EECE 3841 Speaker Detection Final Project

Faris Jugovic, Kristopher Pesnell, Jose De Los Santos
fjugovic@mail.lipscomb.edu krpesnell@mail.lipscomb.edu
jose.e.delossantos90@gmail.com

April 23, 2025

1. Introduction

i. **Overview**
This project aimed to design a speaker detection algorithm using MATLAB that classifies whether a recorded voice comes from a nominal (registered) user or a non-nominal (imposter). Using signal processing tools such as the Fast Fourier Transform (FFT), the system converts audio waveforms into frequency-domain signals that are used for speaker verification.

ii. **Method Summary**
Multiple voice recordings per user were collected and converted into frequency data using FFT. Three recordings per user were averaged to generate a profile. Comparisons between test samples and profiles were made using Root Mean Square Error (RMSE). A recording was classified as a match if the error was below a predetermined threshold.

iii. **Results Summary**
After tuning the match threshold, a value of 0.005 was selected. With this value, the system successfully identified all nominal users and rejected all imposters.

iv. **Significance**
The project demonstrates that a basic signal processing approach can achieve high performance in voice identification. This makes the system suitable for lightweight and efficient applications.

2. Methodology

i. **Data Collection Approach**
Recordings were gathered from 14 participants, including 10 nominal users and 4 imposters. Each participant provided 5 recordings:

- Recordings 1, 2, and 3 were used to generate a voice profile for each nominal user.
- Recordings 4 was used for calibration. This was done by tweaking the match threshold which is the output of the RMSE.
- For imposters, either recording 5 was selected for testing against all nominal user profiles.

Recordings were made using identical hardware under consistent conditions. The spoken sentence contained a variety of phonemes and diphthongs to help with distinction.

ii. **Input Processing**
Each audio file was first loaded using the `audioread` function. After loading, the time-domain waveform was transformed into the frequency domain using MATLAB's `fft()` function. Since the FFT output is symmetric for real-valued signals, only the first half of the spectrum was retained. The resulting frequency magnitudes were then normalized to values under 1. All recordings were either truncated or zero-padded to ensure a consistent FFT length, which allowed accurate element-wise comparison between the test signal and user profile. All three signals are averaged together resulting in one signal representing the users profile. To smooth the signal for the RMSE, a 1000-point moving average filter was applied. This profile represents the expected frequency signature for a given speaker.

iii. **Speaker Detection Algorithm**
After computing the user profile a test recording is then processed in the same way as the user profile and is compared to the reference profile using Root Mean Square Error (RMSE). This error quantifies how similar the test spectrum is to the profile. If the RMSE is below a predefined threshold (set to 0.005 in final testing), the system considers the test recording a match. Otherwise, the result is classified as a non-match.

iv. **High Level Pseudocode**

```
For each user:
    avg_profile = average(FFT(file1), FFT(file2), FFT(file3))
    For each test_file in [file4, file5, imposters]:
        processed_test = normalize(smooth(FFT(test_file)))
        score = RMSE(processed_test, avg_profile)

        if score < threshold:
            match = true
        else:
            match = false
```

3. Results

i. **Training Process**
The training process was performed in the algorithim. Only 3 audio files were used in the creation of the user profile. Using more audio files would result in a more robust user profile.

ii. **Validation Process**
To perform validation, one sample from each user was compared against all user profiles. From there, the parameters were tweaked until all test cases passed. During validation, it was observed that thresholds under 0.003 rejected too many valid users, while thresholds well above 0.005 risked false positives. A value of 0.005 provided an ideal balance.

iii. **Test Process**
In the final evaluation using 14 test samples:

- True Positives (TP): 10
- False Negatives (FN): 0
- False Positives (FP): 0
- True Negatives (TN): 40

Performance Metrics Explanation:
Precision measures how many of the recordings classified as matches were actually correct, calculated as $\frac{TP}{TP+FP}$. Recall reflects how many of the actual nominal users were correctly identified, calculated as $\frac{TP}{TP+FN}$.

Final Performance:

- Precision: 100%
- Recall: 100%

Visual Comparison:

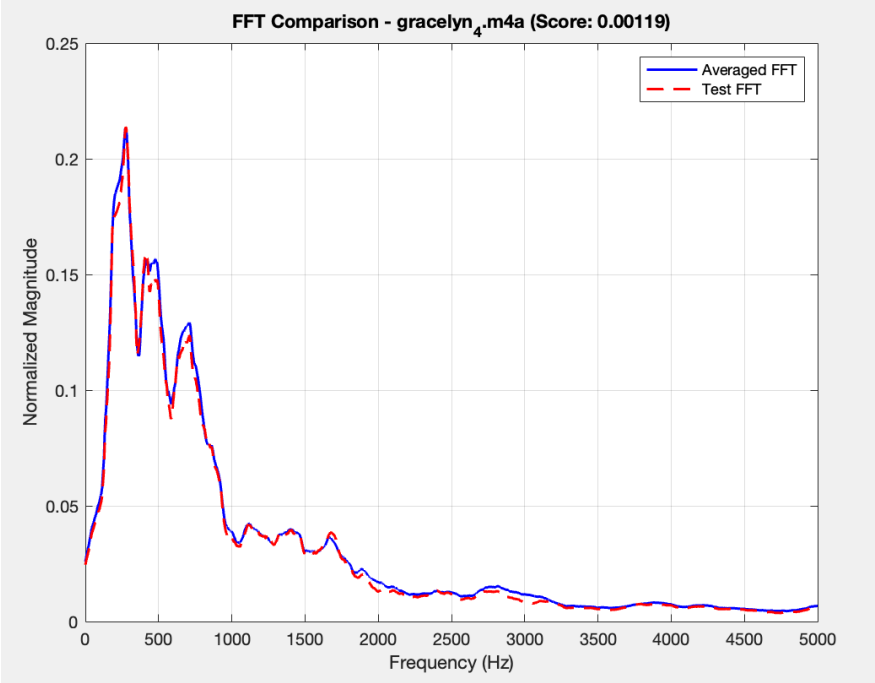

Figure 1: Gracelyn – Match: RMSE = 0.00119 (Accepted)

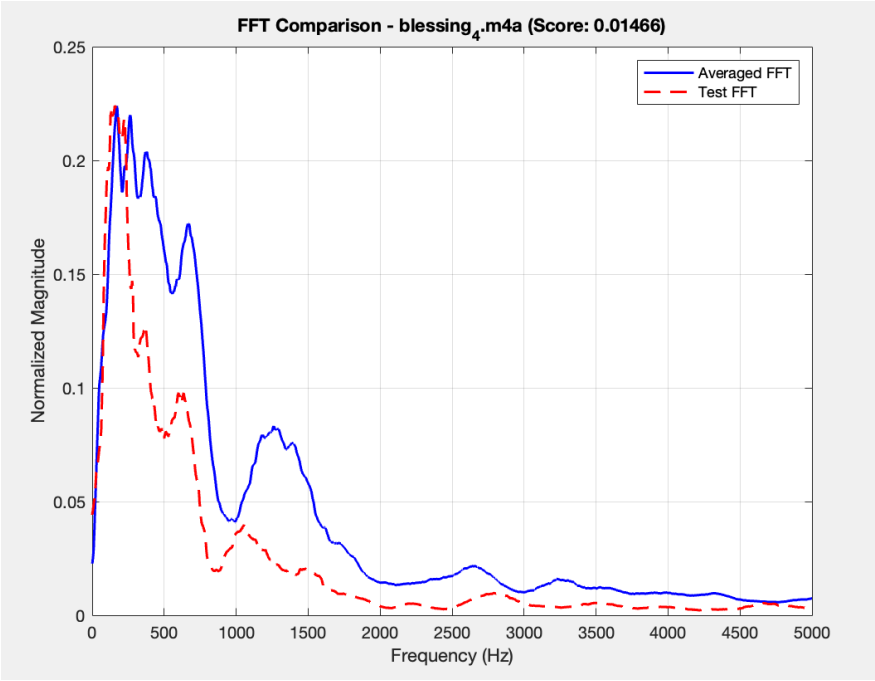

Figure 2: Blessing – Non-match: RMSE = 0.01466 (Rejected)

These plots help show how the system tells the difference between recordings from the same person and those from someone else. In the first plot, the test recording from Gracelyn is very close to the average FFT profile made from her earlier recordings. The two lines follow each other closely, resulting in a low RMSE score of 0.00119, which is below the match threshold and is correctly accepted.

The second plot shows a test from Blessing, who is an imposter. Her recording looks very different from the averaged profile of the user she was tested against. Because of this difference, the RMSE score is much higher at 0.01466, which is above the threshold, so the test correctly fails.

These examples show that RMSE is a useful way to measure how similar two voice recordings are, helping the system correctly identify users and reject imposters.

Runtime Analysis
The total execution time for the complete script was approximately 10.27 seconds. The average runtime per call to `test_user.m` was approximately 0.21 seconds, with consistent times across both self-tests and imposter tests. Within this function, the call to `plot_avg_fft_of_three.m` required approximately 0.17 to 0.20 seconds per user profile. The `compare_fft_to_average.m` function executed very quickly in comparison, averaging 0.05 to 0.06 seconds per call. These results confirm that the system can operate efficiently.

4. Discussion

The speaker detection system demonstrated several key strengths. The use of Fast Fourier Transform (FFT) and Root Mean Square Error (RMSE) enabled accurate and consistent comparisons between voice samples. The final testing results confirmed high classification accuracy, with perfect precision and recall across both nominal users and imposters.

Despite these strengths, the system exhibits some weaknesses. Its performance can be sensitive to background noise, microphone sensitivity, or inconsistent speaking style across recordings. These factors could potentially distort the spectral characteristics that the system relies on for accurate matching.

Various parameters played a critical role in system performance. For example, match threshold tuning had a significant impact on the sensitivity of a match. A threshold set too low would result in false rejections, while a higher threshold could allow imposters to be incorrectly accepted. The use of smoothing, through a 1000-point moving average, was also important in stability and reducing the influence of spectral noise.

A significant lesson learned from this project was the importance of pre-processing and parameter tuning in this project. Furthermore, the project emphasized the effectiveness of signal processing fundamentals when applied in modern applications such as voice detection.

5. Repository

GitHub Repository: `https://github.com/JugovicFaris/DSP-Lab-Final`