The CONTROL OF A SPEECH SYNTHESISER BY AN IBM PC.

H.S.J.Purvis.
Speech Hearing & Language Research Centre
Macquarie University

ABSTRACT- This paper describes a computer program and interface card used to control a serial formant speech synthesiser. The program is used to enter parametric data using the keyboard, the data may be modified using a mouse.

INTRODUCTION

This program is mainly intended to be used by students at Macquarie University. They are given the task of analysing a word or short sentence and extracting the parameter values required to control the speech synthesiser. The data is entered into the computer using the keyboard and saved as a file on the disc. The data is output to the synthesiser and the mouse is used to make any adjustments that are needed.

THE SYNTHESISER

This is a hardware serial formant speech synthesiser which was designed by Associate Professor Clark and built at the Speech Hearing and Language Research Centre. The synthesiser is controlled by a set of twelve 12 bit words, one for each parameter. Bits 0 to 7 contain the data for the parameter, bits 8 to 11 contain the channel number of the parameter. The synthesiser contains a multiplexer controlled by the channel number bits to send the data to the section of the synthesiser to be controlled. Eleven of the channels use eight bit digital to analogue converters and the synthesiser is controlled by the analogue output. The twelfth channel is divided into four two bit channels used to control formant bandwidth and fricative voicing. The data is normally sent to the synthesiser at a rate of 100 sets of data per second.

THE INTERFACE

An interface with a timer that interrupts the computer program after the countdown and a 12 bit parallel port was needed. The interface that was selected after considering the limited number available is known as the 8255 I/O card and contains three timers, two eight bit parallel ports, and two four bit parallel ports for control purposes. Only one timer is used, its input being 1MHZ and it is set to divide by 10,000 to produce a 100HZ output.

The card needed two modifications. The first was to connect the timer output to an interupt line. The second was to provide a 1MHZ clock signal derived from a crystal oscillator. A clock signal is available from the computer but this changes frequency when the 'Turbo' switch is operated.

The computer sends the data to the synthesiser 100 times per second without any 'handshaking'. It is assumed that the synthesiser will follow the changes in data values.

THE MOUSE

The mouse provides a means of making adjustments to the parametric data while listening to the synthesiser sound output. The previous version of this program on another computer used a light pen. After unsuccessfuly attempting to use a light pen on the IBM PC, it was decided that the mouse would be used instead. It would seem that the light pen is intended to be used mainly as a pointing device rather than a curve drawing device as required in this application. The mouse gives reasonable results, the curves can be drawn accurately but the speed of drawing is rather limited. This is not such a severe limitation if the mouse is used for fine adjustments rather than attempting to make large changes to the data.

COMPUTER LANGUAGE AND GRAPHICS

The program is written using Turbo Pascal. Program development, testing, and debugging is easy and quick. It enabled the interrupt subroutine for the output of data to the synthesiser to be written without the need to use an assembler because the small sections of machine language needed could be inserted into the pascal program.

The Turbo Pascal Graphics Toolbox is used for graphics. This simplifies the plotting of curves on the screen but does not provide for either a light pen or a mouse. The mouse uses a driver that must be installed before the program is used. This driver can then be called from the Pascal program to set up the mouse and to fetch its location.

The Graphics Toolbox uses world coordinates to specify data values. This means that the actual data values (formant frequencies in HZ or amplitude values in DB) are used to plot the graphs. The Graphics Toolbox translates these world values to actual screen values in pixels. Because the mouse is not handled by the Graphics Toolbox it returns screen pixel locations, not world values. These have to be translated into world values before being stored in memory or sent to the synthesiser. This is one reason for the slow operation of the mouse.

THE PROGRAM

There are two versions of the program. The complete version must be used on the one computer that has the mouse, interface, and colour graphics board. The other version does not provide any graphics or data output to the synthesiser, but can be used to input the data and save it as a disc file. It can be used on any available IBM PC.

When the program is started it lists the available commands on the screen and asks the user to presses a key. It then plots channels 0 to 5 on the screen and waits for a command. The complete word may be typed but this program only examines the first two characters. The commands 'SPEAK' and 'SP' are the same.

The program has a data store with a capacity of 256 data points. At 100 data points per second this represents 2.56 seconds of speech. The data in the store consists of the value in HZ or DB, before being sent to the synthesiser it must be translated into a number from 0 to 255 (8 bits).

The following commands are provided by the program :-

1. LOw, Display Channels 0 to 5.

2. HIgh, Display Channels 6 to 10 and 12.

Note that Channel 11 is not displayed because it is four channels combined and the display would be difficult to interpret.

3. Data entry using the keyboard.

The command is TYpe,NN where NN is the channel number.

The program asks for the first and last data point numbers and the first and last data values.

Each channel has a maximum and minimum value, these are shown on the screen and the data values typed by the user are checked so that out of range values can't be entered.

To enter a block of data points having the same value, the first and last values will be the same. To enter a linear change type the data values at the start and finish of the change. A single data point can be entered

by entering the same value for the first and last point numbers, and first and last data values. It is considered that most of the data will be in the form of blocks with one value or blocks with a linear transition between values rather than single points.

If channel 11 is selected for data entry the user is asked to select the subchannel to be used. This section of the program does not provide for linear interpolation because each subchannel has only four possible values.

4. Use of the mouse to modify data.

The command to start using the mouse is CHannel,NN where NN is the channel number, the mouse can't be used to modify channel 11.

The display is changed to show a single channel that uses the whole screen. The mouse is shown as a small arrow that moves over the screen as the mouse is moved. The data point number and the data value at the point of the arrow are shown at the bottom of the screen. The mouse can be used to check the data values by moving the arrow to the data graph.

When one of the buttons on the mouse is held down the data curve can be redrawn. After a little practice the curve can be drawn accurately. If the mouse is moved too fast for the program then some points will be missed and will retain their original values, the mouse then has to be moved back over the curve to catch the missed points.

5. Saving and loading the data.

The data can be saved as a disc file by using the PUt command. This command displays a disc directory and asks for the name of the file. If the file already exists it will be used and any previous data will be lost.

As the data is adjusted it is possible to save several versions with different file names.

The GEt command reads the data from a disc file. It also displays a disc directory to help the user to remember the file name.

6. Sound control.

The sound is turned on with the SPeak command and turned off with the QUiet command. The sound can be on while you adjust the data with the mouse.

7. Store Reset.

The ZEro command is used to clear out the data store by filling each of the channels with the minimum data value for that channel.

8. Printing and Listing.

The PRint command is used to print part or all of the data on the line printer. The user is asked for the first and last data points to be printed.

The LIst command works the same way except that the data is listed on the screen.

9. The HElp command produces a list of all available commands.

COMPUTATIONAL MODEL OF THE PERIPHERAL AUDITORY SYSTEM FOR SPEECH RECOGNITION: INITIAL RESULTS

Ara Samouelian † and Clive D. Summerfield †
†School of Electrical Engineering,
The University of Sydney

ABSTRACT - This paper describes the design of a computational model of the peripheral auditory system, which is controlled via the AUDLAB Interactive Speech Signal Processing Package using a programmable harness to interface the AUDLAB command protocol and track file format to the structural model of the cochlear processor. A suite of signal processing modules, originally developed for speech synthesis research has been supplemented by a number of non-linear signal processing modules to model the transduction stage of the Cochlear. Some initial results of the cochlear processor model and its performance on real speech signals are presented.

INTRODUCTION

Reliable recognition of speech is fundamental to man-machine commmunications. Although there exists a number of different recognition strategies, such as dynamic time warping, Markov modelling, neural networks and phonetic feature extraction, all of these approaches rely on the effectiveness of the input signal processing to generate the parameters upon which the recognition is performed.

Traditionally, input speech signal analysis is performed by using linear signal processing algorithms, such as Fourier Transforms or Linear Predictive Coefficient analysis. However, recent results from extensive investigations by Seneff (1988) show that there is significant advantages in using models of the cochlear.

The logarithmic frequency scaling used in these mdels closely matches the frequency resolution known to be effective for speech perception. Moreover, the non-linear temporal properties of the model, enhances the acoustic features of the speech signal known to be of central importance in speech recognition. Ghitza (1986) also presents evidence that synchronous auditory based models exhibit significant improvement in recognition performance in high noise environment .

As well as these advantages, there are significant engineering motivations for developing a computational model of the peripheral auditory system. The computational model exhibits a high degree of structured regularity and process concurrency, and lend themselves to very efficient VLSI implementation using bit-serial design approach primarily reported by Lyons (1982).

Our ultimate aim is to investigate the design of a cochlear model VLSI chip which can be used as a front end signal processing module for a speech recognition unit, incorporating processes which enhance speech recognition performance. The design strategy utilises a hierarchical design approach,

which has previously been used to implement VLSI formant speech synthesis ASIC (Summerfield, 1988).

The AUDLAB Interactive Speech Signal Processing software package was developed at the Centre for Speech Technology Research, at the University of Edindurgh and is designed primarily for speech recognition research. It extensively uses real-time data I/O capability of the MASSCOMP UNIX to record and reproduce speech signals and the colour graphics interface to display speech waveforms and analyse results. The package has a large inventory of speech signal processing programmes which can be easily combined to produce complex signal processing algorithms. Thus, AUDLAB provides an interactive, flexible and extremely versatile and productive research environment for cochlear model processing for speech recognition.

The present computational model used in this research has been implemented within AUDLAB. Our aim is by using this approach to provide a mechanism to investigate the trade-off between structural complexity and the performance of the model. AUDLAB also provides an ideal environment for evaluating and determining the most prominant features of speech, which need to be extracted to enhance the speech recognition signal.

SOFTWARE STRUCTURE OF THE COMPUTATIONAL MODEL

The computational model of the peripheral auditory system utilises a set of generic signal processing modules, written in the 'C' programming language. All the modules share a common generic communication protocol, which enables a structured model of the cochlear process to be constructed, using the UNIX piping, redirection and tee facilities. The suite of linear signal processing modules ,such as resonators (poles), anti-resonators (zeros) and differential filters, which were originally developed for speech synthesis research, has been supplemented by a number of non-linear signal processing modules, such as detectors, adaptors and automatic gain control to model the cochlear transduction stage. Construction, calibration and execution of the signal processing C-shell scripts is controlled by a software harness, which interfaces to the AUDLAB Interactive Speech Processing Package.

Figure 1 shows the software architecture of the AUDLAB harness. The cochlear processor architecture is controlled via the AUDLAB Interactive Speech Signal Processing Package using programmable harness to interface the AUDLAB command protocol and track file format to the structural model of the cochlear processor. The programmable harness is installed in the AUDLAB as an external signal processing module and is activated via the internal AUDLAB menu controls.

The harness calls the relevant programming modules and parses to it the appropriate parameters such as number of filters, bark spacing, and sampling rates to create C-shell UNIX scripts. It also controls AUDLAB track file format conversion and file header manipulation necessary to process the speech sample data files. It creates appropriate track files to display within AUDLAB the filter characteristics, spectogram of the wide band input speech signal and the cochleogram.

The harness can perform several processes from simple menu selection within AUDLAB. These in-