EE7204: COMPUTER VISION AND IMAGE PROCESSING

TAKE HOME ASSIGNMENT 2

NAME : ABEYRATHNE H.G.D.C.N.

REG No. : EG/ 2019/ 3520

SEMESTER : 07

DATE : 21 /03 /2024

1. Consider an image with 2 objects and a total of 3-pixel values (1 for each object and one for the background). Add Gaussian noise to the image. Implement and test Otsu's algorithm with this image.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import filters
from skimage.util import random_noise
from skimage.io import imread

# Add Gaussian noise to the image
def add_gaussian_noise(image, mean=0, std=0.1):
    noisy_image = random_noise(image, mode='gaussian', mean=mean,
var=std**2)
    return noisy_image

# Implement Otsu's algorithm
def apply_otsu_threshold(image):
    threshold_value = filters.threshold_otsu(image)
    binary_image = image > threshold_value
    return binary_image

# Main function
def main():
    # Load the image
    image_path = 'test.jpg'
    original_image = imread(image_path)
    original_image_gray = imread(image_path, as_gray=True)

    # Add Gaussian noise to the image
    noisy_image = add_gaussian_noise(original_image_gray)

    # Apply Otsu's algorithm
    segmented_image = apply_otsu_threshold(noisy_image)

    # Display original image, noisy image, and segmented image
    fig, axes = plt.subplots(1, 3, figsize=(12, 4))
    ax = axes.ravel()
    ax[0].imshow(original_image)
    ax[0].set_title('Original Image')
    ax[1].imshow(noisy_image, cmap='gray')
    ax[1].set_title('Noisy Image')
    ax[2].imshow(segmented_image, cmap='gray')
    ax[2].set_title('Segmented Image (Otsu)')
    for a in ax:
        a.axis('off')
```

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

if __name__ == "__main__":
    main()
```

Result:

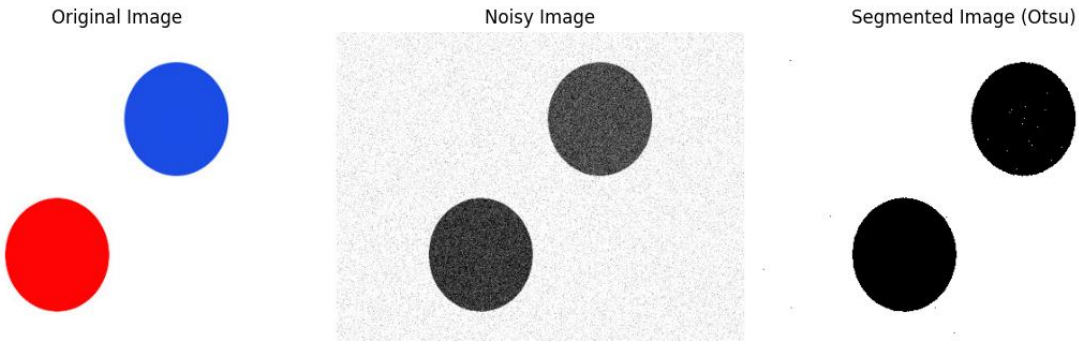

Figure 1: Implementing and test Otsu's algorithm with this image

2. Implement a region-growing technique for image segmentation. The basic idea is to start from a set of points inside the object of interest (foreground), denoted as seeds, and recursively add neighboring pixels as long as they are in a pre-defined range of the pixel values of the seeds.

Code:

```
import numpy as np
import cv2

def region_growing(image, seed, threshold):
    # Create a mask that will contain the segmented region
    segmented = np.zeros_like(image)

    # Mark the seed point as visited
    visited = np.zeros_like(image, dtype=bool)

    # Define the connectivity (8-connectivity)
    connectivity = [(x, y) for x in range(-1, 2) for y in range(-1, 2) if
not (x == 0 and y == 0)]

    # Define a function to check if a pixel is within the image bounds
    def is_valid_pixel(pixel):
        return 0 <= pixel[0] < image.shape[0] and 0 <= pixel[1] <
image.shape[1]

    # Define a function to check if a pixel is in the threshold range
    def is_in_threshold(pixel, seed_value):
        return abs(image[pixel[0], pixel[1]] - seed_value) <= threshold

    # Start growing the region from the seed point
    stack = [seed]
    seed_value = image[seed[0], seed[1]]
    while stack:
        current_pixel = stack.pop()
        segmented[current_pixel[0], current_pixel[1]] = 255
        visited[current_pixel[0], current_pixel[1]] = True

        # Check neighboring pixels
        for dx, dy in connectivity:
            neighbor = (current_pixel[0] + dx, current_pixel[1] + dy)
            if is_valid_pixel(neighbor) and not visited[neighbor[0],
neighbor[1]]:
                if is_in_threshold(neighbor, seed_value):
                    stack.append(neighbor)

    return segmented

# Load the image
image = cv2.imread('test2.png', cv2.IMREAD_GRAYSCALE)
```

```
seed_point = (100, 100)

threshold_value = 20

# Perform region growing segmentation
segmented_image = region_growing(image, seed_point, threshold_value)

# Display the original image and segmented image
cv2.imshow('Original Image', image)
cv2.imshow('Segmented Image', segmented_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result:

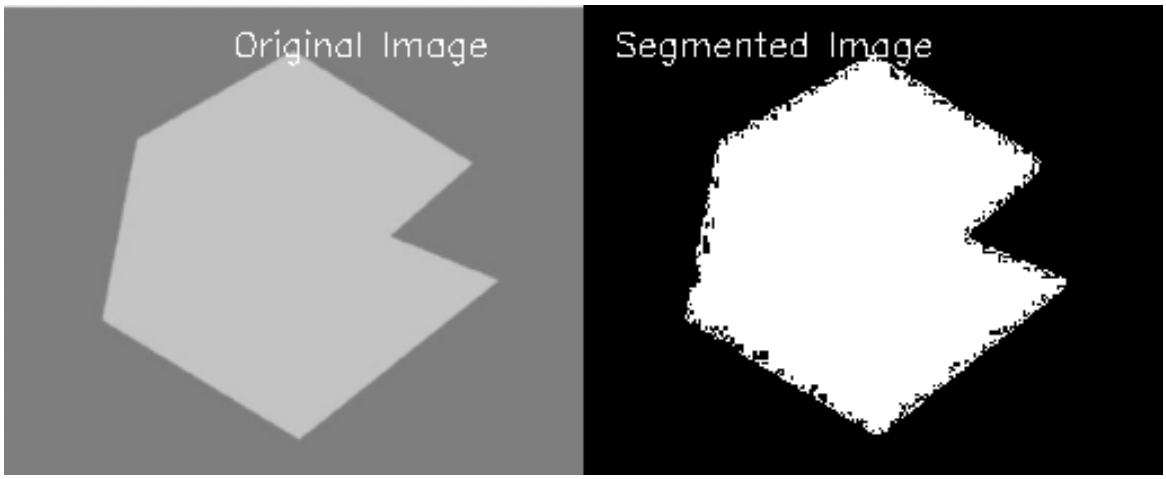

Figure 2: Implement a region-growing technique for image segmentation

GitHub Repository Link:

https://github.com/chinthaka99/EE-7204-Computer-Vision-and-Image-Processing-Assignment-2