

# Speech Recognition by using MATLAB

Yousuf Islam<sup>1</sup>

<sup>1</sup>School of Electronic Information Engineering, Nanjing University of Information Science and Technology, China. Email: [ibyousuf@163.com](mailto:ibyousuf@163.com);

Voice recognition has evolved into one of the most essential instruments of the contemporary period, with applications in a wide range of industries. Voice recognition technology has advanced dramatically in the last decade, to the point that systems and high-performance algorithms are now widely available. The most frequent way to describe speech recognition system performance is in terms of speed and accuracy. Recognition accuracy is the most essential and easy way to assess voice recognition performance. The goal of this paper was to compare numerous speech recognition algorithms in terms of detection accuracy and processing overhead, and to find the optimum trade-offs between processing cost (speed, power) and accuracy. Also, to implement and verify the chosen voice recognition algorithm using MATLAB. Ten words “yes” and “no” were spoken in an isolated way by male speakers using MATLAB as a simulation environment, these words were used as a reference signal to trained the algorithm, for evaluating phase, all algorithms dictate to subject them to similar test criteria. From the simulation results, the Wiener Filter algorithm outperforms the algorithms in terms of all measures of performance, and power requirement with the moderate complexity of the algorithm and its prospective implementation as hardware.

**Keywords:** *Automatic Speech Recognition, Human-Computer Interaction, Computer Vision.*

## Introduction

Voice recognition is a popular theme in today's life. Voice recognition programs are available which make our life far better. Voice recognition is a technology that the system can be controlled by people with their language [1]-[5]. Rather than typing controlling the buttons for the system or the computer keyboard, using language to control the system is more suitable. Additionally, it may reduce the price of the business production at the same time. The efficiency of daily life enhances, but also makes people's life more diversified [6]. Voice recognition technology is the process of identifying, understanding, and converting voice signals into text or commands. There are different types of authentication mechanisms available today like alphanumeric passwords, graphical passwords, etc. [7]-[9]. Along with these, biometric authentication mechanisms like fingerprint recognition systems, voice recognition systems, iris recognition systems, etc. add more security for data. One of the important areas of research is voice recognition technology. Research in voice recognition involves studies in physiology, psychology, linguistics, computer science, signal processing, and many other fields. Voice recognition technology consists of two different technologies such as speech recognition and speech recognition. Speech recognition is a technique that enables a device to recognize and understand spoken words, by digitizing the sound and

matching its pattern against the stored patterns [10]. Speech is the most natural form of human communication and speech processing has been one of the most exciting areas of signal processing. Speech recognition technology has made it possible for a computer to follow human voice commands and understand human languages [11]. Speech recognition algorithms can be in general divided into speaker-dependent and speaker-independent. A Speaker-dependent system focuses on developing a system to recognize the unique voiceprint of individuals. Speaker independent system involves identifying the word uttered by the speaker.

The purpose of this project was to research, design, and implement a speech recognition algorithm on the ADSP-21262 SHARC digital signal processor (DSP). The original goal was to be able to distinguish two spoken words from each other and to match the recordings with a member of a previously built database [12]. Later on, the database was extended to contain four words instead of two. Another objective was to validate if a recorded word is a good enough match with the library. Since the DSP had hardware limitations and an unfamiliar programming platform, the project was split into two main parts. The first part consisted of implementing the speech recognition algorithm, hereby denoted as SRA, in Mat lab [13]. Compared to the DSP, Mat lab allows the use of seemingly infinite memory and processing power. Mat lab also gives the benefit of having access to the entire signal that needs processing at all times. The DSP, on the other hand, runs in real-time and has limited memory, meaning no data can be stored. Part one of the project consisted of making sure the SRA design worked when being simulated in the Mat lab. Part two consisted of translating the Mat lab code into C code and making sure the algorithms still worked despite the limitations of the DSP.

## **Theory**

### **Basic Speech Recognition**

A basic speech recognition algorithm can be split into three states: listening, processing, and matching. In the listening phase, the DSP analyses the present audio signal to determine if speech is present. When speech is detected, the DSP starts to process the information to describe the speech compactly. A common way to describe speech is to divide it into subsets of 20 ms and build mathematical models of each subset. When the processing is done, the DSP continues to the last phase, which compares the speech model to a pre-defined database. Each of these phases will be more thoroughly described later on in this report.

### **Unvoiced and voiced speech**

Speech can be categorized into unvoiced and voiced speech. Unvoiced speech is the noise-like parts of a word containing consonants, and the tonal voiced speech contains the vowels. As an example, take the word "ask", where "a" is tonal, and "SK" is noisy. In this project, we assume that the reflection coefficients for the voiced speech will result in dominating reflection coefficients.

### **Reflection coefficients and the Schur Algorithm**

As mentioned above, a common way to describe speech is to split the speech into subsets of 20 ms each and mathematically describe each subset. For a short time as 20 ms, speech can be viewed as

a wide-sense stationary stochastic process and can thus be fairly well described with a direct-form FIR-model (AR-model). This is a process that looks like

$$y_t = -a_1 y_{t-1} - a_2 y_{t-2} - \dots - a_p y_{t-p} + e_t \quad (1)$$

Where  $e_t$  is white Gaussian noise, the filter coefficients are represented by  $a_i$ ,  $i=1, 2, \dots, p$  and  $y_t$  is the output at time  $t$ . Of course, more coefficients mean better abilities to describe the speech subset. However, a cruder model with fewer coefficients will describe only the vital parts of the word. This makes matching speech from another person easier since the vital parts describing a spoken word are similar from person to person.

Another way of describing the direct-form FIR-model is with a set of reflection coefficients  $\{K\}$  of the lattice filter. These reflection coefficients are closely related to the AR-coefficients and are calculated recursively. Linear prediction is used to calculate the optimal model describing the speech. The reflection coefficients can be calculated directly from the autocorrelation of the subset signal using the Schur algorithm. The Schur algorithm is fast and will produce the same result as the Levinson-Durbin algorithm, but without having to compute the AR-coefficients. The resulting  $N$  reflection coefficients describe the speech subset. Doing this for each subset will produce a matrix that describes the most vital parts of the spoken word, and can be used to compare to other matrices to match and validate what word was spoken.

### **Digital Signal Processor (DSP)**

A Digital Signal Processor (DSP) is a purpose-built device specialized in processing data that is continuously supplied. The DSP used in this project was a Sharc ADSP-21262 fitted on a custom-made PCB. This DSP has two parallel buses on which messages can be sent, allowing for high-speed processing. The DSP has hardware support for addition, subtraction, and multiplication. The division is emulated and therefore substantially slower than the other mathematical operations. When possible, multiplication by a constant, e.g.  $\text{scale} = 1/q$  should be used instead of dividing repeatedly. The total memory of the DSP is limited to  $4 \times 128$  kB all in all. This puts restrictions on the algorithms that should run on the device, as these cannot use too much memory. An average of all reflection coefficients describing a word is stored in a  $10 \times 10$  array on the DSP in order to avoid noise having a too large effect on the database.

### **Filtering the sampled signal**

Digital filters are applied to boost and cut certain characteristics of the sampled signal to make for better mathematical models. To deal with this, two different filters are needed. Environmental noise-causing mic bias, i.e., a door being shut, are low in frequency but high in energy and are dealt with by a high pass filter. Much of the information in human speech lies in the higher frequencies. In order to boost the lower energy levels of high frequencies present in the human speech, a pre-emphasis filter is used. A pre-emphasis filter boosts the high frequencies while attenuating the lower ones, which flattens the spectrum. It just so happens that the filter characteristics of the high pass and the pre-emphasis filter combined can be reproduced using only one filter in order to conserve resources. As displayed in figures 1 to 3 below, combining the characteristics of the magnitude response of the high pass filter and the pre-emphasis filter will

result in the combined filter shown in the figure. The combined filter has a damping effect on low frequencies of the high pass filter, and the boosting effect of high frequencies from the pre-emphasis filter. Figure 4 below displays the magnitude response of the used filter, and it's easy to see the similarities to the combined filter.

## Method

### Design

In figure 1, the general design structure of the speech recognition algorithm is shown. First of all, the recorded audio needs to be analyzed to determine if speech is present. The algorithm that determines if speech is present listens to the background noise on the channel and sets a threshold for when the level is breached. To avoid spikes in the audio channel originating from e.g. a cup of coffee that is placed on the same table as the microphone, the algorithm requires the signal to maintain its energy for a short period of time. To avoid microphone bias from e.g. a pressure wave originating from a door being closed, a high pass filter was implemented before the speech-detection algorithm.

Reflection coefficients are always calculated, independent of whether speech has been detected or not. The difference is that if no speech is detected, the reflection coefficients of the three previous audio blocks are stored in a circular buffer. When finally speech is detected, the buffers are stored as the first columns in a coefficient matrix. This is done to avoid relevant data being lost due to a recording not starting at the beginning of a word. When speech has been detected, there is an algorithm that checks whether or not the speech has ended. This is done in a similar fashion as when detecting speech, with some additional features that prevent the algorithm to cut in a "natural pause" in a word. The DSP then continues to calculate reflection coefficients and store them in the coefficient matrix. Ten coefficients are calculated, which is a common number that will describe the vital parts of the speech well enough without the risk of over-fitting.

Finally, when speech has ended, the following three coefficient vectors are stored in the matrix. Three coefficient vectors in the beginning and end are intended both as a safeguard to catch all the speech, but also as simple windowing to apply weight on the middle parts. Now the DSP will compress the matrix from a  $10 \times N$  matrix to  $10 \times 10$  matrices. This is done to allow for comparison between the recorded speech and the entries of the database. The compression function was implemented in such a way that it for the first  $10 - u$ ,  $u = \text{mod}(N, 10)$  columns in the compressed matrix calculates the row-wise average of  $K$  columns, where  $K = \text{floor}(N/10)$ , in the original matrix. These average values are then saved in the new, compressed, matrix. The remaining  $u$  columns in the compressed matrix is calculated as the row-wise average of  $K + 1$  column in the original  $10 \times N$  matrix.

The compressed matrix is compared to the  $10 \times 10$  matrices in the database by

$$e_k = \sum_{i,j} (R_{ij} - D_{ij}^k)^2 \quad (2)$$

where  $D^k$ ,  $k = 1, 2, 3, 4$  represent the different database matrices, and where  $R$  represents the recorded and compressed matrix at hand. The matrix  $D^k$  that results in the smallest leak will be regarded as the best match.

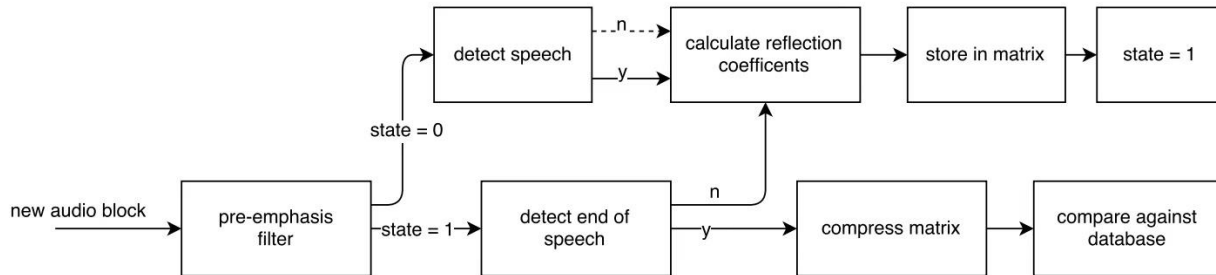


Figure 1: Flow chart of the general design y = yes, n = no

### System Workflow

This project runs in real-time. It all starts from the speech signal, yes or no, which makes use of the built-in microphone of any computer. Then the signal will be processed in MATLAB where fft takes place. Finally, if the value is below the threshold it would display Yes and no if it's the other way around.

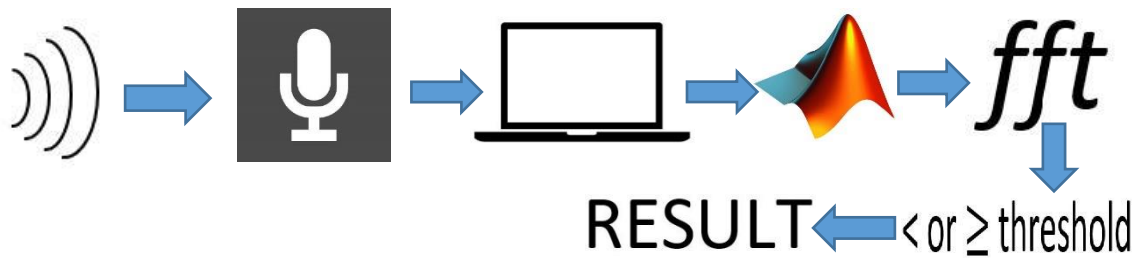


Figure 2: System Workflow

### Simulation

To verify that the previously mentioned design worked as intended, simulations in Matlab were made. As a first step, the reflection coefficients for a pre-recorded speech sample were calculated using Matlab's built-in `score`- and `Schurrc`-functions. It should be mentioned that the `xcorr`-function returns the autocorrelation sequence of the supplied signal and that this sequence is used in the Schur algorithm. It was suitable to use 10 reflection coefficients for each block of data, each block being  $20 \text{ ms}^{-1}$  of speech. With a sampling frequency of 16 kHz, each block became 320 samples. The next step was to record an audio sample. Since Matlab supports audio recording, a microphone was used to record speech. The recording was then divided into N blocks, each block

of 320 values, using Matlab's buffer function. For each block of data, a pre-emphasis filter was applied. This filter was an FIR filter with coefficients 1 and  $-1$ , corresponding to a high-pass filter. Matlab's filter function was used to implement this. A vector of 10 reflections coefficients for each block was then calculated. This was done for all  $N$  blocks of data. Each vector of coefficients was then saved in a  $10 \times N$  matrix, where  $N$  is determined by the length of the recorded audio signal. To make the implementation in C easier, Matlab's built-in Schurrc-function was replaced by a self-written function that had the same functionality as the built-in one. Since recordings of speech will have some silence before and after the speech, a function that detects the start and the end of the spoken word and removes the silent parts was written. This function was then split into two separate functions, one that removes the "silent" samples before the speech and one that removes the "silent" samples after the speech. The reason that this is important is that the "silent" samples contain no information that is of interest when the reflections coefficients are calculated. When the samples containing no information have been removed, the data is divided and reflection coefficients calculated as described in the first paragraph. To make a comparison between two matrices possible, all the matrices have to be compressed to the same size. This was achieved by implementing a function that compresses a  $10 \times N$  matrix to a  $10 \times 10$ -matrix, more thoroughly described in the design section of this report. After this was done, a database of three words was created; this was changed to four words during the implementation on the DSP. The words of the database, during the simulation, were: "Mikael", "Swartling" and "hundratjugosju". At first, only one recording of each word was used to build the database. This was to test that the recording worked as intended. Later this was replaced with the average of 12 recordings per word, 3 recordings per group member. The averaging was done on the compressed  $10 \times 10$  matrices generated at every recording. The average was then saved in the database. If only one database was to be used, averaging gave a better result for multiple persons than just a database built from one person.

## Result

In Matlab, the algorithms worked just as intended, and the program was able to distinguish the right word most of the time. It seemed to match roughly the same number of times regardless of which of the four recorded words in the database was spoken. The number of matches seemed to be the same no matter whether or not the person speaking was part of creating the database. However, the smallest value of the error described by equation 2, which is used to determine what word was being said, tended to vary. The smallest recorded error was around 0.9, and the largest error still yielding a correct match was somewhere around 7. The big difference made validation difficult, even though the program still managed to find a correct match. In a conducted test, three of the group members said each member word of the database ten times, and the match frequency was noted. They can be seen in Table 1 below.

Table 1: Data analysis of YES

Voice	Frequency	Amplitude	F value	E.Yes
Yes1	1400Hz	0.42	8.57	6.615
Yes2	1800Hz	0.38	7.23	
Yes3	1600Hz	0.45	5.93	
Yes4	1400Hz	0.48	6.47	
Yes5	1300Hz	0.44	7.73	

Further on, a test where a group member used his database and validation mode ON was performed. In this test, the four words of the database were repeated ten times each. Also, the words "YES" and "NO" were repeated ten times to test the validation threshold. The results are shown in Table 2 below.

Table 2: Data analysis of NO

Voice	Frequency	Amplitude	F value	E.NO
No1	1000Hz	0.34	8.94	8.185
No2	1000Hz	0.32	9.48	
No3	1200Hz	0.35	8.29	
No4	1000Hz	0.28	6.13	
No5	1200Hz	0.28	9.21	

```

Editor - E:\Education\5th report\DSP PIC\New folder (2)\system1_ZCR.m
Command Window
>> system1_ZCR
The ZCR of yes is
    0.0693    0.0592    0.0744

The ZCR of no is
    0.0668    0.0232    0.0657

Test file [yes] #1 classified as yes
Test file [yes] #2 classified as yes
Test file [yes] #3 classified as yes
Test file [yes] #4 classified as yes
Test file [yes] #5 classified as yes
Test file [yes] #6 classified as yes
Test file [yes] #7 classified as yes
Test file [yes] #8 classified as yes
Test file [yes] #9 classified as yes
Test file [yes] #10 classified as yes
Test file [no] #1 classified as no
Test file [no] #2 classified as yes
Test file [no] #3 classified as no
Test file [no] #4 classified as no
Test file [no] #5 classified as no
Test file [no] #6 classified as no
Test file [no] #7 classified as no
Test file [no] #8 classified as no
Test file [no] #9 classified as no
Test file [no] #10 classified as no
fx >>

```

```

Editor - E:\Education\5th report\DSP PIC\New folder (2)\system2_energy_ZCR.m
Command Window
>> system2_energy_ZCR
The ZCR of yes is
    0.0693    0.0592    0.0744    6.6155

The ZCR of no is
    0.0668    0.0232    0.0657    8.1899

Test file [yes] #1 classified as no
Test file [yes] #2 classified as yes
Test file [yes] #3 classified as yes
Test file [yes] #4 classified as yes
Test file [yes] #5 classified as yes
Test file [yes] #6 classified as yes
Test file [yes] #7 classified as yes
Test file [yes] #8 classified as yes
Test file [yes] #9 classified as yes
Test file [yes] #10 classified as yes
Test file [no] #1 classified as no
Test file [no] #2 classified as yes
Test file [no] #3 classified as no
Test file [no] #4 classified as no
Test file [no] #5 classified as no
Test file [no] #6 classified as no
Test file [no] #7 classified as no
Test file [no] #8 classified as no
Test file [no] #9 classified as yes
Test file [no] #10 classified as no
fx >>

```

Figure 3: Testing results YES or NO Recognitions

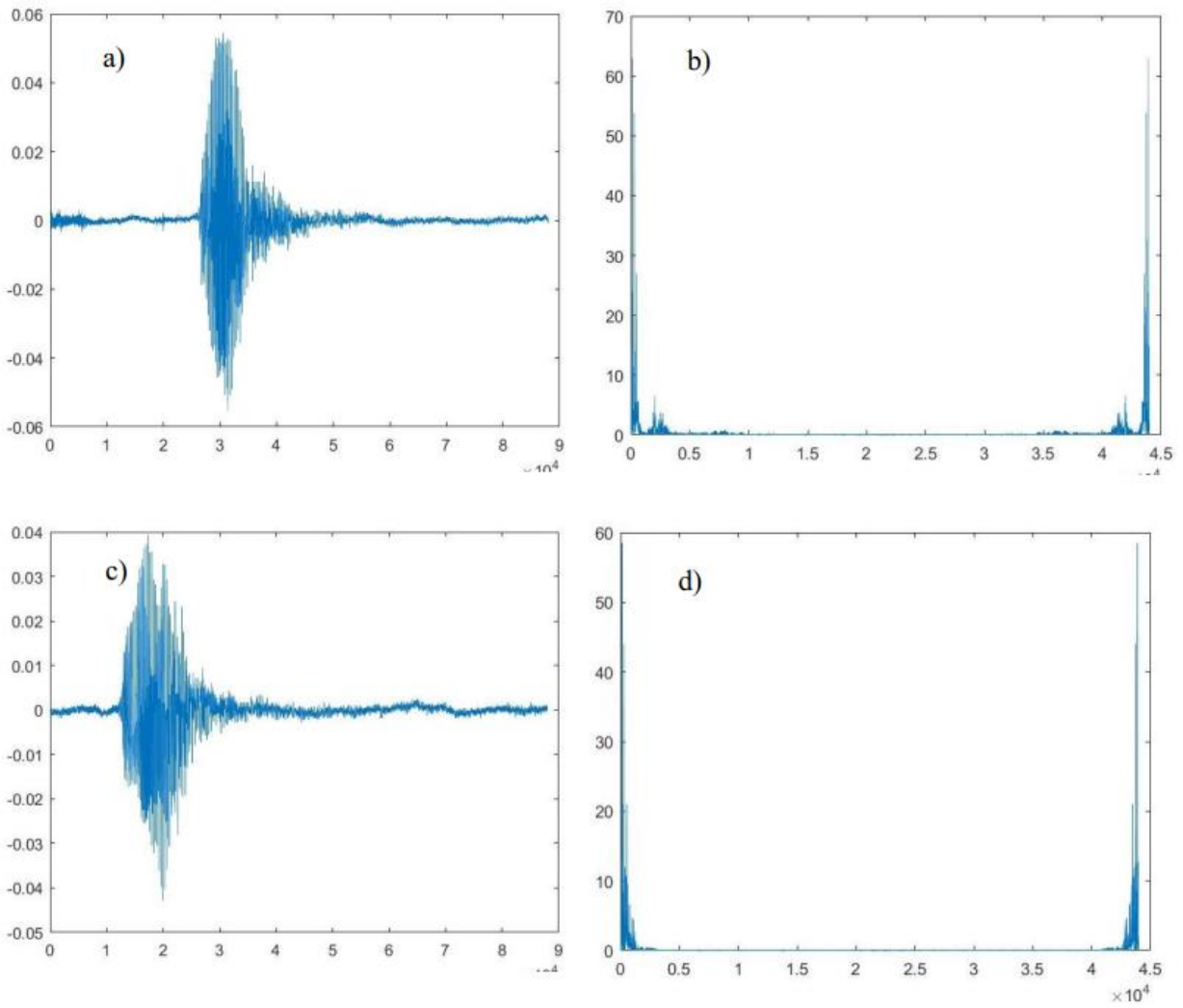


Figure 4: Fourier transform: a and b yes frequency wave and c and d no frequency wave



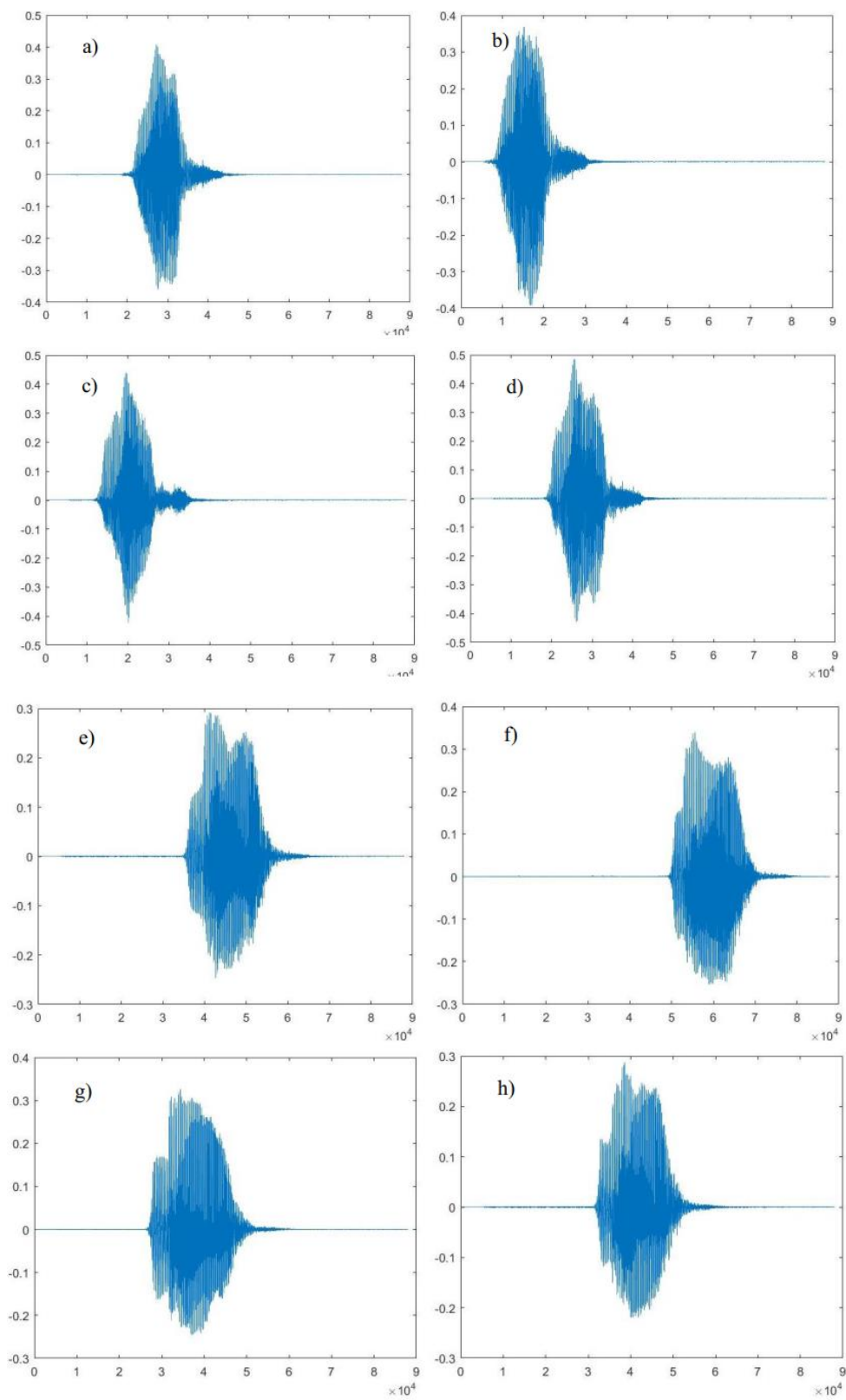


Figure 5: a, b, c, and d YES audio wave frequency and e, f, g, and h No audio wave frequency

## **Discussion and conclusion**

The result of the project was very satisfying; we managed to accomplish more than the basic requirements. These were to make a program that could distinguish between two words for multiple speakers, our speech recognition program can distinguish between two words for multiple speakers with satisfying results. As can be seen in table 2, we never get an incorrect match, which is the most important thing in speech recognition. However, choosing a threshold giving raise to this result will cause some of the words in the database to not be recognized as a word in the database. Different words give different magnitudes of errors, making it hard to implement a common threshold level. Setting a separate threshold for each word can be done with some more fine-tuning. Of course, there might be a better way to set a threshold than just an error magnitude threshold. It is also possible that different matrix compression techniques would generate a better result, or that different sizes of the compressed matrix would work better for our purposes. The reason that we chose the previously mentioned method to compress a  $10 \times N$  matrix was because of its simplicity. The size of the compressed matrix was chosen to be  $10 \times 10$  since it was desirable to use as little memory as possible, and the size  $10 \times 10$  was considered a reasonable size. This might however not be the optimal size, nor the method to compress the  $10 \times N$  matrix. Also worth mentioning is that at first, we only had two words in the database. This amount was later increased to two words, the reason being that one of the two words was quite hard to identify and often resulted in incorrect matches being displayed. The reason behind the problems might the word length, or that the structure of the words is too similar. In any case, the word was changed and two other words were entered into the database instead. This change resulted in better matching even though the size of the database was increased. When discussing the words that were used in the database, it should be mentioned that the SRA relies on two parameters for the correct identification of a recording. These are the vowels of the detected word as well as the position of these vowels within the word. The reflection coefficients are very much depending on what vowels are being spoken. Consonants contain less information, and reflection coefficients describing consonants are less prominent than reflection coefficients of vowels. Perhaps using other words as our database would have increased the hit rate using validation methods, but this was nothing we investigated further due to lack of time. We've also concluded that accentuation matters. Since the vowels are the information containing part of a word, and the vowel is spoken differently depending on your accent, it is quite hard to build a database that works for a common speaker. This was especially apparent when a group member tried to use the database of another person, as this resulted in more errors by the SRA.

## **Acknowledgment**

The author would like to thank Google and some search engine for the process and their continued support and encouragement for the completion of this report, as well as thanks to the Ma'am for her support to me and to facilitate a lot of the difficulties that I faced during the study period.

## Reference

- [1] Personal communication between Dr. Sanjeev Bhalla with author (CJA) on 10.11.2011 [Google Scholar]
- [2] Hayt DB, Alexander S. The pros and cons of implementing PACS and speech recognition systems. *J Digit Imaging*. 2001; 14:149–57. [PMC free article] [PubMed] [Google Scholar]
- [3] Gutierrez AJ, Mullins ME, Novelline RA. Impact of PACS and voice-recognition reporting on the education of radiology residents. *J Digit Imaging*. 2005; 18:100–8. [PMC free article] [PubMed] [Google Scholar]
- [4] Kauppinen T, Koivikko MP, Ahovuo J. Improvement of report workflow and productivity using speech recognition--a follow-up study. *J Digit Imaging*. 2008; 21:378–82. [PMC free article] [PubMed] [Google Scholar]
- [5] Krishnaraj A, Lee JK, Laws SA, Crawford TJ. Voice recognition software: Effect on radiology report turnaround time at an academic medical center. *AJR Am J Roentgenol*. 2010; 195:194–7. [PubMed] [Google Scholar]
- [6] Akhtar W, Ali A, Mirza K. Impact of a voice recognition system on radiology report turnaround time: Experience from a non-english-speaking South Asian Country. *AJR Am J Roentgenol*. 2011;196: W485. [PubMed] [Google Scholar]
- [7] IBM.com. [Last accessed on 2011 Nov 16]. Available from: [http://www03.ibm.com/ibm/history/exhibits/specialprod1/specialprod1\\_7.html](http://www03.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html).
- [8] Schwartz LH, Kijewski P, Hertogen H, Roossin PS, Castellino RA. Voice recognition in radiology reporting. *AJR Am J Roentgenol*. 1997; 169:27–9. [PubMed] [Google Scholar]
- [9] Torrieri M. Talk vs Type: Taking another look at voice recognition. [Last accessed on 2011 Jul]
- [10] Liu D, Zucherman M, Tulloss WB., Jr Six characteristics of effective structured reporting and the inevitable integration with speech recognition. *J Digit Imaging*. 2006; 19:98–104. [PMC free article] [PubMed] [Google Scholar]
- [11] Pezzullo JA, Tung GA, Rogg JM, Davis LM, Brody JM, Mayo-Smith WW. Voice recognition dictation: Radiologist as transcriptionist. *J Digit Imaging*. 2008; 21:384–9. [PMC free article] [PubMed] [Google Scholar]
- [12] McGurk S, Brauer K, Macfarlane TV, Duncan KA. The effect of voice recognition software on comparative error rates in radiology reports. *Br J Radiol*. 2008; 81:767–70. [PubMed] [Google Scholar]
- [13] Quint LE, Quint DJ, Myles JD. Frequency and spectrum of errors in final radiology reports generated with automatic speech recognition technology. *J Am Coll Radiol*. 2008; 5:1196–9. [PubMed] [Google Scholar]

## Code:

### Audio recorder

```
recObj = audiorecorder(44000, 24, 1);% record at Fs=44khz, 24 bits per sample
```

```

for i=1:5
fprintf('Start speaking for audio #%d\n',i)
recordblocking(recObj, 2); % record 2 seconds
fprintf('Audio #%d ended\n',i)
%play(recObj);

y = getaudiodata(recObj);
y = y - mean(y);
file_name = sprintf('test/no/no%d.wav',i);
audiowrite(file_name, y, recObj.SampleRate);
figure
plot(y);
end

```

### Fourier transform “Yes” audio

```

[y, fs] = audioread('Yes_audio.wav');
plot(y);
f = abs(fft(y));
index_f = 1:length(f); % from 1 to number of samples in y
index_f = index_f ./ length(f); % index will be from 0:1/length(f):1
index_f = index_f * fs;
figure;
plot(index_f,f);

```

### Fourier transform “No” audio

```

[y, fs] = audioread('NO_audio.wav');
plot(y);
f = abs(fft(y));
index_f = 1:length(f); % from 1 to number of samples in y
index_f = index_f ./ length(f); % index will be from 0:1/length(f):1
index_f = index_f * fs;
figure;
plot(index_f,f);

```

### Training files and calculate the energy

```

training_files_yes = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\yes\*.wav');
testing_files_yes = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\yes\*.wav');
training_files_no = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\no\*.wav');
testing_files_no = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\no\*.wav');

% ----- Training -----

% read the 'yes' training files and calculate the energy of them.
data_yes = [];
for i = 1:length(training_files_yes)
file_path = strcat(training_files_yes(i).folder,'\',training_files_yes(i).name);% get the file path with name
[y,fs] = audioread(file_path); % read the audio file

energy_yes=sum(y.^2); % calculate the energy
data_yes = [data_yes energy_yes]; % append the energy with all other energies of the other files
end
energy_yes=mean(data_yes); % calculate the average energy
fprintf('The energy of yes is \n');

```

```

disp(energy_yes);

% read the 'no' training files and calculate the energy of them.
data_no = [];
for i = 1:length(training_files_no)
file_path = strcat(training_files_no(i).folder,'\',training_files_no(i).name);
[y,fs] = audioread(file_path);

energy_no=sum(y.^2);
data_no = [data_no energy_no];
end
energy_no=mean(data_no);
fprintf('The energy of no is \n');
disp(energy_no);

% ----- Evaluation -----

% read the 'yes' tesing files and calculate the energy of them.

for i = 1:length(testing_files_yes)
file_path = strcat(testing_files_yes(i).folder,'\',testing_files_yes(i).name);
[y,fs] = audioread(file_path);

y_energy = sum(y.^2);
% test if the energy of this file is closer to YES or NO average energies
if(abs(y_energy-energy_yes) < abs(y_energy-energy_no))
    fprintf('Test file [yes] #%d classified as yes ,E=%d\n',i,y_energy);
else
    fprintf('Test file [yes] #%d classified as no E=%d\n',i,y_energy);
end
end

for i = 1:length(testing_files_no)
file_path = strcat(testing_files_no(i).folder,'\',testing_files_no(i).name);
[y,fs] = audioread(file_path);

y_energy = sum(y.^2);

if(abs(y_energy-energy_yes) < abs(y_energy-energy_no))
    fprintf('Test file [no] #%d classified as yes ,E=%d\n',i,y_energy);
else
    fprintf('Test file [no] #%d classified as no ,E=%d\n',i,y_energy);
end
end

```

## **System2**

```

training_files_yes = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\yes\*.wav');
testing_files_yes = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\yes\*.wav');
training_files_no = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\no\*.wav');
testing_files_no = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\no\*.wav');

```

```

% read the 'yes' training files and calculate the energy of them.
data_yes = [];
for i = 1:length(training_files_yes)
file_path = strcat(training_files_yes(i).folder, '\', training_files_yes(i).name);
[y,fs] = audioread(file_path);
%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_yes1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_yes2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_yes3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
%calculate the energy
energy = sum(y.^2);
ZCR_yes = [ZCR_yes1 ZCR_yes2 ZCR_yes3 energy];
data_yes = [data_yes ;ZCR_yes];
end
ZCR_yes=mean(data_yes);
fprintf('The ZCR of yes is \n');
disp(ZCR_yes);

```

```

% read the 'no' training files and calculate the energy of them.
data_no = [];
for i = 1:length(training_files_no)
file_path = strcat(training_files_no(i).folder, '\', training_files_no(i).name);
[y,fs] = audioread(file_path);

%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_no1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_no2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_no3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
%calculate the energy
energy = sum(y.^2);

ZCR_no = [ZCR_no1 ZCR_no2 ZCR_no3 energy];

data_no = [data_no ;ZCR_no];
end
ZCR_no=mean(data_no);
fprintf('The ZCR of no is \n');
disp(ZCR_no);

```

```

% read the 'yes' testing files and calculate the energy of them.

for i = 1:length(testing_files_yes)
file_path = strcat(testing_files_yes(i).folder, '\', testing_files_yes(i).name);
[y,fs] = audioread(file_path);

%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_yes1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_yes2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_yes3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
%calculate the energy
energy = sum(y.^2);

y_ZCR = [ZCR_yes1 ZCR_yes2 ZCR_yes3 energy];

```

```

%make the decision based on cosine distance
if(pdist([y_ZCR;ZCR_yes],'cosine') < pdist([y_ZCR;ZCR_no],'cosine'))
    fprintf('Test file [yes] #%d classified as yes \n',i);
else
    fprintf('Test file [yes] #%d classified as no \n',i);
end
end

% read the 'no' tesing files and calculate the energy of them.
for i = 1:length(testing_files_no)
file_path = strcat(testing_files_no(i).folder,'\',testing_files_no(i).name);
[y,fs] = audioread(file_path);
%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_no1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_no2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_no3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
energy = sum(y.^2);

y_ZCR = [ZCR_no1 ZCR_no2 ZCR_no3 energy];
%make the decision based on cosine distance
if(pdist([y_ZCR;ZCR_yes],'cosine') < pdist([y_ZCR;ZCR_no],'cosine'))
    fprintf('Test file [no] #%d classified as yes \n',i);
else
    fprintf('Test file [no] #%d classified as no \n',i);
end
end
end

```

### System3

```

training_files_yes = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\yes\*.wav');
testing_files_yes = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\yes\*.wav');
training_files_no = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\no\*.wav');
testing_files_no = dir('E:\Education\5th report\DSP PIC\New folder (2)\train\no\*.wav');

% read the 'yes' training files and calculate the energy of them.
data_yes = [];
for i = 1:length(training_files_yes)
file_path = strcat(training_files_yes(i).folder,'\',training_files_yes(i).name);
[y,fs] = audioread(file_path);
%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_yes1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_yes2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_yes3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
%append the ZCR for the 3 parts as a row vector
ZCR_yes = [ZCR_yes1 ZCR_yes2 ZCR_yes3];
%store the raw vector in a matrix
data_yes = [data_yes ;ZCR_yes];
end
ZCR_yes=mean(data_yes);
fprintf('The ZCR of yes is \n');
disp(ZCR_yes);

% read the 'no' training files and calculate the energy of them.
data_no = [];

```

```

for i = 1:length(training_files_no)
file_path = strcat(training_files_no(i).folder,'\',training_files_no(i).name);
[y,fs] = audioread(file_path);
%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_no1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_no2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_no3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
%append the ZCR for the 3 parts as a row vector
ZCR_no = [ZCR_no1 ZCR_no2 ZCR_no3];
%store the raw vector in a matrix
data_no = [data_no ;ZCR_no];
end
ZCR_no=mean(data_no);
fprintf('The ZCR of no is \n');
disp(ZCR_no);

% read the 'yes' tesing files and calculate the energy of them.

for i = 1:length(testing_files_yes)
file_path = strcat(testing_files_yes(i).folder,'\',testing_files_yes(i).name);
[y,fs] = audioread(file_path);
%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_yes1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_yes2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_yes3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
%append the ZCR for the 3 parts as a row vector
y_ZCR = [ZCR_yes1 ZCR_yes2 ZCR_yes3];
%make the decision based on euclidean distance
if(pdist([y_ZCR;ZCR_yes],'euclidean') < pdist([y_ZCR;ZCR_no],'euclidean'))
    fprintf('Test file [yes] #%d classified as yes \n',i);
else
    fprintf('Test file [yes] #%d classified as no \n',i);
end
end

% read the 'no' tesing files and calculate the energy of them.
for i = 1:length(testing_files_no)
file_path = strcat(testing_files_no(i).folder,'\',testing_files_no(i).name);
[y,fs] = audioread(file_path);
%divide the signal into 3 parts and calculate the ZCR for each part
ZCR_no1 = mean(abs(diff(sign(y(1:floor(end/3))))))./2;
ZCR_no2 = mean(abs(diff(sign(y(floor(end/3):floor (end*2/3))))))./2;
ZCR_no3 = mean(abs(diff(sign(y(floor(end*2/3):end)))))./2;
y_ZCR = [ZCR_no1 ZCR_no2 ZCR_no3];
%make the decision based on euclidean distance
if(pdist([y_ZCR;ZCR_yes],'euclidean') < pdist([y_ZCR;ZCR_no],'euclidean'))
    fprintf('Test file [no] #%d classified as yes \n',i);
else
    fprintf('Test file [no] #%d classified as no \n',i);
end
end

```