Signal Processing Lab Project

Aniruth S
2022102055
IIIT Hyderabad
aniruth.suresh@students.iiit.ac.in

Priya Mandot
2022102053
IIIT Hyderabad
priya.mandot@students.iiit.ac.in

T. Sri Vishnu Varun
2022102031
IIIT Hyderabad
thorthi.varun@students.iiit.ac.in

***Abstract*—This project delves into various aspects of audio signal processing, specifically focusing on echo creation, cancellation, and background noise classification. In the first part, an echo effect is generated for a given audio file using signal processing techniques. The resulting echo is designed to be audible yet natural, simulating acoustic environments. Adjustable parameters, such as delay time, allow customization of the echo effect.**

In the second part, an echo cancellation algorithm is implemented to remove non-uniform echoes from an audio signal. The algorithm effectively preserves the original, clean audio by identifying and modeling the non-uniform delays in the signal. Various signal processing techniques and algorithms are explored, promoting experimentation and creativity.

The third part shifts the focus to background noise classification in music recordings without removal. The objective is to accurately identify and categorize different types of noise, such as fan, pressure cooker, water pump, and traffic. Utilizing signal processing methods, the algorithm provides a detailed classification report specifying the types of noises present in the audio recording.

This comprehensive project not only explores fundamental concepts in audio signal processing but also encourages experimentation and creativity in algorithm design and implementation.

I. Part 1 - Echo Creation

A. Objective

The objective is to understand the principles of echo in audio signal processing, including its generation, characteristics, and perception.

B. Input

The input consists of a signal wave that propagates through a medium and undergoes reflection off a discontinuity in the propagation medium.

C. Understanding echo

In audio signal processing, an *echo* refers to a reflection of sound that reaches the listener later than the direct sound. A true echo is a single reflection of the original sound, characterized by a time delay that is proportional to the extra distance traveled by the reflected wave divided by the speed of sound.

A true echo can be realized as a signal wave that reflects off a medium discontinuity in the propagation medium, returning with sufficient magnitude and delay to be perceived by the human ear. In echo effects, distinguishing between the true sound and the artificial sound is crucial for human listeners. The audible nature of echoes is due to the relatively slow speed of sound, approximately 343 meters per second.

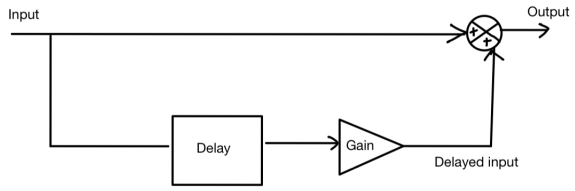

Fig. 1. Illustration of echo generation.

For simulating a single echo path, the following equation can be used:

$$\text{Output} = \text{Input} + \text{Delayed Input} \times \text{Gain}$$

$$y(n) = x(n) + a \cdot x(n - N)$$

where $y(n)$ represents the discrete-time audio output, $x(n)$ is the discrete-time audio input, $0 \leq a < 1$ is a real constant, and N is an integer constant in the set of natural numbers. The parameters a and N determine the amplitude and the time delay for the occurrence of the echo in the audio signal.

D. Echo Generation

The process of echo generation involves two paths for signal propagation, as illustrated in Fig 1. The first path involves the direct transmission of the signal from the source to the listener, while the second path includes reflection off a wall before reaching the listener. The delayed arrival of the reflected signal results in the perception of two sounds by the listener, each arriving at different times.

In audible frequencies, the human ear cannot distinguish an echo from the original sound if the delay is less than 1/10 of a second. Therefore, considering the speed of sound at approximately 343 m/s (at room temperature of about 20 °C), a reflecting object must be more than 16.2 meters away from the sound source for an echo to be perceptible. In practical scenarios, echoes are typically around one-half second or approximately half this distance, as sounds attenuate with distance.

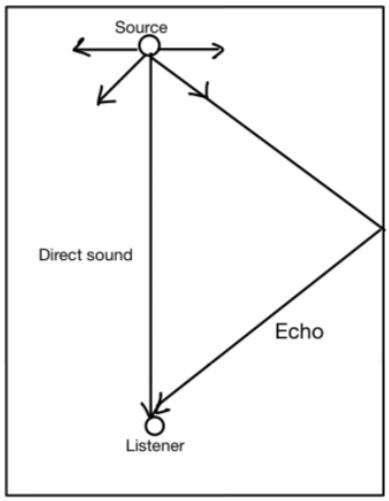

Fig. 2. Paths for signal propagation

```
function [y1,diff] = generateEcho(x,Fs,alpha,distance)
    timelag = 2*distance/343;
    delta = round(Fs*timelag);

    orig = [x;zeros(delta,1)];
    echo = [zeros(delta,1);x]*alpha;

    y0 = orig + echo;

    next = [y0;zeros(delta,1)];
    echo2= [zeros(2*delta,1);x]*alpha*alpha;

    y1 = next + echo2;

    diff = zeros(length(y1),1);
    diff(1:length(x)) = y1(1:length(x)) - x;
    diff(length(x)+1: length(y1)) = y1(length(x)+1: length(y1));

    y1 = y1/(max(abs(y1)));
    diff = diff/max(abs(diff));

end
```

Fig. 3. MATLAB code to generate two echoes

The strength of an echo is commonly measured in decibels (dB) of sound pressure level (SPL) relative to the directly transmitted wave.

E. Problem Solving Approach

To simulate an echo effect:

1) **Input Signal:** The original audio signal is loaded using the `audioread` function. If a stereo signal is present, it is converted to mono by taking the mean of both channels.
2) Utilize the `generateEcho` function, which introduces a time lag based on the user-provided distance.
   - **First Echo:** The first echo is generated by introducing a time delay (`delta`) to the original signal. The delayed version is denoted as `echo` and is scaled by the attenuation factor (α). The original and delayed signals are then combined to create the first echo (`y0`).
   - **Second Echo:** Another time delay is introduced to the combined signal (`y0`), creating a second delayed version denoted as `echo2`. This delayed signal is further scaled by α^2, representing the squared attenuation factor. The second echo (`y1`) is obtained by combining the previous signal (`y0`) with the second delayed signal (`echo2`).
3) **Difference Calculation:** The code also calculates the difference (`diff`) between the generated signal (`y1`) and the original input signal (`x`). The difference is set to zero for the duration of the original signal length to compare the echo components.
4) **Normalization:** The final output (`y1`) is normalised to a range typically between -1 and 1, ensures consistent amplitude, prevents clipping, and improves perceptual quality. It facilitates compatibility across systems and simplifies analysis and processing tasks.

F. Results

The results include the successful generation of an audible echo effect that accurately simulates the characteristics of natural echoes.

1) Sample file: (`q1.wav`). For a distance of 350m, the delays would then be 1.02 and 2.04 seconds.

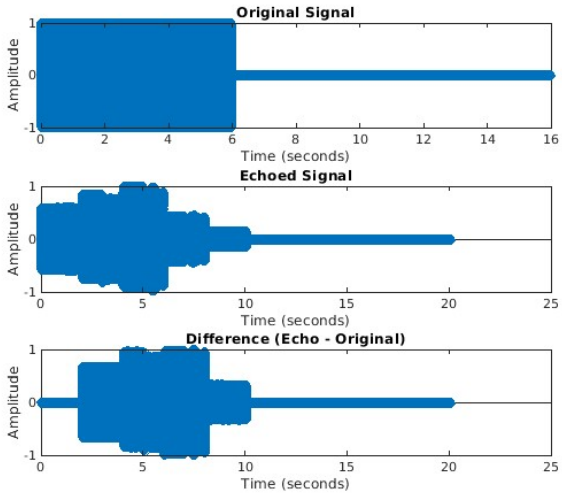

Fig. 4. q1.wav

2) Sample file: (`q1-hard.wav`). For a distance of 1000m, the delays would then be 2.91 and 5.82 seconds.

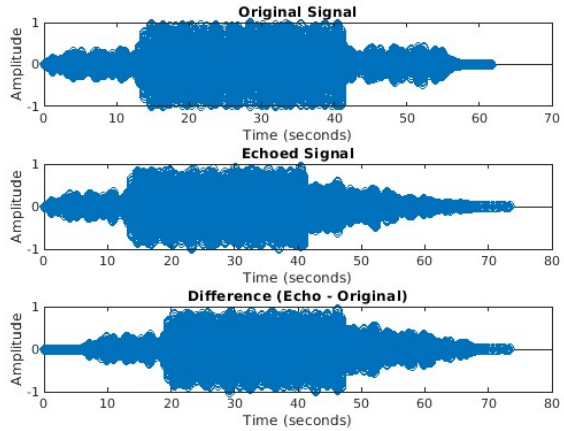

Fig. 5. q1-hard.wav

3) Sample file: (`hindi-2s.wav`). For a distance of 200m, the delays would then be 0.58 and 1.16 seconds.

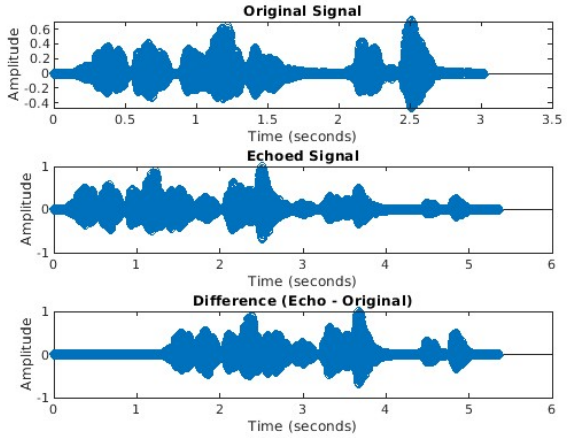

Fig. 6. hindi-2s.wav

G. Conclusion

The implemented MATLAB code successfully achieves the objective of creating an echo effect on an input audio signal. By utilizing the `generateEcho` function with user-defined parameters for distance, the script simulates the reflection of sound off a wall, producing a modified audio signal with two echoes. The resulting output file, effectively demonstrates the echo effect, providing a foundation for further exploration and experimentation in audio signal processing. The normalization step ensures consistent amplitude and compatibility across various playback systems. Part 1 lays the groundwork for subsequent sections, addressing echo cancellation and noise classification.

II. Part 2 - Cancel the Echo

A. Objective/Aim

The objective of this experiment is to create a robust echo cancellation algorithm for audio signals with varying delays. The input includes a mixed audio signal containing both the original sound and echo, along with a file containing the original sound without echo. The aim is to develop a signal processing algorithm that can accurately detect and model non-uniform delays related to the echo. The algorithm should skillfully eliminate the echo components, leaving a clean audio signal that closely resembles the original sound. The desired outcome is an audio signal that ideally only retains the untouched, clear sound.

B. Input

The input comprises an audio signal, delivered in a specific format (.WAV). This signal contains both the pristine original audio and echoes with non-uniform delays.

Any echo can be expressed in the form:

$$y[n] = x[n] + \alpha_1 x[n - d_1] + \alpha_2 x[n - d_2] + \dots$$

Here, $y[n]$ is the output signal with echo, and $x[n]$ is the original input signal. The terms $\alpha_1, \alpha_2, \dots$ represent the attenuation factors, and $d_1, d_2, \dots$ represent the corresponding delays.

First, we plot the input signal to analyse their properties and understand how echo cancellation can be applied.

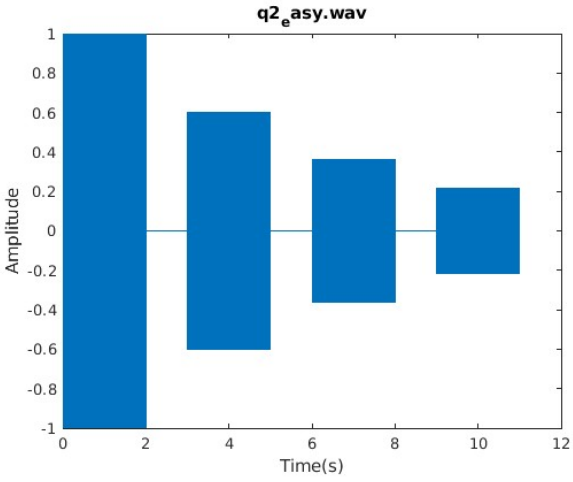

Fig. 7. q2-easy.wav

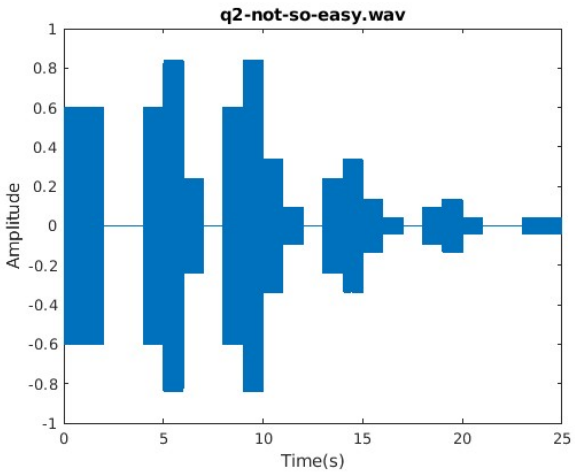

Fig. 8. q2-not-so-easy.wav

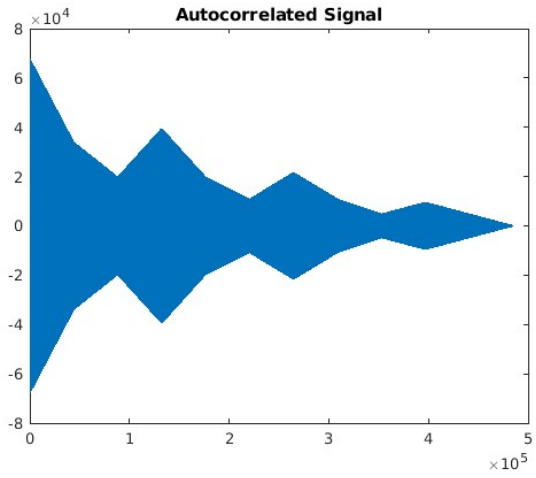

Fig. 9. Autocorrelated q2-easy.wav

C. Problem Solving Approach

We employ auto-correlation and envelope function to determine the delay and their attenuation factors in the signal. Subsequently, with these parameters, we employ an IIR filter to eliminate the echo.

1. **Auto-Correlation:**

The initial step in echo removal involves computing the auto-correlation of the signal with echo to identify the sample numbers (indices) where the delayed signal has been added to create an echo. The auto-correlation of a signal $r[n]$ is expressed as:

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

We then truncate the vector for only positive time stamps ($t > 0$), as the auto-correlation is symmetric, and for echo analysis, we are concerned with positive time lags. After performing the auto-correlation, we observe spikes in the plot only when the signal completely overlaps itself. This behavior is leveraged to identify the delay in the echo signal. The peaks in the plot provide information about the sample/time at which the delay occurs and the magnitude of the auto-correlation at that time. The auto-correlations obtained for the two inputs mentioned above are:

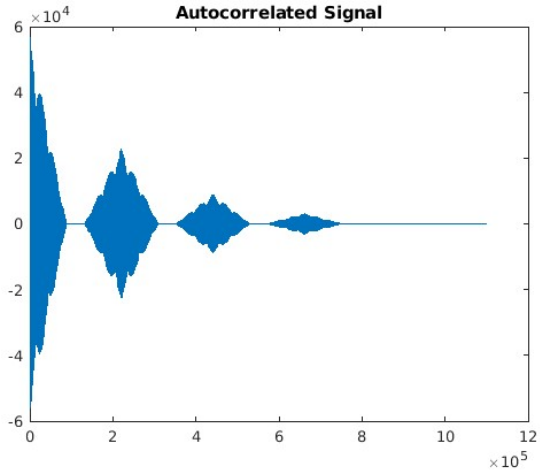

Fig. 10. Autocorrelated q2-not-so-easy.wav

- Delay analysis

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.

To obtain the corresponding x-values (the delays) for the rest of the peaks, we attempted to utilize the built-in MATLAB envelope function. However, this approach proved ineffective because the MATLAB code for the envelope function introduced a slight shift (4-6 samples) in the delay-associated x-values, leading to inconsistent results.

As a solution, we implemented our own envelope function to determine the start time of the delay. This custom envelope function follows a standard procedure, utilizing a constant line ($p = e^{-\lambda}$) and iteratively increasing λ to enhance the function and identify multiple delays (values corresponding to different peaks).

Points above the line at the intersection with a peak

are considered, and a separate analysis is conducted on them to ensure the identification of a unique maximum point within a peak. The analysis involves measuring the slope of these points and determining the maximum value among them. We repeat this analysis for all peaks, finding the maximum of the next peak until the maximum of the peak is less than a specified cutoff.
We set the cutoff as `cutoff` = Initial peak value/25. Additionally, we ensure that the maxima of two peaks are not in close proximity to each other to prevent irregular delays. To address this, we impose a condition explicitly stating that the minimum distance between any two peaks must be at least $0.1 \times F_s$. With these measures, we have successfully obtained all the start-of-delay values.

- Attenuation factor
  Now that we have the delay and the value of the peaks at these delays, we can calculate the attenuation 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 the attenuation factor at every delay by:

$$\text{Scaling factor}(\lambda) = \frac{\text{peak value at } k\text{th delay}}{\text{First max peak value}}$$

As we devised our custom envelope functions, they impose specific limitations on the types of input signals that can be effectively processed. Specifically, the signal characteristics exclude the presence of an echo at a particular frequency overlapping itself. This absence results in a lack of prominent peaks in the auto-correlation of the signal, impeding the identification of delays and scaling factors. Consequently, this method cannot be employed to determine the delay and scaling factor in such cases.

Resutls of the envelope function giving the exact peaks:

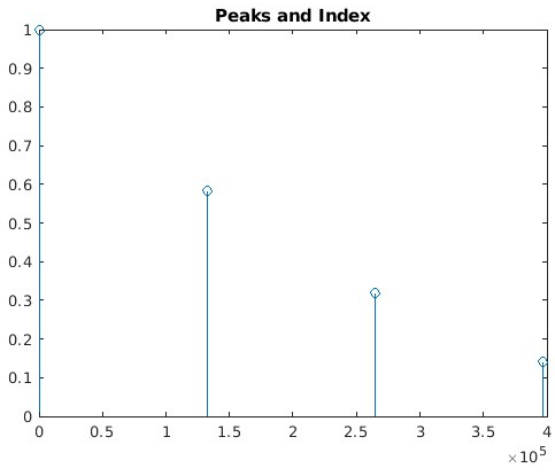

Fig. 11. Peaks and Indices of q2-easy.wav

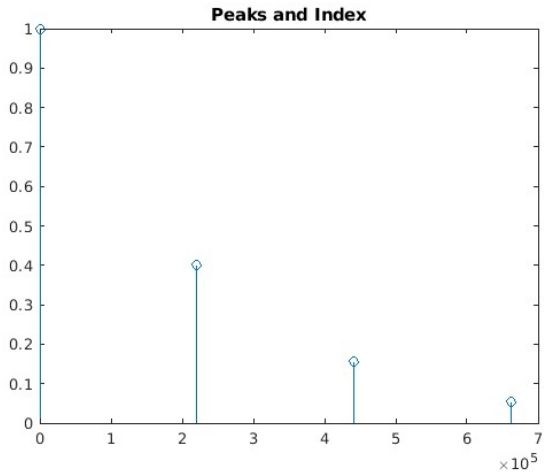

Fig. 12. Peaks and Indices q2-not-so-easy.wav

2. **Filter Function:**

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 (in units of time) can be written as:

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

Generalizing this filter for k delays at the kth samples and finding the transfer function of this system, we get:

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

, where b_k is the attenuation factor at delay k.

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

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

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.

D. Results

The implemented echo cancellation algorithm removed non-uniform echoes from the input audio signal. The resulting output audio contains the original signal in majority demonstrating the effectiveness of the algorithm in enhancing audio quality.

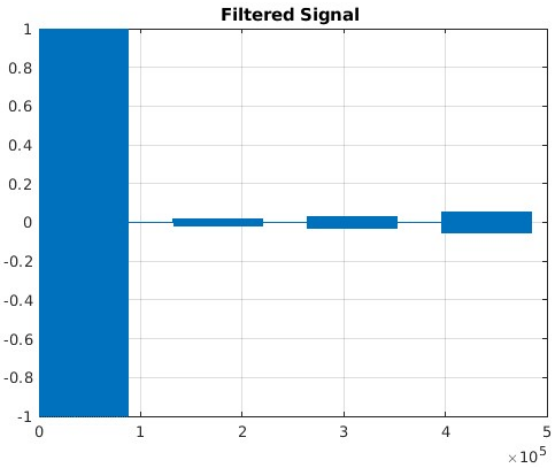

Fig. 13. Echo Cancellation in q2-easy.wav

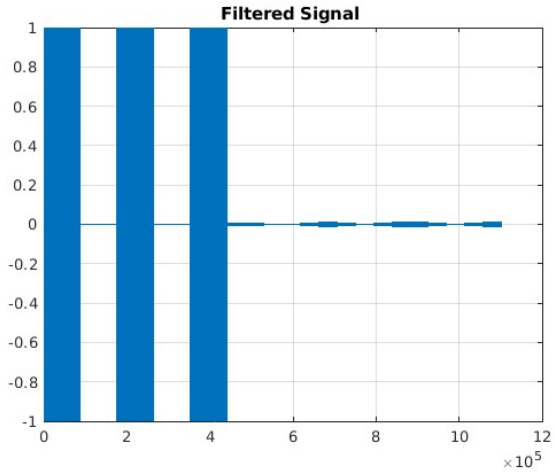

Fig. 14. Echo Cancellation in q2-not-so-easy.wav

On observation, there is still some echo remaining and this can be due to a few reasons: the cutoff chosen for the envelope was insufficient to capture all peaks and thus failed to attenuate at all of them. Also, the λ determining the slope in the envelope function could be too high which would also lead to losing a few peaks. However, decreasing the λ can make it more time-consuming, this is a trade-off we make.

E. Conclusion

Part 2 of our study focused on echo cancellation in audio signals with non-uniform delays. We developed a tailored approach, emphasizing the extraction of critical parameters like delays and scaling factors for effective echo removal. By addressing limitations in existing MATLAB functions, we refined our method to precisely identify delay times. Visual results showcased the efficacy of our approach, underscoring the importance of adapting techniques to signal characteristics for enhanced echo cancellation in scenarios with non-uniform delays.

III. Part 3 - What is this Noise?

A. Objective

The experiment aims to create an algorithm for categorizing background noise in music recordings without eliminating the noise. The goal is to precisely detect and classify different noise sources, including fan noise, pressure cooker/mixer noise, water pump noise, and traffic noise, within the recordings. The algorithm will employ signal processing techniques to analyze the audio data and generate a comprehensive classification report, specifying the presence of distinct noise sources and assigning appropriate labels. The primary objective is to enhance comprehension of background noise composition in music recordings and contribute to the development of effective noise classification systems.

B. Input

The input data consists of four audio files in the .wav format. Each file contains a consistent background music track combined with one of the following background noises: ceiling fan, pressure cooker, water pump, and traffic sounds. This setup aims to simulate real-world scenarios where music recordings may incorporate various environmental noises.

C. Problem Solving Approach

1. **Obtaining reference noise signals:**

In the pursuit of identifying reference noise signals for cross-correlation with the input signal, two distinct approaches are employed. The first method involves directly acquiring noises from the internet, while the second method entails extracting reference noises from provided test files by leveraging Fast Fourier Transform (FFT) analysis. These methodologies aim to establish reliable and diverse sets of reference noise signals for the subsequent identification and classification process.

For the latter method, FFT was applied to each audio signal, providing a frequency response profile and crucial insights into spectral characteristics with background music and various noises. The minimum frequency response at each frequency point was determined across all signals, effectively isolating the background music's frequency response. Subsequently, individual noise components were extracted by subtracting the background music FFT from each signal. The resulting FFTs represented fan, traffic, pressure cooker, and water pump noises. Application of Inverse FFT (IFFT) generated time-domain audio signals, capturing temporal characteristics of each noise type. These reconstructed signals in the time domain were essential for a comprehensive analysis of individual noise components.

2. **Cross Correlation:**

Cross-correlation, a foundational signal processing technique, assesses the similarity between two signals by comparing their values at different time or spatial positions. The method involves sliding a reference signal over a target signal, computing the product of overlapping values to create a correlation function. Max peaks in this function represent

instances of strong alignment and resemblance between the signals. To refine the identification of significant peaks, a threshold is applied, filtering out less substantial correlations and focusing on robust similarities. This threshold plays a pivotal role in enhancing the accuracy of noise classification.

The threshold values employed in the cross-correlation analysis are tailored to each specific noise type and determined through prior knowledge of samples containing that particular noise. These thresholds are established by assessing the maximum cross-correlation values observed when the specific noise is present in known samples. By leveraging this prior knowledge, the methodology ensures that the threshold accurately reflects the distinctive characteristics of each noise type. This customized approach enhances the precision of noise classification, allowing for more nuanced and effective identification of background noises in audio recordings.

The code snippet adeptly utilizes cross-correlation analysis to classify various noise sources within audio signals. By implementing a threshold, the method ensures the significance of identified peaks in correlation functions. This approach seamlessly categorizes noise types, including ceiling fan, traffic, pressure cooker, and water pump, based on the highest correlation peaks. The methodology not only identifies but also accurately labels specific background noises in audio recordings, contributing to the development of a comprehensive noise classification system.

3. **Time grid : For multiple noises**

In the case of an audio file containing multiple noises, we segmented the cross-correlation results into equal time intervals corresponding to the length of the signals. Within each interval, we identified the global maximum among the four signals, ensuring that it exceeded the specified threshold. It was crucial to precisely specify the noise type of interest, as a universal set of thresholds would not be effective for all cases. This approach allowed for a more targeted and accurate assessment of cross-correlation peaks, tailoring the analysis to the distinct characteristics of each noise present in the audio file. Additionally, this method facilitated the determination of noise duration and the identification of a time range during which the noise was added to the original sound. This methodology proves most effective when dealing with audio files containing non-overlapping noises.

D. Results

The results indicate successful noise detection. Firstly, we test files with single noise type:

1) Sample file: `music-ceiling-fan-hp.wav`

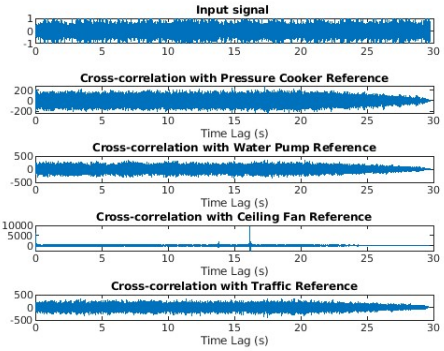

Fig. 15. Cross correlation with ceiling fan has the max peak

2) Sample file: `music-pressure-cooker-hp.wav`

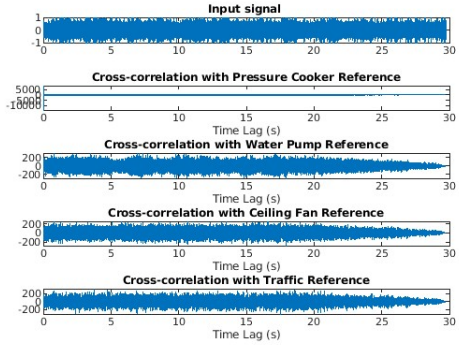

Fig. 16. Cross correlation with pressure cooker has the max peak

3) Sample file: `music-water-pump-hp.wav`

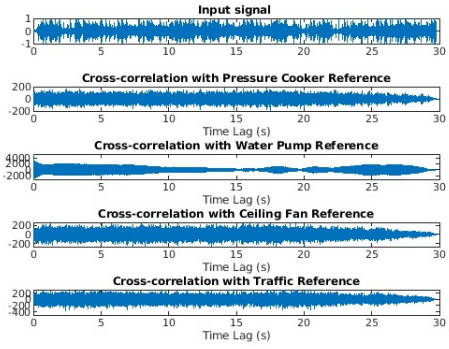

Fig. 17. Cross correlation with water pump has the max peak

4) Sample file: `music-city-traffic-hp.wav`

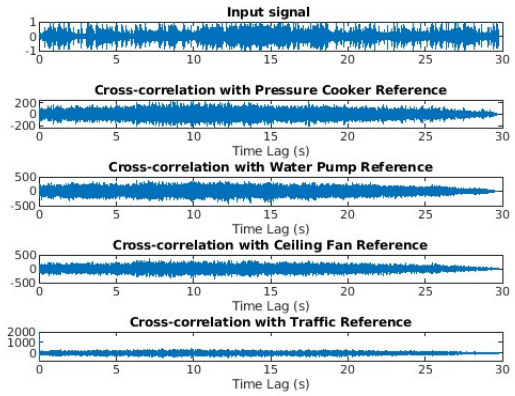

Fig. 18. Cross correlation with city traffic has the max peak

Now, to test over samples with multiple inputs, we merged files. Clarity in the generated graphs was achieved by systematically reducing values at different lags to zero if they fell below the specified threshold. This process significantly enhanced the visual representation of the detected type of noise within the graphs. By eliminating lower values and emphasizing peaks that surpassed the set threshold, the graphs provided a clearer depiction of the presence and characteristics of each noise type. Moreover, this approach allowed for the observation of both the amplitude of the peaks and the duration throughout which each noise was present, contributing to a more comprehensive and interpretable analysis of the audio signals.

- A combination of all

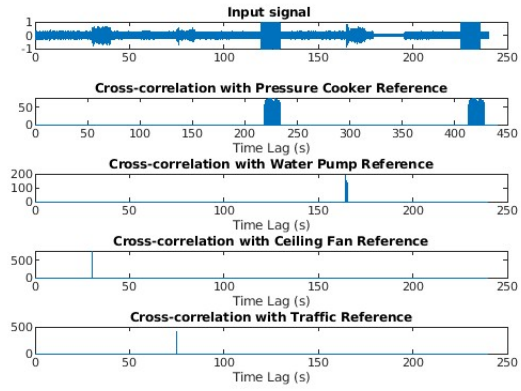

Fig. 19. Combination of all noises

- Pressure cooker, water pump, ceiling fan

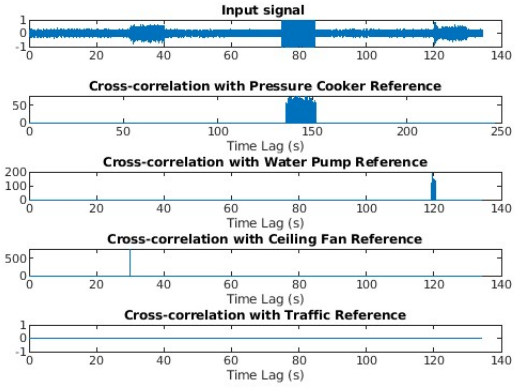

Fig. 20. Presence of pressure cooker, water pump and ceiling fan noises

- Pressure cooker, city traffic

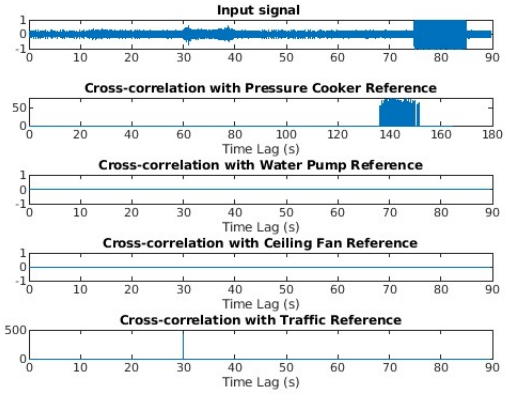

Fig. 21. Presence of pressure cooker and city traffic noises

E. Conclusion

Part 3 of the lab successfully addressed the task of classifying background noise in music recordings without removing the noise. The objective was to accurately identify and categorize various noise sources, namely fan, pressure cooker, water pump, and traffic, thereby distinguishing their sources of origin. The implementation involved a strategic approach, utilizing signal processing techniques and creative experimentation.

The developed algorithm effectively classified background noise sources by segmenting cross-correlation results, identifying global maxima, and refining graphs for enhanced clarity. The systematic reduction of values below the threshold notably improved the visual representation, emphasizing peaks and durations associated with each noise type. The outcome was a detailed classification report that specified the types of noises present in the audio recording.

While optimized for scenarios with non-overlapping noises, the methodology demonstrated efficacy, laying the groundwork for potential adaptations to address overlapping noise

components in future analyses. Overall, the lab successfully contributed to the accurate identification and categorization of background noise sources in music recordings, showcasing the potential for signal processing techniques in noise classification tasks.