Smartphone Sensing 2023
App 1, Option 2
group 9
Samsung Galaxy A13, Android 13

1 Load balance

1.1 Tony, 5651794

- Android Method
  - Chirp signal omitting
  - Cross-correlation on chirp signal and recorded audio
  - Extract 100ms from 500ms audio recording
- Off-line Method
  - FT on audio files and plot spectrum in python.
- Tasks and Deliverable
  - Full and processed spectrum of gathered data in C1-C4

1.2 Giacomo Mazzola, 4872673

- Android Method
  - Audio recording and storing to file
  - Generation of probe and chirp signal
- Off-line Method
  - FT on audio files and plot PSD and MFCC in python.
- Tasks and Deliverable
  - Record and store audio

2 Methods

2.1 Audio

This section records and extracts audio, including starting and stopping functions, time detection for signal omission, and extraction of echo-only audio.

2.1.1 Recording To record and save audio on Android, the app must first request `RECORD_AUDIO` permissions from the user. After granting permission, the app can use the `AudioRecord` Java class with parameters including a sample rate of `44100 Hz` to reach 20kHz, audio format of `ENCODING_PCM_16BIT`, channel configuration of `CHANNEL_IN_MONO`, and a buffer size of 2∗`MIN_BUFFER_SIZE` obtained through the `getMinBufferSize()` function.

2.1.2 Recording extracting To extract audio, cross-correlation is used to determine the time when the signal is omitted, and 100ms of audio is extracted starting from the end of the omission. An array of size equal to the maximum time lag[1] is created. The chirp signal is then slid over the audio signal and the similarity between the two signals is calculated and stored in the array. The highest value in the array indicates the time lag at which the two signals are most similar, corresponding to the time when the chirp signal is omitted. Using this result, 100ms of audio after the omission is extracted from the original audio file. Notably, each millisecond corresponds to `2 bytes` of data in the audio file due to the recorder's `CHANNEL_IN_MONO` setting. Results indicate that using cross-correlation to determine the time of omission of signals are very precise.

2.2 Signals

Two types of signals are used in this project: a steady probe signal at a given frequency, and a chirp signal that varies in frequency over time. Both signals have been implemented, and their performance will be evaluated in assignment 2.

2.2.1 Probe The probe signal is generated at `20kHz` for a duration of `2ms` to make it inaudible to humans. The signal is sampled at a frequency of `44100 Hz`.

2.2.2 Chirp The chirp signal is also generated for a duration of `2ms` starting at `15kHz` and ending at `20kHz` to make it inaudible to humans, with a sampling rate of `44100 Hz`.

2.3 Feature extraction

At present, feature extraction is performed offline in Python, but for the next assignment, we plan to implement it online on the Android device.

2.3.1 Spectrum The spectrum of an audio recording is a graph that shows the power of the signal distributed in the frequency spectrum (in our case 15-20kHz) as a function of time. We generate the spectrum from the .pcm file of the audio recording using the libraries `numpy` and `matplotlib.pyplot`. FT is performed within the function `specgram()`.

2.3.2 PSD An alternative method to extract features is to compute the Power Spectral Density (PSD), which illustrates how the power of the signal is distributed

[1] Length of the audio signal minus the length of the chirp signal

across the spectrum. The PSD can be computed using Welch's method with a window size of 1024 samples.

2.3.3 MFCC Mel-Frequency Cepstral Coefficients (MFCC) is a widely used feature extraction technique in speech and audio signal processing. Although we were able to extract features from the signal using this technique, we did not perform a thorough analysis to tune its parameters. This technique will not be the main one used, but it is a candidate to be chosen to increase the novelty of our project.

3 Evaluation setup

We evaluated our app in the environment consisting of Cells C1 to C4, as defined in the provided map of Building 28. We conducted five recordings for each location, facing towards the tag of that location, while holding the phone vertically in one hand, just like a typical user would. We limited ourselves to five recordings for each tag at this stage because our focus was on implementing effective feature extraction and analyzing different parameters. In a subsequent stage, we plan to conduct additional measurements, potentially holding the phone in different orientations, to better train the model.

4 Results

At this stage of the project, we have two results to share: the feature extraction results and the GUI.

4.1 Feature extraction of cells

The primary feature extraction technique that we implemented is the spectrum of the echo. Figure 1 displays the complete spectrum of the recorded audio, while Figure 2 displays the processed spectrum that only includes the echo of the chirp. **Due to limited space, result of C4 is displayed as an example. Data are gathered in all four cells.**

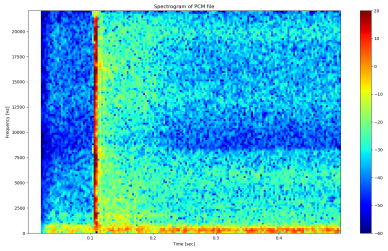

Figure 1: Complete spectrum in spot C4

Unfortunately, due to space limitations, this report does not include any visual illustrations of the PSD and the MFCC. Nonetheless, we will examine them and consider incorporating them in the final report if they will be used for training purposes.

4.2 GUI

A screenshot of the GUI can be seen in Figure 3. It is still fairly simple as it will be improved in the future.

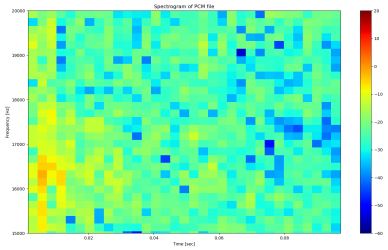

Figure 2: Processed spectrum of only echo in spot C4

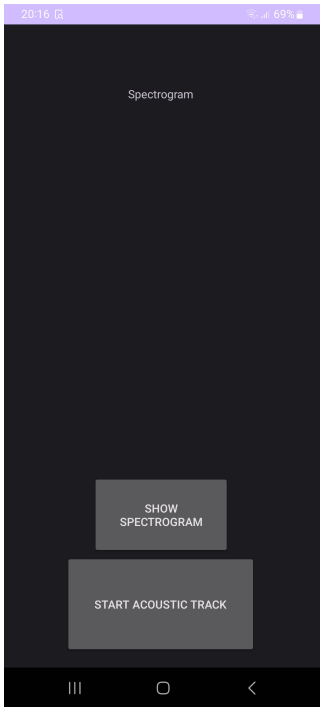

Figure 3: Screenshot if the GUI

5 Discussion

For learning purposes, we developed two versions of the app, one for the Samsung Galaxy A13 and another for the Huawei Mate 20 Pro. We observed that the behavior of the apps differed depending on the phone they were running on. The most significant difference was the delay introduced by the microphone when it started recording. The Samsung had a delay of 20-30ms, while the Huawei had a delay of almost 400ms. This presents an opportunity to innovate our approach by investigating the reasons for these differences and designing an app that can dynamically and autonomously account for these variations, rather than having to develop multiple apps for different architectures.

It is also worth noting that the microphone recorded two spikes present throughout the entire spectrum at the beginning and end of the chirp. We attribute this to the imperfections of the speakers, which vibrate in an uncontrolled manner when turning on and off.