EC7212: COMPUTER VISION AND IMAGE PROCESSING

TAKE HOME ASSIGNMENT 2

NAME	:	SUNDARASEKARA G.O.
REG No.	:	EG/2020/3943
SEMESTER	:	07
DATE	:	27/06/2025

Table of Contents

List of Figures

1 Task 1 – Otsu's Thresholding with Gaussian Noise

1.1 Objective

Create a grayscale synthetic image with two objects and a background, add Gaussian noise, and apply Otsu's thresholding to segment the image.

1.2 Implementation Details

- Image: 100×100 pixels
  - Object 1: pixel value = 85
  - Object 2: pixel value = 170
  - Background: pixel value = 0
- Noise: Gaussian noise with mean = 0, stddev = 10 using NumPy
- Segmentation: Otsu's automatic thresholding using cv2.threshold()

1.3 Code

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

# Create a synthetic image: 100x100, background=0, object1=85, object2=170
image = np.zeros((100, 100), dtype=np.uint8)
image[20:50, 20:50] = 85      # Object 1
image[60:90, 60:90] = 170     # Object 2

# Add Gaussian noise
mean = 0
stddev = 10
noise = np.random.normal(mean, stddev, image.shape).astype(np.int16)
noisy_image = np.clip(image.astype(np.int16) + noise, 0, 255).astype(np.uint8)

# Apply Otsu's thresholding
_, otsu_thresh = cv2.threshold(noisy_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Display results
plt.figure(figsize=(10,4))
plt.subplot(1,3,1)
plt.imshow(image, cmap='gray')
plt.title("Original")
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(noisy_image, cmap='gray')
plt.title("Noisy Image")
plt.axis('off')

plt.subplot(1,3,3)
plt.imshow(otsu_thresh, cmap='gray')
plt.title("Otsu Thresholding")
plt.axis('off')
plt.savefig("otsu_result.png")
```

FIGURE 1: TASK1 – CODE FOR OTSU GAUSSIAN SEGMENTATION

1.4Result

Otsu's method successfully found a threshold to separate noisy object regions from the background.

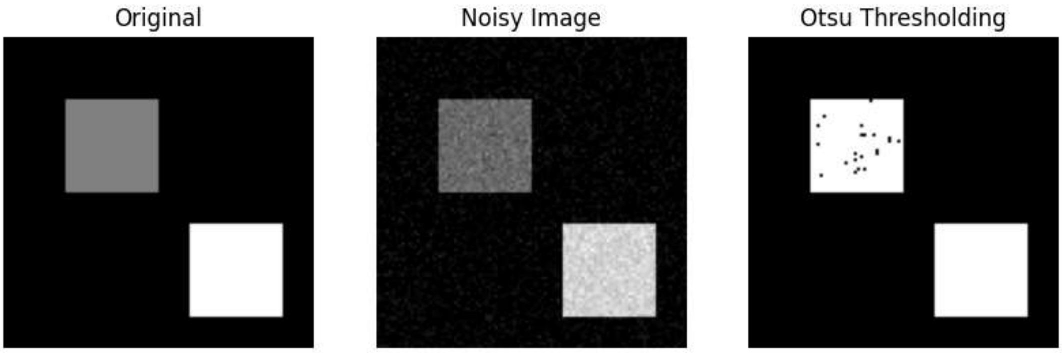

FIGURE 2: RESULT FROM TASK1 PROGRAM EXECUTION

2 Task 2 – Region Growing Segmentation

2.1 Objective

To implement a region growing algorithm that segments image regions based on intensity similarity, starting from seed points.

2.2 Implementation Details

- A 200×200 grayscale image was created with:
  - A rectangle (pixel value 100)
  - A circle (pixel value 200)
- Region growing algorithm was implemented using 8-connected neighbors.
- Two seed points were manually chosen inside the rectangle and circle.
- The algorithm expanded regions by checking if neighboring pixels fall within a threshold difference (≤15).

2.3 Code

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from collections import deque

def region_growing(img, seeds, threshold=10):
    height, width = img.shape
    visited = np.zeros((height, width), dtype=bool)
    output = np.zeros((height, width), dtype=np.uint8)

    # Take intensity from the first seed
    seed_x, seed_y = seeds[0]
    seed_value = img[seed_y, seed_x]
    queue = deque(seeds)

    directions = [(-1, -1), (-1, 0), (-1, 1),
                  (0, -1),           (0, 1),
                  (1, -1),  (1, 0),  (1, 1)]

    while queue:
        x, y = queue.popleft()

        if visited[y, x]:
            continue

        if abs(int(img[y, x]) - int(seed_value)) <= threshold:
            output[y, x] = 255
            visited[y, x] = True

            for dx, dy in directions:
                nx, ny = x + dx, y + dy
                if 0 <= nx < width and 0 <= ny < height and not visited[ny, nx]:
                    queue.append((nx, ny))

    return output
```

FIGURE 3: TASK2 – CODE FOR REGION GROWING SEGMENTATION (REGION_GROWING() FUNCTION)

```python
# Create blank image
img = np.zeros((200, 200), dtype=np.uint8)

# Object 1: Rectangle
top_left = (40, 40)
bottom_right = (90, 100)
cv2.rectangle(img, top_left, bottom_right, 100, -1)  # Fill rectangle with pixel value 100

# Object 2: Circle
center = (150, 150)
radius = 30
cv2.circle(img, center, radius, 200, -1)  # Fill circle with pixel value 200

# Seeds inside each object
seed_rect = [(60, 60)]   # inside rectangle
seed_circle = [(150, 150)]  # inside circle

# Region growing for each shape
region_rect = region_growing(img, seed_rect, threshold=15)
region_circle = region_growing(img, seed_circle, threshold=15)

# Combine the two segmented regions
combined_region = cv2.bitwise_or(region_rect, region_circle)

# Plot the results
fig, axes = plt.subplots(1, 4, figsize=(16, 4))

titles = ["Input Image", "Rectangle Region", "Circle Region", "Combined Region"]
images = [img, region_rect, region_circle, combined_region]

for ax, title, image in zip(axes, titles, images):
    ax.imshow(image, cmap='gray')
    ax.set_title(title)
    ax.axis('off')

plt.tight_layout()
plt.savefig("region_growing_result.png")
```

FIGURE 4: TASK2 – REST OF THE CODE FOR REGION GROWING SEGMENTATION

2.4 Result

Region growing correctly segmented both rectangle and circle shapes.

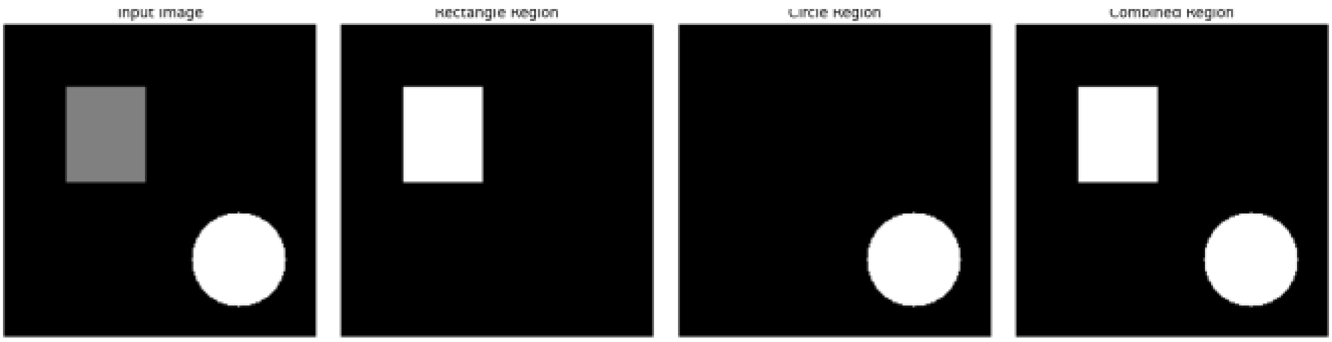

FIGURE 5: RESULT FROM TASK2 PROGRAM EXECUTION

3 GitHub link

The complete code and results are available at:

https://github.com/Githmi123/EC7212-Computer-Vision-and-Image-Processing-Take-Home-Assignment-2

4 Conclusion

In this assignment, we used Otsu’s method to segment a noisy image and region growing to extract two objects. Both methods worked well and helped us understand basic image segmentation clearly.