SUPERIOR UNIVERSITY

Name : **MUHAMMAD AHMAD**

Roll No : **SU92-BSAIM-F23-135**

Section : **4-C**

Task : **05**

Date : **20th March, 2025**

Subject : **Programming AI Lab**

Submitted To : **Prof Rasikh Ali**

Task : 05

Import Libraries

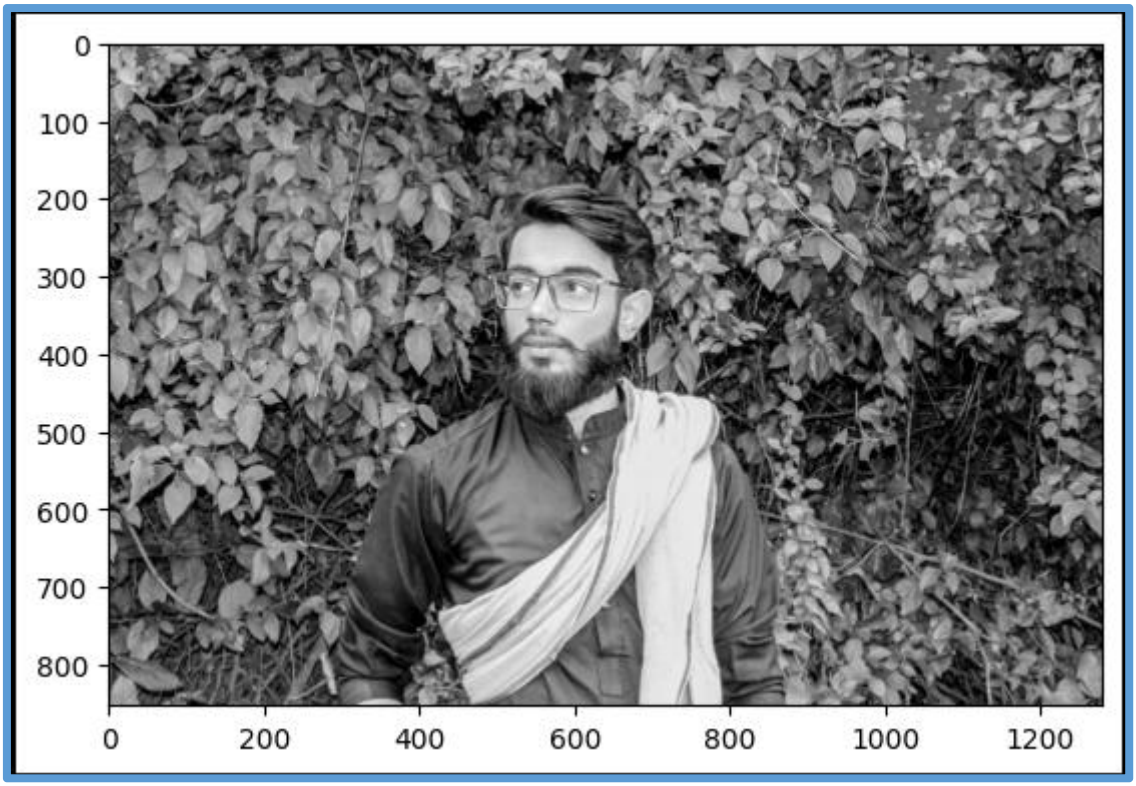
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

Loding Images .....

```
img = cv2.imread(r'C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at 08.58.58_65992efe.jpg')
cv2.imshow("Ahmad", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
img.shape
```

Read Image In Grayscale

```
img_gray = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at
08.58.58_65992efe.jpg")
img_gray = cv2.cvtColor(img_gray, cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray, cmap='gray')
img_gray.shape
```

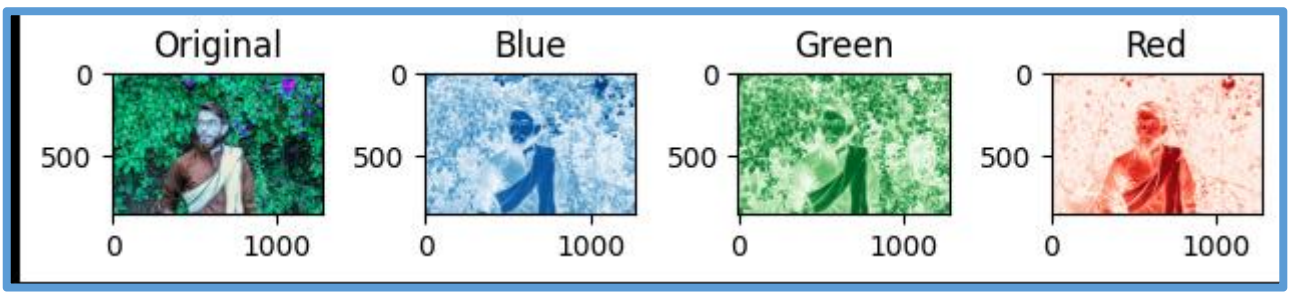


Write & Save Image

```
cv2.imwrite("Ahmad", img_gray)
```

Color Spaces

```
img = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at 08.58.58_65992efe.jpg")
red, green, blue = cv2.split(img)
fig, axes = plt.subplots(nrows=1, ncols=4, facecolor = 'white')

axes[0].imshow(img, )
axes[1].imshow(blue, cmap='Blues')
axes[2].imshow(green, cmap='Greens')
axes[3].imshow(red, cmap='Reds')

axes[0].set_title('Original')
axes[1].set_title('Blue')
axes[2].set_title('Green')
axes[3].set_title('Red')

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



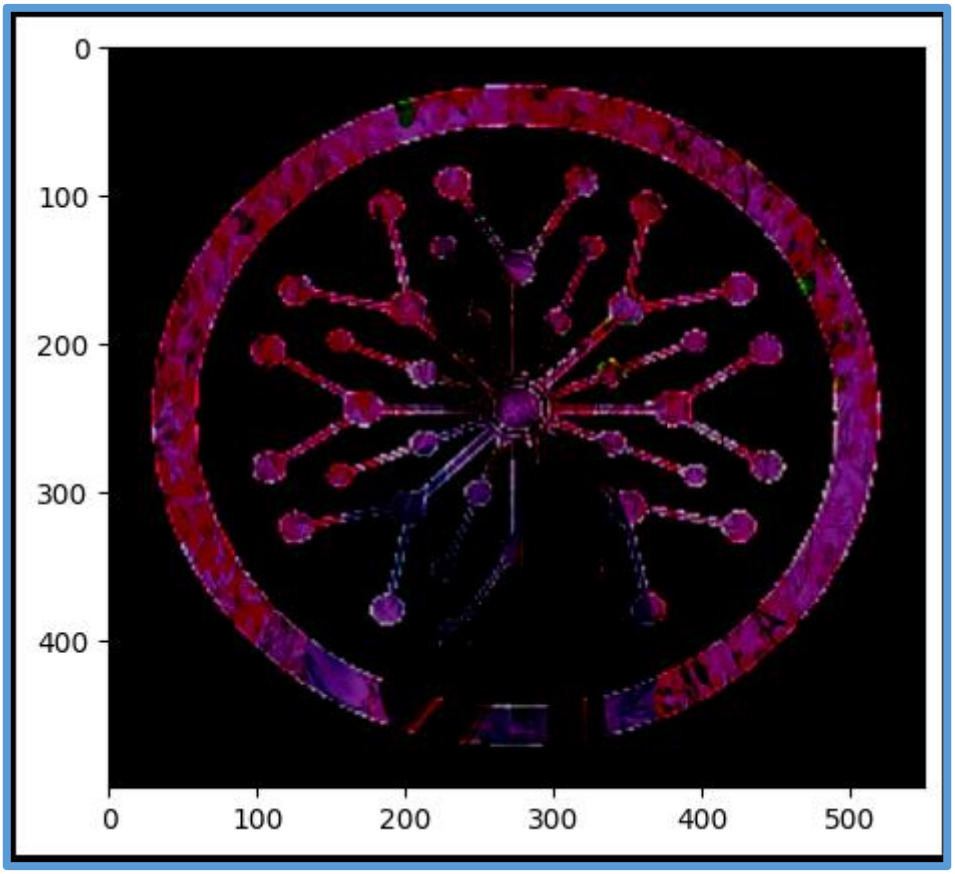
Arithmetic Operations on Images
Addition of Images

To add images both images should have equal shape.

```
img = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at 08.58.58_65992efe.jpg")
img2 = cv2.imread(r"C:\Users\Ahmad\Downloads\images__1_-removebg-preview.png")

if img is None or img2 is None:
    raise ValueError("One or both images could not be loaded. Check file paths.")

img = cv2.resize(img, (550, 500))
img2 = cv2.resize(img2, (550, 500))

if len(img.shape) == 2:
    img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
if len(img2.shape) == 2:
    img2 = cv2.cvtColor(img2, cv2.COLOR_GRAY2BGR)

weighted_sum = cv2.addWeighted(img, 0.5, img2, 0.6, 0)

weighted_sum_rgb = cv2.cvtColor(weighted_sum, cv2.COLOR_BGR2RGB)

plt.imshow(weighted_sum_rgb)
plt.axis("off")
plt.show()
```

Subtraction OF Image

```
sub = cv2.subtract(img2, img)
plt.imshow(sub)
```



Bitwise Operations

```
img = cv2.imread(r'Black _white1.png')
img2 = cv2.imread(r'Black_white2.png')
img2 = cv2.resize(img2, (img.shape[1], img.shape[0]))

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

dest_AND = cv2.bitwise_and(img2, img, mask=None)
dest_OR = cv2.bitwise_or(img2, img, mask = None)
dest_XOR = cv2.bitwise_xor(img2, img, mask = None)
dest_NOT = cv2.bitwise_not(img2, mask = None)

fig, axes = plt.subplots(nrows=2, ncols=3, facecolor = 'white', figsize=(7, 6))
axes = axes.flatten()
fig.tight_layout()

axes[0].imshow(img)
axes[1].imshow(img2)
axes[2].imshow(dest_AND)
axes[3].imshow(dest_OR)
axes[4].imshow(dest_XOR)
axes[5].imshow(dest_NOT)

axes[0].set_title('Binary img1')
axes[1].set_title('Binary img2')
axes[2].set_title('AND Operation')
```

```
axes[3].set_title('OR Operation')
axes[4].set_title('XOR Operation')
axes[5].set_title('NOT Operation')

plt.show()
```

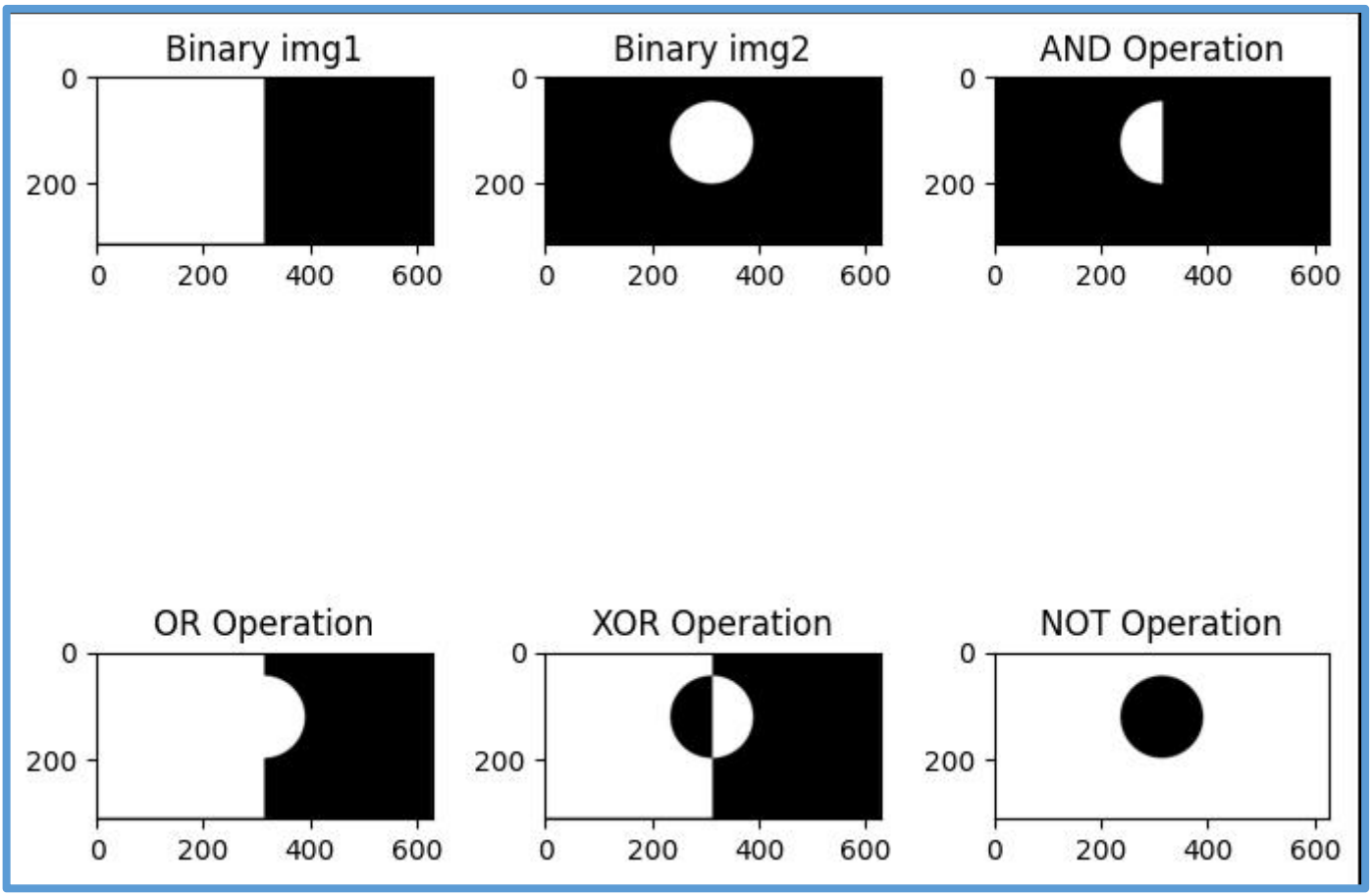


Image Resizing

```
image = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at
08.58.58_65992efe.jpg")
half = cv2.resize(image, (0, 0), fx = 0.1, fy = 0.1)
bigger = cv2.resize(image, (1050, 1610))

stretch_near = cv2.resize(image, (780, 540),
               interpolation = cv2.INTER_LINEAR)


Titles =["Original", "Half", "Bigger", "Interpolation Nearest"]
images =[image, half, bigger, stretch_near]
count = 4

for i in range(count):
    plt.subplot(2, 2, i + 1)
    plt.title(Titles[i])
    plt.tight_layout()
    plt.imshow(images[i])

plt.show()
```

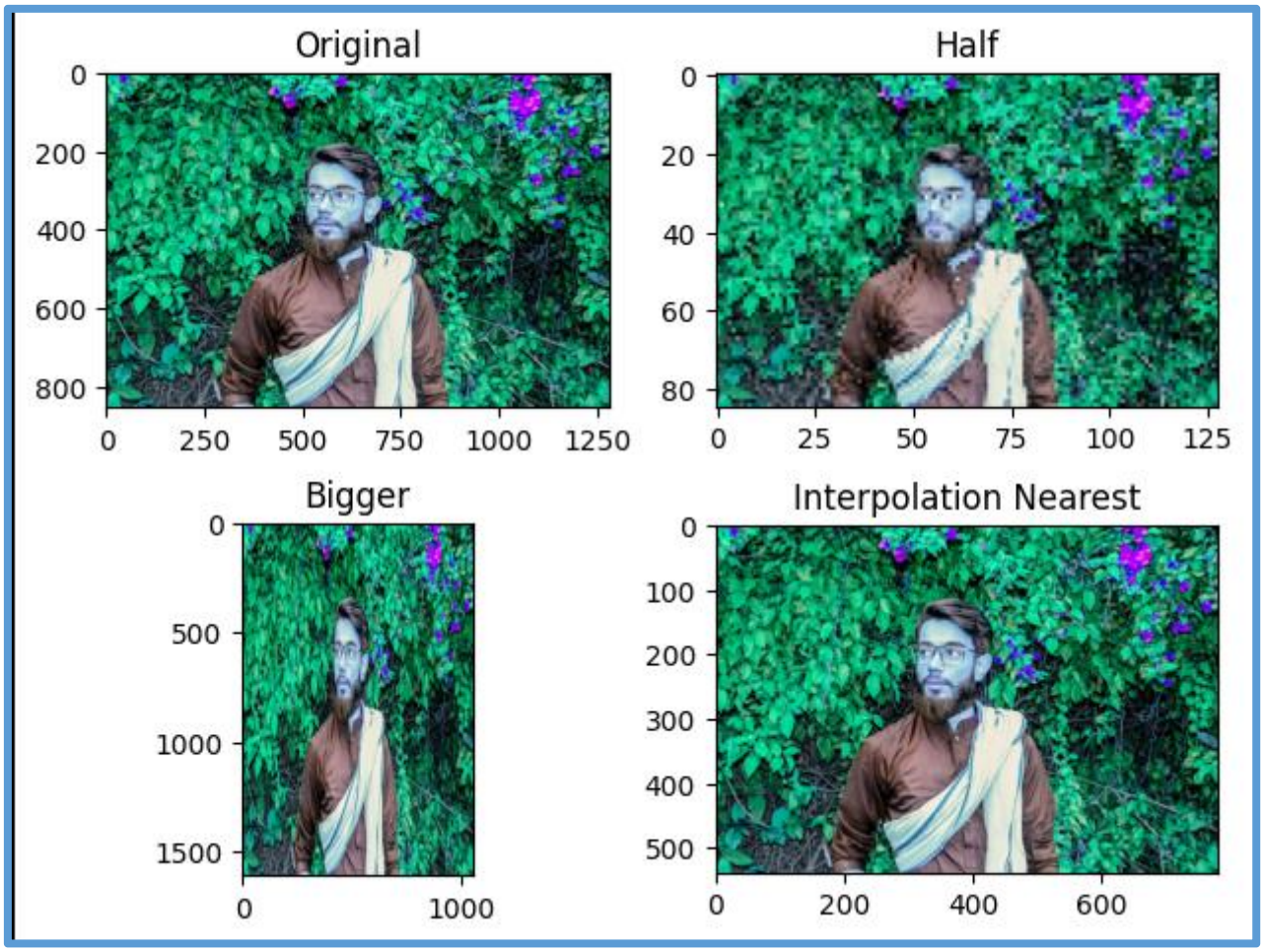
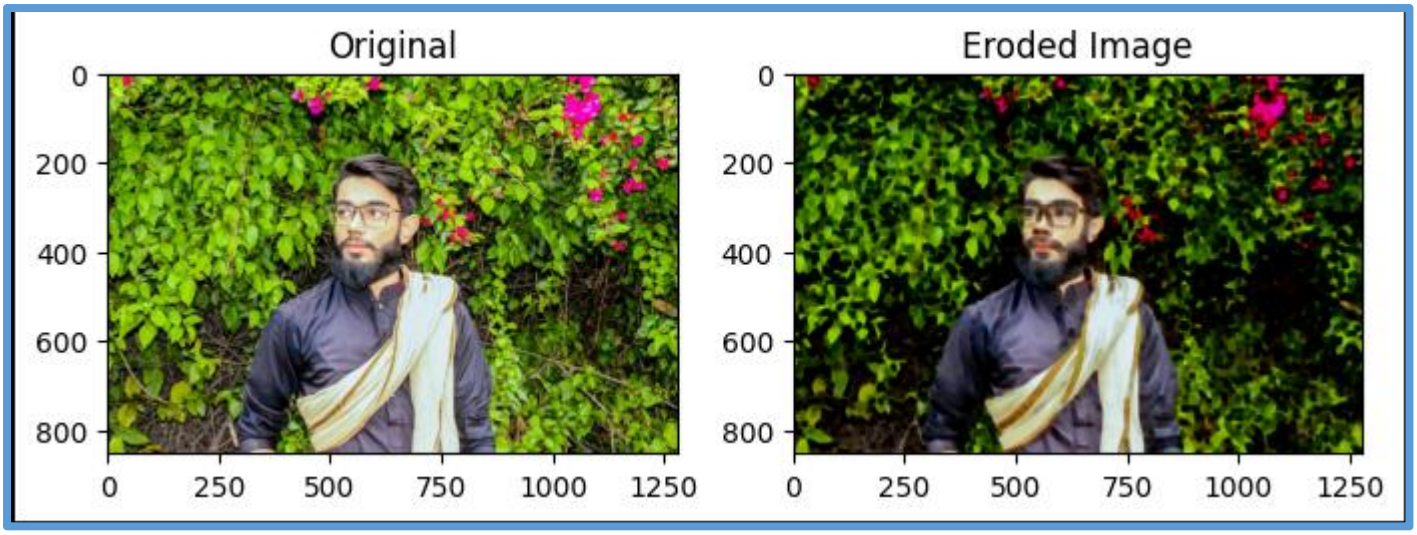


Image Erosion

```
img = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at 08.58.58_65992efe.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

kernel = np.ones((5, 5), np.uint8)
eroded_img = cv2.erode(img, kernel, iterations=2)

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(7,7))
fig.tight_layout()

axes[0].imshow(img)
axes[1].imshow(eroded_img)

axes[0].set_title('Original')
axes[1].set_title('Eroded Image')
plt.show()
```



Blurring An Image

```
img = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at 08.58.58_65992efe.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

gaussian = cv2.GaussianBlur(img, (15, 15), 0)
median = cv2.medianBlur(img, 11)
bilateral = cv2.bilateralFilter(img, 15, 150, 150)


fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 10))
fig.tight_layout()


axes[0].imshow(img)
axes[1].imshow(gaussian)
axes[2].imshow(median)
axes[3].imshow(bilateral)

axes[0].set_title('Original')
axes[1].set_title('Gaussian Blur')
axes[2].set_title('Median Blur')
axes[3].set_title('Bilateral Blur')

plt.show()
```

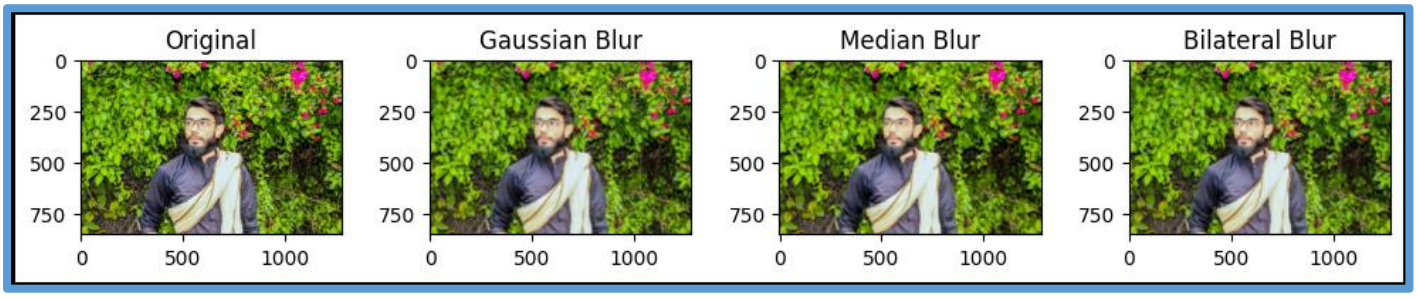
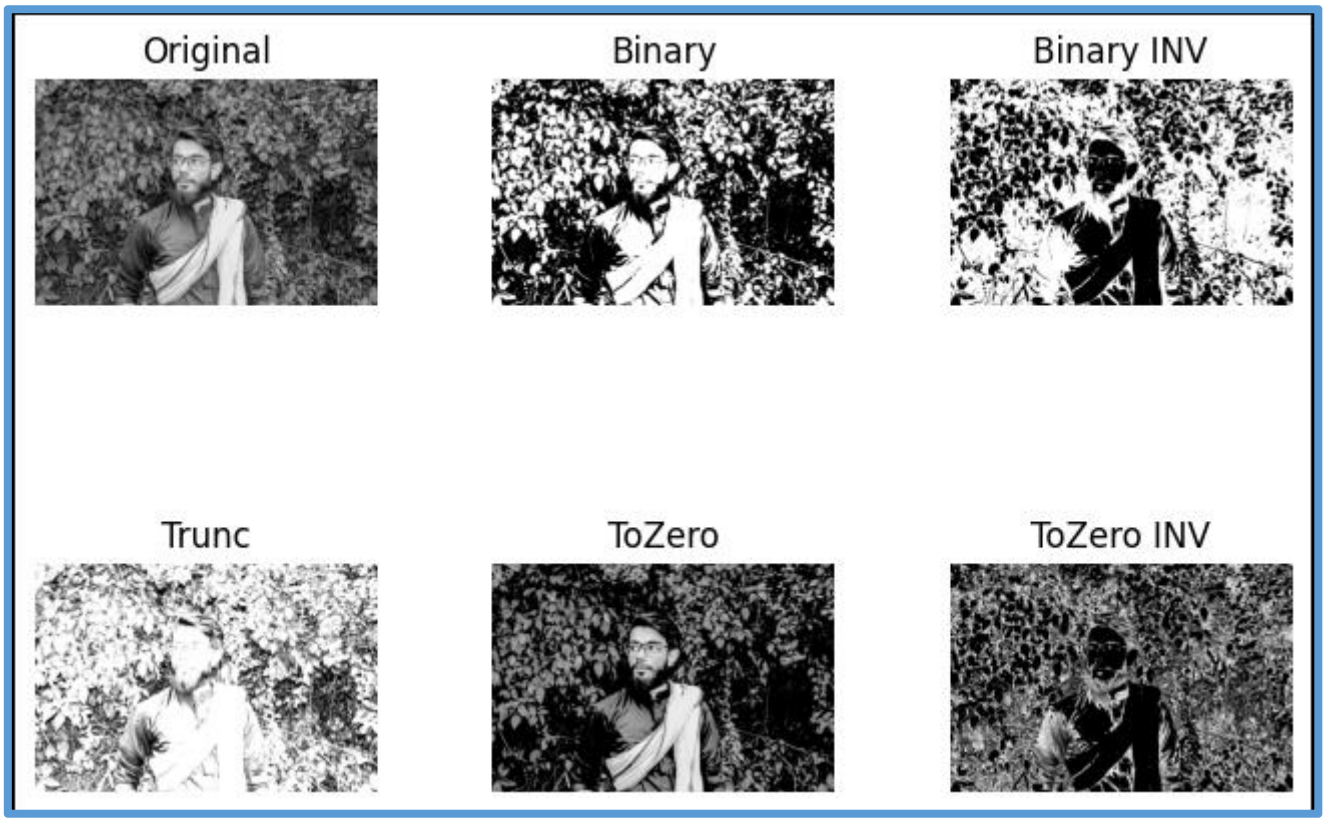


Image Thresholding

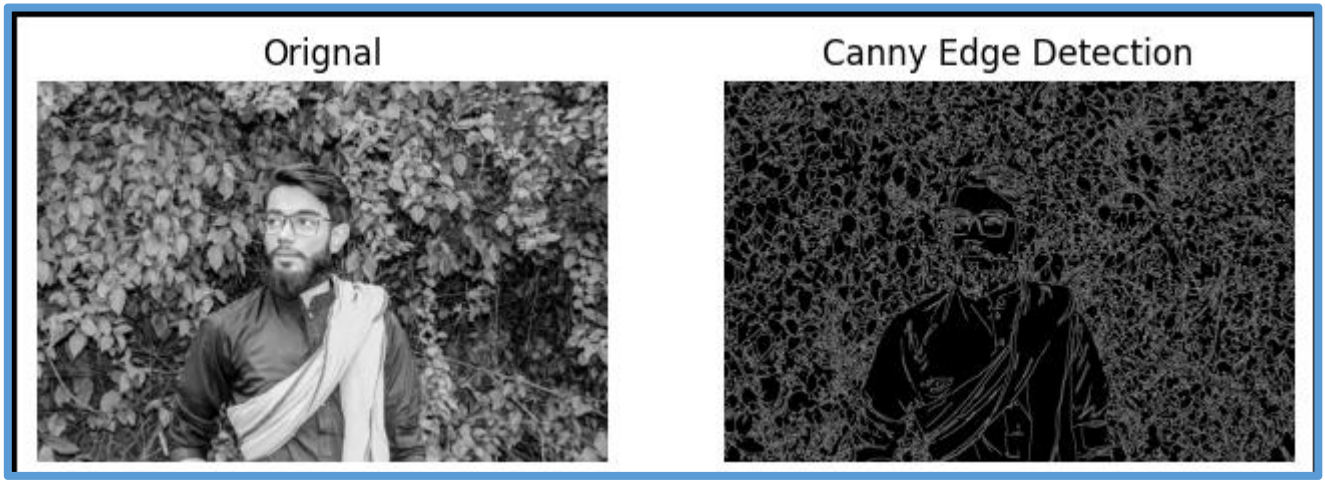
```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(7, 5))
axes = axes.flatten()

ret, Thresh1 = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY)
ret, Thresh1_INV = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)
ret, Thresh2 = cv2.threshold(img_gray, 100, 255, cv2.THRESH_TRUNC)
ret, Thresh3 = cv2.threshold(img_gray, 100, 255, cv2.THRESH_TOZERO)
ret, Thresh3_INV = cv2.threshold(img_gray, 100, 255, cv2.THRESH_TOZERO_INV)

fig.tight_layout()

axes[0].imshow(img_gray, cmap='gray')
axes[1].imshow(Thresh1, cmap='gray')
axes[2].imshow(Thresh1_INV, cmap='gray')
axes[3].imshow(Thresh2, cmap='gray')
axes[4].imshow(Thresh3, cmap='gray')
axes[5].imshow(Thresh3_INV, cmap='gray')

titles = ['Original', 'Binary', 'Binary INV', 'Trunc', 'ToZero', 'ToZero INV']
for ax, title in zip(axes, titles):
    ax.set_title(title)
    ax.axis('off')

plt.show()
```



Edge Detection

```
img = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at 08.58.58_65992efe.jpg", cv2.IMREAD_GRAYSCALE)
edges = cv2.Canny(img, 100, 200)
```

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(7, 6))
fig.tight_layout()

axes[0].imshow(img, cmap='gray')
axes[1].imshow(edges, cmap='gray')

axes[0].set_title('Orignal')
axes[1].set_title('Canny Edge Detection')

titles = ['Orignal', 'Canny Edge Detection']

for ax, title in zip(axes, titles):
    ax.set_title(title)
    ax.axis('Off')


plt.show()
```



Contour Detection

```
original_img = cv2.imread(r"C:\Users\Ahmad\Downloads\WhatsApp Image 2025-03-09 at 08.58.58_65992efe.jpg", cv2.IMREAD_COLOR)
img = original_img.copy()
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

fig, axes = plt.subplots(nrows = 1, ncols=2, figsize=(7, 6))

ret, thresh = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(thresh ,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(img, contours, -1, (0, 255, 0), 2)

fig.tight_layout()

axes[0].imshow(original_img)
axes[1].imshow(img)

titles = ['Orignal', 'Contour']

for ax, title in zip(axes, titles):
```

```
    ax.set_title(title)
    ax.axis('off')

plt.show()
```

HoughLine

```
image_path =
(r"C:\Users\Ahmad\Downloads\Mumbai_Bengaluru_Expressway_1666064412127_1666064412283_16660
64412283.jpeg")
original_img = cv2.imread(image_path)

hsv = cv2.cvtColor(original_img, cv2.COLOR_BGR2HSV)

lower_white = np.array([0, 0, 180])
upper_white = np.array([255, 50, 255])
mask = cv2.inRange(hsv, lower_white, upper_white)

white_lane = cv2.bitwise_and(original_img, original_img, mask=mask)

gray = cv2.cvtColor(white_lane, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150, apertureSize=3)

lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 50, minLineLength=50, maxLineGap=10)

img = original_img.copy()
if lines is not None:
    for line in lines:
        x1, y1, x2, y2 = line[0]
        cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

original_img_rgb = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

fig, axes = plt.subplots(1, 2, figsize=(10, 5))
axes[0].imshow(original_img_rgb)
axes[0].set_title("Original Image")
axes[0].axis("off")

axes[1].imshow(img_rgb)
axes[1].set_title("Detected White Lane Lines")
```

```
axes[1].axis("off")

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