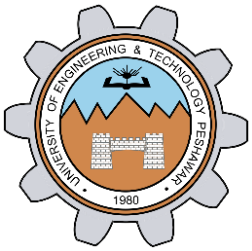
Assignment 3



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 Smoothing (Lowpass Filtering)

Apply and compare frequency domain **smoothing filters** on a grayscale image using Python/ MATLAB. Understand the behavior of each filter in terms of noise reduction and image blurring.

Instructions:

1. Implement and apply the following **Lowpass Filters** in the frequency domain:
    - Ideal Lowpass Filter (ILPF)
    - Butterworth Lowpass Filter (BLPF) with order = 2
    - Gaussian Lowpass Filter (GLPF)
2. Cutoff frequency $D0$ = 50
3. Perform inverse FFT to get the filtered image.
4. Display and compare:
    - Original image
    - Filtered outputs
    - Their corresponding magnitude spectra

Analysis:

- Compare the visual smoothness and edge preservation.
- Discuss which filter best reduces noise and why.

Code:

Assignment4.py Assignment3.py

I: > Users > Engineer > Documents > GitHub > UET_CSE_DataPack4 > 8thSemester > DIP

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

# Load image and perform FFT
image = data.camera()
dft = np.fft.fft2(image)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20 * np.log(np.abs(dft_shift) + 1)

# Create 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)

# Filter settings
D0 = 50
n = 2
```

```python
# Define Filters
H_ideal = np.zeros((M, N))
H_ideal[D <= D0] = 1
H_butter = 1 / (1 + (D / D0)**(2 * n))
H_gauss = np.exp(-(D**2) / (2 * D0**2))

# Apply filters
filtered_dft_ideal = dft_shift * H_ideal
filtered_dft_butter = dft_shift * H_butter
filtered_dft_gauss = dft_shift * H_gauss

# Inverse FFTs
filtered_ideal = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_dft_ideal)))
filtered_butter = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_dft_butter)))
filtered_gauss = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_dft_gauss)))

# Filtered spectra
magnitude_ideal = 20 * np.log(np.abs(filtered_dft_ideal) + 1)
magnitude_butter = 20 * np.log(np.abs(filtered_dft_butter) + 1)
magnitude_gauss = 20 * np.log(np.abs(filtered_dft_gauss) + 1)
```

```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: ILPF Image + Spectrum
# ============================
plt.figure(figsize=(10, 5))

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

```
plt.imshow(filtered_ideal, cmap='gray')
plt.title('ILPF Filtered')
plt.axis('off')

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

plt.tight_layout()

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

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

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

plt.tight_layout()

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

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

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

plt.tight_layout()

# Show all figures
plt.show()
```

Output:

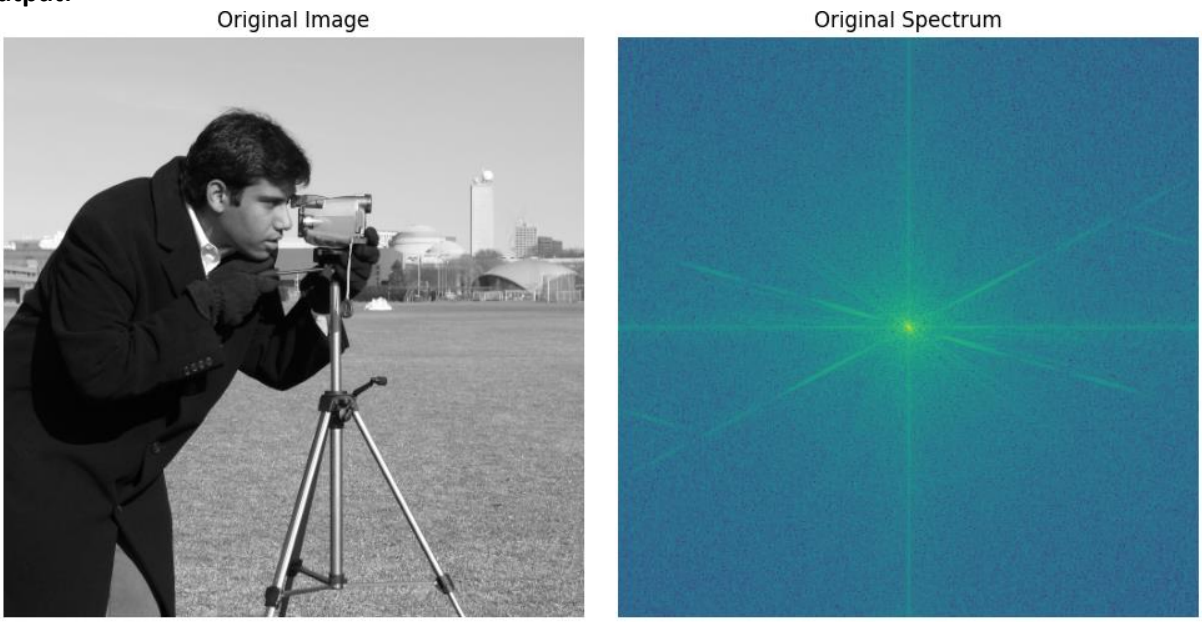


Figure 1: Original Image and Spectrum

The original grayscale image (Cameraman) and its Fourier magnitude spectrum, showing high-frequency components concentrated around the corners.

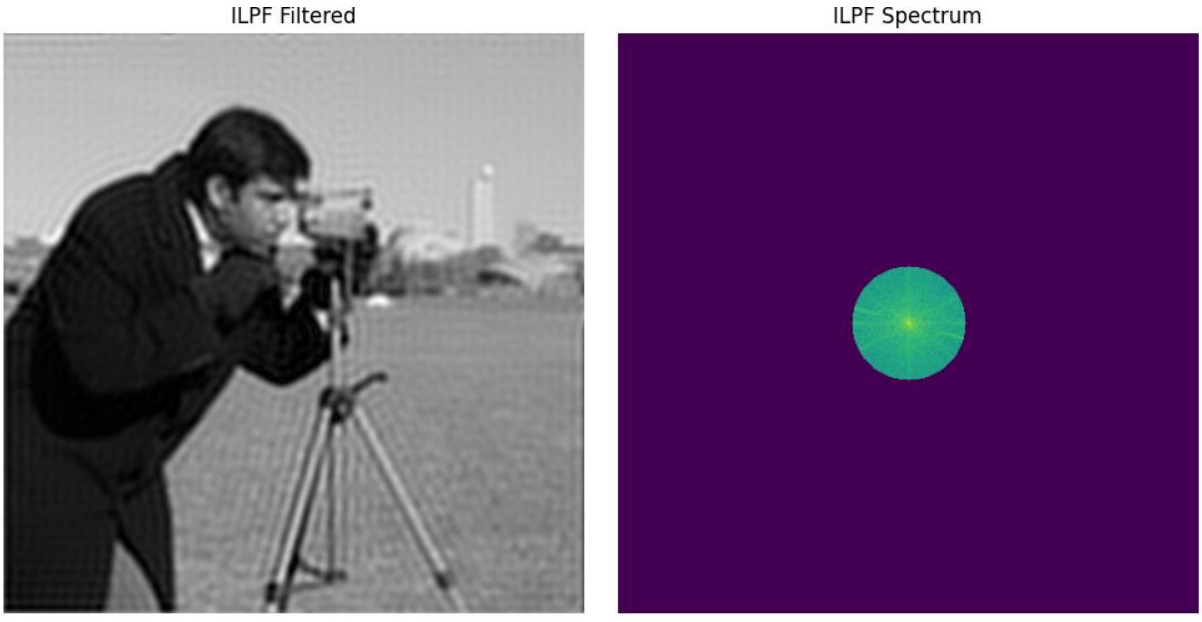


Figure 2: ILPF Filtered Image and Spectrum

Result of applying the Ideal Lowpass Filter (cutoff $D_0 = 50$). The image shows moderate smoothing with noticeable ringing artifacts around edges due to the abrupt frequency cutoff.

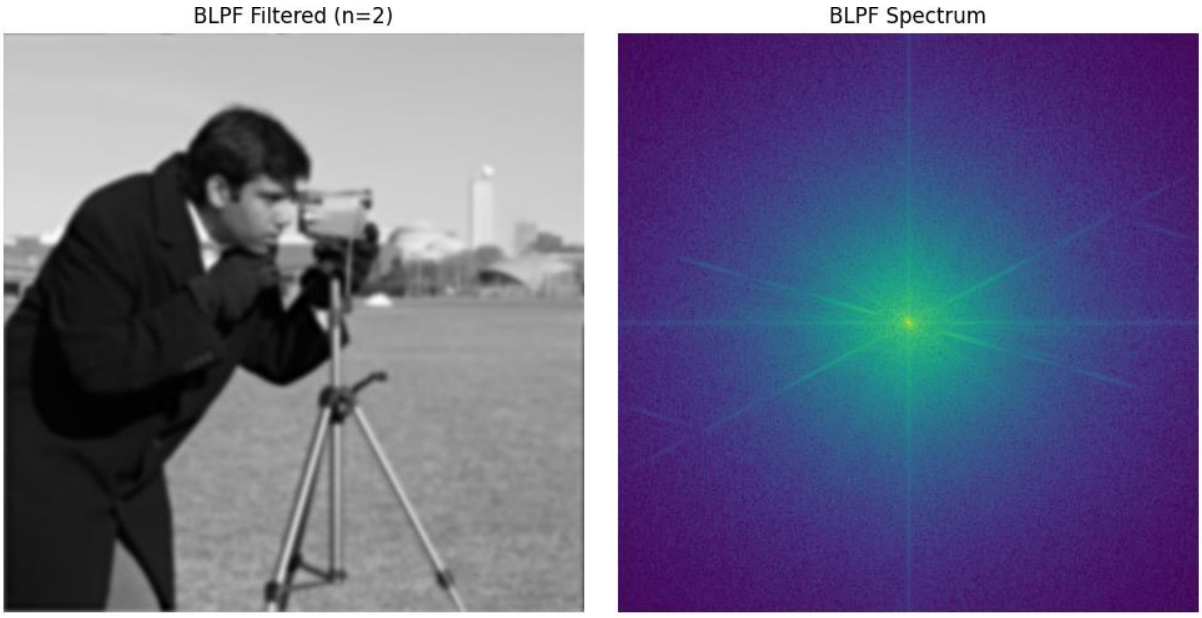


Figure 3: BLPF Filtered Image and Spectrum
Result of applying the Butterworth Lowpass Filter of order 2 (cutoff D_o = 50). The image is smooth, with better edge preservation than ILPF and reduced ringing artifacts.

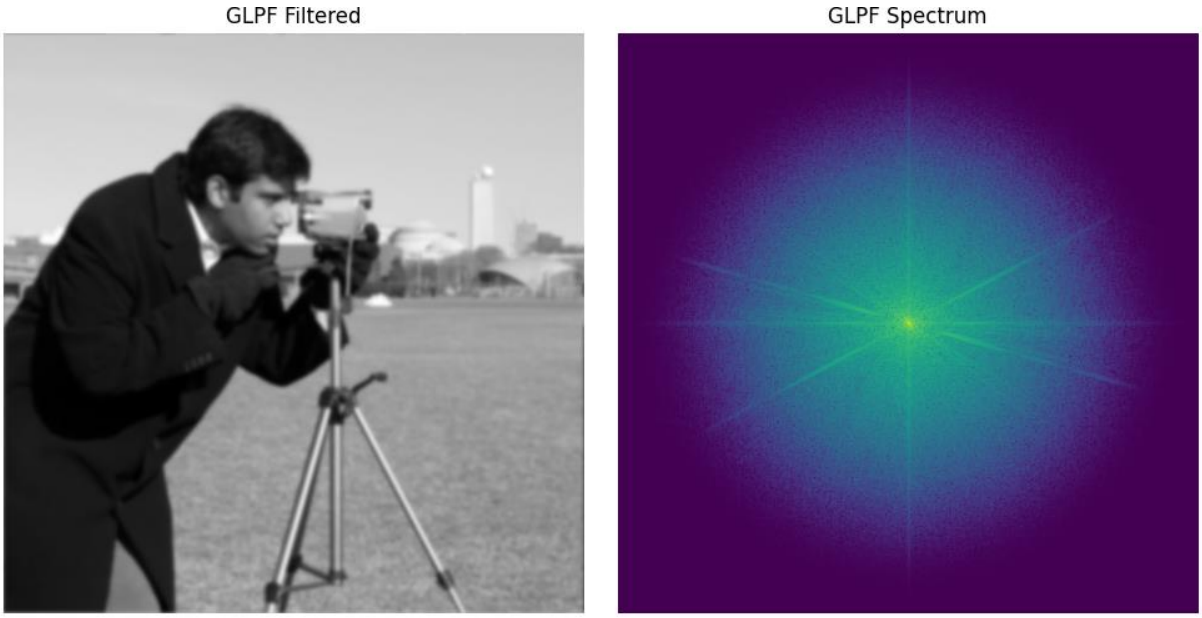


Figure 4: GLPF Filtered Image and Spectrum
Result of applying the Gaussian Lowpass Filter (cutoff D_o = 50). The image is the smoothest, with minimal artifacts and excellent edge retention, showing a natural attenuation of high frequencies.

Analysis:
This analysis evaluates three frequency-domain lowpass filters—Ideal (ILPF), Butterworth (BLPF), and Gaussian (GLPF)—based on smoothness, edge preservation, and noise reduction.

1. Ideal Lowpass Filter (ILPF)

The Ideal Lowpass Filter provides moderate smoothness but performs poorly in preserving edges. It introduces noticeable ringing artifacts around sharp transitions due to its abrupt frequency cutoff (a result of the Gibbs phenomenon). While it reduces high-frequency noise reasonably well, the visible artifacts make it less suitable for applications requiring clean visual quality.

2. Butterworth Lowpass Filter (BLPF)

The Butterworth filter delivers higher smoothness than ILPF and offers better edge preservation by avoiding sudden changes in the frequency domain. Its gradual transition helps reduce ringing effects. It effectively suppresses noise, although some image details may be lost due to its strong lowpass behavior.

3. Gaussian Lowpass Filter (GLPF)

The Gaussian filter achieves the smoothest output among the three. It preserves edges effectively while reducing high-frequency noise without introducing ringing artifacts. Its exponential decay in the frequency response allows for natural attenuation, resulting in clean and visually pleasing output with minimal distortion.

Best Filter

Among the three, the Gaussian Lowpass Filter offers the best overall performance. It achieves excellent noise reduction while maintaining edge fidelity and produces the cleanest result without noticeable artifacts. This makes it the most suitable choice for lowpass filtering in image processing tasks.