LAB: Digital Image Processing: Date 30 September 2025

Dr. Irfan Ali

Assistant Professor & Coordinator Multimedia & Gaming

Objective:

- Understand how images can be represented in the frequency domain.
- **Visualize the magnitude and phase spectrum of an image.**

Magnitude Spectrum:

- Represents the strength (amplitude) of different frequency components in an image.
- Bright areas = strong frequencies.
- Determines how much of each frequency is present (controls contrast and brightness levels).

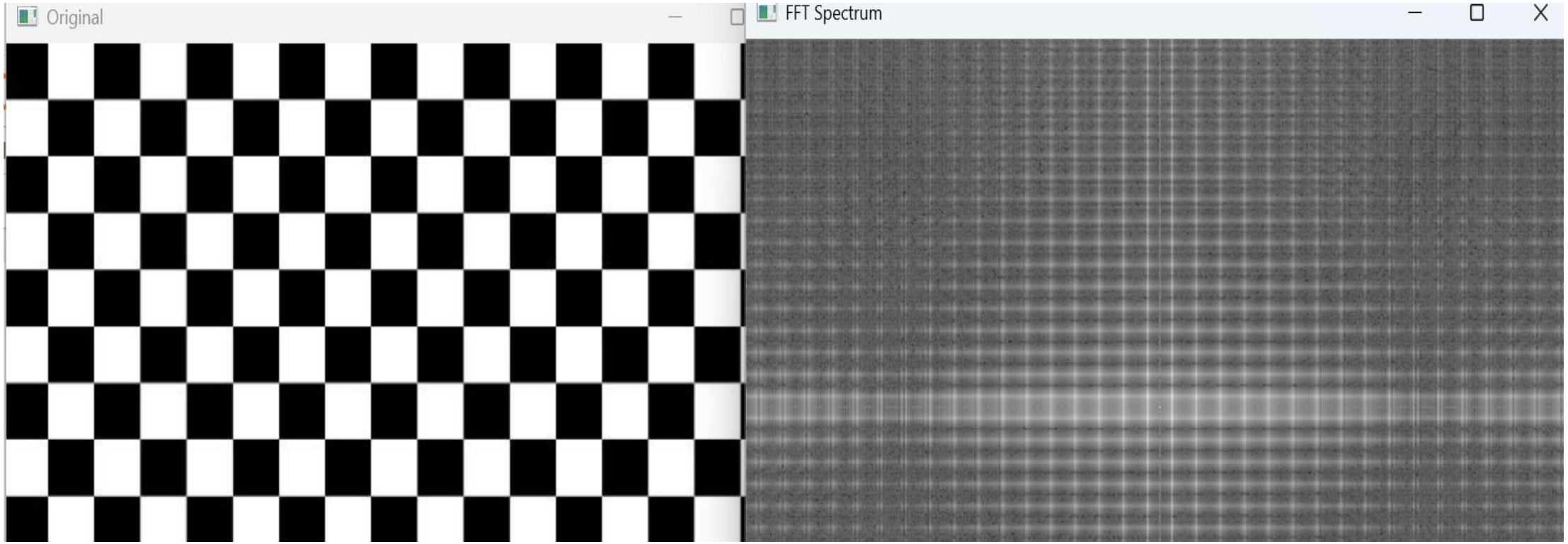
Phase Spectrum:

- Represents the position (angle/shift) of frequency components.
- Looks noisy/random, but it encodes the structural and spatial details of the image.
- Responsible for preserving edges, shapes, and object alignment..

Hands-On Lab: Image Spectrum Visualization and Reconstruction

```
[*]: import cv2, numpy as np

img = cv2.imread("img3.jpg", 0)
f = np.fft.fftshift(np.fft.fft2(img))
spectrum = np.log(np.abs(f)+1)

cv2.imshow("Original", img)
cv2.imshow("FFT Spectrum", np.uint8(spectrum/spectrum.max()*255))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hands-On Lab: Image Spectrum Visualization and Reconstruction



Hands-On Lab: Image Spectrum Visualization and Reconstruction

```
[ ]: f = np.fft.fft2(img)
     f = np.fft.fftshift(f)
```

- **np.fft.fft2(img)** → Performs a 2D Fast Fourier Transform (FFT) on the image.
- This converts the image from the spatial domain (pixel intensities) into the frequency domain (sinusoidal components).
- **np.fft.fftshift(f)** → Moves the low frequencies (DC component) from the corners of the spectrum to the center of the image, which makes it easier to visualize

Hands-On Lab: Image Spectrum Visualization and Reconstruction

```
[ ]: spectrum = np.log(np.abs(f) + 1)
     cv2.imshow("Original", img)
     cv2.imshow("FFT Spectrum", np.uint8(spectrum/spectrum.max()*255))
```

- **np.abs(f)** → Gets the magnitude (removes the imaginary part).
- **np.log(...+1)** → Log scaling is applied because the frequency values can vary a lot, and without log the image would be too bright in the center and too dark elsewhere.
- The +1 avoids log(0) errors.

Hands-On Lab: Image Spectrum Visualization and Reconstruction

spectrum/spectrum.max()*255 → normalizes values between 0–255 (so it can be displayed as an image).

np.uint8(...) → converts to 8-bit image format.

Hands-On Lab: Image Spectrum Visualization and Reconstruction

TASK: Experiment with Different Images by using fft function

•Run the same code on **three different types of images**:

- A smooth image (sky or gradient)
- A textured image (brick wall, grass)
- A text/edge-heavy image (document, chessboard)

•Compare how the **magnitude spectrum** changes.

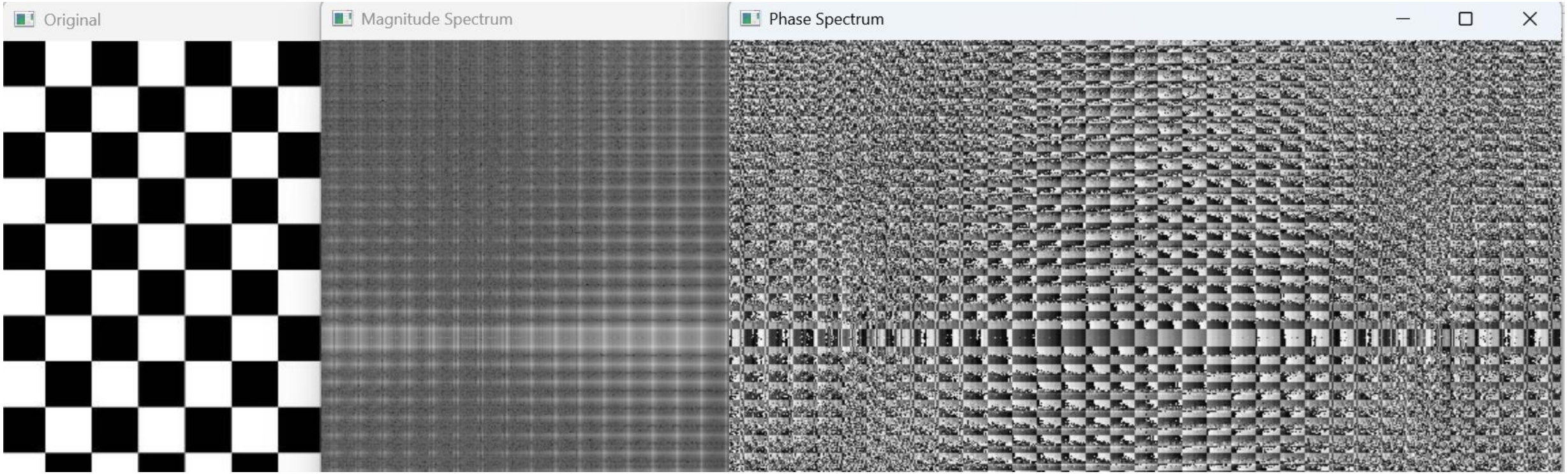
Hands-On Lab: Image Spectrum Visualization and Reconstruction

[*]:
```python
import cv2, numpy as np

# Read image in grayscale
img = cv2.imread("img3.jpg", 0)

# Apply 2D Fourier Transform and shift
f = np.fft.fftshift(np.fft.fft2(img))

# Magnitude Spectrum
magnitude = np.log(np.abs(f) + 1)

# Phase Spectrum
phase = np.angle(f)

# Normalize for display (0-255)
magnitude_display = np.uint8(magnitude / magnitude.max() * 255)
phase_display = np.uint8((phase + np.pi) / (2*np.pi) * 255)  # Map [-π,π] → [0,255]
```

Hands-On Lab: Image Spectrum Visualization and Reconstruction

```
# Show images
cv2.imshow("Original", img)
cv2.imshow("Magnitude Spectrum", magnitude_display)
cv2.imshow("Phase Spectrum", phase_display)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hands-On Lab: Image Spectrum Visualization and Reconstruction



Hands-On Lab: Image Spectrum Visualization and Reconstruction

TASK: Experiment with Different Images by using fft function

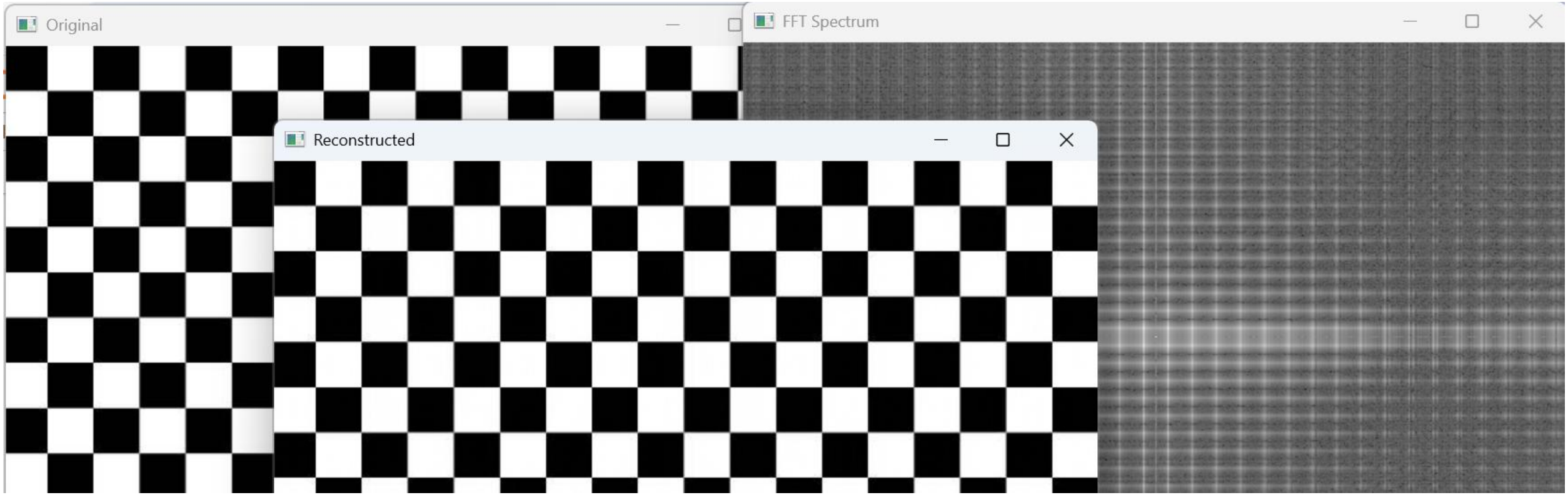
- Run the same code on **three different types of images**:
  - A smooth image (sky or gradient)
  - A textured image (brick wall, grass)
  - A text/edge-heavy image
- Compare how the **magnitude and phase spectrum** changes.

Hands-On Lab: Image Spectrum Visualization and Reconstruction

Inverse Fourier transform to reconstruct image

[8]:

```
import cv2, numpy as np

img = cv2.imread("img3.jpg", 0)
f = np.fft.fftshift(np.fft.fft2(img))
spectrum = np.log(np.abs(f)+1)

# Inverse FFT
reconstructed = np.fft.ifft2(np.fft.ifftshift(f))
reconstructed = np.abs(reconstructed)

cv2.imshow("Original", img)
cv2.imshow("FFT Spectrum", np.uint8(spectrum/spectrum.max()*255))
cv2.imshow("Reconstructed", np.uint8(reconstructed))

cv2.waitKey(0)
cv2.destroyAllWindows()
```


Original
FFT Spectrum
Reconstructed

TASK: Experiment with Different Images to reconstruct image

•Run the same code on **three different types of images**:

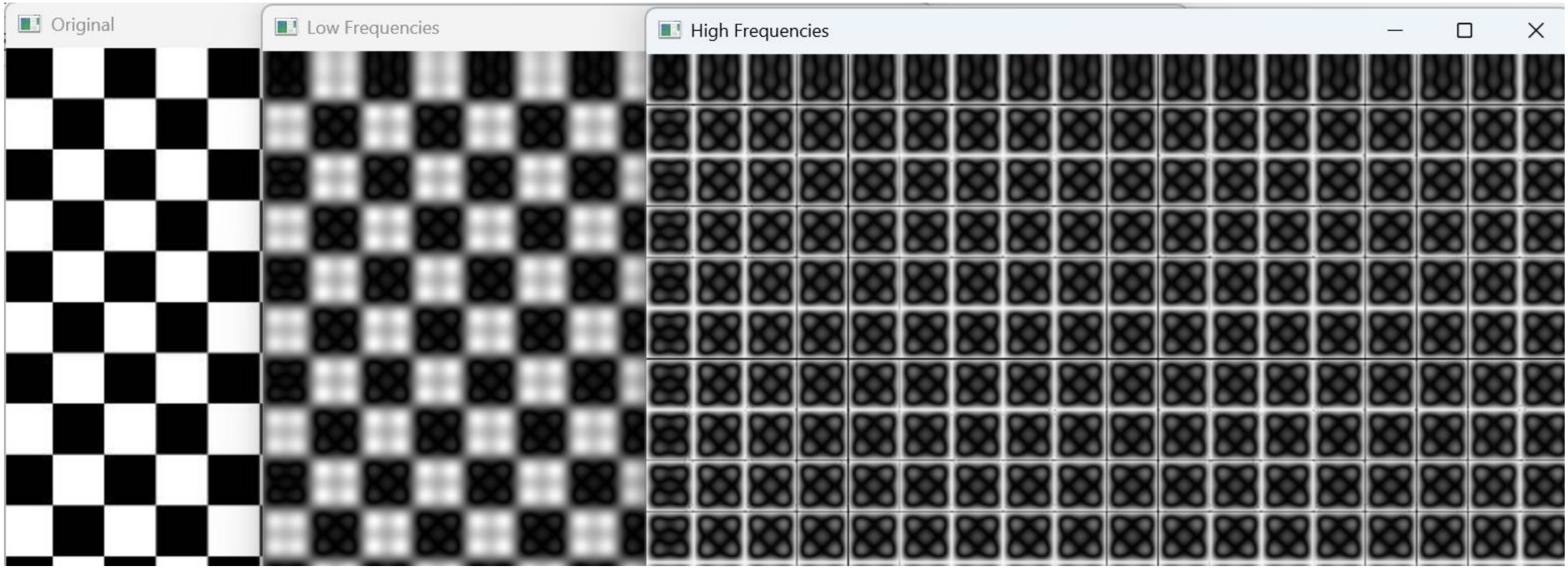
- A smooth image (sky or gradient)
- A textured image (brick wall, grass)
- A text/edge-heavy image (document, chessboard)

•Compare how the **magnitude spectrum** changes.

Hands-On Lab: Image Spectrum Visualization and Reconstruction

Apply high-low frequencies to observe the effects

[*]:
```
import cv2, numpy as np

img = cv2.imread("img3.jpg", 0)
r,c = img.shape
cr,cc = r//2, c//2
f = np.fft.fftshift(np.fft.fft2(img))

mask = np.zeros((r,c), np.uint8)
mask[cr-30:cr+30, cc-30:cc+30] = 1

low  = np.fft.ifft2(np.fft.ifftshift(f*mask))
high = np.fft.ifft2(np.fft.ifftshift(f*(1-mask)))

cv2.imshow("Original", img)
cv2.imshow("Low Frequencies", np.uint8(np.abs(low)/np.abs(low).max()*255))
cv2.imshow("High Frequencies", np.uint8(np.abs(high)/np.abs(high).max()*255))
```


Original
Low Frequencies
High Frequencies

Task:

- Run the code and observe the Original Image, Low Frequencies Image, and High Frequencies Image.
- Change the mask size in the line:
- Try different sizes like 10, 50, 80, 120.
- Observe how the Low Frequency Image becomes more/less blurred.
- Observe how the High Frequency Image changes (edges more/less visible)

Task:

- **Modify the shape of the mask:**
- Instead of a square, try a rectangular region or circular region.
- Discuss how the shape of the mask changes the filtering result.
- **Compare results:**
- Which filter preserves smooth regions better?
- Which filter enhances edges and fine details?
- How does increasing/decreasing the mask size affect the balance between smoothness and sharpness?.