EC7212 Computer Vision and Image Processing

TAKE HOME ASSIGNMENT 2

Index No : EG/2020/4307

Date : 06. 27. 2025

CONTENTS

List of Figures

1. Introduction

This report demonstrates how to perform simple fundamental image segmentation techniques on a synthetic image using python programming language. A test image containing two distinct object with different intensity values and a uniform background is created. Gaussian noise is added to simulate real-world imperfections. The noisy image is then segmented using:

- **Otsu's automatic thresholding**, a global thresholding technique.
- **Region growing**, a local region-based segmentation approach starting from a user-selected seed point Rotation images

This experiment helps to understand how global and local segmentation methods behave under noisy conditions.

2 Software and Tools Used

Tool	Purpose
Python 3.13.3	Main programming language
OpenCV	Image creation, manipulation, and thresholding
NumPy	Matrix and numerical operations
Matplotlib	Visualization of images and results
VS Code	Development environment

3 Background and Theory

3.1 Gaussian Noise

- Random variation in pixel values following a normal (Gaussian) distribution
- Simulates real-world noise from sensors and environment used here to test how robust segmentation methods are under noisy conditions.

3.2 Otsu's Thresholding

- Automatic global threshold selection minimizes intra-class variance and maximizes inter-class variance in a histogram.
- Provides a simple, unsupervised way to binarize an image without manually picking a threshold.

3.3 Region Growing

- A region-based segmentation method that expands from a user-selected seed point by including neighboring pixels with similar intensity values
- Effective for local segmentation and for images with gradual intensity variation

4. Methodology and Results

4.1 Create Synthetic Image and Add Gaussian Noise

```python
def create_synthetic_image():
    img = np.zeros((100, 100), dtype=np.uint8)
    img[20:50, 20:50] = 100   # Object 1
    img[60:90, 60:90] = 200   # Object 2
    return img

img = create_synthetic_image()
show_image("Original Synthetic Image", img)
cv2.imwrite(f"{output_dir}/synthetic_image.png", img)
```

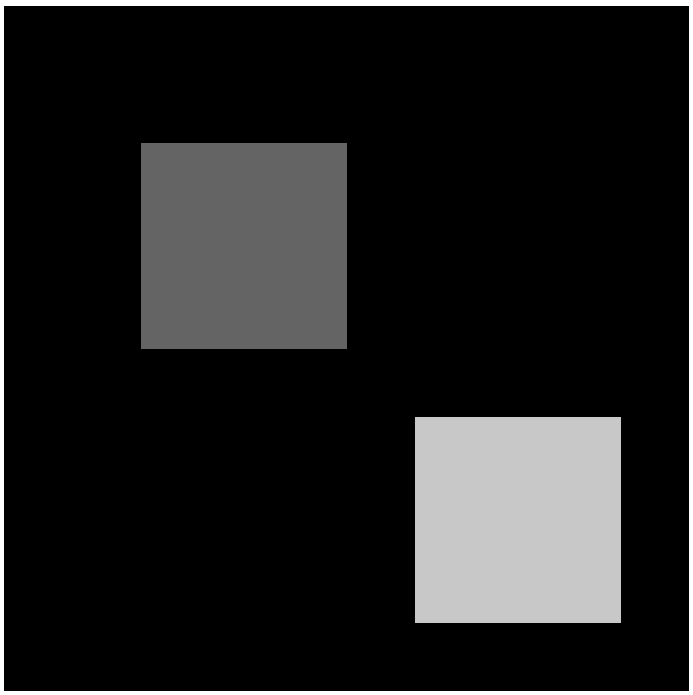

Figure 1: A synthetic grayscale image

```python
def add_gaussian_noise(image, mean=0, sigma=10):
    gauss = np.random.normal(mean, sigma, image.shape).astype(np.float32)
    noisy = image.astype(np.float32) + gauss
    noisy = np.clip(noisy, 0, 255).astype(np.uint8)
    return noisy

noisy_img = add_gaussian_noise(img)
show_image("Noisy Image", noisy_img)
cv2.imwrite(f"{output_dir}/noisy_image.png", noisy_img)
```

✓ 0.1s

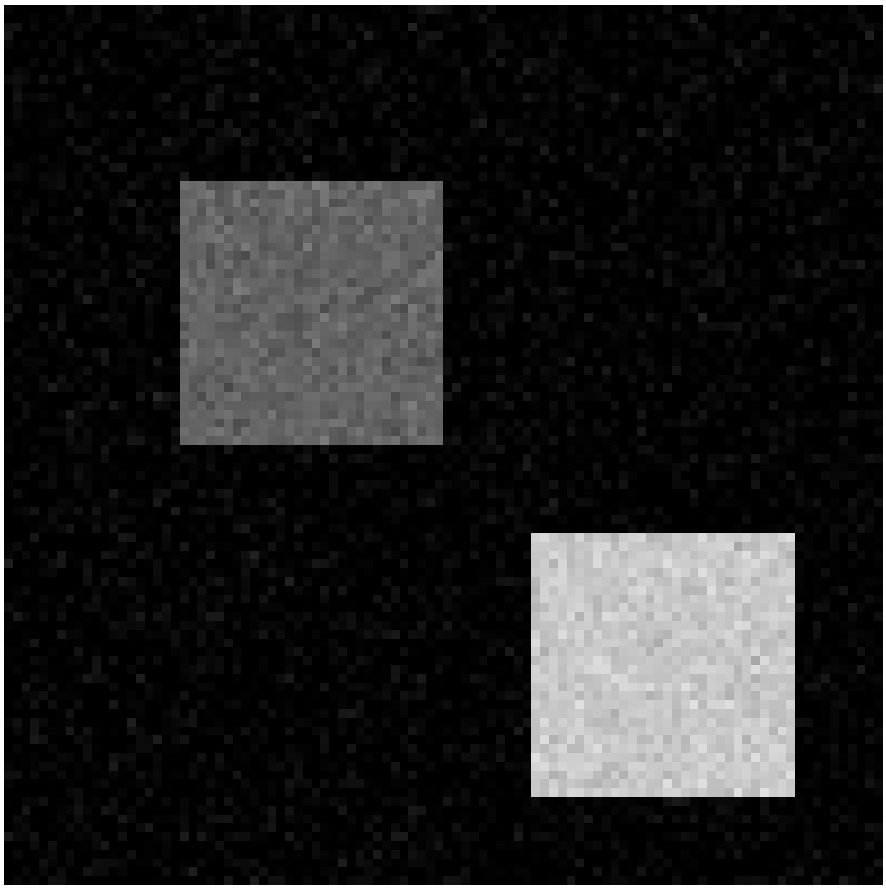

Figure 2: Gaussian noise image

4.2 Apply Otsu's Thresholding

```python
def otsu_threshold(image):
    _, thresh_img = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    return thresh_img

otsu_img = otsu_threshold(noisy_img)
show_image("Otsu Thresholding Result", otsu_img)
cv2.imwrite(f"{output_dir}/otsu_result.png", otsu_img)
```

✓ 0.0s

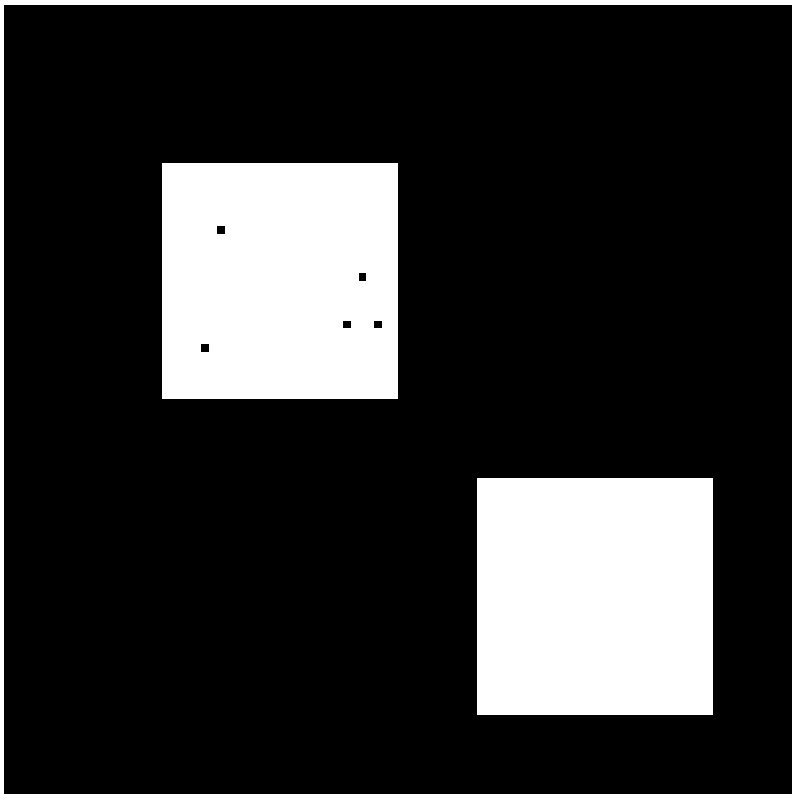

Figure 3: Otsu's Result

4.3 Implement Region Growing

```
def region_growing(image, seed, threshold=20):
    h, w = image.shape
    visited = np.zeros_like(image, dtype=np.uint8)
    region = np.zeros_like(image, dtype=np.uint8)

    seed_val = int(image[seed])
    queue = 
    visited[seed] = 1
    region[seed] = 255

    while queue:
        y, x = queue.popleft()
        for dy in [-1, 0, 1]:
            for dx in [-1, 0, 1]:
                ny, nx = y + dy, x + dx
                if 0 <= ny < h and 0 <= nx < w and not visited[ny, nx]:
                    pixel_val = int(image[ny, nx])
                    if abs(pixel_val - seed_val) <= threshold:
                        queue.append((ny, nx))
                        visited[ny, nx] = 1
                        region[ny, nx] = 255

    return region
```

(parameter) seed: Any

```
# Example seed point (make sure it is within the image dimensions)
seed_point = (3490, 2187)  # (x, y)

# Run region growing (assuming you have a function called region_growing)
segmented = region_growing(img, seed=seed_point, threshold=25)
```

Figure 4 : Original Image

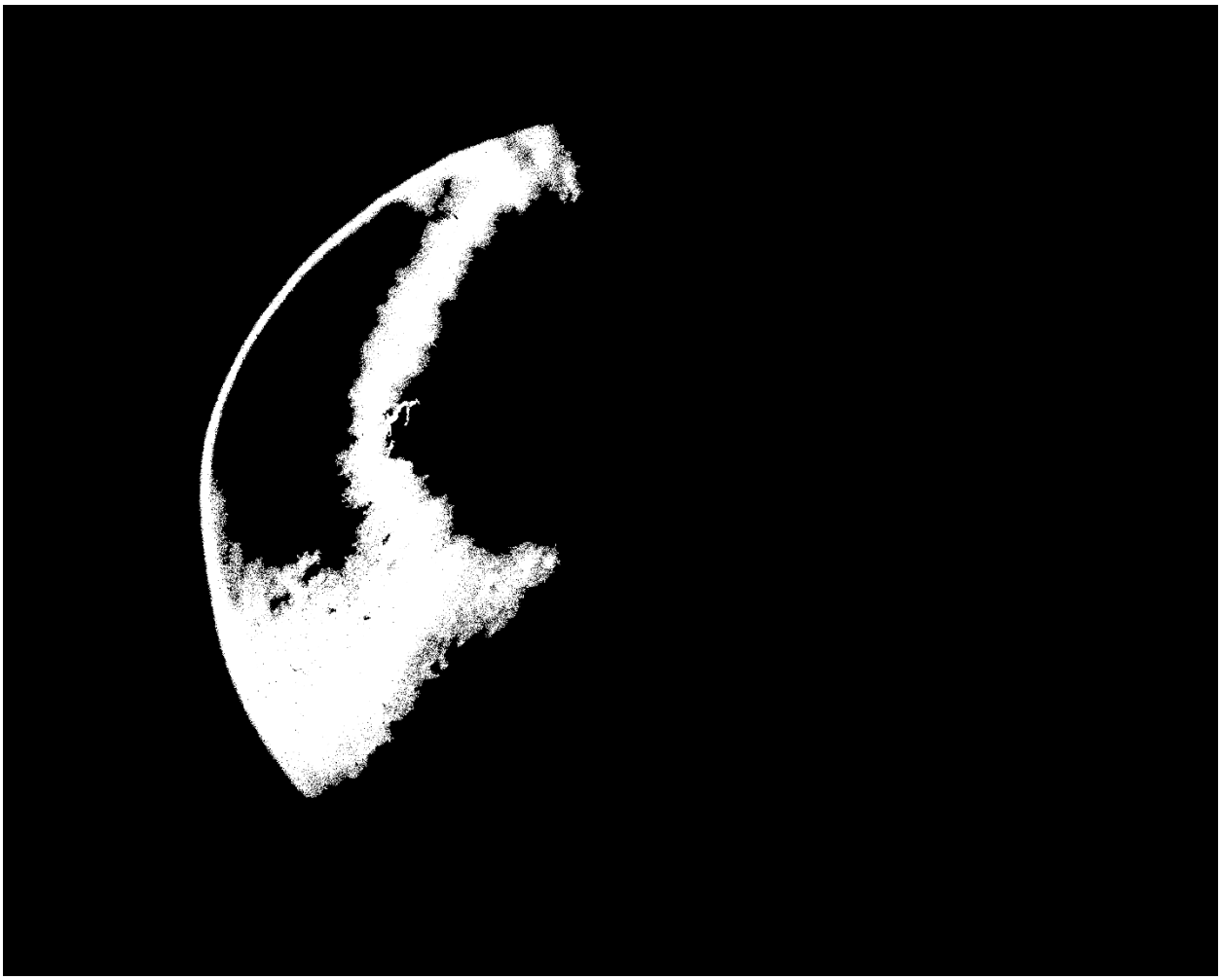

Figure 5 : Region Growing

5 GitHub Link

GitHub repo : https://github.com/srijithyaparathna/EC-7212-Computer-Vision-and-Image-Processing-TakeHomeAssignment2