

Faculty of Engineering

Voice Recognition Project



There are 3 simple phases in developing a Voice Recognition System:

1. **Voice Acquisition**
2. **Voice Processing**
 - 2.1 **Pre Processing**
 - 2.2 **Template Matching**
3. **Data communication**

Step 1:

Capturing an audio signal from Mic and getting it to MATLAB work space is very similar to acquiring an image...

Let's see how this can be done...

First we start to Record the Voice with the MATLAB function **waverecord**

```
fs = 11025;  
ai = waverecord(0.6*fs,fs,'int16');
```

The Fs is the frequency of the wave and the 0.6*fs is the time of the recording process.

Now we will save the wave file which has been saved in the ai to the disk so we use the **wavewrite** function

```
wavewrite(ai,'on');  
wavplay(ai);
```

To play the wave file we used the **waveplay** function.

In our project we will input a 2 voice the first for open the cd room door and the second to close it so we write this code:

```

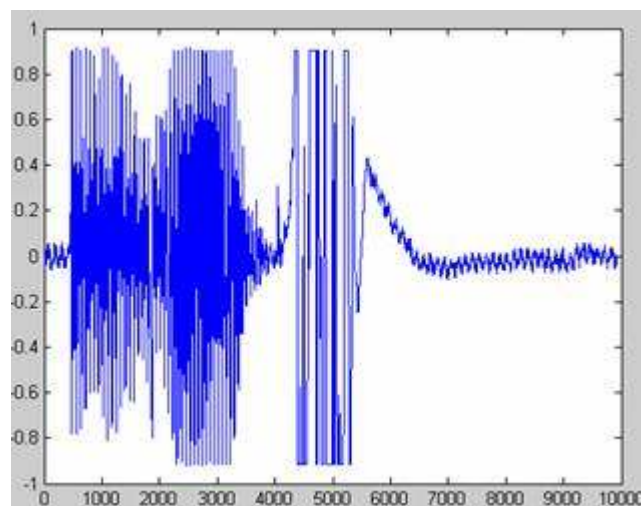
z=input(' press (1) to talk another word or zero to end ');
if z==1
    ai2= wavrecord(0.6*Fs,Fs,'int16');
    wavwrite(ai2,'off');
    wavplay(ai2);
    k=k+1;
    disp(' you have two words in your library ');
else
    disp(' you have just one word in your library ');
    break;
end

```

we do the same

Plot (data)

This will display a figure with number of sample per trigger on X axis and Amplitude values on Y axis....



Now we are ready to the next step

STEP 2:

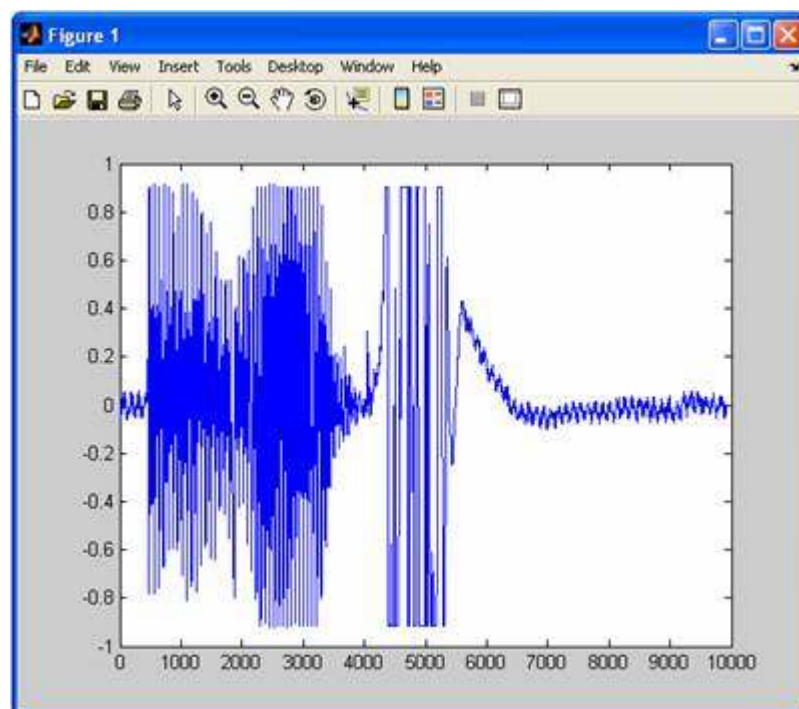
First of all the basic stuff, Voice data is a 1 dimensional array... So dealing with it or ...processing this data is much simpler than processing a 3 dimensional RGB image

1- Calculate the number of peaks in the sample. And take necessary action accordingly.

2-Peak value of different words can be used for taking different actions accordingly.

First method of developing a voice recognition system is very simple and funny to deal with it's like we set a threshold value by analyzing the plot of the word and process the data in such a way that if value of the sample at any point is grater than the threshold then we consider that point as a peak.

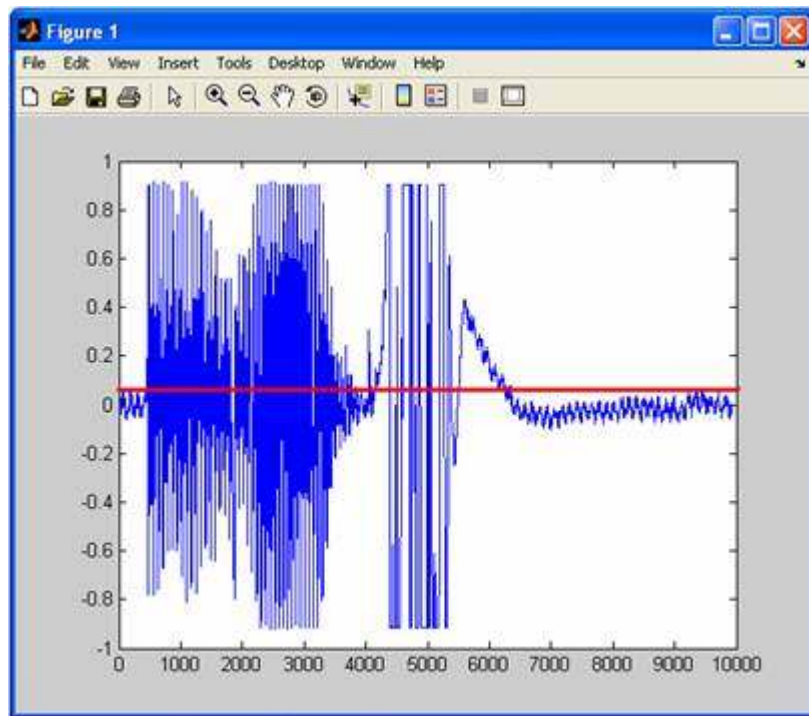
Note: This Method is effective only if you wanna differentiate between 2 or 3 selective words ... But it solves our purpose of voice control at higher repeatability and accuracy than other complex Algorithms



If you work on these Plots for quite some time you will understand what all parameters to be analyzed to develop logic for Voice recognition

In this analysis you can find that there will be some noise in the plot of very low amplitude when nothing is being spelled.

Once you analyzed this concept it's quite simple future on, now we just need to remove all the noises and consider any value above the noise as input.



In the above figure there is a Red line which indicates the level of noise

Now processing it is very simple, we need to calculate the no of values which are greater then the noise level and add some negative tolerance of around considerable no of peaks (values) to this value...

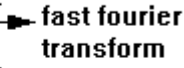
Here we consider all the values above the noise level as peaks and we count the no of peaks in the sample and accordingly take necessary action...

The **Fourier transform translates** the 1D voice data into 2D matrix where each point in the matrix represents the degree of a frequency occurring in the sound at that time.

- Record the wave file
- Read-in the wave file via *wavread*
- Perform Fast Fourier Transformation on the wave file via *fft*
- Represent imaginary numbers in FFT matrix as reals via multiplying by its conjugate
- Look at absolute value of the important (beginning) part of the data in the new matrix
- Split the FFT matrix into n bins and get the average of each
- Standardize the return value by dividing the matrix by it's sum

So we used this Function **SoundSig**

```

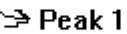
function [p]=soundSig(filename)
in=wavread(filename);
f=fft(in,100000);
q=f.*conj(f);  fast fourier transform
q=abs(q(1:5000));
for i=1:50
    t=(i-1)*100+1;
    p(i)=sum(q(t:t+99));
end
p=p/sum(p);

```

In our project now we will calculate the starting of the first word so the code will be like that:

```

% loop to find the starting of the word...

for n=1:a  [a b] = size(x)
    if (x(n,1)>0.15)
        p1=n;
        break;  Peak 1
    end
end
n=0;

```

Now we will calculate the end of the word:

```

% loop to find the ending of the word...

for n=a:-1:1
    if (x(n,1)>0.15)
        p2=n;
        break;  Peak 2
    end
end
z=1;

```

Now we will Compute the **inner product** of reference Vector with the Real time vector and Find the maximum value of the inner product and its corresponding reference vector

```

% make x to zero...

for n=1:a
    x(n,1)=0;
end

% initilaize x to the values only with the values of word

x=x_new;

p8=0;
v=x(1,1);

for n=2:4000
    if(x(n,1)<=0.02)
        x(n,1)=0;
    else
        x(n,1)=1;
        p8=p8+1;
    end
end
end

```

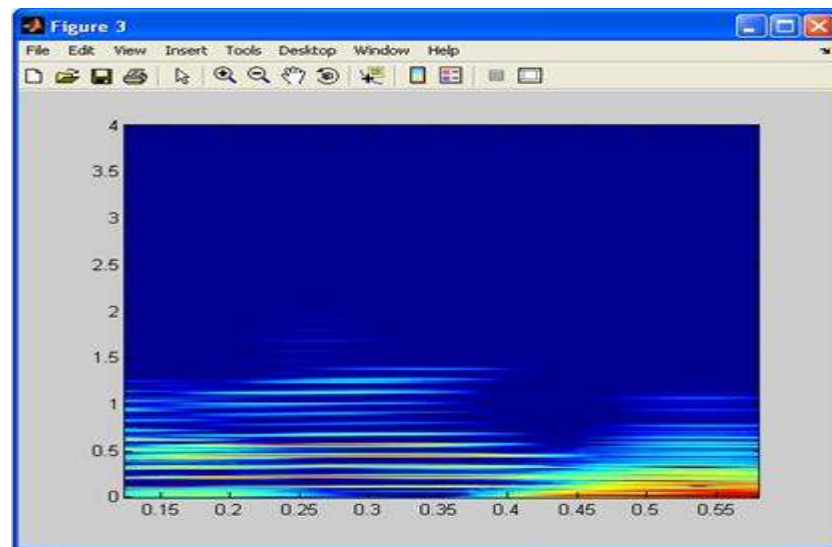
we repeat the above step with the second word that will close the cd

STEP 3:

The waveform needs to be transformed into a form that is suitable to determine speech. So the common practice is to use a Fourier Transform and create a spectrogram

The Fourier transform translates the 1D voice data into 2D matrix where each point in the matrix represents the degree of a frequency occurring in the sound at that time.

This plot of 2D matrix which shows frequency's at certain point in time is called as **spectrogram**



Every different word spelled by the same speaker will have different spectrogram...
And specialists can decode the word by seeing the spectrogram...

The MATLAB code to generate the spectrogram:

```
tic
[x,fs,bw] = wavread('on');
fftmin = 256;

winlen = floor(2*fs/bw);

% Get FFT length
fftlen = max([winlen fftmin]);

% Create window
win = [hamming(winlen) ; zeros(fftlen-winlen,1)];
win = win/sum(win);
windel = (0:(length(win)-1)) * win;

% Set overlap
ntime = 500; % Choose based on trial-and-error for best looking plot
overlap = floor(max(fftlen/2, (ntime*fftlen-length(x))/(ntime-1)));
ntime = floor((length(x)-overlap)/(fftlen-overlap));

% Create arrays
c1=(1:fftlen)';

% Take FFT of real data
b = fft(x(c1(:,ones(1,ntime))+r1(ones(fftlen,1),:)).*win(:,ones(1,ntime))));
if size(b,1) == 1
    m = length(b1);
    b(floor((m+4)/2):m) = [];
else
    m=size(b,1);
    b(floor((m+4)/2):m,:) = [];
end
b = b.*conj(b);

% Setup pixel locations for plot on frequency axis and time axis
f=(0:fftlen/2)*fs/fftlen; % Point spacing
t = (r1+windel)/fs;

% Set limit for dB scale
lim = max(b(:))*0.0001;

% Set dB scale
b=2.5*log10(max(b,lim));

% to plot the spectrogram of the word...
```

STEP 4:

Now we will insert a new voice that will compared with the 2 first voice

```
q=input('enter 1 to talk ');
if q==1

Fs = 11025;
ai = wavrecord(0.6*Fs,Fs,'int16');
wavwrite(ai,'on2');
wavplay(ai);
```

We will use the function **Soundsig** again

```
test=soundsig('on2');
```

STEP 5:

Comparison with reference vectors

```
.....
toc
rr=sum(test.*ah)
vv=sum(test.*yo)
if rr>vv
    mycdopen
else
    mycdcclose
end

else
    break;
end
```

the word we want to compare after the Sounsig function

the first two words

we call an API to do this process

Now we finished our project

Good luck