Assignment 4

Spring 2025

CSE-408 Digital Image Processing

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

Submitted to:

Engr. Mehran Ahmad

Date:

25th June 2025

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Question 1: Frequency Domain Sharpening (Highpass Filtering)

Apply and analyze **sharpening filters** in the frequency domain using Python/ MATLAB. Observe how edges and details are enhanced using different high-pass filtering approaches.

Instructions:

1. Use the same grayscale image as in Assignment 03.
2. Implement and apply the following **Highpass Filters** in the frequency domain:
    - Ideal Highpass Filter (IHPF)
    - Butterworth Highpass Filter (BHPF) with order = 2
    - Gaussian Highpass Filter (GHPF)
3. Use a common cutoff frequency $D0$ = 50
4. Perform inverse FFT to reconstruct the image.
5. Display and compare:
    - Original image
    - Sharpened outputs
    - Their corresponding magnitude spectra

Analysis:

- Compare edge enhancement and noise amplification.
- Which filter gives sharp results without introducing strong artifacts? Explain.

Code:

Assignment4.py × Assignment3.py

Assignment4.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt
from skimage import data

# Load grayscale image
image = data.camera()

# Step 1: Fourier Transform
dft = np.fft.fft2(image)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20 * np.log(np.abs(dft_shift) + 1)

# Step 2: Prepare Distance Matrix
M, N = image.shape
u = np.arange(M)
v = np.arange(N)
U, V = np.meshgrid(u, v, indexing='ij')
D = np.sqrt((U - M//2)**2 + (V - N//2)**2)

# Step 3: Define Highpass Filters
D0 = 50
n = 2  # Butterworth order

# Ideal Highpass Filter
H_ihpf = np.ones((M, N))
```

Assignment4.py > ...

```python
H_ihpf = np.ones((M, N))
H_ihpf[D <= D0] = 0

# Butterworth Highpass Filter
H_bhpf = 1 / (1 + (D0 / (D + 1e-5))**(2 * n))  # Add small epsilon to avoid div/0

# Gaussian Highpass Filter
H_ghpf = 1 - np.exp(-(D**2) / (2 * D0**2))

# Step 4: Apply Filters in Frequency Domain
filtered_ihpf = dft_shift * H_ihpf
filtered_bhpf = dft_shift * H_bhpf
filtered_ghpf = dft_shift * H_ghpf

# Step 5: Compute magnitude spectra of filtered images
mag_ihpf = 20 * np.log(np.abs(filtered_ihpf) + 1)
mag_bhpf = 20 * np.log(np.abs(filtered_bhpf) + 1)
mag_ghpf = 20 * np.log(np.abs(filtered_ghpf) + 1)

# Step 6: Inverse FFT to get sharpened images
sharpened_ihpf = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_ihpf)))
sharpened_bhpf = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_bhpf)))
sharpened_ghpf = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_ghpf)))

# ===========================
```

```python
# Figure 1: Original Image + Spectrum
# ============================
plt.figure(figsize=(10, 5))

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

plt.subplot(1, 2, 2)
plt.imshow(magnitude_spectrum, cmap='viridis')
plt.title('Original Spectrum')
plt.axis('off')

plt.tight_layout()


# ============================
# Figure 2: IHPF Image + Spectrum
# ============================
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(sharpened_ihpf, cmap='gray')
```

```python
plt.title('ILPF Filtered')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(mag_ihpf, cmap='viridis')
plt.title('ILPF Spectrum')
plt.axis('off')

plt.tight_layout()

# ============================
# Figure 3: BHPF Image + Spectrum
# ============================
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(sharpened_bhpf, cmap='gray')
plt.title('BLPF Filtered (n=2)')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(mag_bhpf, cmap='viridis')
plt.title('BLPF Spectrum')
plt.axis('off')

plt.tight_layout()

# ============================
# Figure 4: GHPF Image + Spectrum
# ============================
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(sharpened_ghpf, cmap='gray')
plt.title('GLPF Filtered')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(mag_ghpf, cmap='viridis')
plt.title('GLPF Spectrum')
plt.axis('off')

plt.tight_layout()

# Show all figures
plt.show()

```

Output:

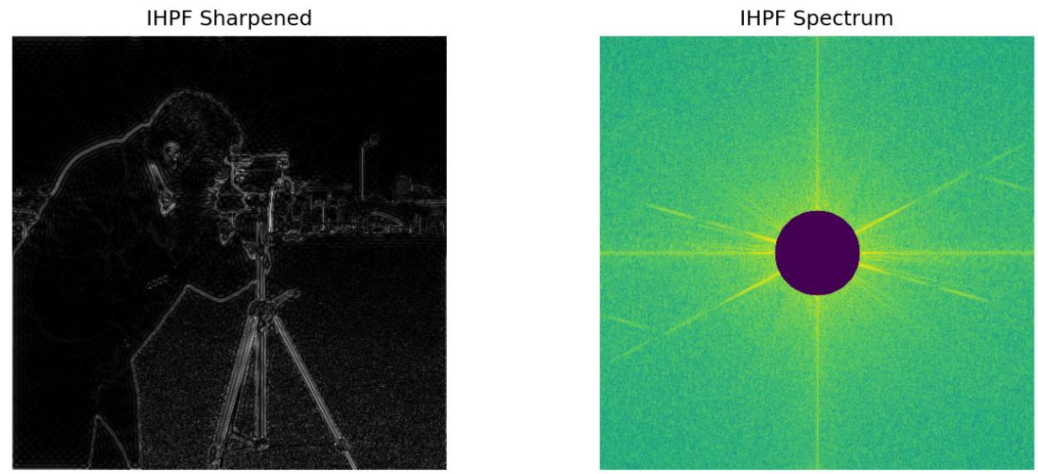

Figure 1: Ideal Highpass Filter (IHPF) Output and Spectrum

The left image shows the result of applying an Ideal Highpass Filter with cutoff D0=50. This filter removes all low-frequency components below the cutoff sharply, enhancing edges and fine details. The right image displays the filtered magnitude spectrum, showing a ring-like pattern due to the ideal circular cutoff.

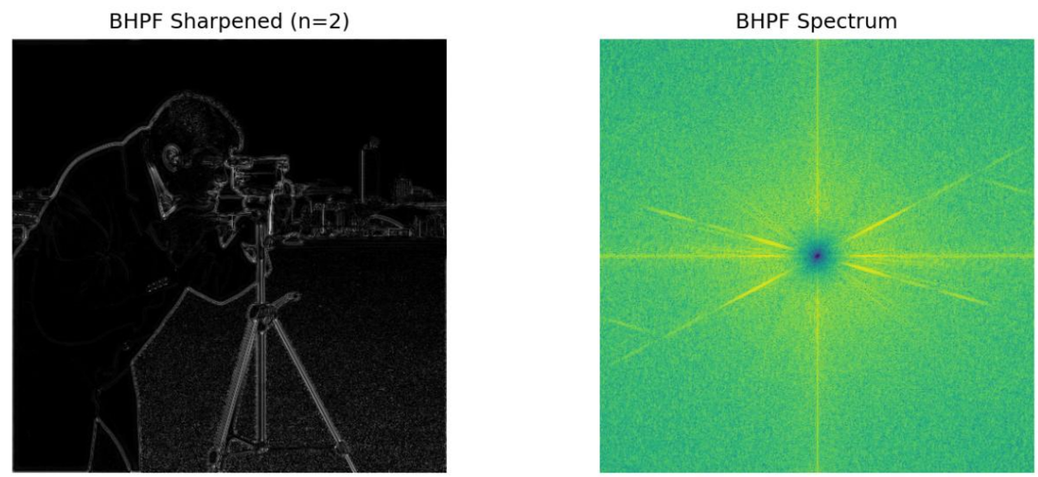

Figure 2: Butterworth Highpass Filter (BHPF) Output and Spectrum

This figure shows the result of applying a second-order Butterworth Highpass Filter. Compared to IHPF, BHPF provides a smoother transition between low and high frequencies, avoiding sharp artifacts. The corresponding spectrum shows a gradual suppression of central (low-frequency) components.

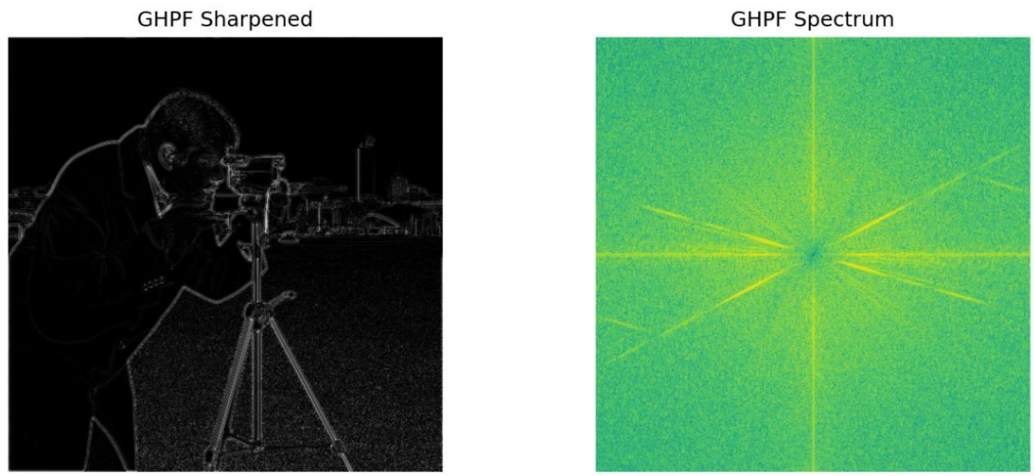

Figure 3: Gaussian Highpass Filter (GHPF) Output and Spectrum

The Gaussian Highpass Filter offers the smoothest frequency attenuation among the three. The output image exhibits edge enhancement with minimal ringing artifacts. The spectrum on the right shows a soft, radial attenuation around the center, consistent with the Gaussian function.

Analysis:

Ideal High-Pass Filter (IHPF)

The sharpened image using the Ideal High-Pass Filter exhibits strong edge enhancement. However, this comes at the cost of introducing ringing artifacts, especially around high-contrast boundaries. The frequency spectrum of the IHPF clearly shows a hard cutoff, with the central low-frequency region completely removed. This abrupt transition causes the Gibbs phenomenon, leading to unnatural visual artifacts in the filtered image.

Butterworth High-Pass Filter (BHPF)

The image processed with the Butterworth High-Pass Filter shows smoother and more natural sharpening. It enhances edges without introducing the harsh ringing seen with IHPF. The filter's spectrum illustrates a gradual attenuation of low frequencies, avoiding the sharp cutoff and thereby reducing spatial domain distortions. This smooth frequency transition helps maintain visual quality while still sharpening the image effectively.

Gaussian High-Pass Filter (GHPF)

The Gaussian High-Pass Filter provides the most balanced and visually pleasing result among the three. It gently enhances image details and edges without introducing significant artifacts or distortions. The associated frequency spectrum shows an even smoother transition, reflecting the exponential decay of the Gaussian function. This leads to a natural-looking enhancement, preserving both fine detail and overall image coherence.

Conclusion

Among the three high-pass filters, the Gaussian High-Pass Filter (GHPF) delivers the best trade-off between edge enhancement and artifact avoidance. While the Ideal filter is aggressive and introduces noticeable ringing, and the Butterworth filter offers a compromise, the GHPF achieves visually appealing results with minimal side effects. For applications requiring both detail sharpening and artifact-free output, GHPF is the most effective choice.