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

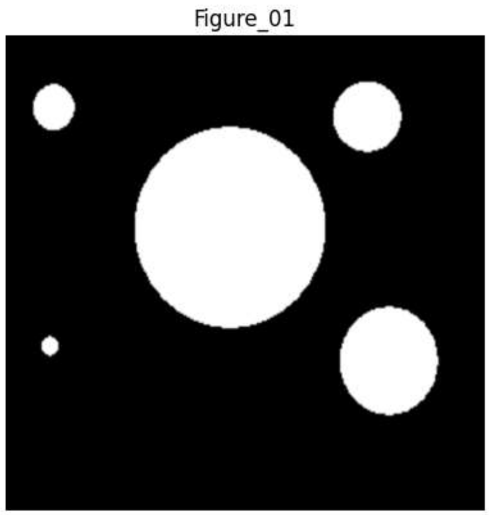
Department of Computer Science & Engineering

Date: 1 July 2025

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

Answer:

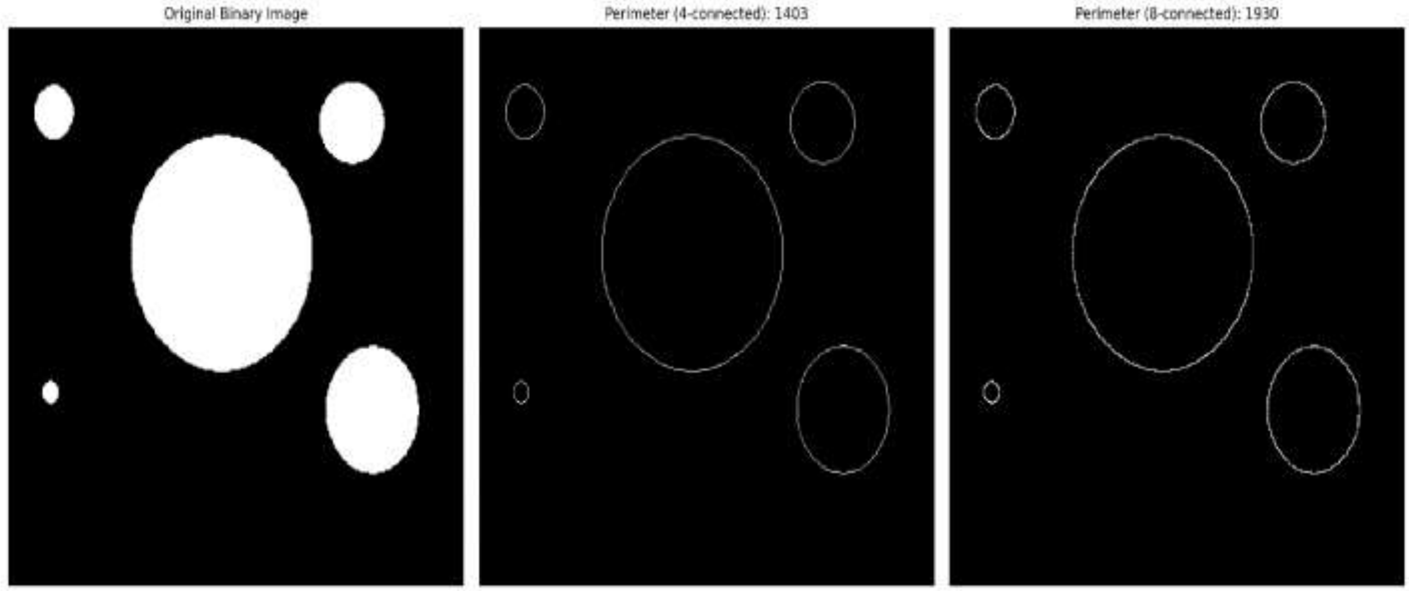
```
import cv2

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')
```

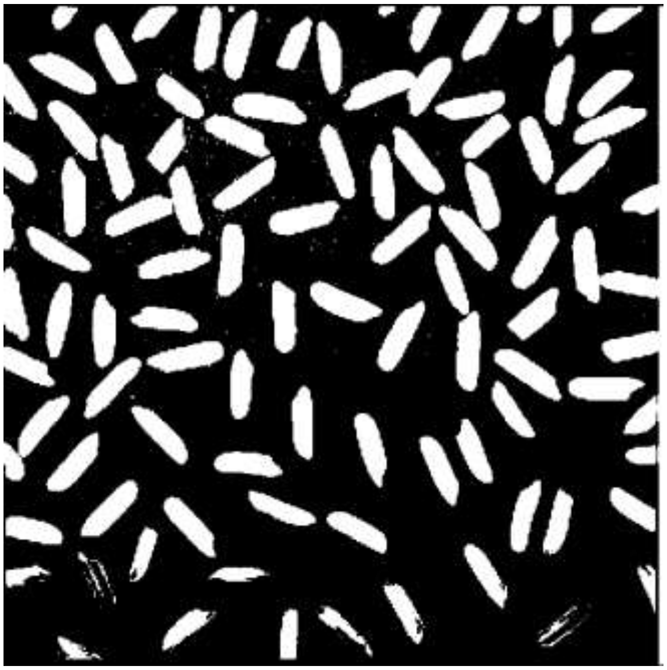
```
# 8-connected perimeter
plt.subplot(1, 3, 3)
plt.title(f'Perimeter (8-connected): {perim8_count}')
plt.imshow(perim8_img, cmap='gray')
plt.axis('off')

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



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

Answer:

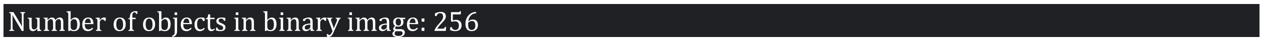
```
def show_image(fig_02_path, title = "", cmap='gray'):
    plt.imshow(fig_02_path, cmap = cmap)
    plt.title (title)
    plt.axis ('off')

    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

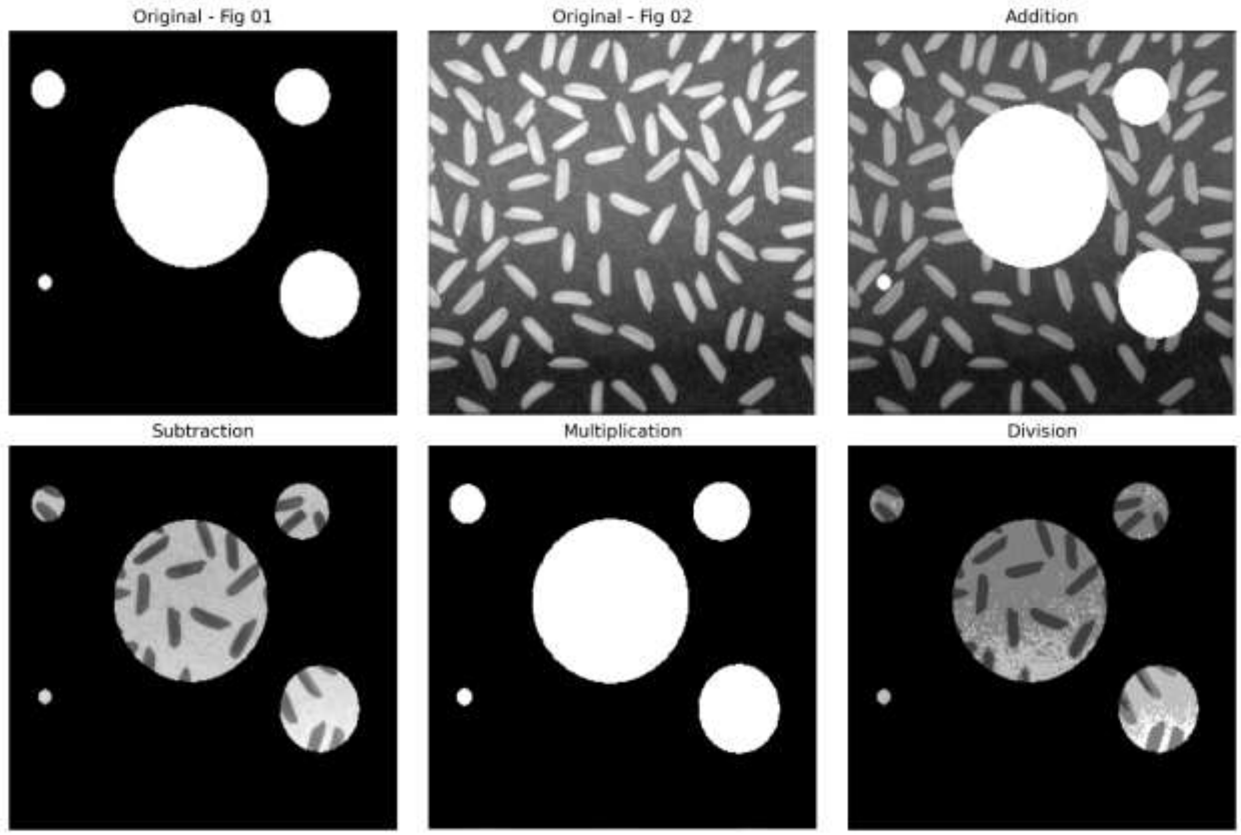
c. Multiplication

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()
```



Original - Fig 01

Original - Fig 02

Addition

Subtraction

Multiplication

Division

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

a. AND

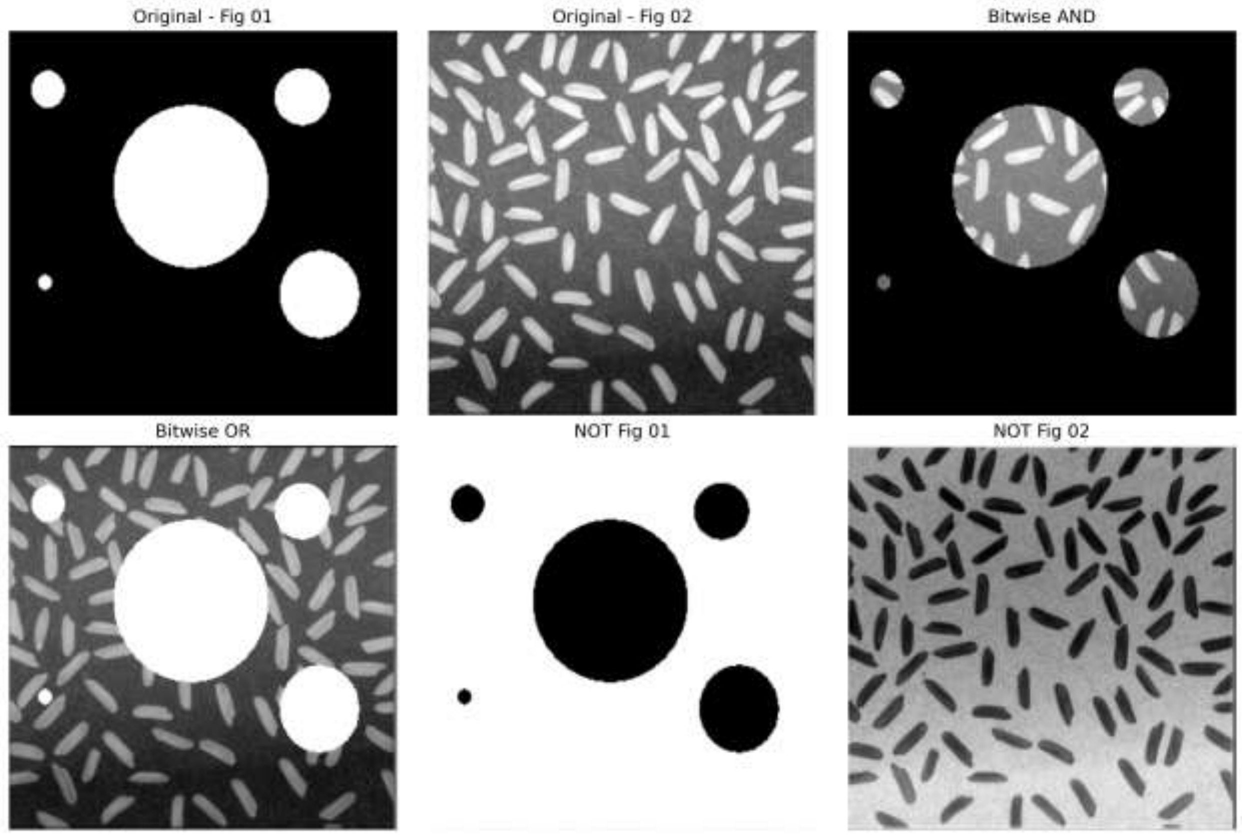
b. OR

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()
```

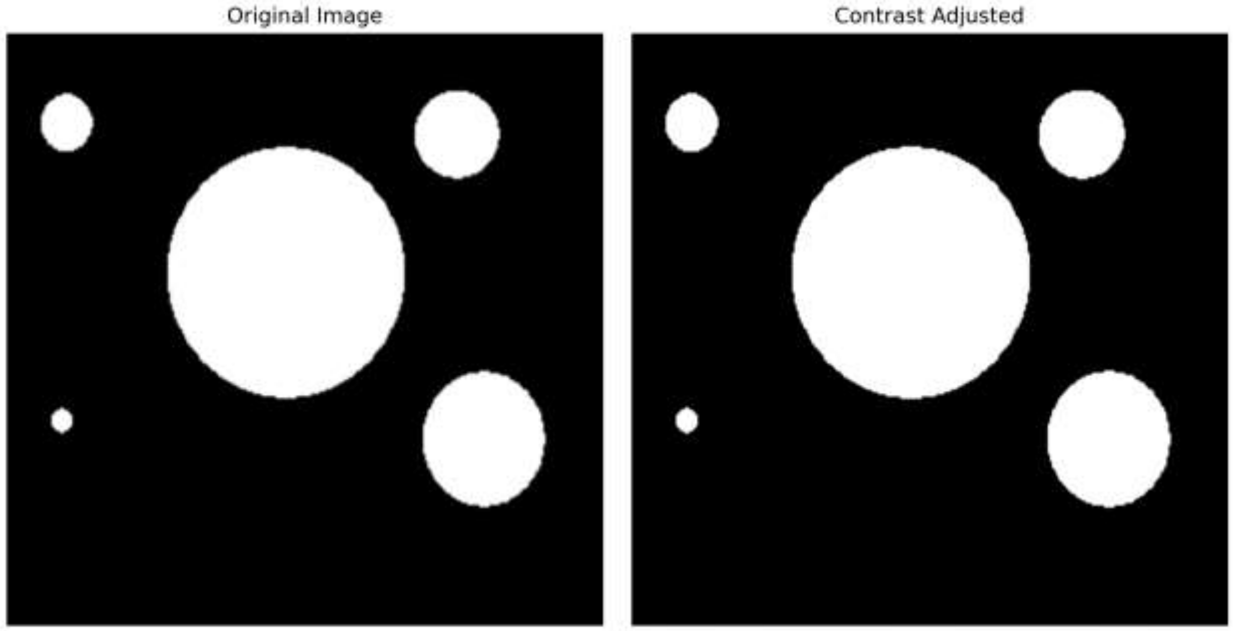


Q7. Adjust the contrast of the following image.

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()
```


Original Image
Contrast Adjusted