DSP Project Report – Speech Recognition

Submitted by:

2014-MC-01	**Usman Farooq Butt**
2014-MC-20	**Tajammal Nawaz**
2014-MC-24	**Taha Amjad**
2014-MC-32	**Arslan Ali**

Submitted to: Dr. Ahsan Naeem

Course: Digital Signal Processing (DSP)

Project Submission Date: 20th December, 2017

Report Submission Date: 1st January, 2018

Department of Mechatronics and Control Engineering

University of Engineering and Technology, Lahore

Problem Statement:

As our semester project, we had to design a speaker dependent speech recognition system for numbers from one to ten. The aim was to implement the signal processing techniques learnt during the course. This project is a pure example of signal processing. After signal processing is done, the output can be used for various purposes. Possible applications can be home automation using voice and an increasing number of IoT applications, speech to text environments for disabled people, and growing AI applications.

Methodology:

Early Methods:

We began our project by using the suggested **Cepstrum Analysis** method. Formants were successfully extracted from the speech but they had close proximity to each other. This posed challenge for us as the coding would have been difficult and the computation would have been time consuming.

We then shifted our focus towards **MFCC-Mel frequency cepstral coefficients**. First, Fourier Transform of windowed signal is taken, then powers of spectrum are mapped on mel scale, log of power is taken at each frequency, then discrete cosine transform of list of mel powers is taken and the amplitudes of resulting spectrum are MFCCs. MFCC method gave 26 coefficients after processing. Coding this would have been harder and time consuming than Cepstrum.

We finally moved to LPC-Linear Prediction Coefficient Analysis method.

Final Method: Linear Prediction Analysis:

We directly used the LPC command in MATLAB, here is the summarized function of what it does: As the cepstral analysis does the deconvolution of speech into source and system components by traversing through frequency domain, the deconvolution task becomes computational intensive process. To reduce such type of computational complexity and finding the source and system components from time domain itself, the *Linear Prediction* analysis is developed.

The redundancy in the speech signal is exploited in the LP analysis. The prediction of current sample as a linear combination of past *p* samples form the basis of linear prediction analysis. The predicted sample is found using linear prediction coefficients and windowed speech sequence. The prediction error e(n) can be computed by the difference between actual sample s(n)and the predicted sample. The primary objective of LP analysis is to compute the LP coefficients which minimized the prediction error e(n). The popular method for computing the LP coefficients by least squares auto correlation method. This achieved by minimizing the total prediction error.

Using Toeplitz matrix R,

$$R = \begin{bmatrix} R(1), R(2), R(3), ..., R(P) \\ R(2), R(1), R(2), ..., R(P-1) \\ R(3), R(2), R(1), ..., R(P-2) \\ . \\ . \\ R(P), R(P-1), R(P-2), ..., R(1) \end{bmatrix}$$

LP coefficients are calculated using $A=-R^{-1}.x.$ Coefficients represent the polynomial (as evident from the roots command in code). Then we take the positive imaginary roots, and calculate their phase which represents frequency.

Formant Table (Feature Tuning):

A formant is a concentration of acoustic energy around a particular frequency in the speech wave. Each formant corresponds to a resonance in the vocal tract. All vowels can be can be characterized by F1 and F2. Vowels traditionally known as *front* have F1 and F2 a good distance apart. Vowels traditionally know as *back* have F1 and F2 so close that they touch. The speaker trained himself by trying for a large number of attempts and made sure similar formants were achieved each time. The following table shows mapping for our formants to the corresponding speech input number. This was finalized after a large number of trials for optimum results for our speaker.

Table No. 1: Formants' mapping to Numbers

Number	Formant 1 (F1)	Formant 2 (F2)
One	200<F1<300	1400<F2<1750
Two	300<F1<370	800<F2<1000
Three	250<F1<350	2440<F2<2700
Four	400<F1<550	750<F2<950
Five	500<F1<650	900<F2<1200
Six	250<F1<400	2100<F2<2400
Seven	200<F1<300	1450<F2<1900
Eight	350<F1<480	2100<F2<2350
Nine	300<F1<400	2400<F2<2750
Ten	500<F1<650	1550<F2<2000

Block Diagram:

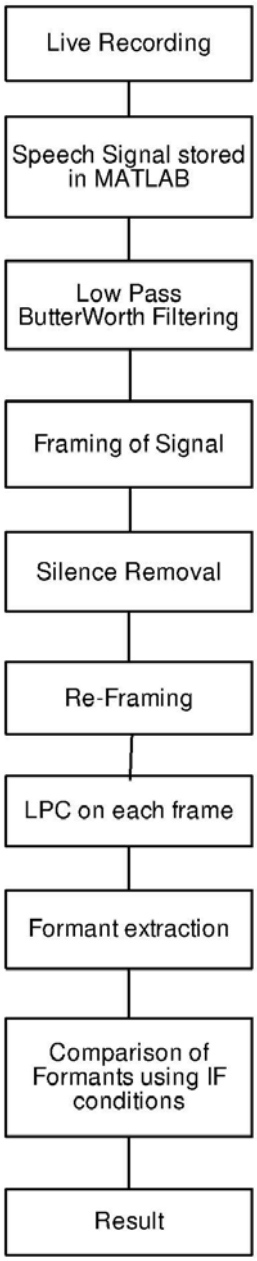

Figure No. 1: Flow Chart

Detailed Functionality via Code Breakdown:

Here is the generalized breakdown of the code. Complete annotated code is available at the end of this document for further reference.

1. We began by recording the voice (live) and storing signal is MATLAB for further processing. Sampling frequency of 11025 Hz was used and speaker's frequency was 2875 Hz. The code extract is shown below:

```
recObj = audiorecorder(11025,8,1);
disp('Start speaking.')
recordblocking(recObj,2);
disp('End of Recording.');
d = getaudiodata(recObj);
fs = 11025;
```

2. ButterWorth filter, which is a low pass filter, was applied on the signal. This allowed us to remove the high frequency noise components from the signal for smoother processing. Following is the code:

```
[b,a] = butter(3,2875*2/11025);
d = filter(b,a,d);
d=d./(1.01*abs(max(d)));
```

3. The signal was then normalized between zero and one to make computation and coding efficient and simpler. The normalized signal was then framed for further processing. Framing was done to break done the speech into individual letters e.g. from ONE to O, N and E. Frame size of 25ms was used. No. of frames=signal length/frame length.

```
d = filter(b,a,d);
d=d./(1.01*abs(max(d)));
f_d = 0.025;
f_size = round(f_d * fs);
n = length(d);
n_f = floor(n/f_size);  %no. of frames
temp = 0;
for i = 1 : n_f

   frames(i,:) = d(temp + 1 : temp + f_size);
   temp = temp + f_size;
end
```

4. A secret recipe of our code was silence removal which proved very fruitful for our project's success. This reduced the redundant part of the signal, increased the efficiency of code and hemce reduced the computation required to process the signal. Here is the code extract of silence removal and a sample outcome:

```
% silence removal based on max amplitude
m_amp = abs(max(frames,[],2)); % find maximum of each frame
id = find(m_amp > 0.1); % finding ID of frames with max amp > 0.03
fr_ws = frames(id,:); % frames without silence

% reconstruct signal
d_r = reshape(fr_ws',1,[]);
d_r=d_r';
plot(d_r); title('speech without silence');
hold on;
plot(d,'r');
legend('After Silence Removal','Original Signal');
```

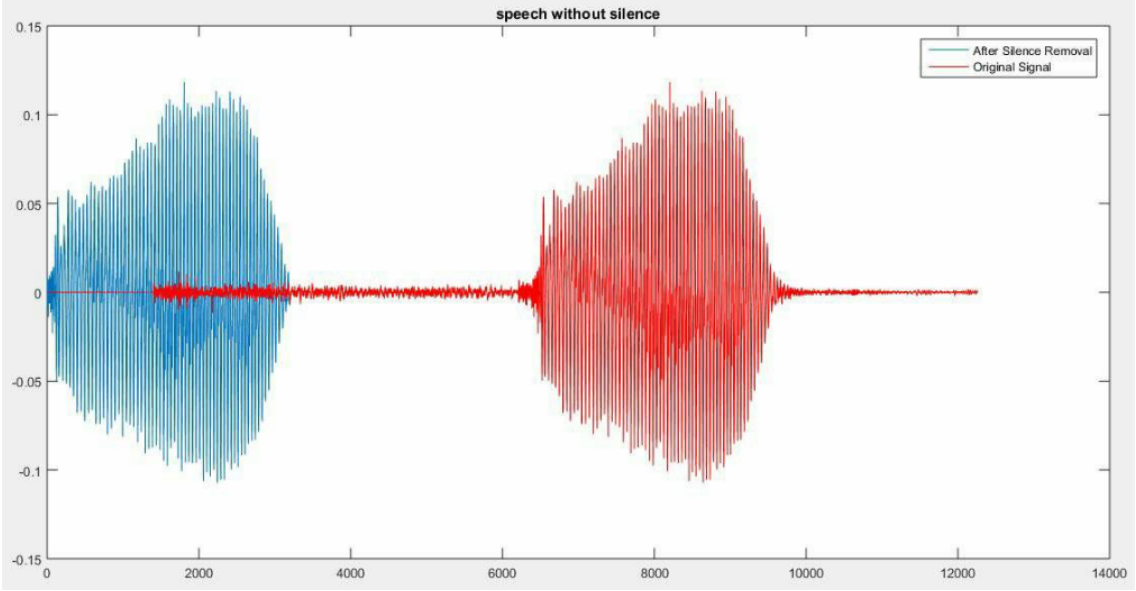

Figure No. 2: Comparison of silence vs non-silence signal

5. The most important stage was reframing of signal after silence removal and its Linear Prediction Analysis. The shortened signal is reframed and then LPC is applied on each frame to extract the formants. Pre-emphasis filter, which is a high pass filter, is first applied on reframed signal. Each frame of this signal then undergoes LPC Analysis. Positive imaginary roots are extracted and then their angle is calculated. This eventually yields the formants.

```
for i = 1 : n_ff

    frames(i,:) = d_r(temp + 1 : temp + f_size);
    temp = temp + f_size;

frames(i,:) = frames(i,:)'.*hamming(length(frames(i,:)));
preemph = [1 0.63];
frames(i,:) = filter(1,preemph,frames(i,:));
A = lpc(frames(i,:),8);
rts = roots(A);
rts = rts(imag(rts)>=0);%Because the LPC coefficients are real-valued,
angz = atan2(imag(rts),real(rts));
[frqs,indices] = sort(angz.*(fs/(2*pi)));
bw = -1/2*(fs/(2*pi))*log(abs(rts(indices)));
nn = 1;
```

6. The formants are tested using IF conditions as per the table given above. First, the number of frames are checked and then further verification is done by comparing formants 1 and 2 respectively. Here is a sample of testing for 'THREE' where formant 1 should fall in range 250-350 and formant 2 should fall in range 2440-2700.

```
if(n_ff<=29)
if  (req(i,1)>=250 & req(i,1)<=350)
 if(req(i,2)>=2440 & req(i,2)<=2700)
        e_3=e_3+1;
   end
```

7. After the formant conditions match, the index for particular number is incremented and a text output of the spoken word is given.

```
if e_3>0
    disp('THREE')
end
```

Results:

Table No. 2: Results of Speech Recognition

Number	Success (x/10)
One	7
Two	9
Three	10
Four	9
Five	8
Six	10
Seven	10
Eight	10
Nine	10
Ten	10

As seen from the table, the overall efficiency of the system exceeded 90%. Given that we recorded signal directly to MATLAB (online), the results were overwhelming.

One and five were a bit less successful than the rest of numbers, and the reasons are discussed in the next section.

Sampling rate of 11025 Hz was used and speaker's frequency was set at 2875 Hz.

Our speaker was Tajammal (MC-20) as he had the lightest voice amongst all group members, the rest of us have grave voice as compared to him.

Our system worked fine in almost all environments. It was tested at home, simulation lab and in robotics lab and it worked fine everywhere. The code malfunctioned a bit in Dr. Ahsan's office and the reason we think is that his office is very compact and sealed so the voice probably echoed and was read by our system along with original input.

Pre-emphasis filter, which is a high pass filter, was used as explained in the code breakdown earlier. Even though the low pass ButterWorth filter in the initial stage filtered out noise, we made sure that as less noise goes into the system as possible. So for this purpose, we ran the code by giving voice input via separate Bluetooth mic – LG HBM-290.

Discussion and Conclusion:

Discussion of Results and Learning:

Let's discuss our result for each input:
One: While speaking one, we elongated it a bit. This ensured lesser silence and more voiced part. It still had the least success rate among our all inputs signal length didn't exactly match our requirement due to uneven prolonging for each attempt.
Two: Two worked fine without any elongation. The reason is the sound of 'ou' at the end is solely in two, and no other number.
Three: Three worked perfectly. We elongated the end part of three 'ee' a bit. This also helped us in differentiating the end part of three from nine as both end with 'ee' sound.
Four: Four worked fine as we spoke the 'o' part in a bit grave voice. This made sure it was the only number with such specs.
Five: The 'I' part was elongated a bit while speaking five. The results were satisfactory.
Six: Six worked perfectly fine as it is a fast word as shown by its spectrogram and it had lesser frames and this uniqueness made it work perfectly.
Seven: Seven was spoken a bit fast and worked fine as this made sure there are lesser frames.
Eight: Eight also worked fine as it had shorter length like six and its starting vowels were unique.
Nine: Nine was elongated a bit by placing more focus on the 'I' part and it made it work perfectly.
Ten: The 'en' part was given more weightage in speaking ten and this uniqueness made it work fine.

We then gave rhyming words of different numbers and input and they gave a good response. Fun was recognized as one, heaven was detected as seven, moo matched with two, free matched with three and hen matched with ten. This was because the focused formants/parts in rhyming words were same as those in numbers and hence they matched the conditions.

The deficiency in our code is that it is strictly speaker dependent. It won't work well for some other speaker. In this code, we trained the speaker. For the code to be speaker independent, we would need to train the code. That would require more man hours as input, making it harder, which was not required for current application. The AI concepts such as Neural networks would be implemented in such case.
We learnt about speech feature extraction e.g. formants. We were able to learn about different schemes that could be used to make more sophisticated speech recognition system e.g. Cepstrum, MFCC, etc. It made us realise that the top applications of speech recognition like Google's speech recognition system would be so perfectly coded and trained and greatly robust as they work amazingly fine. This also gave us the idea how the speech recognition research in different areas might be oriented e.g. the one in ITU where the researcher tries to read the signals of Dolphins.

Conclusion:

Speech recognition is an important goal of engineering worldwide, but to enable the broader study of speech processing as a whole, we have to study it in chunks. For this project, we focused on vowel recognition, studying the theory behind formants and random processes that would enable us to build a highly successful program. Our initial development allowed us to recognize one vowel at a time, but upon further inquiry, we were able to implement an intelligent vowel recognition algorithm for streams of isolated words.

Future Work:

In the future, we might hope to be able to implement some more effective ways to get rid of consonant effects in testing this program. Or better yet - we could figure out how to process the consonants and develop a guess of what those might be as well! An interesting idea for the current status of the project would be to implement a Markov chain relating a sentence to the vowel flow in that sentence. That way, if a user spoke a short phrase, we could make a guess as to what sentence he said!

Team Contribution:

Table No. 3: Contribution by members (Zoom in for better view)

Member	Tasks										
	Cepstrum Analysis	MFCC	LPC	Butterworth	Framing	Silence Removal	Reframing and LPC	Formant extraction	Comparison	Result	Report
2014-MC-1	√			√	√	√		√			√
2014-MC-20		√	√			√	√	√	√	√	
2014-MC-24	√		√	√			√	√	√		
2014-MC-32		√			√			√		√	√

Team Leader: 2014-MC-20