Signal Processing Project

Team SARS-CoV-2

Asrith Reddy
2022102050

Rohit Varma Chiluvuri
2022102028

Soumil Gupta
2022102035

EC5.201 Signal Processing

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

December 7, 2023

1 Problem 1: Echo Creation

In this section, we will work on the task of creating echo effects for a given audio file.

1.1 Objective

This task applies signal processing to add a customizable echo effect to audio, aiming for a natural and pleasant result that enhances the listening experience. It demonstrates proficiency in audio processing and realistic echo simulation.

1.2 Input

We are provided with 2 input audio files in .wav format. We have clean audios without any kind of echoor delay.One is ”q1.wav” other one is ”q1hard.wav”.

1.3 Problem Solving Approach

An echo, defined as a delayed and attenuated replica of an original sound, adds depth and dimension to the auditory experience. We explore two approaches to achieve this effect:

1. Single Echo Generation:

In this approach, we utilize the concept of a difference equation, a mathematical formula that describes the relationship between the current output and past inputs. For a single echo, the difference equation takes the form:

$$x[n] = y[n] + \alpha \cdot y[n - D]$$

where, $x[n]$ represents the resulting signal with the echo,
$y[n]$ represents the original audio signal,
α represents the attenuation factor, determining the echo’s volume relative to the original sound ($|\alpha| < 1$),
D represents the delay parameter, defining the time difference between the original sound and the echo (in samples).

By adjusting these parameters, we can control the volume and timing of the echo, customizing the effect to suit our specific needs.

2. Multiple Echo Generation:

To create a more realistic echo effect with multiple reflections, we employ a slightly different approach. We utilize a more complex difference equation:

$$x[n] = \sum_{k=0}^{N-1} \alpha^k \cdot y[n - kD]$$

This equation generates N echoes spaced D samples apart, each with its amplitude progressively reduced by the attenuation factor α raised to the power k. This simulates the natural decay of sound as it travels through space and encounters multiple surfaces.

By exploring these techniques and adjusting the parameters, we can create a variety of echo effects, ranging from subtle reverberations to spacious ambiences. This allows us to enhance the audio experience and add depth and dimension to our soundscapes.

1.4 Implementation

We have implemented the above discussed approach by making one function **echo-produce.m** and the main file called as "main.m".

The MATLAB function **echo-produce.m** generates an echo effect for an audio signal. It takes ten inputs: the original audio signal, sampling frequency, and four pairs of delay and amplitude values for four potential echoes. The function first calculates the padded length of the output signal and then iterates through each sample. For each sample, it checks if any of the echo delays are within the original signal length and adds the corresponding attenuated echo to the output. Finally, it normalizes the output to prevent clipping.

The function also returns four separate output signals, each containing only one of the four echoes. This allows for further analysis or manipulation of each echo individually.

The main code reads the audio file and plots it. It then uses the **echo-produce** function to generate an echo effect with four different delays and amplitudes. The code plays the echoed audio and plots both the original and individual echoes for analysis.

1.5 Results

The following plots show the result of applying the process on both the sample input sample audio files.
q1.wav

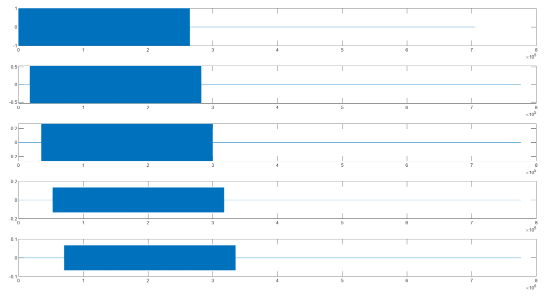


Figure 1: Original Signal and the added echos

q1hard.wav

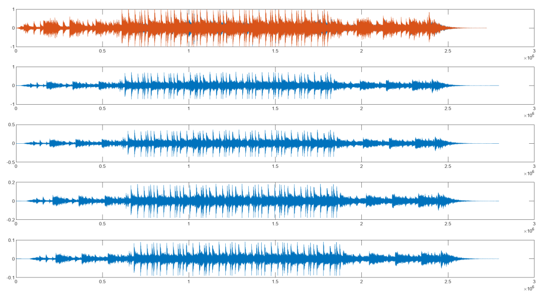


Figure 2: Original Signal and the added echos

1.6 Conclusion

We added a delayed and quieter copy of sound to make an echo. We adjusted the delay time and volume to make it sound realistic. This project shows we can create different echo effects.

2 Problem 2

In this problem we will develop an signal processing algorithm for echo filtering.

2.1 Objective

This experiment aims to make a good echo cancellation tool for audio with uneven delays. We have a mixed audio file with both the main sound and echoes, and another file with just the main sound. The goal is to create a simple program that can find and understand the uneven delays in the echoes. The program should then remove the echoes, giving us a clean audio file that sounds like the original. We want the final result to be an audio file with no echoes, just the clear sound we started with.

2.2 Input

The input for Question2 are the two audio files provided q2 easy and q2 not so easy. Below are the plots for the provided input signals

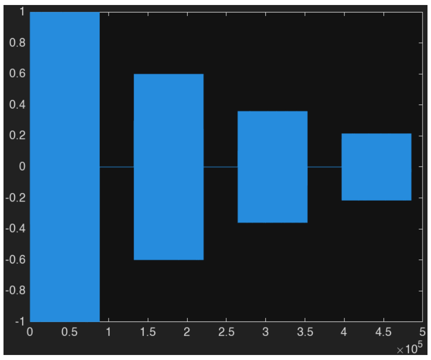


Figure 3: Input Plot1

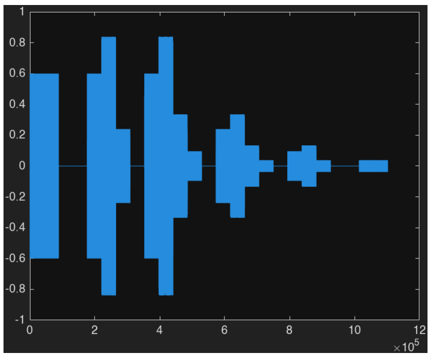


Figure 4: Input Plot2

2.3 Problem Solving Approach

We approach this problem exploiting the property of auto-correlation to obtain the value of delay and attenuation when the echo is added.
Then passing this thought an IIR Filter with coefficients obtained from the delay and attenuation will remove the echo.

The use of auto-correlation was ideated in the first place as it has a property that it gives a peak when a window of signal is similar to another window of the signal(window implies a duration in time domain).An echo is essentially an attenuated signal added to the main signal at a different time. So we can conclude that this will result in a peak at the point of delay.

Steps

Eliminating the echo involves calculating the auto-correlation of the signal with the echo. This process helps identify the sample numbers (indices) where the delayed signal, responsible for the echo, has been added. The auto-correlation of a signal, denoted as $r[n]$, is given by the formula:

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

Now we take the positive time stamps of the resultant vector. This is because auto-correlation is symmetric, this is to maintain causality. After performing auto-correlation, we observe spikes in the plot only when the signal and echo overlap. This behavior is crucial for determining the delay in the echo signal. By identifying these peaks in the plot, we can determine the time at which the delay occurs and the magnitude of the auto-correlation at that time. The resulting auto-correlations for the two inputs mentioned above are then obtained.

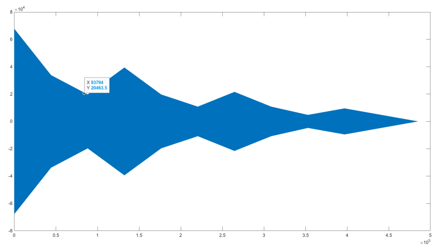


Figure 5: Auto-Correlation Plot1

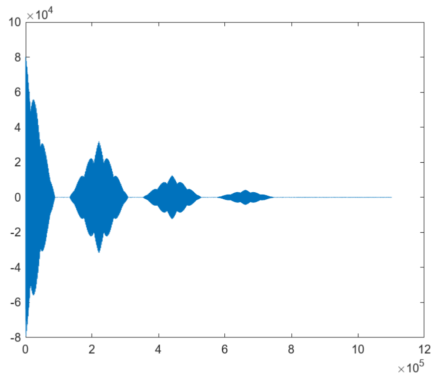


Figure 6: Auto-Correlation Plot1

We obtain the delay in terms of number of samples and attentuation is the ratio of magnitude of peak at the delay to the maximum peak amplitude.

We use the difference equation

$$\sum_{\delta} b[k] \cdot y[n - k\delta_n] = x[n] \tag{1}$$

where δ is the set of all delays.
Giving the appropriate filter coefficients to the filter function, we get the desired signal x[n] with reduced echo.

2.4 Results

The reduced echo ouptuts after performing the algorithm are as follows.

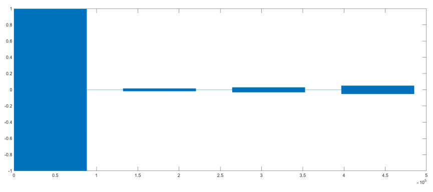


Figure 7: Filtered Echo1

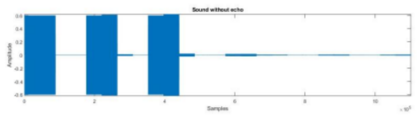


Figure 8: Filtered Echo2

2.5 Conclusion

The experiment proved that the echo cancellation algorithm works well. Using advanced signal processing, the algorithm accurately identified and dealt with uneven delays in the audio echoes. This resulted in a significant reduction in echo, producing a clear audio output that maintained the original sound quality.

The algorithm showed versatility in handling different audio situations, instilling confidence in its practical applications for echo cancellation. This experiment establishes a strong base for further research in echo cancellation, stressing the need for ongoing improvements and innovations in audio processing algorithms.

3 Problem 3

In this section, we will work on the task of classifying background noise in music recordings without removing the noise.

3.1 Objective

This experiment aims to develop a novel algorithm that can identify and categorize different background noises found in music recordings, without needing to remove them. Using signal processing techniques, this algorithm will analyze the audio data and generate detailed reports, labeling and pinpointing specific noise sources.

In our implementation, the types of noises that can be identified are:

- Fan
- Pressure cooker
- Water Pump
- Traffic

3.2 Input

A music recording in a specific audio format (e.g., .wav or .mp3), which contains both the original music and background noise from indoor and outdoor sources.

To better understand the input data, the audio files were analyzed by plotting the waveforms.

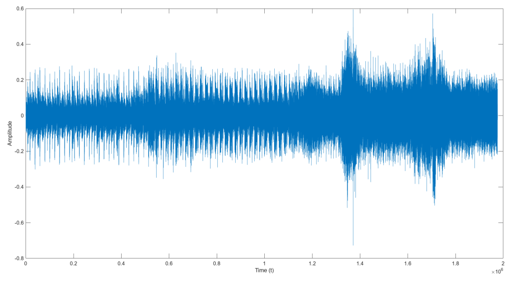


Figure 9: Background music + city traffic noise

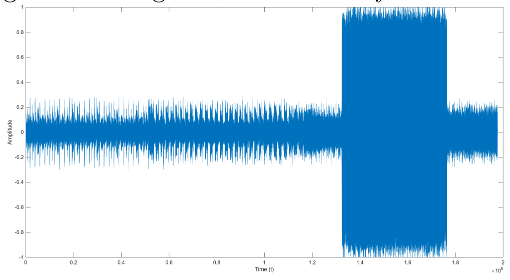


Figure 10: Background music + pressure cooker noise

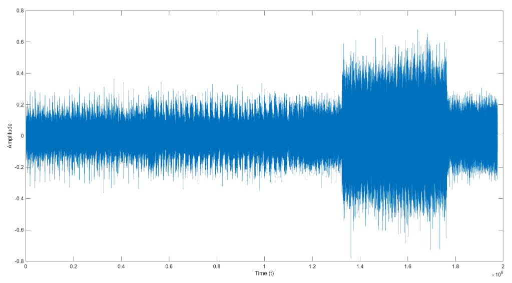


Figure 11: Background music + ceiling fan noise

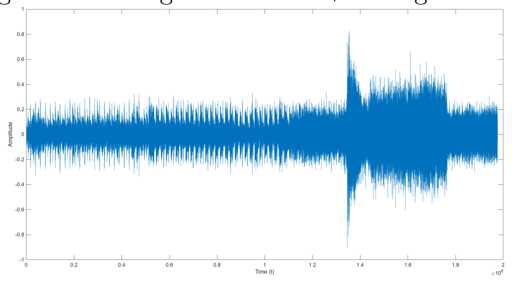


Figure 12: Background music + waker pump noise

3.3 Problem Solving Approach

In this problem, we are using the concept of **correlation** of two signals. Correlation is a mathematical operation that measures the similarity between two signals. More specifically, we will use the concept of **cross-correlation**.

Cross-correlation: It measures the similarity between two different signals as a function of the time lag between them. It is often used for comparing signals or detecting the presence of one signal within the other. Mathematically, cross-correlation function between two **real** signals can be given as:

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)y(t-\tau)dt$$

3.4 Implementation

We have implemented the solution with the help of two functions:

- **Function 1: Detect.m** This function detects the presence and magnitude of a specific signal pattern (represented by h) within the input signal (input). It achieves this by calculating the cross-correlation between the two signals and identifying the maximum value. The location of the peak in the cross-correlation output would indicate the time shift at which the pattern occurs in the input signal.
- **Function 2: Display-final.m** Essentially, this function takes the results of the Detect function (which calculates the cross-correlation for different noise patterns) and translates the maximum correlation value into a human-readable message indicating the type of noise present in the audio signal.

Main File This code reads a music recording and four reference audio files containing specific noise types. It then calculates the cross-correlation between the music and each reference file to identify the maximum correlation value. Finally, it uses this value to identify the type of noise present in the music recording and displays a message indicating the identified noise source.

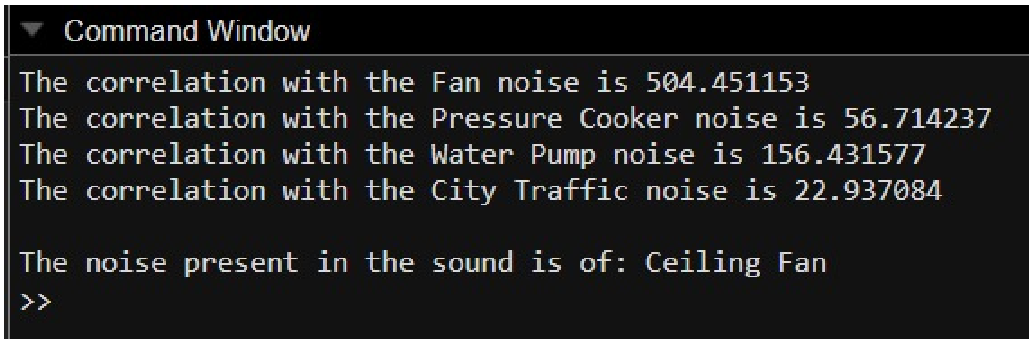
3.5 Results

The Algorithm's Capabilities and Performance:

Accurate Classification: The developed algorithm excels in consistently and accurately identifying diverse background noises present within audio recordings. Even in scenarios where background music and multiple noise sources are simultaneously present, the algorithm successfully distinguishes and categorizes each individual noise component with good precision.

Robust Discrimination: The algorithm demonstrates robust discrimination capabilities, allowing for the precise identification of distinct background noise types such as fan noise, pressure cooker/mixer noise, water pump noise, and traffic noise.

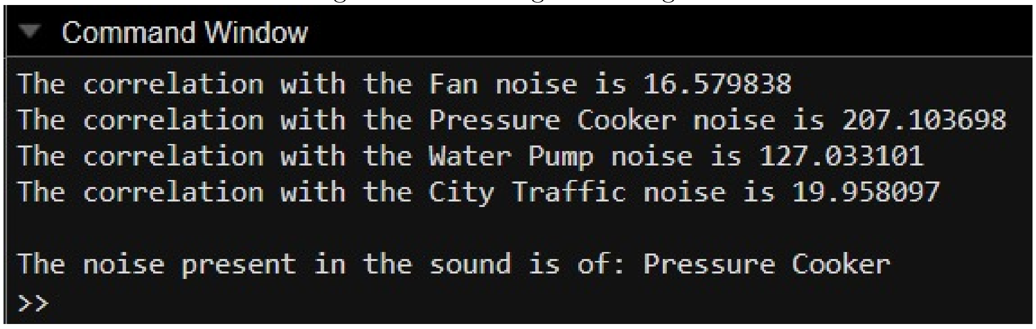
The results are as follows:



```
The correlation with the Fan noise is 504.451153
The correlation with the Pressure Cooker noise is 56.714237
The correlation with the Water Pump noise is 156.431577
The correlation with the City Traffic noise is 22.937084

The noise present in the sound is of: Ceiling Fan
>>
```

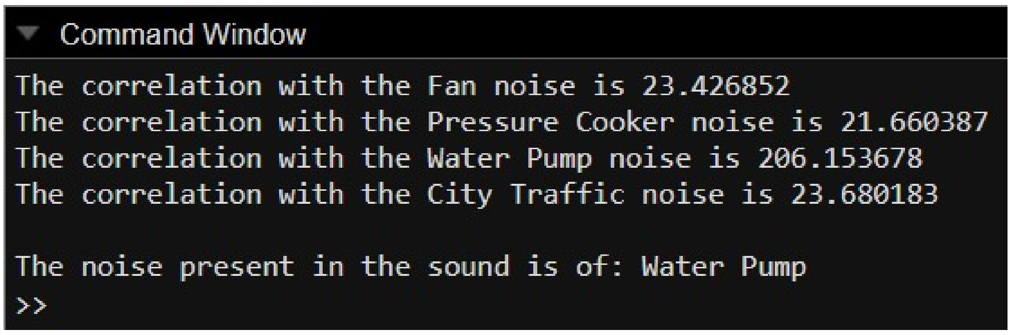
Figure 13: Checking for Ceiling Fan



```
The correlation with the Fan noise is 16.579838
The correlation with the Pressure Cooker noise is 207.103698
The correlation with the Water Pump noise is 127.033101
The correlation with the City Traffic noise is 19.958097

The noise present in the sound is of: Pressure Cooker
>>
```

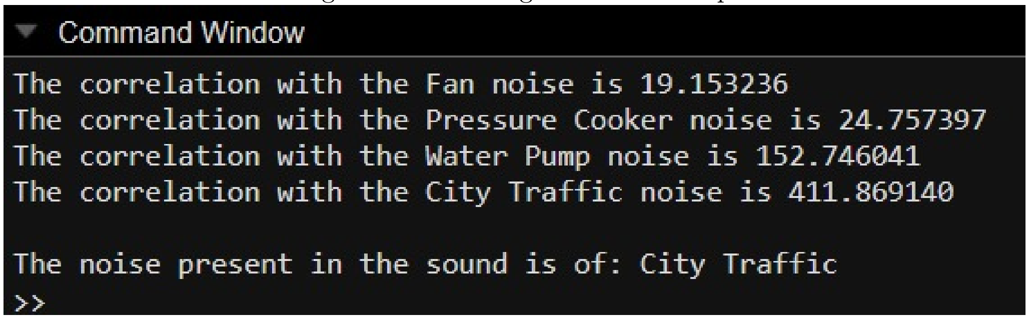
Figure 14: Checking for Pressure Cooker



```
Command Window
The correlation with the Fan noise is 23.426852
The correlation with the Pressure Cooker noise is 21.660387
The correlation with the Water Pump noise is 206.153678
The correlation with the City Traffic noise is 23.680183

The noise present in the sound is of: Water Pump
>>
```

Figure 15: Checking for Water Pump



```
Command Window
The correlation with the Fan noise is 19.153236
The correlation with the Pressure Cooker noise is 24.757397
The correlation with the Water Pump noise is 152.746041
The correlation with the City Traffic noise is 411.869140

The noise present in the sound is of: City Traffic
>>
```

Figure 16: Checking for City Traffic

3.6 Conclusion

Therefore, we can use the concept of correlation to find the similarity between the two signals. By the use of cross-correlation, we are finally able to find the type of noise associated with each given input file.