EE7204: COMPUTER VISION AND IMAGE PROCESSING

TAKE HOME ASSIGNMENT 02

NAME	:	SUDARA N.H.M
REG. NO.	:	EG/2018/3471
SEMESTER	:	07
DATE	:	17/03/2023

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.

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.util import random_noise
# Create the image
img = np.zeros((100, 100))
img[20:40, 20:40] = 120
img[60:80, 60:80] = 200

# Add Gaussian noise
mean = 0
variance = 100
sigma = np.sqrt(variance)
noise = np.random.normal(mean, sigma, img.shape)
noisy_img = img + noise

# Otsu's algorithm implementation
hist, bins = np.histogram(noisy_img, bins=3)
bins = bins[:-1]
pixel_num = np.sum(hist)
mean = np.sum(bins * hist) / pixel_num
max_var = 0
thresh = 0
for i in range(1, 3):
    w0 = np.sum(hist[:i]) / pixel_num
    w1 = np.sum(hist[i:]) / pixel_num
    mu0 = np.sum(bins[:i] * hist[:i]) / (w0 * pixel_num)
    mu1 = np.sum(bins[i:] * hist[i:]) / (w1 * pixel_num)
    var = w0 * w1 * ((mu0 - mu1) ** 2)
    if var > max_var:
        max_var = var
        thresh = i
# Segment the image
segmented_img = np.zeros_like(noisy_img)
segmented_img[noisy_img >= thresh] = 1


# Plot the images
fig, ax = plt.subplots(1, 3, figsize=(10, 4))
ax[0].imshow(img, cmap='gray')
ax[0].set_title('Original Image')
ax[1].imshow(noisy_img, cmap='gray')
ax[1].set_title('Noisy Image')
ax[2].imshow(segmented_img, cmap='gray')
ax[2].set_title('Segmented Image')
plt.show()
```

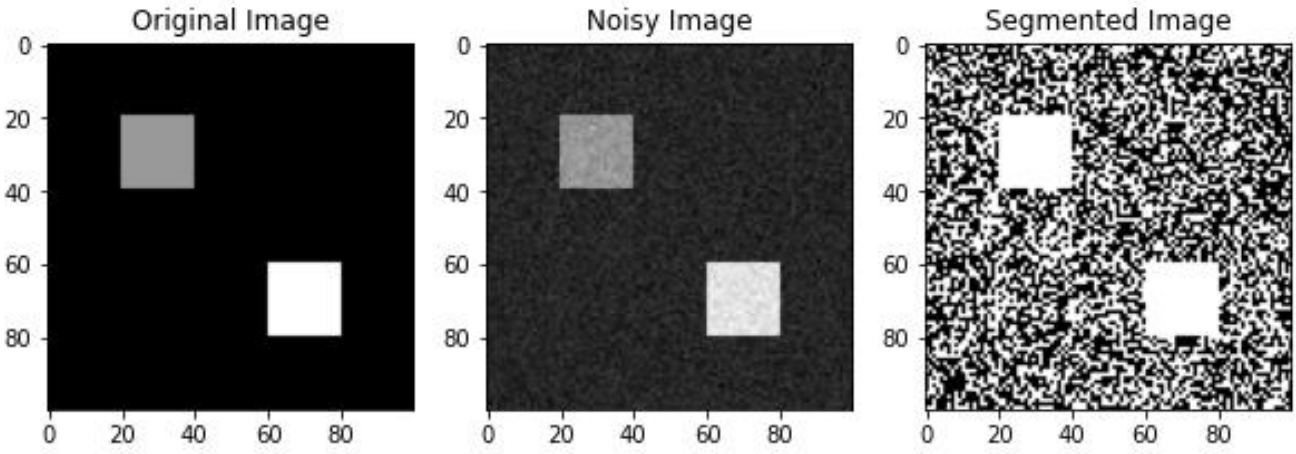

FIGURE 1: RESULT OF SEGMENTATION USING OTSU'S ALGORITHM

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.

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
def region_growing(img, seeds, thresh):
    # Initialize segmented image
    seg_img = np.zeros_like(img)

    # Initialize seed queue
    seed_queue = []
    for seed in seeds:
        seed_queue.append(seed)

    # Loop until seed queue is empty
    while len(seed_queue) > 0:
        # Get next seed point from queue
        curr_seed = seed_queue.pop(0)

        # Check if current seed point is already segmented
        if seg_img[curr_seed] > 0:
            continue

        # Check if current seed point is within threshold
range
        if img[curr_seed] < thresh[0] or img[curr_seed] >
thresh[1]:
            continue

        # Set current seed point as segmented
        seg_img[curr_seed] = 255

        # Add neighboring pixels to seed queue
        x, y = curr_seed
        for i in range(x-1, x+2):
            for j in range(y-1, y+2):
                # Check if neighbor is within image bounds
                if i < 0 or i >= img.shape[0] or j < 0 or j
>= img.shape[1]:
                    continue
                # Check if neighbor is already segmented or
in seed queue
                if seg_img[i, j] > 0 or (i, j) in seed_queue:
                    continue
                # Check if neighbor is within threshold range
```

```
                if img[i, j] < thresh[0] or img[i, j] >
thresh[1]:
                    continue
                # Add neighbor to seed queue
                seed_queue.append((i, j))

    return seg_img

# Create the image
img = np.zeros((100, 100))
img[20:40, 20:40] = 120
img[60:80, 60:80] = 200

# Add Gaussian noise
mean = 0
variance = 100
sigma = np.sqrt(variance)
gaussian = np.random.normal(mean, sigma, img.shape)
noisy_img = img + gaussian

# Define seed points
seeds = [(30, 30),(70, 70)]

# Define threshold range for pixel values
thresh = (100, 220)

# Apply region growing algorithm for image segmentation
seg_img = region_growing(noisy_img, seeds, thresh)


# Plot the images and histogram
fig, ax = plt.subplots(1, 3, figsize=(10, 4))
ax[0].imshow(img, cmap='gray')
ax[0].set_title('Original Image')
ax[1].imshow(noisy_img, cmap='gray')
ax[1].set_title('Noisy Image')
ax[2].imshow(seg_img, cmap='gray')
ax[2].set_title('Segmented Image')
plt.show()
```

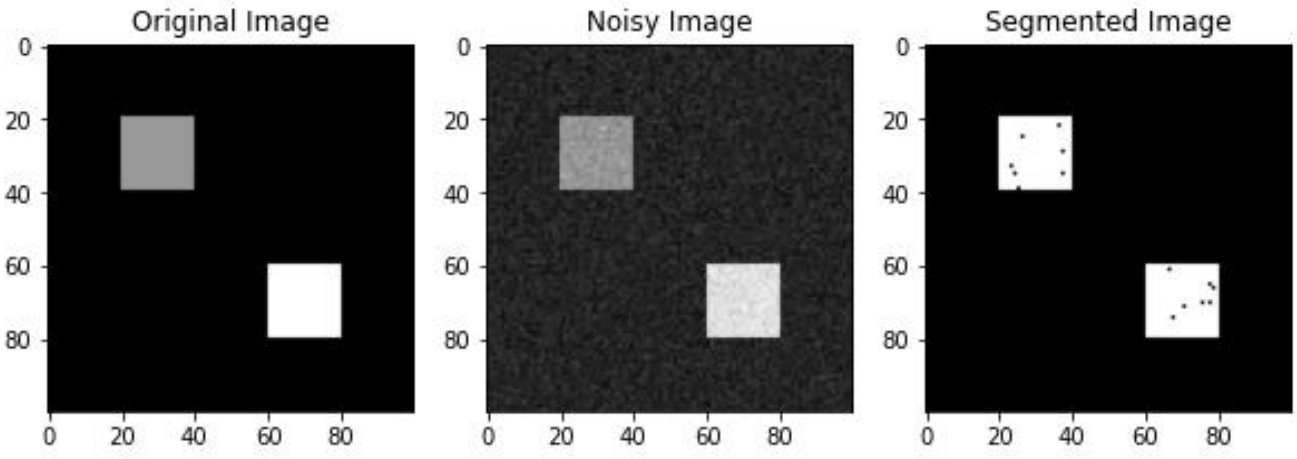

FIGURE 2:RESULT OF SEGMENTATION USING REGION-GROWING TECHNIQUE

- githublink:https://github.com/NHMSudara/Image_processing_Take_Home_Assignment_2.git