

Audio Processing in Time Domain

Authors: Parth Patel, Parmin Patel, Edward Na

Group#6

Date: Friday Nov 25th 2022

SYDE 252: Linear Systems and Signals

Dr Charbel Azzi

Abstract

The goal of this project is to implement audio processing techniques to various audio files outlined in the document. Through these sounds, three different filters were designed with the intention of eliminating all background noise in the audio files. These filters varied slightly, where they filtered based on the moving average, weighted average, and median of the selected samples. The sounds were first written into new files, and tested under these filters to decide which implementation obtained the best result. After this process, the filters were tuned and analyzed to obtain an “accuracy” rating of the filters respectively. The latter portion of the project analyzed the given audio files and analyzed quantifiable characteristics about the audio files themselves. The Drum.wav audio file, for example, was analyzed to find the average bpm (beats per minute).

Based on these operations, the analysis showed that the mean filter was the best overall filter, although having its own drawbacks. In conclusion, a combination of filters instead of just one is most optimal to filter out the background noise while maintaining the integrity of the audio's sound quality. Additionally, a window size of 3 was calculated as the best size, where the majority of the background noise was filtered without sacrificing the quality of the audio itself.

Table of Contents

Abstract	2
Table of Contents	3
List of Figures	5
List of Tables	5
Introduction	17
Background	20
Methodology	22
3.1 Sound Preparations	22
3.2 Filters	22
3.3 Analyzing Signals	39
4 Results and discussion	42
4.1 Part 1 Results	43
4.2 Part 2 Results	44
4.3 Part 3 Results	46
Conclusions and recommendations	47
Acknowledgement	50
References	50
Appendix	51

List of Figures

Figure 1: Taken from Lecture 8 - Convolution Sum and Convolution Integral

The image is the summation equation that describes a convolution between 2 signals.

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m]$$

Figure 2:

The image shows how the median value is found for a given set of numbers.

$$\begin{aligned} [1] \quad m &= x_k; \quad k = \frac{(n+1)}{2} \quad (n \text{ odd}) \\ [2] \quad m &= \frac{(x_k + x_l)}{2}; \quad k = n/2; l = (n/2) + 1 \quad (n \text{ even}) \end{aligned}$$

Figure 3:

This figure shows the original signal and the output when it is passed through the median filter.

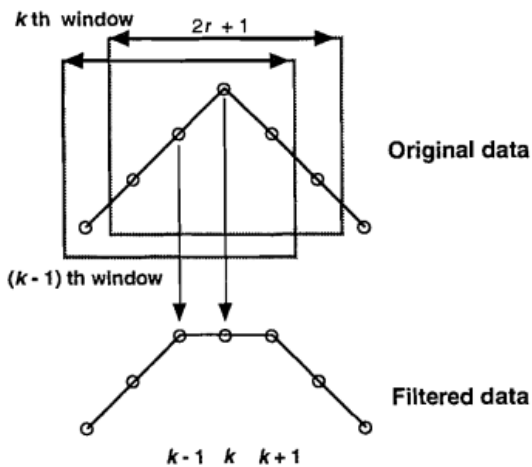


Figure 4:

Moving Average Filter

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[n-k], \quad L = \text{window size}$$

A dirac function: $\delta[n]$ needs to be applied:

$$x[n] = \delta[n]$$

Resulting in:

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} \delta[n-k]$$

Therefore, to describe the moving average function as an impulse response:

$$y[n] = x[n] * y'[n]$$

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[k] \delta[n-k]$$

Figure 5:

Continuous time

$$y(t) = x(t) * y'(t)$$

$$y'(t) = \frac{1}{L} \int_0^{L-1} \delta(t-\tau) d\tau$$

$$y(t) = \frac{1}{L} \int_0^{L-1} x(\tau) \delta(t-\tau) d\tau$$

Figure 6:

Calculations for a high pass filter with a window size of 3

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} b_k x[n-k]$$

$$h[n] = x[n] - y[n]$$

$$h[n] = x[n] - \frac{1}{L} \sum_{k=0}^{L-1} b_k x[n-k]$$

So with a window size of
3 for example:

$$h[n] = x[n] - \left[\frac{1}{3} (x[n] + x[n-1] + x[n-2]) \right]$$

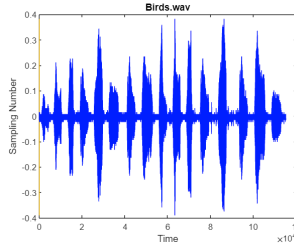
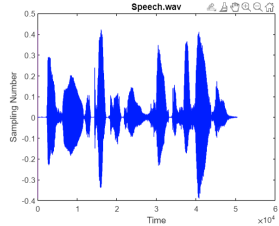
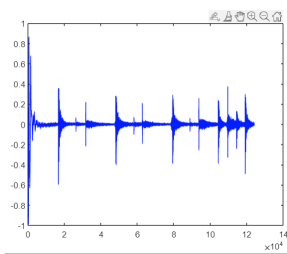
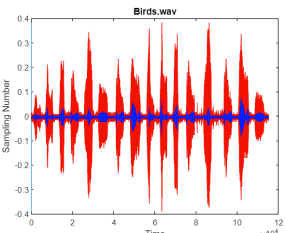
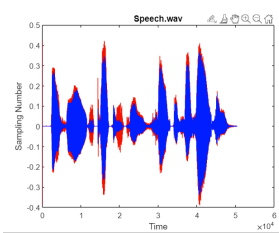
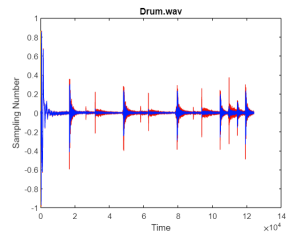
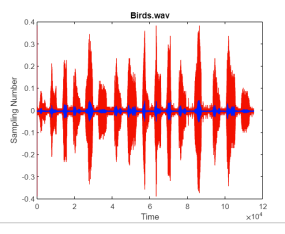
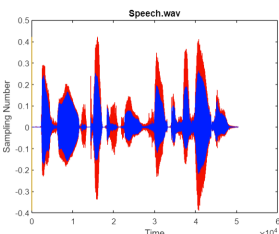
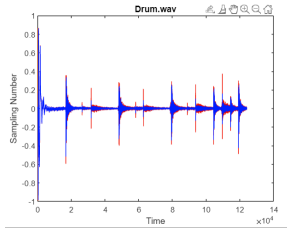
$$= x[n] - \frac{1}{3} x[n] - \frac{1}{3} x[n-1] - \frac{1}{3} x[n-2]$$

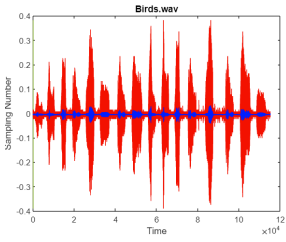
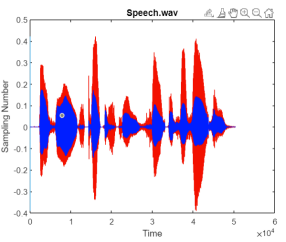
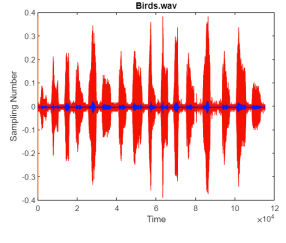
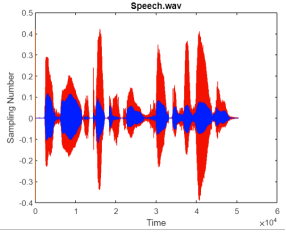
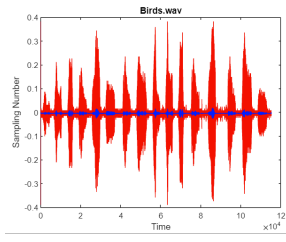
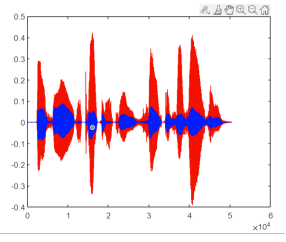
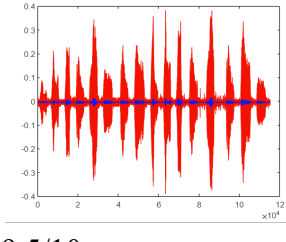
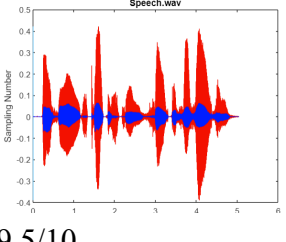
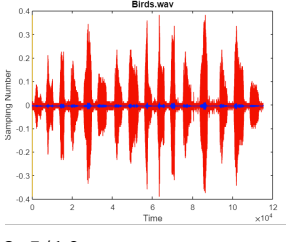
$$h[n] = \frac{2}{3} x[n] - \frac{1}{3} x[n-1] - \frac{1}{3} x[n-2]$$

↳ high pass filter with window size, 3

List of Tables

Table 1: Illustrates the full process into finding the best window sizes for the different sound files by giving a rating on a scale of 10 along with an image which shows the original signal and the filtered signal. Original signal is visualized in blue and the filtered signal is in red.

Window Size	Birds.wav Filtering	Speech.wav Filtering	Drum.wav Filtering
1	3/10 	3/10 	 3/10
5	8/10 	 5/10	Window Size 5  9/10
10	 8.5/10	 7/10	Windows Size 3 

15	 <p>A waveform plot titled 'Birds.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.4. The x-axis is labeled 'Time' and ranges from 0 to 12, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	 <p>A waveform plot titled 'Speech.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.5. The x-axis is labeled 'Time' and ranges from 0 to 6, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	
	9/10	8/10	
20	 <p>A waveform plot titled 'Birds.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.4. The x-axis is labeled 'Time' and ranges from 0 to 12, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	 <p>A waveform plot titled 'Speech.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.5. The x-axis is labeled 'Time' and ranges from 0 to 6, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	
		9/10	
25	 <p>A waveform plot titled 'Birds.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.4. The x-axis is labeled 'Time' and ranges from 0 to 12, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	 <p>A waveform plot titled 'Speech.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.5. The x-axis is labeled 'Time' and ranges from 0 to 6, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	
	9.5/10	10/10	
30	 <p>A waveform plot titled 'Birds.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.4. The x-axis is labeled 'Time' and ranges from 0 to 12, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	 <p>A waveform plot titled 'Speech.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.5. The x-axis is labeled 'Time' and ranges from 0 to 6, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>	
	9.5/10	9.5/10	
35	 <p>A waveform plot titled 'Birds.wav'. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.4. The x-axis is labeled 'Time' and ranges from 0 to 12, with a multiplier of $\times 10^4$ at the end. The plot shows a red waveform with high-frequency oscillations and a blue line representing the mean or a specific component.</p>		
	9.5/10		

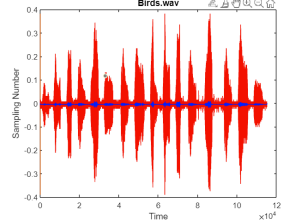
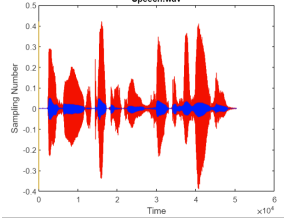
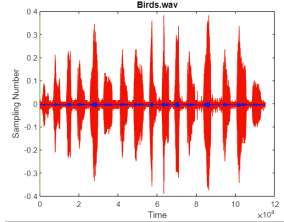
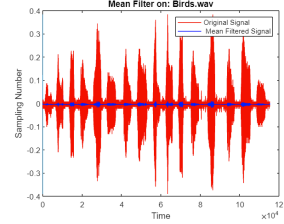
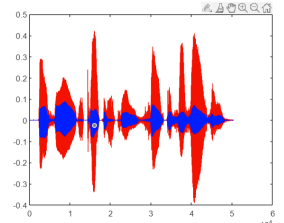
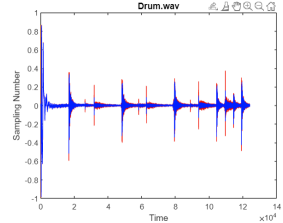
40	 <p>9.5/10</p>	 <p>9/10 (bit too filtered, sounds muffled)</p>	
45	 <p>10/10</p>		

Table 2: The table simplifies table 1 by keeping just the best window sizes for the sound files and explains briefly on why the window size was selected.

Window Size	Birds.wav	Speech.Wav	Drums.Wav
Mean Filter	<p>3</p> <p>At a window size of 3 most of the background noise was removed, however an increase in windows size after caused a large loss of information.</p> 	<p>25</p> <p>All background noise was removed from the audio, and further increase in window size muffled the speaker, hindering the sound quality.</p> 	<p>3</p> <p>The drums audio had a very minimal background sound, so only a minimal window size of 3 was required to get a clear sound.</p> 
Gaussian Filter	40 (Still background noise)	20	4

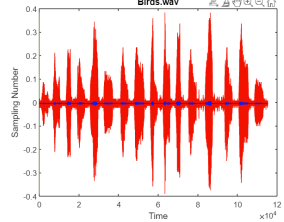
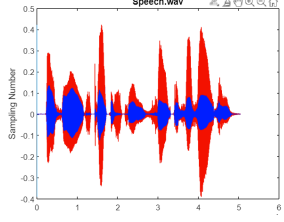
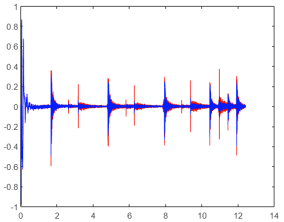
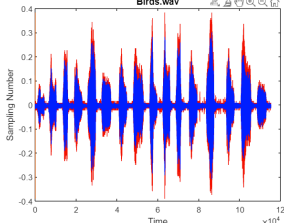
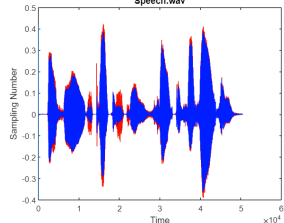
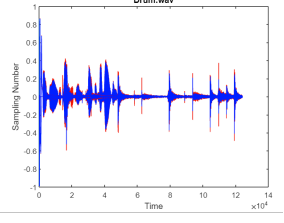
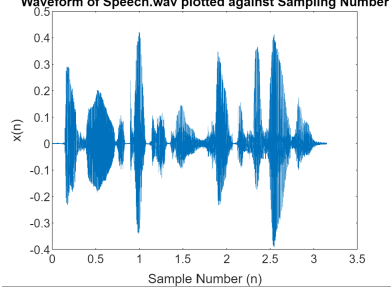
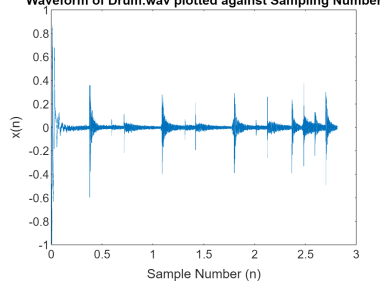
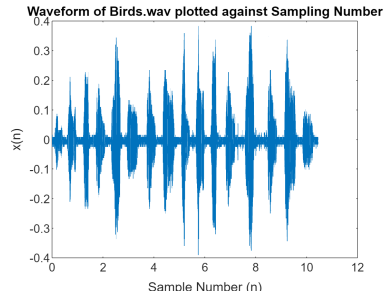
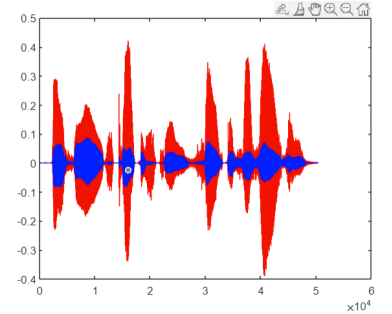
	 <p>While the background noise got considerably lower, it was still audible; The window size could not be increased as the sound amplitude of the output wave would be too small to be heard.</p>	 <p>After a window size of 20, all background noise was removed, however the speaker became increasingly more muffled (more than the mean filter).</p>	 <p>As mentioned in the previous filter, the background noise in the drums was minimal so only a window size of 5 was needed to create a clear audio from the input.</p>
Median Filter	<p>2 (still a lot of background noise)</p>  <p>At a window size 2, a lot of the background noise in the audio was still present. However, after any window size of 3 and above, there was a distinct distortion in the audio that affected the quality of the sound, likely due to the high peaks in the audio file</p>	<p>3 (No background noise)</p>  <p>At a window size of 3, any window size 4 and above caused a distinct distortion impacting the quality of the audio.</p>	<p>3 (No background noise)</p>  <p>At a window size of 3, any window size 4 and above caused a distinct distortion impacting the quality of the audio.</p>

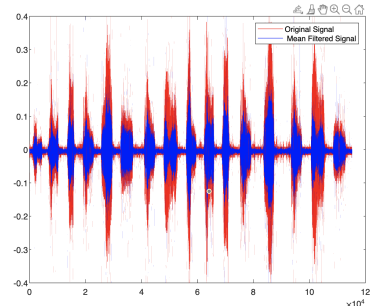
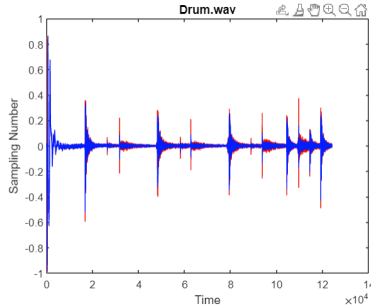
Table 3: The results from Part 1

Sound	Sampling Rate (Hz)	Plot
-------	--------------------	------

Speech.wav	16,000	<p>Waveform of Speech.wav plotted against Sampling Number</p> 
Drum.wav	44,100	<p>Waveform of Drum.wav plotted against Sampling Number</p> 
Birds.wav	11,025	<p>Waveform of Birds.wav plotted against Sampling Number</p> 

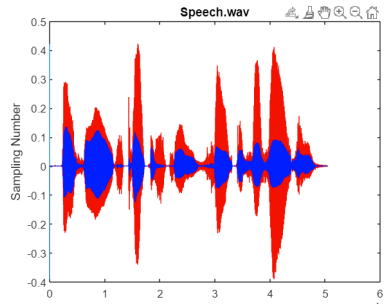
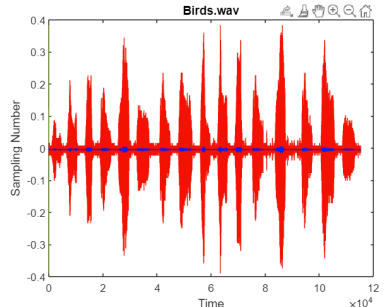
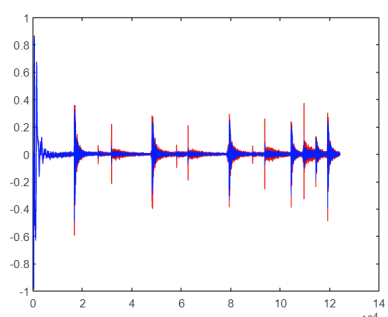
The moving average filter applied to each of the sound clips is shown in the following Table 4:

Sound	Window Size	Plot
Speech.wav	25	

Birds.wav	3	 <p>The plot for Birds.wav shows the original signal (red) and the mean filtered signal (blue) over time. The y-axis ranges from -0.4 to 0.4, and the x-axis ranges from 0 to 12 x 10⁴. The original signal is highly oscillatory, while the mean filtered signal is a smooth, low-frequency curve.</p>
Drum.wav	3	 <p>The plot for Drum.wav shows the original signal (red) and the mean filtered signal (blue) over time. The y-axis ranges from -1 to 1, and the x-axis ranges from 0 to 14 x 10⁴. The original signal is highly oscillatory, while the mean filtered signal is a smooth, low-frequency curve.</p>

The gaussian filter applied to each sound clip is shown in the table 5 below:

Sound	Window Size	Plot
-------	-------------	------

Speech.wav	20	
Birds.wav	40	
Drum.wav	4	

The Median filter applied to each sound clip is shown in the table 6 below:

Sound	Window Size	Plot
-------	-------------	------

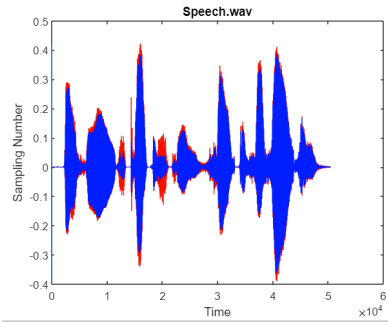
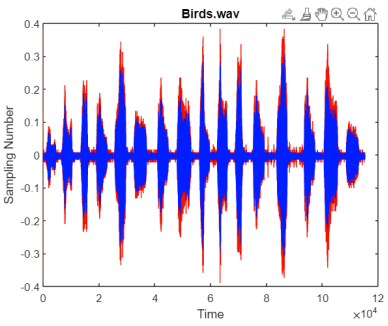
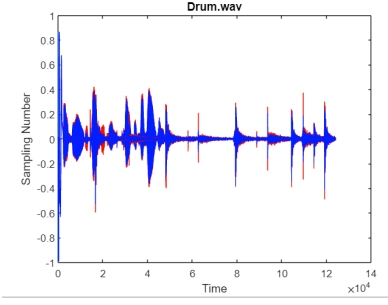
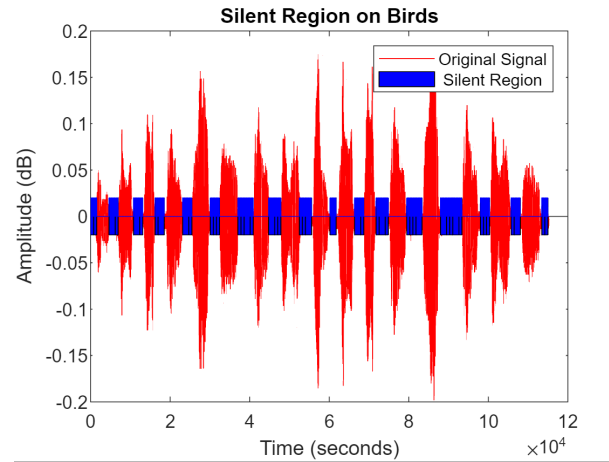
Speech.wav	3	
Birds.wav	2	
Drum.wav	3	

Table 7 depicts the results for part 3. For determining the syllables,

Question	Results
Number of syllables for the Speech.wav clip	10 syllables
Beats per Minute of the Drum.wav clip	170.60 Beats Per Minute

Silent region of the Birds.wav clip



1. Introduction

Over the last decade, audio processing has made a huge jump in popularity, especially in the music industry. It's no wonder either; music is a universal language; over a mere pattern of sounds, emotions can be conveyed and felt to hearts of listeners all over the world. In 2026, the audio processing market will grow nearly 15% [1], so the technology is here to stay. However, it can be confusing on what audio processing can entail. Simply put, a good audio engineer would be able to enhance and suppress certain features of an audio. Throughout the project, different algorithm and analysis techniques were used to enhance audio files so that they can produce clean, production-ready sound.

The report is provided with three audio files in the first part of the project, and asked to provide the sampling rate, its channels, and understand the basics of manipulating audio files using Matlab. From here, the importance of sampling rate and audio channeling was learned, as well as how it can impact the sound quality in an audio file. All in all, a higher sampling rate tends to deliver higher quality audio production, and stereo sounds are able to create a sense of width as opposed to mono sounds.

The next portion of the project focuses more on filtering the background sounds of an audio to make it as clear as possible. To do this, Three different low pass filters were implemented to cut out the noise: the mean (or moving average) filter, the weighted average filter, and the median filter. The mean filter essentially takes the moving average of the filter, the weight average filter is based on an average where a specific sample is given a higher importance. Finally, the median filter takes the median values of the selected averages. It is important that each filter has an adjustable window size in which each average is calculated for

the filter. Increasing this window size tends to improve the filter out the background sound of the audio, although the quality of the sound may get distorted or have a low output volume.

Page 5 of 8

Based on trial and error, it was deduced that the mean average filter produced the clearest audio for the sound files without causing any major disruptions. While the Gaussian weighted average filter also produced a clear sound, the background wasn't completely removed without sacrificing the volume output of the audio. The median filter was able to clear some background sound, but after a small increase in window size, the sound was able to get heavily distorted.

On the third and final part of the project, various analysis techniques were conducted to deduce numeric characteristics about the three audio files. To calculate the number of syllables in the speech audio file, each frame was analyzed and calculated the average amplitude in that window. If the calculated amplitude was greater than the chosen amplitude threshold, the current frame was counted as a syllable. This was repeated until all the frames in the signal were considered. The bpm (beats per minute) on the drums audio file used a very similar calculation, with the final step dividing the total number of counted beats by the time (in minutes) of the audio file; the bpm relatively easily. To detect the silent regions in the bird clip, the clip was first passed into the moving average filter; its purpose was to remove any background noise/useless noise, ultimately smoothing the signal out. Some portions of the signal were taken, with its peak amplitude being recorded. If the amplitude was higher than the chosen threshold, the amplitude was set to zero. This preserved only the "silent" regions of the bird audio clip.

In conclusion, various algorithms were used to filter three different signals, and ultimately removed the background noise of each of the signals. Additionally, analysis was conducted to

calculate certain factors such as number of syllables, beats per minute, and detecting silent regions in the audio clips.

2. Background

Since the accelerometers first discovery in 1923, accelerometers have become a necessity to society in nearly all technological industries [1]. From airbag activation in all commercial cars, to the compasses on modern phones, these electromagnetic sensors are the primary device to measure accelerated movement and sudden changes in force. Recent discoveries have even allowed accelerometers to detect large scale earthquakes and design bionic limbs for amputated patients [2].

Although its use cases are countless, the accelerometer has a simple design. There are two main parts, also known as the piezoelectric effect and the capacitance sensor. The piezoelectric effect is a special type of sensor that uses microscopic crystals to detect changes in acceleration, where the crystals become stressed with movement [3]. This stress creates a noticeable voltage that is picked up from the accelerometer, which can determine both the velocity and direction through the crystals [3]. This type of piezoelectric accelerometer, also known as a vibration sensor, is commonly used in electronic devices and only viable in high heat applications of 120C and upwards [4].

The second most common type of accelerometer is the low impedance accelerometer. Its parts consist of a charge in its front end, a built in circuit, and finally a FET transistor that converts the front end charge into a desired low-impedance voltage. Using this voltage, the accelerometer can determine the velocity and direction accurately. The low impedance accelerometers are commonly used in the mechanical industry, such as safety installations,

control systems, and machines. All in all, this small but powerful sensor may not be noticed but remains a staple in today's technological society.

3. Methodology

3.1 Sound Preparations

To start, each sound was read using Matlab's built in `audioread()` function which then gave the signal information and sampling rate as $[y, Fs]$. The sampling rate for each sound was determined through Fs and to check and see if a sound was Mono (1 channel/column) or Stereo (2 channel/column), the signal information, y , was accessed as an $M \times N$ matrix, where N is the number of columns. If N is greater than 1, it was stereo and then got converted to Mono. To convert to mono, all the channels of the audio were summed and averaged to create the desired audio. Thus, with this the audio was prepared for the next stages.

3.2 Filters

A brief introduction of the filter will be given followed by its equation. Then the plot will be displayed for the best window size for that filter. A mathematical interpretation is required to understand how the filter works so there will be a description for it. After this process is completed with all the filters the question "which filter and window size worked best?" will be answered along with the other 2 questions. 1) describe the moving average filter as an appropriate impulse response function in the discrete and continuous domain. 2) Using Eq.(1) how would you define a high pass filter? provide only one mathematical definition and explain what it does in one example as I described each filter above.

3.2.1 Moving Average Filter Design

The moving average filter is a filter that is used for smoothing signals that overshoot or have other background noise. It is generally used for eliminating noise fluctuations to retain the

actual signal that is required for the purpose [4]. Therefore, it is best at removing random white noise while keeping the sharpest step response [4].

Its equation in time domain can be denoted using:

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[n - k]$$

This equation has a window size which is L and taking the sum of it after applying a time shift results in a $Y[n]$ value. This process is also known as convolution as seen in the equation below. The $x[m]$ is the window size.

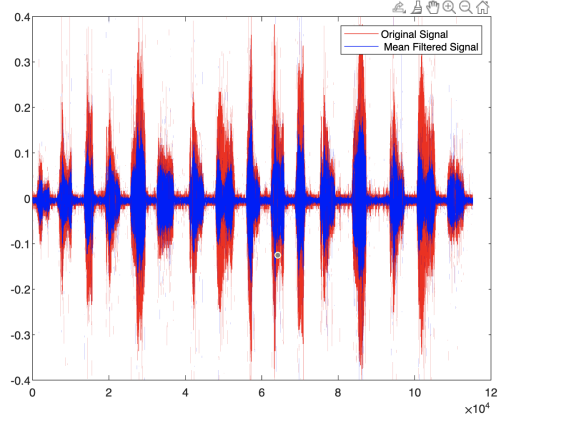
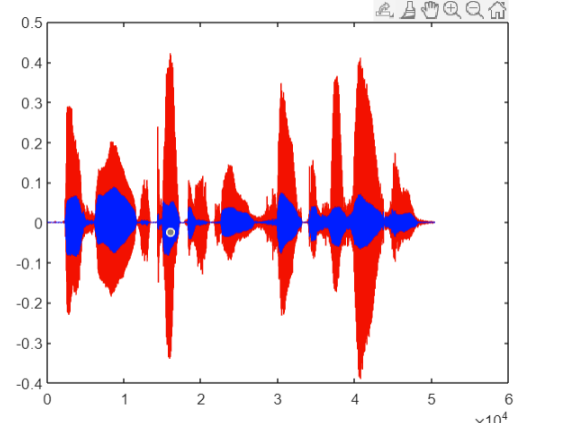
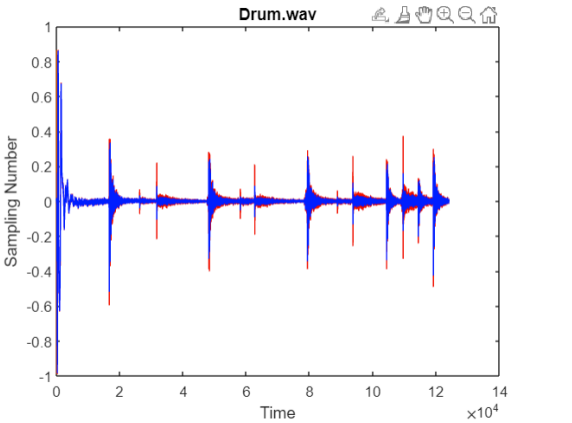
$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n - m]$$

Mathematical interpretation of the filter:

To build the moving average filter the initial step was to use the `audioread` function to parse the sound signal and store it. As already mentioned above some sort of convolution needs to be applied. To apply the convolution there is an inbuilt `conv` function which was utilized to find the convolution between the initial signal and another signal which was set to $[1, 1, 1, 1]$. The length of the new signal is the window size. Using the convolution function shown in class the sound signal can be represented as $x[n]$ and the other signal is determined as $h[n]$ which are convoluted together. The convolution can be written as a sum as well. However, before the convolution the array is divided by the window size to get the average. Therefore, to smoothen the signal the original signal is being convoluted with the average signal leading to a decrease in

background noise.

Plot:

Sound File	Plot
Birds.wav - Window size = 3	
Speech.wav - Window Size = 25	
Drums.wav - Window Size = 3	

The Blue region represents the filtered sound and the red region is the original function.

3.2.2 Gaussian Filter Design

The purpose of the gaussian filter is to prevent the overshoot when a step function is inputted while minimizing rise and fall [5]. To implement the filter the weight was retrieved using the gausswin function for a given window size and since the standard deviation was unknown a value of 1 was used. Then the weight was manipulated by dividing the weight by the sum of the weight. Finally the filtering happened using the filter function by having the original signal weight and the standard deviation. Finally, the signal was filtered using this process.

Gaussian Equation:

$$y[n] = \sum_{k=0}^{L-1} b_k x[n - k]$$

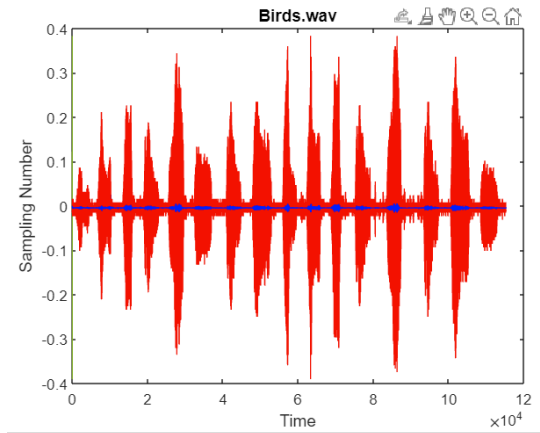
Mathematical interpretation of the filter:

The Gaussian filter modifies the input signal by convolution with a gaussian function which leads to a transformation. It is the same concept such as the moving average filter but Bk's weight should equal to 1. Instead of multiplying it by the average weight it is multiplied by a value whose sum is 1.

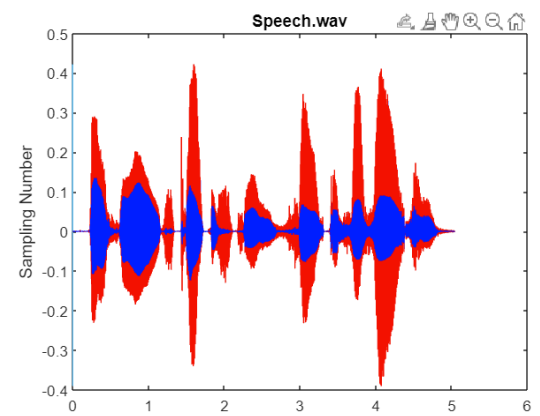
Plot:

Sound File	Plot
------------	------

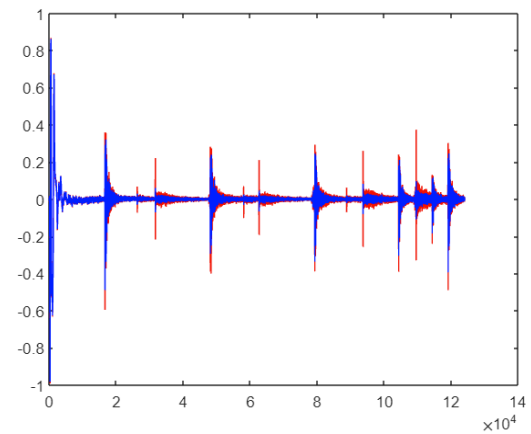
Birds.wav - Window Size = 40



Speech.wav - Window Size = 20



Drums.wav - Window Size = 4



The Blue region represents the filtered sound and the red region is the original function.

3.2.3 Median Filter Design

The median filter serves its purpose by also removing background noise which is similar to the moving average. However, if there is a very high pitch of a sound that is detected in a sound signal the moving average filter may not be the most optimal in removing that since it would be hard for it to detect it due to its high amplitude which impacts its overall average. The median filter would be able to detect that noise and filter it out since it used the concept of median so its value is filtered in a simple way.

Mathematical Interpretation of the Median Filter:

Equation in time domain:

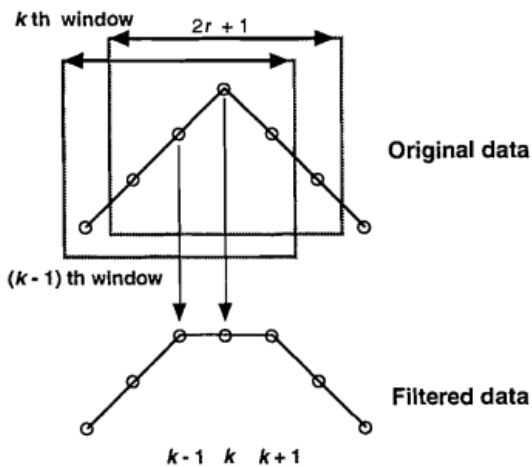
$$y[t] = \text{median}(x[t], x[t - 1], x[t - 2] \dots x[t - k]) \text{ for } k = 0, 1, 2, 3, 4 \dots L$$

Mathematical interpretation of the filter:

To implement the median filter, the filter must be split up into numerous samples. In these samples with a window size L (as shown above), values are replaced to the median value of the entire sample. For reference, the median value of the sample is calculated as follows:

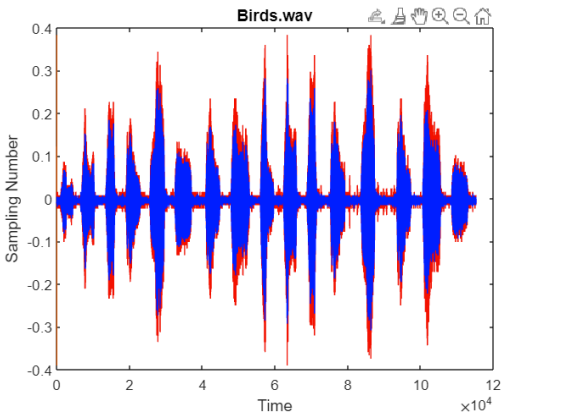
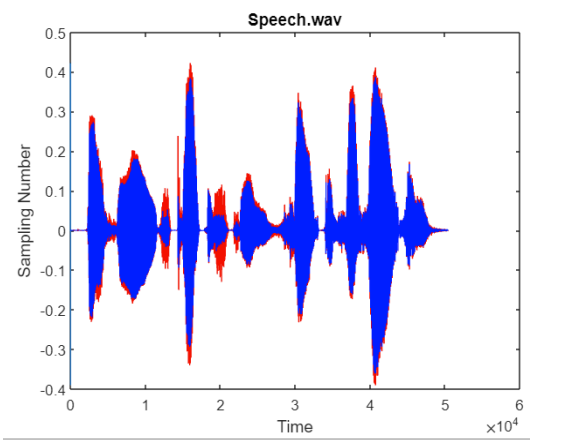
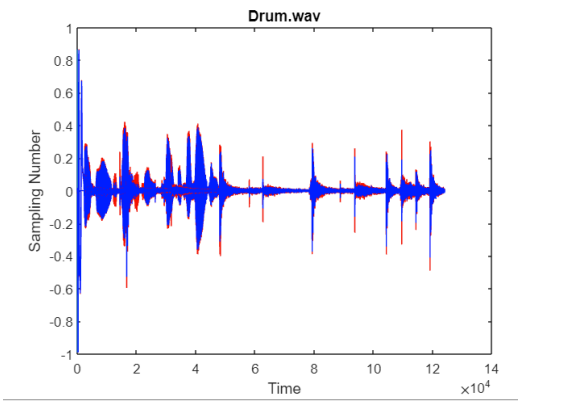
$$\begin{aligned} [1] \quad m &= x_k; \quad k = \frac{(n+1)}{2} \quad (n \text{ odd}) \\ [2] \quad m &= \frac{(x_k + x_l)}{2}; \quad k = n/2; l = (n/2) + 1 \quad (n \text{ even}) \end{aligned}$$

Where n represents the window size of the filter. After this step is complete, the next sample is used to repeat the same procedure.



To use the median filter, the graphical representation above displays our goal as each sample is measured, where k is the current window size. It is important to see that the windows only change all the values in the window, and then move to the next window which may contain the changed values. Referring to the above figure, the $(k-1)$ window starts off with the set of values, while the k th window still includes the majority of the values in the $(k-1)$ window. After all the samples, the result is a truncated filter.

Plot:

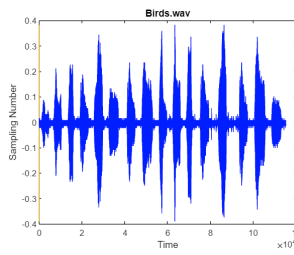
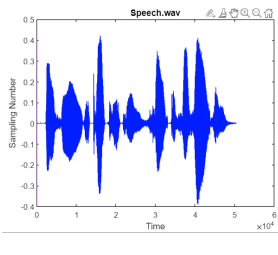
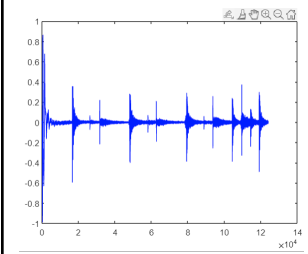
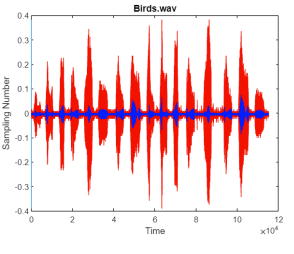
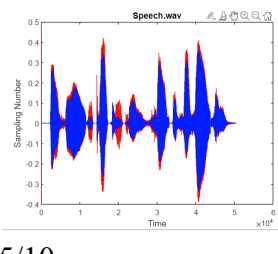
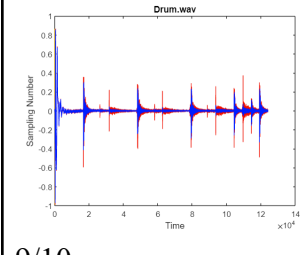
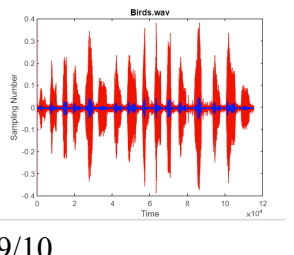
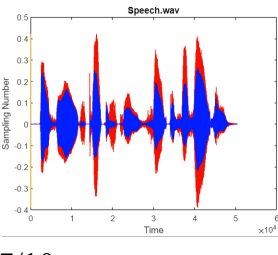
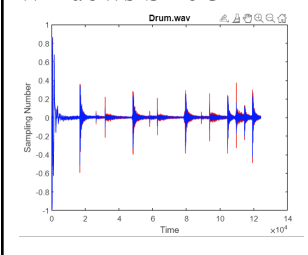
Sound File	Plot
Birds.wav - Window Size = 2	 <p>The plot shows the waveform of the 'Birds.wav' file. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.4. The x-axis is labeled 'Time' and ranges from 0 to 12, with a multiplier of $\times 10^4$. The signal is a complex, periodic waveform with multiple peaks and troughs, colored in blue and red.</p>
Speech.wav - Window Size = 3	 <p>The plot shows the waveform of the 'Speech.wav' file. The y-axis is labeled 'Sampling Number' and ranges from -0.4 to 0.5. The x-axis is labeled 'Time' and ranges from 0 to 6, with a multiplier of $\times 10^4$. The signal is a complex, periodic waveform with multiple peaks and troughs, colored in blue and red.</p>
Drums.wav - Window Size = 3	 <p>The plot shows the waveform of the 'Drums.wav' file. The y-axis is labeled 'Sampling Number' and ranges from -1 to 1. The x-axis is labeled 'Time' and ranges from 0 to 14, with a multiplier of $\times 10^4$. The signal is a complex, periodic waveform with multiple peaks and troughs, colored in blue and red.</p>

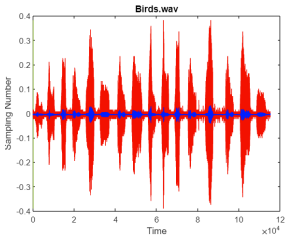
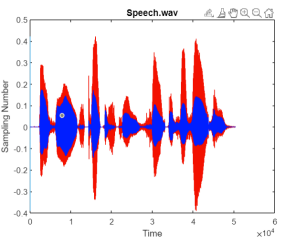
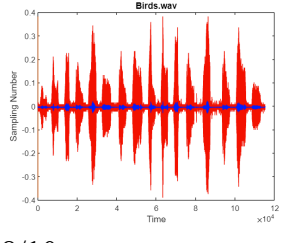
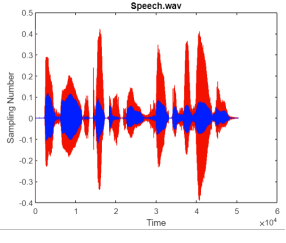
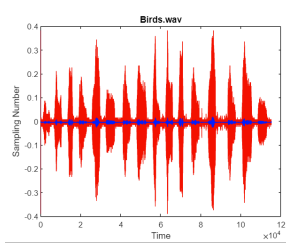
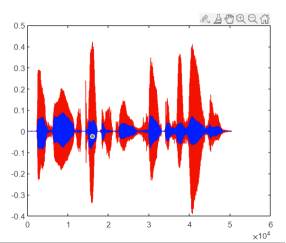
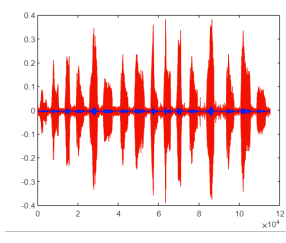
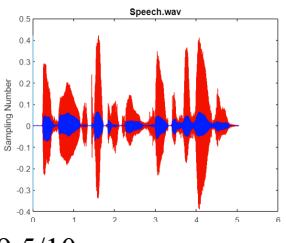
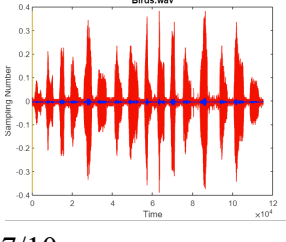
The Blue region represents the filtered sound and the red region is the original function.

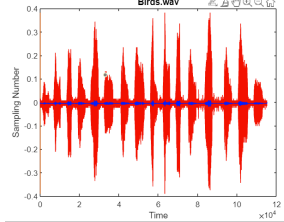
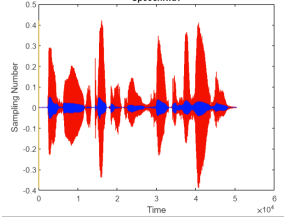
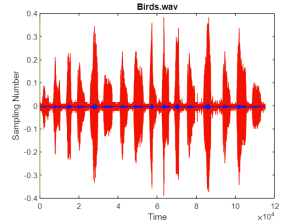
Questions:

1) Which filter and window size worked the best? Provide a full analysis.

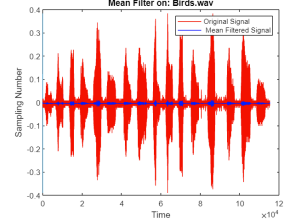
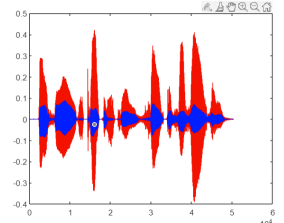
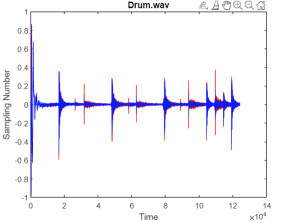
Based on the three audio files, a table was created showing the filter and window size used, as well as the graph data of each of the sound files. A full table was conducted for the mean (moving average) filter, and only the best window sizes for the other filters were recorded in the next table.

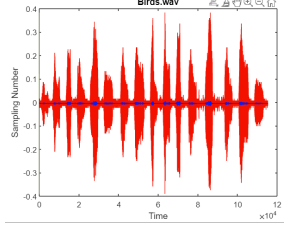
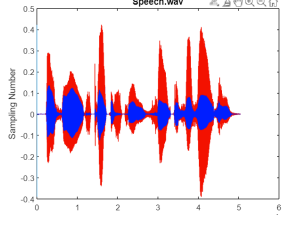
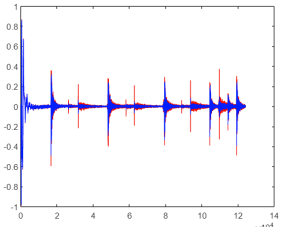
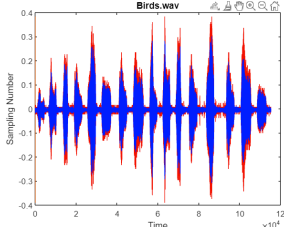
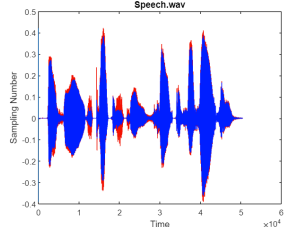
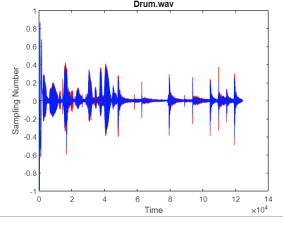
Window Size	Birds.wav Filtering	Speech.wav Filtering	Drum.wav Filtering
1	7/10 	3/10 	 3/10
5	9.5/10 	 5/10	Window Size 5  9/10
10	 9/10	 7/10	Windows Size 3 

15	 <p>Birds.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>8.5/10</p>	 <p>Speech.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>8/10</p>	
20	 <p>Birds.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>8/10</p>	 <p>Speech.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>9/10</p>	
25	 <p>Birds.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>8/10</p>	 <p>Speech.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>9.5/10</p>	
30	 <p>Birds.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>7.5/10</p>	 <p>Speech.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>9.5/10</p>	
35	 <p>Birds.wav</p> <p>Sampling Number</p> <p>Time $\times 10^4$</p> <p>7/10</p>		

40	 7/10	 9/10 (bit too filtered, sounds muffled)	
45	 10/10		

The respective audio files were given a rating out of 10 in terms of both quality and background filtering of the sound. The table below is a summary of best window sizes of each of the audio files, given a selected filter.

Window Size	Birds.wav	Speech.Wav	Drums.Wav
Mean Filter	<p>3</p> <p>At a window size of 3 most of the background noise was removed, however an increase in windows size after caused a large loss of information.</p> 	<p>25</p> <p>All background noise was removed from the audio, and further increase in window size muffled the speaker, hindering the sound quality.</p> 	<p>3</p> <p>The drums audio had a very minimal background sound, so only a minimal window size of 3 was required to get a clear sound.</p> 
Gaussian Filter	40 (Still background noise)	20	4

	 <p>While the background noise got considerably lower, it was still audible; The window size could not be increased as the sound amplitude of the output wave would be too small to be heard.</p>	 <p>After a window size of 20, all background noise was removed, however the speaker became increasingly more muffled (more than the mean filter).</p>	 <p>As mentioned in the previous filter, the background noise in the drums was minimal so only a window size of 5 was needed to create a clear audio from the input.</p>
Median Filter	<p>2 (still a lot of background noise)</p>  <p>At a window size 2, a lot of the background noise in the audio was still present. However, after any window size of 3 and above, there was a distinct distortion in the audio that affected the quality of the sound, likely due to the high peaks in the audio file</p>	<p>3 (No background noise)</p>  <p>At a window size of 3, any window size 4 and above caused a distinct distortion impacting the quality of the audio.</p>	<p>3 (No background noise)</p>  <p>At a window size of 3, any window size 4 and above caused a distinct distortion impacting the quality of the audio.</p>

Based on this data, an analysis was conducted on each of the filters.

Mean (moving average) Filter:

The mean filter was able to filter the background noise in the audio files very well, with minimal reduction in the quality of the audio. In the later window sizes, however, the amplitude of the sounds started to decrease into an almost inaudible range.

1. Birds.wav: Window size 3

The background sound of the birds file was almost completely removed, but a larger window size of 3 significantly decreased the amplitude of the audio; it was concluded that a window size of 3 would be the best.

2. Speech.wav: Window Size 25

At a window size of 25, the background noise of the audio was completely removed to produce the clearest sound. Past this point, the speaker became increasingly more muffled so a window size of 25 was perfect to filter out this audio.

3. Drum.wav: Window 3

A window size of 3 was needed to clear out the background noise, with no impact on the quality of the audio file.

Gaussian(weighted average) Filter:

The gaussian filter cut a large part of the background noise in the audio files, but the amplitude of the sound significantly decreased after the later window sizes.

1. Birds.wav: Window size 40

As mentioned in the table, window size could no longer be increased or the volume output would have been too small to hear the actual file.

2. Speech.wav: Window Size 20

While the background audio was almost all filtered out, a window size past 20 caused the audio to be muffled, hindering the quality of the sound.

3. Drum.wav: Window 5

Similar to the previous audios, distortion in the bass drum hits were noticeable after window size of 4.

Median Filter:

The median filter was initially able to remove a large part of the background sound, but sound quality quickly deteriorated after a certain window size threshold.

1. Birds.wav: Window size 2

The median filter was able to remove nearly all the background noise of the file, but a window size of 4 or more caused a distinct distortion in the sound. The audio transformed into a bubble-like audio, which cut out the original bird chirps.

2. Speech.wav: Window Size 3

While the background audio was almost all filtered out, the distortion in the audio was clear after a window size of 3 and up.

3. Drum.wav: Window 3

Similar to the previous audios, distortion in the bass drum hits were noticeable after a window size of 4.

Overall, the median filter was great at cutting out the background noise, but the distortion effect eliminated it from being the best filter to be used. The Gaussian, or weighted filter, had great potential as it blocked out the background noise of the sound very well. Although it was nearly blocked out, the amplitude of the sound significantly decreased, hindering the quality of the sound as the window size increased. Finally, the mean (moving average) filter had a similar issue to the Gaussian Filter in terms of the volume of the sound decreasing. However, it was able to completely block out the background noise in most cases (Speech.wav, Drums.wav) before reaching this low amplitude in sound. As for the window size, the most common selection range that produced the best results was around 3, partly due to sound quality issues that have risen past this window size.

In conclusion, the mean filter was able to reach the overall best sound in terms of both reducing background noise and preserving the quality of the audios. However, it doesn't come without its drawbacks, as the decrease in amplitude of the sound is still an issue when using this filter. It is ideal to combine multiple filters at once in order to produce the best result, while keeping the sound quality intact.

2) Describe the moving average filter as an appropriate impulse response function in the discrete and continuous domain.

In the discrete domain:

A convolution needs to be used to find the impulse response function. The impulse response of

the moving average filter would be an impulse signal (dirac function) and needs to be applied.

Figure 4 demonstrates this below:

Figure 4

Moving Average Filter

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[n-k], \quad L = \text{window size}$$

A dirac function: $\delta[n]$ needs to be applied:

$$x[n] = \delta[n]$$

Resulting in:

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} \delta[n-k]$$

Therefore, to describe the moving average function as an impulse response:

$$y[n] = x[n] * y'[n]$$

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[k] \delta[n-k]$$

In Continuous domain:

Same process occurs in the continuous domain, where we would take the integral over the interval. Figure 5 demonstrates this below:

Figure 5

Continuous time

$$y(t) = x(t) * y'(t)$$

$$y'(t) = \frac{1}{L} \int_0^{L-1} \delta(t - \tau) d\tau$$

$$y(t) = \frac{1}{L} \int_0^{L-1} x(\tau) \delta(t - \tau) d\tau$$

3) Using Eq.(1) how would you define a high pass filter? provide only one mathematical definition and explain what it does in one example as I described each filter above.

A high pass filter is defined by subtracting the original input signal, $x[n]$, by the low pass filter, $y[n]$. This gives the following equation, where $h[n]$ is the high pass filter: $h[n] = x[n] - y[n]$. A low pass filter's purpose is to preserve the lower frequencies so if you subtract the lower frequencies from the original input signal, the higher frequencies would remain. Figure 3 shows the mathematical definition of a high pass filter using Eq.(1).

Figure 4

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} b_k x[n-k]$$

$$h[n] = x[n] - y[n]$$

$$h[n] = x[n] - \frac{1}{L} \sum_{k=0}^{L-1} b_k x[n-k]$$

So with a window size of
3 for example:

$$h[n] = x[n] - \left[\frac{1}{3} (x[n] + x[n-1] + x[n-2]) \right]$$

$$= x[n] - \frac{1}{3} x[n] - \frac{1}{3} x[n-1] - \frac{1}{3} x[n-2]$$

$$h[n] = \frac{2}{3} x[n] - \frac{1}{3} x[n-1] - \frac{1}{3} x[n-2]$$

↳ high pass filter with window size, 3

3.3 Analyzing Signals

3.3.1 Syllable Classification

To retrieve the syllables from the Speech.wav audio, further classification is needed to define what a syllable is in an audio signal. A syllable is when there is a region where there is a silent region between two peaks in the audio. Knowing this, the algorithm goes as follows: first, filter the signal using a median filter with window size 3. Doing this ensures any excess noise is removed from the audio. Next, the audio is sectioned into frames with a frame duration of 0.045 seconds. This ensures that the amplitude average of the sound can be taken in each frame and

see if it's over a certain threshold that is set as 0.015dB. By taking the average amplitude of each frame, if that value was over the threshold then it is a syllable. But, to figure out if it was a new syllable or the same one, a boolean flag was used to see whether the previous frame already identified a syllable. If the previous frame wasn't a syllable, then it is deducted the current frame is a new syllable that increments the total syllable count in the audio clip. Once the algorithm finished the total syllable count was outputted.

3.3.2 Finding Beats Per Minute

To determine the beats per minute in the Drum.wav audio clip, the algorithm used to retrieve the syllables from 3.3.1 was used. The syllables in the Drum.wav correspond to a beat. As such, that algorithm was used to determine the number of beats within the Drum.wav clip. Once this was found, the beats per second was determined by dividing the total beats by the time length of the audio (Length of signal divided by the sampling rate). Beats per minute is just beats per second multiplied by 60 seconds so, with a quick multiplication, the beats per minute for the Drum.wav clip was found.

3.3.3 Silence Regions

To determine the silent regions in the Birds.wav audio clip, a similar process must be followed to determine the syllables by dividing the audio into different frames. Max values of the amplitude are determined in the frame and if this max amplitude is higher than a threshold (threshold determines a region that is not silent) of 0.02dB, it is set to 0. If it is less than the threshold, then it is a silent region and its original value is kept. By doing this for the entire

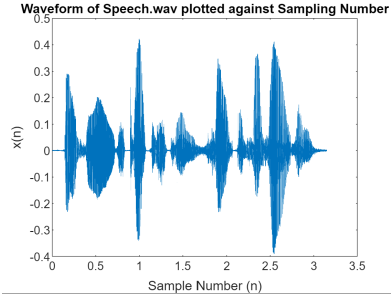
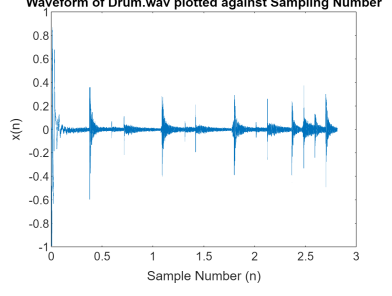
signal, the silent part of the signal will be determined and can be plotted over the original signal to see which regions are silent.

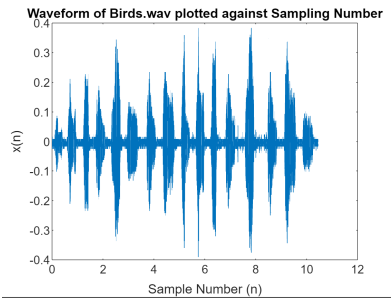
4 Results and discussion

4.1 Part 1 Results

After converting each sound to Mono and determining the sampling rate, the plots for each sound can be seen in table 1. Speech.wav already had a sampling rate of 16KHz so it didn't get affected by the resample. Drum.wav had an original sampling rate of 44.1KHz so it got downsampled to 16KHz. Birds.wav got upsampled from its original sampling rate of 11.025KHz.

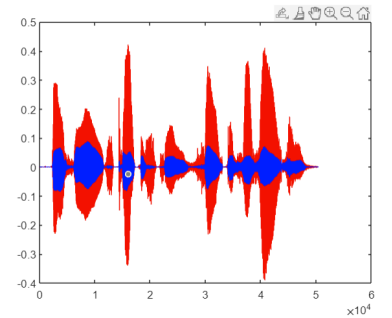
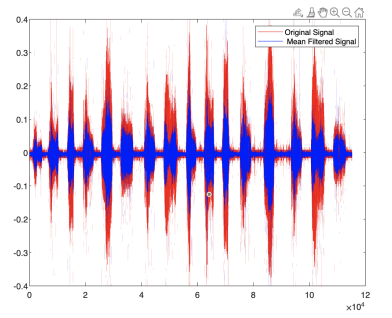
Table 3:

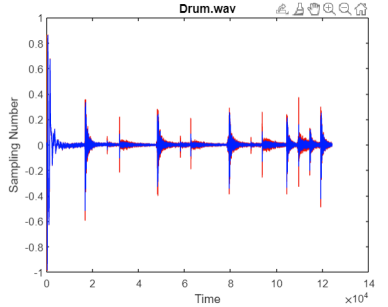
Sound	Sampling Rate (Hz)	Plot
Speech.wav	16,000	
Drum.wav	44,100	

Birds.wav	11,025	
-----------	--------	---

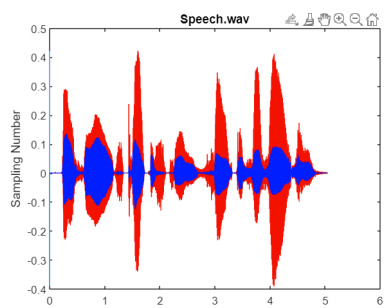
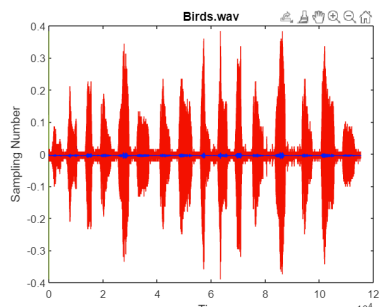
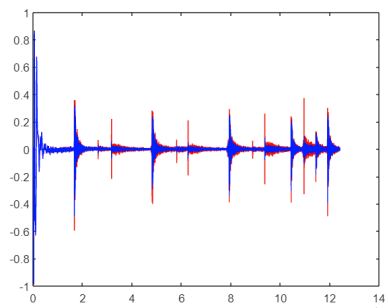
4.2 Part 2 Results

The moving average filter applied to each of the sound clips is shown in the following table 4:

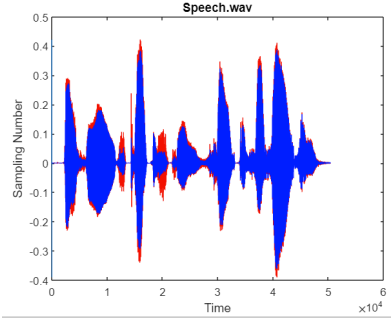
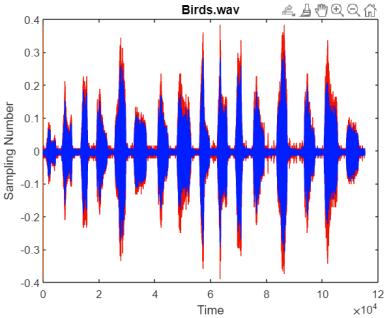
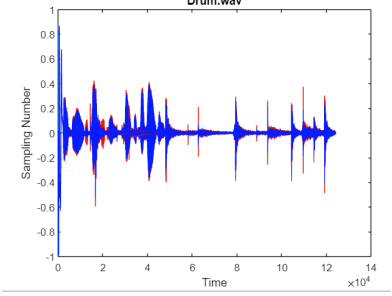
Sound	Window Size	Plot
Speech.wav	25	
Birds.wav	3	

Drum.wav	3	
----------	---	---

The gaussian filter applied to each sound clip is shown in the table 5 below:

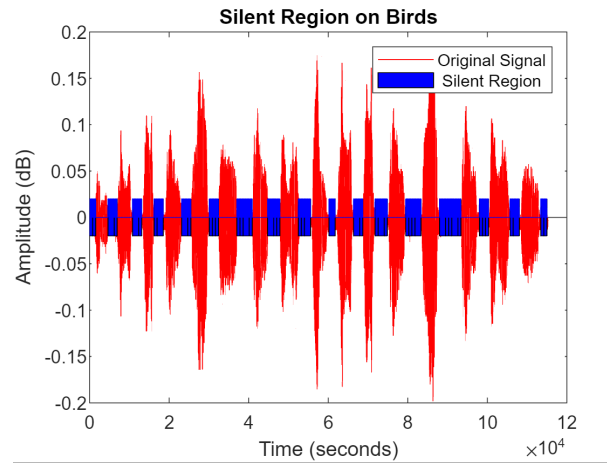
Sound	Window Size	Plot
Speech.wav	20	
Birds.wav	40	
Drum.wav	4	

The Median filter applied to each sound clip is shown in the table 6 below:

Sound	Window Size	Plot
Speech.wav	3	
Birds.wav	2	
Drum.wav	3	

4.3 Part 3 Results

Table 7 depicts the results for part 3. For determining the syllables,

Question	Results
Number of syllables for the Speech.wav clip	10 syllables
Beats per Minute of the Drum.wav clip	170.60 Beats Per Minute
Silent region of the Birds.wav clip	

Conclusions and recommendations

After going through the process in designing and evaluating various different filters, it can be said that these filters are decent at serving their purpose; however, future modification and improvements can be implemented to get a better result. In order to decide which filter would be best for usage, it is important to identify what has to be done to the sound.

If background noise has to be eliminated, then either the moving average filter or median filter should be used. However, if high overshoot of the signal is needed to be controlled, the gaussian filter would work best. However, since our goal was to find the filter that removes background noise the best and maintain sound quality, The gaussian filter was not heavily considered as the best filter. Comparatively, the median filter is better at removing the background noise at the cost of decreasing sound quality. The mean average filter can maintain sound quality, but its ability to remove background noise isn't as potent as a median filter. There seems to be a major tradeoff between sound quality and background noise for both the filters.

The window size that worked the best was a window size of 3. This window size was able to remove a large majority of the background noise when applied to the mean average filter and the median filter. The filter that worked best was the mean filter since it was able to remove background noise better than median filter; however, the sound quality was taken into account since it had decreased. There was a good level of sound quality in the median filter at the cost of a good amount of background noise, which made it hard to hear the audio.

Although the sound quality decreased, it wasn't as hard to hear the clip with the mean filter due to its expertise in removing background noise. Therefore, the tradeoff of background noise to sound quality was far less on the median filter compared to the moving average filter. From this analysis, the mean filter was decided as the best filter.

There were various ways to improve the process of this project. such as finding better methods to test the filters themselves. Since the clips that were provided weren't the most ideal, it was hard to recognize if the clips had high amounts of background noise or if there were high pitches in its sound. If a clip with a large amount of background noise was given with a goal on clarity, choosing which filter performed the best would have been much easier. The audio clips presented already had a degree of clarity to them, so there were no drastic improvements that came from using any of the filters.

Another suggestion to improve certain areas would be to use a combination of filters to produce the best possible sound. Since the gaussian filter prevents the overshoot in an audio file, it can be combined with the moving average filter or the median filter to get a clearer sound, as they can tune out different parts of the background noise. Implementing this theory in coding may be difficult, but can be rewarding in order to bring out the best of both filters.

In terms of coding not sure how effective this will be but in theory this seems like a great way to modify sounds by applying multiple filters on the same signal. For example, if the original signal is given the signal can be modified with the moving average filter and then applying the gaussian filter on the already filtered signal.

Acknowledgement

The authors are particularly grateful to Dr. Charbel Azzi for his insights on signals and systems into the nature of applying signal filtering using convolutions.

References

This is the last section of the report, prior to any appendices. The references should not be double-spaced, but single-spaced. For a technical report, use the CSE style.

- [1] Reference 1 information.
- [2] Reference 2 information.
- [3] Reference 3 information.

[1] https://www.researchgate.net/publication/275421821_Review_Fifty_Years_Plus_of_Accelerometer_History_for_Shock_and_Vibration_1940-1996

[2] <https://www.livescience.com/40102-accelerometers.html>

[3] <https://www.omega.ca/en/resources/accelerometers>

Methodology:

[4]: <https://codemonk.in/blog/moving-average-filter/>

[5] https://en.wikipedia.org/wiki/Gaussian_filter

Appendix

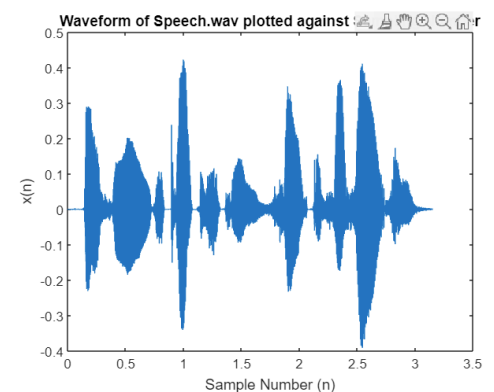
Appendix A: Part 1 (Preparations)

To prepare each of the three audios, Birds.wav, Speech.wav, and Drums.wav, the function call is commented out for each selected signal, with the default function call preparing the Speech.wav file. The user can uncomment out the function call based on their selected audio.

```
% Part 1
load handel.mat
sound_preperation("Speech.wav")
%sound_preperation("Drum.wav")
%sound_preperation("Birds.wav")
% this is sampling rate for part 1
function sound_preperation(filename)
    [y,Fs] = audioread(filename);
    % 1.1: finding the sampling rate
    disp("The sampling rate for " + filename + " is: " + Fs);
    % 1.2: If sound is stereo (n > 1), it will convert it to mono
    [m, n] = size(y);
    if n > 1
        y = sum(y, 2) / size(y, 2);
    end
    %1.3 Play the sound from the sound files
    sound(y,Fs);
    %1.4
    audiowrite("new" + filename,y,Fs);
    %1.5
    [newY, newFs] = audioread("new" + filename);
    N = length(newY); % N is the number of samples
    dt = 1/newFs;
    time=(0:dt:(N*dt)-dt); % Time vector on x-axis
    plot(time,newY);xlabel('Sample Number (n)');ylabel('x(n)');title("Waveform of " + filename + " plotted against Sampling
    Number");
    %1.6 % downsample to 16KHz

    new_sample = resample(newY, 16000, Fs);
    sound(new_sample);
    pause(4);
end
```

Sample soft copy of output using Speech.wav:



Appendix B: Part 2 (Filters)

Function calls are all initially commented out, with the default function call on the mean filter to the Birds.wav file. Based on the user's discretion, they can select which audio file they would like to play and which specific filter they would like to use. Simply look for the associated function call below and uncomment the line.

```
% Part 2
load handel.mat
% PART 2 FUNCTION CALLS
meanFilter("Birds.wav")
clf
%pause(7)
%meanFilter("Speech.wav")
clf
%pause(7)
%meanFilter("Drum.wav")
clf
%pause(7)
%gaussianFilter("Birds.wav")
clf
%pause(7)
%gaussianFilter("Speech.wav")
clf
%pause(7)
%gaussianFilter("Drum.wav")
clf
%pause(7)
%medianFilter("Birds.wav")
clf
%pause(7)
%medianFilter("Speech.wav")
clf
%pause(7)
%medianFilter("Drum.wav")
clf
%pause(7)
function [data, Fs] = meanFilter(filename)
    [y,Fs] = audioread(filename);
    sound(y, Fs);
    [m, n] = size(y);
    % convert to Mono if its stereo
    if n > 1
        y = sum(y, 2) / size(y, 2);
    end
    % get the correct window size for the file inputted
    if filename == "Birds.wav"
        windowSize = 3;
    elseif filename == "Speech.wav"
        windowSize = 25;
    else
        windowSize = 3;
    end
    %Filtering using convolution
    meanFilteredSignal = conv(y, ones(1,windowSize)/windowSize, 'full');
    %Plotting
    N = length(y); % N is the number of samples
    dt = 1/Fs;
    time=(0:dt:(N*dt)-dt); % Time vector on x-axis
    plot(time,y);xlabel('Time');ylabel('Sampling Number');title('Mean Filter on: ' +filename);
    % we will have a if statement here to have different window size for
    % diff files
    figure(1)
    plot(y, "r")
    hold on;
    plot(meanFilteredSignal, "b")
    legend(["Original Signal", " Mean Filtered Signal"]);
```

```

    pause(10);
    sound(meanFilteredSignal, Fs);
    % return the values
    data = meanFilteredSignal;
    Fs1 = Fs;
end
function [data, Fs1] = gaussianFilter(filename)
    [y,Fs] = audioread(filename);
    sound(y, Fs)
    % Convert to mono
    [m, n] = size(y);
    if n > 1
        y = sum(y, 2) / size(y, 2);
    end
    %Window Size configuration
    if filename == "Birds.wav"
        windowSize = 40;
    elseif filename == "Speech.wav"
        windowSize = 20;
    else
        windowSize = 4;
    end
    %Filtering
    weight = gausswin(windowSize, 1);
    weight = weight/sum(weight);
    gaussianFiltered = filter(weight, 1, y);
    % Plotting
    N = length(y); % N is the number of samples
    dt = 1/Fs;
    time=(0:dt:(N*dt)-dt); % Time vector on x-axis
    plot(time,y);xlabel('Time');ylabel('Sampling Number');title('Gaussian Filter on: ' + filename);

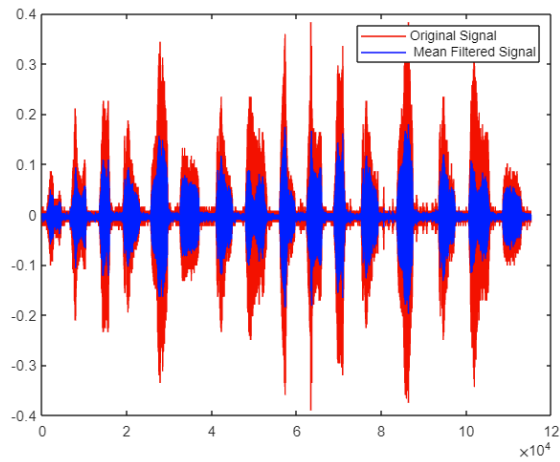
    plot(y, "r")
    hold on;
    plot(gaussianFiltered, "b")
    legend(["Original Signal", " Gaussian Filtered Signal"])
    pause(5)
    sound(gaussianFiltered, Fs)

    % return values
    data = gaussianFiltered;
    Fs1 = Fs;
end
function [data, Fs1] = medianFilter(filename)
    [y,Fs] = audioread(filename);
    sound(y, Fs);
    % Convert to mono
    [m, n] = size(y);
    if n > 1
        y = sum(y, 2) / size(y, 2);
    end
    % Window Size configuration
    if filename == "Birds.wav"
        windowSize = 2;
    elseif filename == "Speech.wav"
        windowSize = 3;
    else
        windowSize = 3;
    end
    % Applying median filter
    medianFilteredSignal = medfilt1(y,windowSize);
    % Plotting
    N = length(y); % N is the number of samples
    dt = 1/Fs;
    time=(0:dt:(N*dt)-dt); % Time vector on x-axis
    plot(time,y);xlabel('Time');ylabel('Sampling Number');title("Median Filter on: " + filename);
    plot(y, "r");
    hold on;
    plot(medianFilteredSignal, "b");
    legend(["Original Signal", " Median Filtered Signal"]);
    pause(5);
    sound(medianFilteredSignal, Fs);
    % Return values
    data = medianFilteredSignal;
    Fs1 = Fs;
end

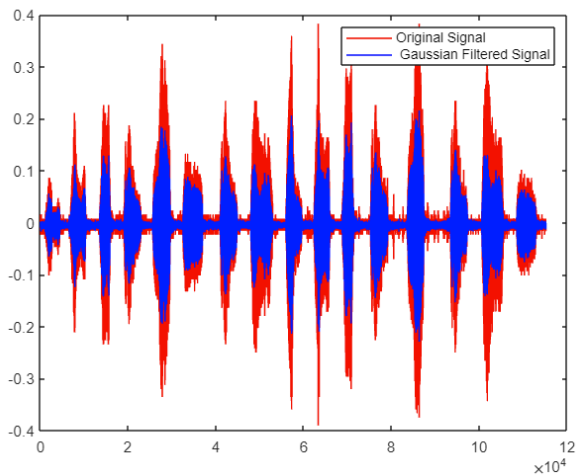
```

Sample soft copy of output using Birds.wav:

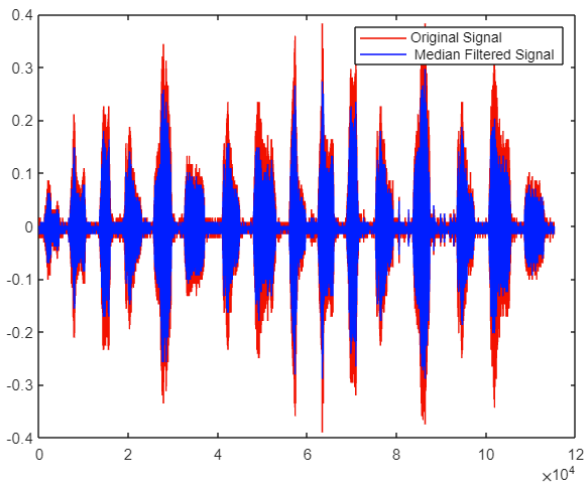
Mean Filter (window size 3):



Gaussian Filter (window size 3):



Median Filter (window size 3):



Appendix C: Part 3 (Analyzing Signals)

All function calls are commented out, where the default function call is the number of syllables in the Speech.wav file. The user can uncomment the the function based on which part the user would like to run (either the bpm for the Drums.wav file, or the Silent region of Birds.wav file)

```
% PART 3 FUNCTION CALLS
clf
numberOfSyllables("Speech.wav")
%pause(7)
clf
numberOfBeats("Drum.wav")
%pause(7)
clf
%silentRegionBirds()
%pause(7)
%----PART 3 CODE BELOW -----
function [data, Fs1] = meanFilter(filename)
    [y,Fs] = audioread(filename);
    sound(y, Fs);
    [m, n] = size(y);
    % convert to Mono if its stereo
    if n > 1
        y = sum(y, 2) / size(y, 2);
    end
    % get the correct window size for the file inputted
    if filename == "Birds.wav"
        windowSize = 3;
    elseif filename == "Speech.wav"
        windowSize = 25;
    else
        windowSize = 3;
    end
    %Filtering using convolution
    meanFilteredSignal = conv(y, ones(1,windowSize)/windowSize, 'full');
    %Plotting
    N = length(y); % N is the number of samples
    dt = 1/Fs;
    time=(0:dt:(N*dt)-dt); % Time vector on x-axis
    plot(time,y);xlabel('Time');ylabel('Sampling Number');title( "Mean Filter on: " +filename);
    % we will have a if statement here to have different windowSize for
    % diff files
    figure(1)
    plot(y, "r")
    hold on;
    plot(meanFilteredSignal, "b")
    legend(["Original Signal", " Mean Filtered Signal"]);
    pause(10);
    sound(meanFilteredSignal, Fs);
    % return the values
    data = meanFilteredSignal;
    Fs1 = Fs;
end
function numOfSyllables = numberOfSyllables(filename)
    [y, Fs] = audioread("Speech.wav");
    %filter signal
    windowSize = 3;
    y = medfilt1(y, windowSize);
    %Breaking the sound into frames
    N = length(y);
    frameDuration = 0.045;
    lengthOfFrame = frameDuration*Fs;
    totalFrames = floor(N/lengthOfFrame); %Round to integer value
    numOfSyllables = 0;
    isSyllable = false;
    %Go through all frames within signal and determine the amplitude in that
    %region
    for i=1:totalFrames

        %Every time we go through the loop, we extract the values from each
        %frame in the signal
        startingPoint = (i-1)*lengthOfFrame + 1;
        endingPoint = lengthOfFrame * i;
        frame = abs(y(startingPoint: endingPoint)); %
        meanAmplitude = mean(frame);
    end
end
```



```

    %We want the average amplitude of the frame to be higher than the
    %minimum threshold amplitude to be considered a syllable
    if meanAmplitude > 0.015
        %
        %We only want new syllables to be added. Thus, if the previous
        %frame is not a syllable then we know it's a new one
        if isSyllable == false
            numOfSyllables = numOfSyllables + 1;
            isSyllable = true;
        end
    else
        isSyllable = false;
    end
end
disp("The number of Syllables for Speech.wav is: " + numOfSyllables);
end
function numberOfBeats(filename)
    [y, Fs] = audioread(filename);
    windowSize = 3;
    %filter data
    y = medfilt1(y, windowSize);
    N = length(y); % N is the number of samples
    %The beats correspond to the number of syllables in the drum.wav file
    beats = numberOfSyllables("Drum.wav");
    %Determining the beats per second in the signal
    time_len = N/Fs;
    beatsPerSecond = beats/time_len;
    %Converting from beats per second to beats per minute
    bpm = beatsPerSecond * 60;
    disp("The beats per min for Drum.wav is: " + bpm);
end
function silentRegionBirds()
    %disp(meanFilter("Birds.wav"));
    [y, Fs] = meanFilter("Birds.wav");
    original_signal = y;
    y = abs(y);
    frame_duration = 0.07;
    frame_len = frame_duration*Fs;
    N = length(y);
    num_frames = floor(N/frame_len);
    SilentSignal = zeros(N,1);
    count = 0;
    for k = 1 : num_frames
        frame = y((k-1)*frame_len+1 : frame_len*k);
        mean_value = max(frame);
        % this frame is not silent
        if (mean_value < 0.03)
            SilentSignal((k-1)*frame_len+1: frame_len * k) = 0.02;
        else
            count = count + 1;
            SilentSignal((k-1)*frame_len+1: frame_len * k) = 0;
        end
    end
end
% Plotting
figure(3)
dt = 1/Fs;
time=(0:dt:(N*dt)-dt); % Time vector on x-axis
plot(time,y);
plot(original_signal, "r");xlabel('Time (seconds)');ylabel('Amplitude (dB)');title("Silent Region on Birds");
hold on
positiveArea = area(SilentSignal);
positiveArea.FaceColor = 'Blue';
negativeArea = area (-SilentSignal);
negativeArea.FaceColor = 'Blue';
plot(SilentSignal, "b")
legend(["Original Signal", " Silent Region"])
end

```

Sample Output of Beats per Minute of Drums.wav:

```
The beats per min for Drum.wav is: 213.2478
>>
```

Sample Output of Number of Syllables in Speech.wav:

```
The number of Syllables for Speech.wav is: 10
```

```
ans =
```

```
10
```

Sample output of Silent Region of Birds.wav:

