EEC 201 Final Project

Automatic speaker recognition based on VQ Clustering

Gengchen Liu

I. Introduction

Automatic speaker recognition (ASR) plays an important role in applications such as security, mobile device forensics, and identity authentication [1]. Fig. 1 shows the basic architecture of an ASR system [2]. Because of the fact that raw voice signal contains way too much information, it is critical to perform feature extraction to obtain compact and robust representations from the original signal. These representations are used to train and evaluate certain reference models. Once we have a set of well-trained models, we can use these models to test an unknown input. In this project, I adopted a pre-processing stage for the input voice signal before the feature extraction, which increases the resilient of the ASR model we built. Vector quantization (VQ) clustering is used as the adaption method to train the reference models. To verify the performance of our ASR system, we evaluate our system by using three distinct testing datasets, which include the testing sets provided by Prof. Zhi Ding, an augmented testing sets obtained from the training sets with additive noise, and another augmented testing sets obtained by notch-filtering the training sets. Very high recognition accuracy has been achieved for both of the testing sets.

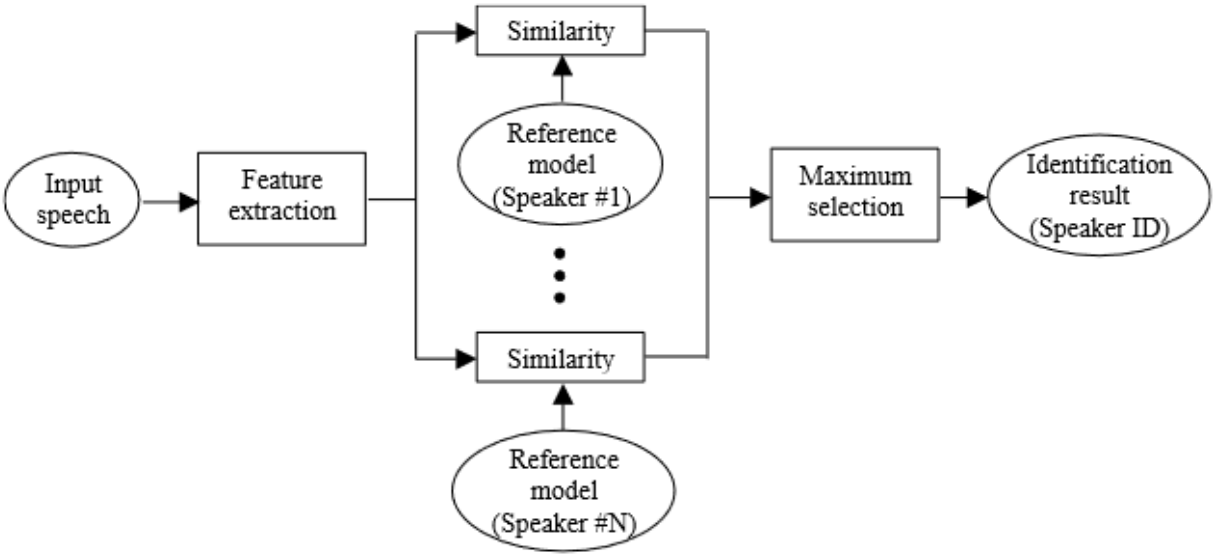

Figure 1. Architecture of the ASR system

This report is organized as follows. Section II presents the design details for each of the sub blocks of our ASR system. Section III shows and discusses the results from the evaluation sets. Section IV summarizes this work.

II. System design

1. Input pre-processing

Our input voice signals contain consist of 11 different speakers saying the word “zero”. The sampling rate for each voice signal is 12.5 kHz. Fig. 2 visualizes the raw voice signal from speaker #1 and speaker

#3. Notice that the raw voice signal from each speaker contains “quite” regions. These “quite” regions cannot contribute to the task of speaker recognition since they contain no information regarding the speaker. To overcome this issue, we performed a block-by-block energy detection operation on the raw signal with block size equals to 256. For each block of 256 samples, if the root-mean-square (RMS) value of this block is lower than 1e-4, the current block will be identified as a non-voice block and discarded from the raw signal. We also subtracted the mean of the raw signal, as well as performed amplitude normalization.

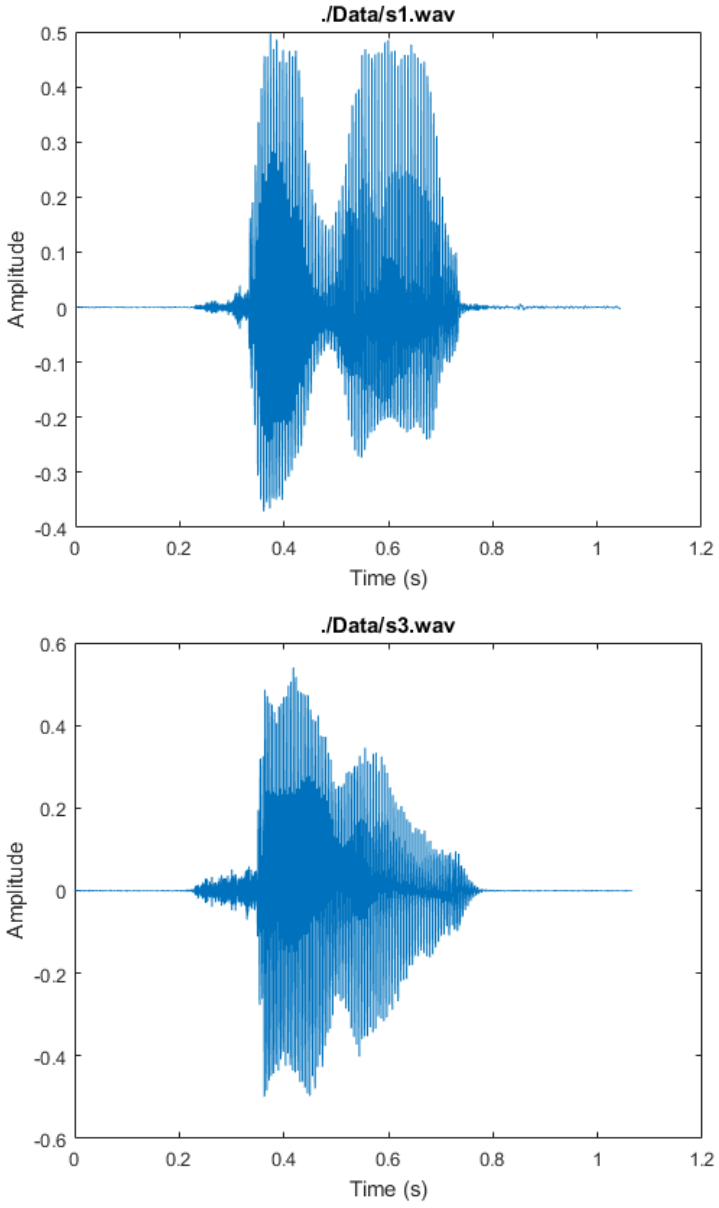

Figure 2. Raw voice signal for speaker #1 and #3

Once the above operations are completed, we performed block-by-block windowing with hamming window define in Eq. 1. The block length is 128, 256, and 512 whereas the incremental size is set to 43, 85, and 171 respectively. Fig. 3 shows the overlapped and windowed samples, notice the quite regions are removed. We will be using these samples to calculate MFCC coefficients in the next steps.

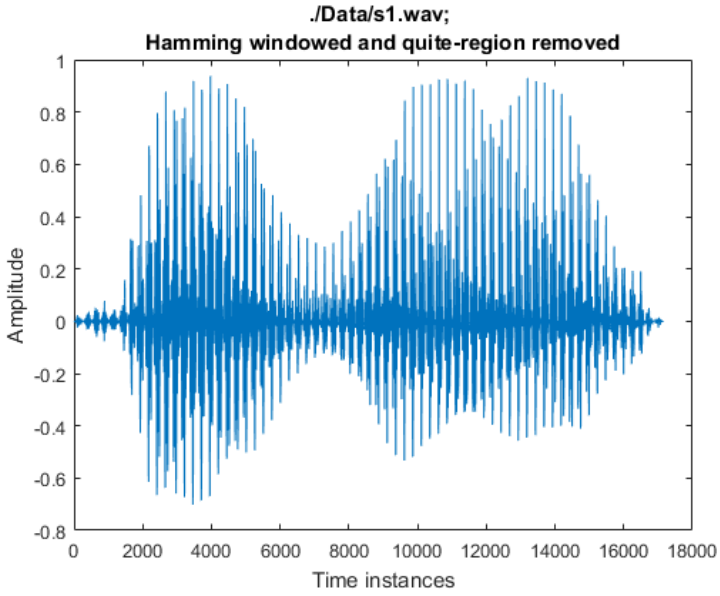

Figure 3. Hamming windowed speaker #1 with normalization and quite-region removed.

$$w(n)=0.54-0.46\cos(2\pi nM-1)\ ,\quad 0\le n\le M-1 \qquad (1)$$

2. Feature extraction

The first step for calculating MFCCs is performing short-time Fourier transform (STFT) on the windowed samples. The FFT length should match the block length in the previous step, which is 128, 256, and 512 respectively. Once the STFT is completed, we perform square-magnitude operation on the spectral data to obtain the periodogram, which is shown in Fig. 4 (I added the frequency index manually). From the periodogram we should notice that most of the energies are contained in the low-frequency regime.

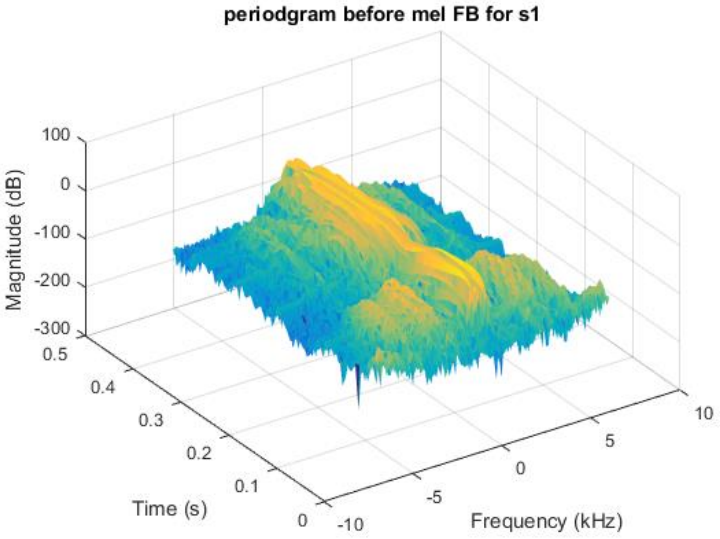

Figure 4. Periodogram for speaker #1.

In the next step, we need to perform MEL frequency warping on the periodogram to reduce the dimensions of the processed signal. We choose to use 26 MEL filter-banks as shown in Fig. 5(a). Notice that a pre-emphasis of 1.3 is added to the filter-banks that corresponding to higher frequencies. This allows us to put a higher emphasis on the high frequency components of the processed signal when performing the MEL frequency warping. By counting the energies located in each of the MEL filter-bank, we get the MEL-warped periodogram plot shown in Fig. 5(b). Comparing Fig. 4 and Fig. 5(b), one should notice that MEL filter-banks "smooth" out the original periodogram but still preserves the temporal-spectral shape of the original periodogram, which indicates our dimensional reduction technique (with filter-banks) did works. Only 1 sideband of the spectral shape is preserved since the input is real.

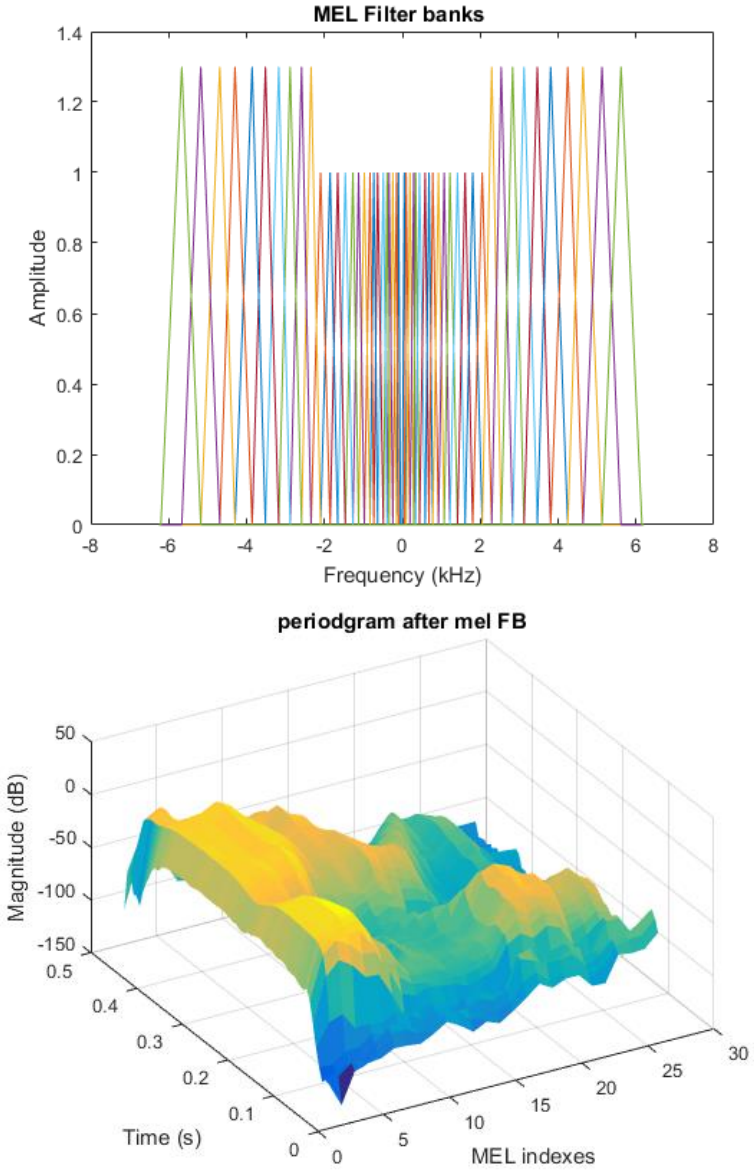

Figure 5. (a) Frequency response of the 26 MEL filter banks; (b) Periodogram for speaker #1 after the MEL warping

The MEL-warped periodogram is then converted to temporal domain by using discrete-cosine transform (DCT). Since the 1st element of the DCT coefficients only quantifies mean value of the input block which does not provide us much information, we removed the 1st element from the DCT coefficients. We keep the 2nd to 14th DCT coefficients as our features, resulting in 13 MFCC coefficients for each time instance. Fig. 7(a) shows the MFCC feature map for speaker #1 and Fig. 7(b) and (c) visualizes the MFCC coefficients on two dimensions for speaker #1 and speaker #5. One can observe distinct differences between the two different speakers. We will be using these training samples' MFCC features to let our reference models learn the unique characteristics of each speaker. Fig. 6 shows the block diagram of the whole feature extraction module we discussed above.

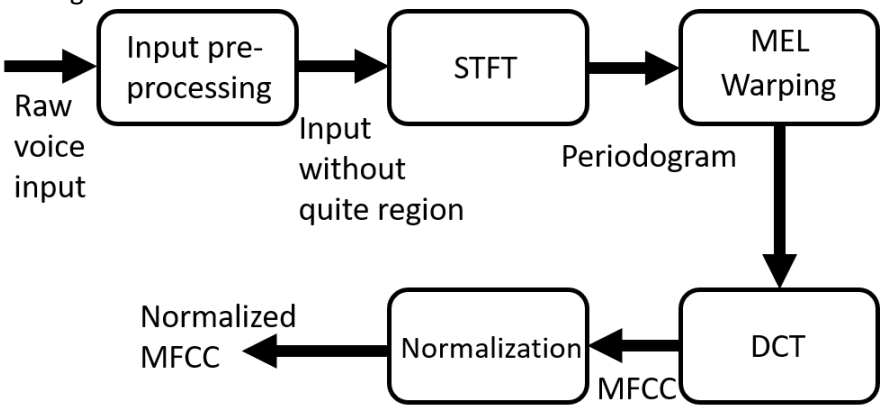

Figure 6. Block diagram for feature extraction module

3. VQ clustering

Before we start the training process, it is of great importance to normalize the obtained MFCC features as well. We achieve MFCC features normalization by subtract the mean value of the features and normalize their maximum magnitude to 1.

After the normalization, we initialize the training process by generating one centroid vector whose values correspond to the centroid of all training vectors. If the targeted cluster number is larger than 1, we split the current centroid vector we had into two new centroid vectors. By performing nearest-neighbor search, we update the centroid vectors based on the training samples that belongs to each current centroid point. This process is executed iteratively until the number of centroid vectors matches the targeted cluster number. There are couple of hyper-parameters here involved in this training process that we can play with, such as the targeted cluster number, the step size used to perform the centroid splitting, the error threshold used to trigger a new splitting, and the way we define our error threshold. In this work, we have been using 32 clusters, 1e-2 as the step size, 12 as the cumulative error threshold, and L2 distance as the error criteria. The selection of these hyper-parameters mainly involves in trail-and-error method. Fig 8. (a) and (b) visualizes the learned centroids for speaker #1 and speaker #5 with 8 clusters. Clearly the training process is converged, and we are able to obtain good reference model for each speaker.

Another technique that I have been using to improve the performance is to concatenate the MFCC features of one speaker multiple times in the time-domain. We may or may not add a small perturbation between each concatenation. Even if the concatenated features are identical, it can still enhance the evaluation accuracy for the three tests we will be covered in the next section.

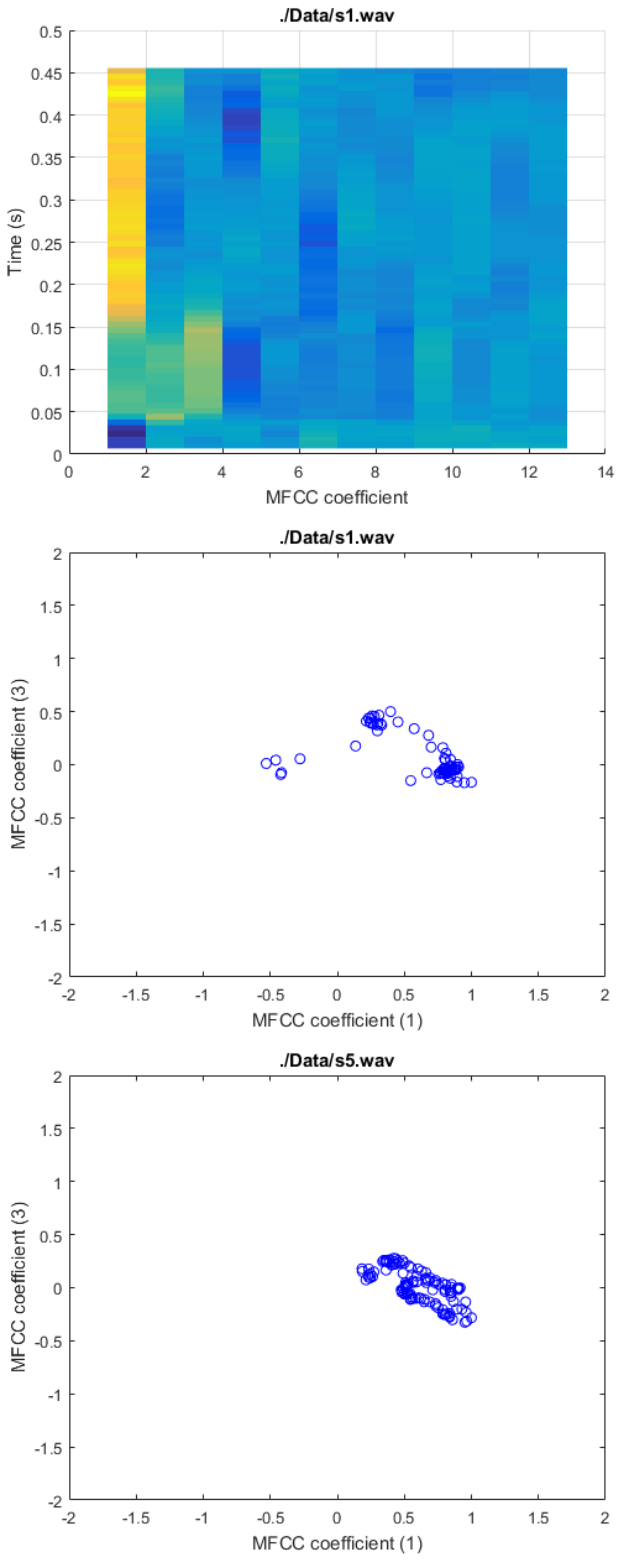

Figure 7. (a) MFCC map for speaker #1; (b) Visualized MFCC distribution for speaker #1; (c) MFCC distribution for speaker #5.

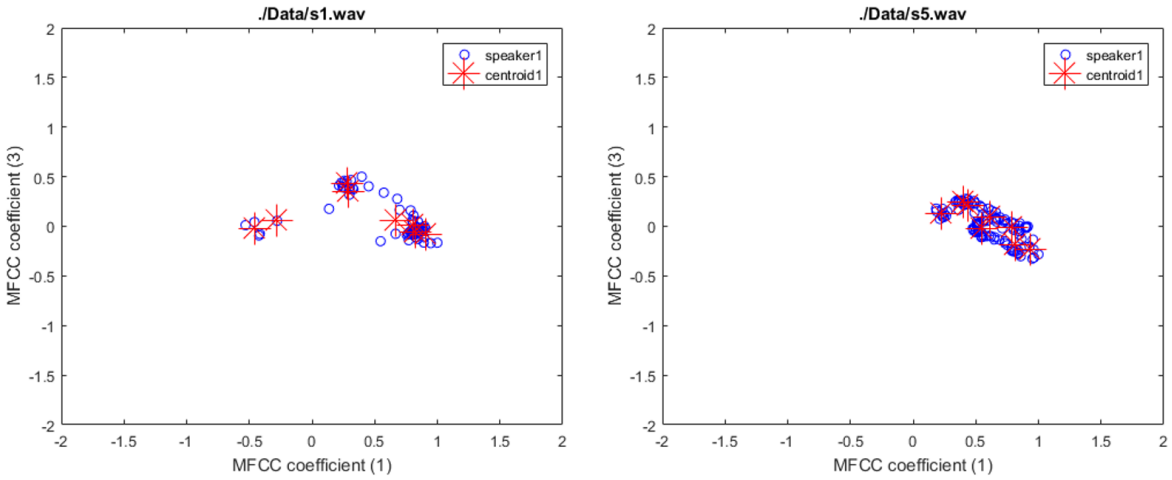

Figure 8. Converged centroids for speaker #1 and #5.

III. Results

1. Testing samples provided by Prof. Ding

In the first evaluation, we tested the performance of our ASR system using the testing samples provided by Prof. Ding. Since only 1 testing samples is provided for each of the eight speakers, the evaluation result will be a binary value – either 100% or 0%. The way we utilize our ASR system is by calculating the L2 distance between the MFCC coefficients of the testing samples and each of the eight reference models. A successful recognition is registered if the corresponding reference model produces the lowest L2 error otherwise a failure is registered. Table 1 illustrates the measurement accuracy for the eight testing samples. We have achieved 100% accuracy for all testing samples, which indicates the strong performance of our ASR system. This strong performance can be easily interpreted if we visualize the MFCC distribution between the training sample and testing sample as shown for speaker #2 and speaker #6 in Fig. 9. Similarities of the MFCC feature distribution between the training and testing sample indicate the robustness of these features, resulting in high recognition accuracy.

Table 1. Testing sets provided by Prof. Ding

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%

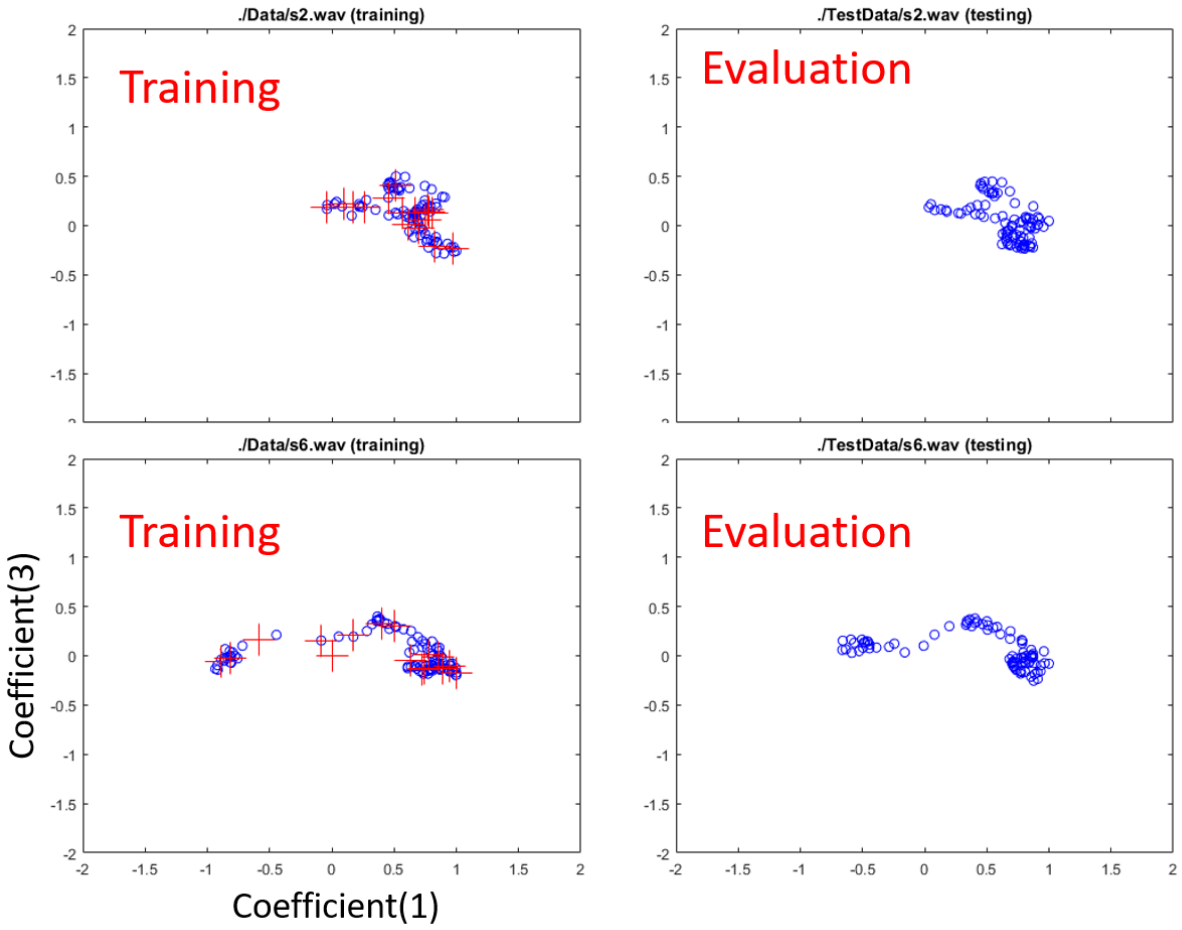

Figure 9. Similarity between training and testing features

2. Testing samples obtained by adding noise

In the second test, we added different amount of noise on both training and testing data, and we use those testing data to evaluate the recognition accuracy. Three different level of white gaussian noise are added, with variance of 0.001, 0.01, and 0.1. These variance values corresponding to SNR (it is quite ambiguous on how we should define SNR here…) values approximately equal to 30 dB, 10 dB, and 3 dB. Fig. 10 illustrates the amount of noise we added, notice that the spectral shape of the signal might "merge" under the noise floor. Table 2-4 show the measurement accuracy for the noise-corrupted testing sets with different SNRs. At high SNR, the performance of our ASR system is nice as 100% recognition accuracy can be achieved. With 3 dB SNR, speaker #9's recognition rate degrades to 90% and speaker #11's recognition accuracy degrades to only 30%. The performance degradation can also be explained by visualizing the MFCC maps shown in Fig. 11. Notice that the heavy amount of noise "hides" the distinct features of each speaker's MFCC, which results in the reduced recognition accuracy.

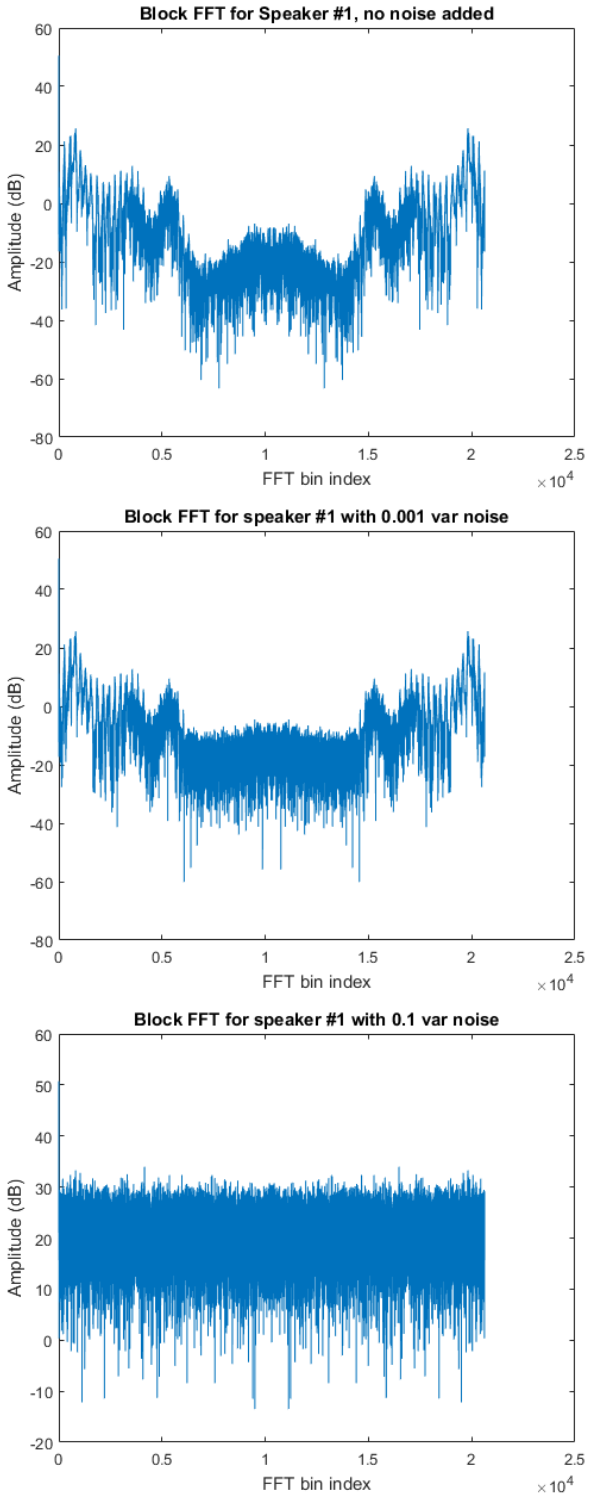

Figure 10. Noise corrupted training data. Notice that the x-axis represents frequency components from DC to 12.5 kHz

Table 1. Testing sets with 0.001 gaussian noise (~30 dB SNR)

Speaker	Accuracy
#1	100%

#2	100%
#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

Table 3. Testing sets with 0.01 gaussian noise (~10 dB SNR)

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

Table 4. Testing sets with 0.1 gaussian noise (~3 dB SNR)

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	90%
#10	100%
#11	30%

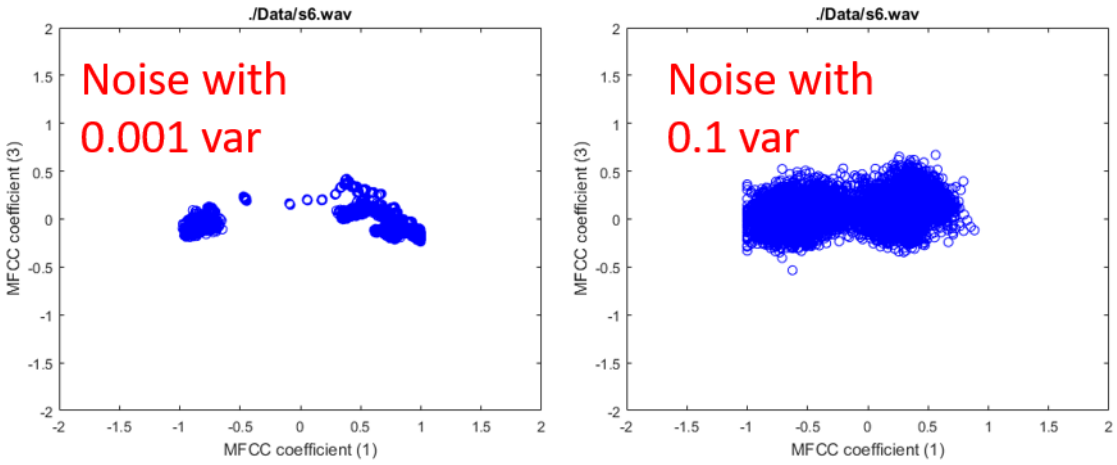

Figure 11. Effect of heavy amount of noise on MFCC distribution. The clipping is due to the normalization

3. Testing samples obtained by notch filtering

In the third test, we applied notch-filtering on the training signal to generate different testing signals. The bandwidth of our notch filter is 100 Hz, 300 Hz, and 500 Hz centered at DC, 500 Hz, and 3000 Hz. Fig. 12 shows the effect of notch filtering on the training samples. Table 5-7 shows the measurement accuracy for the notch-filtered signal with 100 Hz BW. Notice that 100% recognition accuracy is achieved. Table 8-10 shows the measurement accuracy for the notch-filtered signal with 300 Hz BW. We start to observe failures for different speaker when the notch filter is located on DC and 500 Hz. If we further increase the notch filter BW to 500 Hz, lots of errors are produced when the notch filter is centered at DC, but 100% recognition accuracy can still be achieved if the notch filter is centered at 3000 Hz shown in Table 11-12.

Table 5. Testing sets 100 Hz notch filtered at DC

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

Table 6. Testing sets 100 Hz notch filtered at 500Hz

Speaker	Accuracy
#1	100%
#2	100%

#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

Table 7. Testing sets 100 Hz notch filtered at 3000Hz

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

Table 8. Testing sets 300 Hz notch filtered at DC

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	0%
#5	100%
#6	100%
#7	100%
#8	0%
#9	100%
#10	100%
#11	100%

Table 9. Testing sets 300 Hz notch filtered at 500Hz

Speaker	Accuracy
#1	100%
#2	0%
#3	100%

#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

Table 10. Testing sets 300 Hz notch filtered at 3000Hz

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	100%
#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

Table 11. Testing sets 500 Hz notch filtered at DC

Speaker	Accuracy
#1	0%
#2	0%
#3	0%
#4	0%
#5	0%
#6	0%
#7	100%
#8	0%
#9	0%
#10	0%
#11	0%

Table 12. Testing sets 500 Hz notch filtered at 3000Hz

Speaker	Accuracy
#1	100%
#2	100%
#3	100%
#4	100%

#5	100%
#6	100%
#7	100%
#8	100%
#9	100%
#10	100%
#11	100%

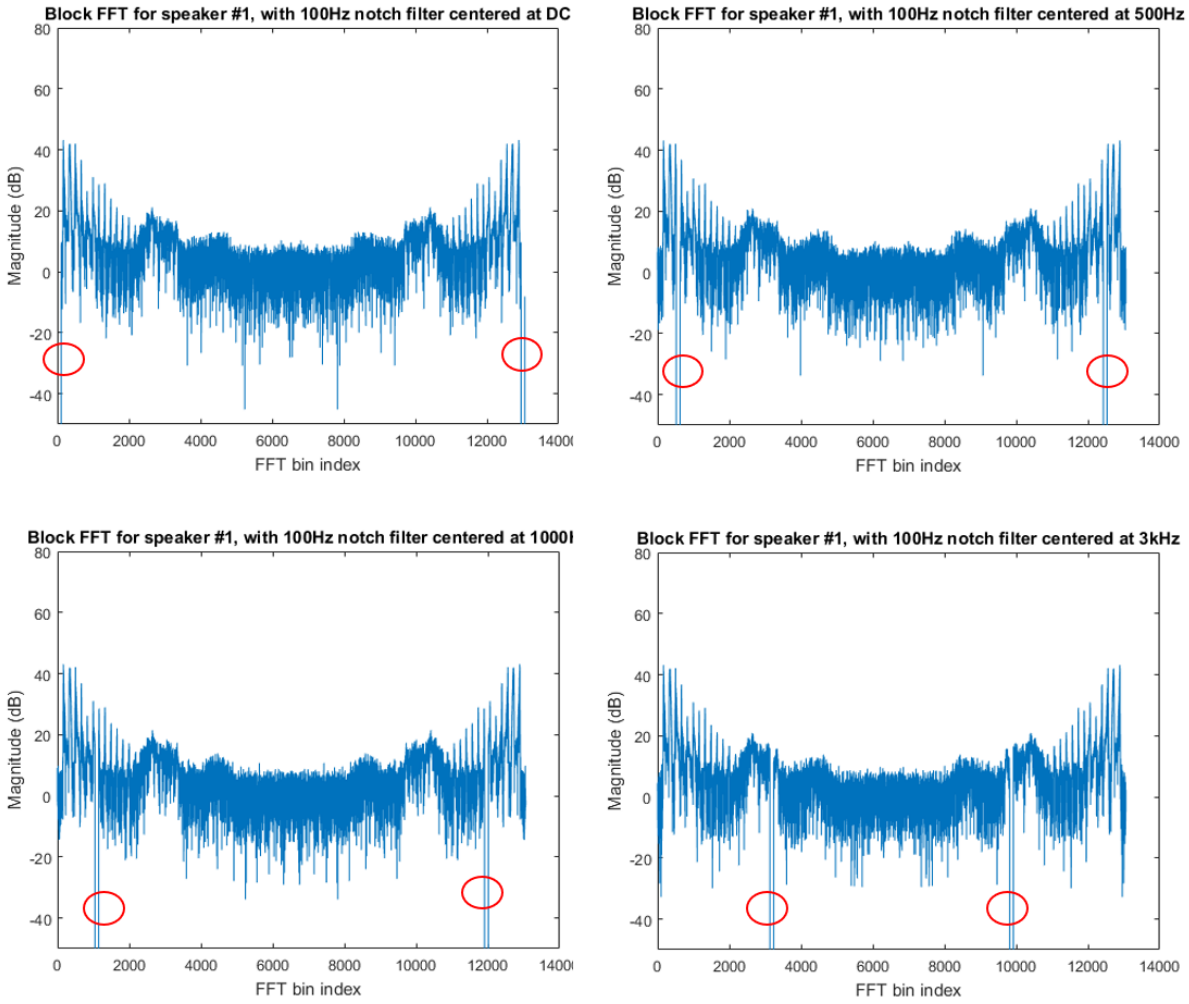

Figure 12. Notch filtered training signals; Notice the x-axis corresponding to frequencies from DC to 12.5 kHz

From these measurement results we can conclude that low-frequency components are definitely more important in the tasks of speaker recognition.

IV. Remarks

In this project, we build an ASR system using MFCC coefficients and VQ clustering models with some custom pre-processing techniques to boost the performance. We evaluate our ASR system based on three different evaluation schemes. High recognition accuracy is achieved for all evaluation schemes, indicating the robust and high performance of the ASR we built.

V. Answers to some of the questions asked by the project description

Q: Can you distinguish the voices of the 11 speakers in the database? …… Record what is your (human performance recognition rate)

A: ***I can distinguish the voices of the 11 speakers in the database. The human recognition rate in my case is 100%.***

Q: Compute how many milliseconds of speech are contained in a block of 256 samples?

A: ***Sampling rate = 12.5 kHz; Time duration of a block of 256 samples = 20.48 millisecond***

Q: Use STFT to generate periodogram, locate the region in the plot that contains most of the energy, in time (msec) and frequency (in Hz) of the input speech signal. Try different frame size

A: ***See the tables below.***

Q: record one student's voice and test the performance.

A: ***I recorded myself saying "zero" two times, labeled as s12.wav for training and s12_test.wav for testing. We can run test_gengchen_voice.m to check the performance. 100% recognition accuracy is achieved. Seen in the figures below. The 12th codebook provides the lowest L2 error.***

```
s12_test.wav


ans =

  Columns 1 through 10

    0.4173    0.5038    0.4307    0.4799    0.4008    0.4758    0.4232    0.5062    0.4920    0.5047

  Columns 11 through 12

    0.4979    0.1981
```

Other requirements, such as plotting, and explanations are covered in the previous sections.

Table 13. N=256

Speaker	Time slot (s)	Frequency (Hz)	Periodogram amplitude (dB)
#1	0.3536	390.6	15.42
#2	0.6052	781.3	11.13
#3	0.3672	488.3	12.44
#4	0.4692	390.6	13.6
#5	0.5168	439.5	19.28
#6	0.51	488.3	15.62
#7	0.578	683.6	8.79
#8	0.3876	293	14.45
#9	0.8364	634.8	9.44
#10	0.6868	48.83	10.22
#11	0.9248	537.1	10.95

Table 14. N=512

Speaker	Time slot (s)	Frequency (Hz)	Periodogram amplitude (dB)
#1	0.3536	366.2	20.86
#2	0.5848	781.3	18.23
#3	0.3536	488.3	16.75
#4	0.4624	366.2	19.14
#5	0.5168	415	24.99
#6	0.4896	488.3	22
#7	0.5712	659.2	14.01
#8	0.3672	268.6	18.74
#9	0.7752	634.8	15.07
#10	0.6392	24.41	20.23
#11	0.9112	537.1	14.13

Table 15. N=128

Speaker	Time slot (s)	Frequency (Hz)	Periodogram amplitude (dB)
#1	0.35362	390.6	7.65
#2	0.5947	878.9	6.099
#3	0.3595	585.9	10.04
#4	0.4872	390.6	8.283
#5	0.5174	488.3	13.43
#6	0.5074	585.9	10.86
#7	0.5746	781.3	0.965
#8	0.3595	293	6.592
#9	0.793	683.6	3.726
#10	0.6518	97.66	16.02
#11	0.9307	585.9	6.733

VI. Reference

[1] S. Furui, “automatic speech recognition and its application to information extraction, “Proceedings of the 37th annual meeting of the ACM on computational linguistic, pp 11-20, June 1999.

[2] EEC201 Final project description. https://canvas.ucdavis.edu/courses/409075/files/folder/Final%20Project?preview=7801674

VII. Appendix – how to do the demo

1. **Evaluation with the testing samples provided by Prof. Ding : run main_script.m**
2. **Evaluation with the noise corrupted testing samples: run main_script_gaussian.m**
3. **Evaluation with the notch filtered testing samples: run main_script_notch.m**
4. **Evaluation based on Gengchen’s training and testing samples: run test_gengchen_voice.m**

The gaussian test might take a while (~3-5 mins).