Logo

Description automatically generated**EAST WEST UNIVERSITY**

**CSE438**

**Section: 01**

**Lab: 01 Report**

**Topic: Image Operation, Threshold,** **4 connected neighborhoods, and 8 connected neighborhoods,** **Euclidean distance**

**Submitted By:**

**Name: Md Sifatullah Sheikh**

**ID: 2022-1-60-029**

**Submitted To:**

**K.M. Safin Kamal**

**Lecturer**

**Department of Computer Science & Engineering**

**Date: 1 July 2025**

**Q1. Determine the perimeter of an object by using 4 connected neighborhoods and 8 connected neighborhoods from Figure 1.**

**Answer:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

from skimage.measure import perimeter

from scipy.spatial import distance

fig\_01\_path = '/kaggle/input/lab-01/Lab\_01/fig\_01.png'

fig\_01 = cv2.imread(fig\_01\_path,cv2.IMREAD\_GRAYSCALE)

def show\_image(fig\_01\_path, title = "", cmap='gray'):

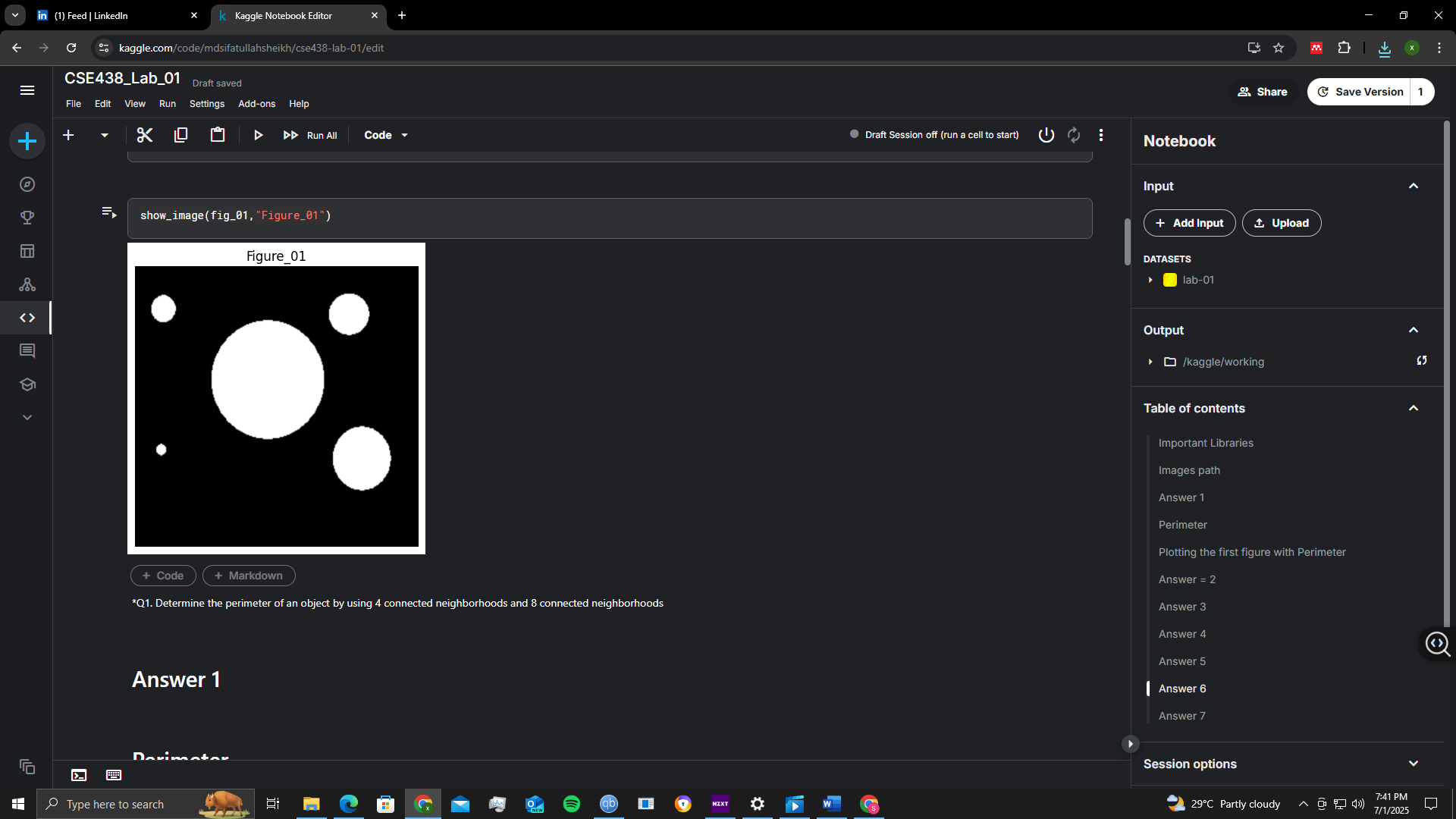
plt.imshow(fig\_01\_path, cmap = cmap)

plt.title (title)

plt.axis ('off')

plt.show

show\_image(fig\_01,"Figure\_01")



from scipy.ndimage import binary\_erosion

\_, binary = cv2.threshold(fig\_01, 127, 1, cv2.THRESH\_BINARY)

def compute\_perimeter(binary\_img, connectivity=4):

struct = np.array([[0, 1, 0],

[1, 1, 1],

[0, 1, 0]]) if connectivity == 4 else np.ones((3, 3))

eroded = binary\_erosion(binary\_img, structure=struct).astype(int)

perimeter = binary\_img - eroded

return perimeter, int(np.sum(perimeter))

perim4\_img, perim4\_count = compute\_perimeter(binary, connectivity = 4)

perim8\_img, perim8\_count = compute\_perimeter(binary, connectivity = 8)

# Display original and both perimeter results

plt.figure(figsize=(18, 6))

# Original binary image

plt.subplot(1, 3, 1)

plt.title('Original Binary Image')

plt.imshow(binary, cmap='gray')

plt.axis('off')

# 4-connected perimeter

plt.subplot(1, 3, 2)

plt.title(f'Perimeter (4-connected): {perim4\_count}')

plt.imshow(perim4\_img, cmap='gray')

plt.axis('off')

# 8-connected perimeter

plt.subplot(1, 3, 3)

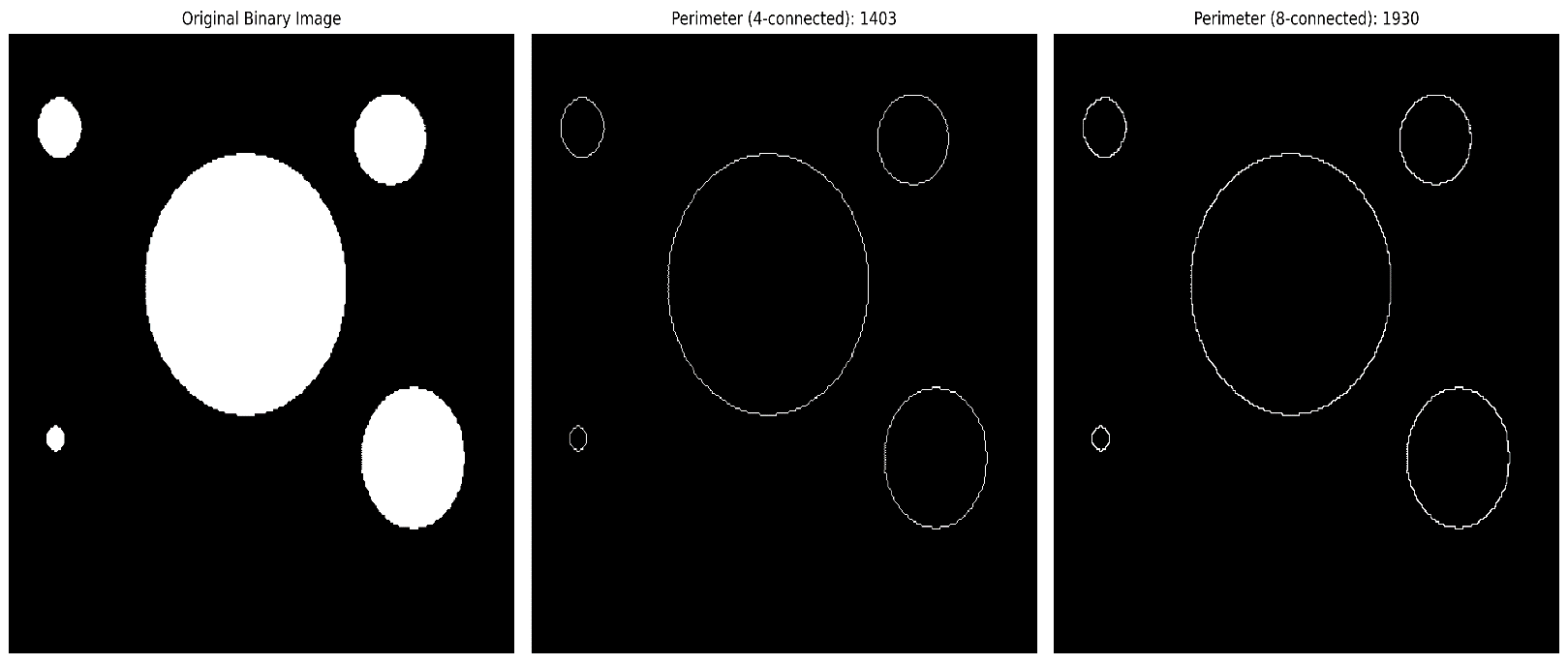
plt.title(f'Perimeter (8-connected): {perim8\_count}')

plt.imshow(perim8\_img, cmap='gray')

plt.axis('off')

plt.tight\_layout()

plt.show()



**Q2. Create a binary image using a threshold for Figure 2.**

**Answer:**

def show\_image(fig\_02\_path, title = "", cmap='gray'):

plt.imshow(fig\_02\_path, cmap = cmap)

plt.title (title)

plt.axis ('off')

plt.show

import os

import cv2

os.makedirs('/kaggle/working/plots', exist\_ok=True)

\_, binary\_img = cv2.threshold(fig\_02, 127, 255, cv2.THRESH\_BINARY)

cv2.imwrite('/kaggle/working/plots/Figure\_02\_binary.png', binary\_img)

def show\_image(image, title):

plt.imshow(image, cmap='gray')

plt.title(title)

plt.axis('off')

plt.show()

show\_image(binary\_img, "Figure\_02")



**Q3. Determine the number of objects in the binary image generated in Question 2 using the**

**Concept of connectivity.**

**Answer:**

num\_labels, labels = cv2.connectedComponents(binary\_img)

num\_objects = num\_labels - 1

print("Number of objects in binary image:", num\_objects)

Number of objects in binary image: 256

**Q4. Find the Euclidean distance between two points of the image.**

**Answer:**

import numpy as np

point1 = (30.67, 40.67)

point2 = (100.25, 80.25)

distance = np.sqrt((point2[0] - point1[0])\*\*2 + (point2[1] - point1[1])\*\*2)

print("Euclidean distance between the points:", distance)

Euclidean distance between the points: 80.04968956841743

**Q5. Apply the following operations using Fig.1 and Fig.2:**

**a. Addition**

**b. Subtraction**

**c. Multiplication**

**d. Division**

**Answer:**

fig\_02 = cv2.resize(fig\_02, (fig\_01.shape[1], fig\_01.shape[0]))

add\_img = cv2.add(fig\_01, fig\_02)

sub\_img = cv2.subtract(fig\_01, fig\_02)

mult\_img = cv2.multiply(fig\_01, fig\_02)

div\_img = cv2.divide(fig\_01, fig\_02 + 1) # Avoid division by zero

titles = ['Original - Fig 01', 'Original - Fig 02', 'Addition', 'Subtraction', 'Multiplication', 'Division']

images = [fig\_01, fig\_02, add\_img, sub\_img, mult\_img, div\_img]

os.makedirs('/kaggle/working/plots', exist\_ok=True)

plt.figure(figsize=(12, 8))

for i in range(6):

plt.subplot(2, 3, i + 1)

plt.imshow(images[i], cmap='gray')

plt.title(titles[i])

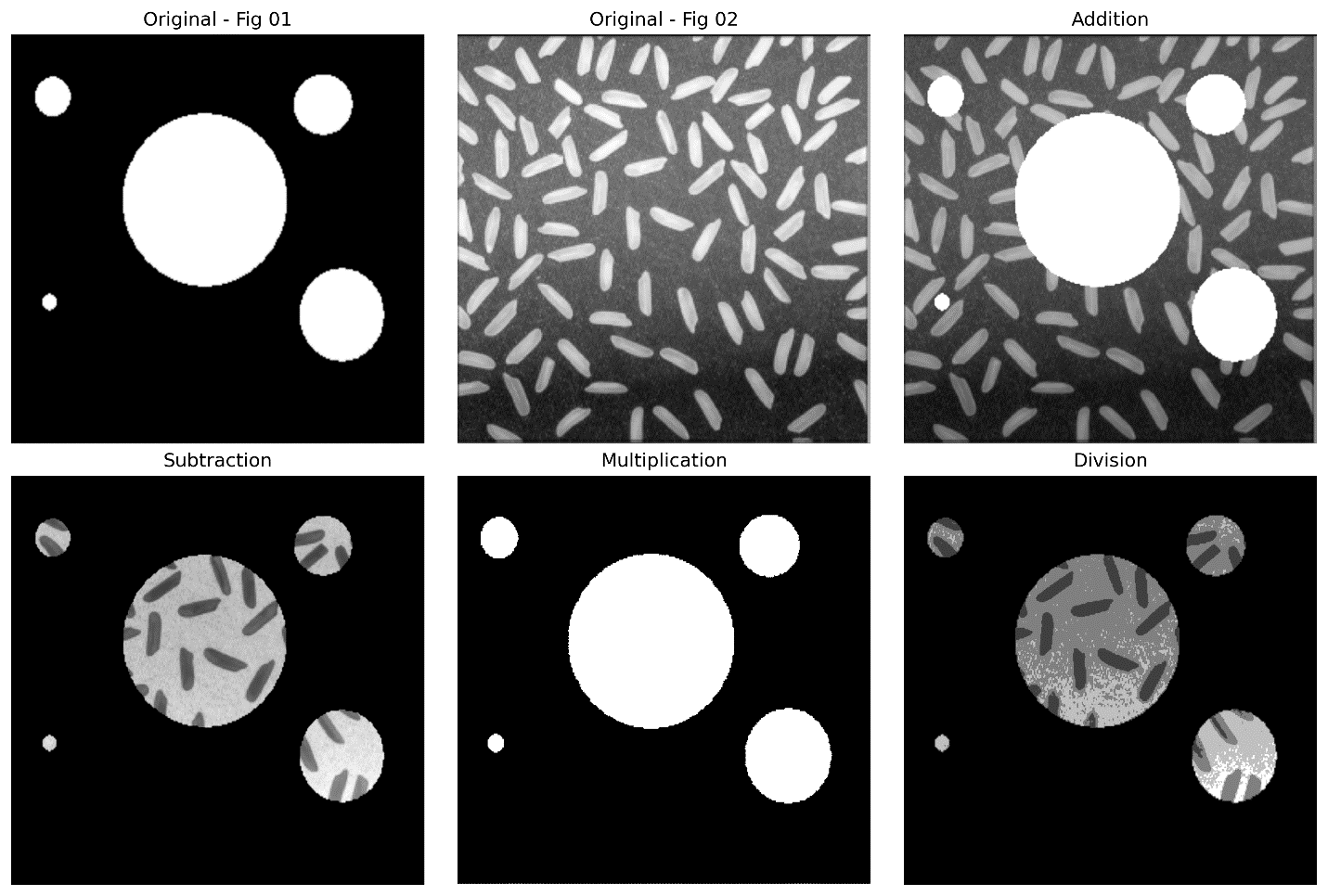
plt.axis('off')

plt.tight\_layout()

# Save the figure

plt.savefig('/kaggle/working/plots/pixelwise\_operations.png', dpi=300, bbox\_inches='tight')

plt.show()



**Q6. Apply the following operations using Fig.1 and Fig.2:**

**a. AND**

**b. OR**

**c. NOT**

**Answer:**

fig\_02 = cv2.resize(fig\_02, (fig\_01.shape[1], fig\_01.shape[0]))

and\_img = cv2.bitwise\_and(fig\_01, fig\_02)

or\_img = cv2.bitwise\_or(fig\_01, fig\_02)

not\_img\_01 = cv2.bitwise\_not(fig\_01)

not\_img\_02 = cv2.bitwise\_not(fig\_02)

titles = ['Original - Fig 01', 'Original - Fig 02', 'Bitwise AND', 'Bitwise OR', 'NOT Fig 01', 'NOT Fig 02']

images = [fig\_01, fig\_02, and\_img, or\_img, not\_img\_01, not\_img\_02]

os.makedirs('/kaggle/working/plots', exist\_ok=True)

plt.figure(figsize=(12, 8))

for i in range(6):

plt.subplot(2, 3, i + 1)

plt.imshow(images[i], cmap='gray')

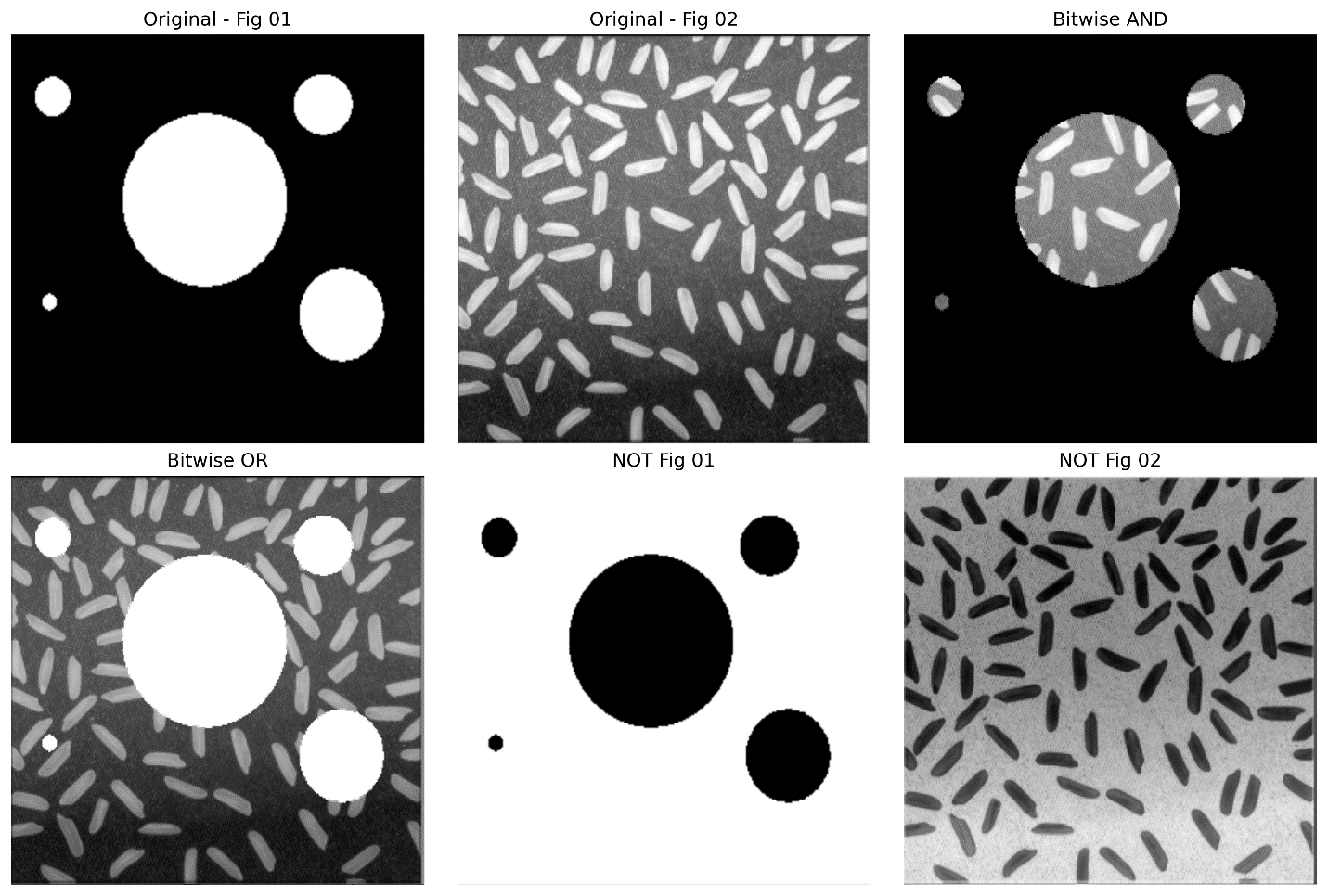
plt.title(titles[i])

plt.axis('off')

plt.tight\_layout()

plt.savefig('/kaggle/working/plots/bitwise\_operations.png', dpi=300, bbox\_inches='tight')

plt.show()



**Q7. Adjust the contrast of the following image.**

**Answer:**

import os

import matplotlib.pyplot as plt

import cv2

os.makedirs('/kaggle/working/plots', exist\_ok=True)

alpha = 1.5

beta = 0

adjusted = cv2.convertScaleAbs(fig\_01, alpha=alpha, beta=beta)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Original Image')

plt.imshow(fig\_01, cmap='gray')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.title('Contrast Adjusted')

plt.imshow(adjusted, cmap='gray')

plt.axis('off')

plt.tight\_layout()

plt.savefig('/kaggle/working/plots/contrast\_adjustment.png', dpi=300, bbox\_inches='tight')

plt.show()

