On Design and Implementation of an Embedded Automatic Speech Recognition System

Sujay Phadke Rhishikesh Limaye Siddharth Verma Kavitha Subramanian
Indian Institute of Technology, Bombay
Dept. of Electrical Engineering
IIT Bombay, Powai, Mumbai, 400076, India.
{sujay,rhishi,siddharth,kavitha}@ee.iitb.ac.in

Abstract

We present a new design of an Embedded Speech Recognition System. It combines the aspects of both hardware and software design to implement a speaker dependent, isolated word, small vocabulary speech recognition system. The feature extraction is based on modified Mel-scaled Frequency Cepstral Coefficients (MFCC) and template matching employs Dynamic Time Warping (DTW). A novel algorithm has been used to improve the detection of start of a word. The hardware is built around the industry standard TMS320LF2407A DSP. The board is designed to serve as a general purpose DSP development board for the 24X series of TI DSPs. It contains, apart from the DSP, the external SRAM, FLASH, ADC interface, I/O interfacing blocks and JTAG interface. Both the hardware and the software have been designed concurrently, with a view to achieve high-speed recognition with maximum accuracy in minimum power and making the device portable. The proposed solution is a low-cost, high-performance, scalable alternative to other existing products.

1. Introduction

Speech Recognition has been an active area of research for many years. With advances in VLSI technology, high performance compilers, it has become possible to incorporate these algorithms in hardware. In the last few years, various systems have been developed to cater to a variety of applications. There are many ASIC solutions which offer small-sized, high performance systems. However these suffer from low flexibility and longer design cycle times. A complete software-based solution is attractive for a desktop application, but fails to provide a portable, embedded solution. High-end Digital Signal Processors (DSPs) from companies like TI, Analog Devices, provide an ideal platform for developing and testing algorithms in hardware. The advanced software tools like C-compiler, simulator and debugger provide an easy approach to optimize the algorithms and reduce the time-to-market. However, in order to gain maximum advantage, the hardware and the software have to be designed hand-in-hand.

Speech recognition is either speaker independent or dependent [1]. Speaker independent mode involves extraction of those features of speech which are inherent in the spoken word. This class of algorithms is generally more complex and makes use of statistical models and language modeling. On the other hand, speaker dependent mode involves extracting the *user-specific* features of the speech. A template of extracted coefficients of words has to be created for every user and the matching is done to determine the spoken word. Furthermore, using isolated words rather than a complex continuum of words helps in increasing the accuracy of recognition. Our work involves development of a speaker dependent, isolated word speech recognition system. The system is capable of recognizing a spoken word from a template of 10-15 words. It has a high recognition accuracy and a modest rejection ratio.

This paper is organized as follows. Section II deals with the software part. It explains the theory behind Mel cepstrum based coefficient extraction and Dynamic Time Warping techniques, which form the basis of the application. Section III describes the custom hardware developed for this application and the various design issues related to it. Software optimization and porting of C-code to the DSP platform is discussed in section IV. The results and comparisons are explained in section V. Finally we conclude with potential applications of the system in section VI.

2. Software

This section presents the software aspects used in the speech recognition engine. The theory of MFCC is ex-

Proceedings of the 17th International Conference on VLSI Design (VLSID'04)
1063-9667/04 $ 20.00 © 2004 IEEE

plained followed by its implementation. A novel start detection and wrong word rejection algorithm developed by the authors is also presented. It concludes with the Dynamic Time Warping (DTW), the template matching algorithm used for recognition.

2.1. Feature Extraction – Mel-scaled Frequency Cepstral Coefficients (MFCC)

The feature extraction involves identifying the *formants* in the speech, which represent the changes in the speaker's vocal tract. There are many approaches used viz. Linear Predictive Coding (LPC), Mel-scaled Frequency Cepstral Coefficients (MFCC), Linear Prediction Cepstral Coefficients (LPCC), Reflection Coefficients (RCs). Among these, MFCC has been found to be more robust in the presence of background noise compared to other algorithms [2]. Also, it offers the best trade-offs between performance and size (memory) requirements.

The primary reason for effectiveness of MFCC is that, it models the non-linear auditory response of the human ear which resolves frequencies on a log scale [3]. The mapping from linear frequency to mel frequency is defined as,

$$Mel(f) = c \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{1}$$

To capture the auditory frequency content usefully, speech signal is best passed through a filter bank consisting of overlapping triangular filters called the Mel Filter Bank. On the Mel scale, the center frequencies of these filters are linearly spaced and the bandwidths are equal. The mel scale is often approximated by a linear scale for $f < 1$ KHz and logarithmic afterwards. Thus we get the following approximation of the Mel filter bank —

$$U_m(k) = \begin{cases} 1 - \frac{|k-c_m|}{\Delta_m}, & |k-c_m| < \Delta_m \\ 0, & |k-c_m| \geq \Delta_m \end{cases} \tag{2}$$

$$c_m = c_{m-1} + \Delta_m \tag{3}$$

$$\Delta_m = \begin{cases} 4, & f < 1KHz \\ 1.2 \times \Delta_{m-1}, & f \geq 1KHz \end{cases} \tag{4}$$

where k is the DFT domain index, $2\Delta_m$ is the bandwidth and c_m is the central frequency of the m^{th} filter in the bank of size M.

The m^{th} energy coefficient for n^{th} frame of input signal $X(k)$ given by:

$$Y_n(m) = \sum_{k=c_m-\Delta_m}^{c_m+\Delta_m} X_n(k) \cdot U_m(k) \tag{5}$$

The logarithm of the magnitude of each of these energy coefficients is taken to account for the logarithmic relation of intensity and loudness. These log energy coefficients so obtained are then orthogonalized by using inverse DCT (IDCT) [3]. The resulting parameters are called Mel-scaled Frequency Cepstral Coefficients (MFCC). Mathematically, this is as follows:

$$y_n(j) = \sum_{m=1}^{M} \log|Y_n(m)| \cos\frac{j\left(m-\frac{1}{2}\right)\pi}{L}, j = 0, 1, \ldots, L \tag{6}$$

We use 16 filter banks (M = 16) and L = 15 which is the final number of coefficients per frame of input signal.

2.2. Implementation of MFCC

The software implementation of the feature extraction is depicted in Fig. 1. Each of the steps is explained further in the following text.

2.2.1. Sampling. The sampling frequency of 8 KHz is sufficient for human speech. This frequency gives the window of 125 μs between the two consecutive samples. Thus a sizable part of the processing can be done in real time. A higher frequency would shrink this window and would also demand greater memory and higher processing time. For a word of duration of 0.5 s, the number of samples at 8 KHz is 4000. With each sample stored as 16 bit value, it amounts to about 8 KB of storage space.

2.2.2. Start Detection. Detection of start of an utterance in the presence of background noise is non-trivial. There are two issues – (i) avoiding false detections triggered by background sounds, and (ii) accurately capturing the first syllable of a word. A novel scheme is presented here which is able to effectively address the two issues. The scheme continuously fetches the audio samples and maintains a sliding window of the past N samples at every point of time. The window size N is a design parameter. The scheme uses the following two criteria based on the sliding window.

1. *Energy* – A running average of the energy content E_s of the window is maintained. To be recognized as the start of an utterance, this average must exceed a *threshold value* which is set at a constant multiple C of the energy average, E_n, of the background noise. The constant C is a customizable parameter and can be adjusted to suite individual speaker's characteristics.

2. *Zero crossings* – Empirically, it is found that noise has very few zero crossings as compared to normal speech. This fact is exploited to strengthen the detection process. Similar to energy, number of zero crossings Z_s of the input signal in the sliding window should exceed a threshold value Z_t which is predetermined by experimentation. This criterion helps in detecting the first

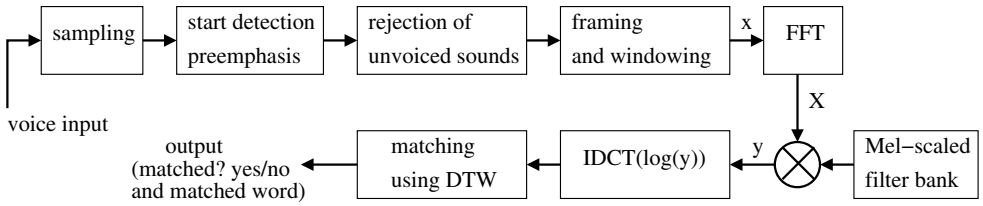

Figure 1. Algorithm Flowchart

syllable of a word accurately even if its energy content is not sufficient to fulfill the energy criterion.

A combination of the above criteria leads to a simple yet powerful start detection scheme which is implemented in real time. The calculation of noise energy average E_n is performed on the system start-up and then periodically whenever the system is in idle mode (i.e. – when no start of utterance is detected). Thus the threshold dynamically adjusts to the changing noise environment.

Upon detection of start of an utterance, the samples in the sliding window and the subsequent samples are stored in the *word buffer*. We store the samples for a fixed duration of 0.5 s which sufficient for most of the typical words. Ideally there should be an *end detection* mechanism as well. But it is difficult to have an accurate, causal end detection scheme. In the subsequent text, we present our own non-causal end detection scheme.

2.2.3. Preprocess. Firstly the amplitude of the input signal is normalized to remove the effect of varying intensity. The spectral characteristics of speech at higher frequencies are subdued in relation to the lower frequencies. In order to enhance their weight in the extracted parameters, we apply a pre-emphasis filter which is a first order high-pass filter described by:

$$y_n = x_n - 0.95x_{n-1} \tag{7}$$

2.2.4. Rejection of unvoiced sounds. Although the above scheme is very effective in discarding the background sounds, certain random sounds (e.g. – clicks, chirps, air blowing) which have sufficient energy, also satisfy the start-detection criteria. However such a sounds can be distinguished by adding few more checks. These checks are executed on the word buffer.

End Detection — This is a novel, *non-causal* technique for determining the end of the word. It basically runs the start-detection algorithm backward in time starting from the end of the word buffer. This technique accurately prunes the silent part towards the end of the word buffer and determines the exact length of the word. If the length is too small, then the word is considered invalid. This eliminates single clicks and chirps.

Excessive Silence — The word buffer is divided into a set of frames and average energy content of each frame is computed. If a significant fraction of the frames have relatively low energy, then it indicates that it is made up of narrow isolated peaks of sound – not a characteristic of a valid word. This eliminates multiple clicks and chirps.

Excessive Energy — Similarly if a significant fraction of frames has relatively high energy, then it represents a continuous unmodulated sound (e.g. – blowing into the mike or mechanical vibrations). Thus the word is declared invalid. It is also important to restrict the maximum amplitude to avoid overflow errors in the subsequent processing. The sensitivity of the mike is adjusted so that normal speech does not give this error.

The qualitative conditions stated above are implemented by having quantitative parameters whose values are adjusted so that the normal speech is passed forward while the unwanted sounds are eliminated.

2.2.5. Framing and windowing. Input speech is divided into a number of *overlapping* frames – each of size 256 – and the *picket-fences* effect is minimized by applying hamming windowing function, as described below, on each frame.

$$w(n) = \alpha - (1-\alpha)\cos\left(\frac{2n\pi}{N-1}\right), \quad \alpha = 0.54 \tag{8}$$

2.2.6. FFT. On each frame, we apply 256 pt FFT as given in [4]. The algorithm runs *in-place* thus conserving on the memory requirements. This converts the stored word into its frame-by-frame DFT.

2.2.7. Application of Mel Filter Bank. On each frame, the Mel filter bank is applied as discussed in section 2.1. The IDCT is taken by using the inverse FFT algorithm as given in [4]. We get, for each of the 20 frames, a set of 15 MFC coefficients.

2.3. Template Matching – Dynamic Time Warping

After feature extraction of a spoken word, we get a frame-wise sequence of feature vectors. The next step is

to compare it with set of stored templates for the current user. We use a popularly used technique called Dynamic Time Warping (DTW) [5]. It is a technique which "warps" the time axis to detect the best match between given two sequences.

For the spoken word, S, let S_i^k denote the k^{th} coefficient of the i^{th} frame. The DTW comparison of S and a template word T starts with calculation of a *Local Distance Matrix* of size 20×20 where each entry LD_{ij} is given by,

$$LD_{ij} = \sum_{k=1}^{15} \left(S_i^k - T_j^k\right)^2 \quad (9)$$

Thus the local distance LD_{ij} is the vector distance between the corresponding coefficients of i^{th} frame of the spoken word and the j^{th} frame of the template word.

We then find a minimal warping path through the local distance matrix. The corresponding warping cost gives the DTW distance between the two sequences [6]. The minimum warping cost can be found efficiently using dynamic programming having the following recurrence:

$$D_{ij} = LD_{ij} + \min\left\{D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}\right\} \quad (10)$$

where, D_{ij} is the cumulative distance and $D_{20,20}$ gives the DTW distance.

Thus, DTW distance is calculated between the spoken word and each of the template words. The template word having the least distance is taken as a correct match, if its distance is smaller than a predetermined threshold value.

3. Hardware

The software is implemented on a custom-designed board built around the TI TMS320LF2407A DSP. This section describes the hardware design in terms of the important design considerations which are instrumental in achieving the maximum efficiency.

3.1. Processor

The central processor, TMS320LF2407A, is part of the TMS320C2000 platform of fixed-point DSPs [7]. It offers the enhanced TMS320x DSP architectural design of the C2xx core CPU for low-cost, low-power, and high-performance processing capabilities. Several advanced peripherals have been integrated to provide a true single-chip DSP controller. Among the code-compatible C24x DSP controller family, TMS320LF2407A offers the highest processing performance (40 MIPS) and greater degree of peripheral integration. JTAG-compliant scan-based emulation provides non-intrusive real-time programming and debugging capabilities.

The use of high-end DSPs like the TI 6x family is uncalled-for because (i) the application doesn't require that high processing power, and (ii) the total cost of the system has to be kept minimum. The high-end processors would require complex hardware and advanced PCB fabrication techniques. Due to all these factors, the TMS320LF2407A is the ideal choice for our application.

3.2. Memory

The TMS320LF2407A employs the harvard architecture with 64K $\times$ 16 words in each of the program, data, I/O spaces. The processor, internally, has 32K of program FLASH, 544 words of register bank and 2K words of RAM. This on-chip memory, when enabled, occupies some of the off-chip memory space.

The board uses an external 64K $\times$ 16 SRAM with its lower 32K words in the lower program space and upper 32K words in the data space. The program area of the SRAM optionally substitutes the internal program FLASH. This is particularly convenient during the code development and testing, where burning the internal FLASH again and again is uncalled for. The data area of the SRAM provides general data memory.

The board also has an external 64K $\times$ 16 FLASH of which, 32K words are configured to be in the upper program space. This provides an extension to the processor's internal program FLASH. The external FLASH can also be used as general purpose non-volatile storage (e.g. – storage of user templates). The Atmel FLASH chosen, features easy programmability with small sector size and in-built erase-write sequence.

The combination of on-chip memory and external SRAM, FLASH exploits the program and data memory spaces to fullest possible extent. Thus making the board suitable for general purpose development.

3.3. Analog to Digital Converter (ADC)

The TMS320LF2407A has an in-built, high-performance, 10-bit analog-to-digital converter (ADC) with a minimum conversion time of 375 ns. We tested out the accuracy of recognition by varying the resolution of the incoming raw samples. It was found that a 9-bit A/D provides sufficient recognition accuracy, without any adverse effect of the quantization error. The signal integrity issue of the audio signal is tackled by separation of analog and digital nets in the board design.

3.4. Power Supply Management Circuitry

The TMS320LF2407A is a low power chip with the main supply voltage of 3.3 V. A separate 5 V supply is

needed for programming the internal FLASH. The board uses two *low-dropout voltage regulators* from TI to obtain stable 3.3 V and 5 V from external 9 V. They have the current capacity of 1 A. The fidelity of the voltages is maintained by an on-board *dual supervisor* which monitors both the supply voltages. The board is resetted whenever any of the supply voltages falls below 98% of their expected values. Memory chips also operate at 3.3 V. To facilitate the interface with external TTL peripherals, a bidirectional *level shifter* (3.3 V–5 V) is used.

4. Software Optimizations

The code is developed and tested using the TI C compiler. In order to achieve the maximum performance, in addition to the compiler optimizations, several techniques are used in the code design. These are explained in this section.

4.1. Hardware API

The code which handles the hardware aspects like data acquisition, interrupt management, and other peripheral control is designed to hide these details from the main algorithm, thus making the design more modular. The API contains several utility functions and macros.

4.2. Memory Management and Variable Allocation

The on-chip memory of TMS320LF2407A allows full-speed execution, but it is smaller in size. The external memory provides much larger size but may degrade the performance. Use of the internal memory also reduces the power consumption. Hence the variables have to be allocated appropriately depending on their frequency of access and size requirements.

The C environment stack, which is accessed during every function call, is the most accessed memory part. Hence it is kept in the internal RAM. The stack is also used for allocating the local variables in a function, thus to avoid stack overflow, large arrays are kept global. The scratch-pad variables, e.g. – loop indices, are kept in the internal register bank.

The large arrays, e.g. – word buffer, have to be kept in external SRAM. But for the extraction of coefficients of each frame, the frame is first copied into the internal RAM and the entire extraction process for that frame runs *in-place*. The final set of features for the entire word are also stored internally.

Since FFT requires the use of trigonometric functions, look-up tables of sin, cos are maintained. The table size is modest as we require only some specific values. The Mel filter bank is also stored statically though it can be calculated analytically. The memory overheads incurred are well compensated by the performance gains.

DTW comparison function consists of nested loops and accesses the memory very frequently. Hence, before starting DTW between the spoken word and a template word, the features of the template word are copied into the internal RAM.

4.3. Use of integer data type

The TMS320LF2407A is most-suited for 16-bit integer arithmetic. Hence the entire dataflow is adapted to work in integers. The overflow is avoided at all places without compromising on accuracy. At places where error becomes unacceptable, floating point arithmetic library is used. The raw samples are scaled down to 8 bits to avoid overflow during the 256-point integer FFT. The trigonometric look-up tables, filter banks are also scaled appropriately.

4.4. Functions, stack, heap

In order to reduce function calling overheads, most of the functions were collapsed into a single function. Since the C environment uses a stack space to store return values and pointers, this optimization cut down on much of execution time. The use of static arrays eliminated the need for large dynamic memory allocation, thereby cutting down on the heap size.

5. Results and Comparisons

The techniques employed here provide high accuracy for the recognition of isolated words. Mel cepstrum based technique improves considerably over simple energy-based approach. This is shown in Fig. 2 which shows distances of the word “two” with each of the template words. In the energy-based approach, the distances are not easily distinguishable and hence it is difficult to set an acceptance threshold in this case. In case of Mel-based technique, there is a sharp difference between the distance of “two” with template of “two” and with the others. Thus it is easy to set the threshold in this case.

We tested the performance of this system for a large pool of users with different sets of words. Both male and female subjects were used and words were chosen which are well separated in frequency domain. (Words like “light”, “might”, “sight” which differ only in the first syllable are not too well distinguished).

The experiments show that common words like English digits – “one”, “two”, “three”, . . .), or words like “light”, “fan”, “tube”, etc. give excellent results as far as recognition accuracy is concerned. This is true for both male

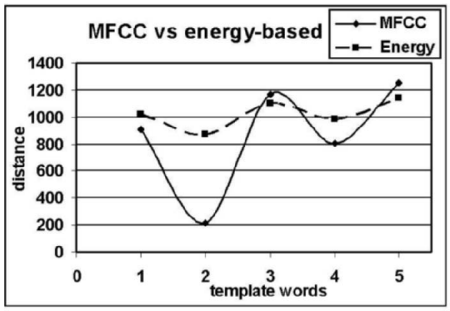

Figure 2. Distance of word "two" using i) Mel-based and ii) energy-based approach

Table 1. Recognition accuracy for english digits

	"one"	"two"	"three"	"four"	"five"
Male 1	99.8	92.5	99.9	90.5	95.6
Male 2	99.2	91.8	99.6	89.5	96.0
Female 1	99.0	88.9	99.4	89.3	94.3
Female 2	98.9	89.7	99.3	88.6	95.1
mean	99.23	90.73	99.55	89.48	95.25

and female subjects; however it has been observed that for certain words, the recognition accuracy is less for females. This is shown in Table 1.

6. Conclusions and Applications

This paper discussed the implementation of a low cost, robust embedded speech recognition system. The algorithm discussed is memory efficient, scalable and provides sufficient accuracy of recognition. The custom DSP board developed is a power efficient, flexible design and can also be used as a general purpose prototype board. The ingenious start detection technique and optimized speech recognition algorithm can be easily ported to a variety of embedded platforms. Some of the typical applications of the system are as follows –

- A voice enabled interaction device for physically handicapped persons.
- Low cost voice enabled switches for households.
- Unit fitted into automobiles, aeroplanes could serve as a voice activated interface.

This design can be improved further, both at the hardware as well as software levels. A faster DSP with floating point capabilities, extended memory addressing space and an audio codec (in place of the in-built ADC) would increase the performance of the system with respect to speed and accuracy. On the software side, several sophisticated algorithms exist which have a better recognition accuracy and rejection rate, and also support continuous time recognition. The most common of these are algorithms using Hidden Markov Models (HMMs), phoneme-based speech modeling and language modeling. However, these are ideally suited for very large databases (50,000 words or more) and require elaborate training. They are more demanding on the hardware and would require the use of high-end DSP platforms.

References

[1] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, 1st ed., ser. Prentice Hall Signal Processing Series. Prentice Hall Professional Technical Reference, Apr. 1993.

[2] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[3] H. Combrinck and E. Botha, "On the mel-scaled cepstrum," department of Electrical and Electronic Engineering, University of Pretoria.

[4] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C - The Art of Scientific Computing*, 2nd ed. Cambridge University Press, Feb. 1993.

[5] M. Brown and L. Rabiner, "Dynamic time warping for isolated word recognition based on ordered graph searching techniques," in *Intl. Conf. on Acoust., Speech, Signal Processing, ICASSP'82*, vol. 7, May 1982, pp. 1255–1258.

[6] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," department of Information and Computer Science, University of California, Irvine.

[7] *TMS320LF2407A DSP Controllers Datasheet (SPRS145I)*, Texas Instruments, July 2000, revised Sep. 2003. [Online]. Available: http://focus.ti.com/lit/ds/symlink/tms320lf2407a.pdf