Signal Processing Project Report

Course Code: EC5.201
Term: Monsoon Semester 2023

Team: Koustubh Jain,
ID. 2023122003
koustubh.jain@research.iiit.ac.in

Raagav Ramakrishnan,
ID. 2022102018
raagav.ramakrishnan@students.iiit.ac.in

Yash Seri,
ID. 2022102015
yash.seri@students.iiit.ac.in

Date: 6 December 2023

1 Echo Creation

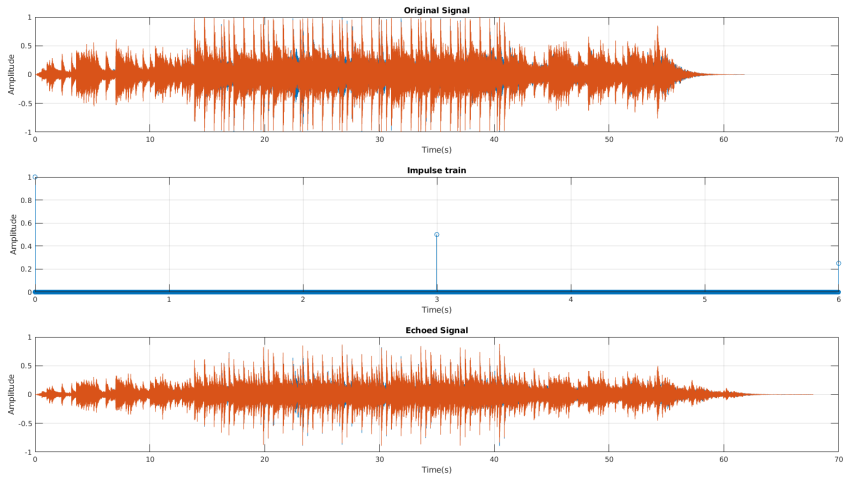

Figure 1: Echo creation process

Our program creates an echo which has two reflections of the original sound, each reflection is 50% attenuated as compared to the previous echo. In order to achieve this effect we convolve a modified impulse sequence with the audio sequence.

We first find out the sample at which the first echo will be heard, denoted by `delayed_sample`, then we proceed to create an empty zero vector of twice that length i.e. $2 \times$ `delayed_sample`

Then we se the value of the first index of the vector as 1, to account for audio being played initially before any echo is heard back. And then the value at the first delay to be 0.5, since we are assuming 50% attenuation with each reflection and then 0.25 at the second delay.

Then we convolve the original audio sequence with the generated impulse sequence. The result of which is normalised between -1dB to +1dB and written to a `.wav` at the sampling frequency of the original signal.

```
1     delayed_sample = delay*fs;
2     imp_train = zeros(2*delayed_sample,1);
3     imp_train(1,1) = 1;
4     imp_train(delayed_sample,1) = 0.5; %first echo with 50%
    attenuation
5     imp_train(2*delayed_sample,1) = 0.25; %second echo with
    50% attenuation
```

2 Echo Cancellation

For echo cancellation we have utilised a least mean square filter, which iteratively adjusts its filter coefficients in order to minimise the least mean squared error between the filter output and the desired signal. This algorithm is relatively cheap computationally to implement.

This algorithm is based on the steepest decent algorithm, which basically tries to minimise the mean squared error by moving in the direction of negative gradient of the mean squared error.

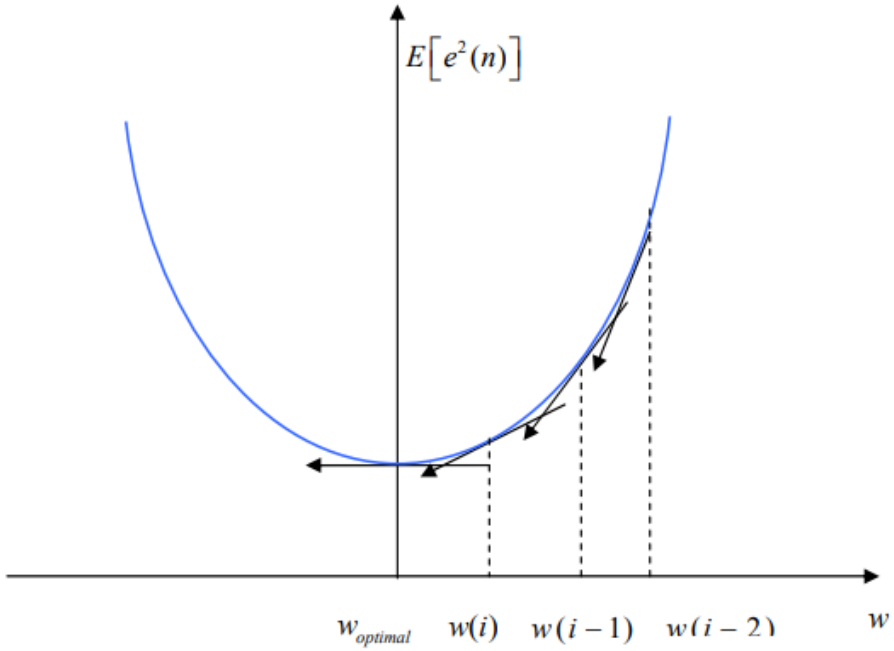

Figure 2: Illustration of steepest decent search method

The equation resulting from the steepest decent search method for the filter coefficients is:

$$w(n+1) = w(n) - \mu\nabla\xi$$

where $w(n)$ denotes filter coefficients in current iteration, $w(n+1)$ denotes filter coefficients in next iteration, mu denotes the step size and ξ denotes the mean squared error.

Gradient of mean squared error can be re-written as follows:

$$\nabla\xi(n) = \nabla E\{|e(n)|^2\} = E\{\nabla|e(n)|^2\} = E\{e(n)\nabla e^*(n)\}$$

Since, $\nabla e^*(n) = -x^*(n)$. Hence, $\nabla \xi(n) = -E\{e(n)x^*(n)\}$.

Hence, $w(n+1) = w(n) + \mu E\{e(n)x^*(n)\}$

Since, we don't usually know the value of $E\{e(n)x^*(n)\}$ we approximate it as the sampled mean over past L samples as:

$$\frac{1}{L}\sum_{l=0}^{l-1} e(n-l)x^*(n-l)$$

So, we get the equation for updating the filter coefficients as follows:

$$w(n+1) = w(n) + \frac{\mu}{L}\sum_{l=0}^{l-1} e(n-l)x^*(n-l)$$

.

If we use a, 1 point sampled mean i.e. $L = 1$ then the equation becomes:

$$w(n+1) = w(n) + \mu e(n)x^*(n)$$

.

In our code, we have used a 32-tap LMS filter, increasing the filter size will improve the quality of the output but it will also increase the run-time and memory requirements.

Filter coefficients intially all start out as zero and are then iteratively updated via the algorithm.

We have chosen a small step size of 0.014 in order to ensure stability of the system and keep track of any rapid changes in the sequence. A larger step size may lead to faster convergence but it will also fail to keep track of the rapid changes in the audio sequence.

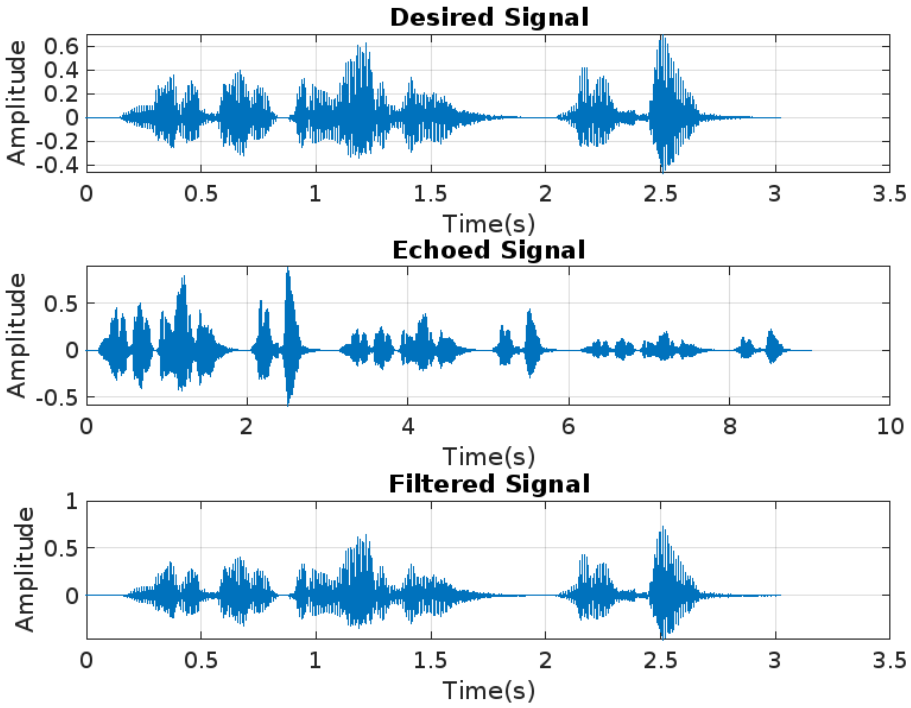

Figure 3: Output of echo cancellation function

3 Noise Classification

We have 2 major components in our code, an `extract_features` function to obtain important characteristics in our background noise, and a classifier to identify the kind of noise it is.

3.1 Extract Features

It's important to note that this `extract_features` function needs only the noise from the input signal, and not the combination of input and noise.

We first compute the cross-correlation by using xcorr(y), storing these value into a vector called `autocorr_values`. Since we are only concerned with all positive lags in the above vector, we modify it to extract the same. This helps us identify if there are any repeating patterns in the signal.

We then find the Mel-frequency cepstral coefficients (MFCCs) by using mfcc function from the audio toolbox. Since different background noises have

different spectral patterns, this helps us characterize the kind of signal we are dealing with.

We perform mel spectrogram calculations to represent short term power spectrum of input signal using the mel scale.

Spectral Centroid: CoM of Spectrum, find dominant frequencies based on freq content (distinguish b/w water pump and fan)

3.2 The Classifier

We feed in a couple 100 different noises into the classifier to build our model (SVM). By offering such a diverse set of options for traffic sounds, water pumps and ceiling fans, all sourced from the internet, we have a model that can reliably identify the kind of background noise we are dealing with for the given input signal.

By setting up a 30% margin for testing(30% of data) and training (70% of data), we can accurately determine the noise.