



PITCH DETERMINATION ALGORITHMS FOR SPEECH AND THEIR IMPLEMENTATION
USING A HIGH PERFORMANCE SINGLE CHIP DIGITAL SIGNAL PROCESSOR

M.R. Varley, Prof. R.J. Simpson and Prof. T.J. Terrell

School of Electrical and Electronic Engineering, Lancashire
Polytechnic, Preston, Lancs. PR1 2TQ UNITED KINGDOM

ABSTRACT

This paper describes two algorithms for pitch determination of speech signals. One is an extensively modified Tucker and Bates algorithm which extracts features of the signal in the time domain and uses these to estimate the pitch period, and the other is a modified SIFT algorithm involving short-term autocorrelation of the LPC residual signal.

For a real-time implementation, the modified Tucker-Bates algorithm was written in assembly language for a single chip DSP device, the Motorola DSP56001. Details of the algorithm and its implementation are explained and its performance discussed.

INTRODUCTION

There are many application areas in which an estimate of the pitch of a speech signal is useful, both as an independent parameter [1,2], and as a part of a speech processing system [3,4]. Many techniques exist for pitch determination; however none of these are 100% reliable under all conditions (e.g. different background noise types and levels, different speakers). In this paper two pitch determination algorithms (PDAs) are briefly introduced, one operating in the time domain and the other involving a short-term analysis technique. Initial development of these algorithms has been undertaken in non real-time using the language C on an IBM PC-XT.

For faster processing, the time domain PDA was selected for implementation using the Motorola DSP56001, a high speed high density CMOS digital signal processor based on a Harvard architecture. This processor has the useful feature of two independent data memory spaces, which facilitates the development of efficient DSP software. An additional advantage of this choice of processor is the availability of an inexpensive assembler, simulator and development system compatible with an IBM PC. The PDA has been written in assembly language for this device, thus enabling a real-time implementation for speech sampled at 10kHz.

PITCH DETERMINATION ALGORITHMS

The algorithms used in this investigation are based on a time-domain algorithm originally proposed by Tucker and Bates, and a short-term analysis technique known as the SIFT (simplified inverse filter tracking) algorithm. The algorithms, and modifications made by the authors to improve performance, are described in detail in [5].

The time-domain algorithm [6] is termed the 'secondary feature' algorithm, and operates by reducing the formant content of the signal by applying adaptive centre clipping, then processing the resultant pulse train by extracting features of each pulse. These features are:

- i) pulse width,
- ii) energy contained in pulse,
- iii) ratio of maximum amplitude to square root of energy.

The parameters are compared to those of previous pulses, and if corresponding parameters are sufficiently similar, a pulse match is defined and a pitch period estimate obtained. This estimate is validated if it lies within a given range of the previous valid estimate, the range being determined by a maximum rate of change of pitch frequency of 1%/ms.

The SIFT algorithm [7] involves windowing the signal over a short time interval and applying an inverse filter to yield an approximation to the glottal excitation signal, i.e. the LPC residual, for that window. Downsampling from 10kHz to 2kHz prior to this enables the inverse filter to be of order 4, which simplifies filter design and implementation. Short-term autocorrelation analysis is carried out on the LPC residual, and a strong peak in the autocorrelation function is taken as an initial pitch period estimate. Again, possible errors are reduced by limiting the maximum rate of change of pitch frequency to 1%/ms.

For both the time domain and SIFT algorithm, the effect of introducing this additional criterion,

and also other modifications, are documented in [5], where relevant results are also presented.

MODIFIED SECONDARY FEATURE PDA IMPLEMENTATION USING THE DSP56001

The DSP56001 [8,9] is a 24-bit HCMOS device, based on a Harvard architecture, and is designed specifically for real-time DSP applications. Data movement occurs over three bidirectional 24-bit buses: the X data bus, Y data bus and global data bus, and program code is transferred over the P data bus. The X and Y memory spaces each have 256 x 24-bit words of on-chip RAM, while the P memory has 512 x 24-bit words. Each of the three memory spaces may be expanded off-chip to 64K x 24-bits. It is however more efficient to use internal memory whenever possible, since internal X, Y and P memory can be accessed in parallel, whereas when external memory is used, at least one instruction cycle is needed for each memory access. The core of the processor consists of the data ALU, the address ALU and the program controller which operate in parallel. The device also has on-chip peripherals and a memory expansion port.

The assembly language DSP56001 program for the modified secondary feature PDA may be explained by making reference to Fig. 1, which shows a flowchart for the main steps. It can be seen that it is necessary to have in memory two 100-point frames of data, since the centre clipping levels for a particular frame are determined from the preceding and subsequent frames' extrema. Two 'circular scratchpads', i.e. areas of memory which are accessed using modulo addressing, are used to store the two data frames and the extrema of the most recently read three frames. A third circular scratchpad stores the parameters of the most recent 32 pulses for use in the matching stage of the algorithm. Several other areas of X and Y memory are used for storage of intermediate parameters throughout the program. The memory map is defined such that the scratchpads, temporary storage areas, and program code reside in internal memory, thus allowing faster implementation of the algorithm.

For prototype versions of the program, data is taken from a block of external X memory. Six hundred 100-point frames of data are initially stored at X:\$1000-\$FA5F, enabling six seconds of speech sampled at 10kHz to be processed. In a practical single board hardware realization, the speech would be sampled at 10kHz and converted to a 12-bit binary representation using an inexpensive A/D convertor. These sample values would be stored in a dual port RAM, and accessed by the DSP56001 in blocks of

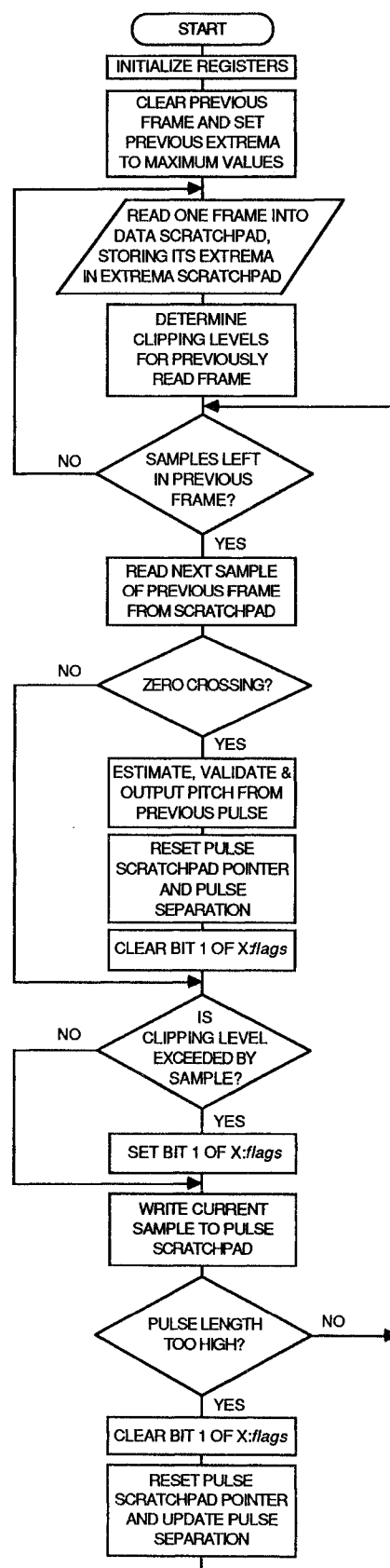


Fig. 1 Flowchart for 56001 PDA Implementation

100 points (one frame) as external X memory for analysis frame by frame. At least two frames of signal data need to be stored to ensure there is sufficient processing time and to guarantee no loss of data. Control and timing may be achieved using the Memory Ready (MRDY) and Memory State (MRDS) lines of the DSP56001 in conjunction with the RD, DS and X control lines. This arrangement would enable processing of a continuous signal in a real-time situation.

As a frame of data is read in and stored in the data scratchpad, its extreme values are found and stored in the extrema scratchpad. After reading in a complete frame K, the previous frame (K-1), stored in the data scratchpad, may be processed. The +ve and -ve clipping levels for frame (K-1) are determined by taking 80% of the smaller of the appropriate extrema of adjacent frames, i.e. frames (K-2) and K. These clipping levels are held in registers X1 (+ve) and Y1 (-ve) ready for application to data frame (K-1).

Frame (K-1) is centre clipped by examining its sample values sequentially until a zero crossing, defined by a sign change or zero sample, is located. Since the centre clipper passes a full pulse, i.e. a signal segment between zero crossings, if any sample in that pulse exceeds the appropriate clipping level [5], a pulse scratchpad is used to store the samples following a zero crossing irrespective of their magnitude. Each sample written to the pulse scratchpad is compared to the appropriate clipping level, and a bit in flag memory set if the level is exceeded. The status of this bit is examined when the next zero crossing is encountered, and used to determine whether the pulse is removed by the centre clipper. Pulse length is limited to a maximum of 99 samples: any pulses exceeding this length are removed by the centre clipper.

Encountering a zero crossing whilst tracking through the sample values signifies the end of the current pulse, which is stored in the pulse scratchpad. If no samples in this pulse exceeded the appropriate clipping level, then the corresponding bit in flag memory is clear and no further processing of the pulse is required. If the bit in flag memory is set, then the pulse is passed through the centre clipper and is to be represented by the three parameters outlined in the previous section. A fourth parameter is also evaluated: the separation of the peak values of the current pulse and the previous pulse passed through the centre clipper. This and the pulse width are easily found from the contents of address registers R2 and R6 which are used as a pointer to the pulse scratchpad and a separation counter respectively.

Evaluation of the energy of the pulse requires

computation of the sum of the squares of the pulse samples. The 12 bits representing the sample value are stored in the most significant 12 bits of the 24-bit word, with zeros as the least significant 12 bits, so squaring this yields a 48-bit result with zeros as the least significant 24 bits (B0). Consequently B1 contains a scaled representation of the pulse energy. In cases where the total energy exceeds the maximum representable in 24 bits, the running total overflows into extension register B2. However, when placed on the data bus for writing to scratchpads, the value is saturated since B2≠0. For all real speech signals processed, it was found that saturation does not occur, since the energy can be fully represented in B1 without overflow into B2.

Whilst tracking through the pulse samples and evaluating the energy E, the maximum amplitude A of the pulse is found by successive sample comparison. The square root of the energy is found by applying the iterative Newton-Raphson technique, and a division subroutine, employing the DIV iteration, is then called to evaluate the parameter C, where:

$$C = \frac{A}{\sqrt{E}}$$

The four pulse parameters are then stored in the parameter scratchpad and the pulse matching scratchpad, completing characterization of the current pulse.

Attempts to match the current pulse with previous pulses are made in order to produce a pitch estimate. Parameters of the most recent 32 pulses are held in the parameter scratchpad, and these are compared to those of the current pulse in sequential order, starting with the most recent, to check for a pulse match. Parameters are said to match if they are sufficiently close to each other in magnitude [6]. For pulses to match, all corresponding parameters must match and the pulses must be of like sign, i.e. the C parameters must have the same sign. A location in X memory holds the address of the first parameter of the first pulse which should not be checked. This is necessary because pulses which have been matched with a later pulse should not be matched again, and so matching attempts should stop at that pulse.

The parameters of previous pulses in the parameter scratchpad are tracked through, and attempts to match them with those of the current pulse are made. A running total of the separation of consecutive pulses is kept, to be used as a pitch period estimate when a pulse match is found. If this exceeds 250 samples, then no further matches are attempted with the current pulse, thus applying an absolute lower

limit of 40Hz to the pitch frequency. If the separation of matched pulses is less than 20 samples, then the match decision is reversed and further attempts are made with previous pulses. This applies an absolute upper limit of 500Hz to the pitch frequency.

Having successfully matched the current pulse with a previous pulse and defined a pitch period estimate, the pitch frequency is evaluated using the division subroutine. The numerator for the division is set to the integer value 1, yielding a scaled estimate of the pitch frequency, which must be validated by checking that it lies within a certain range centred about the previous valid estimate. The maximum permitted rate of change of pitch frequency for the algorithm is 1%/ms, corresponding to 0.1%/sample for the sampling frequency used (10kHz). For ease of programming and speed, this is approximated by $\frac{1}{1024}$ /sample, i.e. 0.997%/ms. A running total of the number of samples elapsed since the previous valid pitch estimate is kept throughout processing, and this is used, along with the previous valid estimate, to compute the permitted range for the current estimate. If the new estimate lies outside this range, then the match decision is reversed and further attempts to match the current pulse with previous pulses are made.

When a pitch frequency estimate has been validated, it may be output to the appropriate output port. In prototype versions of the code, a location in Y memory is used, writes to which are filed using appropriate commands in the simulator or development system software packages. Following the pitch estimate, the time elapsed in samples since the preceding valid estimate is also written to the same location. This enables a pitch versus time plot to be created from the pitch-time pairs written to memory.

CONCLUDING REMARKS

Six second speech segments were obtained and analysed by the program, using both the application development system (ADS) and the DSP56001 simulator. In the case of the ADS, the sample values were read from files on the IBM PC-XT hard disk, and therefore did not run in real-time. Use of the simulator provided cycle counts for each program run, which may be related to real-time performance since the system clock frequency is known (20.5MHz). Cycle counts differ from one speech signal to the next, with female speech generally taking longer to process than male speech, since the pitch frequency is typically higher and consequently there are more pulses to characterize and compare. Simulations run with cycle counts

indicating run times from 412ms to 668ms, i.e. 7% to 12% of the total processing time available. These observations indicate that a single board, incorporating the DSP56001, appropriate memory, A/D and control circuitry could satisfactorily comprise a real-time implementation of the modified secondary feature PDA.

REFERENCES

- [1] W Hess, "Pitch Determination of Speech Signals: Algorithms and Devices", p5, Springer-Verlag, Berlin 1983
- [2] J Walliker, S Rosen & A Fourcin, "Speech Pattern Prostheses for the Profoundly and Totally Deaf", Proc. IEE Int. Conf. Speech Input/Output; Techniques and Applications, pp194-199, March 1986
- [3] L C Whitaker & D J B Pearce, "Larynx Synchronous Formant Analysis", Proc. Eur. Conf. Speech Tech., pp323-326, Sept. 1987
- [4] M R Varley, R J Simpson & T J Terrell, "On aspects of Automotive Noise and its effect on the performance of Pitch Determination Algorithms for Speech", I. Mech. E. 7th Int. Conf. Automotive Electronics, Oct. 1989 (to be published)
- [5] M R Varley, R J Simpson & T J Terrell, "On the Performance of Pitch Extraction Algorithms for Speech with Acoustic Noise", Proc. Speech '88, 7th FASE Symposium, book 2, pp621-628, August 1988
- [6] W H Tucker & R H T Bates, "A Pitch Estimation Algorithm for Speech and Music", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-26, no. 6, pp597-604, December 1978
- [7] J D Markel, "The SIFT Algorithm for Fundamental Frequency Estimation", IEEE Transactions on Audio and Electroacoustics, vol. AU-20, no. 5, pp367-377, Dec. 1972
- [8] DSP56000 Digital Signal Processor User's Manual, chapter 3, Motorola Inc. 1986
- [9] R J Simpson and M R Varley, "Digital Filtering Using the Motorola DSP56000/01: A Laboratory Based Case Study", Int. J. Appl. Engng. Ed. vol. 5, no. 3, 1989