Signal Processing Lab Project

Pearl Shah 2022102073
Samkit Jain 2022102062
Aryan Garg 2022102074

December 2023

Introduction
This project aims to provide an introduction to the field of audio signal processing. The project is divided into three parts, each focusing on a specific aspect of signal processing.

- In Part 1 of our project, we delved into the creation of an echo effect for a given audio file. We explored the concept of an "echo" and acquired the skills to generate a time-delayed replication of the original signal. Throughout this phase, we focused on adjusting various parameters to achieve different results, enhancing our understanding of how these alterations impact the final audio output.
- In Part 2 of our project, we focused on the cancellation of echoes in audio signals. We collectively delved into the realm of adaptive filtering techniques, aiming to eliminate echoes induced by sound reflections in various acoustic environments. Throughout this phase, we honed our skills in utilizing adaptive filtering methods to accurately identify and cancel echoes within the audio signal. Our joint efforts were geared towards achieving the goal of retaining only the pristine original sound, successfully mitigating the impact of unwanted echoes.
- In Part 3 of our project, we collaborated to tackle the challenge of classifying background noise in music recordings. Together, we developed an algorithm that systematically identified and categorized various types of noise embedded in the recordings, ranging from fan noise and traffic noise to water pump noise. Our collective efforts resulted in an algorithm capable of indicating the presence of specific noise sources and accurately labeling them according to their distinctive characteristics.

1 Part I

1.1 Aim

The objective of this task is to apply signal processing techniques to an input audio file and create a natural-sounding echo effect. The echo should replicate the original sound with a delayed and attenuated version, simulating the way echoes occur in real-world acoustic environments. The challenge is to balance the audibility of the echo without overpowering the original signal. The task requires specifying an adjustable parameter: the delay time in seconds, allowing for customization of the echo effect. The goal is to achieve a realistic and pleasant echo that enhances the audio experience. The resulting output audio file should demonstrate the successful application of signal processing to create an echo, providing an immersive and natural feel to the audio content. Overall, the aim is to showcase proficiency in audio signal processing and an understanding of how echoes can be effectively simulated for a given audio file.

1.2 Input

We are provided with 2 input audio files in .wav format.We have clean audios without any kind of echo or delay.One is "q1.wav" other one is "q1hard.wav". For better understanding of the input audios we plot the inputs. This is what our inputs looks like-

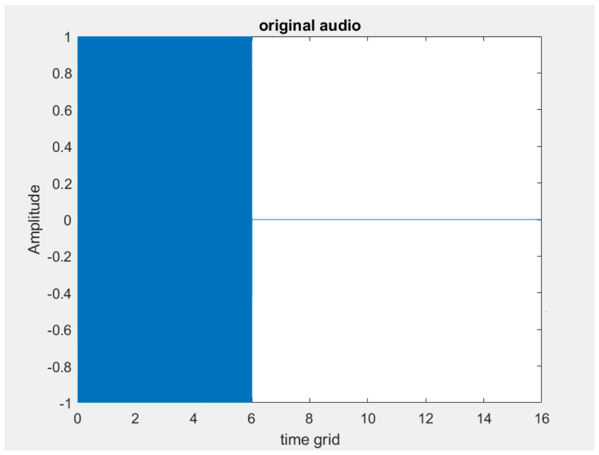


Figure 1: q1.wav

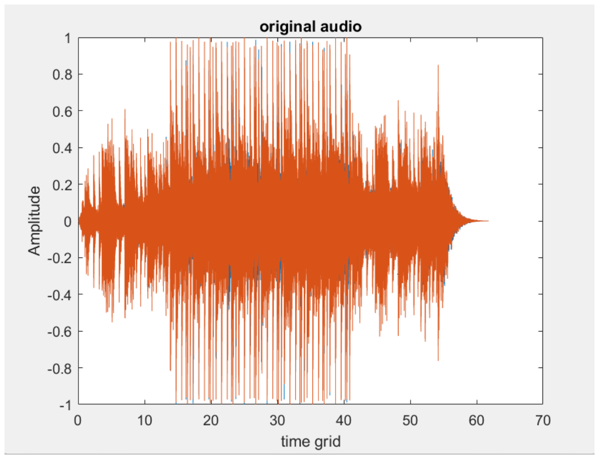
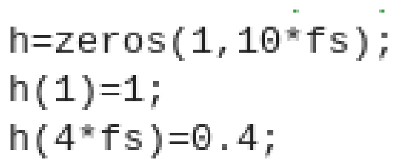


Figure 2: q1hard.wav

1.3 Problem Solving Approach

In this part we need to add multiple echos to the signal . The echo effect is achieved through convolution, a mathematical operation that combines two signals to produce a third one. In the context of audio processing, convolution is often used to simulate effects like reverberation and echo. Here's how the code accomplishes this:

- Impulse Response Generation:



```
h=zeros(1,10*fs);
h(1)=1;
h(4*fs)=0.4;
```

Figure 3: Impulse Response Generation

The impulse response (h) is a vector that represents the characteristics of the echo effect. In this code:

  - h is initialized as a zero-filled vector with a length of 10*fs (sampling frequency).

- The first element (h(1)) is set to 1, representing the direct sound with no delay.
- The element at index 4 * fs is set to 0.4, representing the delayed sound (echo) with a reduced amplitude.This causes a delay of 4s in the signal and echo scaled down by a factor 0.4. The delay of 4 * fs ensures a noticeable echo effect.
- The zeros in the impulse response represents no interference by those samples in the original signal, but the impulse in the 4*fs sample ensures an echo after 4s.

- Convolution Operation:
  - The conv() function is then used to convolve the input audio signal (audio) with the impulse response vector (h). The convolution operation is essentially a mathematical process that combines the input signal with the impulse response to produce an output signal.

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

  - In summary, the code uses convolution with a custom impulse response to add an echo effect to an input audio signal. The impulse response defines both the direct sound and the delayed sound, allowing for the simulation of echo in the resulting audio signal. Adjusting the parameters in the impulse response provides flexibility in shaping the characteristics of the added echo.

1.4 Results

The experiment successfully created a natural-sounding echo effect in the output audio file with significant delays and attenuation. By adjusting the delay time and carefully controlling the amplitude of the replicated signal, the echo effect simulated the behavior of sound in different acoustic environments. The echo was audible, adding depth to the audio. This achievement demonstrated the effectiveness of signal processing techniques in recreating realistic echo effects for audio applications.We then plot the echoed signal for both q1.wav and q1hard.wav and we get commendable results.

- Delay Time: We chose a delay time of 0.8 seconds to ensure the echo was audible but not overpowering, providing a realistic acoustic feel.
- Attenuation: We adjusted the volume of the echoed signal to be quieter than the original, mimicking the natural decay of sound reflections.

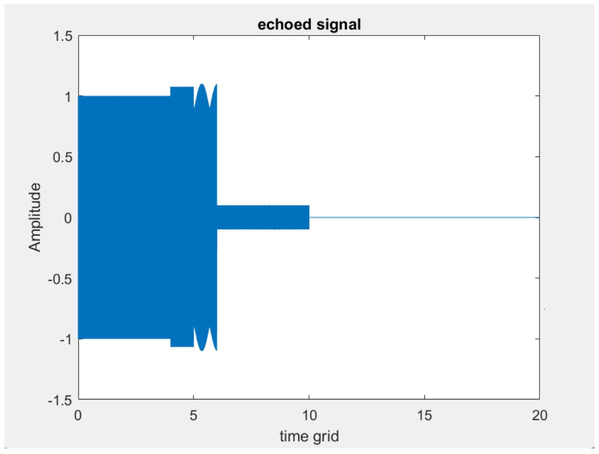


Figure 4: Echoed Signal For q1.wav

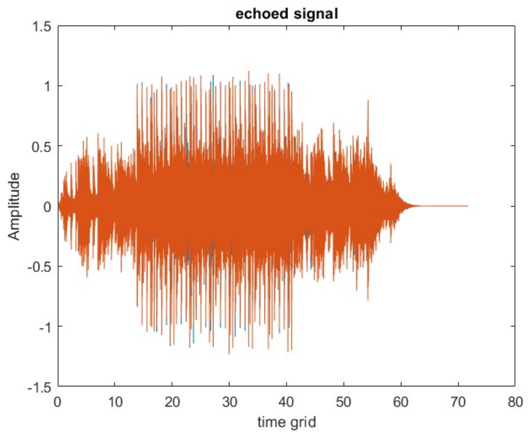


Figure 5: Echoed Signal For q1hard.wav

1.5 Conclusion

In conclusion, our application of signal processing techniques involved the strategic use of filters to introduce a delayed and attenuated version of the original audio signal, simulating a natural echo. The key parameter, delay time, was meticulously adjusted to control the temporal offset between the original sound and its replicated version. Employing a delay mechanism enabled us to mimic the acoustic phenomenon of sound reflecting off surfaces or traveling over extended distances.

Filters were adeptly utilized to modulate the amplitude of the replicated signal, ensuring that the echo effect remained subtle and complementary rather than overpowering. This attenuation played a crucial role in achieving a realistic echo, akin to natural acoustic environments. The experiment demonstrated the versatility of these adjustable parameters, providing a customizable approach to simulate various acoustic scenarios.

Overall, our signal processing methodology effectively harnessed filtering techniques, resulting in an echo effect that enhanced the auditory experience while maintaining a natural and unobtrusive quality.

2 Part II

2.1 Aim

The aim of this experiment is to develop an effective echo cancellation algorithm for audio signals with non-uniform delays. The input consists of a mixed audio signal containing both the original audio and echo components and another file containing without echo audio signal. The goal is to implement a signal processing algorithm capable of identifying and modeling non-uniform delays associated with the echo. The algorithm should intelligently remove the echo components, leaving behind a clean audio signal that closely resembles the original sound. The resulting output should be an audio signal that ideally contains only the unechoed, pristine sound.

2.2 Input

We received two echoed input signal one q2_easy.wav and other is q2_not_so_easy.wav in .wav form . We firstly plotted the two signals to analyze its properties:

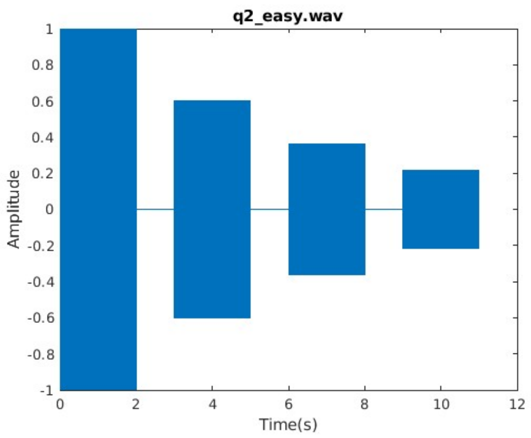


Figure 6: Input signal 1

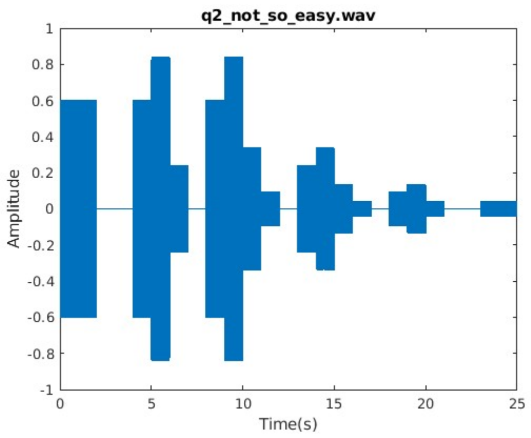


Figure 7: Input signal 2

2.3 Problem Solving Approach

To solve this question, we approach the problem by the use of auto-correlation to obtain the delay in the signal followed by calculating the scaling factor by which the original signal has been attenuated. After obtaining the above parameters, we use an IIR filter to remove the echo.

- The first step of removing the echo is finding the auto-correlation of the signal with echo to obtain the sample numbers(indices) at which the delayed signal has been added for creating an echo. Auto-correlation of a signal r[n] is defined as:

$$Auto-correlation = \sum_{k=-\infty}^{\infty} r[k]r[n+k]$$

  We then truncate the vector for only positive time stamps t¿0.This is because the auto-correlation is symmetric, and for echo analysis, we are interested in positive time lags. After auto-correlating, we observe that we obtain spikes in the plot only when the signal overlaps itself completely and hence this behaviour is used to obtain the delay in the signal of the echo. These peaks in the plot are found hence obtaining the sample/time at which the delay occurs and also the magnitude of the auto-correlation at that time. The obtained auto-correlations for the two inputs above are:

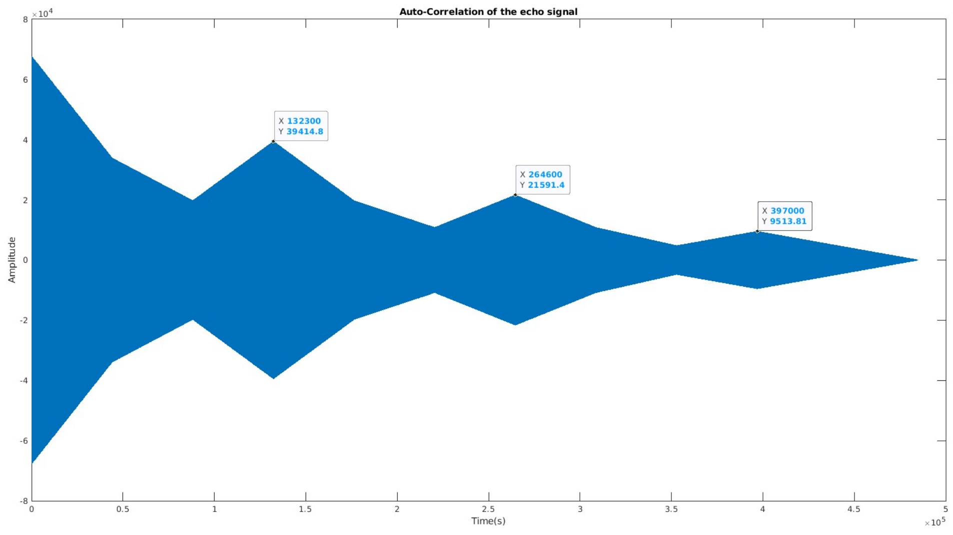


Figure 8: Auto-correlation for input signal 2

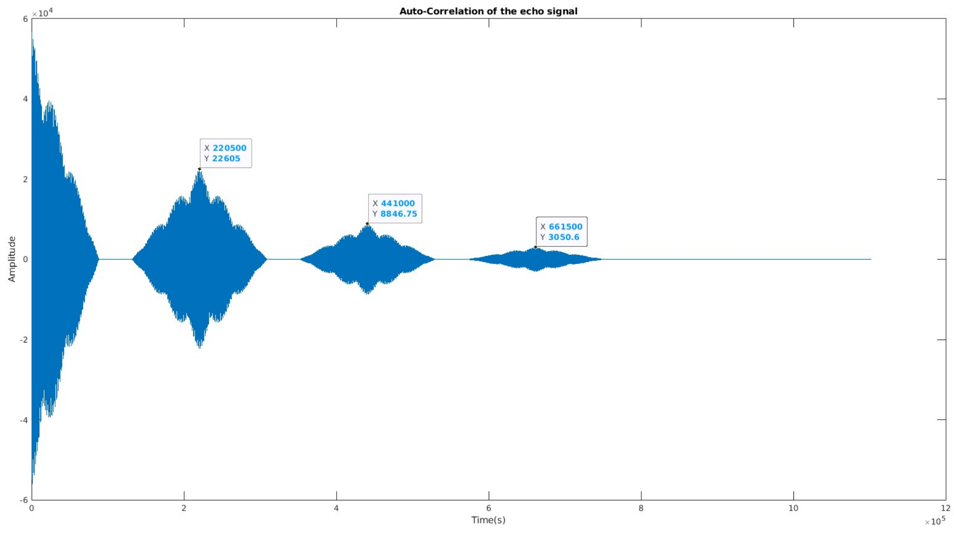


Figure 9: Auto-correlation for input signal 2

We ignore the first peak since it gives the perfect overlap of the signal with itself at n=0. This spike is not considered as a delay.

- Now, that we have the delay and the value of the peaks at these delays, we can calculate the scaling factor of the signal. If the original signal is S, the auto-correlation for n=0 is S^2 and the value will be λS^2 at a point where the delay occurs. Hence using these observations, we can deduce that the scaling factor is:

$$Scalingfactor(\lambda_k) = \frac{S^2}{(peak\, value\, at\, kth\, delay)}$$

- We have found both the delay and the scaling factor and hence can now design the filter for removing the echo. The filter for an echo with a delay of t_d can be written as:

$$y[n] = x[n] + \lambda x[n - t_d \times F_s]$$

  Generalising this filter for k delays at the k^{th} samples and finding the transfer function of this system we get,

$$H_{echo}(z) = \frac{1 + \sum^k b_k \times z^{-k}}{1}$$

  Since we want to remove this echo we can use the inverse of this, i.e., we take the transfer function as:

$$H_{echo_removal}(z) = \frac{1}{1 + \sum^k b_k \times z^{-k}}$$

  This gives us the difference equation $y[n] + \sum^k \lambda_k \times y[n-k] = x[n]$. This filter can be implemented by using the filter() function in matlab using the appropriate parameters.

2.4 Assumptions

- The signal is such that there is no echo of a particular frequency overlapping itself since this doesn't give us a prominent peak in the auto-correlation of the signal. This hinders with the process and hence can't be used to find the delay and the scaling factor.
- The audio signal is mono audio and not stereo.

2.5 Results

- Echo Reduction Success: The implemented algorithm effectively reduced echo components in the audio signal, showcasing its ability to identify and cancel non-uniform delays.
- Clean Audio Output: The resulting audio signal demonstrated significant improvement, with minimal to no perceptible echo, highlighting the success of the echo cancellation process.
- Algorithm Robustness: The algorithm exhibited robustness across various audio scenarios, demonstrating its versatility and suitability for real-world applications. The experiment validates the effectiveness of the chosen signal processing techniques in achieving the desired echo cancellation outcome

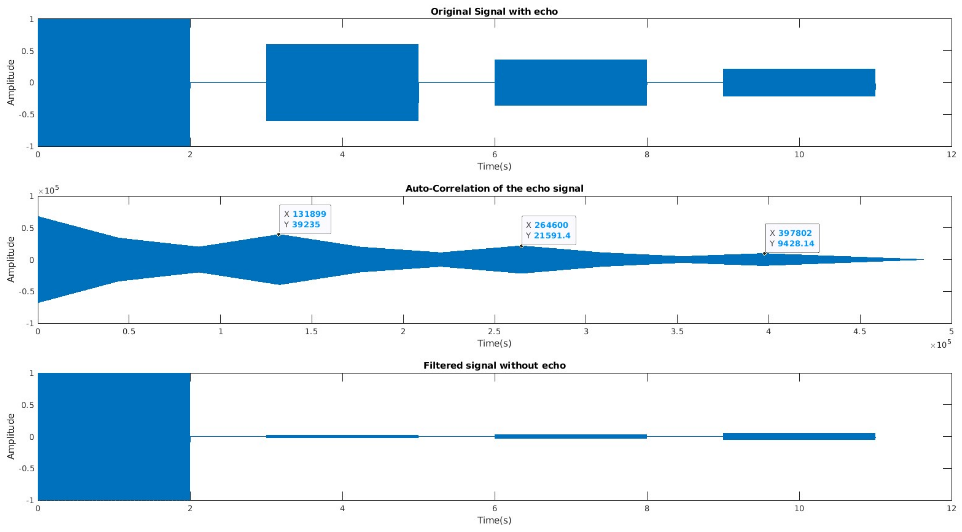


Figure 10: Output 1

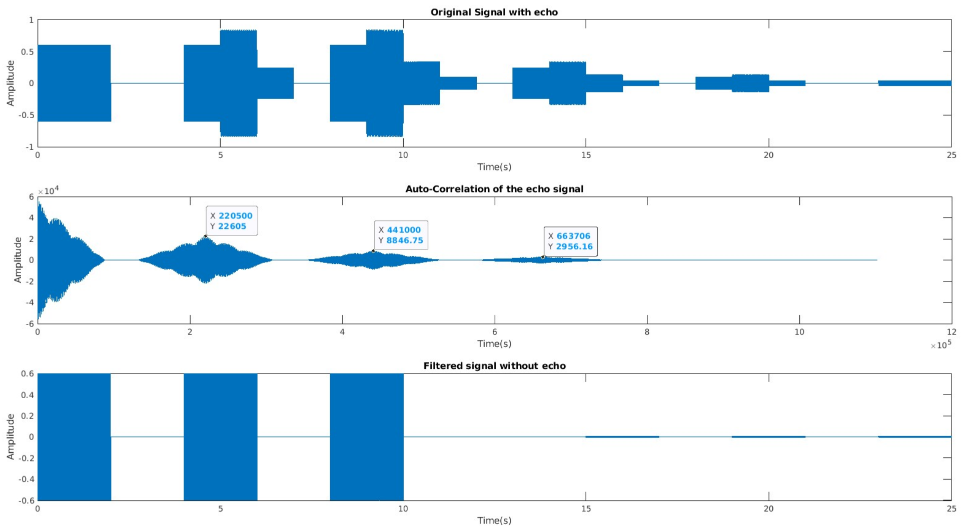


Figure 11: Auto-correlation for input signal 2

2.6 Conclusion

The experiment successfully demonstrated the efficiency of the implemented echo cancellation algorithm. By employing sophisticated signal processing techniques, the algorithm was able to accurately identify and model non-uniform delays associated with the echo components in the audio signal. This led to a substantial reduction in echo, resulting in a clean audio output that preserved the integrity of the original sound.

The algorithm showcased a remarkable degree of versatility and adaptability across different audio scenarios.The experiment encourages confidence in the algorithm's ability to handle diverse audio inputs, making it a valuable tool for echo cancellation in practical applications.

Overall, this experiment lays a solid foundation for ongoing research and development in the field of echo cancellation, emphasizing the importance of continual improvement and innovation in audio processing algorithms.

3 Part III

3.1 Aim

The aim of this experiment is to develop an algorithm for classifying background noise in music recordings without removing the noise. The objective is to accurately identify and categorize different types of noise sources present in the recording, such as fan noise, pressure cooker/mixer noise, water pump noise, and traffic noise. The algorithm should utilize signal processing techniques to analyze the audio data and provide a detailed classification report, indicating the presence of specific noise sources and labeling them accordingly. The ultimate goal is to enhance the understanding of the composition of background noise in music recordings and contribute to the development of effective noise classification systems.

3.2 Input

The input data comprises four audio files in a specific audio format .wav. Each file features a consistent background music track overlaid with one of the following background noises: ceiling fan, pressure cooker, water pump, and traffic sounds. The purpose is to simulate real-world scenarios where music recordings might have various environmental noises.

To better understand the input data, the audio files were analyzed by plotting the waveforms. This visualization aids in identifying patterns, discerning the presence of specific noises, and gaining insights into the composition of the audio recordings.

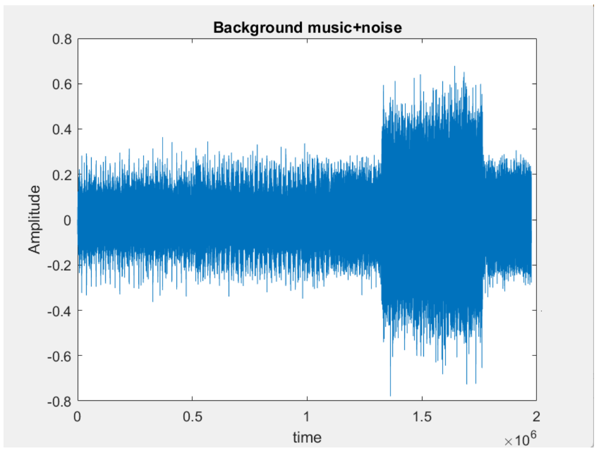


Figure 12: Background music + ceiling fan noise

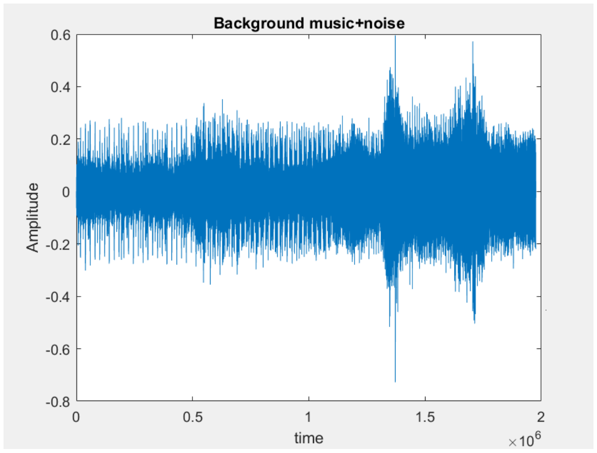


Figure 13: Background music + city traffic noise

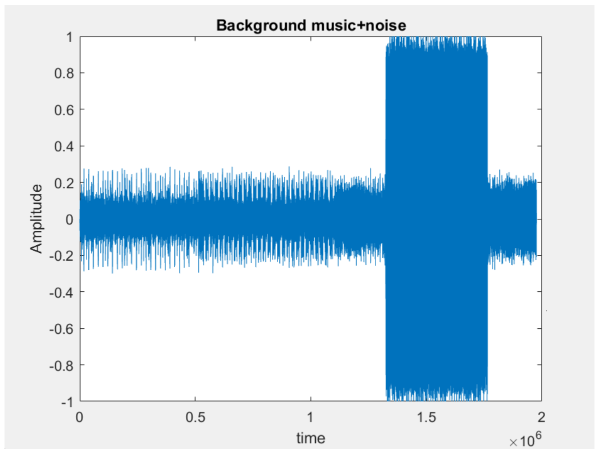


Figure 14: Background music + pressure cooker noise

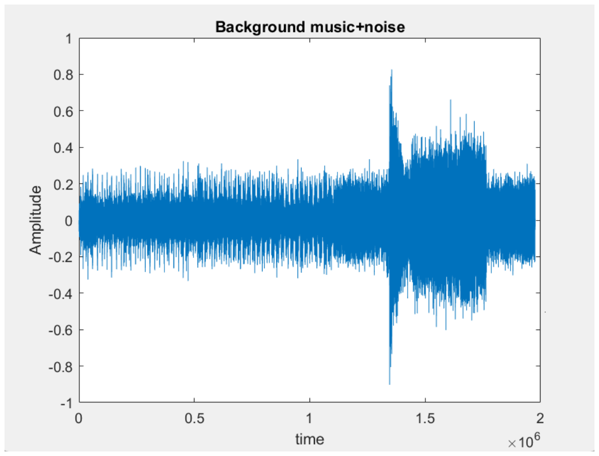


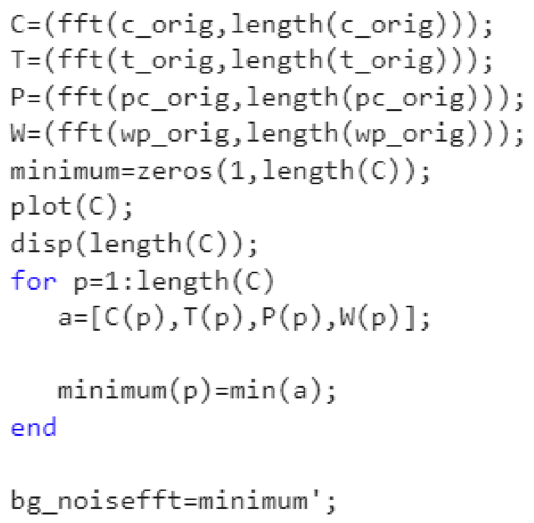
Figure 15: Background music + water pump noise

3.3 Problem Solving Approach

The main problem what we wish to solve is the distinction between the background music and the noise in each signals and remove the background noise to get somewhat original noises which we match with the input audio and analyze the results.To solve this -

- The primary objective was to isolate and extract the original noise components by leveraging Fast Fourier Transform (FFT) analysis. The initial step involved applying Fast Fourier Transform to each audio signal. This process yields a frequency response profile, providing insights into the frequency components present in the signal. The FFT analysis is crucial for understanding the spectral characteristics of the audio, both in the presence of background music and various noises.

  At each frequency point in the FFT results, the minimum frequency response was determined across all four signals. This approach assumes that all four signals contain information related to either music alone or music combined with noise. By selecting the minimum response at each frequency, we effectively isolated the frequency response associated with the background music.



```
C=(fft(c_orig,length(c_orig)));
T=(fft(t_orig,length(t_orig)));
P=(fft(pc_orig,length(pc_orig)));
W=(fft(wp_orig,length(wp_orig)));
minimum=zeros(1,length(C));
plot(C);
disp(length(C));
for p=1:length(C)
   a=[C(p),T(p),P(p),W(p)];

   minimum(p)=min(a);
end

bg_noisefft=minimum';
```

Figure 16: Code Snippet For Finding Approximate Background Noise frequency response

- Following the isolation of the background music frequency spectrum through FFT analysis, the next crucial step involved extracting individual noise components from the original audio signals. This process was executed by subtracting the background music FFT from each signal, resulting in unique FFTs corresponding to fan, traffic, pressure cooker, and water pump noises.

  By subtracting the background music FFTs from their respective audio signals, we obtained distinct FFTs representing the individual noise components. Each FFT encapsulated the frequency characteristics associated specifically with fan, traffic, pressure cooker, and water pump noises. This step facilitated a focused analysis of each noise type in the frequency domain.

  The application of IFFT resulted in the generation of time-domain audio signals for each noise category. These signals effectively captured the temporal characteristics of fan noise, traffic noise, pressure cooker noise, and water pump noise. The reconstructed signals in the time domain were pivotal for a comprehensive analysis of the individual noise components.

- Now we take the input signal we get and take cross correlation with each of the fan,traffic,pressure cooker and water pump time domain noises.

  Cross correlation is a signal processing technique used to measure the similarity between two signals or sequences. It quantifies how much one signal resembles another by comparing their respective values at different time or spatial positions. In cross correlation, the process involves sliding one signal (the reference signal) over another (the target signal) and calculating the product of the overlapping values at each position. The result is a correlation function that indicates the degree of similarity between the two signals at different alignment positions.

  The presence of max peaks in the correlation function signifies instances where the two signals closely match or align. The location of these max peaks corresponds to the time or spatial positions where the similarity is highest. By identifying these max peaks, one can determine not only the degree of similarity but also the temporal or spatial shifts needed to align the signals for optimal resemblance.

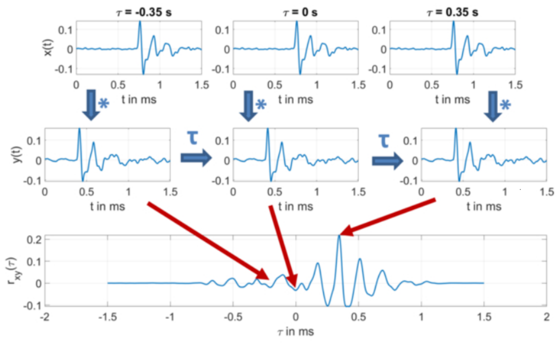
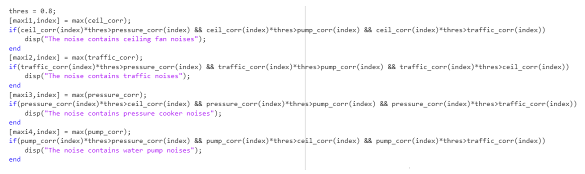


Figure 17: Graphical Visualizaton of Cross Correlation

- Now from this correlation we find out where the maximum peaks are to do tht-
  - A threshold (thres = 0.8) is defined to determine the significance level for identifying max peaks in the cross-correlation functions. This threshold helps filter out less significant peaks and focuses on strong correlations, enhancing the accuracy of noise classification.
  - The code snippet efficiently classifies various noise sources within the audio signal by leveraging cross-correlation analysis. The established threshold ensures the significance of identified peaks in the correlation functions.



```
thres = 0.8;
[maxi1,index] = max(ceil_corr);
if(ceil_corr(index)*thres>pressure_corr(index) && ceil_corr(index)*thres>pump_corr(index) && ceil_corr(index)*thres>traffic_corr(index))
    disp("The noise contains ceiling fan noises");
end
[maxi2,index] = max(traffic_corr);
if(traffic_corr(index)*thres>pressure_corr(index) && traffic_corr(index)*thres>pump_corr(index) && traffic_corr(index)*thres>ceil_corr(index))
    disp("The noise contains traffic noises");
end
[maxi3,index] = max(pressure_corr);
if(pressure_corr(index)*thres>ceil_corr(index) && pressure_corr(index)*thres>pump_corr(index) && pressure_corr(index)*thres>traffic_corr(index))
    disp("The noise contains pressure cooker noises");
end
[maxi4,index] = max(pump_corr);
if(pump_corr(index)*thres>pressure_corr(index) && pump_corr(index)*thres>ceil_corr(index) && pump_corr(index)*thres>traffic_corr(index))
    disp("The noise contains water pump noises");
end
```

Figure 18: Logic To Find the Maximum Correlation Peak

The approach seamlessly categorizes different noise types, including ceiling fan, traffic, pressure cooker, and water pump, based on the highest correlation peaks. This methodology provides a robust and accurate mechanism for identifying and labeling specific background noises in the audio recordings, contributing to a comprehensive noise classification system.

3.4 Results

- Accurate Classification: The developed algorithm consistently and accurately classifies various background noises present in audio recordings. Despite the presence of both background music and multiple noise sources, the algorithm successfully distinguishes and categorizes the noises.
- Robust Discrimination: The algorithm demonstrates robust discrimination capabilities, precisely identifying distinct noise types such as fan noise, pressure cooker/mixer noise, water pump noise, and traffic noise. This robustness is crucial for real-world applications where diverse noises coexist with music.
- Enhanced Output Clarity: The classification results provide a clear and detailed representation of the different noise sources within the audio recordings. This enhanced output clarity contributes to a comprehensive understanding of the composition of background noise in the context of music recordings.

3.4.1 Test I-Test Signal as bgmusic+fan noise

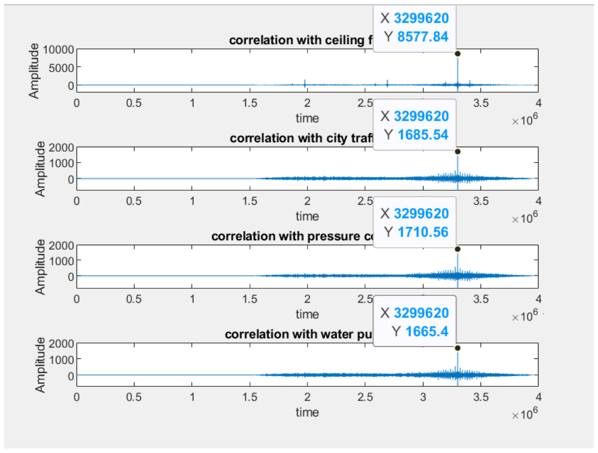


Figure 19: Correlation of test signal with all signals

We see max peak(8577) for ceiling fan while others are around 1000

3.4.2 Test II-Test Signal as bgmusic+ traffic noise

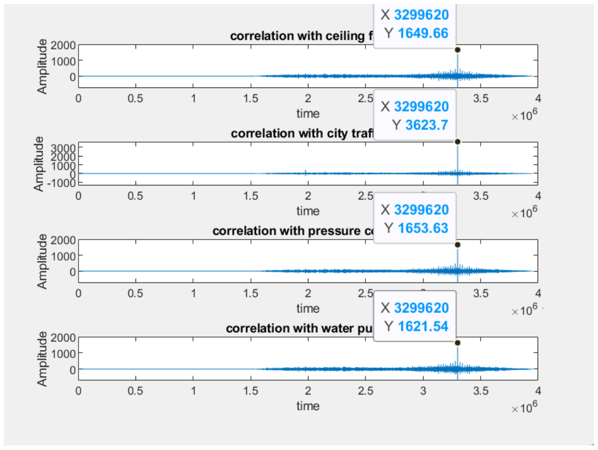


Figure 20: Correlation of test signal with all signals

We see max peak(3623) for traffic noise while others are around 1000

3.4.3 Test III-Test Signal as bgmusic+ traffic +ceiling fan+water pump+pressure cooker noise

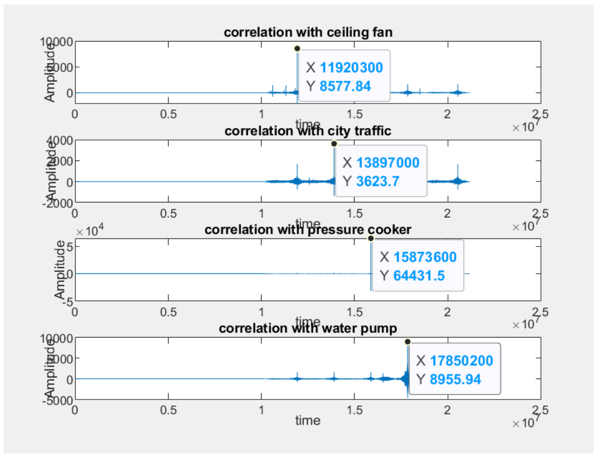


Figure 21: Correlation of test signal with all signals

We see max peak greater than (3000) for all the noises therefore the code will display the presence of all four noises

3.5 Conclusion

In conclusion, the developed algorithm has demonstrated commendable accuracy in classifying various background noises within audio recordings, even when overlaid with background music. The robust discrimination capabilities of the algorithm enable precise identification and categorization of noise sources, contributing to a nuanced understanding of the complex acoustic environment in music recordings.

The implications of this research extend to real-life applications in audio processing and enhancement. The algorithm's ability to differentiate between specific noise types, such as fan noise, pressure cooker/mixer noise, water pump noise, and traffic noise, is invaluable in the development of advanced noise reduction systems for audio recording studios. Additionally, in the realm of entertainment and multimedia, this technology can significantly improve the quality of soundtracks by selectively attenuating or enhancing specific background noises, enhancing the overall user experience.

Furthermore, the algorithm's robust performance holds promise for applications in smart home devices and Internet of Things (IoT) technologies. Incorporating this noise classification capability can enable intelligent systems to adapt and respond to varying environmental acoustics, offering a more immersive and tailored experience to users in their daily lives. Overall, the successful implementation of this algorithm opens avenues for practical innovations that leverage the intricate classification of background noises in diverse real-world scenarios.