M.Sc. (Five Year Integrated) in Computer Science

(Artificial Intelligence & Data Science)

Fourth Semester

Assignment

23-813-0403: DIGITAL SIGNAL PROCESSING

Submitted in partial fulfillment
of the requirements for the award of degree in
Master of Science (Five Year Integrated)
in Computer Science (Artificial Intelligence & Data Science) of
Cochin University of Science and Technology (CUSAT)
Kochi

Submitted by

MARIYA JYOTHY
(81323014)

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI-682022

MARCH 2025

Contents

1 Parseval's Theorem (Energy Conservation Law)

Parseval's theorem states that the total energy (sum of squared magnitudes) of a signal in the spatial domain is equal to the total energy of the signal in the frequency domain. In mathematical terms, for an image $f(x, y)$ and its 2D Fourier Transform $F(u, v)$, the theorem can be written as:

$$\sum_{x,y} |f(x, y)|^2 = \frac{1}{M \cdot N} \sum_{u,v} |F(u, v)|^2 \tag{1}$$

where:

- $f(x, y)$ is the image in the spatial domain.
- $F(u, v)$ is the Fourier Transform of the image.
- M and N are the dimensions of the image (height and width).
- $|f(x, y)|^2$ represents the squared magnitude of the image pixel values.
- $|F(u, v)|^2$ represents the squared magnitude of the Fourier transform coefficients.

This equation shows that the total energy (sum of squared magnitudes) in the spatial domain is equal to the energy in the frequency domain, confirming the energy conservation law for Fourier Transforms.

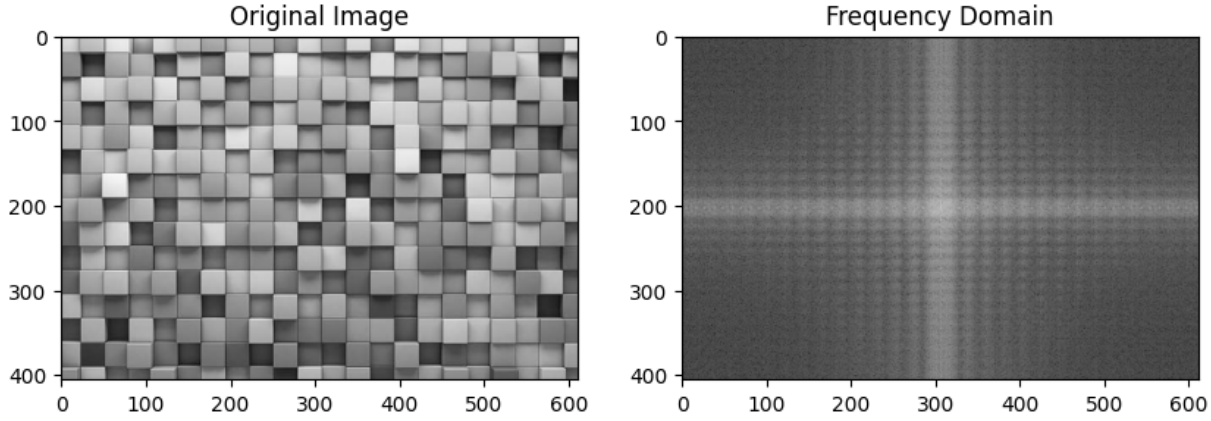
1.1 CODE

```
from google.colab import files
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift
import matplotlib.image as mpimg

# Upload image
uploaded = files.upload()

# Assuming you upload a file named 'test.png', use the uploaded file name
image_path = list(uploaded.keys())[0]  # Get the first uploaded file name

# Load the image (if it's a color image, it will be converted to grayscale)
img = mpimg.imread(image_path)
```

```
# If the image is colored (e.g., RGB), convert it to grayscale
if img.ndim == 3:
    img = np.mean(img, axis=-1)  # Convert to grayscale by averaging the channels

# Compute 2D Fourier Transform (Frequency domain)
F_uv = fft2(img)

# Shift the zero-frequency component to the center
F_uv_shifted = fftshift(F_uv)

# Calculate energy in the spatial domain
energy_spatial = np.sum(np.abs(img)**2)

# Calculate energy in the frequency domain
energy_frequency = np.sum(np.abs(F_uv_shifted)**2) / img.size

# Print the results
print(f"Energy in spatial domain: {energy_spatial}")
print(f"Energy in frequency domain: {energy_frequency}")

# Verify Parseval's theorem (energy conservation)
assert np.isclose(energy_spatial, energy_frequency),
"Energy conservation does not hold!"

# Optionally, visualize the image and its frequency domain representation
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(img, cmap='gray')
plt.title('Original Image')

plt.subplot(1, 2, 2)
# Visualize the log of the absolute value of the shifted frequency domain
#Log scale for better visualization
plt.imshow(np.log(np.abs(F_uv_shifted) + 1), cmap='gray')
plt.title('Frequency Domain')

plt.show()
```

1.2 RESULTS

```
• testImage.jpg(image/jpeg) - 57207 bytes, last modified: 2/6/2025 - 100% done
Saving testImage.jpg to testImage (3).jpg
Energy in spatial domain: 5388984599.666666
Energy in frequency domain: 5388984599.666671
```



1.3 CONCLUSION

- **Energy in the spatial domain:** Represents the total energy of the image (sum of pixel intensities squared).
- **Energy in the frequency domain:** Represents the total energy after transforming the image to the frequency domain and summing the squared magnitudes of the Fourier coefficients.
- **Verification:** If the assertion doesn't raise an error, the energy conservation law is satisfied.

2 Magnitude Spectrum Analysis

The magnitude spectrum is a critical component of Fourier Transform analysis, representing the intensity of different frequency components of an image. By applying the 2D Fourier Transform to an image, we can observe the distribution of frequency components in both normal and centered frequency domains. The normal frequency spectrum places the zero-frequency component at the top-left corner, while the centered frequency spectrum shifts the zero-frequency component to the center of the image. In this analysis, a logarithmic scale is applied to the magnitude spectrum to enhance the visibility of both low and high-frequency components.

$$\text{Magnitude Spectrum} = |F(u, v)| \tag{2}$$

where:

- $F(u, v)$ is the Fourier Transform of the image.
- $|F(u, v)|$ represents the magnitude of the Fourier Transform coefficients, capturing the intensity of different frequency components.

This analysis allows us to visually explore the frequency characteristics of the image by examining both the normal and centered magnitude spectrums.

2.1 CODE

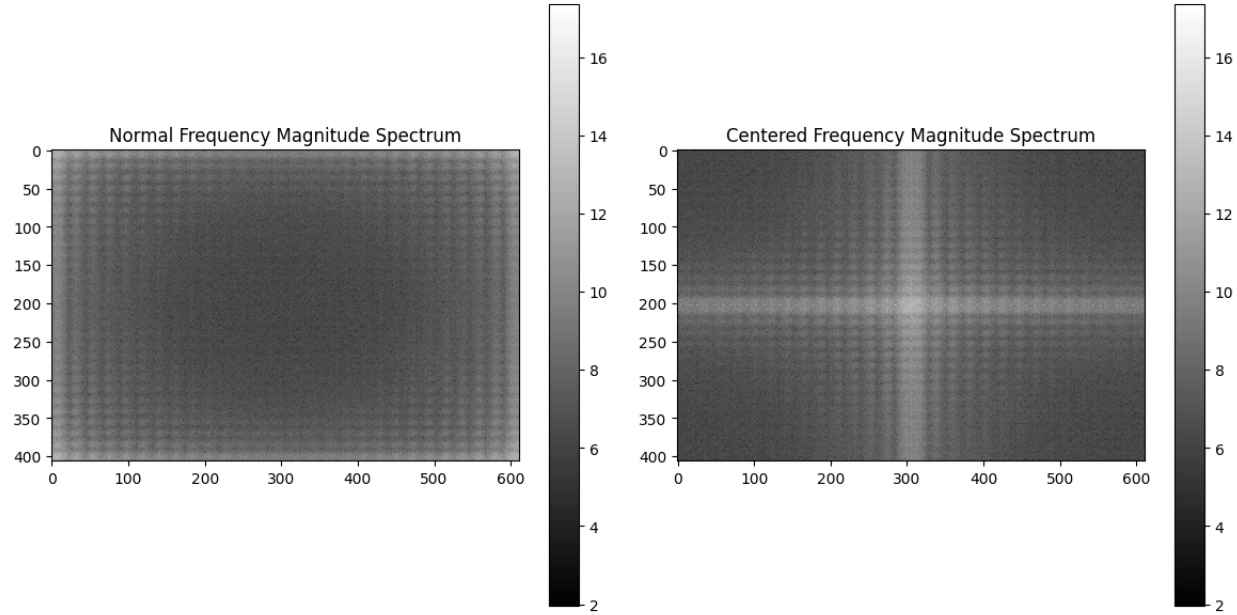
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift
import matplotlib.image as mpimg
from google.colab import files

# Upload image
uploaded = files.upload()

# Assuming you upload a file named 'test.png', use the uploaded file name
image_path = list(uploaded.keys())[0]  # Get the first uploaded file name

# Load the image
img = mpimg.imread(image_path)

# If the image is colored (e.g., RGB), convert it to grayscale
```

```
if img.ndim == 3:
    img = np.mean(img, axis=-1)  # Convert to grayscale by averaging the channels

# Compute 2D Fourier Transform (Frequency domain)
F_uv = fft2(img)

# Compute magnitude spectrum (normal frequency rectangle, no shift)
magnitude_spectrum_normal = np.abs(F_uv)

# Compute magnitude spectrum (centered frequency rectangle,
shift zero-frequency component)
F_uv_shifted = fftshift(F_uv)
magnitude_spectrum_centered = np.abs(F_uv_shifted)

# Plot both the magnitude spectrums
plt.figure(figsize=(12, 6))

# Plot the normal (uncentered) frequency spectrum
plt.subplot(1, 2, 1)
# Log scale for better visualization
plt.imshow(np.log(magnitude_spectrum_normal + 1), cmap='gray')
plt.title('Normal Frequency Magnitude Spectrum')
plt.colorbar()

# Plot the centered frequency spectrum
plt.subplot(1, 2, 2)
# Log scale for better visualization
plt.imshow(np.log(magnitude_spectrum_centered + 1), cmap='gray')
plt.title('Centered Frequency Magnitude Spectrum')
plt.colorbar()

plt.tight_layout()
plt.show()
```

2.2 RESULTS

- **testImage.jpg**(image/jpeg) - 57207 bytes, last modified: 2/6/2025 - 100% done

```
Saving testImage.jpg to testImage (4).jpg
```



2.3 CONCLUSION

- **Magnitude Spectrum Normal:** Represents the frequency components of the image with the zero-frequency component at the top-left corner of the image.
- **Magnitude Spectrum Centered:** Represents the frequency components with the zero-frequency component shifted to the center of the image.
- **Logarithmic Scaling:** Used to enhance the visibility of both low and high-frequency components in the spectrum.
- **Visualization:** By comparing both frequency spectrums, we can analyze the frequency distribution and the image's overall frequency characteristics.