R(M-2) \\ R(2) & R(1) & R(0) & \cdots & R(M-3) \\ \vdots & \vdots & \vdots & & \vdots \\ R(M-1) & R(M-2) & R(M-3) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ \vdots \\ a(M) \end{bmatrix} = - \begin{bmatrix} R(1) \\ R(2) \\ R(3) \\ \vdots \\ R(M) \end{bmatrix} \qquad \text{Eq 1.6}$$

The vector R(n) in this equation is the autocorrelation vector of s(n). The matrix is symmetric and it has equal diagonals. Such matrices are said to be Toeplitz matrices and equations involving them are open to solution using efficient techniques. Once the a(n) coefficients have been found, the polynomial can be evaluated to give the spectrum of the voice signal. The constant M in the equation gives the order of the polynomial. Since a polynomial of limited order is involved (usually $M \approx 12$) a smooth curve is guaranteed. Features of this spectrum can be treated in the same way as in homomorphic analysis to recognise speech.

1.2 Recognising Speech from Signal Features

As has been described in the previous section, various parameters can be extracted from signals arriving from a microphone. These parameters give much more useful information than the large amount of raw data contained in the original signal but it remains to convert them into a useful representation of the speech. Such a representation is often the *grapheme* form of a language, i.e. a sequence of symbols readable by people. There are two levels at which this can be done.

1.2.0 Word or Phrase Matching

The simplest technique for producing text from speech is to store the parameter sets of all the words or phrases as spoken by all the people who need to use the system. This involves a training period where each user speaks the required words. Parameter sets usually arrive at intervals of around 30ms so that there are many sets for each word. An algorithm for detecting the beginning and end of each utterance is required to split the incoming parameter sets into separate words. In continuous recognition systems, this is done by advanced techniques which require a knowledge of the language's syntax and semantics. In discrete utterance systems, the user artificially introduces short periods of silence between each word which can be detected easily by the system. New parameter sets can then be compared with stored sets and the best match used to provide the text of the spoken words.

A complication to this system is that words are spoken at different speeds at different times so that, for example, at one utterance the first syllable of a word may take 30% of the total time of the utterance, and at another utterance 50%. If this occurs, the sets of parameters which arrive will be mismatched. This can be allowed for using a technique known as *dynamic time warping*. This attempts to stretch and shrink the word being matched at each comparison to obtain the best possible match in each case.

The first step in dynamic time warping is to compare all the parameter sets of the two words against each other and obtain a single number representing the difference between each parameter set. If the stored word contains ten parameter sets and the new utterance has resulted in twelve, one hundred and twenty comparisons will be required. The numbers resulting from each comparison can be arranged in a matrix D(n, m), where, for example, element D(1, 3) is the number resulting from the comparison of the first and third parameter sets of the new and stored words respectively. A new matrix T(n, m) is then generated as defined by the equation below.

$$T(1, 1) = D(1, 1)$$
$$T(n+1, m+1) = D(n+1, m+1) + \mathrm{Max}[T(n, m), T(n, m+1), T(n+1, m)] \qquad \text{Eq 1.7}$$

When T(n, m) has been fully calculated from D(n, m), T(N, M) will contain the minimum difference measure between the two words. Obviously considerable calculation is involved to compare two parameter sets. If a large vocabulary of stored words is present, the time required to find a good match will be prohibitive. This is the major limitation of systems which employ this method.

1.2.1 Allophone Matching

A more advanced technique for recognition operates at the allophone level, attempting to recognise individual allophones from the sequence of parameter sets resulting from the input speech. This is done by employing an internal list of the properties of each allophone in terms of the parameter set evaluated by the system. Various algorithms can be used to give a probability that a spoken allophone matches any of the stored allophone properties. These probabilities are then combined with a knowledge of language syntax and grammar to make a final choice of the allophones (and therefore of the phonemes) spoken and to split utterances into words and sentences. This method is very difficult to implement and few systems use it in full generality successfully.

2 Project Aims

2.0 Overall Aims

The overall aim of this project was to produce a system to experiment with voice recognition by computer. The implementation was to run on an Apple Macintosh computer utilising all of the user friendly features of it's environment and hopefully benefiting from the power of it's C.P.U.. A major point was that the system would not attempt to attain a 100% success rate but instead would allow the user to select from a list of best matches after each word was spoken. The Macintosh's mouse seemed to provide the easiest mechanism for making this choice. Some external hardware was required to allow the Macintosh to detect sounds and to handle some of the labours of digital signal processing.

The Macintosh is a self contained computer with very limited access to it's internal circuitry. For this reason the additional hardware would have to be intelligent enough to communicate via a serial link to the Macintosh. Also, it should be an easy system to set up so that anyone could use it.

2.1 Equipment Sources

Fortunately a ready built circuit board for the Apple IIe computer was available for use in the project. Called the Lis'ner 1000 it was designed by Steve Ciarcia for a Byte Magazine article and manufactured by Micromint, an American electronics company. Apart from the circuit board, the unit came supplied with working demonstration software. This software performed recognition of up to thirty two words spoken by a single speaker as discrete utterances, i.e. with the words separated by short periods of silence. Tests I conducted indicated that 90% accuracy was attainable with care. The text of the words could be changed by the user to any character sequence and sets of words could be saved to disk after training.

The hardware provided was designed in such a way that significant variation from the recognition method used by the supplied software was not easily possible. My project aims were therefore limited to the modification of this software so that it passed information on to the Macintosh. This reduced the system development time but also impaired my ability to experiment with other speech recognition techniques and to fully optimise the system.

2.2 System Generality

Initially it was intended to improve system performance by using it to recognise the Pascal computer language. This goal seemed to provide a vocabulary of acceptable size and a grammar which, unlike natural language grammars, was well defined and understood. The aim was to use the grammar to limit the search of the stored vocabulary of trained words based on the current context, so speeding operation. An alternative performance increase could have been in accuracy, by searching the entire vocabulary list and using the context information to bias the selection of words.

Eventually this idea was discarded for two reasons. Firstly, even simple context sensitivity requires considerably more programme intelligence and the grammar of Pascal is non trivial. There would not have been time to implement Pascal context sensitivity for this reason. Secondly, a system tuned to the grammar of Pascal would be much less general and therefore less useful than a fully general system.

2.3 General System Operation

The system envisaged was to have the ability to train any list of words. Operating using two modes, it could change between a training and an editing mode.

2.3.0 The Training Mode

In the training mode, the user would type several characters of text and speak an utterance several times. That utterance would then be associated with the typed text. The ability to save the (possibly very lengthy) list of trained words and associated text should be present.

2.3.1 The Editing Mode

In the editing mode the system would operate similarly to other Macintosh text editors with the addition of speech input features. On the detection of an utterance, the system would display a Macintosh window containing a list of the text associated with the best matches in order of the closeness of the matches. A cutoff point would prevent too many words appearing in the window. Clicking on the a word in the list would cause it to be inserted at the current insertion point in the text. If a no word was selected before more text was typed or a further utterance was detected, the best match in the list would be inserted automatically. This last feature would allow 'hands free' operation provided that the best matches were correct matches.

2.4 Standardisation

The system, being a Macintosh application, should comply as closely as possible with the guidelines specified by Apple for such applications. Details about these guidelines are contained in the Macintosh documentation for software developers.

3 Hardware Configuration

The hardware used in the system consisted of an Apple IIe fitted with two disk drives, a serial communications card and a special speech processing card, along with an Apple Macintosh. The Apple IIe software was modified and developed on the IIe itself but development for the Macintosh was eventually done with the help of an Apple Lisa with an internal ten megabyte hard disk.

3.0 The Apple IIe

The Apple IIe used was a 48K machine with four of it's expansion slots in use. Slot six contained the disk drive controllers and the auxiliary connector contained the R.G.B. colour driver used to run the colour monitor connected to the machine. Slot one contained a serial card capable of communicating in the RS232C format at speeds up to 19.2K baud. The various speeds were selectable by changing D.I.P. switch settings. Finally, slot four contained the specialised speech processing card called the Lis'ner 1000.

3.1 The Micromint Lis'ner 1000

This card for the IIe contains a custom designed V.L.S.I. chip to perform L.P.C. analysis and synthesis and support circuitry. Only the analysis modes of the chip were used for this project. The chip, the SP1000 produced by General Instruments, performs the L.P.C. analysis using a modified form of the Autocorrelation technique discussed in section 1.1.3. Block and pin-out diagrams of the chip are given in diagram 3.0.

The modifications allow it to function in real time using only 300 bits of working storage. A consequence of the small size of the chip is that it operates at the lower limit of acceptable performance. The polynomial used is of the eighth order only, i.e. $M = 8$ in Eq 1.5. Also, all the mathematics done uses nine bit integers. The A.D.C. used on the card produces eight bit values although a further three bits of accuracy is attained through the use of an automatic gain control system implemented on the SP1000. Diagram 3.1 shows the configuration of the card.

The first stage of the system, the high pass filter, removes signals below 250Hz, to eliminate unwanted low frequency noise from the system. Next follows the amplifier stage. The automatic gain control introduces a further level of complexity to this stage. The gain of the amplifier is variable over eight different levels spaced at six decibel intervals. The gain is controlled by three signals from the SP1000. As the gain changes suddenly, high frequency fluctuations are created in the signal. The anti aliasing, low pass filter stage which follows the amplifier does double duty in removing these fluctuations. The filter attenuates sounds with a frequency greater than 3200Hz. Both filters used are two pole filters only so their cutoff is not particularly sharp but is adequate for the purposes of the system. The A.D.C. used produces eight bit values which are passed to the SP1000 serially, i.e. one bit at a time, thus reducing the number of pins required on the chips.

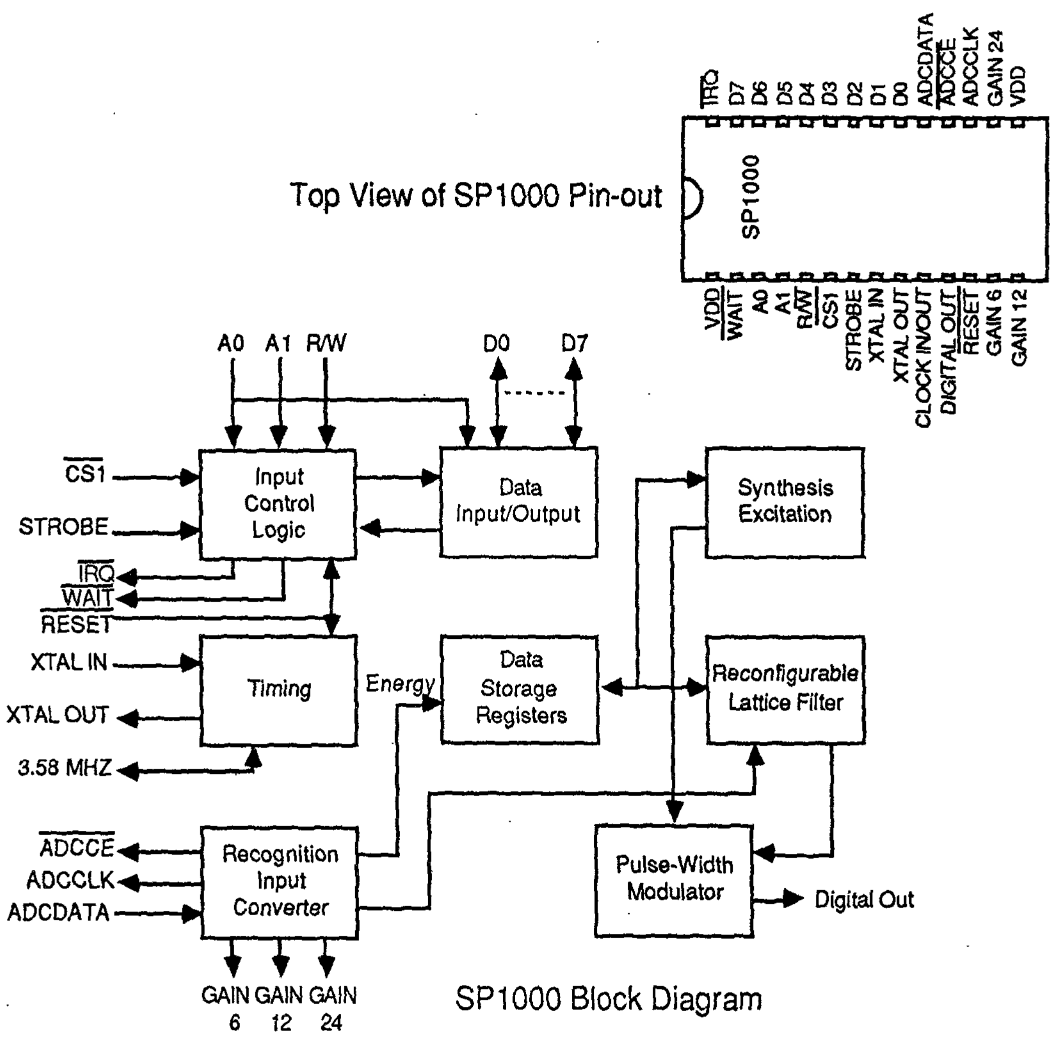


Diagram 3.0

The SP1000, after performing it's analysis, passes sets of nine eight bit integers to the data buss of the Apple IIe. The nine integers are the eight L.P.C. coefficients and an energy parameter. The energy parameter essentially gives the volume of the sound detected. The mechanism by which the Apple IIe's 6502 C.P.U. accesses the parameters is somewhat long winded due to the simplified internal design of the SP1000 chip. The parameter set used internally is contained in a rotating shift register for ease of calculation with the L.P.C. analysis method used. At any time the C.P.U. can store a parameter number (in the range zero to eight) to a parameter address register. It must then poll a status register until a busy flag is cleared, indicating that the parameter output data register contains the value of the requested parameter. The chip can also interrupt the processor each time it completes an analysis operation. Diagram 3.2 shows the register set of the SP1000.

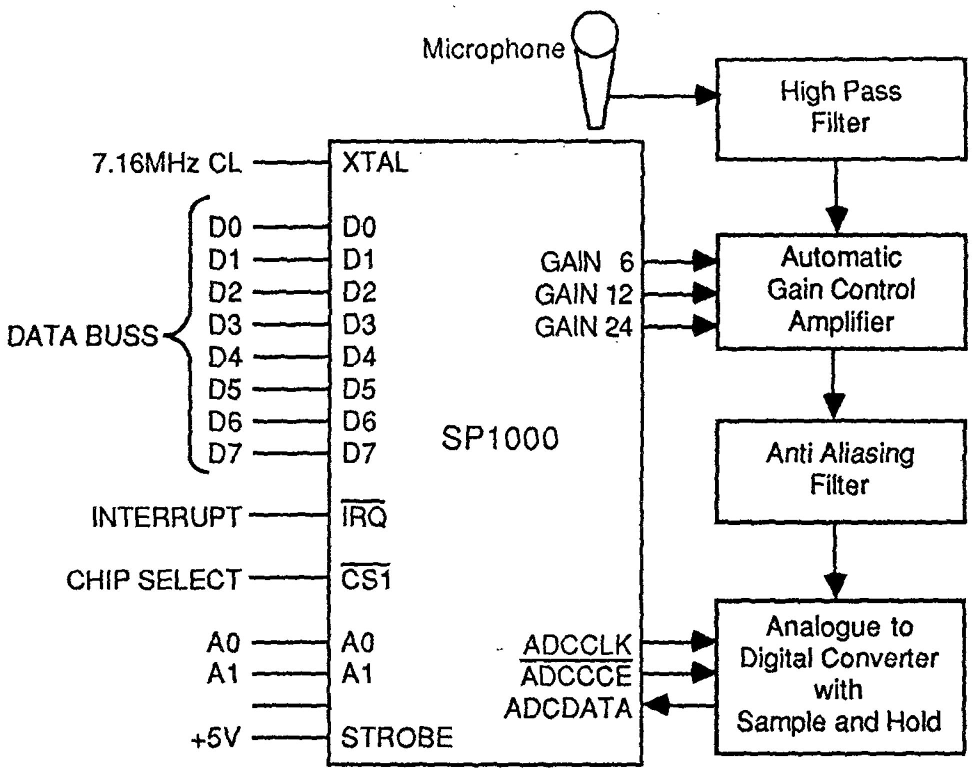


Diagram 3.1

The control register of the SP1000, when written to, sets various internal parameters which control the chips mode of operation. The status register, when read from, gives information about the chip's current status. Additional parameters exist within the chip, having parameter numbers greater than eight. Three of these are timer counters which control the frequency at which the A.D.C. is sampled and at which analysis operations are performed. The fixed frequency of the anti aliasing filter means that the sampling frequency must similarly remain fixed at about 6.25kHz. In fact the SP1000 can handle sampling rates of between 5kHz and 16kHz. Analysis is set up to be performed on every 128 data samples, resulting in a set of nine parameters every 20ms or at a rate of 50Hz. Thus the Apple IIe must handle a data rate of 50Hz × 9 bytes or 450 bytes per second.

3.2 Serial Communications

While further processing of data obtained from the SP1000 could be done by the Apple IIe, a considerable percentage of the 450 bytes per second, or about 4800 baud data rate, would require transmission to the Macintosh. Since both the serial card and the Macintosh were capable of communication at 19.2K baud and tests revealed that operation was reasonably reliable at this speed, 19.2K baud was the data rate chosen. This high rate has the advantage of reducing the delay between the speaking of a word and the Macintosh's response.

STROBE	$\overline{CS1}$	$R/\overline{W}$	A1	A2	Code	Operation
L	X	X	X	X		(Chip not selected)
X	H	X	X	X		
H	L	L	L	L	0	Write to Control Register
H	L	L	L	H	1	Write to Parameter Address Register
H	L	L	H	L	2	Write to Parameter Data, LSB = 0
H	L	L	H	H	3	Write to Parameter Data, LSB = 1
H	L	H	L	L	4	Read Status Register
H	L	H	L	H	5	(Not Implemented)
H	L	H	H	L	6	Read Parameter Data Output Register
H	L	H	H	H	7	Same as Code 6 & Initiate Internal Fetch

SP1000 Registers and Addressing

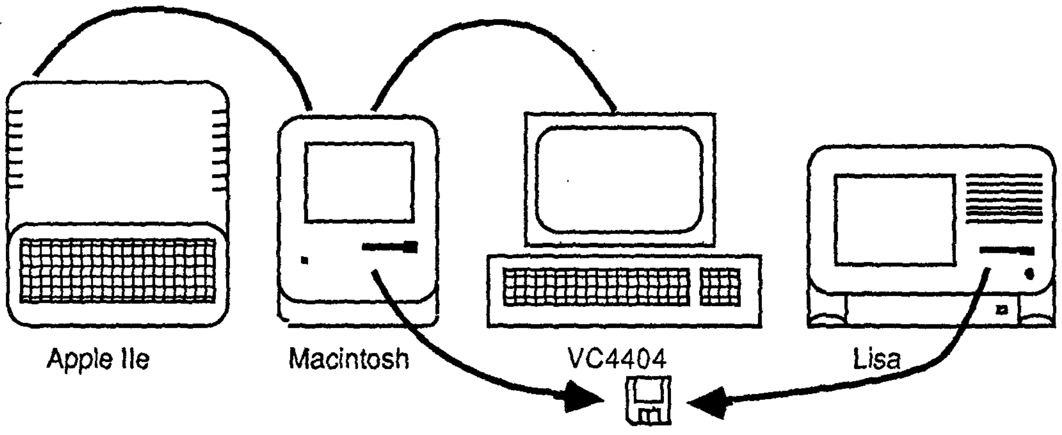
SP1000 Parameter Addressing

Parameter Numbers	Parameter Names
0-7	a(1)-a(8)
8	Energy
9	Timer Counter 1
10	Timer Counter 2
11	Sample Rate

Diagram 3.2

3.3 The Apple Macintosh

The Macintosh most used for development was a 512K 'Fat Mac', with two micro floppy disk drives. The modem serial port was connected to the Apple IIe serial card (because the modem port is guaranteed not to lose characters whatever disk activity is occurring, unlike the printer port) and the printer serial port was connected to a VC4404 data terminal. This latter connexion was for debugging purposes only and in the finished system the Macintosh could use this port for AppleTalk or Imagewriter connexions. The application was intended to have the ability to print documents in the same way the MacWrite programme can. Development programming for the Macintosh occurred on a Apple Lisa using micro floppy disks as the medium of programme and data transfer. The whole hardware development environment is summarised in diagram 3.3.



The Hardware Development Environment
Diagram 3.3

4 Software Development

4.0 Task Division

Software for the speech recognition is in two parts, one part written for the Apple IIe and the second, major component for the Apple Macintosh. At an early stage a decision had to be made as to how the total task was to be split between the two machines. The IIe had to perform at least the interface with the SP1000 chip and the Macintosh had to perform at least the standard Macintosh user interface functions, but the many intermediate functions had to be divided between the two in a clean and reliable way which gave good overall system performance. This last criterion and the fact that the Macintosh's 8MHz MC68000 processor is much more powerful than the IIe's 1MHz 6502 prompted the decision to use the IIe as little more than an intelligent peripheral

4.1 Apple IIe Software

The Lis'ner 1000 hardware card used in this project came supplied with working demonstration software which performed a task similar to the one the overall system was to perform. The supplied software was in two parts: a programme in BASIC which performed initialisation, disk handling and user interface functions; and a large assembly language programme which performed interfacing with the Lis'ner 1000 hardware and the actual speech processing. Apart from some parts of the initialisation done by the BASIC programme, the operation of the assembly language routines were of most interest.

4.1.0 Supplied Software Operation

The assembly routines included device driver code for the SP1000 chip. This code read parameters from the chip into a 2K buffer in the IIe's memory thus providing almost five seconds of buffering. Each set of nine parameters from the chip is known as a *frame*. An algorithm to detect the beginning end ending of words operated on the data, utilising the energy parameter of each frame. This algorithm implemented a four state state machine with the states being silence, increasing energy, constant energy and decreasing energy. The states and transitions are illustrated in diagram 4.0.

On transition from the decreasing energy energy state to the silence state, the data buffered since the last silence period is processed by further routines. The first step is to transform all the frames in the buffer into a list of twelve frames. Where more than twelve frames are buffered, the code detects similar consecutive frames and removes one of them until only twelve frames remain. Where a short utterance results in fewer than twelve buffered frames, frames are duplicated. The number twelve was chosen arbitrarily but allows adequate information about each utterance to be kept without requiring an unduly large amount of storage space. Also, comparison between utterances is simpler because each utterance is represented by an equal number of data. A set of twelve frames is known as a *template* and requires 108 bytes of storage.

When the supplied software is operating in the training mode the twelve frames extracted from each utterance are simply stored away. If a word is trained more than once (three is the usual frequency) the arithmetic means of each parameter of a new template with the corresponding parameter's of the stored template are found before the resulting averaged template is stored in memory. When operating in the recognition mode, the supplied software compares the template from each new utterance with the templates of each of (the up to thirty two) stored templates, printing the text of the best match on the screen. The comparisons are done using dynamic time warping as discussed in section 1.2.0. The distance measure between any two frames is made using the simple technique of summing the absolute values of the nine differences of corresponding parameters.

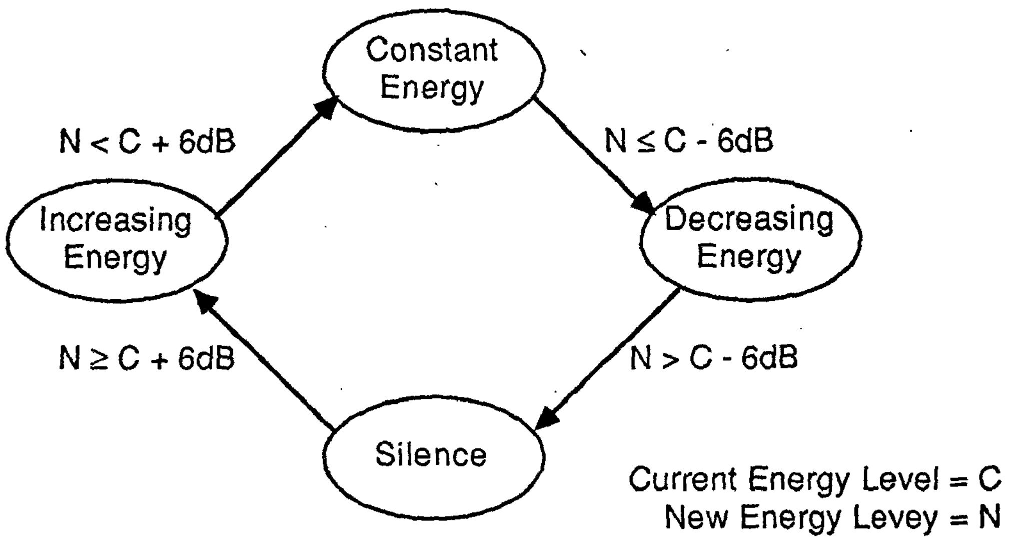


Diagram 4.0

The supplied software operated satisfactorily but somewhat slowly. Recognising an utterance requires thirty two comparisons between templates which seemed to constitute the major delay in processing. I decided that in my system the IIe should only perform the processing required to produce a template for each utterance. These 108 byte templates would then passed via the serial link to the Macintosh. The IIe need not then know whether the system is in the training or recognition mode, removing the need for any communication from the Macintosh to the IIe. All the software for template distance measures, training, dynamic time warping and searching lists for best matches then needed to be transferred to the Macintosh. Further, the supplied software needed to be modified so that these functions were no longer performed and a data transfer protocol implemented.

4.1.1 Modifications to Supplied Software

The first stage in modifying the supplied code was to change the BASIC programme. Most of the code could be discarded since the Apple IIe user interface was no longer required so all the 'fancy' screen handling was unnecessary. The first few lines of initialisation were retained and a single call to the assembly routine inserted. Originally the system made one call to the assembly routines for each utterance to be trained or recognised, but I intended to modify an assembly routine so that once called it waits for, processes and transmits utterances in an infinite loop.

Modification to the assembly routines was a time consuming task. The operation of a large number of assembly routines had to be discovered from disassembled listings. Eventually, a single routine of about one hundred instructions was rewritten. It implemented the infinite loop for processing utterances by making calls to existing routines and implemented the data transfer by making calls to the IIe's operating system. The protocol was a simple one, where the 108 bytes were each converted into two bytes by adding 65_{10} (ascii 'A') to their first and least most significant four bits respectively. These 216 bytes were then prefixed with '^' and suffixed with '$<cr><lf>' to make a total 220 bytes per utterance to be transferred. At 19.2K baud these characters could be transmitted in about 120ms.

4.2 Software Development for the Macintosh

While application development for the Macintosh can be done on the Macintosh itself, currently the best development system appears to be the one supplied by Apple for use on the Apple Lisa. This was discovered after some time experimenting with an Aztec C compiler system. Several features of the Aztec C system proved to be unreliable and memory limitations on a 128K Macintosh (the machine on which work began on my project) caused further difficulties. In particular, no resource compiler was available on the Macintosh and the resource compilers available were designed more for the modification of existing resource files rather than the creation of new ones.

4.2.0 The Lisa Development System

The Lisa system provided a Pascal compiler, a resource compiler and a MC68000 assembler. All of these tools functioned reliably throughout the course of my work with them. In addition, several sample programmes were supplied, illustrating clearly how several typical, if simple, applications could be written. One such example was a simple text editor. I decided that the best course was to modify this editor by making the addition of speech recognition code. Thus the work of writing a text editor was largely avoided and I could concentrate more on speech recognition problems.

Macintosh application programmes usually have a somewhat different structure than do other programmes. They are constructed about a single main event loop. This loop detects all events such as key presses, mouse button clicks, window updates or disk insertion events as they occur and processes them. Processing is usually performed by the main event loop passing information relevant to the different events to appropriate subroutines.

4.2.1 Asynchronous Serial I/O

The first step in the modification was to add the code to read the data sent from the IIe on the RS232C serial line. Because data could arrive at any time, even while typing or other editor functions were being performed, this reading had to be done asynchronously. Also, code was inserted to convert the 220 byte encoded templates back into 108 byte arrays of signed bytes. An alert box was set up to appear in the event of any invalid data being received. Because of the high data rate being used and the chance that a great deal of data could be sent, the input buffer used by the serial driver was expanded so that at least ten templates could be stored. Testing of this stage was achieved by sending each array as it was received to the

VC4404 and comparing the numbers with those in the IIe's memory (as found by entering the IIe's monitor).

4.2.2 Template Comparison

Once frames were being received properly, I decided to begin the implementation of the template comparison routines before the implementation of routines to establish the training or recognition protocols. This was because I felt that template comparison had to work adequately before it could be used for the many comparisons required for recognition. I decided to compare consecutive templates as they were received, printing the distance measures obtained to the VC4404. Despite initial problems caused by the a bug in my dynamic time warping implementation, I was able to reimplement the recognition routines as described in the documentation of the software supplied for the Apple IIe. Testing consisted simply of speaking words into the microphone. Speaking the same word twice should produce differences significantly less than if different words were spoken.

4.2.3 The Training Mode

Before recognition could be performed, a list of templates and associated text had to exist in memory for comparison with each new utterance. Therefore I first needed to implement routines to allow training to occur. The first step was to add a new menu to the menu bar. This was titled 'Speech' and included the items 'Active' and 'Train...'. The 'Active' item initially contained a check mark indicating whether or not utterances received in the recognition mode would used or discarded.

The 'Train...' item brings up a modal dialogue box allowing training to occur. The training list consists of a circular list of records each containing a template, a string and a count of the number of times the word has been trained. The training dialogue box allows the user to enter text and speak an utterance repeatedly. Each new utterance is averaged with the last using the count to weight new utterances so that all utterances used for training a word have equal weight. Also the user may request: a new (empty) record to train more words; that the current word be retrained; that the current list of words be forgotten; or to move to the last or the next item in the list of words. All these actions are requested through the use of buttons. Finally a 'Quit' button causes an exit from the training mode. Much of the coding for the training mode involved setting up the data structure of the circular list of words and initialising and maintaining this structure.

4.2.4 The Recognition Mode

The implementation of the recognition mode required the combination of the template matching routines with code to repeatedly compare each new template with each template in the trained list. An ordered list of the trained templates needed to be created so a further two elements were added to the structure of the training list records: a pointer to the next best match with the new template and an integer to keep the distance measure to the new template. As each word in the training list has it's distance measure from the new template taken, it is inserted in the list. The length of the list is limited to ten elements and the distance measure must be within a limit for an element to be included on it.

The most difficult part of the recognition mode to code was that involving the display by the system of the choices and the selection by the user of the required match. The most obvious way to display the best match list is to create a small window with a button for each word. The Macintosh user interface specifies that only one window be active at any time but while recognition is in progress the user may wish to use both the test window and the window containing the list of choices. It would be most inconvenient to have to repeatedly change between these windows. The only other alternative would have been to define a special type of window which could contain both text and the selection list at the same time. This would have been difficult to do and moreover would have been contrary to the Macintosh user interface guidelines.

Eventually the solution adopted was to allow characters typed while the selection window was active to be inserted in the most recently active text window. In this way the selection window could be left active most of the time with both selection and typing being done from it. Only moving the cursor or selecting items for cutting and pasting would require the user to make the text window active.

The recognition mode was found to work with a fifty word vocabulary with at least ninety percent accuracy. Even when the spoken word was not at the top of the selection list it was very high on the selection list. Despite this, the added feature of a cancel button at the bottom of the selection window was included to allow the user to cancel the list.

The actual insertion of the selected word into the text window was done using a call which caused the word to be inserted before the current insertion point. Any selection ranges active would be unchanged so that cut and paste operations would be entirely unaffected by the addition of speech recognition to the editor. At this stage of development, speech recognition was operating satisfactorily and it remained only to add a few features to make the system easier to operate.

4.2.5 Saving Trained Lists

The last main feature added was the addition of the ability to save the lists of trained words and their associated templates. Two ways of doing this were available. The trained words could be saved in the resource fork of the text files they were used with or they could be saved in a special file. The first approach has problems in that the added resources may interfere with the operation of other text editors and will, in any case, make the files larger than they need to be. The second approach has problems in that the user interface does not make it easy to work with two different types of files simultaneously.

Eventually the second approach was adopted with the three menu items 'Open...', 'Save', and 'Save As...' being added to the 'Speech' menu. These then applied to the use of word list files while the 'File' menu retained it's normal use but manipulating text files only. The editor could thus use the ordinary text files also used by other text editors such as MacWrite, Microsoft Word, the Pascal editor and others.

Implementation of the file handling was done by adding a flag to the parameter list of the file handling routines already in use. When this flag was true, the routines would behave as before, but when false they would manipulate word list files. In this way the interface was guaranteed to be the same as the usual Macintosh interface for file manipulations.

4.2.6 Bells and Whistles

A last feature to be added was the creation of three icons, one each for the application file, text files produced by the application and word list files created by the application. These were designed using a resource editor to create appropriate shapes (as appropriate as my artistic talent allowed anyway). The MacTools programme was then used to find the hexadecimal numbers corresponding to these patterns. These numbers were then entered into the resource compiler source file so that the icons could be recreated by the compiler each time changes were made to the programme. Diagram 4.1 shows the icons used.

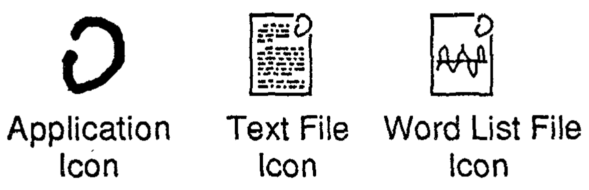


Diagram 4.1

Additional changes had to be made to parts of the resource file (specifically the bundle) which dealt with the finder's interpretation of the icon resources. Also changes had to be made to the 'exec' file which controlled the compilation sequence so that the bundle bit was set, the application was given the signature 'SPEE', text files were given the type 'TEXT' and word list files were given the type 'SPCH'. Both the text and word list files created by the application were given the creator 'SPEE' so that they would automatically cause the speech application to start on their selection.

A little known feature of the finder is it's ability to start an application with more than one file selected. Applications are most often begun by double clicking on the icon of a file created by that application, meaning that only one file is in use. An alternative method is to click on an icon and choose the 'Open' item from the 'File' menu. If more than one file is selected, either by shift clicking or selecting all the icons in a rectangular area, applications should attempt to use all of the selected files.

My application was set up so that each file of type 'TEXT' selected would be opened into a separate text window. The last file of type 'SPCH' selected would be opened and used as the initial list of trained words. By selecting both a text and a word list file, the application could be started in a single step with both the text and the word list being loaded at the start despite their being stored in different files. Files of unknown type (i.e a type other than 'TEXT' or 'SPCH') would be marked as unused. Future versions of the finder will attempt to start applications to handle unused files once the first application has terminated.

4.3 Summary

The work of implementing the system on the Macintosh went comparatively smoothly with the only major delay being caused by a bug in my dynamic time warping algorithm which took some time to find. Also, I attempted to begin development using the Aztec C compiler but decided to change to the Lisa Pascal compiler which caused additional delay. On the IIe, most time was consumed in understanding the disassembled assembly listings and in working out the DOS interface to the serial driver. Apart from that, development was smooth there also. The assembly code (about a hundred lines) was short enough to assemble by hand.

5 Results

5.0 Reliability

Overall the system seemed to perform satisfactorily. Despite considerable testing and use by myself and others, it's operation remained correct and it did not cause the system to crash during it's operation. A small problem seems still to exist with the use of an expanded file buffer on the Macintosh for receiving asynchronous data from the IIe. After the termination of the application this buffer seems to remain in place, causing the system to crash if characters are received from the IIe. This does not affect the application itself, only the finder or other applications executing subsequent to it's use.

5.1 Accuracy

As has been said, the system has a performance in excess of ninety percent when a fifty word vocabulary is in use. I have not tested vocabularies larger than this but I suspect that if more than a hundred words were in use, accuracy may begin to suffer. For smaller vocabularies, accuracy is improved. At fifteen words performance is in excess of ninety five percent. Of course, due to the use of the 'Choice' window for selecting from among the ten best matched words, the system is highly tolerant to errors.

Sources of error apart from failure of the recognition methods used include data glitches in the serial communications line, (very rare) communications buffer overruns and background noise. Of these three the last two are related in that background noise can generate many spurious 'utterances', causing templates to be generated too quickly for the system to process them. Noise sources such as the keyboard, printers, the Macintosh's speaker, background conversations and others can interfere with system accuracy. Use of the 'Cancel' button in the recognise mode and of the 'Retrain' button in the training mode can overcome spurious noise to some extent. Deactivating the system by selecting the 'Active' item from the 'Speech' menu is a useful option if background noise becomes too severe.

5.2 Speed

The system currently runs at an acceptable speed with vocabularies up to about twenty five words. As each word in the vocabulary has to be compared with each incoming utterance, the degradation of performance is proportional to vocabulary size. If more time had been available, it would probably have been possible to at least double the system's speed by rewriting the recognition routines in MC68000 assembly language.

5.3 Usability

While written as an experimental system, my application seems to usable in speeding up certain data entry tasks. With practice in it's use and especially if the recognition routines could be made faster, Pascal programme entry at least would be such a task. Entry of numbers or of other limited character sets is also feasible, provided only that a reasonably distinctive utterance can be associated with each character.

6 User's Manual

6.0 Introduction

The programme ***Speech*** is a speech recognition application designed to run on an Apple Macintosh computer. In addition to a Macintosh, an Apple IIe with a Lis'ner 1000 speech processing card, a serial communications card, a disk drive and 48K of memory is required. The application will run on a single or dual drive Macintosh with at least 128K of memory.

Speech operates in a way very similar to text editors written for the Macintosh such as MacWrite and Microsoft Word. While it does not posses the advanced character and layout formating features of either of these editors, it produces files which are usable by them so that it can be used in conjunction with them.

Speech's advantage over other text editors lies in its ability to recognise words spoken into a microphone. The text of these words are then inserted into the document being edited. The words which can be recognised are limited in number and the best recognition is achieved for a single speaker only. To be recognised, consecutive words must be separated by a period of silence.

This manual assumes that you are familiar with the Macintosh and know how to operate the mouse, the keyboard, menu bars, windows, icons, controls and other common Macintosh features.

6.1 Setting Up the Hardware

A 48K Apple IIe with a disk drive is required. The disk controller card should be inserted in slot six, the Lis'ner 1000 card in slot four and the serial communications card in slot one. The serial communications card should be set to transmit data at 19.2K baud and a cable connected from it to the Macintosh's modem port. A specially wired cable will be required for this purpose.

Insert the 'LISTEN' disk into the IIe's disk drive and turn the machine on. Type

```
        PR#1 <RETURN>
        RUN LISTEN <RETURN>
```

and wait for the 'In Use' light on the disk to go off. The 'LISTEN' disk can now be removed from the drive. There is no need to touch the IIe again as the Macintosh can now receive speech recognition data from it as required.

6.2 Starting the Application

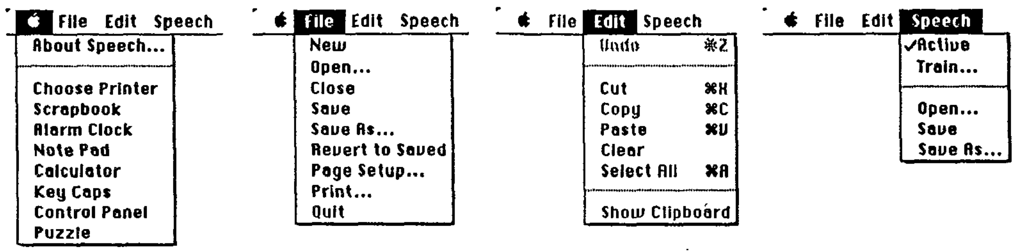
Start up the Macintosh in the usual way, by turning it on and inserting the ***Speech*** disk into a drive. ***Speech*** may be started in two separate ways, by double clicking on a ***Speech*** icon or by selecting a ***Speech*** icon and then choosing the 'Open' item from the 'File' menu. The three different ***Speech*** icons are shown in diagram 4.1. If you start up using the application icon only, ***Speech*** will start with an untitled text window (i.e. no text files will be opened) and it will not recognise any words. Starting with text icons causes ***Speech*** to created a separate window for each text file selected. Starting with a word list file means ***Speech*** can recognise the list of words contained in that file. Starting with more than one file (often starting with both a text and a word list file is useful) can only be done by selecting all of the files required at once and choosing 'Open' from the menu bar.

Editing with a window can be achieved by typing the required text from the keyboard. The characters typed will appear at the *insertion point* on the screen. Backspacing will delete text. To move the insertion point, move the 'I' beam cursor to the new position required and click the mouse button. Areas of text can be *selected* by moving the 'I' beam cursor to one point in the text, holding down the mouse button, and dragging the cursor to a second position. All text between the two positions will be selected and displayed in inverse colouration to reflect this. Any typing done once text is selected will replace the selected text.

The windows can be resized by using the grow icon in the normal way and if the file extends beyond the window, the scroll bars can be used to move the window, again in the normal way. The more advanced features of the editor, along with the speech recognition features, are accessed from the menus of the application.

6.3 Menus

Speech operates just like other text processing applications on the Macintosh. Shown below in diagram 6.0 are the various menu's available using ***Speech***.



The Menus from Speech
Diagram 6.0

The first of these menus (the '' menu) allows the use of all the desk accessories currently installed on your disk and the 'About Speech...' item displays the author and version number of the software you are using. The desk accessories listed may vary depending on the system file of the startup disk in use.

6.3.0 The 'File' Menu

The second or 'File' menu allows the manipulation of the text files by the application. ***Speech*** can work with several text files at once (subject to the memory limitations of the Macintosh in use) with each file having a corresponding window on the screen. The foremost or *active* window is the one in which all editing will take place. Making a different window active is done, as in most Macintosh applications, by clicking somewhere on an inactive window.

6.3.0.0 The 'New' Item

If a new text file is required at any time, choose the 'New' item from the 'File' menu. This will create a window containing no text and titled 'Untitled-*n*', where ***n*** is an integer used to make sure no windows or files of the same name are created. The first new window will have *n* = 1. When untitled windows are saved ***Speech*** prompts for a name to be given.

6.3.0.1 The 'Open...' Item

If a text file needs to be used by ***Speech*** but is not currently open, i.e. it does not have a window on the screen, choose the 'Open...' item from the 'File' menu. This will allow the any text file on any inserted disk to be opened whereupon a window will be created for that file and made the foremost (and active) window. If a file which is already open is opened again, it will be renamed with the text 'Copy of ' prepended to it's name.

6.3.0.2 The 'Close' Item

Any text window can be closed in two ways, either by clicking in the go away box on it's title bar or by making it active and choosing the 'Close' item from the 'File' menu. When a window is closed without any changes made to it having been saved to disk, ***Speech*** will prompt the user to allow saving to occur as required. Also, an opportunity to title untitled windows will be given.

6.3.0.3 The 'Save' Item

Any text window can be saved by making it the active window and choosing the 'Save' item from the 'File' menu. If the window is untitled, the user will be prompted for a title. Saving causes any changes made to the text in the window to be reflected in the file stored on disk

6.3.0.4 The 'Save As...' Item

If a text window needs to be saved with a new name because it is untitled, because the old version of the file is still required or for any other reason, the 'Save As...' item in the 'File' menu can be used after the window is made active. ***Speech*** will prompt for a name and save the contents of the of the window in a file of that name. The window will also be retitled to match the new name.

6.3.0.5 The 'Revert to Saved' Item

If changes made to the active window need to be discarded, the version of the file last saved on disk can be retrieved by choosing the 'Revert to Saved' item in the 'File' menu.

6.3.0.6 The 'Page Setup...' Item

To set the page format of the active window to be used in printing, choose the 'Page Setup...' item from the 'File' menu.

6.3.0.7 The 'Print...' Item

To print the text of the active window, choose the 'Print...' item from the 'File' menu. A dialogue box will appear allowing the selection of various modes of printing. The text will then be sent to the attached printer.

6.3.0.8 The 'Quit' Item

When ***Speech*** is no longer needed, choose 'Quit' from the 'File' menu. If any text windows remain open they will be closed. The Macintosh will then return to the finder.

6.3.1 The 'Edit' Menu

Various basic text editing items are available from the 'Edit' menu. These allow the user to perform most operations available from other text editors. One missing feature is the 'Undo' item. This item remains disabled whilst ***Speech*** is in use.

6.3.1.0 The 'Cut' Item

Choosing the 'Cut' item from the 'Edit' menu causes whatever text is selected (i.e. text shown as white on a black background) to be removed from the active window and placed on the clipboard. The clipboard is a special file which keeps the contents of text required for later use.

6.3.1.1 The 'Copy' Item

Choosing 'Copy' from the 'Edit' menu causes the selected text to be placed on the clipboard. Unlike the 'Cut' item, 'Copy' does not change the contents of the active window.

6.3.1.2 The 'Paste' Item

The 'Paste' item from the 'Edit' window places any text on the clipboard in the active window at the insertion point (i.e. at the cursor or selection range). If any text is selected on the active window, this is deleted before pasting occurs, thus the clipboard effectively replaces selected text when the 'Paste' item is chosen.

6.3.1.3 The 'Clear' Item

The 'Clear' item from the 'Edit' window works like the 'Cut' item except that the selected text is deleted without a copy being place on the clipboard.

6.3.1.4 The 'Select All' Item

When the 'Select All' item is chosen, the selection range of the active window becomes all the text in that window. All the text is shown as white on black to reflect this.

6.3.1.5 The 'Show Clipboard' and 'Hide Clipboard' Items

The 'Show Clipboard' or 'Hide Clipboard' items allow the contents of the clipboard to be displayed (if they are not already) or concealed (if they are displayed). The clipboard appears as a window initially at the bottom of the screen.

6.3.2 The 'Speech' Menu

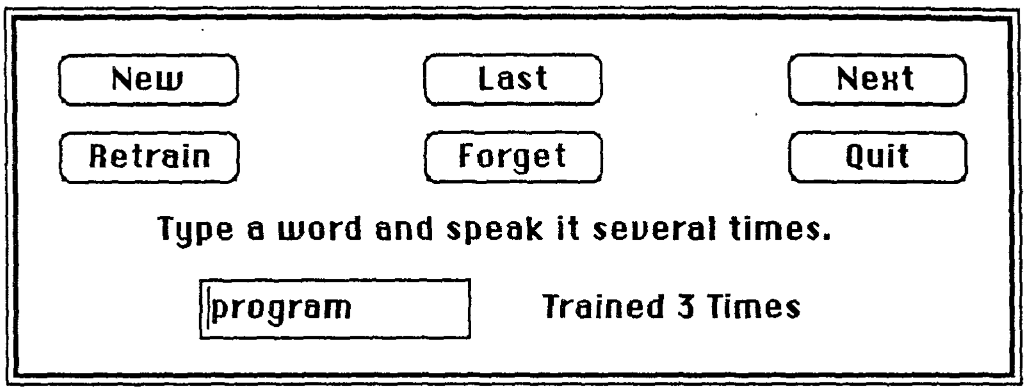
The last menu is the 'Speech' menu. It controls both the current mode in which ***Speech*** is operating and the manipulation of word list files. ***Speech*** recognises a list of words entered by the user in the training mode. The user can type several characters of text and speak the corresponding utterance one or more times. This causes the utterance and the text to be associated. During recognition, the utterance stored with the current word list which best matches a new utterance will have it's corresponding text inserted before the insertion point of the active window. The number of utterances in use at once is limited by the memory available on the Macintosh in use.

6.3.2.0 The 'Active' Item

The 'Active' item from the 'Speech' menu allows the user to turn the recognition of words on and off. Selecting the item toggles between these two alternatives. A check mark indicates that recognition is currently on and no check mark indicates that words spoken into the microphone will be ignored.

6.3.2.1 The 'Train...' Item

The 'Train...' item from the 'Speech' menu is used to enter the training mode. In the training mode, utterances spoken into the microphone are not used for recognition purposes and the training mode is unaffected by whether the system is active or not. Choosing 'Train...' causes the dialogue box shown below in diagram 6.1 to appear.



Type a word and speak it several times.

program

Trained 3 Times

The Training Dialogue Box
Diagram 6.1

To associate an utterance with text, the text should be typed in the edit text item on the dialogue box. In the example shown, the text 'program', has been entered. The utterance should then be spoken one or more times into the microphone. The dialogue box will show a count of the number of times the utterance has been trained. Training twice or thrice will usually be adequate.

To enter further text and associated utterances, click on the 'New' button. This button inserts a new association into the system's vocabulary when at least one utterance is received. If a wrong utterance is spoken or if spurious noise occurs, clicking the 'Retrain' button will cause the utterances associated with the currently displayed text to be forgotten. The 'Forget' button causes the entire current vocabulary to be lost.

The system's vocabulary of associations is arranged as a list. By clicking the 'Next' and 'Last' buttons, the user can move through the list to modify, delete or insert words. The order of the list is unimportant for recognition purposes. The 'New' button causes new words to be added *after* the word displayed at the time it was clicked. When training is completed, the 'Quit' button will remove the training dialogue box and return the system to the recognition mode.

6.3.2.2 The 'Open...' Item

The 'Open...' item from the 'Speech' menu can be used to load a new word list into the system, effectively changing the current vocabulary. Only one vocabulary can be stored at a time so the 'Open...' item causes the existing vocabulary to be lost.

6.3.2.3 The 'Save' Item

Choosing the 'Save' item from the 'Speech' menu causes the current vocabulary to be saved to disk. If the vocabulary has not been saved before, the user will be asked to name it and a new file of that name will be created. Files containing vocabularies are stored in a special format and are therefore unreadable by any application except ***Speech***.

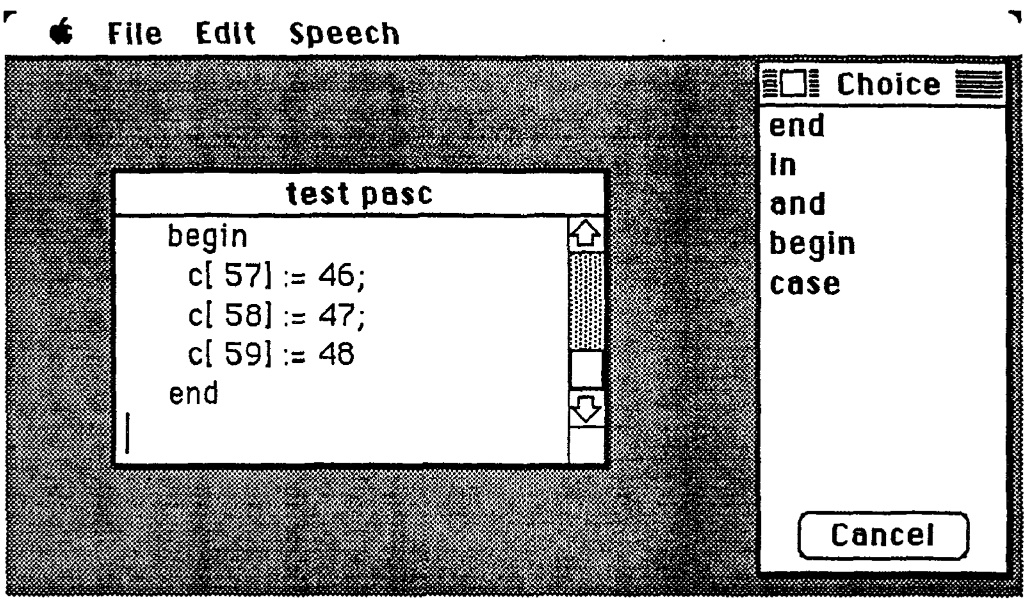
6.3.2.4 The 'Save As...' Item

The 'Save As...' item from the 'Speech' menu allows the current vocabulary to be saved in a file with a name different from the vocabulary's name. This is useful if the current vocabulary has no name or if it has been modified but the old version is still required.

6.4 Recognition

The 'Speech' system can be described as a single user, discrete utterance voice recognition system. This means that utterances trained in a vocabulary by one person will not be recognised very well when spoken by any other person. Also, utterances must be separated by a period of silence so that the system can separate consecutive words in a phrase. When a large vocabulary (more than about twenty five words) is in use, it may take some time for each utterance to be recognised.

As each utterance is heard, it is compared with all the utterances stored in the system's current vocabulary. This vocabulary can be produced and modified as described in section 6.3.2. The system prepares a list of the best matches between the new utterance and the stored utterances. The list is in order so that the best matches come first. Only reasonable matches go on the list and it has a maximum length of ten items. When the list is ready, the text associated with the items on the list is displayed on the screen in a window titled 'Choice', as shown on the sample screen in diagram 6.2.



test pasc

```
begin
  c[ 57] := 46;
  c[ 58] := 47;
  c[ 59] := 48
end
```

- end
- in
- and
- begin
- case

The Appearance of the 'Choice'
Window During Editing
Diagram 6.2

The user can now select the correct text by clicking on it in the 'Choice' window. The selected text will then be inserted into the foremost text window. If the correct text is at the top of the list, the user need not select it. It will be automatically selected when either a new utterance is spoken or when characters are typed from the keyboard. In this way a long list of utterances can be entered on the screen without the need to use either the mouse or the keyboard. If the correct text is not shown on the 'Choice' window or if background noise has caused a spurious 'utterance' to be heard, the 'Clear' button on the 'Choice' window can be clicked to cause the current list to be emptied.

The 'Choice' window can be closed at any time by clicking in it's go away box or by deactivating the system. It will reappear only when a new utterance is detected while the system is active.

7 Bibliography and Acknowledgements

7.0 Bibliography

[1] S. Ciarcia (Nov. 1984) 'The Lis'ner 1000', *Byte* **Vol. 9,** No. 12 pp. 111-24

[2] Micromint Inc. (Oct. 1984) 'Lis'ner 1000 Voice Recognition User & Assembly Manual' Micromint Inc.

[3] M.Wagner (Apr. 1981) 'Automatic Labelling of Continuous Speech with a Given Phonetic Transcription Using Dynamic Programming Algorithms', *IEEE Conference on Acoustics, Speech and Signal Processing*, pp. 1156-9.

[4] G. Bristow (1984) *Electronic Speech Synthesis.* McGraw-Hill.

[5] Apple Computer Inc. (1984) *Inside Macintosh* Apple Computer Inc.

7.1 Acknowledgements

The first person to acknowledge is my supervisor Prof. Reinfelds who encouraged me to begin, continue and eventually to complete this project. Also, many of his ideas are present in it's implementation. Dr. Wagner must be thanked, for without his excellent speech processing course and advice my success would have been much more limited. Dr. Gray was of great assistance in helping me understand the innermost workings of the Macintosh, a machine easy to use but often difficult to programme. My thanks also go to all those who lent books, documentation, advice and expertise to me for this project.