2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India

Real Time Speech Recognition using IIR Digital Filters Implemented on an Embedded System

Bhavin Kamdar [1], *Bhisham Mirchandani*[1], *Dhaval Shah* [1]
bhavin.kamdar.28@gmail.com, bhishamm@gmail.com, dhavalshah.1089@gmail.com
1. U.G. Student, Electronics and Telecom. Department, Sardar Patel Institute of Technology, University of Mumbai, Mumbai 400058, India

Dr Y. S. Rao [2],
ysrao@spit.ac.in
2. Head Of Department, Electronics and Telecom Department, Sardar Patel Institute of Technology, University of Mumbai, Mumbai 400058, India

***Abstract*—In this paper we implemented real time speech processing and recognition using Infinite Impulse Response (IIR) digital filters. We used an array of 7 digital bandpass filters to filter out various frequencies and recognize each word by its frequency spectra. The system is implemented on a low cost AVR architecture microcontroller which also explored the use of simple architectures as real time DSP processors. This opens up new vistas for the AVR applications and also provides a low cost solution for small and medium DSP applications.**

***Index Terms*—Speech recognition, template matching, digital filter, Infinite Impulse Response (IIR), correlation.**

I. Introduction

Speech Recognition, or the process of automatically recognizing speech uttered by human users just a human listener would, is one of the most intriguing areas of speech processing. Indeed,it can be the most natural means for a human to interact with any electronics-based system as it is similar to how people receive and process a large proportion of information in their daily lives. The complex waveforms produced when one speaks is the result of the intricate human speech prduction system. [1] provides details of how this biological system functions as well as how it can be modeled in a form appropriate for digital systems.

There are mainly two methods to implement speech recognition: template matching and hidden markov model(HMM) [1] [2]. The latter necessitates modelling of speech and less reliance on data. While this is preferable for continuous speech recognition and speaker independent speech recognition, the authors of [2] posit the use of template matching when these are not application requirements. The authors note that an intrinsic advantage of template based recognition is that we do not need to model the speech process. This is very convenient, since our understanding of speech is still limited, especially with respect to its transient nature.

We, therefore, view the speech recognition algorithm as a Pattern Recognition problem: detecting a pattern of uttered speech to be a close match to prespecified reference patterns, or templates of corresponding utterances.

In order to implement the template matching algorithm, one may simply find the discrete fourier transforms (DFT) of two speech utterances and evaluate the correlation coefficient of the DFTs. However, this approach requires the abundant availability of memory. In an embedded environment, it is preferable that the algorithm perform even if the availability of RAM is low [1]. We have, therefore, divided the vocal frequency range into 7 sub-ranges according to the recommendations of [3]. We have used a digital filter for each sub-range. Each filter is followed by an accumulator to give the magnitude of the signal contained in that frequency range. This gives us an array of 7 numbers that uniquely define each speech utternace. By finding the correlation coefficient of any two arrays we have classifed the degree of similarity between the utterances (i.e. words). We have recorded an accuracy of 80% for distinctly different words and 30% for highly similar (correlated) words. Since each word is defined only by the array of 7 numbers, it leads to a memory efficient and inexpensive design of embedded system.

A number of applications, ranging from medical dictation to mobile telephony, are related to speech recognition [4]. Hence, we have developed a stand-alone embedded system for performing speech recognition. In the context of embedded systems, the result can often be the detection of specific commands spoken by the end-user of the system. The number of such commands is dictated by the needs of the application as well as the capabilities and memory size of the processing hardware platform being used to implement the system. We have developed an embedded system consisting the ATmega32 as the microcontroller and other components for input (microphone, amplifier, tactile switch) and for output (LCD display). The microcontroller is programmed with the template matching algorithm.

II. Methods of Speech Recognition

There are broadly two methods of speech recognition which are broadly recognized. Each method has many variants and dialects but the core concept remains the same. Selecting and

978-1-4577-2078-9/12/$26.00©2011 IEEE

implementing a particular method depends on various factors such as the accuracy required,computational time available, computational memory available, etc. The methods are briefly explained below.

A. Speech Recognition using Template Matching

One of the early and well-understood methods for recognizing words is by breaking up the uttered speech into its constituent phonetic units or Phonemes. The phonemes, in turn, are divided into short time windows or frames, similar to what we have seen for other speech processing algorithms. In this methodology, it is assumed that when the same person utters the same word on different occasions, the speech waveform does not vary considerably,except for the whole word or parts of it being spoken faster or slower. Therefore, these algorithms utilize a set of prestored reference waveforms,or Templates, for specific words spoken by each speaker. Subsequently, during actual operation of the system, when the user speaks into the device, the sampled waveforms are decomposed in a similar manner and compared with all the reference templates. Whether all such templates would be compared or only the ones corresponding to a given word or a given speaker depends on the specifics of the application.

Now, for each frame (which is typically 10 to 20 ms), instead of directly using the waveform samples, the algorithm may be represented as a set of parameters or Feature Vectors. These can be represented in a multidimensional vector space,and the vector distance between each Feature Vector of the incoming speech frame and every Feature Vector in the templates is calculated as shown in [1]. Generally, greater weight is given to those features that represent the phonetics of the uttered speech rather than the specific subtleties of the person's speech. This method has been developed and implemented in [2]. The authors have found results with high accuracy.

B. Speech Recognition using Hidden Markov Model

Instead of a fixed template match that provides the minimum distance metric along an optimal path from the first frame of a word to the last, an alternative methodology would be to develop a statistical representation,or Model, that would produce sets of Feature Vectors such that the statistical properties over all these sets would be similar to that of the uttered speech. The ultimate objective in this case is to determine which word model would have been most likely to have produced the speech feature vectors that have been observed; this is a case of minimizing an a posteriori probability (since the speech frame has already been sampled).

In this methodology, the successive template frames of each word is thought of as a sequence of states; in fact,a single such state can contain multiple frames. For simplicity, this entire sequence of states culminating in the final output (last frame) is considered as a Stochastic Process,that is,a non-deterministic process in which each possible state transition is defined by statistical probabilities.

III. Method Implemented

In this project, we have implemented a variant of the template matching method described in [2]. This method was selected because the AVR microcontroller is an 8 bit 16 MHz microcontorller with 2 kbyte of on chip SRAM. Hence we had limitation on the amount of samples it can store and the time in which it can process those samples. Also we did not want to interface any external memories to the system to emphasize its use as a low cost stand alone system for Digital Speech Processing. And the implementation of HMM method is computationally heavy. Hence the template matching method was selected.

IV. System Overview

The System consists of a ATmega32 AVR microcontroller at its heart. There are 7 bandpass digital IIR filters implemented in the microcontroller which operate in real time. The speech input is given through an op amp pre-amplifier and converted to digital form using the on chip 10 bit ADC in ATmega.

The system builds of dictionary of templates by passing the speech through the bandpass filter array and decomposing it into individual frequency spectra and magnitude related to each range. When the word to be recognized is spoken, it is decomposed in similar fashion and compared with the dictionary. The template which matches the most is selected as the recognized word.

V. Digital Filters

A digital filter is characterized by its transfer function, or equivalently,its difference equation. Mathematical analysis of the transfer function can describe how it will respond to any input. As such, designing a filter consists of developing specifications appropriate to the problem (for example, a second-order low pass filter with a specific cut-off frequency), and then producing a transfer function which meets the specifications [7] [8].

The transfer function for a linear, time-invariant, digital filter can be expressed as a transfer function in the Z-domain; if it is causal, then it has the form:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}} \quad (1)$$

where the order of the filter is the greater of N or M.

This is the form for a recursive filter with both the inputs (Numerator) and outputs (Denominator),which typically leads to an IIR infinite impulse response behavior,but if the denominator is made equal to unity i.e. no feedback,then this becomes an FIR or finite impulse response filter. Generally for a second order bandpass filter, the transfer function can be given as

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (2)$$

The above equation can be realised in Direct two form as shown in Fig 1.

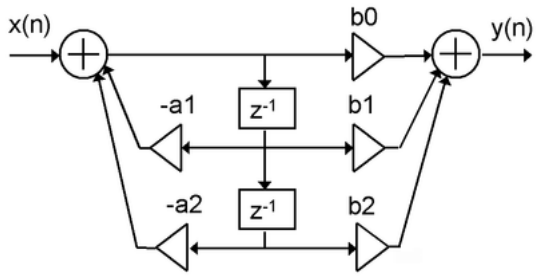

Fig. 1. Realisation of a second order IIR filter in DF II form

A. Filter Simulation

This project uses a Chebyshev second order bandpass filter. The Digital Filter design has been tested on Matlab by making a basic GUI which implements similar 7 bandpass filters and passing a test signal to the filters. The filter ranges are selected according to the bark scale as suggested in [3] and are given in Table I.

TABLE I
FILTER RANGES

Filter 1	100Hz to 200Hz
Filter 2	200Hz to 300Hz
Filter 3	300Hz to 400Hz
Filter 4	400Hz to 510Hz
Filter 5	510Hz to 630Hz
Filter 6	630Hz to 770Hz
Filter 7	770Hz to 920Hz

The output waveform of each filter represents the magnitude of the input signal in the frequency range for which the filter is designed. Hence in our algorithm, the filter is followed by an accumulator. The filter output for each sample can therefore be treated as a temporary variable and be immediately added to the accumualtor output. This is shown in the iteration in table II where, the variable tmp (the output of the filter) is added to y1 (the output of the accumulator).

The filters were tested by feeding the microphone connected to the PC with a fixed frequency sine wave sound generated by the PC itself. This sound was captured and passed as input to the filter program. In the program, the filter co-efficients were generated by using the Matlab Filter design Tool. The code for a sample filter is explained in [6].

The filters where tested for three different frequencies and it gave proper output on all the cases as shown in Fig 2. Hence the filter design was verified.

B. Filter Implementation on AVR

The filters design by the above method requires floating point arithmetic. The AVR architecture does not contain a dedicated FPU hence floating point calculations take multiple cycles to execute. But to implement the system in real time, all the seven filters required to be executed before the next sample was acquired. Hence this put a limit on the sampling frequency as a higher sampling frequency meant lesser time

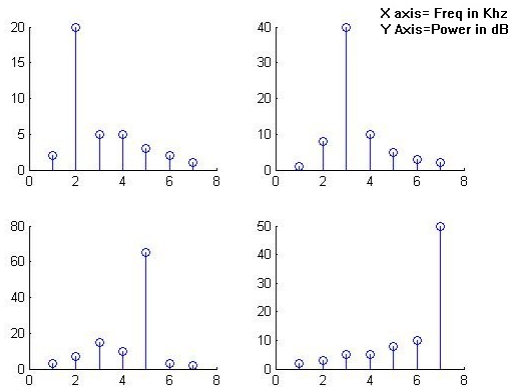

Fig. 2. Filters tested with frequencies of 250Hz,350Hz,570Hz and 850Hz

between samples to implement the filters. But the sampling frequency cannot be made too low as that would cause aliasing of signals. Hence a compromise had to be done. We used Matlab to plot the Fourier transform of various words. The Fourier transforms of words ”One”,”Two”,”Stop” and ”Go” is shown in Fig 3. From the transforms we concluded that human speech power is concentrated in the frequency range of 100Hz to 1000Hz and contains very less energy less outside this. Hence according to Nyquist theorem [5] any sampling frequency above 2000Hz will be suitable. To keep a buffer and according to the internal timer constants of the ATmega,we selected a sampling frequency of 3731Hz.

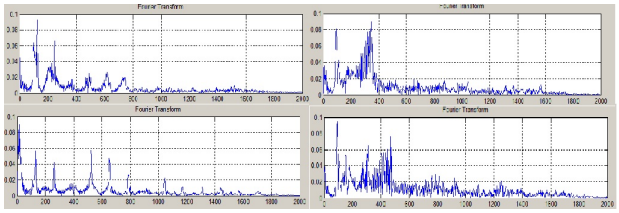

Fig. 3. FFT of Words ”One”,”Two”,”Stop”, and ”Go” in clockwise manner

Each filter was implemented by the following equation in the AVR.

TABLE II
FILTER KERNEL CODE IN AVR

```
v12=smpl[ucIndex]-(A11*v11)-(A12*v10);
temp=(B10*v12)+(B11*v11)+(B12*v10);
y1=y1+fabs(temp);
v10=v11;
v11=v12;
```

Here A11, A12, B10, B11, B12 represent the filter co-efficients which are acquired from Matlab. v12,v11 and v10 are present and past intermediate outputs of DF 2 form realization. smpl is an array storing the input values and y1 acts as an accumulator which stores the absolute value of each filter output. This filter kernel is repeated 7 times with seven different filter co-efficients set and intermediate output

variables. The variables are not defined as an array as it would have decreased program size but array indexing and retrieving increases execution time.

1) AVR Filter simulation: The filters were also tested in the similar manner in which Matlab filter design was tested. However before interfacing an audio circuit to the microcontroller, the filters code was simulated on AVR Studio 4 to check the time constrains. The filter was excited by internally generated sine wave of a definite frequency and the code was executed. It was found that with a crystal of 8 Mhz, the μc could execute only 4 filters in between two consecutive samples of 250 μs. Hence we increased the clock to 16 Mhz. After simulating and getting the desired results,the program was loaded on the system and tested in real time. This approach saved us much of debugging time as many run time errors got rectified before in the simulation mode

VI. Template Matching Algorithm

The filter output is accumulated and averaged to generate a template of 7 vectors. These vectors are stored in the internal EEPROM of the ATmega. In recognize mode, the word is decomposed in similar manner and it is compared against the stored templates. For template matching we tried various algorithms which are mentioned below

A. Eucledian Distance

This calculates the distance between the template and acquired word and the template with minimum distance is taken as the matched word. The euclidean distance formula is given as

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}. \tag{3}$$

Due to computational weight of the square root function,we used only the sum of squares. But this did not give proper output always and depended heavily on the magnitude of the filter outputs. Hence it was intensity dependent which was undesirable feature. However it could had been tuned by normalizing the filter outputs, it was not very accurate.

B. Correlation

A correlation coefficient is a single number that describes the degree of relationship between two variables. It is given by

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\ \sqrt{n\sum y_i^2 - (\sum y_i)^2}}. \tag{4}$$

It ranges from -1 to 1. If both the waves are same, the there correlation will result in 1. Hence higher correlation coefficient signifies a higher degree of similarity. Also it is independent of magnitudes and phase of the waves. Hence the comparison will not be affected by the intensity of the speaker. The simulation results by correlation were found to be more accurate. Hence we used correlation as the template matching method in our final code.

VII. Hardware Design

The hardware consists of a pre-amplifier and a microcontroller circuit. The amplifier is used to take input from the microphone and amplify it. In our circuit LM358 is a dual op amp. It is configured in non inverting mode with a ac gain of about 40. The gain cannot be increased further as the output saturates because the op amp does not feature rail to rail output. The circuit for pre-amplifier is shown in Fig. 4

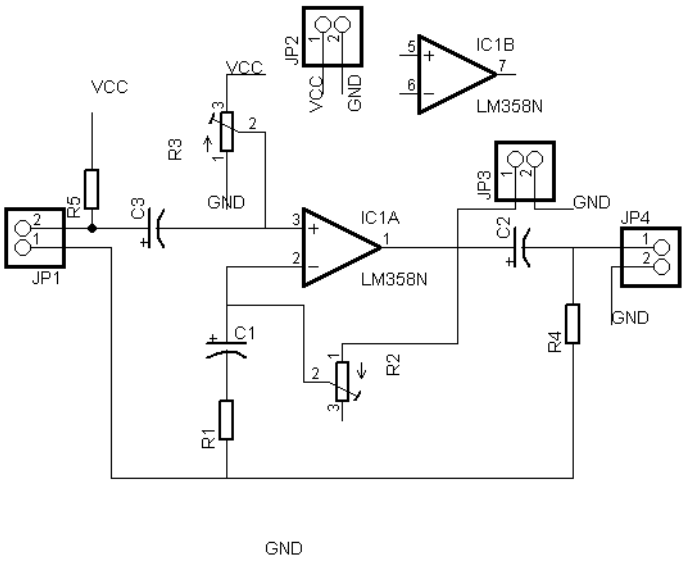

Fig. 4. Pre-amplifier Circuit Diagram

The amplifier output is fed to the analog input pin of the ATmega. The ATmega board contains as crystal circuit of 16Mhz which clock the controller. It also has LEDs connected to its PORTB and two switches connected to PORTD to make it user selectable. It also has an LCD interface. The microcontroller circuit diagram is as shown in the Fig 5.

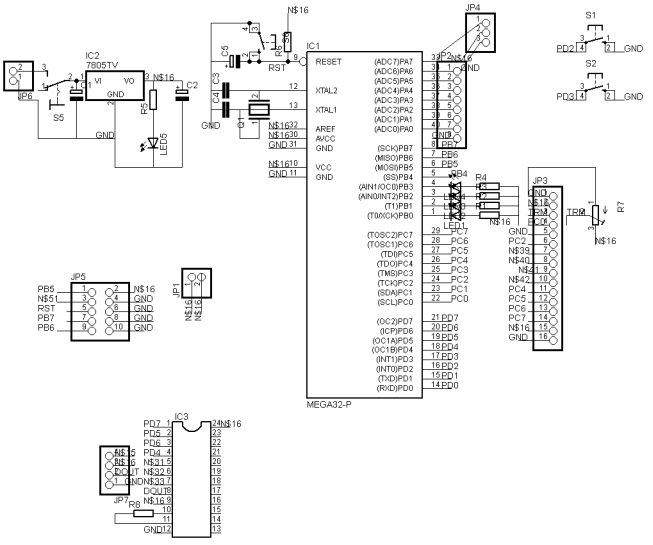

Fig. 5. Microcontroller circuit

VIII. Implementation Results

We recorded the words 'one', 'two', 'play' and 'help' to store the templates. We repeated these words for testing, and all were successfully classified. We repeated the process with with 4 sample words for 10 trial runs. Out of those, we achieved proper results in 6 trials. We also repeated the trials in varying amount of ambient noise. We found that the system accuracy decreased as the ambient noise increased. Also we tried various others word combination also and found that the recognition accuracy is about 80 % if the words are orthogonal and reduces to about 30 % if similar sounding words are stored in the database. Our test results for words can be summarized as shown in table III

TABLE III
STATISTICS

Words	Total Tries	Success
One, Two, Play, Help	10	8
One, Two, Three, Play	10	5
One, Three, Go, Stop	10	7
Two, Three, Four, Five	10	3

IX. Applications and Future Scope

The project can be used in many varied applications such as a speech controlled remote,speech controlled security etc. Its accuracy can be increased by increasing the number of filters. But this would require either interfacing an external RAM to the system or using a DSC for faster processing. Also the other methods (HMM and LPC) described in the paper can be implemented on DSCs which will give better accuracy and robustness to the system.

X. Conclusion

Thus, using the template matching algorithm and digital filters,a low cost real time speech recognition or isolated word recognition system is achieved. We conclude that the AVR can be used for general purpose DSP applications without considerable time lag and memory constrains. Proper code optimization and algorithm selection enables the AVR to perform various low end DSP applications. Also this project aims to be a prelude for implementation of digital filters as hardwired units on FPGAs.

References

[1] Priyabrata Sinha,*Speech Processing in Embedded Systems*, Springer Science, New York, USA, 2009.

[2] Mathias De Wachter, et. al., "Template Based Continuous Speech Recognition", *IEEE Transactions On Audio, Speech & Language Processing*, Vol.15, No. 4, pp.1377-1390, 2007.

[3] B. S. Atal and L. R. Rabiner, "Speech Research Directions", *AT&T Technical Journal, Vol. 65, No. 5, pp. 75-88*, Oct. 1986.

[4] Emerging Technologies, "Voice Recognition 101: Understanding the Fundamentals",[Online]. Available:http://www.em-t.com/Voice-Recognition-101-Understanding-the-Fundamentals-s/214.htm

[5] H. Nyquist, "Certain topics in telegraph transmission theory", *Transactions of the American Institute of Electrical Engineers*, April 1928.

[6] Mathworks Inc., "Chebyshev Type II filter design (stopband ripple)",[Online]. Available: http://www.mathworks.in/help/toolbox/signal/ref/cheby2.html

[7] J.G. Proakis,et al., *Digital Signal Processing: Principles, Algorithms, And Applications*,4th ed., Pearson Education, 2007.

[8] A.V. Oppenheim, R.W. Shcafer, *Digital Signal Processing*, Pearson Education, 1975.