EE200 Practical

Yash Bhardwaj
Roll No: 231184

2D Fourier Transform

The equations for the 2D Discrete Fourier Transform (DFT) and its inverse are:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \cdot e^{j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

We have used `fft` function from NumPy library and then plotted absolute value of that, this gives us the magnitude plot. Taking 20 log of that modulus we get the Bode plot.

Frequency Domain Observations

Plotting the magnitude spectrum using Mod and Bode plots we observe:

- Low-frequency components are concentrated at the top left corner of the transform.
- To center the zero-frequency component, we apply the `fftshift()` function.
- When the image is rotated, the Fourier transform rotates correspondingly.
- This means if we rotate original image by some angle the fourier transform also rotates in same direction by same angle. This is property of **Rotation Invariance**.

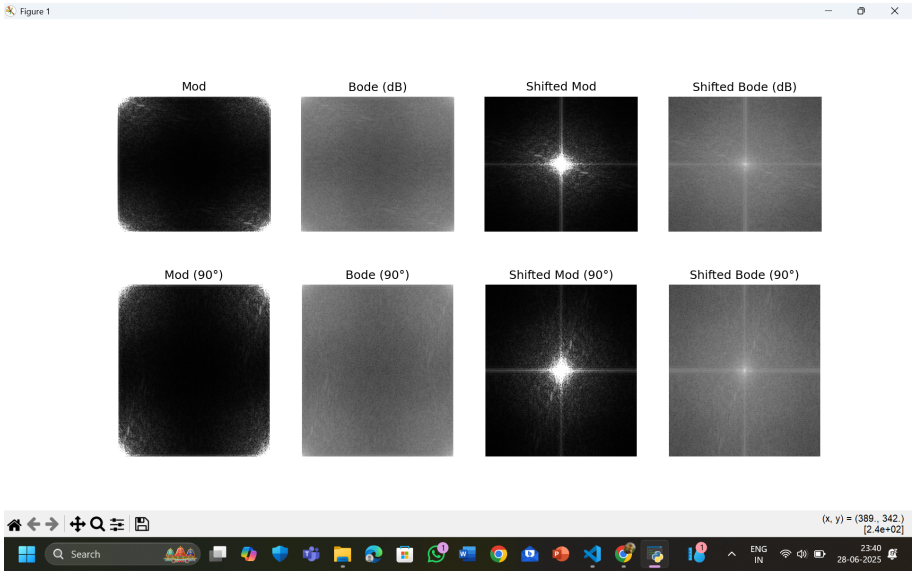

Figure 1: Magnitude Spectrum (was completely black so threshold to 99%ile to see difference between original and rotated) and Bode Magnitude Plots

Image Spectrum Plots

Original Image Frequency Analysis

We used the `np.fft.fft2` and `np.abs` functions to compute and visualize the magnitude spectrum of the image. Taking $20 \log_{10}$ of the magnitude gives us a Bode-Magnitude .

Frequency Mixer Concept

Design Rationale

Initially, I considered using a Gaussian filter using OpenCV library to get blur, and to extract fine details subtracting the blurred image from the original (i.e., `Image - Gaussian`). This approach worked well in practice.

However, to adhere to the course content and gain a deeper understanding, I instead implemented the required filters directly using NumPy arrays in the frequency domain.

In the frequency domain:

- **High-frequency components** correspond to rapid intensity changes—such as edges and fine textures.

- **Low-frequency components** correspond to smooth variations—such as general structure and color gradients.

Thus, to design the system:

- We removed high-frequency content to blur the image and retain structural components.
- We removed low-frequency content to extract edges and fine details.

Thus, we design a system where:

- A **low-pass filter (LPF)** is applied to one image's Fourier transform to extract smooth, structural content.
- A **high-pass filter (HPF)** is applied to the other image's transform to extract fine details and edges.

Ideal Low-Pass Filter Transfer Function

The transfer function of an ideal low-pass filter is defined as:

$$H_{\text{LPF}}(u,v) = \begin{cases} 1, & \text{if } D(u,v) \leq R \\ 0, & \text{otherwise} \end{cases}$$

where the distance from the center frequency is given by:

$$D(u,v) = \sqrt{(u-u_0)^2 + (v-v_0)^2}$$

Here, (u_0, v_0) represents the center of the frequency domain (i.e., after shifting).

Ideal High-Pass Filter Transfer Function

The ideal high-pass filter can be obtained by changing the values of blacks and whites:

$$H_{\text{HPF}}(u,v) = 1 - H_{\text{LPF}}(u,v)$$

Observations by changing radius: To display results such that we see both dog and cat images by looking from different distances, the value of radius has to be adjusted properly by experimenting, because some images may not have a lot of high frequency content. So we need to select lower radius in this case to display blurry image of that image, because a higher radius would mean almost all content of that image is visible.

The two filtered transforms are then added together to form a combined frequency representation:

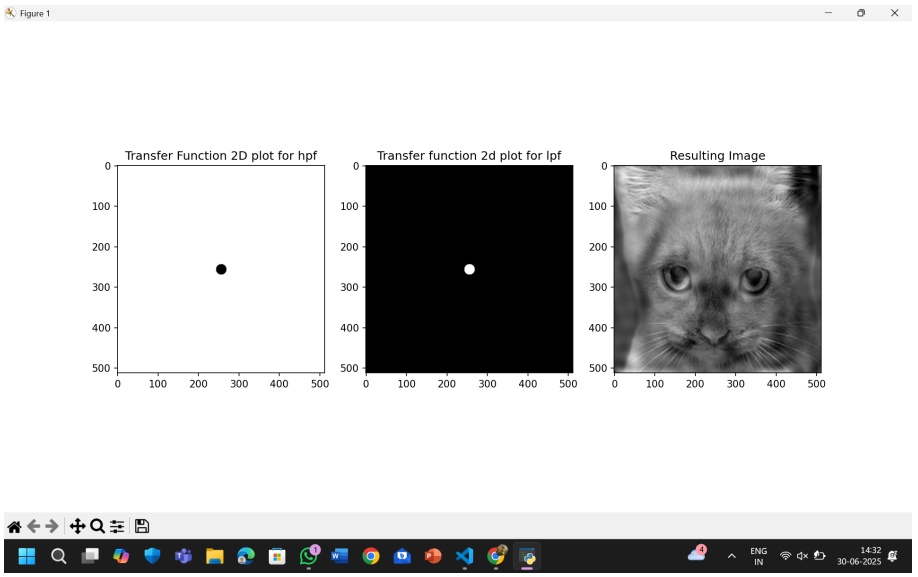

Figure 2: Transfer Function for Filters, Resulting Image

$$F_{\text{result}}(u, v) = F_{\text{low}}(u, v) + F_{\text{high}}(u, v)$$

Finally, we apply the inverse Fourier transform to retrieve the hybrid image:

$$f_{\text{result}}(x, y) = \left| \mathcal{F}^{-1} \left\{ F_{\text{result}}(u, v) \right\} \right|$$

Conclusion

By mixing frequency components from two images, we create a hybrid image that shows:

- One image when viewed from a distance (low-frequency perception),
- Another when viewed closely (high-frequency details).

Question 2: Audio Restoration Using Frequency Filtering

Power Spectral Density Analysis

Plotted the power spectral density (PSD) of the corrupted audio to identify regions of high energy.

- Significant power was observed in the ranges **0–700 Hz** and **1000–1800 Hz**.

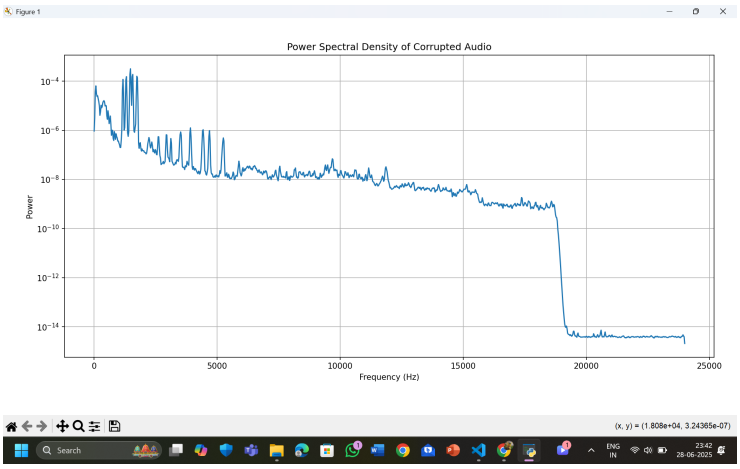

Figure 3: Power Spectral Density of the Original Audio

FFT and Frequency Analysis

The FFT of the signal was computed to analyze its spectral composition:

- Magnitude and Bode plots revealed dominant energy in **0–800 Hz** and **1000–2000 Hz**.
- A simple google search revealed, the unwanted instrument is likely a flute, which typically has fundamentals in **800–2100 Hz** and harmonics up to **12 kHz**. And since we were told that music has been corrupted due to presence of unwanted instrumental beats I decided to remove flute noise.

Filter Design via Frequency Masking

We designed a binary frequency-domain mask to suppress the following frequency bands:

$$(800, 2100)\ \text{Hz (fundamentals)}, \quad (2600, 5300)\ \text{Hz (harmonics)}, \quad (8000, 12000)\ \text{Hz (air noise)}$$

- The filter was applied to the FFT of the signal.
- The inverse FFT produced the time-domain filtered signal.
- Taking complete impulse response involved very high computation for zeroes so we took only first few components (100). I tried using 1000 as well and the Bode Magnitude plot became less oscillating.

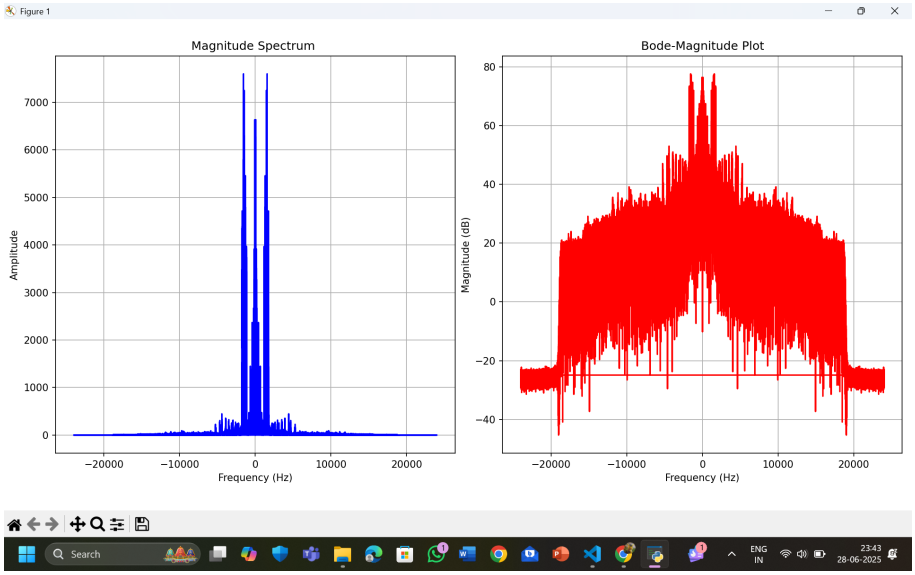

Figure 4: Left: FFT Magnitude Spectrum, Right: Bode Magnitude Plot

Filter Characterization

Bode Magnitude Response

The filter response was computed from the FFT of its impulse response:

Pole-Zero Plot

The zeros of the FIR filter were plotted on the Z-plane.

Results

- **Flute-like interference was attenuated successfully.**
- **PSD comparison confirms suppression in target bands.**

Conclusion

- Using FFT-based analysis and a frequency-domain mask, we created an FIR-like filter to suppress unwanted instrumental components. This method effectively attenuated the flute content while preserving the rest of the audio.
- **Results were satisfying, resulting audio was as expected. Bode magnitude plot was quite as expected, except for the fact that pole-zero plot had many zeroes. This is because I have used a high order polynomial approximation to my ideal filter which**

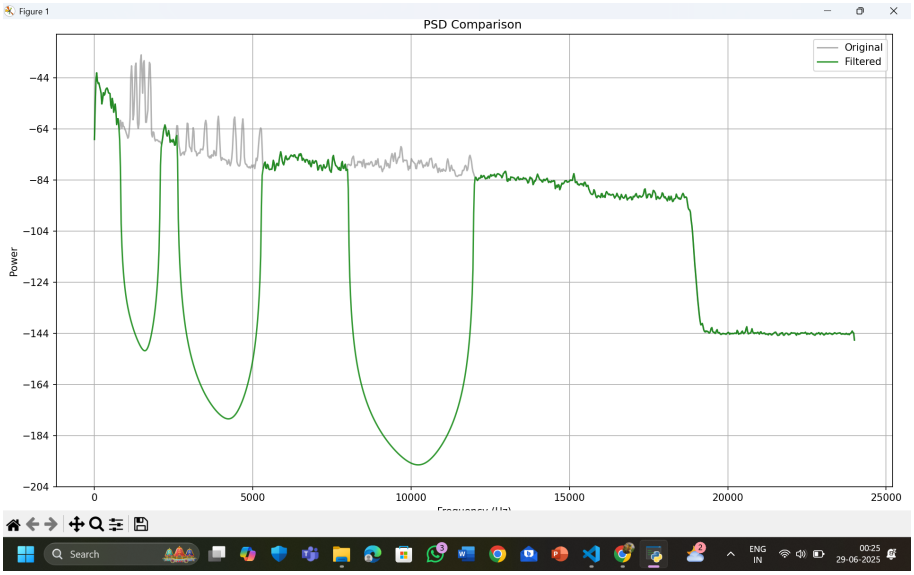

Figure 5: Power Spectral Density Comparison after applying filter

has sharp transitions like rectangular pulse. This will also result in ringing effects in time domain if not approximated precisely so I used 1000 points from impulse response that means a 1000 order polynomial approximation.

- **Since we can never achieve ideal filter using polynomial approximation ploting a pole-zero plot would be difficult unless we approximate it with a real filter.**
- Few other libraries can be used to get non ideal filters and better bode-plots.

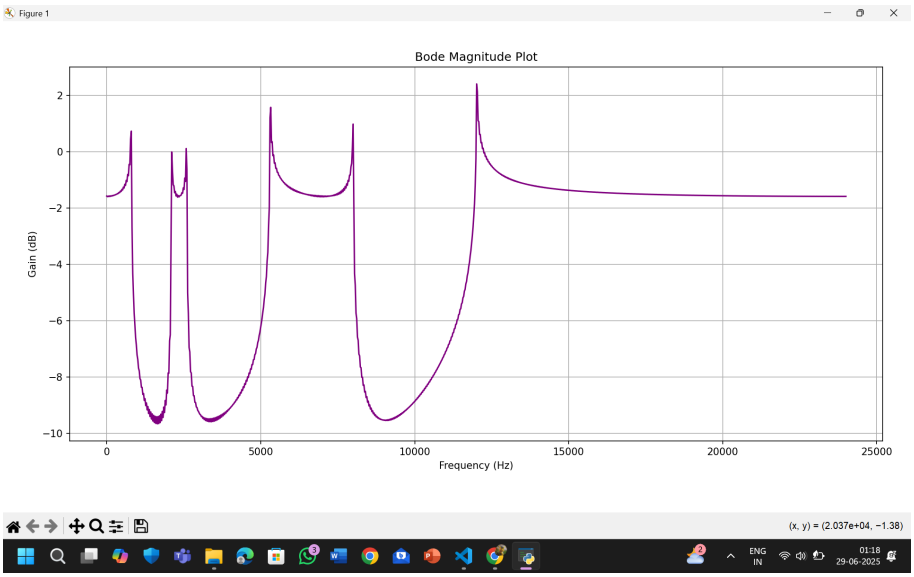

Figure 6: Bode Magnitude Plot

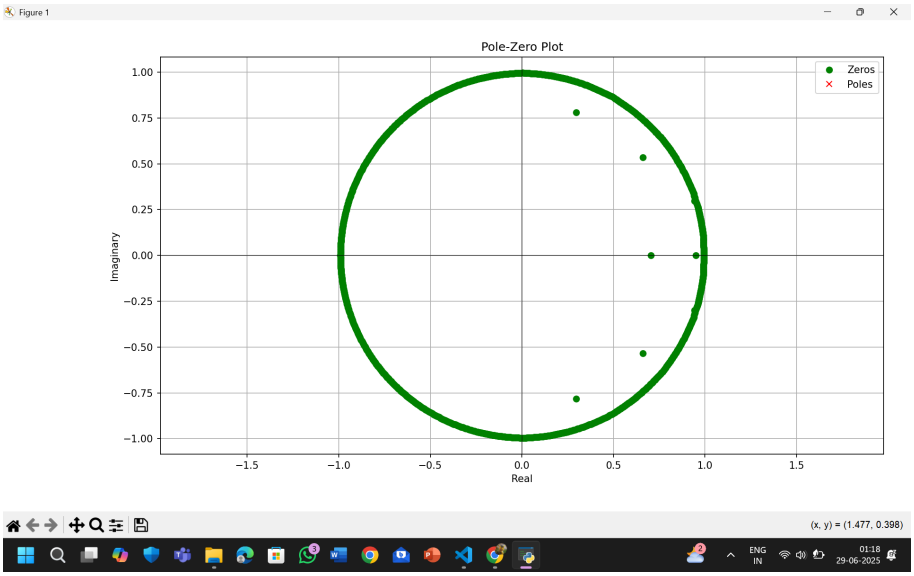

Figure 7: Pole-Zero Plot