EC7212 - Computer Vision and Image Processing

Assignment 2

Submitted by:

Tharsi S.
Registration Number: EG/2020/4232
Department of Computer Engineering
University of Ruhuna

Task Overview

1. Generate a synthetic image with 2 objects and a background (3 pixel values total). Add Gaussian noise and apply Otsu's thresholding.

2. Implement a region-growing segmentation method using seed points and a predefined intensity range.

Task 1: Otsu's Thresholding with Gaussian Noise

A 100x100 synthetic image was created with:

- Background (pixel value: 0)
- Object 1 (pixel value: 100)
- Object 2 (pixel value: 200)

Gaussian noise (mean = 0, sigma = 20) was added. Otsu's algorithm was applied to determine the optimal threshold.

Example Threshold Value: 84.0

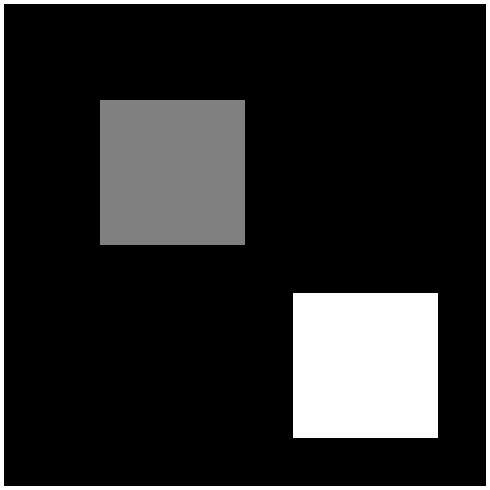

Figure 1: Original Image

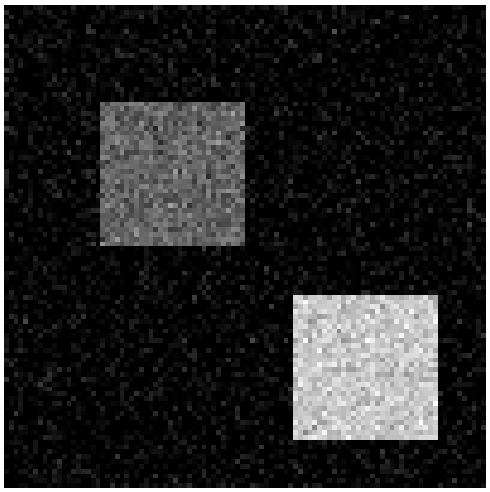

Figure 2: Noisy Image

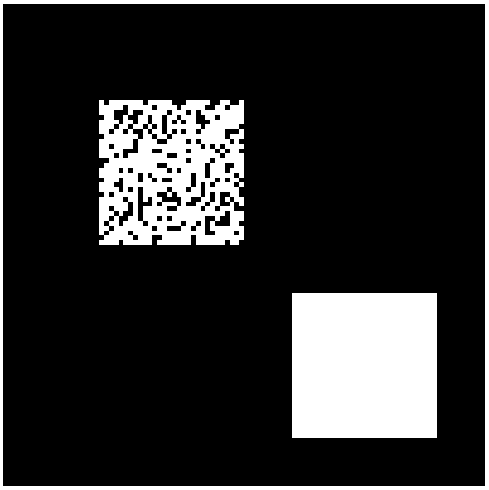

Figure 3: Otsu Thresholded Image

Task 2: Region Growing Segmentation

Seed Points: (30, 30) and (70, 70)

Threshold: 30 (pixel value difference)

4-connected neighbors were evaluated and included based on intensity difference from the seed values.

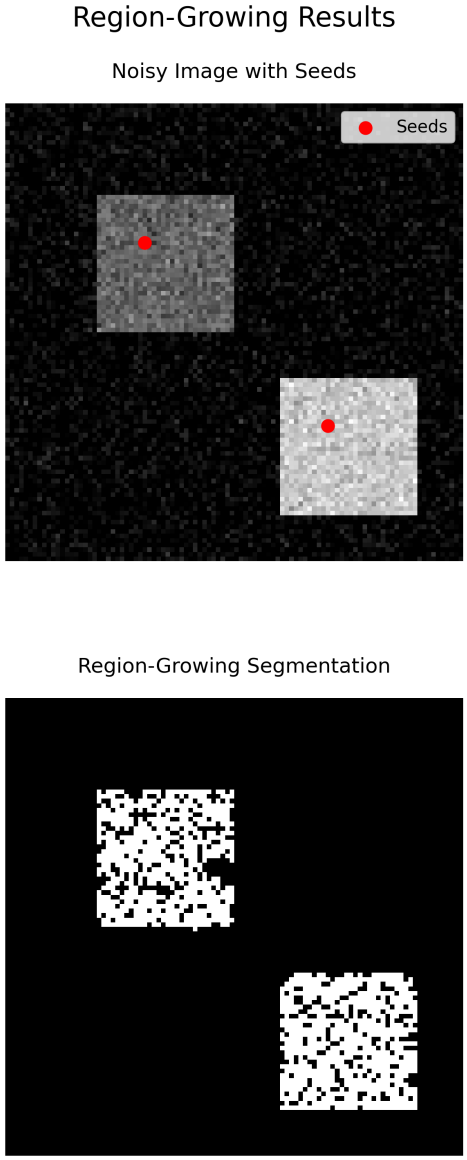

Figure 4: Region Growing Segmentation Results

Source Code

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.filters import threshold_otsu
from collections import deque
import os

def create_synthetic_image(shape=(100, 100)):
    img = np.zeros(shape, dtype=np.uint8)  # Background = 0
    img[20:50, 20:50] = 100  # Object 1 with pixel value 100
    img[60:90, 60:90] = 200  # Object 2 with pixel value 200
    return img

def add_gaussian_noise(image, mean=0, sigma=20):
    noise = np.random.normal(mean, sigma, image.shape)
    noisy_img = image + noise
    noisy_img = np.clip(noisy_img, 0, 255)
    return noisy_img.astype(np.uint8)

def otsu_threshold(image):
    thresh = threshold_otsu(image)
    binary = image > thresh
    return thresh, binary

def region_growing(image, seeds, threshold):
    h, w = image.shape
    segmented = np.zeros_like(image, dtype=bool)
    visited = np.zeros_like(image, dtype=bool)

    queue = deque(seeds)
    for seed in seeds:
        segmented[seed] = True
        visited[seed] = True

    while queue:
        x, y = queue.popleft()
        seed_val = image[x, y]

        neighbors = [(x-1, y), (x+1, y), (x, y-1), (x, y+1)]
        for nx, ny in neighbors:  # Fixed: No 'materialen'
            if 0 <= nx < h and 0 <= ny < w and not visited[nx, ny]:
                neighbor_val = image[nx, ny]
                if abs(int(neighbor_val) - int(seed_val)) <= threshold:
                    segmented[nx, ny] = True
                    queue.append((nx, ny))
                visited[nx, ny] = True

    return segmented

def save_image(img, filename, cmap='gray'):
    plt.imsave(filename, img, cmap=cmap)

if __name__ == "__main__":
    # Create output directory
    output_dir = "output"
    os.makedirs(output_dir, exist_ok=True)
```

```
# Create synthetic image and add noise
img = create_synthetic_image()
noisy_img = add_gaussian_noise(img)

# Task 1: Otsu's thresholding
thresh, segmented_otsu = otsu_threshold(noisy_img)
print(f"Otsu's threshold: {thresh}")

# Task 2: Region-growing
seeds = [(30, 30), (70, 70)]
threshold = 30
segmented_region = region_growing(noisy_img, seeds, threshold)

# Save individual images
save_image(img, os.path.join(output_dir, "original_synthetic.png"))
save_image(noisy_img, os.path.join(output_dir, "noisy_image.png"))
save_image(segmented_otsu, os.path.join(output_dir, "otsu_segmented
    .png"))
save_image(segmented_region, os.path.join(output_dir, "
    region_growing_segmented.png"))

# Plot Otsu results vertically
plt.figure(figsize=(6, 15))
plt.suptitle("Otsu's Algorithm Results", fontsize=16, y=0.98)

plt.subplot(3, 1, 1)
plt.title("Original Image", pad=15)
plt.imshow(img, cmap='gray')
plt.axis('off')

plt.subplot(3, 1, 2)
plt.title("Noisy Image", pad=15)
plt.imshow(noisy_img, cmap='gray')
plt.axis('off')

plt.subplot(3, 1, 3)
plt.title(f"Otsu Thresholded (T={thresh:.2f})", pad=15)
plt.imshow(segmented_otsu, cmap='gray')
plt.axis('off')

plt.tight_layout(pad=3.0)
plt.subplots_adjust(top=0.90, hspace=0.3)  # Increased top margin
plt.savefig(os.path.join(output_dir, "otsu_results_vertical.png"),
    bbox_inches='tight', dpi=300)
plt.show()

# Plot region-growing results vertically
plt.figure(figsize=(6, 10))
plt.suptitle("Region-Growing Results", fontsize=16, y=0.98)  #
    Adjusted y to avoid overlap

plt.subplot(2, 1, 1)
plt.title("Noisy Image with Seeds", pad=15)
plt.imshow(noisy_img, cmap='gray')
plt.scatter([y for x, y in seeds], [x for x, y in seeds], c='red',
    s=50, label='Seeds')
plt.legend()
```

```
    plt.axis('off')

    plt.subplot(2, 1, 2)
    plt.title("Region-Growing Segmentation", pad=15)
    plt.imshow(segmented_region, cmap='gray')
    plt.axis('off')

    plt.tight_layout(pad=3.0)
    plt.subplots_adjust(top=0.90, hspace=0.3)  # Increased top margin
    plt.savefig(os.path.join(output_dir, "
        region_growing_results_vertical.png"), bbox_inches='tight', dpi
        =300)
    plt.show()

    print(f"Images and plots saved in '{output_dir}' directory.")
```

GitHub Repository

The full code and outputs are available at: https://github.com/Sivasothy-Tharsi/CV_Assignment_02