DESIGNING A SIMPLE SPEECH RECOGNITION SYSTEM

This project asks you to use the Fast Fourier Transform (FFT) to extract information from speech signals. In order to complete this project, you need to calculate the FFT, convert between the FFT bin number and the corresponding frequency in Hz, and interpret graphs of the FFT of signals.

You also would be able to design digital filters to remove noise or unwanted signals from the available speech signals. To do so, you will design digital filters to suppress the noise such that it will no longer affect the decision making of the speech recognition system. The specifications of the filter are not given, which requires the students to design a filter, test it, and if necessary, modify it.

The first scenario: As the main part of this project, you will design a simple speech recognition system which is able to recognize if a recorded voice signal is "Yes" or "No". This part of the project can be implemented in the following steps:

- Data collection
- Partitioning the database into training and test sets
- Calculating FFT
- Feature extraction
- Classification
- Test and verification

The second scenario: Aiming to improve the performance of the speech recognition system in the first scenario, students will redesign the system to make it capable of working under a noisy condition. That is, when a specific noise (such as a single tone) is added to the voice samples.

The third scenario: In this scenario, students write a code which is capable of recognizing "Yes" and "No" from the echo-affected recording samples.

The fourth scenario: As the fourth scenario, you will design a speech recognition system which is able to distinguish male speaker from the female ones. This part can be implemented using an approach similar to the first scenario. The database prepared (data collected) for the first scenario can also be used for this scenario.

The fifth scenario: In this scenario, you will design a speech recognition system similar to that of the first scenario which is able to distinguish four arbitrary chosen words (instead of the two words "Yes" and "No" in the first scenario).

The sixth scenario: Implementing the fifth scenario on a remote control toy car

I. The first Scenario

A. Data Collection

Use Matlab to record more than ten voice samples for words "Yes" and "No" uttered by different people. You should collect/record samples said by both male and female speakers. Note that increasing the number of recording samples and/or the range of diversity will improve the system performance.

To record data from an audio input device (such as a microphone connected to your system) for processing in MATLAB:

- Create an `audiorecorder` object.
- Call the `record` or `recordblocking` method to record voice samples for test, where:
  - `record` returns immediate control to the calling function or the command prompt even as recording proceeds. You can specify the length of the recording in seconds, or end the recording with the stop method.
  - `recordblocking` retains control until the recording is complete. You can specify the length of the recording in seconds.
- Create a numeric array corresponding to the signal data using the `getaudiodata` method.

The following example shows how to use the `recordblocking` method to record voice for 2 seconds:

```
recObj = audiorecorder;
disp('Start speaking.');
recordblocking(recObj, 2);
disp('End of Recording.');
play(recObj);
y = getaudiodata(recObj);

plot(y);
```

Using `audiorecorder`, you would be able to set the sampling frequency and the number of bits per sample. See `Matlab Help` for further information.

Keep the length of the recordings less than or equal to 2 seconds and record voice samples in a noise-free or very low-noise environment.

B. Partitioning the database into training and test sets

Partition the set of "Yes' words said by male speakers into two sets. One set containing 75 percent of randomly chosen audio files and the other containing the remaining 25 percent. We call the first set the training set and the latter the test set. We then randomly choose 75 percent of the "Yes" recording files uttered by female speakers and add them to the training set. We then add the remaining 25 percent of female uttered "Yes" recordings to the test set. Using the same approach, we build training and test sets for "No" recording files.

C. Calculating FFT

Take a pair of "Yes" and "No" samples uttered by the same speaker, calculate the corresponding FFT. Plot the FFT spectrums of "Yes" and "No" samples uttered by the same speaker in the same figure

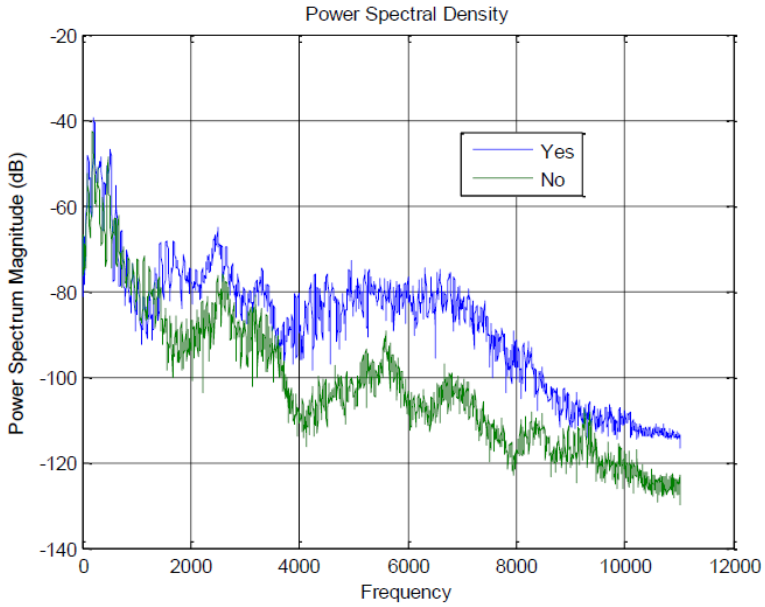

Fig. 1. Power Spectral Density for "Yes" and "No" by the same speaker (Reference: Dr. Joseph P. Hoffbeck, University of Portland).

in which different colours are used for FFT of different words. Choose the X-axis of the plot as the frequencies in the range $(0, f_s)$ and plot the Y-axis in log-scale. Plotting the same figure for more pairs of "Yes' and "No" files uttered by the other speakers, you should observe that the speech samples of "Yes" words have more energy in the high frequencies (because of the "S" sound in "Yes").

Note that you should once use the embedded `fft` command available in Matlab and then write a function to calculate the DFT using its definition. You should then compare the running time for calculating each of these ways of obtaining DFT transform. Moreover, students should discuss what should we do if instead of a computer, we want to implement the FFT on a digital signal processors such as TI C6000 DSPs.

D. Feature Extraction

In order to distinguish "Yes" from "No", it is necessary to find a feature that can be computed on an unknown signal whose value is usually small for "Yes" and large for "No" (or vice versa). The power spectral density, which is based on the FFT, is a plot of the estimated spectrum of a signal. If the power spectral density of a recording of a person saying "Yes" is compared to that of "No", students can see that usually the spectrum of "Yes" has more energy in the high frequencies because of the "S" sound in "Yes" (see Figure 1). Therefore one possible feature would be to take the sum of the magnitude of the FFT components that correspond to the low frequencies and divide it by the sum of the magnitude of the FFT components that correspond to the high frequencies.

As such, for example, one possible feature is the sum of magnitude of the FFT values for the frequencies

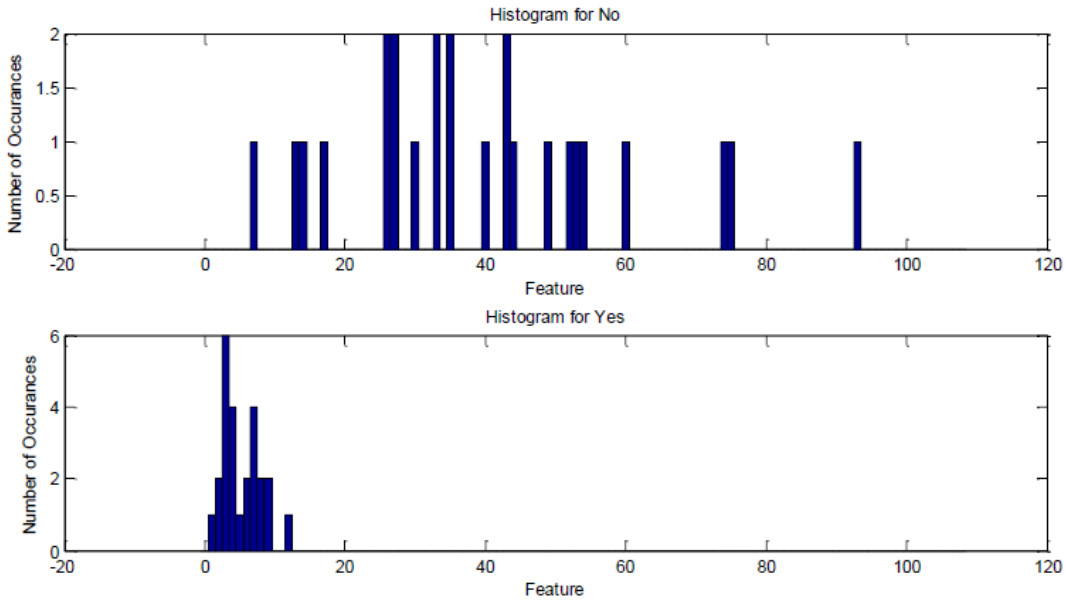

Fig. 2. Histogram of Feature Values (Reference: Dr. Joseph P. Hoffbeck, University of Portland).

0 to $f_s/4$ divided by the sum of the magnitude of the FFT values from $f_s/4$ to $f_s/2$, which is the highest frequency for the training files, when f_s is the sampling rate. A recording that contains the word "No" will usually have a higher value than would be found if the recording contained "Yes". Since this feature is a ratio, its value is not affected by different volume levels. Also, this feature is insensitive to the duration of the recording.

E. Classification

Once a feature is selected, it is necessary to find a threshold value that separates the feature value for "Yes" from that of "No". One way to find an appropriate threshold is to calculate the feature for all of the recording files in the training set and to examine the histogram for the "Yes" values and the "No" values. You should compute this feature for all of the training files. The feature values from the "Yes" files are stored in one vector, the feature values from the "No" files are stored in a different vector, and then histograms are generated for each vector. Careful examination of the histograms in Figure 2 reveals that, for this particular choice of feature, a threshold value of 12 separates most of the "Yes" and "No" values.

F. Test and Verification

Once the feature and threshold values are determined, the contents of a new, unknown recording can be determined by computing its feature value and comparing it to the obtained threshold. The following MATLAB program shows how one can test if an audio file contains "Yes" or "No":

```
function output = YesNo(x,fs)
% function output = YesNo(x,fs)
% Simple algorithm for deciding whether the audio signal
% in vector x is the word 'yes' or 'no'.
% x (vector) speech signal
% fs (scalar) sampling frequency in Hz
% output (string) 'yes' or 'no'
threshold = 12;     % threshold value
N = length(x);
k1 = round(N*5000/fs);
k2 = round(N*11025/fs);
X = abs(fft(x));
f = sum(X(1:k1))/sum(X(k1:k2));
if f < threshold
    output = 'yes';
else
    output = 'no';
end
```

You can use this program to test how well your algorithm works on the training set of recording files. Use the recording files in the training set to determine the accuracy of your algorithm and to see how well feature and the threshold are selected. If the accuracy of your algorithm is less than 80 percent, you probably need to change the cut off frequency and/or the threshold to increase the accuracy.

If your code is not working properly for all “Yes” and “No” tests try to make sure you are on the right path by testing your code using a recording of a continuous “S” sound for “Yes” and/or a continuous “O” sound for “No’.

II. The Second Scenario

Aiming to improve the performance of the speech recognition system in the first scenario, students will redesign the system to make it capable of working under a noisy condition. That is, when a specific noise (such as a single tone or a beep) is added to the voice samples. In this scenario, students should take the following steps:

- Generate a single tone with frequency 2500 Hz as a specific type of noise.
- Add the generated noise (here a single tone) to the recorded voice samples. Increase the amplitude of the tone until the voice sample is no longer audible.
- Calculate and illustrate the FFT.
- Design an FIR filter that filters the added tone while allows as much as possible of the voice samples to pass.
- Apply the designed filter to the noise-contaminated voice signal.
- Use the system designed in the first scenario to distinguish “Yes” from “No” when is applied to the filtered version of the noise-contaminated voice samples.

- If needed, modify the speech recognition system (change threshold or cut-off frequency) to make it capable of working with the filtered signals.
- You should discuss how different values of noise frequency and amplitude can affect your hearing performance as a speech recognition system and also discuss how these values can affect the performance of your designed speech recognition system. Determine the highest frequency of the noise signal that can affect the hearing system of each group member.

III. The third Scenario

In this scenario, students write a code which is capable of recognizing "Yes" and "No" from the echo-affected recording samples. In this scenario, students need to

- Generate a 20 ms delayed copy of each recording and subtract it from the main recording
- Play the generated echo-affected audio file to see how echo can affect the original signal
- Calculate and illustrate the FFT
- Design a filter (inverse system of the echo-generating system)
- Apply the designed filter to the echo-affected signal
- Use the system designed in the first scenario to recognize "Yes" or "No"
- If needed, modify the speech recognition system to work with echo-contaminated signals
- You should discuss how much delay in an echo-affected signal can affect your hearing performance as a speech recognition system and also discuss how echo can affect the performance of your designed speech recognition system.

IV. The fourth Scenario

As the fourth scenario, you will design a speech recognition system which is able to distinguish male speaker from the female ones. This part can be implemented using an approach similar to the first scenario. Students can use the same database as that for the first scenario. Steps are as follows:

- Data collection (same as the first scenario)
- Partitioning the database into training and test sets (same as the first scenario)
- Calculating FFT (same as the first scenario)
- Feature extraction (Students should find a (set of) feature(s) that can be used for distinguishing male from female speaker)
- Classification (similar to that of the first scenario, but with different value(s))
- Test and verification

One of the features that can be used for gender recognition is **pitch** or **fundamental frequency**. The male fundamental frequency changes between 85 Hz to 180 Hz, whereas a typical female fundamental frequency is between 165 Hz to 255 Hz. Note that these fundamental frequencies change with age such that as age increases the mean of the fundamental frequencies drop down. The fundamental frequency, is defined as the lowest frequency of a periodic waveform. In terms of a superposition of sinusoids (e.g. Fourier series), the fundamental frequency is the lowest frequency sinusoidal in the noiseless sum. Since the fundamental is **the lowest frequency** and is also perceived as **the loudest**, the ear identifies it as the specific pitch of the recorded voice. You can take pitch as a feature and use it for gender recognition. Doing so, you can determine a threshold around 200 Hz. And if the pitch frequency of a person is below the threshold, he is a male. Otherwise, she is a female. Note that, the pitch-based system works well only with clean speech (very low level of noise). You can divide the recorded samples of voice into frames of length 25 ms. After extracting all the pitch information of all the frames, the mean value of the pitches of all these frames can be considered as the pitch of the recorded sample.

V. The fifth Scenario

As the last scenario, you will design a speech recognition system which is able to distinguish four arbitrary chosen words. These words can be used to control a remote-control toy car. The chosen words are interpreted as go, stop, turn left and turn right commands. The words such as “go”, “stop”, “left” and “right” can properly convey the message. However, if you find recognizing these words difficult, you can choose four different words with more observable difference in their FFT spectrums. The steps one should take are as follows:

- Data collection (similar to that of the first scenario, but this time with four arbitrary words)
- Partitioning the database into training and test sets (similar to that of the first scenario, but this time with four arbitrary words)
- Calculating FFT
- Feature extraction (You should come up with new features. Chosen words have direct effect on the level of difficulty of this step. i.e., a wise selection of words can make this step very simple.)
- Classification
- Test and verification