Real-Time Implementation and Evaluation of Wide Dynamic Range Compression and Noise Reduction

https://github.com/ilucasgoncalves/real-timeNoiseReductionApp

December 20, 2021

Contents

1 Introduction

This document contains detailed information about the implementation of an Android smartphone app that implements functions used in hearing aids consisting of wide dynamic range compression (WDRC) and noise reduction (NR). In this document, the results from the app are presented and contrasted using audios captured in real-world environments. This report also contains an user's guide which explains how the Android app is operated and explains in detail the steps taken to complete the app and its algorithm components.

This report contains four sections: first we have a section for the introduction, the second section presents experimental evaluation obtained from the app in real-world environments and discussion, third we will do a deep dive into the Android app GUI and its functionalities, and lastly we explain in depth the contents of the project shell and its locations.

2 The App and Experiments

In this section we will talk about the steps taken to complete the app in real-time on an Android device and presents results and discussion based on real-world scenarios we explored while experimenting with the app.

2.1 Implementing the App

The Android application was designed and implemented using the flowchart shown in Figure 1. The application was designed to process audio in real-time with a sampling frequency of 48000 Hz and where each input frame has 256 samples. As seen in the flowchart. Each input frame is processed by the VAD and SPL functions, which measure the probability of speech presence and SPL for each frame. After retrieving VAD and SPL values, the algorithm uses the values to update the WDRC flag and NR flag, which are used to selected the most suitable filter for that specific frame. The filter used in this app comes from a FIR filter bank consisting of 6 FIR filters that was specifically designed ofr gain adjustments in different frequency bands. The filter value obtained for each frame is the stored in an array that will be capturing filter values until the decision rate is met. The decision is defined to be performed every few frames with 50% overlap (e.g. is the decision rate is 200 frames we set a step size of 100 frames for decisions). Once the decision rate is met, a majority voting mechanism is used to decide what filter to apply on the following frames based on the filters computed for previous frames processed. The filter chosen is then used to filter the upcoming frames until

Table 1: Different real-world audio environments we collected audio in for experimental evaluation corresponding to the six cases of interest (3 audios for each scenario).

Filter 1	Filter 2	Filter 3	Filter 4	Filter 5	Filter 6
(WDRC, NR) (Soft, Off)	(WDRC, NR) (Soft, On)	(WDRC, NR) (Moderate, Off)	(WDRC, NR) (Moderate, On)	(WDRC, NR) (Loud, Off)	(WDRC, NR) (Loud, On)
1. Dance Hall	1. Parking Lot	1. Study Room	1. Gym Locker	1. Study Room	1. Car radio
2. House Env.	2. Raining	2. Open Air	2. Gym	2. House Env.	2. Parking Lot
3. Study Room	3. Gym	3. House Env.	3. Inside Car	3. Open Air	3. Open Air

decision rate is met again. This process it performed many times until the audio is fully processed. During development we experimented with 6 audio files that represented each one of the environments where each of the 6 filter would be needed. For further detailed explanation and deep dive into the algorithm refer to section 4.

2.2 Experimental Evaluation

We conducted experimental evaluation on the app using captured real-world audio files that corresponds to various scenarios a user might experience. The scenarios were chosen to match situations where each filter would need to be applied. We collected a total of eighteen 30-second audio files in different real-world audio environments corresponding to the six cases of interest (3 audios for each scenario) as seen in Table 1. During the audio collection the same sentence was uttered for a fair evaluation across all scenarios. The sentence used is a popular English pangram containing all the letters in the English alphabet "The quick brown fox jumps over the lazy dog. The dog wakes up and follows the fox in the forest. But again the quick brown for jumps over the lazy dog."

Using the real-world recorded files, we performed evaluations using different app setting as well. The setting explored were: different decision rates (50, 100, 200, 300, and 500 frames) and different thresholds for VAD and SPL flags updates (0.55, 0.65, 0.75, 0.85, and 0.95).

2.2.1 Real-world Captured Audios Evaluation

The first evaluations performed using the app was on the real-world audios using the standard settings given for the app. In the standard setting first

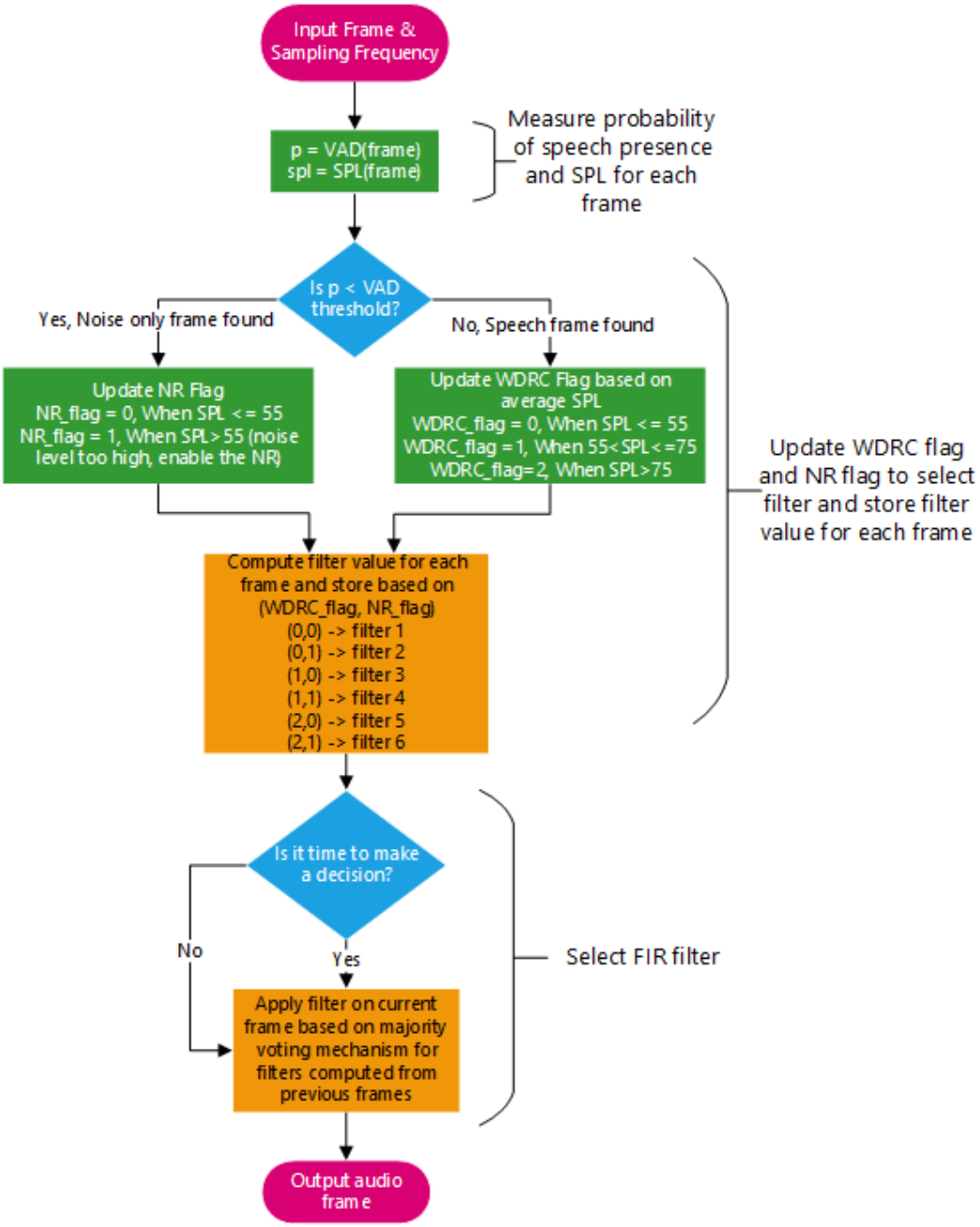


Figure 1: App flowchart used for implementation.

used in the app development, the decision rate is set to 200 frames and the threshold is set to 0.75. The experiments here consists of verifying the application is performing the filtering and applying the correct filter in different situations.

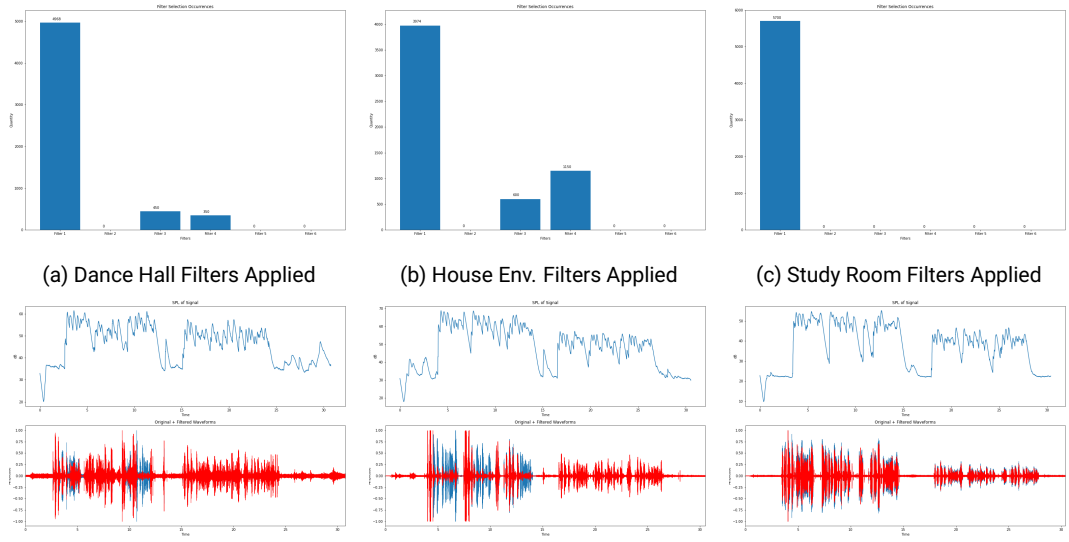
Figure 2, shows the results for the three audios generated for Filter 1 scenarios where WDRC is soft and NR is off. Looking at the figure we see that figures a-c shows that filter 1 was the one applied the most across the audios, matching our expectations. Also looking at the overlap figures for the original audios and filtered audios, we can see what the filters are doing to the input signals. Following analysis for the other scenarios are similar to this the analysis for F1 filter scenarios. The second scenarios are shown in Figure 3, here we see that the filter being mostly selected does not match our expectations. Instead of 2, we receive filter 4 the most, which is unexpected, but after listening to the audios they seem to sound better than original so filtering is working well here too. Scenario results for filter 3 and 4 are shown in Figure 4 and Figure 5, respectively. Similar results to F1 and F2 scenarios were obtained, we see a lot of different filters selected for scenario F3, but in F4 the filters seem consistent across all audios. Lastly, the analysis for F5 and F6 scenarios, are similar to previous scenarios as well, Figure 6 shows results for scenarios F5 and Figure 7 shows results for scenario F6.

The results here shows, that the application is working properly, but it can be improved by experimenting with different settings. In the next sections we experiment with decision rates and thresholds.

2.2.2 Decision Rates Evaluation

In this section, we experiment with different decision rates to find one that suits best our app's purpose. Decision rates can affect the app's reaction time to different scenarios and having a optimal decision rate is important so the filtering capabilities of the application is taken great advantage of. The decision rates analyzed in the experimental evaluation are: 50 frames, 100 frames, 200 frames, 300 frames, and 500 frames. Decision rates used all have a 50% overlap for decision. The results shown here compares one audio for each scenario collected, processed using all different decision rates, refer to Figures 8, 9, and 10 for side-by-side comparisons.

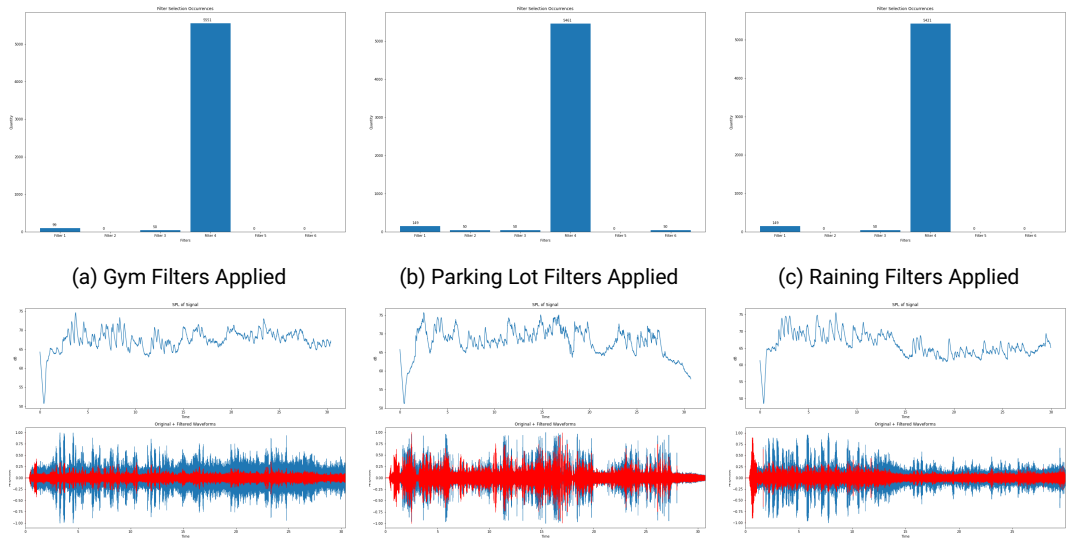
Across all experiments performed in this section we can see that when using high decision rates, such as, 300 or 500 frames the algorithm takes too long to select the appropriate filter leading to major distortions in the beginning and through the audio signal. Figure 9 d, e, i, and j all are good examples of this phenom happening. Having to wait 500 or 300 frame steps to make a decision means that 2.6 or 1.6 seconds have to go by to make an



(a) Dance Hall Filters Applied (b) House Env. Filters Applied (c) Study Room Filters Applied

(d) Dance Hall Resulting Audio and SPL (e) House Env. Resulting Audio and SPL (f) Study Room Resulting Audio and SPL

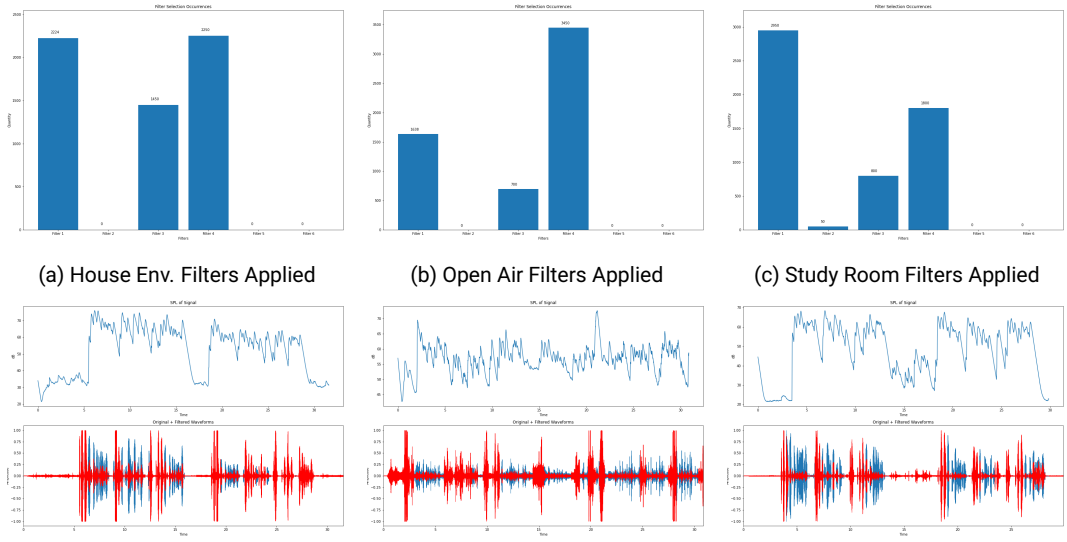
Figure 2: Figures a-c show the majority of filters applied during audio processing. Figures d-e shows the SPL values computed for each frame in time and the overlapping plot of the original signal in blue and resulting signal in red.



(a) Gym Filters Applied (b) Parking Lot Filters Applied (c) Raining Filters Applied

(d) Gym Resulting Audio and SPL (e) Parking Lot Resulting Audio and SPL (f) Raining Resulting Audio and SPL

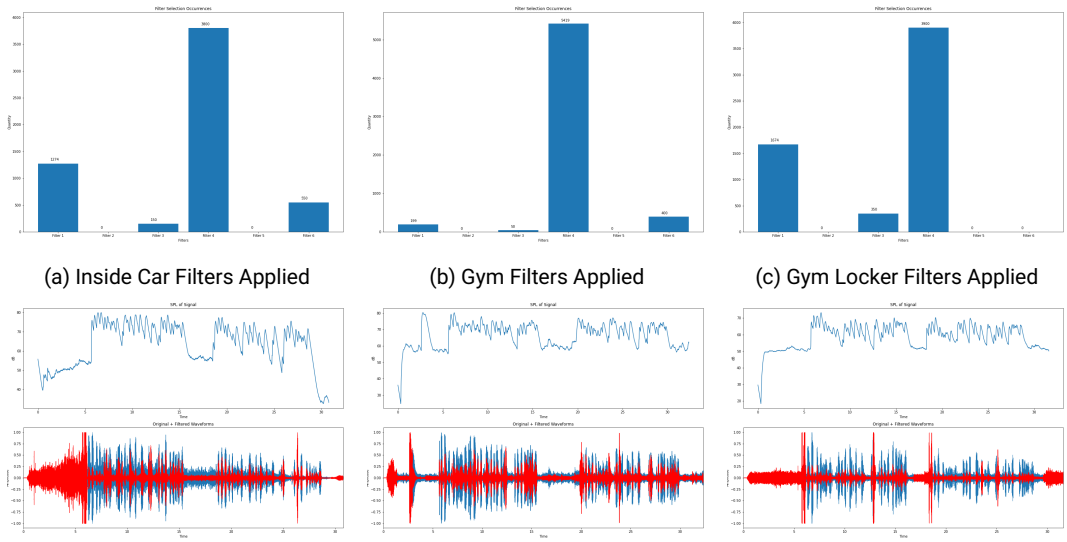
Figure 3: Figures a-c show the majority of filters applied during audio processing. Figures d-e shows the SPL values computed for each frame in time and the overlapping plot of the original signal in blue and resulting signal in red.



(a) House Env. Filters Applied (b) Open Air Filters Applied (c) Study Room Filters Applied

(d) House Env. Resulting Audio and SPL (e) Open Air Resulting Audio and SPL (f) Study Room Resulting Audio and SPL

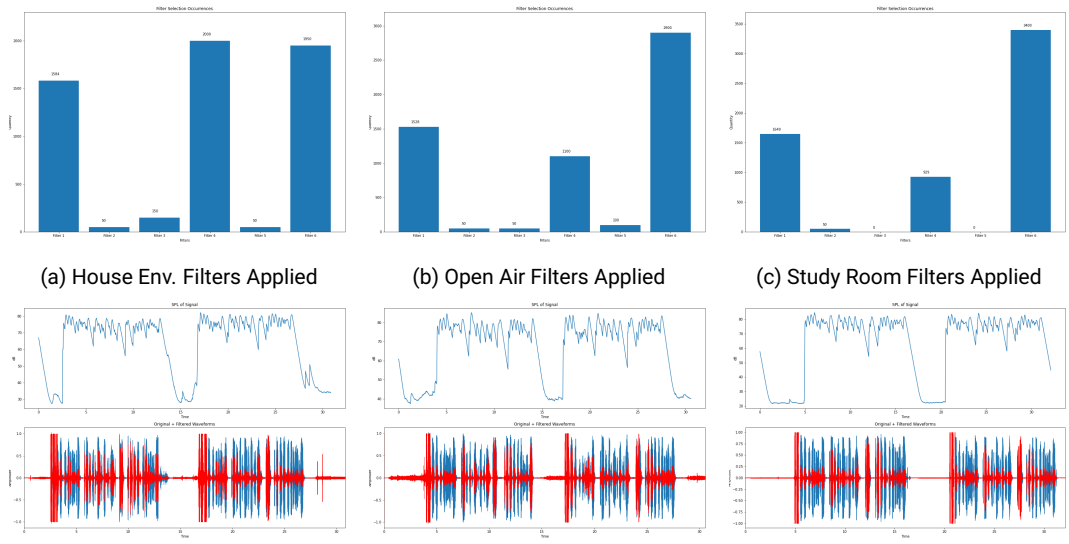
Figure 4: Figures a-c show the majority of filters applied during audio processing. Figures d-e shows the SPL values computed for each frame in time and the overlapping plot of the original signal in blue and resulting signal in red.



(a) Inside Car Filters Applied (b) Gym Filters Applied (c) Gym Locker Filters Applied

(d) Inside Car Resulting Audio and SPL (e) Gym Resulting Audio and SPL (f) Gym Locker Resulting Audio and SPL

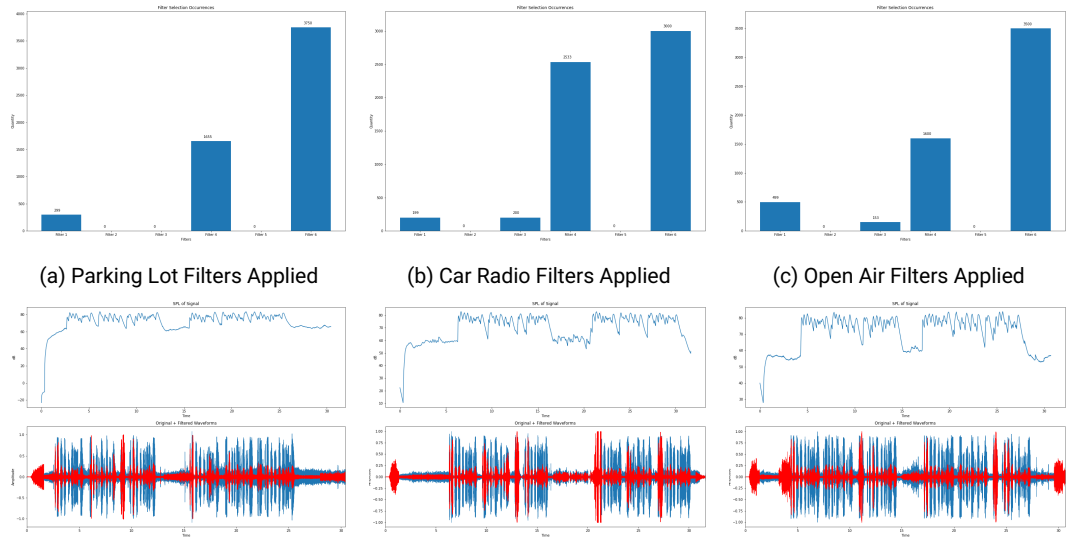
Figure 5: Figures a-c show the majority of filters applied during audio processing. Figures d-e shows the SPL values computed for each frame in time and the overlapping plot of the original signal in blue and resulting signal in red.



(a) House Env. Filters Applied (b) Open Air Filters Applied (c) Study Room Filters Applied

(d) House Env. Resulting Audio and SPL (e) Open Air Resulting Audio and SPL (f) Study Room Resulting Audio and SPL

Figure 6: Figures a-c show the majority of filters applied during audio processing. Figures d-e shows the SPL values computed for each frame in time and the overlapping plot of the original signal in blue and resulting signal in red.



(a) Parking Lot Filters Applied (b) Car Radio Filters Applied (c) Open Air Filters Applied

(d) Parking Lot Resulting Audio and SPL (e) Car Radio Resulting Audio and SPL (f) Open Air Resulting Audio and SPL

Figure 7: Figures a-c show the majority of filters applied during audio processing. Figures d-e shows the SPL values computed for each frame in time and the overlapping plot of the original signal in blue and resulting signal in red.

initial decision, which causes the first decision to not be very accurate for the input being processed sequentially. Furthermore, a very low decision rate of 50 is not very optimal either, since too many changes will happen during the processing which can lead to distortions. Having too many decisions and changes in the filter can be bad in audio, for example where there are several pauses and the speech probability is constantly oscillating, such as the case seen in Figure 9 a and f.

Lastly, amongst the last two decision rates we explored. We conducted both visual and hearing analysis to determine which decision rates is better: 200 or 100. We noticed that setting the decision rate to 100 works better for our purposes. When decision rate is set to 100 the first decision happens 0.5 seconds in and then .25 seconds subsequently. This allows enough time and frames to capture the audio properties to make a filter decision and it also is not a very short decision time that would make our algorithm oscillate between filters constantly.

2.2.3 Thresholds Evaluation

In this section, we experiment with different threshold values. The threshold value is what determines how the NR flag and the WDRC flag will be set. The flags are then used to directly decide with filter the algorithm should apply to the input audio. The thresholds we experiment with are: 0.55, 0.65, 0.75, 0.85, and 0.95. The analysis performed in this section uses a decision rate set at 100, which is the best decision rate as established in the previous section. All audio recorded in different real-world environment are processed and compared using audio wave plots, spectrograms, and listening to the app's outputs.

The first analysis was conducted on the audios provided with the project shell for each scenario. Table 2 shows the amount of times each filter was applied to each frame of the input audios for each threshold explored. We see that in table 2 the thresholds that are more successful in selecting the expected filters for the audio are thresholds 0.55 and 0.65. In some cases, such as when WDRC is soft and noise reduction is on, threshold 0.85 seems to work better, but when listening to the output audios we did not notice a big difference between using 0.55-0.65 or 0.85. Furthermore, when WDRC is loud and noise reduction is off, threshold 0.95 seems to work slightly better than the other thresholds, but is it not a significant amount. The resulting waveforms and spectrograms shows a better insight on the resulting audios and original audios side-by-side as seen in Figures 11, 12, 13, 14, 15, and 16. The algorithm successfully filter the noise and adjusts the speech signals as seen in Figure 14 and Figure 16. We also see a great performance when no

Table 2: Testing threshold values in each one of the proposed scenarios and verifying the amount of times each filter gets applied to each frame of the audio signal.

	(WDRC, NR) (Soft, Off)	(WDRC, NR) (Soft, On)	(WDRC, NR) (Moderate, Off)	(WDRC, NR) (Moderate, On)	(WDRC, NR) (Loud, Off)	(WDRC, NR) (Loud, On)
Thres.	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)
0.55	**2487, 150, 650, 0, 0, 0**	**1637, 700, 650, 300, 0, 0**	649, 550, 400, 1150, 63, 100	99, 0, 200, 2788, 0, 200	99, 150, 0, 700, 200, 4476	**99, 0, 50, 1000, 0, 1763**
0.65	2337, 0, 800, 150, 0, 0	2087, 100, 900, 200, 0, 0	799, 0, 1000, 950, 163, 0	**99, 0, 50, 2888, 0, 250**	249, 0, 0, 650, 200, 4526	**99, 0, 50, 1000, 0, 1763**
0.75	1849, 0, 1188, 250, 0, 0	1849, 100, 1038, 300, 0, 0	499, 100, 1200, 950, 163, 0.	99, 0, 50, 2738, 0, 400	249, 0, 0, 550, 100, 4726	**99, 0, 50, 1000, 0, 1763**
0.85	1949, 0, 1088, 250, 0, 0	1649, 100, 1238, 300, 0, 0	**349, 100, 1300, 950, 213, 0**	99, 0, 50, 2738, 0, 400	149, 0, 100, 550, 100, 4726	**99, 0, 50, 1000, 0, 1763**
0.95	1549, 0, 1488, 250, 0, 0	1399, 100, 1388, 400, 0, 0	362, 150, 1250, 1000, 150, 0	99, 0, 50, 2488, 0, 650	**99, 0, 0, 550, 250, 4726**	**99, 0, 50, 1000, 0, 1763**

Table 3: All experiments conducted with the 18 collected audios to determine the best threshold (3 audios for each scenario).

	Filter 1	Filter 2	Filter 3	Filter 4	Filter 5	Filter 6
	(WDRC, NR) (Soft, Off)	(WDRC, NR) (Soft, On)	(WDRC, NR) (Moderate, Off)	(WDRC, NR) (Moderate, On)	(WDRC, NR) (Loud, Off)	(WDRC, NR) (Loud, On)
	1. Dance Hall 2. House Env. 3. Study Room	1. Parking Lot 2. Raining 3. Gym	1. Study Room 2. Open Air 3. House Env.	1. Gym Locker 2. Gym 3. Inside Car	1. Study Room 2. House Env. 3. Open Air	1. Car radio 2. Parking Lot 3. Open Air
Thres.	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)
0.55	1. **(5018, 0, 750, 0, 0, 0)** 2. (3924, 100, 250, 1450, 0, 0) 3. **(5700, 0, 0, 0, 0, 0)**	1. (199, 0, 350, 5211, 0, 0) 2. (149, 0, 50, 5421, 0, 0) 3. (99, 0, 250, 5351, 0, 0)	1. **(2400, 600, 1250, 1350, 0, 0)** 2. **(1588, 100, 3950, 150, 0, 0)** 3. **(2324, 150, 2100, 1350, 0, 0)**	1. (199, 0, 350, 5219, 150, 150) 2. (199, 0, 350, 5219, 150, 150) 3. **(1374, 50, 0, 3800, 250, 300)**	1. **(1549, 150, 0, 1029, 250, 3050)** 2. (499, 1135, 100, 2050, 150, 1850) 3. **(1478, 150, 200, 1000, 400, 2500)**	1. (249, 0, 100, 2583, 0, 3000) 2. (299, 0, 150, 2105, 0, 3150) 3. (199, 400, 100, 1753, 0, 3300)
0.65	1. (4568, 450, 600, 150, 0, 0) 2. (3924, 0, 700, 1100, 0, 0) 3. **(5700, 0, 0, 0, 0, 0)**	1. (199, 0, 50, 5461, 0, 50) 2. (149, 0, 50, 5421, 0, 0) 3. (99, 0, 200, 5401, 0, 0)	1. (2800, 100, 1050, 1650, 0, 0) 2. (1638, 0, 1200, 2950, 0, 0) 3. (2374, 0, 1450, 2100, 0, 0)	1. **(199, 0, 50, 5519, 0, 300)** 2. **(199, 0, 50, 5519, 0, 300)** 3. (1374, 0, 150, 3700, 0, 550)	1. (1649, 50, 0, 929, 0, 3400) 2. **(1584, 50, 100, 2050, 200, 1800)** 3. (1578, 50, 0, 1150, 150, 2800)	1. (199, 0, 250, 2483, 0, 3000) 2. (299, 0, 0, 1905, 0, 3500) 3. (599, 0, 253, 1500, 0, 3400)
0.75	1. (4968, 0, 450, 350, 0, 0) 2. **(3974, 0, 600, 1150, 0, 0)** 3. **(5700, 0, 0, 0, 0, 0)**	1. **(149, 50, 50, 5461, 0, 50)** 2. (149, 0, 50, 5421, 0, 0) 3. (99, 0, 50, 5551, 0, 0)	1. (2950, 50, 800, 1800, 0, 0) 2. (1638, 0, 700, 3450, 0, 0) 3. (2224, 0, 1450, 2250, 0, 0)	1. (199, 0, 50, 5419, 0, 400) 2. (199, 0, 50, 5419, 0, 400) 3. **(1274, 0, 150, 3800, 0, 550)**	1. (1649, 50, 0, 929, 0, 3400) 2. (1584, 50, 150, 2000, 50, 1950) 3. (1528, 50, 50, 1100, 100, 2900)	1. (199, 0, 200, 2533, 0, 3000) 2. (299, 0, 0, 1655, 0, 3750) 3. (499, 0, 153, 1600, 0, 3500)
0.85	1. (4968, 0, 300, 500, 0, 0) 2. **(3974, 0, 500, 1250, 0, 0)** 3. **(5700, 0, 0, 0, 0, 0)**	1. **(149, 50, 50, 5461, 0, 50)** 2. (149, 0, 50, 5421, 0, 0) 3. (99, 0, 50, 5551, 0, 0)	1. (2699, 50, 901, 1950, 0, 0) 2. (1388, 0, 400, 4000, 0, 0) 3. (2224, 0, 1250, 2450, 0, 0)	1. (199, 0, 50, 5419, 0, 400) 2. (199, 0, 50, 5419, 0, 400) 3. **(1274, 0, 150, 3800, 0, 550)**	1. (1699, 0, 0, 929, 0, 3400) 2. (1634, 0, 150, 1950, 50, 2000) 3. (1528, 0, 100, 950, 100, 3050)	1. (199, 0, 250, 2483, 0, 3000) 2. (299, 0, 0, 1505, 0, 3900) 3. (499, 0, 200, 1353, 0, 3700)
0.95	1. (4968, 0, 100, 700, 0, 0) 2. (3874, 0, 300, 1550, 0, 0) 3. **(5700, 0, 0, 0, 0, 0)**	1. (149, 50, 50, 5361, 0, 150) 2. (149, 0, 50, 5421, 0, 0) 3. (99, 0, 50, 5551, 0, 0)	1. (2399, 0, 901, 2300, 0, 0) 2. (1338, 150, 250, 4050, 0, 0) 3. (1849, 0, 1175, 2900, 0, 0)	1. (199, 0, 0, 5469, 0, 400) 2. (199, 0, 0, 5469, 0, 400) 3. (1149, 0, 325, 3750, 0, 550)	1. (1699, 0, 0, 929, 50, 3350) 2. (1634, 0, 150, 1900, 0, 2100) 3. (1299, 0, 0, 900, 329, 3200)	1. **(199, 0, 250, 2433, 0, 3050)** 2. **(299, 0, 0, 1455, 0, 3950)** 3. **(349, 0, 300, 1303, 0, 3800)**

noise is present but the speech is too loud and needs to be adjusted (Figure 15).

The same experiments were performed on each of the 18 real-word environment audios collected. Table 3 shows the amount of times each filter was applied for each frame or each one of the audio for different thresholds. Similar results are observed here, which shows that when thresholds are set to 0.55 or 0.65 the filters getting applied are what is expected for those scenarios more often.

2.2.4 Application Decision Algorithm Evaluation

In this section, we experiment with three versions of the app to get some insight on how the app performs if different decision methods are to be

Table 4: Testing different versions of the application in each one of the proposed scenarios and verifying the amount of times each filter gets applied to each frame of the audio signal.

	(WDRC, NR) (Soft, Off)	(WDRC, NR) (Soft, On)	(WDRC, NR) (Moderate, Off)	(WDRC, NR) (Moderate, On)	(WDRC, NR) (Loud, Off)	(WDRC, NR) (Loud, On)
Version	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)	(F1, F2, F3, F4, F5, F6)
V1	(2787, 0, 500, 0, 0, 0)	(1987, 100, 1200, 0, 0, 0)	(899, 300, 1413, 300, 0, 0)	(99, 0, 100, 2988, 0, 100)	(299, 0, 1326, 0, 4000, 0)	(99, 0, 1400, 0, 1413, 0)
V2	(2837, 0, 450, 0, 0, 0)	(1799, 100, 1338, 50, 0, 0)	(499, 500, 813, 1050, 0, 50)	(99, 0, 100, 2888, 0, 200)	(249, 0, 800, 0, 4576, 0)	(99, 0, 650, 0, 2000, 163)
V3	(2587, 50, 550, 100, 0, 0)	(2337, 0, 600, 350, 0, 0)	(1199, 0, 550, 1000, 163, 0)	(99, 0, 50, 2938, 0, 200)	(249, 0, 0, 700, 200, 4476)	(99, 0, 50, 1000, 0, 1763)

implemented. Three version of the app were generated using the settings we decided would be best from the previous experiments (Decision rate = 100 and Threshold = 0.60). The three versions are the following:

Version 1: We compute the average P and the average L for the processed frames to decide which filter to apply to the frames of a next decision buffer. In this mechanism, the filter update rate (decision rate) is equal to the size of the decision buffer.

Version 2: We compute the average P and the average L for the processed frames to decide which filter to apply to the frames of a next decision buffer. Then a 50% overlap is used by making a decision on one half of the frames in a next decision buffer based on all the frames in a previous decision buffer. In this case, the decision rate is equal to one half of the size of the decision buffer.

Version 3: Is the algorithm here, where we used a majority voting mechanism with 50% overlap to decide on the next filter.

Table 4, shows the results obtained from different scenarios where each one of the 6 filters would be applied. We see from this evaluation that there is not a version that ultimately works best for all situations. All versions work well and do a good job in applying the filters accordingly. In some cases like, (WDRC = moderate, NR = off) or (WDRC = loud, NR = off) version 1 and 2 seems to be working more as expected, now when WDRC = loud, NR = on version 3 seems to be working more as expected. In all other cases, we notice that all versions work similarly. Further experimental evaluations need to be conducted to determine the best version to be used.

2.3 Discussion

Based on the experimental evaluations performed in this study, we noticed better results from the app when the decision rate is set to 100 and threshold is set between 0.55 - 0.65. The final settings incorporated for the app is then the following: sampling frequency = 48000Hz, frame size = 256, decision rate = 100 frames with 50% overlap, and threshold = 0.60. The app performs

best under these settings; however, it is not perfect, it was observed for some scenarios in the experiments that different settings could work better in some cases. Also, this app can be improved in the future by implementing different adaptive filtering methods or exploring deep learning methods for this purpose as well.

3 Android App GUI - User's Guide

In this section we will talk about the Android app GUI and describe how to access its functionalities.

3.1 Main Screen

The main screen in our app is the starting point to access all of its functionalities and observe the outputs. Figure 17, show the main screen of our app. In this screen you can see the icons that gives you access to settings, reading files, and starting or stopping the algorithm from processing more audio frames in real-time scenarios.

3.2 Settings

If you tap the button that says "settings" on the main screen, you get access to all of the app settings as seen in Figure 18. Inside settings we can now make various changes to the app such as: enabling the output of audio to the phone's speaker, changing the output type to be playback to the users as seen in Figure 19, adjusting the sampling frequency as seen in Figure 20, changing the frame size to be processed at each step as seen in Figure 21, and lastly the app also enables user's to change its debugging level as seen in Figure 22. The debugging level settings allows user's to save text files with the outputs related to the VAD probabilities and SPL values the app computes for each frame processed, as well as the filter being applied. Once the app complete its running cycle and file is generated such as the one seen in Figure 23. Furthermore, the app also allow user's to save wave files outputs in PCM format.

3.3 Output Files

The app has two output types during the audio processing, the app allows users to visualize the waveform being generated during the audio processing

as seen in Figure 24, and the app shows users the current sate of the filter, VAD, and SPL values during audio processing as well as seen in Figure 25.

4 Code Organization - User's Guide

The Android project shell can be opened in Android Studio using the following steps:

+ Open Android Studio
+ Select "Open an existing project" under the files menu
+ Navigate to the folder "Project_Shell" and click on the android app icon
+ Hit open

After the project is opened, navigate to the file "local.properties" in the project. In the project, make sure that "sdk.dir" (Android SDK), "ndk.dir" (Android NDK) is pointing to the proper location in your machine. (Note: the ndk version used for this project is 21.0.6113669). Sync the project with your gradle files and the project is ready to be built.

4.1 Project Organization

The project's organization can be seen in Figure 26. Here you can see where the gradle local properties is located, where the MatlabNative.c file is located, as well as all the other files used in our project shell.

4.2 Functions Used

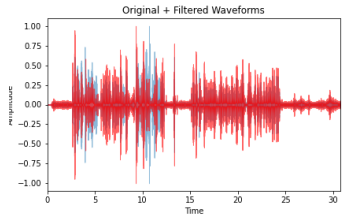
To accomplish the goal of our application. We have many functions that incorporate this app. The main functions used such as voice activity detector (VAD), sound pressure level (SPL), wide dynamic range compression (WDRC), and noise reduction (NR), and other auxiliary functions can be found in the location shown in Figure 27. These function were used in our MatlabNative.c file, which is the skeleton of the app and it combines the functions to make them work together.

4.3 MatlabNative.c

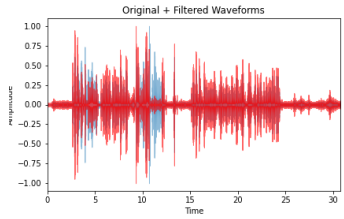
This file contains the main logic module of our application. We start the skeleton of our app by defining some variables we have to keep track of and the coefficients of the six different filters we will be using, as seen in Figure 28. MatlabNative.c contains everything needed to process each video frame,

store information, retrieve VAD probabilities and SPL values, raise flags, and determine what filter needs to be used based on input audios.

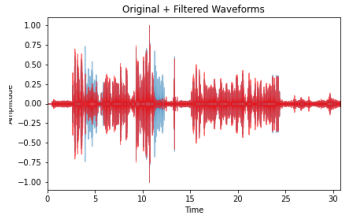
The filters to be used are decided based on a majority voting mechanism performed on the filter decisions from VAD and SPL values computed at each frame processed. The decision rate can be specified inside this file. Figure 29 shows how we compute the VAD and SLP values across the frames. As we compute the VAD probability and SPL value for each frame, we updated the flags based on these values and increase and iteration counter which keeps track of the number of iterations, seen in Figure 30. Following computation and flag updates, we retrieve and store filter values decided for each frames into a filter array, shown in Figure 31. At this step, we also check the iteration and frame counter values to make sure a 50% overlap is performed correctly. All of these steps are performed at the frame level and after each iteration, we also increment a overlap counter and a frame counter to keep track of when to make a decision. When the decision rate is met, as seen in Figure 32, we use majority voting to decide what filter to apply to the input signal. After completing the filter updating, VAD and SPL values computation, and flags updating we pass the input signal to and input buffer and filter it with the filter based on the decision made earlier. The filtered signal is passed on to a output array to be saved/played, as seen in Figure 33. Lastly, every time the app finishes running the flags, counters, and filters (Figure 34).



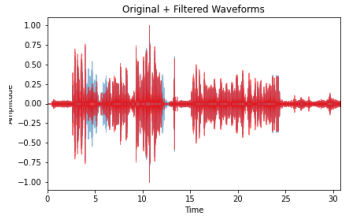
(a) Dance Hall Audio Filtered (Decision Rate = 50 frames).



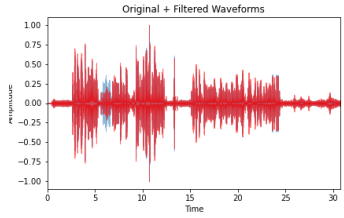
(b) Dance Hall Audio Filtered (Decision Rate = 100 frames).



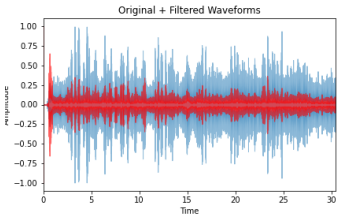
(c) Dance Hall Audio Filtered (Decision Rate = 200 frames).



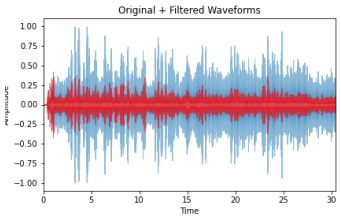
(d) Dance Hall Audio Filtered (Decision Rate = 300 frames).



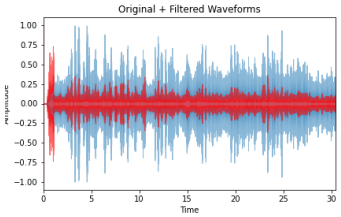
(e) Dance Hall Audio Filtered (Decision Rate = 500 frames).



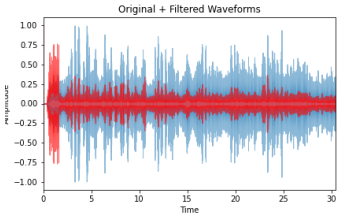
(f) Gym Audio Filtered (Decision Rate = 50 frames).



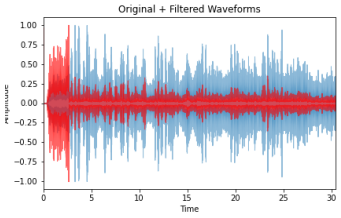
(g) Gym Audio Filtered (Decision Rate = 100 frames).



(h) Gym Audio Filtered (Decision Rate = 200 frames).

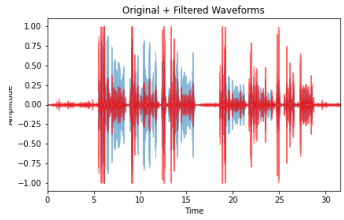


(i) Gym Audio Filtered (Decision Rate = 300 frames).

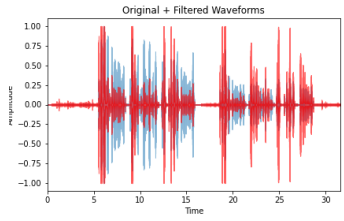


(j) Gym Audio Filtered (Decision Rate = 500 frames).

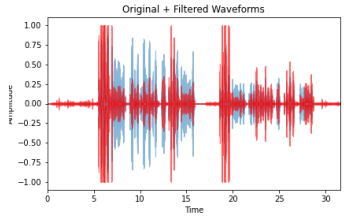
Figure 8: Figures a-e shows audio scenario where filter 1 would be mostly applied with different decision rates. Figures f-j shows audio scenario where filter 2 would be mostly applied with different decision rates. The plots shows the original audio signal (blue) overlapped by the resulting filtered signal (red).



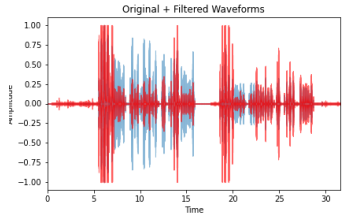
(a) House Env. Audio Filtered (Decision Rate = 50 frames).



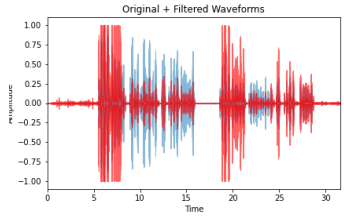
(b) House Env. Audio Filtered (Decision Rate = 100 frames).



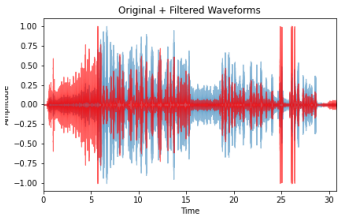
(c) House Env. Audio Filtered (Decision Rate = 200 frames).



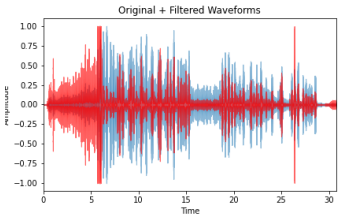
(d) House Env. Audio Filtered (Decision Rate = 300 frames).



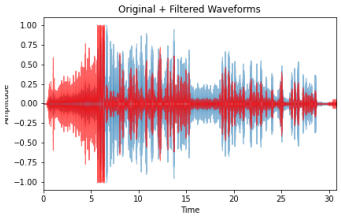
(e) House Env. Audio Filtered (Decision Rate = 500 frames).



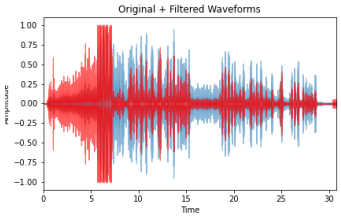
(f) Inside Car Audio Filtered (Decision Rate = 50 frames).



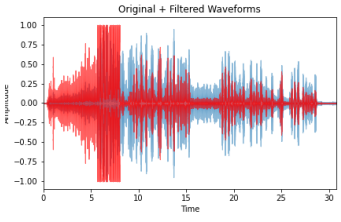
(g) Inside Car Audio Filtered (Decision Rate = 100 frames).



(h) Inside Car Audio Filtered (Decision Rate = 200 frames).

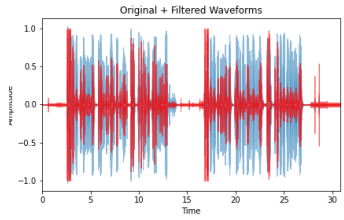


(i) Inside Car Audio Filtered (Decision Rate = 300 frames).

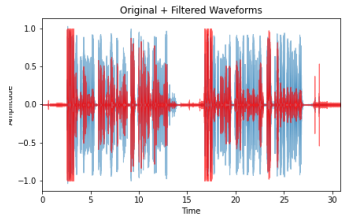


(j) Inside Car Audio Filtered (Decision Rate = 500 frames).

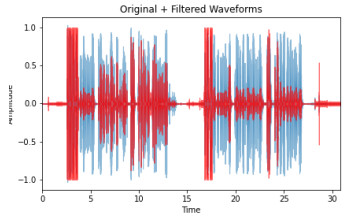
Figure 9: Figures a-e shows audio scenario where filter 3 would be mostly applied with different decision rates. Figures f-j shows audio scenario where filter 5 would be mostly applied with different decision rates. The plots shows the original audio signal (blue) overlapped by the resulting filtered signal (red).



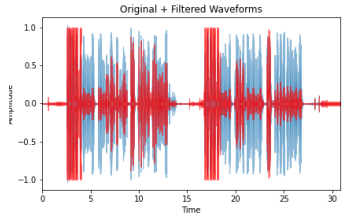
(a) House Env. Audio Filtered (Decision Rate = 50 frames).



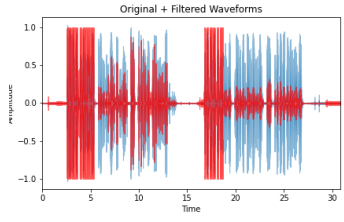
(b) House Env. Audio Filtered (Decision Rate = 100 frames).



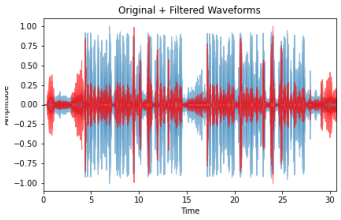
(c) House Env. Audio Filtered (Decision Rate = 200 frames).



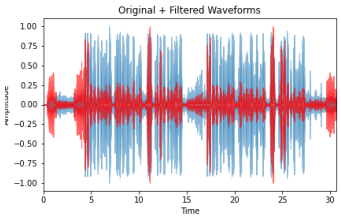
(d) House Env. Audio Filtered (Decision Rate = 300 frames).



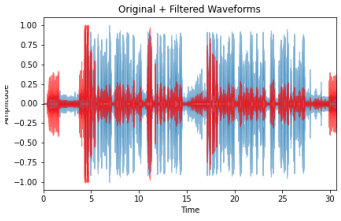
(e) House Env. Audio Filtered (Decision Rate = 500 frames).



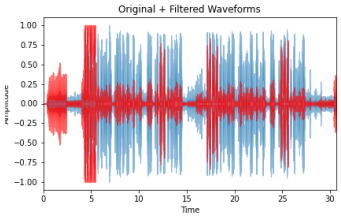
(f) Open Air Audio Filtered (Decision Rate = 50 frames).



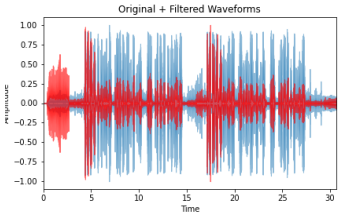
(g) Open Air Audio Filtered (Decision Rate = 100 frames).



(h) Open Air Audio Filtered (Decision Rate = 200 frames).



(i) Open Air Audio Filtered (Decision Rate = 300 frames).



(j) Open Air Audio Filtered (Decision Rate = 500 frames).

Figure 10: Figures a-e shows audio scenario where filter 5 would be mostly applied with different decision rates. Figures f-j shows audio scenario where filter 6 would be mostly applied with different decision rates. The plots shows the original audio signal (blue) overlapped by the resulting filtered signal (red).

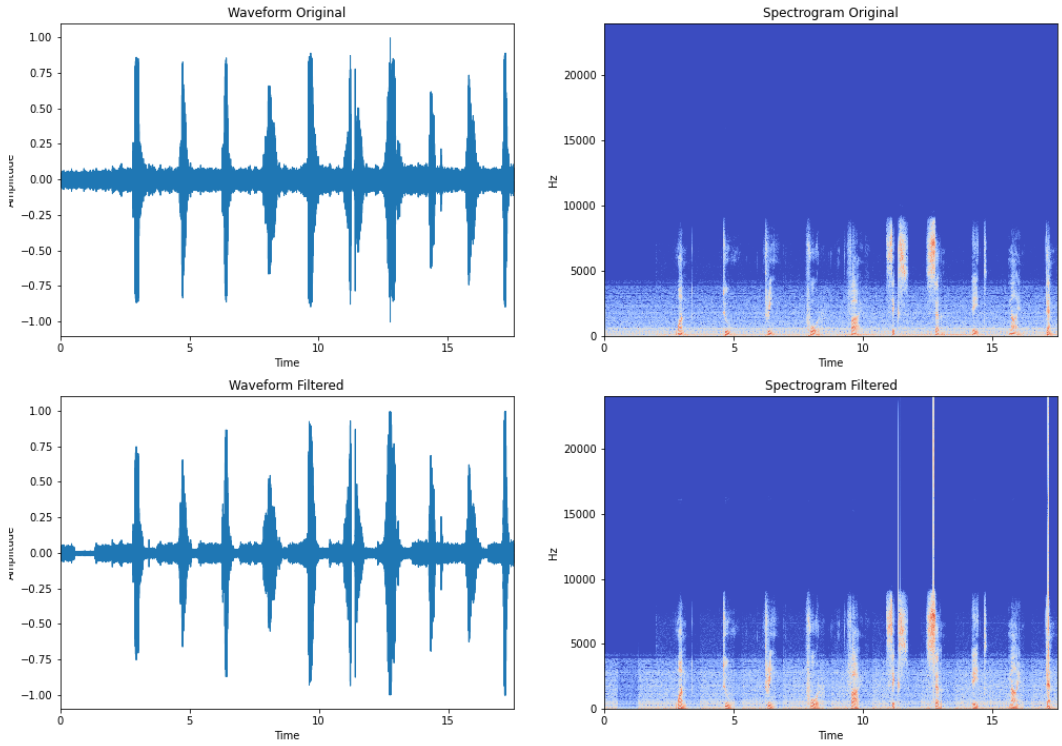


Figure 11: Comparison of original audio and resulting audio from scenario were speech is soft and noise reduction is off. Threshold set to 0.55.

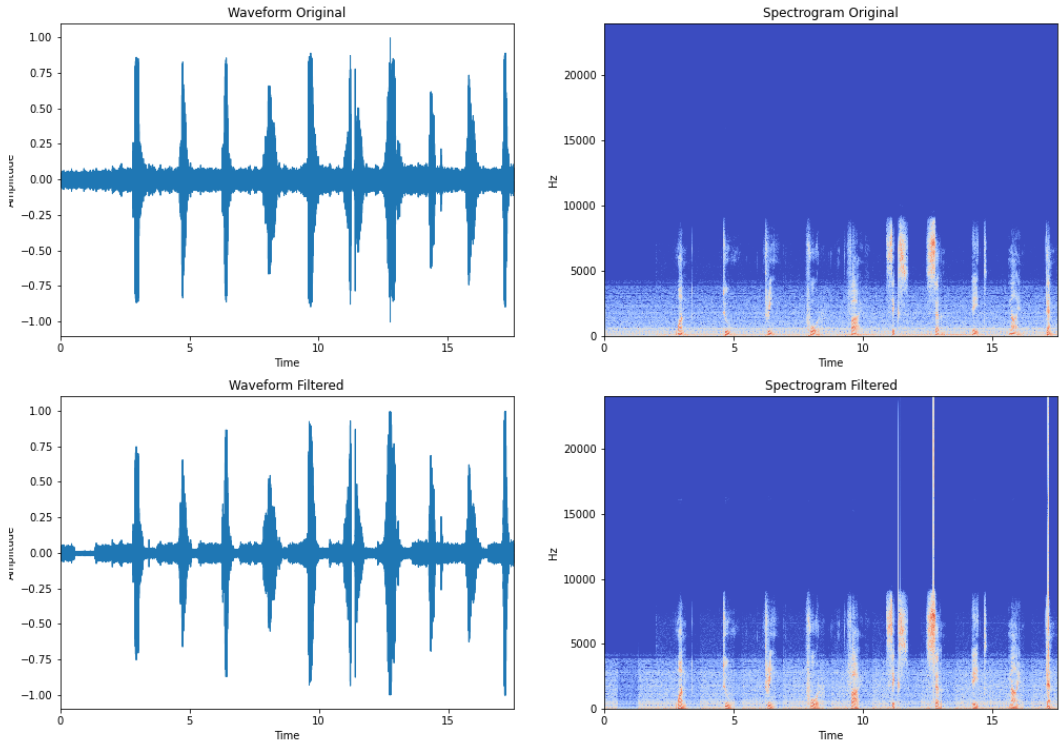


Figure 12: Comparison of original audio and resulting audio from scenario were speech is soft and noise reduction is on. Threshold set to 0.55.

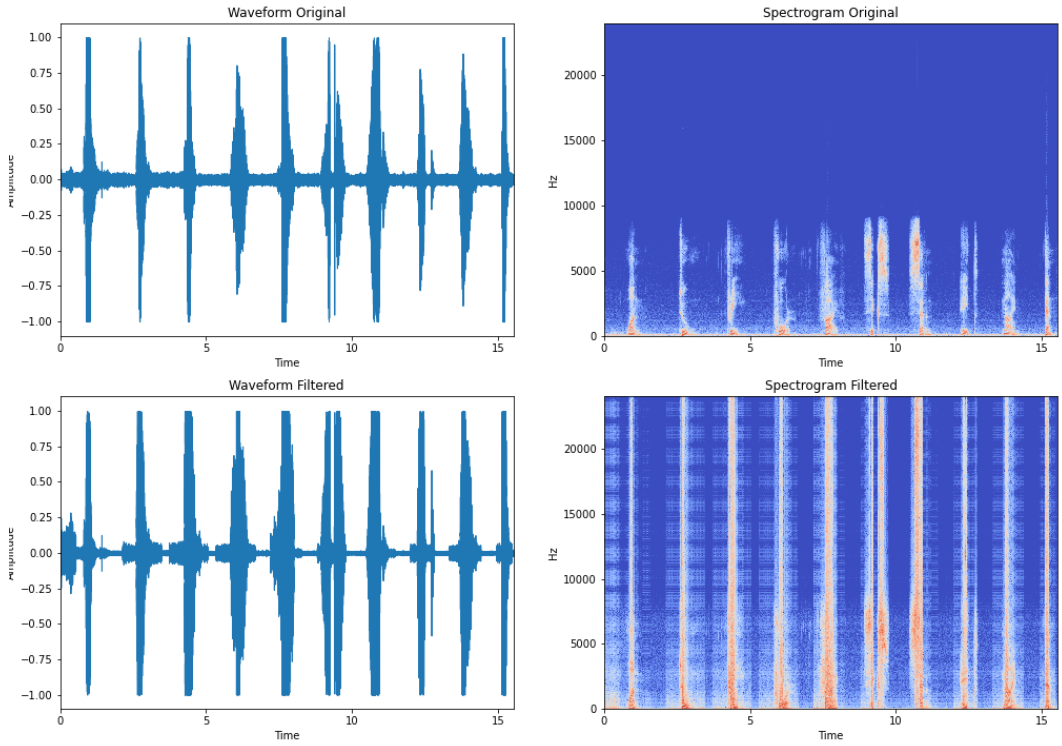


Figure 13: Comparison of original audio and resulting audio from scenario were speech is moderate and noise reduction is off. Threshold set to 0.85.

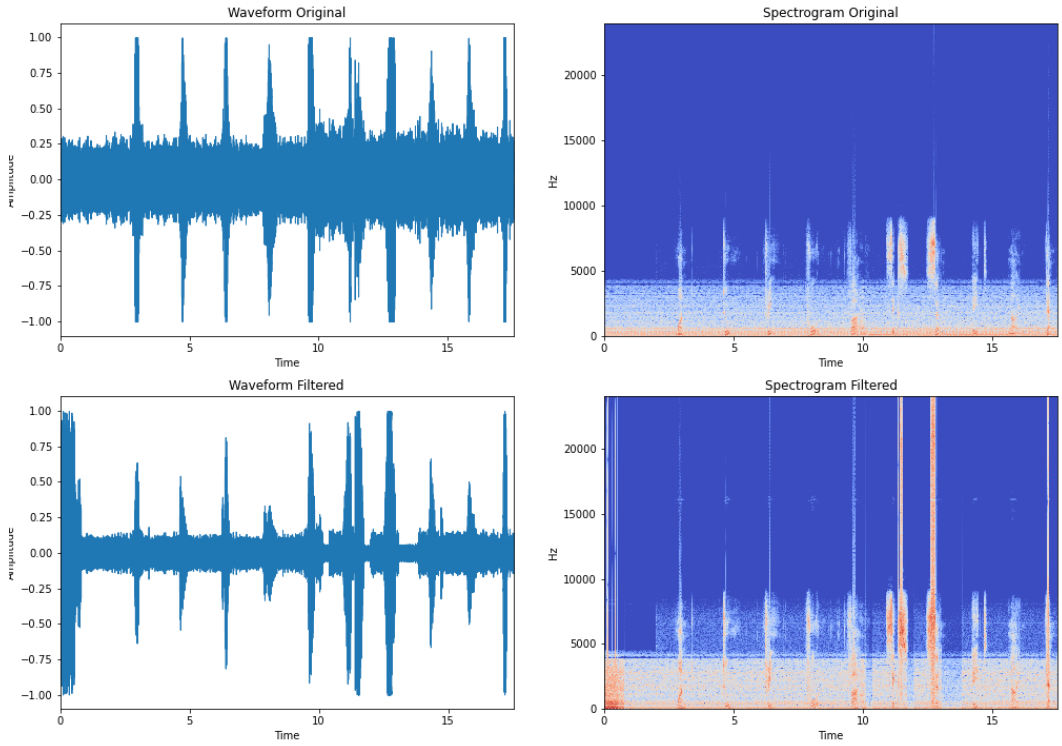


Figure 14: Comparison of original audio and resulting audio from scenario were speech is moderate and noise reduction is on. Threshold set to 0.65.

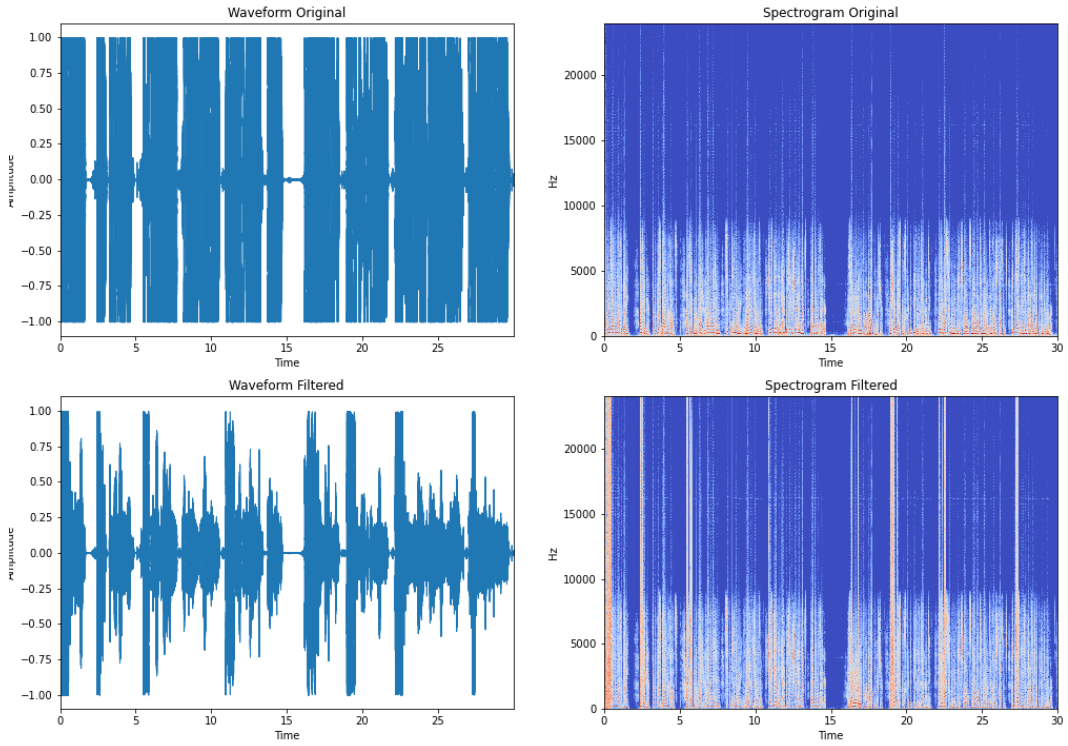


Figure 15: Comparison of original audio and resulting audio from scenario were speech is loud and noise reduction is off. Threshold set to 0.95.

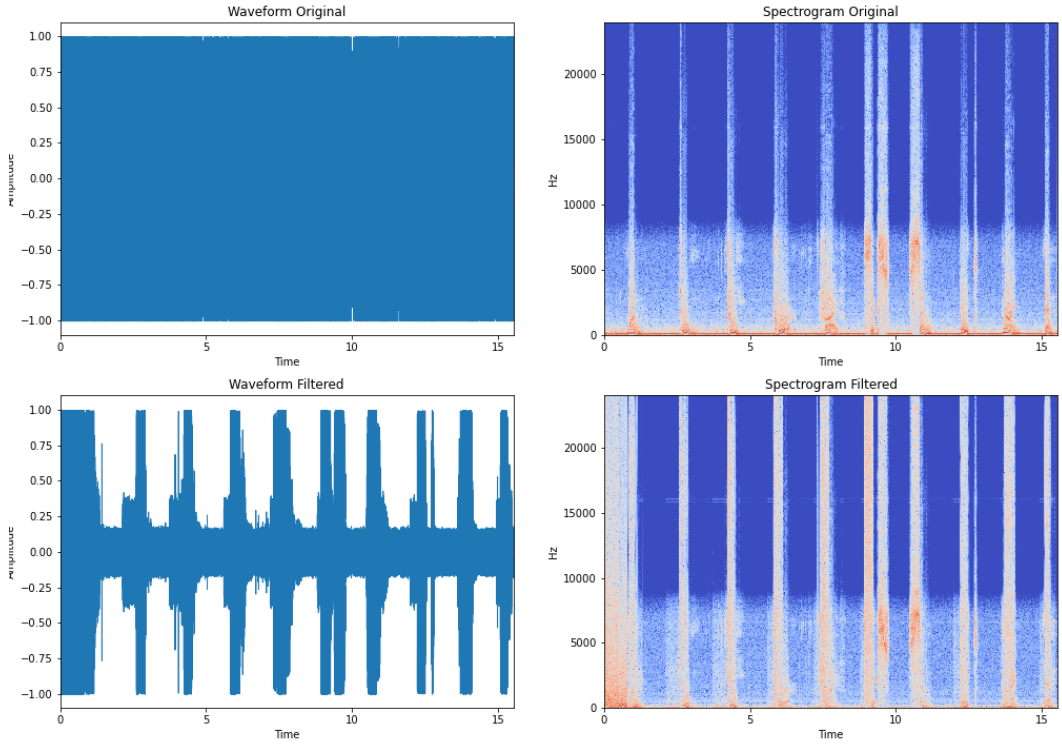


Figure 16: Comparison of original audio and resulting audio from scenario were speech is loud and noise reduction is on. Threshold set to 0.65.

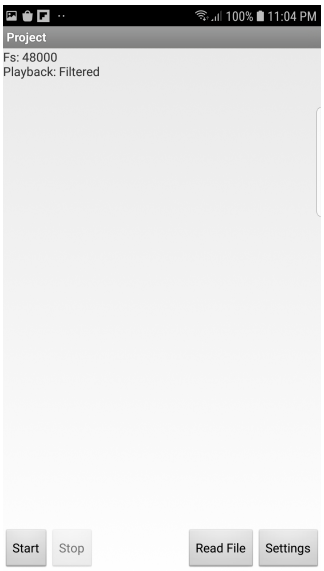


Figure 17: Android App Main Screen.

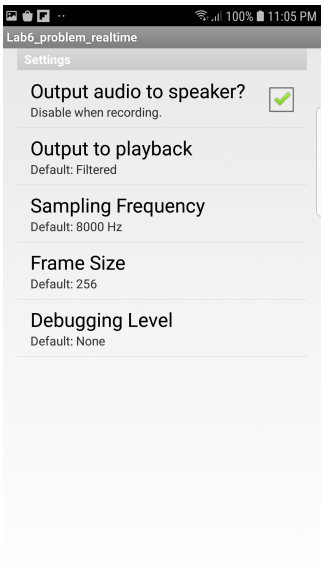


Figure 18: Android App Settings.

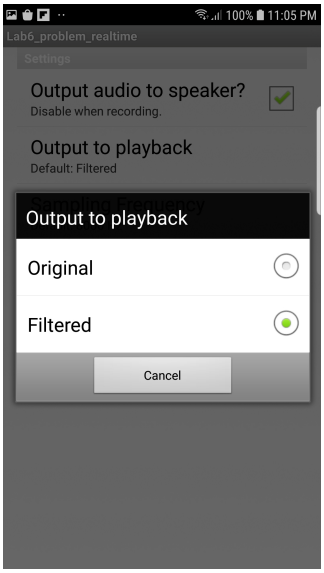


Figure 19: Android App Output Playback Settings.

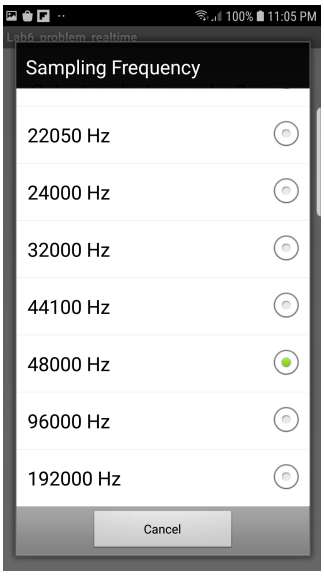


Figure 20: Android App Sampling Frequency Settings.

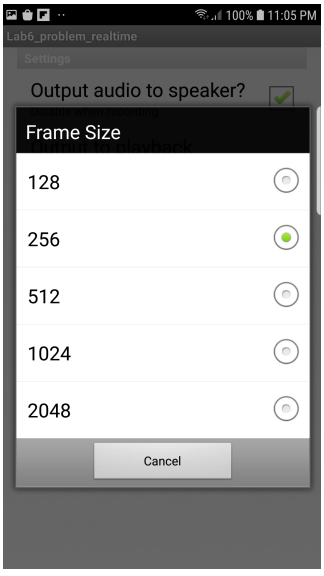


Figure 21: Android App Frame Size Settings.

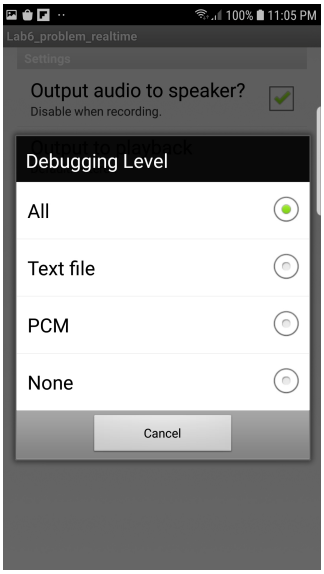


Figure 22: Android App Debuggin Level Settings.

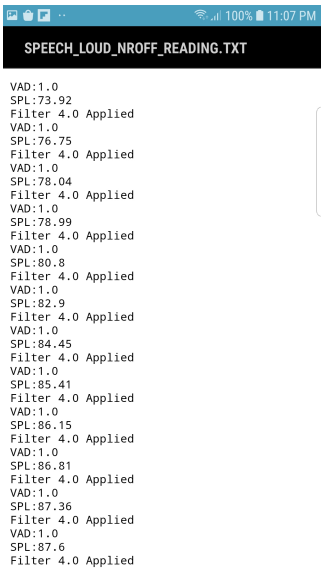


Figure 23: Output file generated from the app when debugging level text file is enabled.

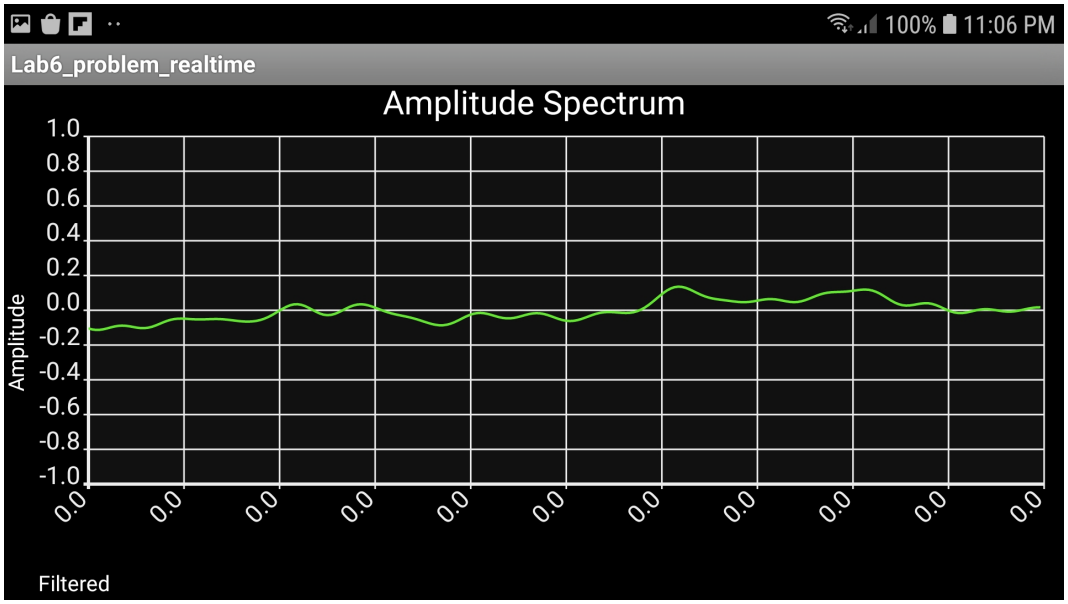


Figure 24: Output waveform generated during audio processing.

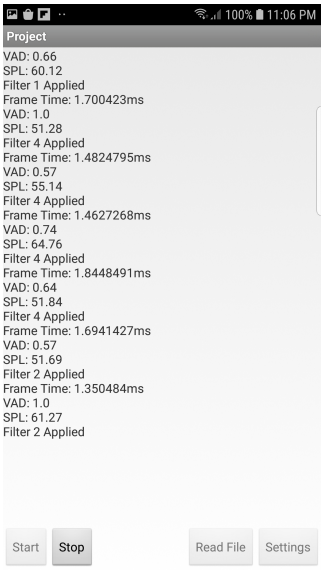


Figure 25: Output shown in main app screen to show current filter state, VAD probability, and SPL value.

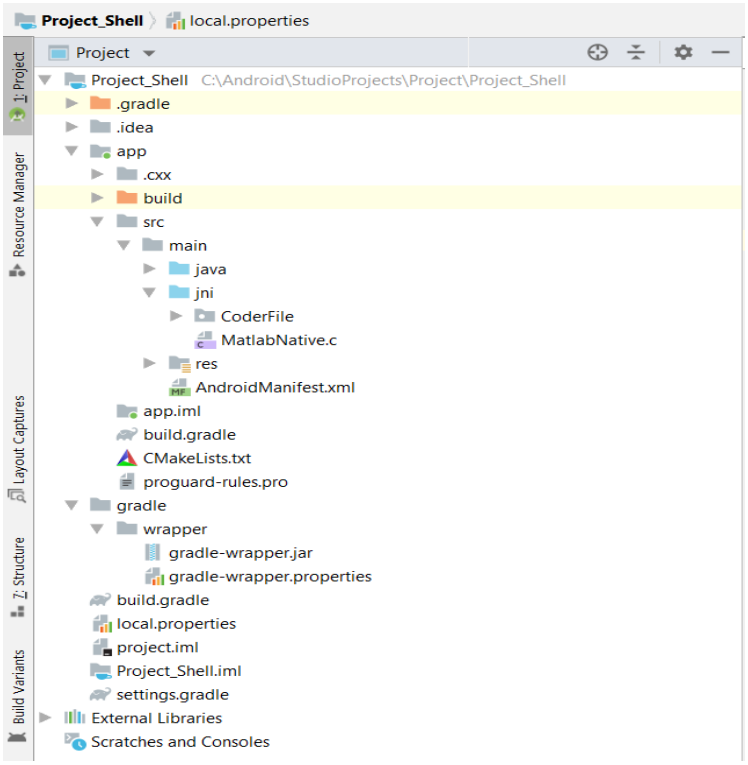


Figure 26: Project organization tree.

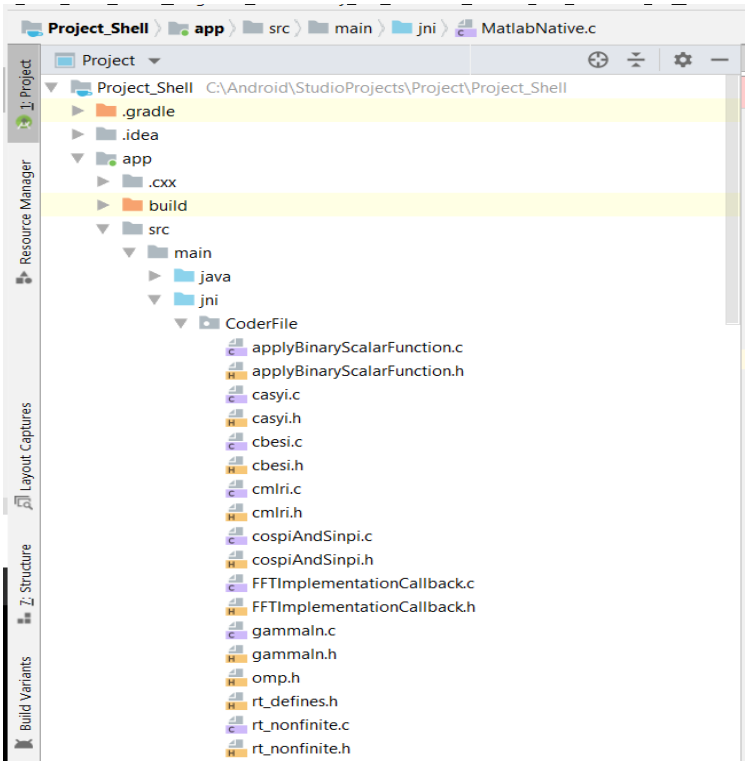


Figure 27: Project functions location.

```
static int frameSize;
static int order = 64;
static float DecisionRate = 100;
static float *buffer = NULL;
static int iteration_counter = 0;
static int overlap_counter = 0;
static int frame_counter = 0;
static double input_buffer[256];
static int filters[100];
static int filter = 1;

//Put six filters here
static double coeffs1[] = {5.57196413532837,-0.103107247646826,-0.135247613103773,-0.181666441913823
static double coeffs2[] = {4.41022692217349,-1.83218821570241,-1.14432409976405,-0.580617087286439,-
static double coeffs3[] = {2.63493719282854,0.138633028511956,0.0942553330069445,0.0250264343153435,
static double coeffs4[] = {2.06008093014519,-0.698392054613850,-0.474672158586362,-0.280440239731934
static double coeffs5[] = {0.877504499139835,-0.00450437615938012,-0.00911523928642912,-0.0158548171
static double coeffs6[] = {0.720880716351334,-0.241317543788798,-0.164369089419349,-0.09412679083993

//Define VAD value and SPL value
double P,L;

//Define flags
int WDRC_flag, NR_flag;
WDRC_flag = 0;
NR_flag = 0;
```

Figure 28: Defining filters with its coefficients and initializing variables and flags.

```
jfloatArray
Java_com_dsp_matlab_Filters_compute(JNIEnv *env, jobject thiz, jfloatArray input, jint Fs)
{
    float *_in = (*env)->GetFloatArrayElements(env, input, NULL);
    jfloatArray output = (*env)->NewFloatArray(env, frameSize);
    float *_output = (*env)->GetFloatArrayElements(env, output, NULL);
    //Your code starts here

    //Initializing variables
    int i,j,n,idx;
    int index;
    double temp;

    //initializing threshold
    float threshold = 0.95;

    //Setting overlap rate
    int overlap = DecisionRate/2;

    //Adding new frames to input Buffer
    for (n=0; n < frameSize; n++){
        input_buffer[n] = _in[n];
    }

    //Computing the VAD and SPL for each input frame
    P = VAD(input_buffer);
    L = SPL(input_buffer);
```

Figure 29: Snippet of the first few lines present in the MatlabNative.c file. Here, we have the variables initialization for threshold and keeping track of overlaps. The computation of VAD and SPL values is seen here as well.

```
//Updating NR flag based on VAD threshold and Noise Level
if( P  < threshold){
    //Noise only frame found
    if ( L <= 55){
        NR_flag = 0;
    }
    else if(L  > 55){
        NR_flag = 1;
    }
}
else{
    //Speech frame found
    if(L <= 55){
        WDRC_flag = 0;
    }
    else if(L > 55 && L <= 75){
        WDRC_flag = 1;
    }
    else if(L > 75){
        WDRC_flag = 2;
    }
}

//updating the iteration counter
if(frame_counter != 0  && frame_counter % overlap == 0){
    iteration_counter++;
}
```

Figure 30: Updating NR and WDRC flags based on values computed for VAD and SPL of each frame processed. Also, increasing of iteration counter.

```
//Making sure 50% overlap for decison making is performed
if(frame_counter < DecisionRate) {
    index = frame_counter;
}
else if(iteration_counter%2 == 0) {
    index = overlap_counter;
}
else{
    index = overlap_counter + overlap;
}

//deciding with filter would be best for each frame and storing decision
//to be used in majority voting when decision rate is met
if(WDRC_flag == 0 && NR_flag ==0){
    //Filter 1
    filters[index] = 1;
}
else if(WDRC_flag == 0 && NR_flag ==1){
    //Filter 2
    filters[index] = 2;
}
else if(WDRC_flag == 1 && NR_flag ==0){
    //Filter 3
    filters[index] = 3;
}
else if(WDRC_flag == 1 && NR_flag ==1){
    //Filter 4
    filters[index] = 4;
}
else if(WDRC_flag == 2 && NR_flag ==0){
    //Filter 5
    filters[index] = 5;
}
else if(WDRC_flag == 2 && NR_flag == 1){
    //Filter 6
    filters[index] = 6;
}
```
fi

Figure 31: Managing 50% overlap based on iteration values and updating filter values based on flags and storing it for majority voting.

```
//Increasing overlap counter to keep track of overlap state
overlap_counter ++;

//Increasing count to keep track of amount of frames processed
frame_counter ++;

//Checking if Decision Rate is met
if(frame_counter >= DecisionRate){

    //Checking if overlap is met
    if(overlap_counter >= overlap){

            //computing majority voting to decide what filter to use
            int maxCount = 0;
            for (i = 1; i < 7; ++i) {
                int count = 0;

                for (j = 0; j < DecisionRate; ++j) {
                    if (filters[j] == i)
                        ++count;
                }
                if (count > maxCount) {
                    maxCount = count;
                    filter = i;
                }
            }
        //Resetting overlap counter
        overlap_counter = 0;
    }
}
```

Figure 32: Increasing counter and getting filter value to be applied using majority voting system based on previous frames.

```
//Inputting frames into a buffer to be filtered
for(i=0; i<order; i++) {
    buffer[i] = buffer[frameSize + i];
}
for(i=order; i<(frameSize+order); i++)
{
    buffer[i] = input_buffer[i-order];
}

//Filtering input audio frames based on WDRC flag and NR flag to select filter
for(i=0;i<frameSize;i++)
{
    temp = 0.0;
    for(j=0;j<order;j++)
    {
        idx = 32 + (i - j);

        if(filter == 1){
            //Using Filter 1
            temp += buffer[idx]*coeffs1[j];
        }
        else if(filter == 2){
            //Using Filter 2
            temp += buffer[idx]*coeffs2[j];
        }
        else if(filter == 3){
            //Using Filter 3
            temp += buffer[idx]*coeffs3[j];
        }
        else if(filter == 4){
            //Using Filter 4
            temp += buffer[idx]*coeffs4[j];
        }
        else if(filter == 5){
            //Using Filter 5
            temp += buffer[idx]*coeffs5[j];
        }
        else if(filter == 6){
            //Using Filter 6
            temp += buffer[idx]*coeffs6[j];
        }
    }

    //Filtered output
    _output[i] = temp;
}
```

Figure 33: Passing input frames into buffer and filtering input based on filter decision performed previously to generate output signal.

```
Java_com_dsp_matlab_Filters_finish(JNIEnv *env, jobject thiz)
{
    //Resetting back all flags to original state
    if(buffer != NULL){
        free(buffer);
        buffer = NULL;
    }
    WDRC_flag = 0;
    NR_flag = 0;
    filter = 1;
    iteration_counter = 0;
    overlap_counter = 0;
    frame_counter = 0;
}
```

Figure 34: Resetting flags, counters, and filters after app is done running.