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

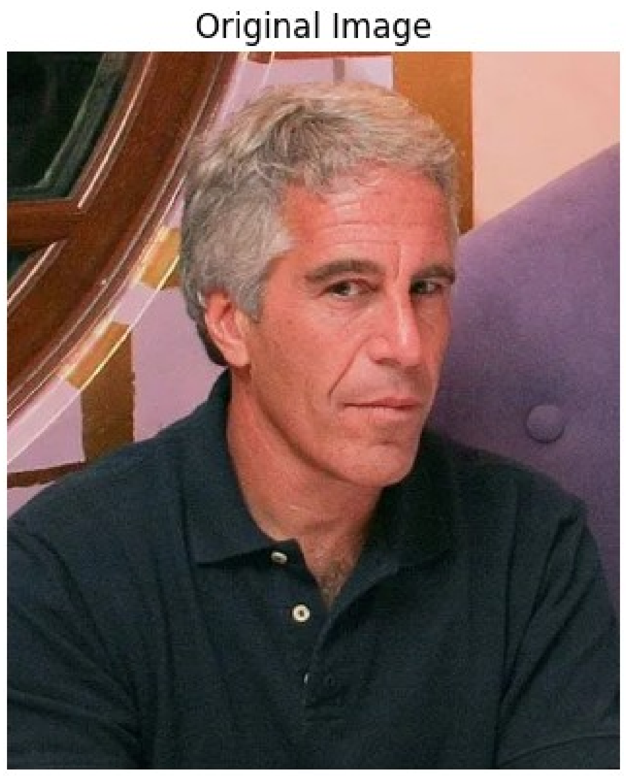
Input Image

```python
img = cv2.imread(r"D:\MIT WPU B.Tech Data\3nd Year Data\IPPR\Jeffrey-Epstein-Sex-offender.webp")


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

plt.imshow(img_rgb)
plt.title("Original Image")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-0.5))
```



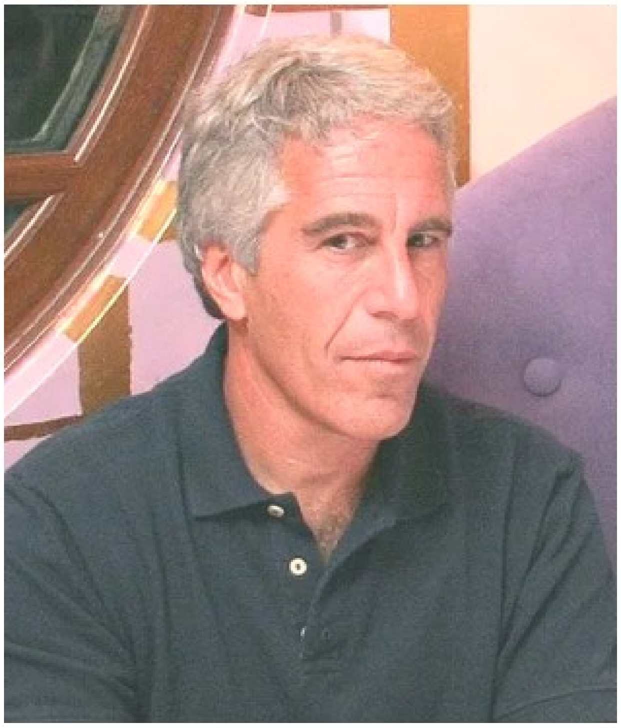
Arithmetic Operations

```
bright = cv2.add(img, np.ones(img.shape, dtype='uint8') * 50)
bright_rgb = cv2.cvtColor(bright, cv2.COLOR_BGR2RGB)

plt.imshow(bright_rgb)
plt.title("Image Addition (Brightness Increased)")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

Image Addition (Brightness Increased)

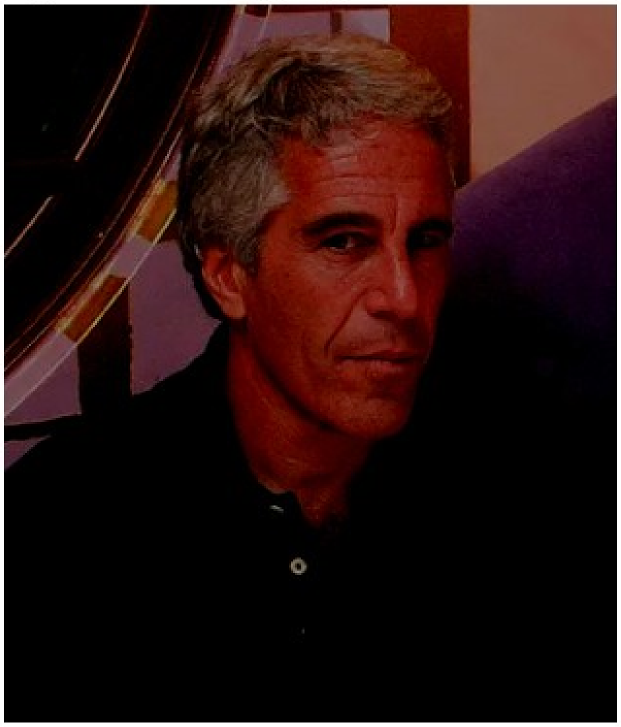


```
dark = cv2.subtract(img, np.ones(img.shape, dtype='uint8') * 100)
dark_rgb = cv2.cvtColor(dark, cv2.COLOR_BGR2RGB)

plt.imshow(dark_rgb)
plt.title("Image Subtraction (Brightness Decreased)")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

Image Subtraction (Brightness Decreased)



```
mul = cv2.multiply(img, np.array([1.2]))
mul_rgb = cv2.cvtColor(mul, cv2.COLOR_BGR2RGB)

plt.imshow(mul_rgb)
plt.title("Image Multiplication")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

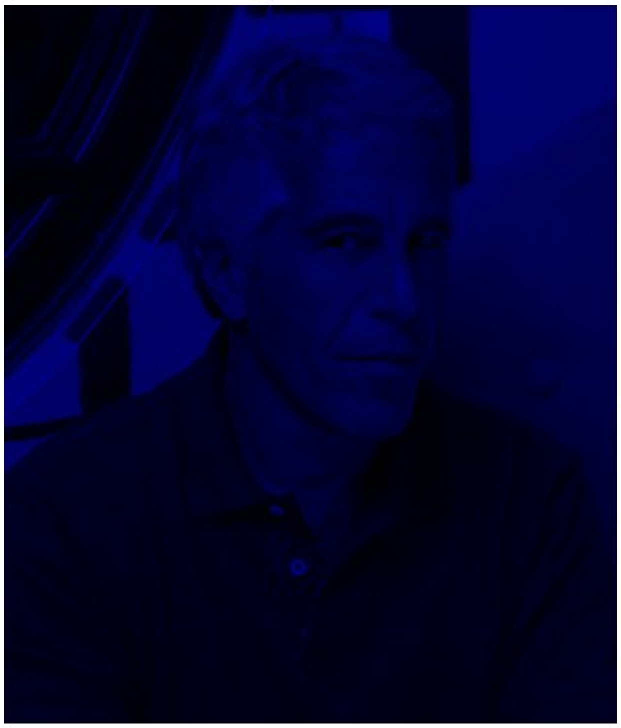
Image Multiplication

```
div = cv2.divide(img, np.array([1.5]))
div_rgb = cv2.cvtColor(div, cv2.COLOR_BGR2RGB)

plt.imshow(div_rgb)
plt.title("Image Division")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

Image Division

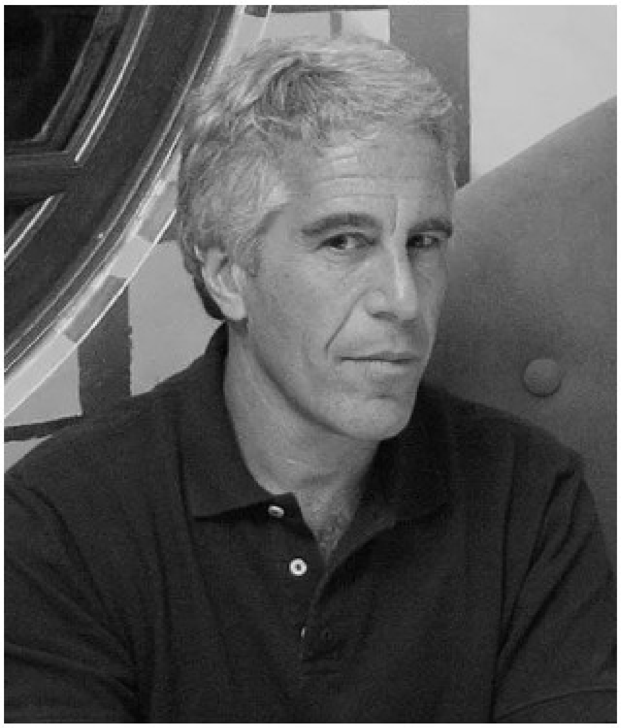


Logical Operations

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray')
plt.title("Grayscale Image")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

Grayscale Image



```
_, mask = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
plt.imshow(mask, cmap='gray')
plt.title("Binary Mask")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

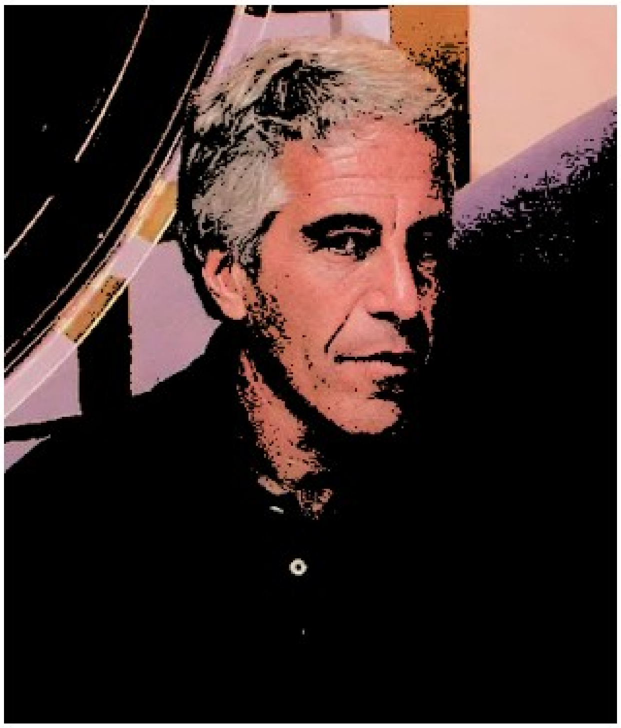
Binary Mask

```
bit_and = cv2.bitwise_and(img, img, mask=mask)
bit_and_rgb = cv2.cvtColor(bit_and, cv2.COLOR_BGR2RGB)

plt.imshow(bit_and_rgb)
plt.title("Bitwise AND")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

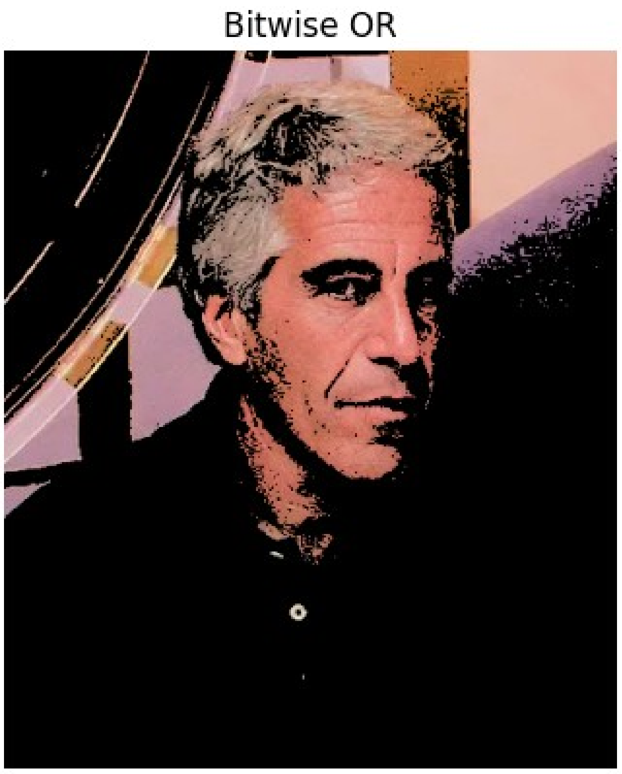
Bitwise AND



```
bit_or = cv2.bitwise_or(img, img, mask=mask)
bit_or_rgb = cv2.cvtColor(bit_or, cv2.COLOR_BGR2RGB)

plt.imshow(bit_or_rgb)
plt.title("Bitwise OR")
plt.axis("off")
```

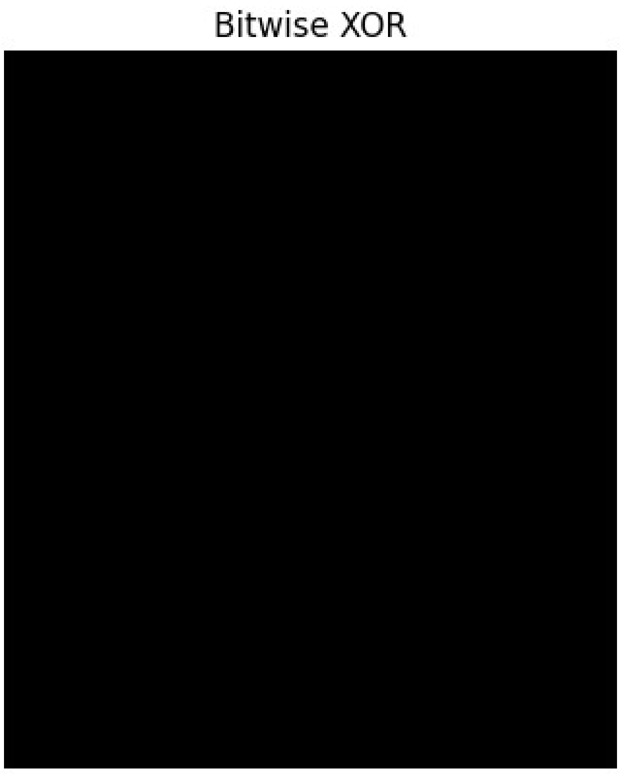
```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```



```
bit_xor = cv2.bitwise_xor(img, img, mask=mask)
bit_xor_rgb = cv2.cvtColor(bit_xor, cv2.COLOR_BGR2RGB)

plt.imshow(bit_xor_rgb)
plt.title("Bitwise XOR")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```



```
bit_not = cv2.bitwise_not(img)
bit_not_rgb = cv2.cvtColor(bit_not, cv2.COLOR_BGR2RGB)

plt.imshow(bit_not_rgb)
plt.title("Bitwise NOT (Negative Image)")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

Bitwise NOT (Negative Image)

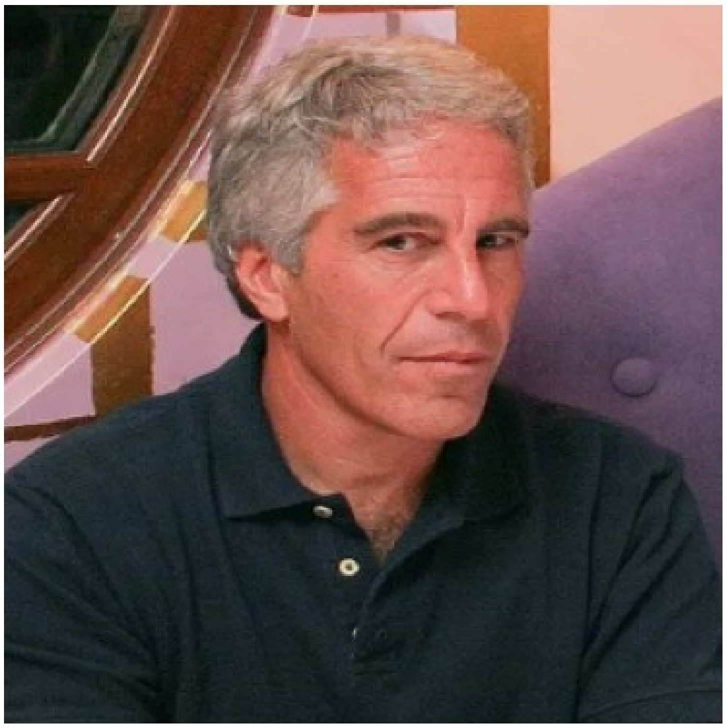
Image Resizing

```
resized = cv2.resize(img, (300, 300), interpolation=cv2.INTER_LINEAR)
resized_rgb = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)

plt.imshow(resized_rgb)
plt.title("Resized Image")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(299.5), np.float64(299.5), np.float64(-
0.5))
```

Resized Image



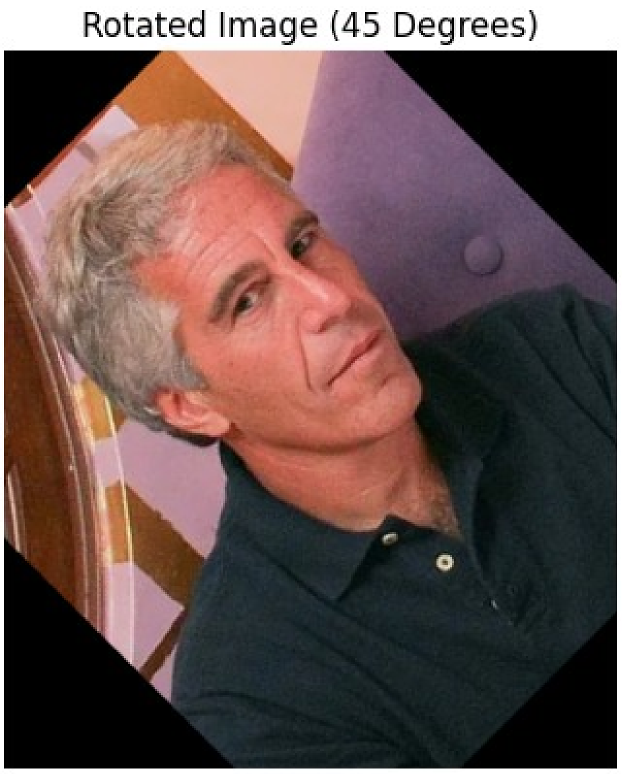
Rotation and Translation

```
(h, w) = img.shape[:2]
center = (w//2, h//2)

matrix = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated = cv2.warpAffine(img, matrix, (w, h))
rotated_rgb = cv2.cvtColor(rotated, cv2.COLOR_BGR2RGB)

plt.imshow(rotated_rgb)
plt.title("Rotated Image (45 Degrees)")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

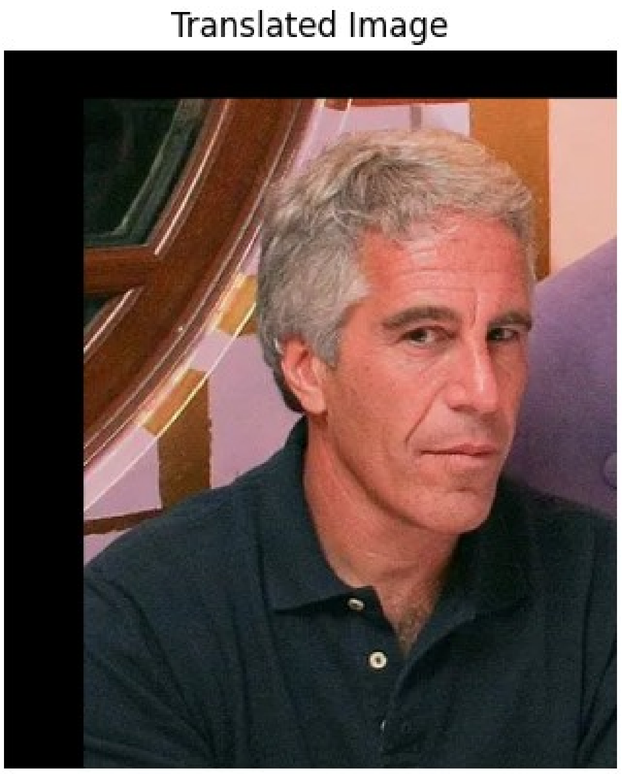


```
matrix_t = np.float32([[1, 0, 50],
                       [0, 1, 30]])

translated = cv2.warpAffine(img, matrix_t, (w, h))
translated_rgb = cv2.cvtColor(translated, cv2.COLOR_BGR2RGB)

plt.imshow(translated_rgb)
plt.title("Translated Image")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

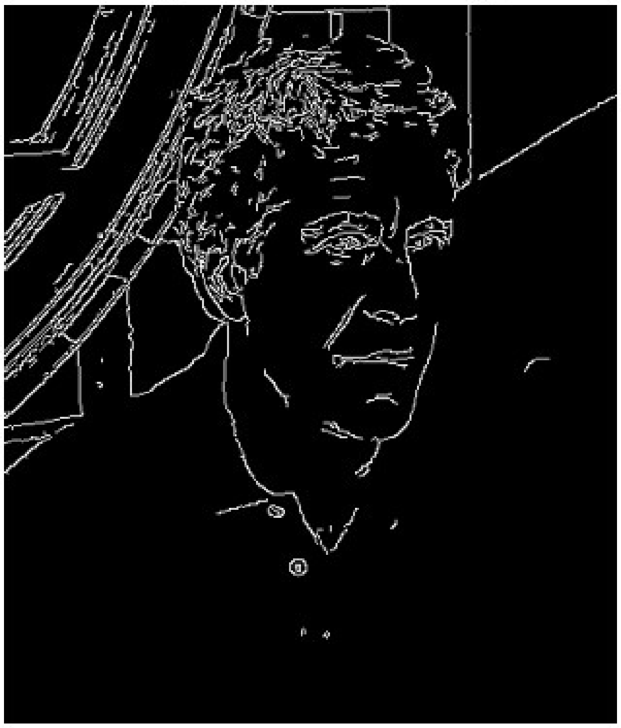


Additional Basic Operations

```python
edges = cv2.Canny(gray, 100, 200)
plt.imshow(edges, cmap='gray')
plt.title("Edge Detection (Canny)")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

Edge Detection (Canny)



```
blur = cv2.GaussianBlur(img, (7,7), 0)
blur_rgb = cv2.cvtColor(blur, cv2.COLOR_BGR2RGB)

plt.imshow(blur_rgb)
plt.title("Gaussian Blurred Image")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(383.5), np.float64(449.5), np.float64(-
0.5))
```

Gaussian Blurred Image

```
img1 = np.zeros((256, 256), dtype=np.uint8)

for i in range(256):
    for j in range(256):
        if j > i:
            img1[i, j] = 255   # White
        else:
            img1[i, j] = 0     # Black

plt.imshow(img1, cmap='gray')
plt.title("Diagonal Half Black–White Image")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(255.5), np.float64(255.5), np.float64(-
0.5))
```

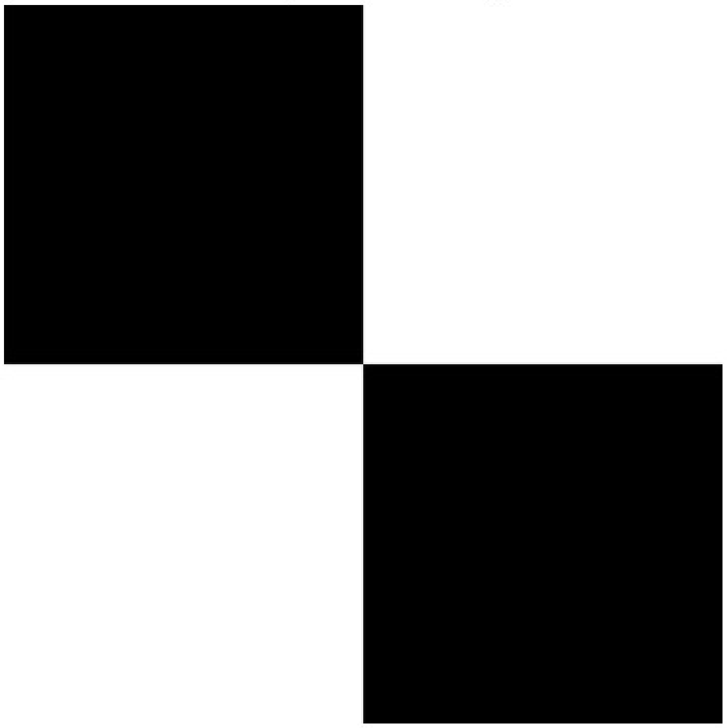
Diagonal Half Black–White Image

```
img2 = np.zeros((256, 256), dtype=np.uint8)

img2[0:128, 0:128] = 0
img2[0:128, 128:256] = 255
img2[128:256, 0:128] = 255
img2[128:256, 128:256] = 0
plt.imshow(img2, cmap='gray')
plt.title("2x2 Quadrant Image")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(255.5), np.float64(255.5), np.float64(-
0.5))
```

2x2 Quadrant Image



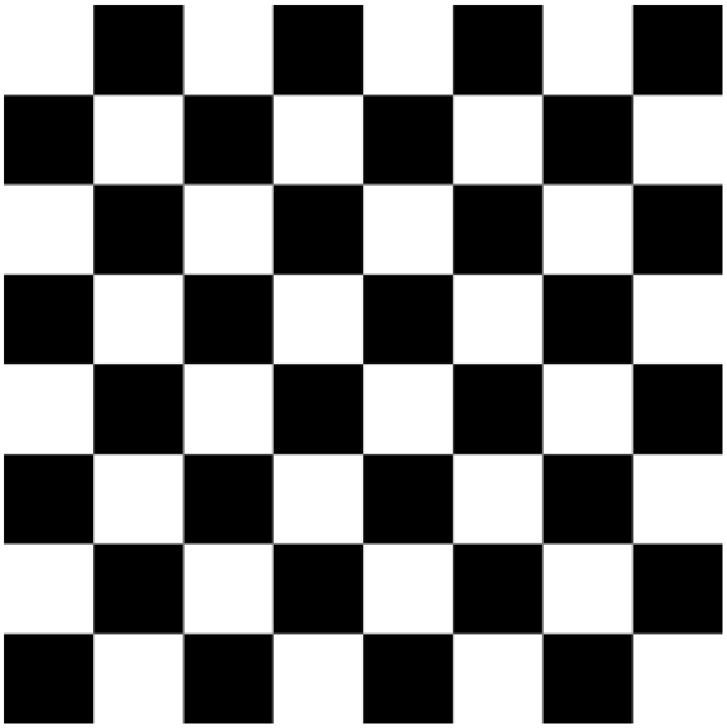
```
img3 = np.zeros((256, 256), dtype=np.uint8)
block = 32

for i in range(0, 256, block):
    for j in range(0, 256, block):
        if (i//block + j//block) % 2 == 0:
            img3[i:i+block, j:j+block] = 255

plt.imshow(img3, cmap='gray')
plt.title("Chessboard Pattern")
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(255.5), np.float64(255.5), np.float64(-
0.5))
```

Chessboard Pattern



Uzair Shaikh
1032240100

DATE

IPPR

[illegible] 22/1/26

* Exp-1

* Post lab Questions.

1) What do you mean by gray level?
→ In image processing, a grey level refers to the intensity value of a pixel in a grayscale image. It tells you how bright or dark pixel is.

2) Write the expression to find the number of bits to store a digital image.
→ The expression to store digital image using number of bits is.

Num of bits = M x N x K

✓

M - number of rows.
N - number of columns.
K - number of bits per pixel (bpp)

3) Name types of resolutions w.r.t a digital image.
→ The main types of resolution w.r.to a digital image are:

1) Spatial resolution
2) Grey level (intensity) Resolution
3) Temporal resolution.

4) Specify the elements of the DIP system.

→ A typical Digital Image processing system includes.

1) Image Acquisition
2) Image storage
3) Image Processing
4) Image Display
5) Image Transmission
6) Image Interpretation

figure ?

5) Write any four application of DIP

→ Common application of DIP are.

1) Medical Imaging (e.g, CT, MRI)
2) Remote sensing (satellite)
3) Robotics & Machine vision
4) Bio metrics (Face, fingerprint).

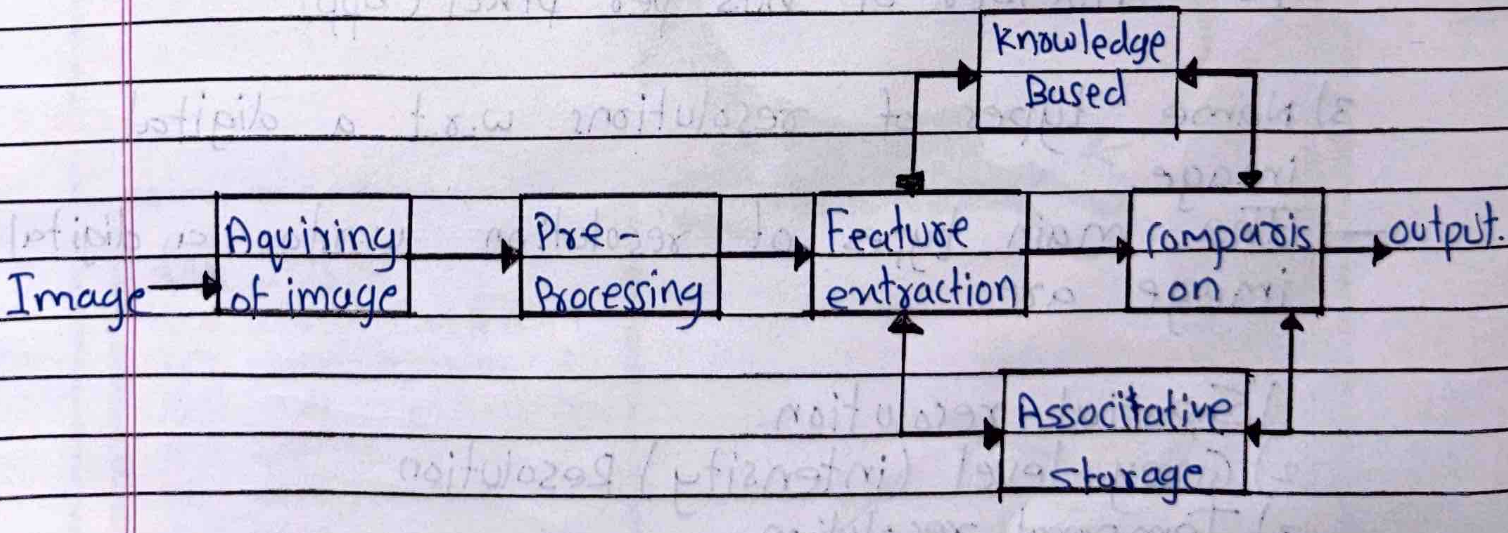


Image
Aquiring of image
Pre-Processing
Feature extraction
comparison
output.
Knowledge Based
Associtative storage