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:**

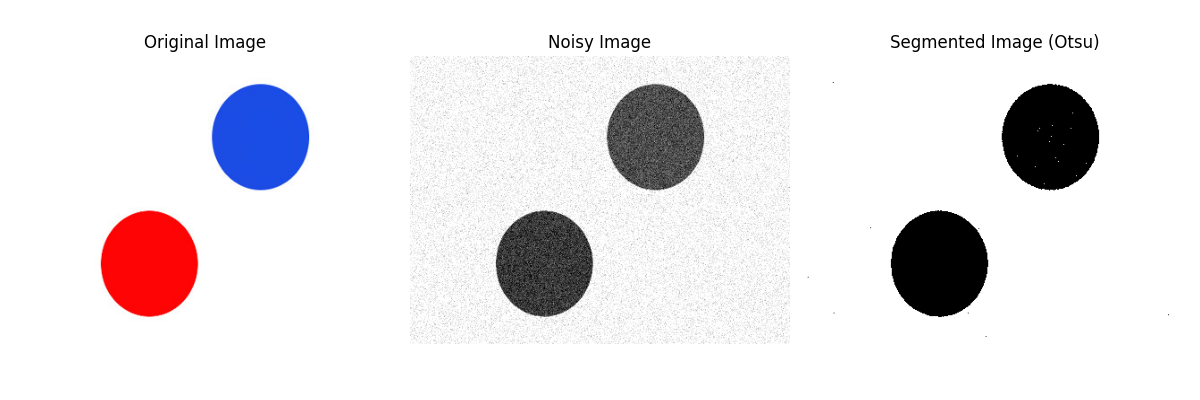


Figure 1: Implementing and test Otsu’s algorithm with this image

1. 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:**

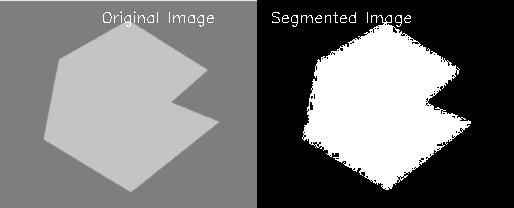


Figure 2: Implement a region-growing technique for image segmentation

**GitHub Repository Link:**

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