```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import wiener
from skimage import restoration
```

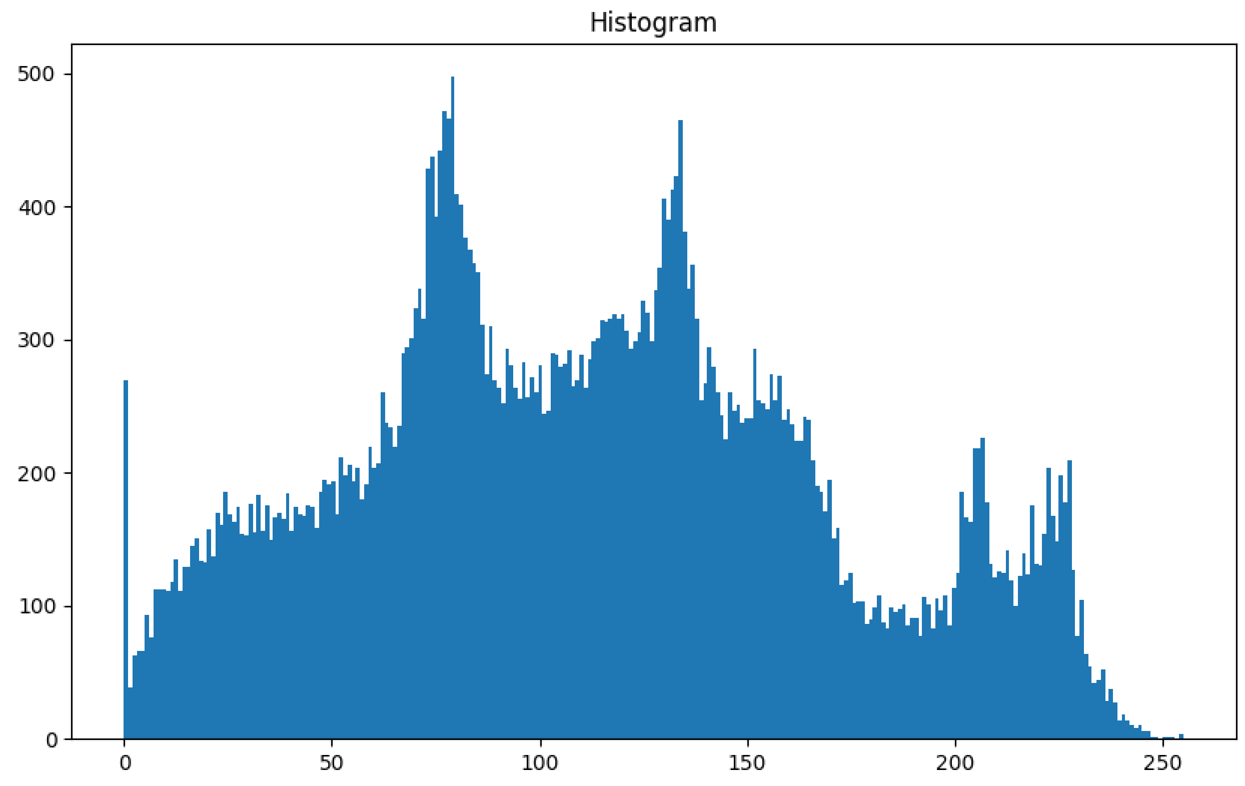
Load Image

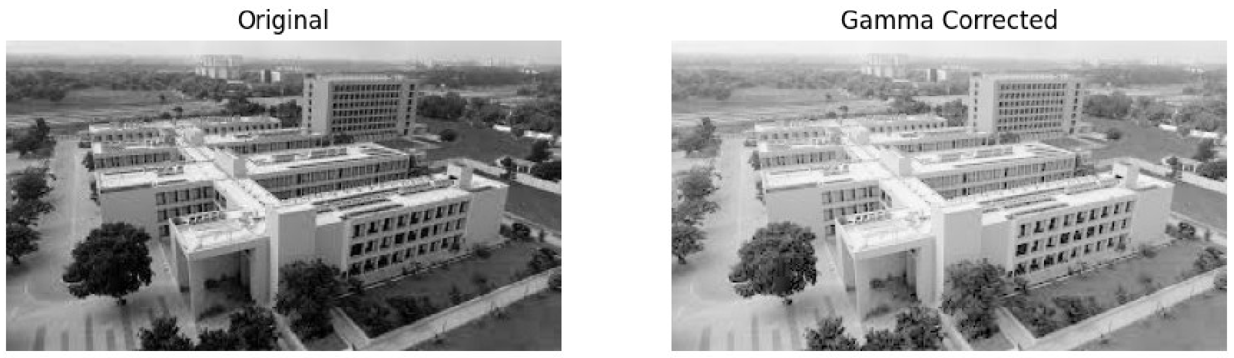
```python
img_gray = cv2.imread("pdeu.png", cv2.IMREAD_GRAYSCALE)
img_color = cv2.imread("pdeu.png")
img_color = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(8, 5))
plt.imshow(img_gray, cmap="gray")
plt.title("Original Image")
plt.axis("off")
plt.show()
```

Q1 – Histogram & Gamma Correction

```python
plt.figure(figsize=(10,6))

plt.hist(img_gray.ravel(), 256)
plt.title("Histogram")
plt.show()
```

```
gamma = 0.5
gamma_corrected = np.uint8(255 * (img_gray / 255) ** gamma)

plt.figure(figsize=(11,6))
plt.subplot(1,2,1); plt.imshow(img_gray, cmap='gray');
plt.title("Original"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(gamma_corrected, cmap='gray');
plt.title("Gamma Corrected"); plt.axis('off')
plt.show()
```

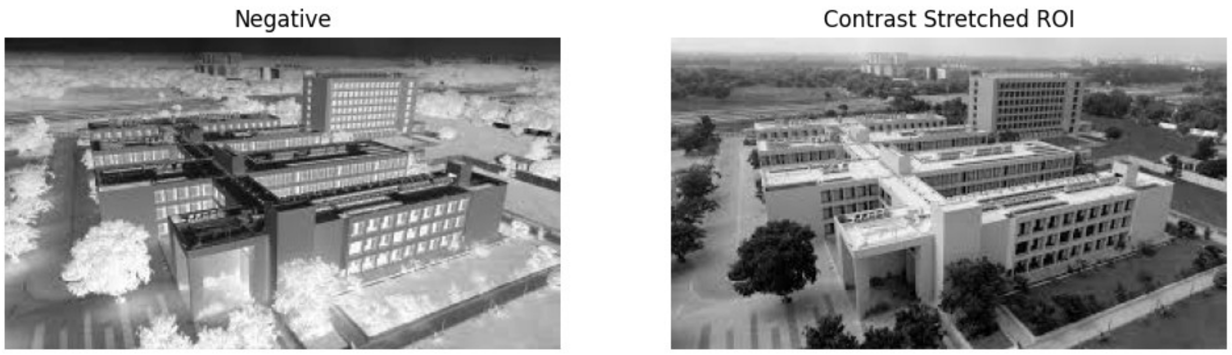




Q3 – Negative & Contrast Stretching

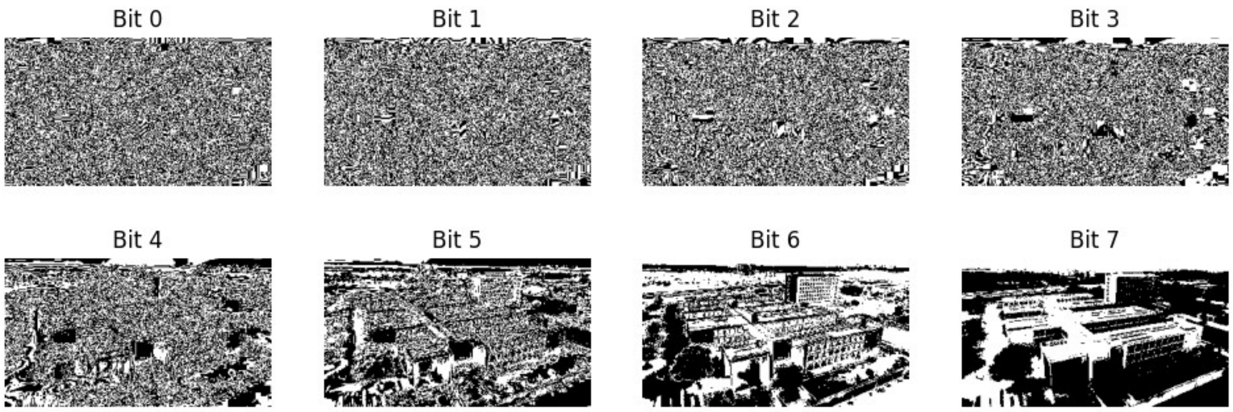
```
negative = 255 - img_gray
```

```
roi = img_gray[100:300, 100:300]
minv, maxv = roi.min(), roi.max()
contrast = ((roi - minv) / (maxv - minv)) * 255

cs_img = img_gray.copy()
cs_img[100:300, 100:300] = contrast

plt.figure(figsize=(12,8))
plt.subplot(1,2,1); plt.imshow(negative, cmap='gray');
plt.title("Negative"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(cs_img, cmap='gray');
plt.title("Contrast Stretched ROI"); plt.axis('off')
plt.show()
```



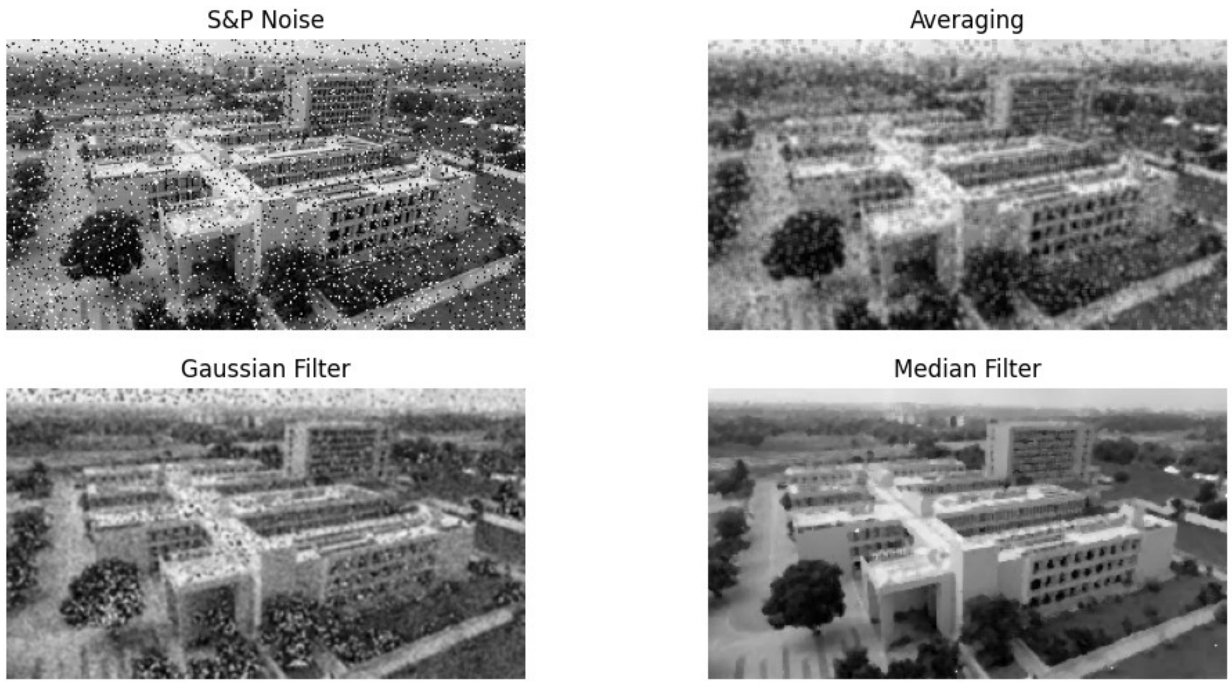
Q4 – Bit Plane Slicing

```
plt.figure(figsize=(12,4))
for i in range(8):
    bit_plane = (img_gray >> i) & 1
    plt.subplot(2,4,i+1)
    plt.imshow(bit_plane, cmap='gray')
    plt.title(f"Bit {i}")
    plt.axis('off')
plt.show()
```



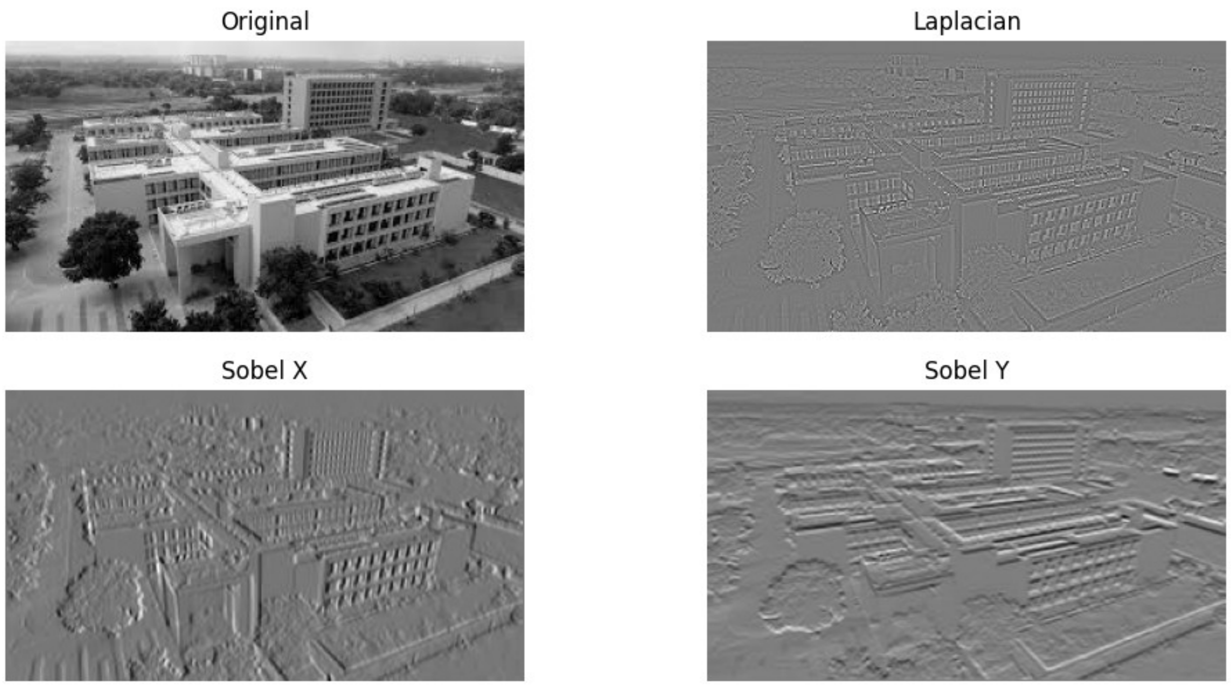
Q5 – Noise Addition & Filtering

```python
# Salt & Pepper
sp = img_gray.copy()
prob = 0.05
rand = np.random.rand(*img_gray.shape)
sp[rand < prob] = 0
sp[rand > 1 - prob] = 255

# Gaussian noise
gauss = img_gray + np.random.normal(0, 25, img_gray.shape)

avg = cv2.blur(sp, (3,3))
gauss_f = cv2.GaussianBlur(gauss.astype(np.uint8), (3,3), 0)
median = cv2.medianBlur(sp, 3)

plt.figure(figsize=(12,6))
imgs = [sp, avg, gauss_f, median]
titles = ["S&P Noise", "Averaging", "Gaussian Filter", "Median
Filter"]

for i in range(4):
    plt.subplot(2,2,i+1)
    plt.imshow(imgs[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```



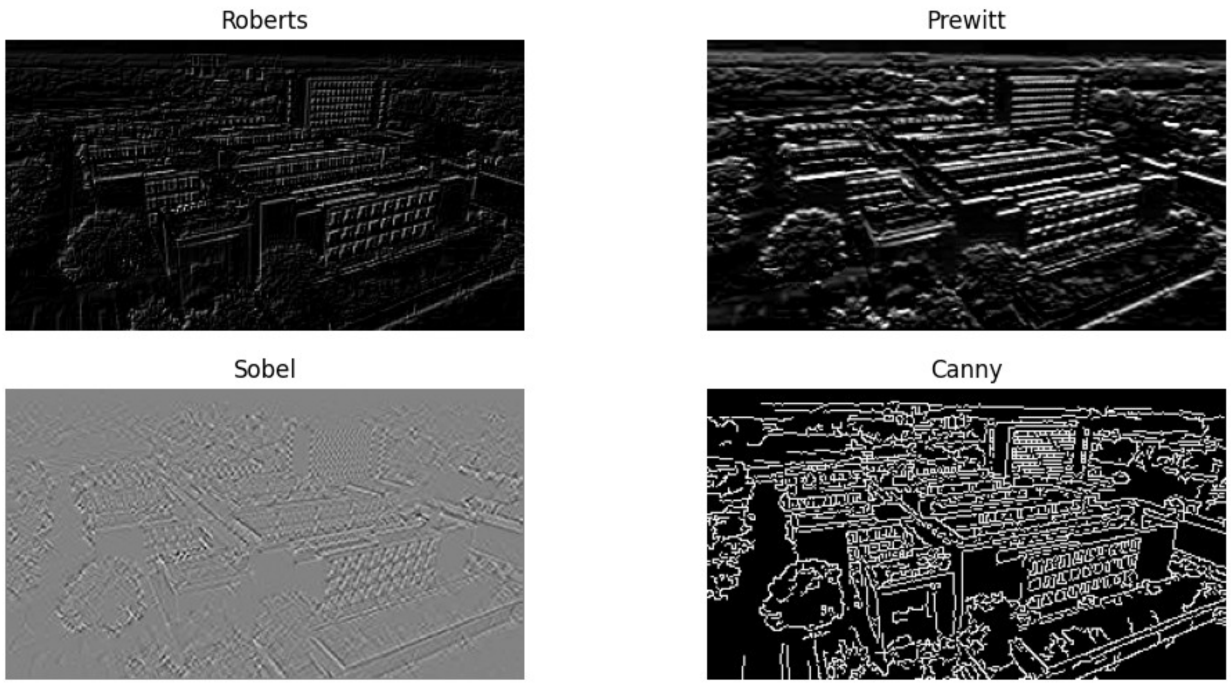
Q6 – Edge Enhancement

```
lap = cv2.Laplacian(img_gray, cv2.CV_64F)
sobelx = cv2.Sobel(img_gray, cv2.CV_64F, 1, 0)
sobely = cv2.Sobel(img_gray, cv2.CV_64F, 0, 1)

plt.figure(figsize=(12,6))
imgs = [img_gray, lap, sobelx, sobely]
titles = ["Original", "Laplacian", "Sobel X", "Sobel Y"]

for i in range(4):
    plt.subplot(2,2,i+1)
    plt.imshow(imgs[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```



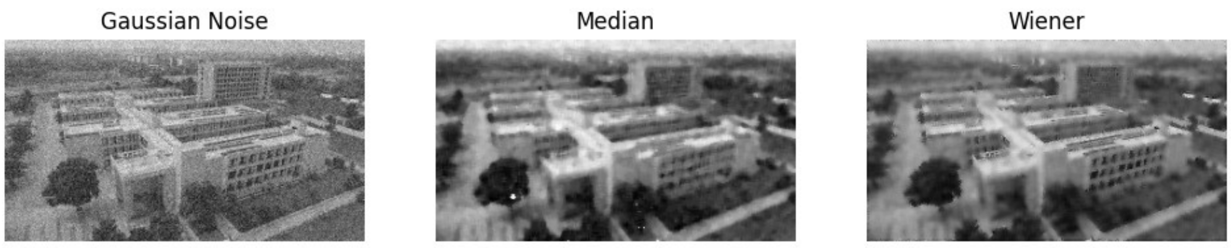
Q7 – Edge Detection Operators

```
roberts = cv2.filter2D(img_gray, -1, np.array([[1,0],[0,-1]]))
prewitt = cv2.filter2D(img_gray, -1, np.array([[1,1,1],[0,0,0],[-1,-
1,-1]]))
sobel = cv2.Sobel(img_gray, cv2.CV_64F, 1, 1)
canny = cv2.Canny(img_gray, 100, 200)

plt.figure(figsize=(12,6))
imgs = [roberts, prewitt, sobel, canny]
titles = ["Roberts", "Prewitt", "Sobel", "Canny"]

for i in range(4):
    plt.subplot(2,2,i+1)
    plt.imshow(imgs[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```



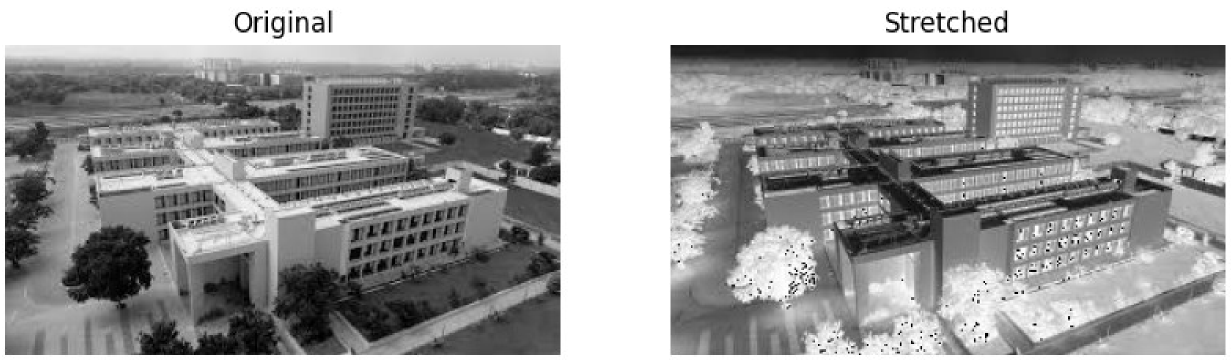
Q8 – Noise & Denoising

```
gauss = img_gray + np.random.normal(0, 25, img_gray.shape)
median_d = cv2.medianBlur(gauss.astype(np.uint8), 5)
wiener_d = wiener(gauss, (5,5))

plt.figure(figsize=(12,8))
plt.subplot(1,3,1); plt.imshow(gauss, cmap='gray');
plt.title("Gaussian Noise"); plt.axis('off')
plt.subplot(1,3,2); plt.imshow(median_d, cmap='gray');
plt.title("Median"); plt.axis('off')
plt.subplot(1,3,3); plt.imshow(wiener_d, cmap='gray');
plt.title("Wiener"); plt.axis('off')
plt.show()
```



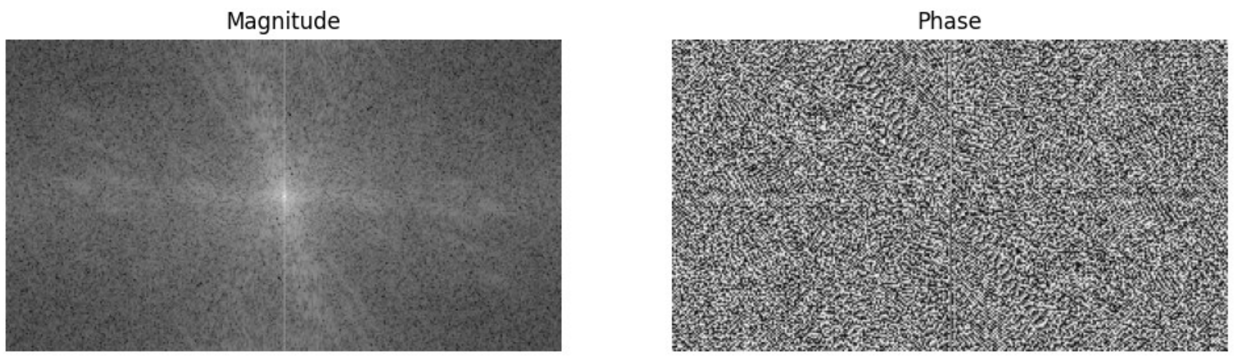
Q9 – Contrast Stretching

```
minv, maxv = img_gray.min(), img_gray.max()
stretch = (img_gray - minv) * 255 / (maxv - minv)

plt.figure(figsize=(10,8))
plt.subplot(1,2,1); plt.imshow(img_gray, cmap='gray');
plt.title("Original"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(stretch, cmap='gray');
plt.title("Stretched"); plt.axis('off')
plt.show()
```



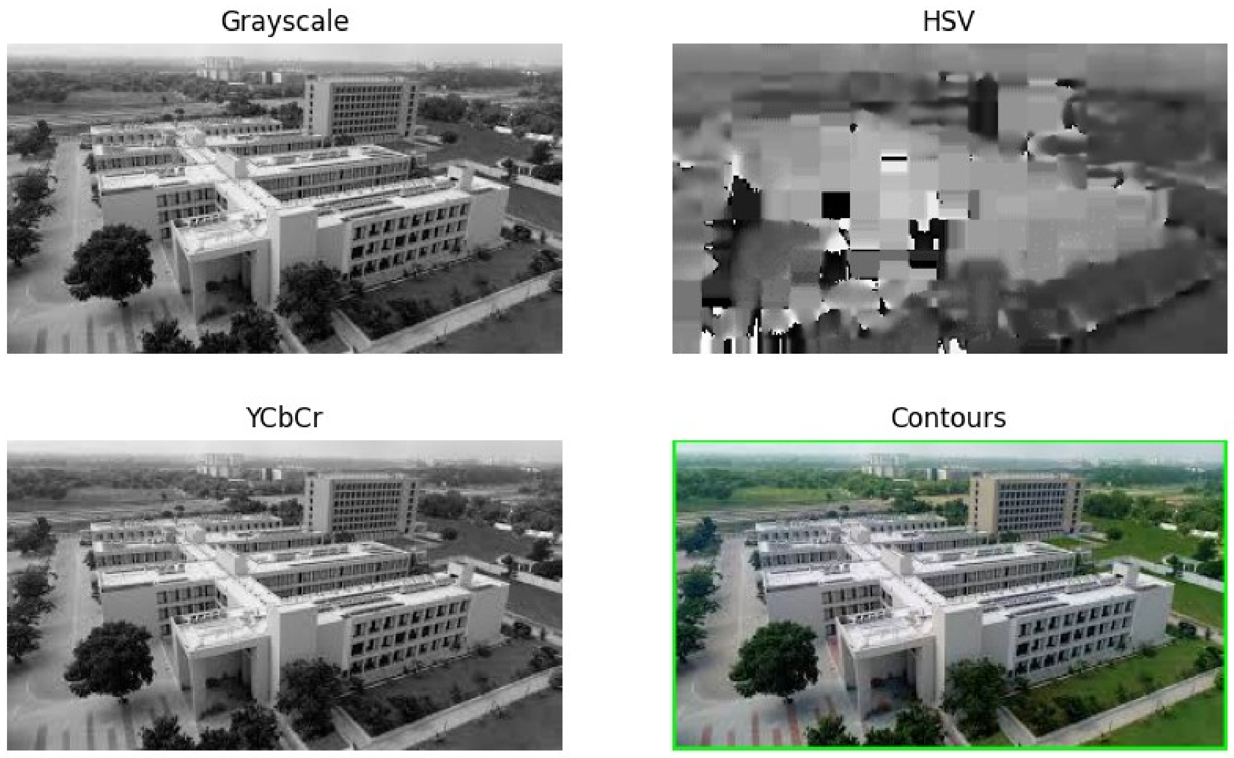
Q10 – Fourier Transform

```
f = np.fft.fftshift(np.fft.fft2(img_gray))
mag = np.log(1 + np.abs(f))
phase = np.angle(f)

plt.figure(figsize=(12,6))
plt.subplot(1,2,1); plt.imshow(mag, cmap='gray');
plt.title("Magnitude"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(phase, cmap='gray');
plt.title("Phase"); plt.axis('off')
plt.show()
```



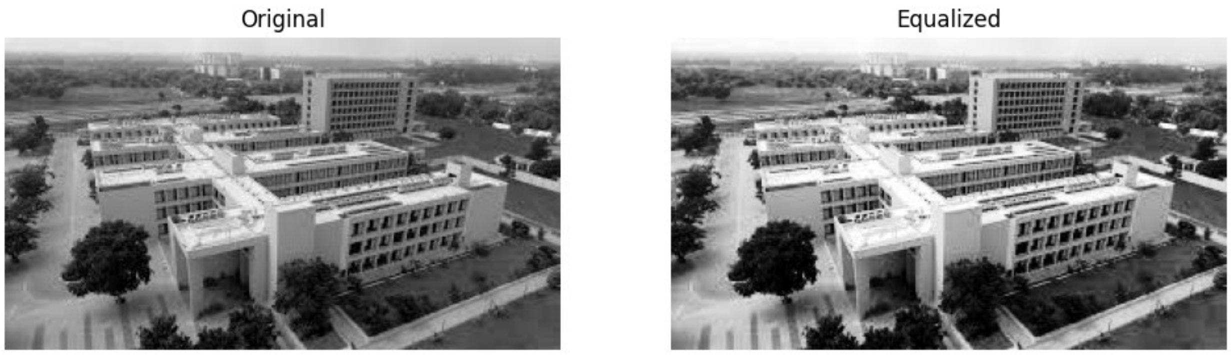
Q11 – Color Spaces & Contours

```
gray = cv2.cvtColor(img_color, cv2.COLOR_RGB2GRAY)
hsv = cv2.cvtColor(img_color, cv2.COLOR_RGB2HSV)
ycbcr = cv2.cvtColor(img_color, cv2.COLOR_RGB2YCrCb)

contours,_ = cv2.findContours(gray, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cont_img = img_color.copy()
cv2.drawContours(cont_img, contours, -1, (0,255,0), 2)

plt.figure(figsize=(10,6))
plt.subplot(2,2,1); plt.imshow(gray, cmap='gray');
plt.title("Grayscale"); plt.axis('off')
plt.subplot(2,2,2); plt.imshow(hsv[:,:,0], cmap='gray');
plt.title("HSV"); plt.axis('off')
plt.subplot(2,2,3); plt.imshow(ycbcr[:,:,0], cmap='gray');
plt.title("YCbCr"); plt.axis('off')
plt.subplot(2,2,4); plt.imshow(cont_img); plt.title("Contours");
plt.axis('off')
plt.show()
```



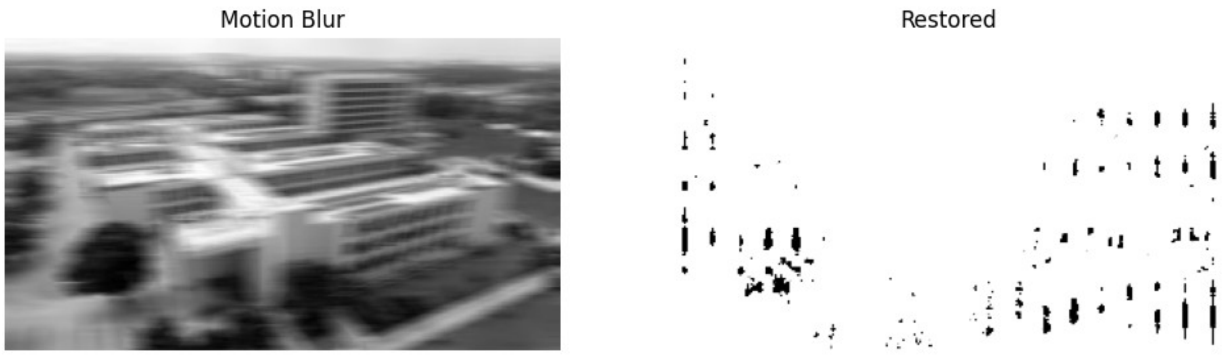
Q12 – Histogram Equalization

```
eq = cv2.equalizeHist(img_gray)

plt.figure(figsize=(12,8))
plt.subplot(1,2,1); plt.imshow(img_gray, cmap='gray');
plt.title("Original"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(eq, cmap='gray');
plt.title("Equalized"); plt.axis('off')
plt.show()
```



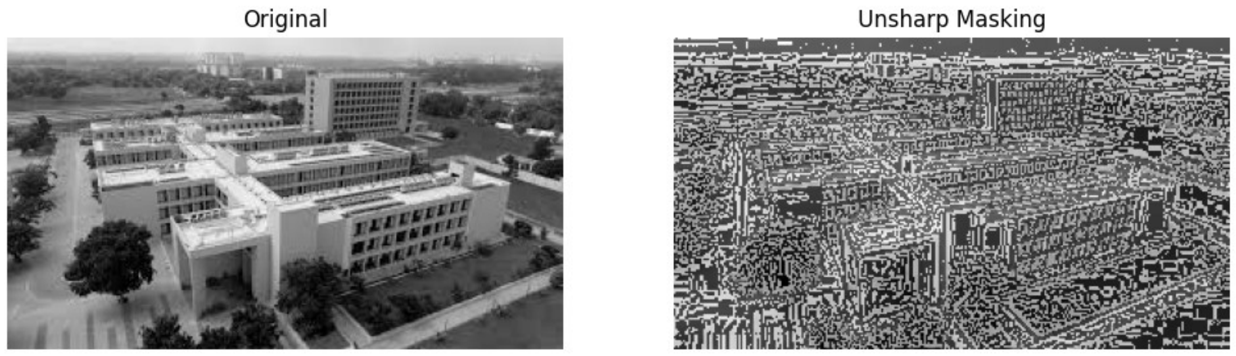
Q13 – Motion Blur & Restoration

```
kernel = np.zeros((15,15))
kernel[7,:] = 1/15

blur = cv2.filter2D(img_gray, -1, kernel)
restored = restoration.unsupervised_wiener(blur, kernel)[0]

plt.figure(figsize=(12,8))
plt.subplot(1,2,1); plt.imshow(blur, cmap='gray'); plt.title("Motion
Blur"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(restored, cmap='gray');
plt.title("Restored"); plt.axis('off')
plt.show()
```



Q14 – Image Sharpening

```
blur = cv2.GaussianBlur(img_gray, (5,5), 0)
unsharp = img_gray + 1.5 * (img_gray - blur)

plt.figure(figsize=(12,8))
plt.subplot(1,2,1); plt.imshow(img_gray, cmap='gray');
plt.title("Original"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(unsharp, cmap='gray');
plt.title("Unsharp Masking"); plt.axis('off')
plt.show()
```



Q15 – Homomorphic Filtering

```
img = img_gray.astype(np.float32) / 255
log_img = np.log1p(img)

fft = np.fft.fftshift(np.fft.fft2(log_img))
rows, cols = img.shape
u, v = np.meshgrid(np.arange(cols), np.arange(rows))
D = np.sqrt((u-cols/2)**2 + (v-rows/2)**2)

H = 1 - np.exp(-(D**2)/(2*(30**2)))
filtered = fft * H

result = np.exp(np.real(np.fft.ifft2(np.fft.ifftshift(filtered))))

plt.figure(figsize=(12,6))
plt.subplot(1,2,1); plt.imshow(img, cmap='gray');
plt.title("Original"); plt.axis('off')
plt.subplot(1,2,2); plt.imshow(result, cmap='gray');
plt.title("Enhanced"); plt.axis('off')
plt.show()
```

Original

Enhanced