End-Term Assignment

Signal, Systems and Networks
EE200

Instructor: Dr. Tushar Sandhan
Submitted by: Kshitij Srivastava

Q1) Frequency Mixer: 'Beauty and the Blur'

1 Introduction

Our visual system interprets images by processing different spatial frequencies—high frequencies capture fine details like edges, while low frequencies convey overall structure and shape. This principle is evident in hybrid images where perception changes with viewing distance, such as seeing Einstein up close and Marilyn Monroe from afar.

This exercise explores spatial frequency analysis using the 2D Discrete Fourier Transform (DFT). It includes visualizing the frequency content of facial images, studying the effects of rotation in the frequency domain, and implementing a frequency mixer that fuses two images—one contributing detail and the other structure—using Python-based Fourier techniques.

2 Objective

The exercise aims to visualize frequency spectra, study the effects of spatial transformations, and design a frequency mixer system to fuse two images—one providing fine details and the other overall structure—using Python and its libraries.

- Compute 2D Discrete Fourier Transforms of the given images.
- Design and apply **frequency-domain masks** (low pass and high pass).
- Combine masked frequency components and reconstruct the fused image.

3 Fourier Transform

The Fourier Transform is a fundamental mathematical tool that decomposes a signal into its constituent frequencies. In image processing, the 2D Discrete Fourier Transform (DFT) enables analysis of spatial frequency content by representing an image in the frequency domain. Low-frequency components capture smooth variations and overall structure, while high-frequency components represent sharp edges and fine details.

Mathematically, for an image $I(x, y)$ of size $M \times N$, the 2D DFT is defined as:

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

The inverse 2D DFT to reconstruct the image is:

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

The result $F(u, v)$ is a complex-valued function:

- **Magnitude** $|F(u, v)|$ — indicates strength of frequency components
- **Phase** $\angle F(u, v)$ — captures positional or spatial information

To implement the 2D Discrete Fourier Transform (DFT) of the input image, we use the `fft2` function from the `numpy.fft` module. This function efficiently converts spatial-domain images into their corresponding frequency-domain representations.

4 Magnitude Spectra

The magnitude spectrum is defined as the absolute value of the Fourier transform:

$$|F(u, v)| = \sqrt{(\mathrm{Re}(F(u, v)))^2 + (\mathrm{Im}(F(u, v)))^2}$$

Due to the wide dynamic range, visualizing this directly often produces dark images. To resolve this, we apply a logarithmic scale:

$$|F(u, v)|_{\mathrm{dB}} = 20 \log_{10} (|F(u, v)| + \varepsilon)$$

Where $\varepsilon = 10^{-5}$ prevents logarithmic singularities and improves visibility of all frequency components. Two types of magnitude spectra were used for visualisation, mainly :

- **Normal Magnitude Spectrum**: The normal (or linear) magnitude spectrum represents the absolute values of the 2D Fourier transform directly without any scaling. While it accurately reflects the distribution of frequency components, the wide range of values causes most high-frequency details to appear very dim. This is because a few low-frequency components dominate the spectrum, making subtle variations difficult to visualize. As a result, the linear spectrum often appears nearly black, especially in the presence of strong low-frequency signals.
- **Logarithmic(dB) Magnitude Spectrum**: The logarithmic (dB) magnitude spectrum is used to compress the dynamic range of the Fourier transform's magnitude values, enhancing visibility across both strong and subtle frequency components. It is computed using the formula:
$$|F(u, v)|_{\mathrm{dB}} = 20 \log_{10} (|F(u, v)| + \varepsilon)$$
  This scaling allows finer details in the frequency domain to become visible, making it easier to interpret both dominant and weak frequencies in an image.

5 Libraries Used

- **NumPy:** Used for efficient numerical computations and to perform the 2D discrete Fourier transform operations.
- **Matplotlib:** Utilized to visualize images and plot magnitude spectra in both linear and logarithmic scales.
- **Pillow (PIL):** Employed for loading, converting, and manipulating input images in various formats for processing.

6 Input Images

- For high frequency characteristics (the details) in the resulting image, choose the cat image.
- For low frequency characteristics (the overall outline/structure) in the resulting image, choose the dog image.

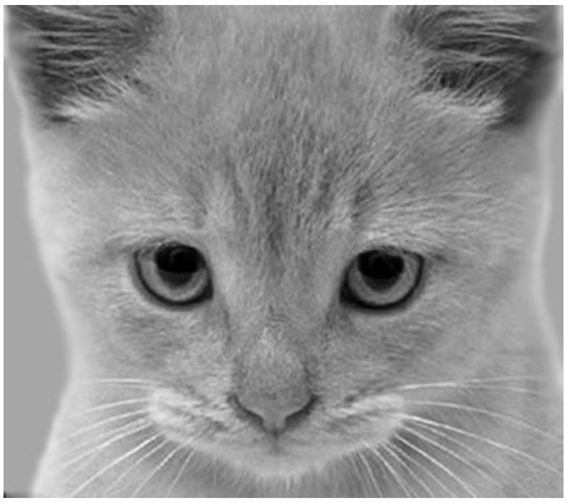
Figure 1: Cat image

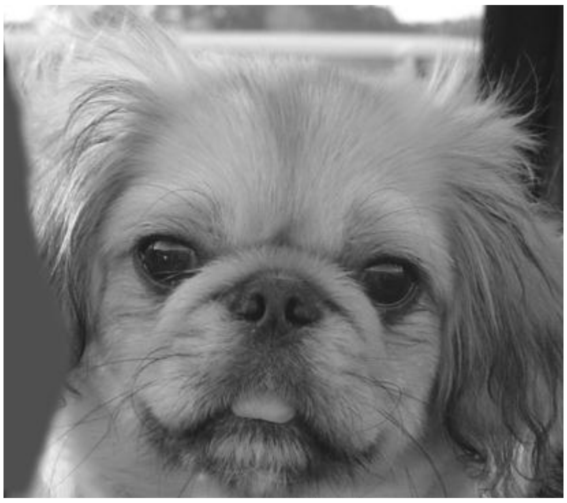
Figure 2: Dog image

7 Spectrum Centering

By default, the low-frequency components (representing the average brightness) obtained from the 2D discrete Fourier transform are located at the corner of the spectrum (as evident from the magnitude spectra). This process, known as spectrum centering, can be achieved by using `fftshift` from the `numpy.fft` module in Python, enabling one to shift the low-frequency components to the centre of the image.

This centered representation makes it easier to observe the spatial frequency distribution, with low frequencies at the center and higher frequencies radiating outward

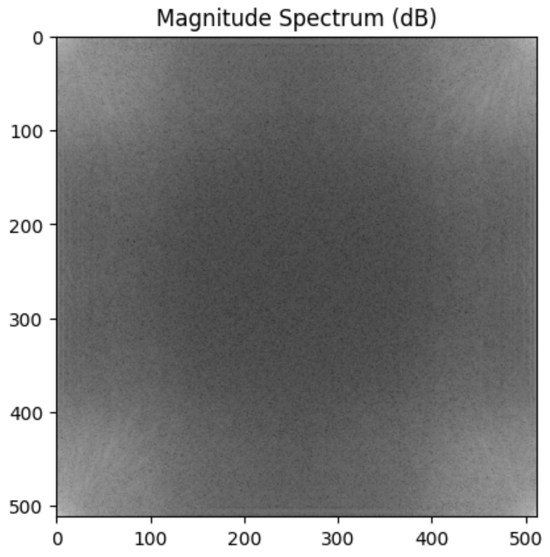

Figure 3: Before centering

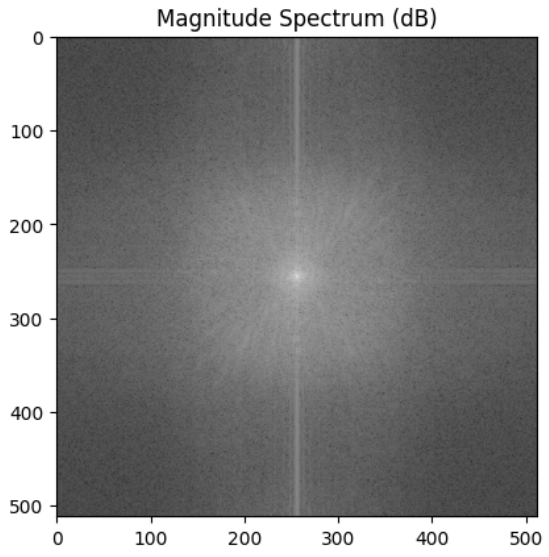

Figure 4: After centering

Figure 5: Spectrum comparison after centering

8 Effect of Image Rotation

To analyze the behaviour of frequency-domain representation, we rotated one of the images by 90° in a counter-clockwise direction and then computed and plotted its 2D fourier transform using the same procedure: `fft2, fftshift` and then logarithmic scaling for the spectrum

Observations:

- The overall pattern of the magnitude spectrum is **preserved** after rotation.
- Every feature in the frequency-domain plot of the rotated image appears **rotated** by exactly 90° compared to the original spectrum.
- The experiment validates a key property of the 2D Fourier Transform: a 90° counter-clockwise rotation in the spatial domain results in a corresponding 90° counter-clockwise rotation in the frequency domain, without altering magnitude values.

9 Implementation

The objective of the frequency mixer is to combine **low frequency** characteristics from one image, contributing to the **overall structure** of the image, and **high frequency characteristics** from the other image, contributing to the **fine details (textural)**.

- The **Dog image** was used as the source for low-frequency components, representing broader shape and structures.
- The **Cat image** was used as the source for high-frequency components, representing edge and fine textures.

Transfer Function (Masks)

- **Low frequency mask**: A circular binary mask that passes only the low-frequency components (i.e., values near the center of the shifted spectrum) and blocks the high-frequency details.
- **High frequency mask**: A binary mask that is the complement of the low-pass mask. It passes high-frequency components (i.e., edges and fine textures) and blocks the structural, low-frequency content.

Approach:

- To separate low- and high-frequency components, a binary circular mask was used, generated using a custom function.
- This function returns a boolean mask with `True` values inside the specified radius from the center frequency and `False` elsewhere.
- For the radius of the mask, multiple values were tested empirically (e.g., 10, 20, 30) to observe how much detail or blurring was introduced.
- A moderate radius (e.g., 14 pixels) was found to balance coarse structure retention in low-pass filtering and fine detail extraction in high-pass filtering.
- Also, weights were added after the application of masks to the respective transforms to each operation of applying the masks and results observed following it, yielding the weights as used in the code.

Steps:

- First computed the 2D DFT of both the images using `fft2`, shifted using `fftshift`.
- Generated masks centered at (N/2, N/2) where N is the dimension of the Fourier transform.
- A circular region of radius r was used to define the low-pass region.
- The high frequency mask was obtained from the negation of the low frequency mask.
- Then fused the images applying the masks respectively to each of them and performed the inverse Fourier transform to generate the fused image as a result using `ifft2` and `ifftshift`

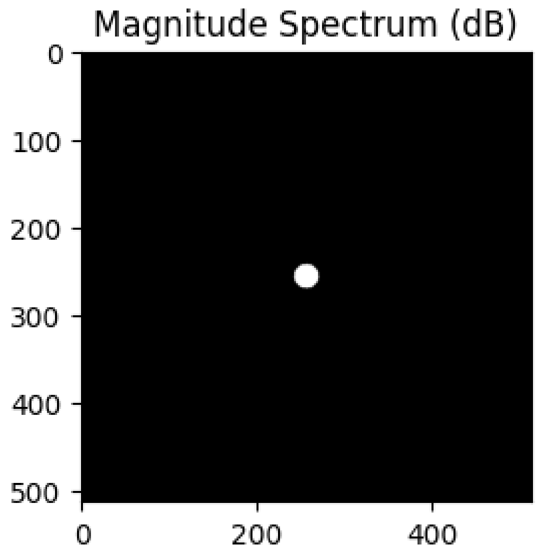

Figure 6: Low-Frequency mask

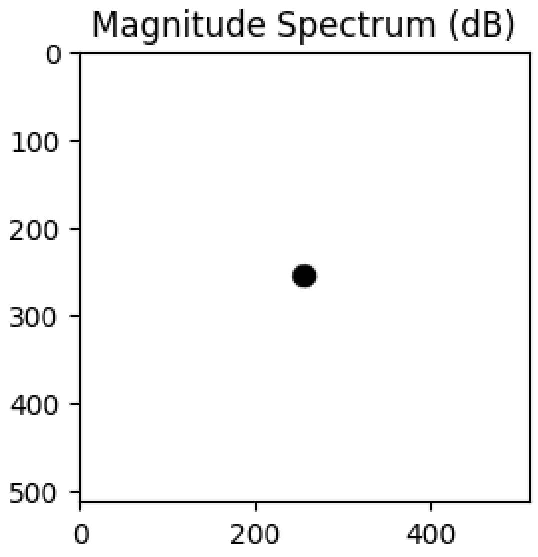

Figure 7: High-Frequency mask

Figure 8: Magnitude Spectra of Masks

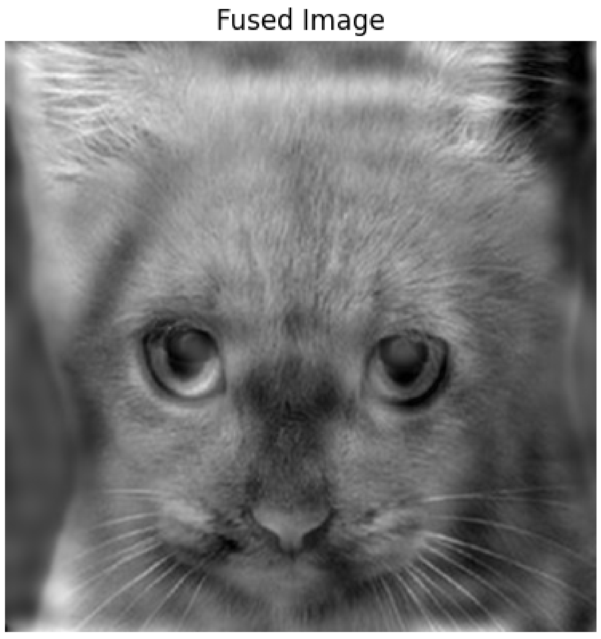

Figure 9: Fused Image: Combination of Dog (Low-frequency) and Cat (High-Frequency)

Q2) Frequency De-Mixer: 'Unwanted Solo'

1 Introduction

This exercise addresses the problem of unwanted instrumental beats disrupting the harmonic structure of a song. These elements, whether due to production errors or post-processing issues, interfere with the intended musical experience. To resolve this, we employ frequency-domain signal processing techniques to analyze and isolate the disruptive components based on their spectral signatures. By designing and applying targeted filters, we aim to suppress the unwanted content while preserving the integrity of the original audio.

2 Objective

The primary objective of this project is to design a frequency-selective filtering system—referred to as a frequency demixer—to suppress unwanted instrumental elements from a corrupted audio track. Specifically, the goals are to:

- Identify and isolate the frequency range of the unwanted solo instrumental component.
- Design **appropriate filters** (e.g., band-stop, high/low-pass) in Python to attenuate the interfering frequencies.
- Analyze the frequency characteristics of the designed filter using tools such as Bode plots, power spectral density (PSD), and pole-zero plots.
- Restore the original musical structure as faithfully as possible.

3 Libraries Used

- **NumPy:** Used for efficient numerical computations and to perform the 2D discrete Fourier transform operations.
- **Matplotlib:** Utilized to visualize images and plot magnitude spectra in both linear and logarithmic scales.
- **Librosa:** A high-level audio analysis library, well-suited for music and speech processing. It was used to load audio files and perform waveform analysis and resampling.
- **SciPy.signal:** The `scipy.signal` module provides tools for designing and analyzing filters, computing frequency responses (Bode plots), and generating pole-zero plots.
- **Soundfile:** Used for reading and writing audio files in high quality. This library allowed saving the restored audio output after filtering in .wav format.

4 Audio Signal Analysis

The signal was analyzed in the time domain by plotting its waveform. This provided insight into the amplitude variations over time. The corresponding frequency domain representation was obtained by computing and plotting the magnitude spectrum using the Fourier Transform.

To visualize how the frequency content of the signal evolves, a spectrogram was also generated. The spectrogram offered a time-frequency representation that helped identify dominant frequency bands and regions potentially affected by noise. These initial visualisations formed the foundation for designing appropriate filters to manipulate or remove specific frequency components from the signal.

Observation from Analysis

- The time-domain waveform showed complex amplitude patterns with sharp peaks, hinting at the presence of a dominant, possibly unwanted, audio component.
- The spectrogram (log scale) revealed a bright, narrow horizontal band between 1024 Hz and 2048 Hz, active throughout the track.
- Most other frequencies showed moderate to low energy, consistent with background vocals or instrumentation.
- The magnitude spectrum confirmed concentrated energy between 1100 Hz and 1800 Hz, sharply distinct from the surrounding bands. This frequency band likely represents a solo instrumental overlay that is not part of the original mix.

5 Filter Design and Implementation

Following the initial analysis of the audio signal, targeted filtering of the 1100-1800 Hz was required to suppress undesired components while preserving the quality of the underlying signal. To achieve this, a flexible digital filter framework was developed using Python, incorporating both FIR (Finite Impulse Response).

Finite Impulse Response (FIR) Filters: These filters were constructed using the window method with **blackman** window to ensure effective sidelobe attenuation. FIR filters offer the benefit of a strictly linear phase response, which is particularly desirable for audio processing where phase distortions must be minimized.

Filter Analysis

A series of frequency-selective filters were applied to analyze and manipulate different components of the audio signal. Each filter helped in isolating or suppressing specific elements, leading to a better understanding of the audio composition and guiding noise removal decisions.

Low Pass Filter

- Applying a low-pass filter with a cutoff at 1000 Hz resulted in the removal of most high-frequency content from the audio track.
- The filtered output was dominated by vocal elements, while high-frequency instruments, including the unwanted flute, were effectively attenuated.
- This indicates that the vocals are primarily concentrated below 1000 Hz, whereas the interfering flute component lies in the higher frequency range, validating the target band for suppression.

High Pass Filter

- Applying a high-pass filter with a cutoff at 1500 Hz to retain the high frequency characteristics, including characteristics of harmonics, while some of the piccolo flute itself.
- The resulting audio preserved only the instrumental components, while the vocals—primarily located in the lower frequency range—were effectively suppressed.
- This confirms that the unwanted piccolo flute and other instrumental noise are concentrated above 1500 Hz, supporting the choice of a high-frequency suppression strategy.

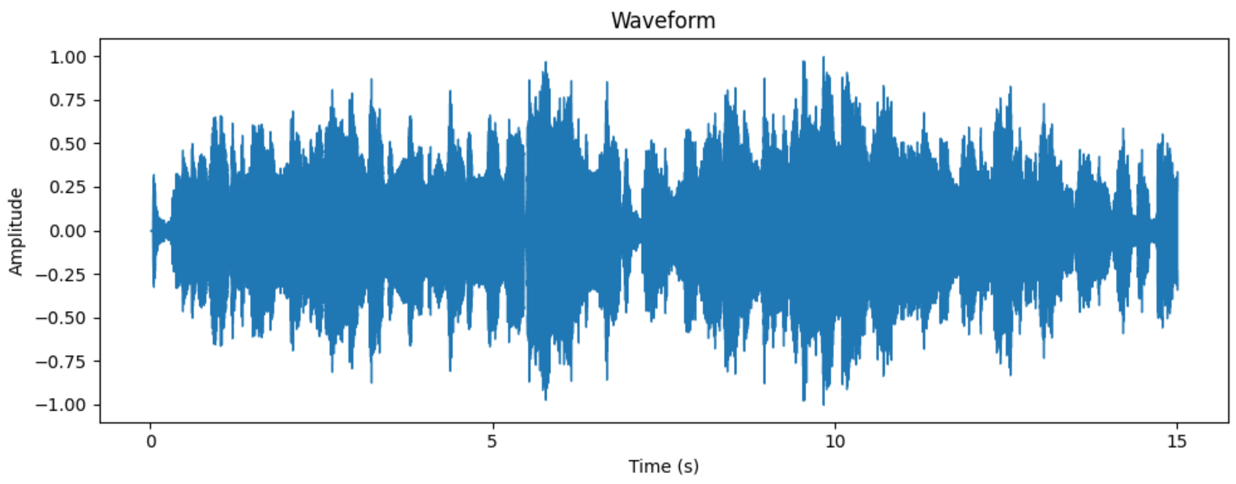

Figure 10: Time-Domain Waveform of the sample audio

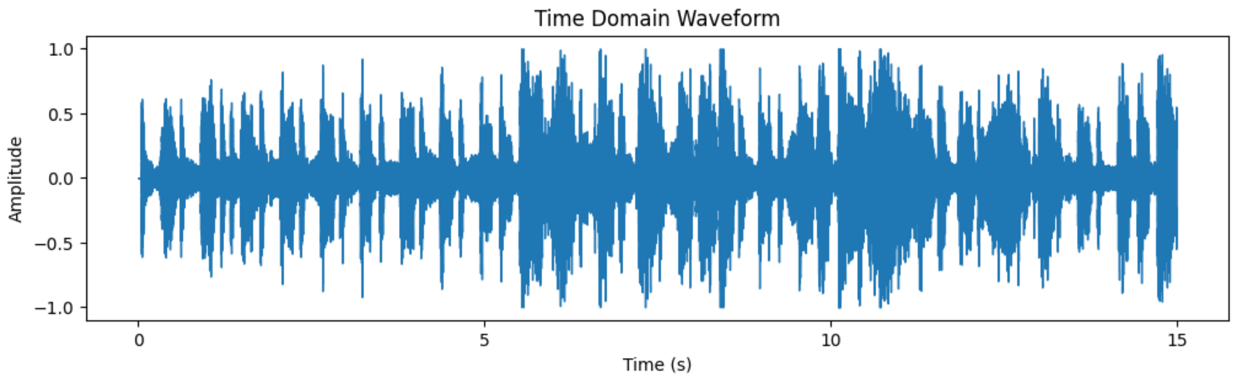

Figure 11: Time-Domain Waveform of the recovered audio

Band-Stop Filter

- A band-stop filter was applied to suppress the frequency range between 1000 Hz and 6500 Hz to suppress a broader mid-frequency range.
- The unwanted flute component was significantly attenuated, while the vocals and background music were largely preserved with minimal distortion.
- This confirms that the targeted frequency band effectively isolates the intrusive element, and that band-stop filtering can restore the audio quality by removing noise without compromising key musical elements.

Summary

- Each filter contributed uniquely to understanding the spectral structure of the audio.
- The band-stop filter offered a strong baseline for noise suppression and led to effective demixing with a favourable balance between noise reduction and audio fidelity.