Image Processing - Exercise 2

Yotam Gardosh, yotam.gardosh, 208541334

Introduction

The goal of this exercise is to implement a denoising algorithm for two audio files, “q1.wav” and “q2.wav”. The primary tool used for this task was breaking down the frequencies using The Short Time Fourier Transform and altering the amplitude of the frequencies using a band pass.

The main difference between the noise was that in q1 we had one frequency with a much higher amplitude then the rest and in q2 we had more dispersed noise.

Algorithm

The denoising algorithm for q1.wav involves the following steps:

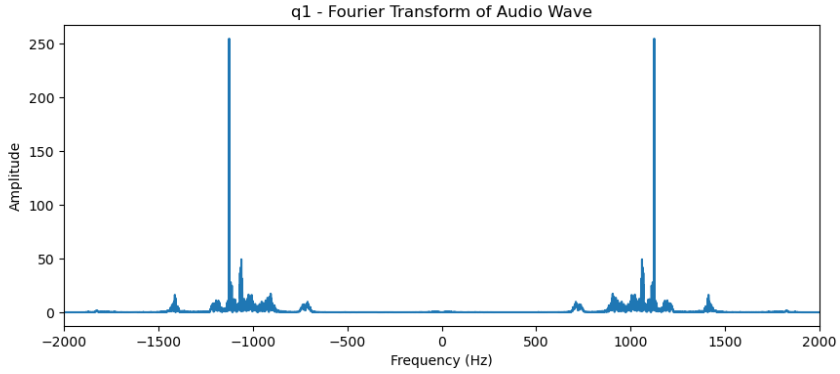
1. Compute the Fourier Transform of the audio data.
2. Identify the index of the highest magnitude frequency.
3. Set the magnitudes of the max frequency and its corresponding negative frequency to zero.
4. Perform Inverse Fourier Transform (IFFT) to obtain the denoised audio data.

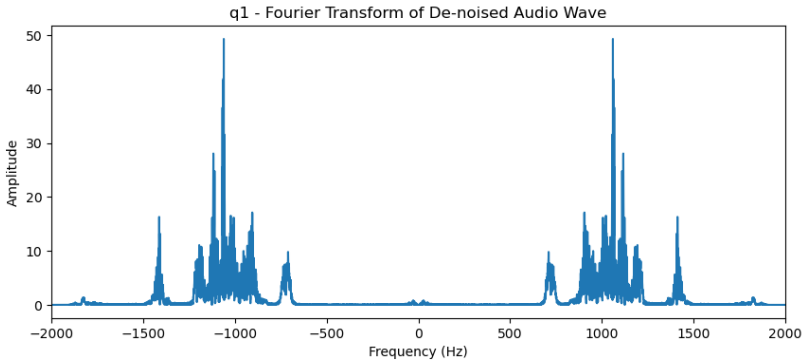
Implementation Details

My implementation of the algorithm described above was done in the following steps:

1. I Computed the Fourier Transform of the sound wave using the NumPy implementation of the algorithm.
2. Find and set highest magnitude frequency to 0 using simple arithmetic
3. I Performed the Inverse Fourier Transform using the NumPy implementation of the algorithm as well.

To analyze the data and determine the best algorithm to denoise the audio wave I created a function that uses the matplotlib’s pyplot library to plot the spectrogram of the audio data, as well as plot the FT. There we can clearly see the frequency that needed to be removed since its amplitude is 5 times higher than all the rest.





Algorithm

The denoising algorithm for q2.wav involves the following steps:

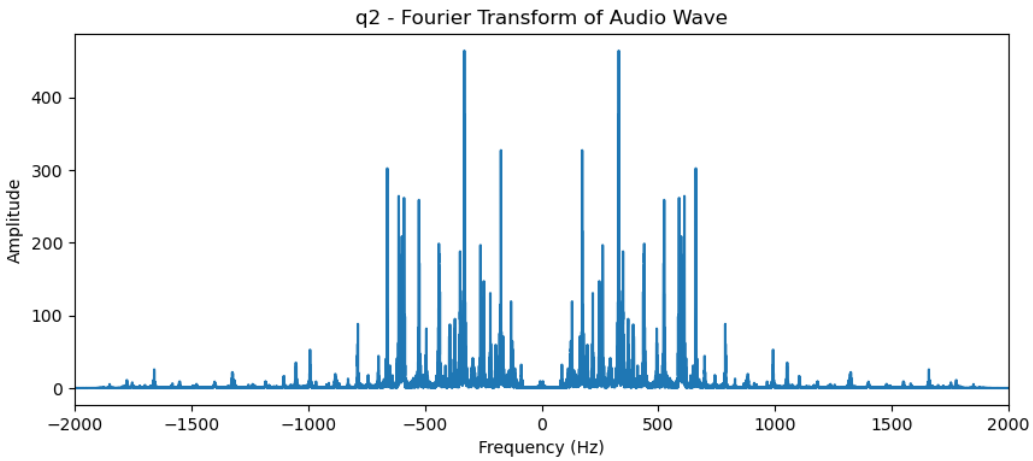
1. Compute the Fourier Transform of the audio data.
2. Identify the specific frequency ranges to set to zero.
3. Set the magnitudes of the max frequency and its corresponding negative frequency to zero.
4. Perform Inverse Fourier Transform (IFFT) to obtain the denoised audio data.

Implementation Details

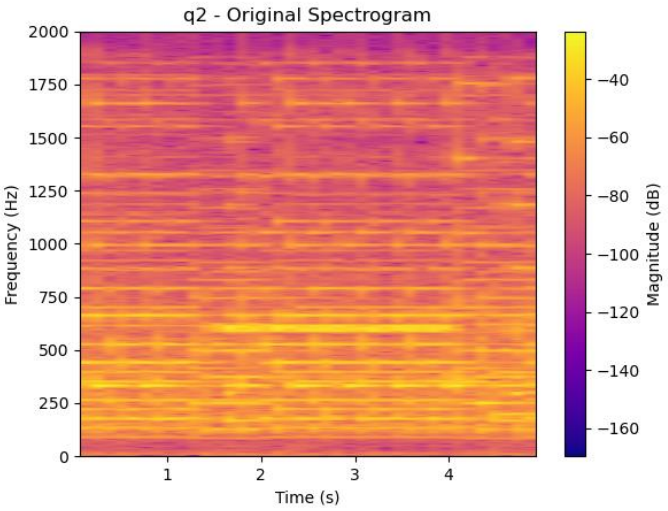
My implementation of the algorithm described above was done in the following steps:

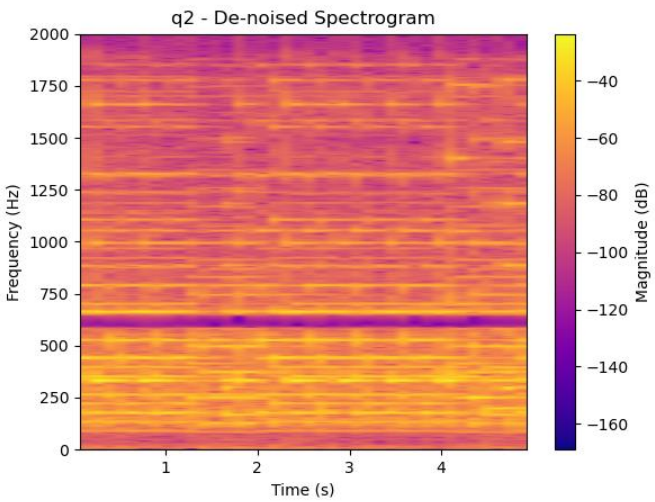
1. I Computed the Fourier Transform of the sound wave using the NumPy implementation of the algorithm.
2. Find and set highest magnitude frequency to 0 using simple arithmetic
3. I Performed the Inverse Fourier Transform using the NumPy implementation of the algorithm as well.

To analyse the data and determine the best algorithm to denoise the audio wave I created a function that uses the matplotlib's PyPlot library to plot the spectrogram of the audio data, as well as plot the FT.
This audio file presented much more difficulty to denoise, when looking at graph of the FT we cannot see anything much out of the ordinary.



Looking at the spectrogram I started to get a better idea at what kind of noise I needed to remove, listening to the audio file we hear a musical piece, my background in music and specifically in playing classical piano helped me understand that notes typically fluctuate in intensity, therefore I would expect the spectrogram to have “waves” of intensity. When looking at the spectrogram I noticed a continues spot at the area around 600 Hz between 1.5 and 4 seconds. Seeing how there is not much information in these frequencies before or after the noise I decided to remove all the frequencies from 580 Hz to 650 Hz. This proved a great solution to the noise.





Conclusion

In conclusion denoising audio files is harder than it might seem, noise does not necessarily stand out in either a spectrogram or a FT graph. Domain knowledge can greatly help identify the source of a given problem And tool like editing frequencies in the Fourie spectrum can cleanly remove the noise without damaging the audio.