Proceedings of The Institute of Acoustics

TMS320 KNOWLEDGE BASED PITCH DETECTION IN REAL TIME.

Zbigniew Gorzynski

Formerly: Department of Computer Science and Applications,
Queens University, Belfast.
Now: British Telecom, Systems Software Engineering Centre, Belfast.

INTRODUCTION

The design of pitch detectors in recent years has been influenced by the application of Artificial Intelligence in speech recognition and developments in low rate speech transmission coding. The extraction of several features and the use of pattern recognition principles in classification and decision making in pitch detection have been described by Dove [2]. At the same time, technological development in VLSI rendered dedicated DSP chips available for speech analysis. The project described in this paper was designed to combine these developments and to implement a real time pitch detector on a DSP chip. The detector incorporates the extraction of seven features from the speech signal, a decision module, and a gateway for future connection to the feedback path from a Speech Recognition Processor. Knowledge is built into the design on three levels. Level 1 - knowledge of speech and pitch mechanisms present in individual feature extraction algorithms. Level 2 - knowledge of performance aspects of the algorithms, gained from studies comparing pitch detection algorithms ([5], [6]). Level 3 - rules in the classification module. This combined knowledge allows the detector to make use of the features selectively in making its final decision.

The pitch detector was implemented on the TMS320. This processor was selected because of its excellent timing facilities and efficient instruction set. It has a speed of 5 MIPS (Million Instructions Per Second), an on board hardware multiplier that multiplies in a single 200nS cycle, and a host of instructions that combine several parallel functions, also in one cycle.

Speech is sampled at 8 kHz providing 125 uS between samples. The TMS executes 625 instructions in this time, equivalent to 100,000 instructions in a 20 mS frame.

Table 1. Family of TMS320 DSP chips

Type	Generation	Speed (MIPS)	Memory (words) Prog	Memory (words) Data	Power cons. (mW)	Extra features
TMS320 10	1st	5	4k	144	950	2 auto-pointers
TMS320 C10		5	4k	144	100	
TMS320 11		5	1.5k	144	950	
TMS32010-25		6.25	4k	144	950	
TMS320 20	2nd	5	64k	544+64k	1200	5 auto-pointers Autorepeat instruction
TMS320 C20		10	64k	544+64k	100	8 autopointers 8 deep stack FFT reverse bit addressing

When this project was begun, the author had access only to the TMS320 model 10, with all the corresponding development software and the XDS/22 emulator. This constrained the implementation to only 4 out of 7 features; the full version of the detector will be implemented when access is gained to the most recent version of the TMS320, ie model C20. This new version (Table 1) has new facilities which are particularly useful in the efficient implementation of autocorrelation and FFT [8], which are needed for the remaining features of the detector.

DESIGN

It was decided that at least three domains of pitch detection should be represented in the detector, namely time domain, autocorrelation and spectral analysis. From the numerous algorithms available, the following features were selected: periodicity factor and pitch frequency in the Gold-Rabiner scoring table, zero crossing rate of the full bandwidth signal, energy of the low pass filtered signal, AMDF performed on the LPC inverse filtered waveform, first coefficient of the LPC analysis, and a spectral comb.

Decision module. Having established the algorithms to be included in the design, the process of decision making was devised. This process assumes that all the extracted features are available simultaneously near the end of the frame. It processes the features according to decision rules based on the empirically derived probability density distributions for the individual features presented by Atal [1]. In the sequence of decisions, silence is determined first, and if detected the noise gate shuts off the output. Otherwise, the module determines the presence or absence of voicing. In the case of voiced frames only the pitch frequency value is then computed and output.

Architecture. The repetitive nature of speech sampling suggests at first the sequential, repetitive structure of a Real Time program, and obscures the fact that sampling actually triggers asynchronous activities within the program [4]. Consequently in the present project these activities were identified as separate asynchronous tasks, and their activation is controlled by a real time Executive task. These software tasks are divided into several groups, depending on the source of their activation. Clock tasks, which are driven by the sampling frequency, generate the relevant time-bases, eg frame length, decay and blanking in the peak detectors. A separate task performs all the computation in the foreground, and is activated on the arrival of each speech sample. Tasks activated by events include peak and valley analysis. A frame task, activated at the beginning of the frame, performs all the background processing, including the output. The initialisation and Executive tasks take care of householding functions and global control.

Communication between functional modules. The hierarchy of the system is shown in Fig 1. It illustrates how the tasks are decomposed into functional modules, limited by the inter-task communication boundaries. The flow of data between the modules is via the DCVs (Data Communication Variables), and the control information is passed between tasks and the Executive task via the CCVs (Control Communication Variables). Local variables are used internally by each of the functional modules. Example of this communication, as implemented in the peak detection part of the code, is illustrated in Fig 2. This clearly defined modular structure enables efficient coding and development of the program, and also allows real time visual debugging by monitoring the relevant Communication Variables.

IMPLEMENTATION

The techniques used in the final coding stage were selected for their time efficiency. Fig 3 shows how the processing is split in time between the foreground and background routines. In the foreground processing, the techniques are based on two principles:

a. removal of the overhead of conventional programming constructs, eg FOR, WHILE etc, instead writing the code explicitly, using in-line code and thread. The input filter serves as a good example of the in-line technique, where identical sections of code are explicitly repeated, rather than looped.

b. utilisation of particular facilities which increase the speed of operation of the chip, eg its specialised architecture and parallel instructions. In the example of the same filter two instructions (LTD and MPY) perform the following functions: load the sample to the multiplier register, multiply the sample by the coefficient, add the result cumulatively, and shift the sample, thus performing the delay operation.

In the background, where timing conditions are somewhat more relaxed, more processing takes place, and techniques related to multi-tasking and efficient communication between tasks are of primary importance. Although routines are written in assembly code, their design was based on a pseudo high level procedural language. In the context of nested procedure calls one important feature of the TMS is its hardware stack. Unfortunately, it allows only two levels for the user, which is far from sufficient. In this project therefore the stack was extended by means of two simple software routines. The hardware, so enhanced, can now call the nested subroutines, and this produces neat software modules in the background.

TESTING

Testing of the program during its development was done using an analogue function generator as the input, and displaying the variables on an oscilloscope screen. The behaviour of the program in real time could be monitored throughout all stages of development. Performance testing of functionally complete versions was also done with a special analogue signal source. This was considered more useful than testing with a series of computed data points or digitised waveforms stored in the memory test files. To provide a quantitative evaluation which enables meaningful comparisons between versions of the program, a special test bed and testing procedures were designed [3]. A Harmonic Generator was designed and built. It provided a signal which is similar to filtered speech, and therefore enabled quantitative measurements and comparisons to be made. This gave insight into the system's performance before the final tests were run on real speech. The Harmonic Generator provides a mix of the first three harmonic frequencies: F0, F2 and F3. The amplitudes and phases of the individual frequencies can be adjusted. The stability of the component frequencies is maintained by means of two PLL (Phase Locked Loops) controlling the three VCO's (Voltage Controlled Oscillators).

The main objective of this testing was to find the area of correct operation of the detector. Of special interest was how the detector performs when the fundamental frequency of speech is buried in strong adjacent harmonic components (eg due to the formant influence, or telephone filtering). Fig 5 shows the dynamic area of correct detection found from the tests of the current version of the detector, and Fig 6 the area of weak F0 detection in the presence of strong second and third harmonics.

Harmonic mix tests. In the first test a single frequency source shows that the area of correct detection lies within the rectangle outlined in Fig 5. The dynamic range of the signal amplitude is about 40 dB, where 0 dB corresponds to the maximum amplitude of the signal accepted by the A/D converter (10 V). The frequency range is approximately 60 to 300 Hz, which covers average male voices and lower female voices.

The second test comprises a series of measurements which were performed to evaluate the detector's ability to detect F0 from the mix of several harmonic frequencies. The main objective was to find the minimum amount of F0 sufficient for reliable detection. The following measuring procedure was adopted:

1. the Fo was increased in steps through its capture range of 60-300 Hz. The frequencies of the harmonics were locked with respect to the fundamental by means of the Phase Locked Loop mechanism in the harmonic generator.
2. for each of the fixed frequency steps, the proportion of the amplitudes was altered in steps, to cover the 0 to 100% contents of the F0 in the mix.
3. for each fixed proportion of amplitudes, the phase between the harmonics was swept from 0 to 360 deg.

It was found (Fig 6) that when F0 exceeds the critical values of 10% relative to F2 contamination, and 15% relative to F3 contamination, it is detected correctly. Below 2% there is a permanent false reading (pitch doubling and trebling respectively), and between these boundaries the pitch fequency detection 'flickers' between the correct and incorrect (doubled or tripled) value.

Speech tests. Live speech from a high quality dynamic microphone was preamplified, and the volume was adjusted manually so that the amplitude of the signal fell into the region of the correct dynamic range: between the minimum detection sensitivity, and the saturation of the ADC (10 V). Test results were registered on a graph plotter, which allowed tracing of natural speech sounds, including words and phrases. An example of the plotted pitch contour in continuous voicing (ascending vibrato) is shown in Fig 4.

FUTURE WORK

Plans for future work include the implementation on the TMS320 C20 of the three remaining features mentioned in the introduction, and their integration with the software already implemented. Future development of the test bed, and testing of the complete detector will be performed, and the reliability of the detection evaluated in various environments.

TMS320 KNOWLEDGE BASED PITCH DETECTION IN REAL TIME.

REFERENCES

1. Atal, B.S. & Rabiner L.R., 1976, A pattern recognition approach to Voiced-Unvoiced-Silence classification with application to speech recognition, IEEE, Vol. ASSP-24, No 3.

2. Dove. W. et al, 1983, Knowledge Based Pitch Detection, ICASSP 83, pt 28.6, 1348.

3. Gorzynski, Z., 1985, Pitch Detection in Real Time on the TMS320, unpublished MSc dissertation, Computer Science and Applications Department, Queens University, Belfast.

4. Gorzynski, Z., 1986, Real Time software engineering on the TMS320: application in a pitch detector implementation, Proceedings of the International Conference on Speech Input/Output: Techniques and Applications, London, 24-26 March, IEE Conference Publication No. 258, 270-275.

5. Oh, K.A. & Un, C.K., 1984, A performance comparison of pitch extraction algorithms for noisy speech, ICASSP 84, paper 18B.4.1.

6. Rabiner, L.R. et al, 1976, A comparative study of several pitch detection algorithms, IEEE, Vol ASSP-24, No 5.

7. Specker, P. 1984, A powerful post-processing algorithm for time-domain pitch trackers, ICASSP 84, paper 18.B.2.

8. Texas Instruments, 1986, Digital Signal Processing Applications with the TMS320 Family.

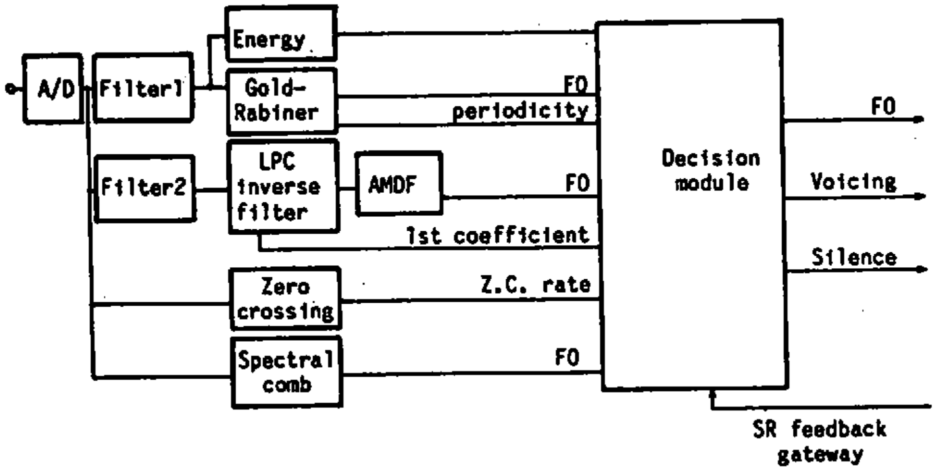

Figure 1 Hierarchy of the software system.

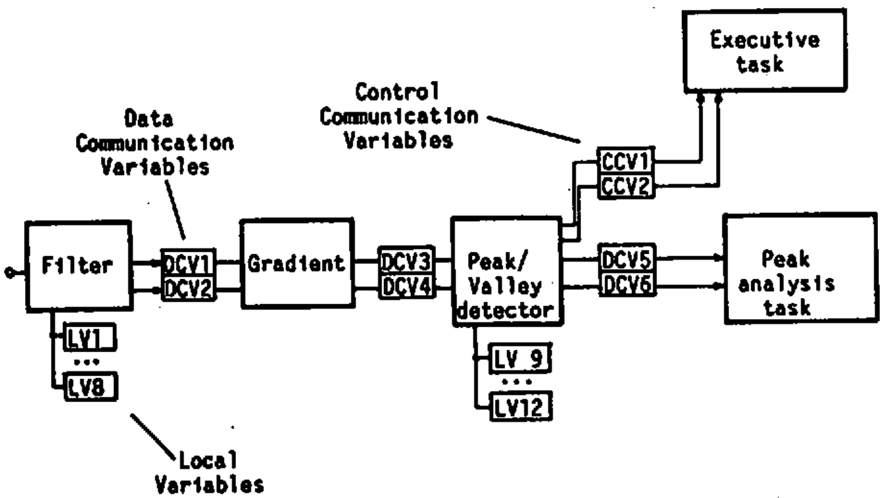

Figure 2 Example of communication between functional modules and the tasks.

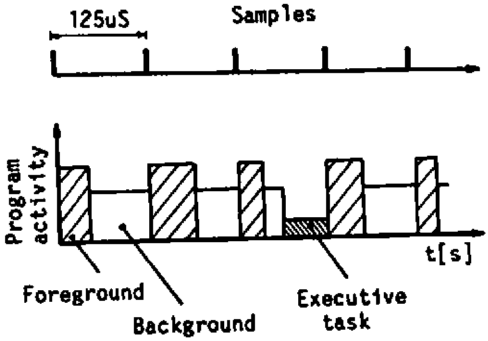

Figure 3 Interleaved execution.

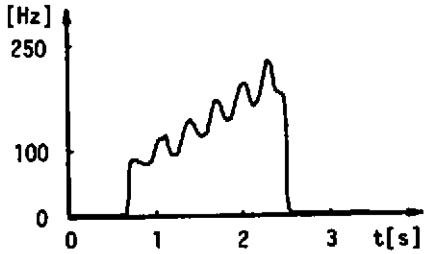

Figure 4 Example of detected pitch contour.

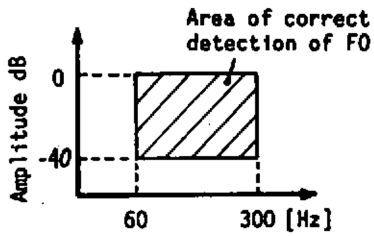

Figure 5 Correct detection of a single frequency

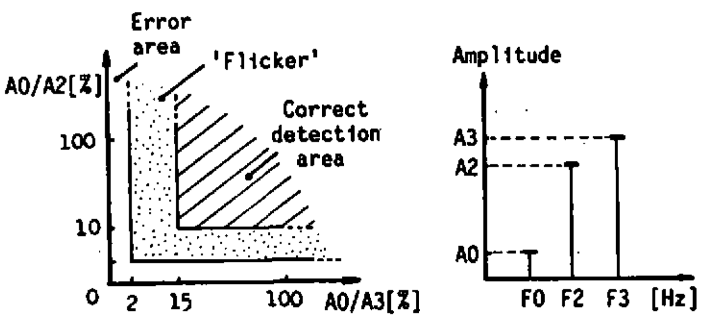

Figure 6 Detection of weak F0 in a harmonic mix.