EE7204: COMPUTER VISION AND IMAGE PROCESSING

TAKE HOME ASSIGNMENT 02

NAME : TENNAKOON T.M.D.C.B.

REG No. : EG / 2020 / 4226

SEMESTER: 07

DATE : 27 / 06 / 2025

# Contents

[Contents i](#_Toc201778029)

[1 Task 1 1](#_Toc201778030)

[1.1 Python Code for Task 1 1](#_Toc201778031)

[1.2 Output of Task 1 2](#_Toc201778032)

[2 Task 2 2](#_Toc201778033)

[2.1 Python Code for Task 2 2](#_Toc201778034)

[2.2 Output of Task 2 4](#_Toc201778035)

[GitHub Link 5](#_Toc201778036)

# Task 1

## Python Code for Task 1

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Creating a synthetic image

my\_img = np.zeros((100, 100), dtype=np.uint8)

my\_img[20:50, 20:50] = 100

my\_img[60:90, 60:90] = 200

# Adding gaussian noise

mean = 0

standard\_dev = 20

gaussian\_noise = np.random.normal(mean, standard\_dev, my\_img.shape).astype(np.uint8)

noisy\_image = my\_img.astype(np.int16) + gaussian\_noise.astype(np.int16)

noisy\_image = np.clip(noisy\_image, 0, 255).astype(np.uint8)

# Otsu's Thresholding

\_, otsus\_threshold = cv2.threshold(noisy\_image, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)

plt.title('Original Image')

plt.imshow(my\_img, cmap='gray')

plt.axis('off')

plt.subplot(1, 3, 2)

plt.title('Noisy Image')

plt.imshow(noisy\_image, cmap='gray')

plt.axis('off')

plt.subplot(1, 3, 3)

plt.title("Otsu's Thresholded")

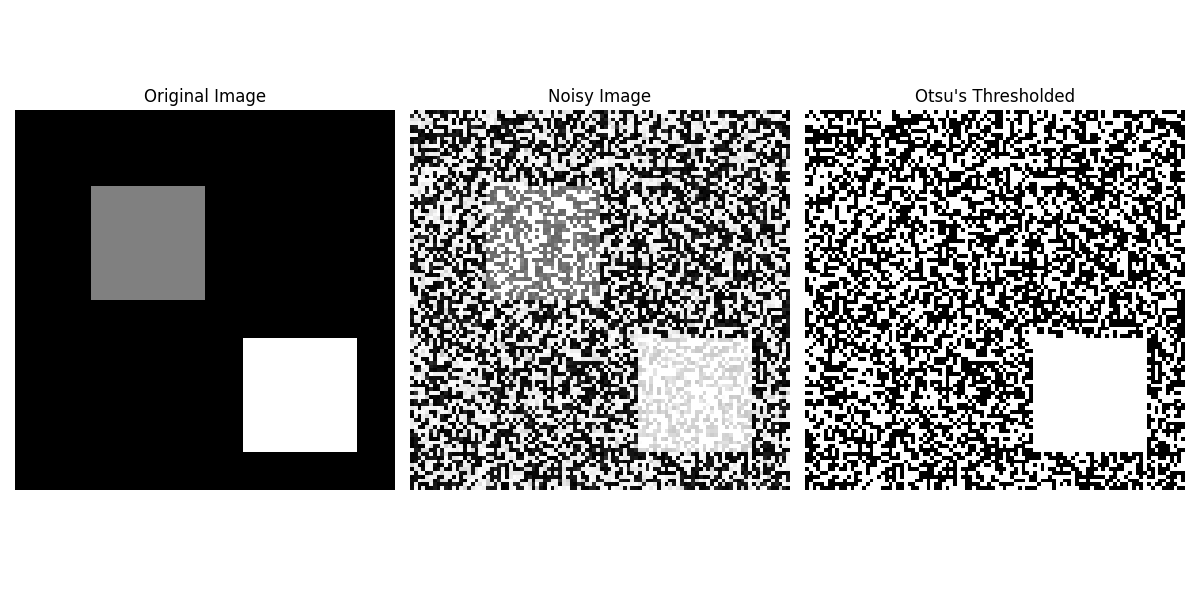
plt.imshow(otsus\_threshold, cmap='gray')

plt.axis('off')

plt.tight\_layout()

plt.show()

## Output of Task 1



# Task 2

## Python Code for Task 2

import cv2

import numpy as np

import matplotlib.pyplot as plt

def seed\_growing(img, seeds, threshold):

    height, width = img.shape

    segmented = np.zeros\_like(img)

    visited = np.zeros\_like(img, dtype=bool)

    for seed in seeds:

        if visited[seed]:

            continue

        seed\_val = int(img[seed])

        region\_mean = seed\_val

        region\_size = 1

        stack = [seed]

        segmented[seed] = 255

        visited[seed] = True

        while stack:

            y, x = stack.pop()

            for dy in [-1, 0, 1]:

                for dx in [-1, 0, 1]:

                    ny, nx = y + dy, x + dx

                    if 0 <= ny < height and 0 <= nx < width and not visited[ny, nx]:

                        intensity = int(img[ny, nx])

                        if abs(intensity - region\_mean) < threshold:

                            segmented[ny, nx] = 255

                            visited[ny, nx] = True

                            stack.append((ny, nx))

                            region\_size += 1

                            region\_mean = (region\_mean \* (region\_size - 1) + intensity) / region\_size

return segmented

image\_path = 'subject.jpg'

img\_color = cv2.imread(image\_path)

gray\_img = cv2.cvtColor(img\_color, cv2.COLOR\_BGR2GRAY)

# Multiple seed points

seed\_points = [(1450, 1400), (1600, 1500), (1800, 1350)]

# Threshold for similarity

threshold = 30

segmented = seed\_growing(gray\_img, seed\_points, threshold)

plt.figure(figsize=(15, 6))

plt.subplot(1, 3, 1)

plt.title("Original Image")

plt.imshow(cv2.cvtColor(img\_color, cv2.COLOR\_BGR2RGB))

plt.axis('off')

plt.subplot(1, 3, 2)

plt.title("Grayscale Image with Seeds")

plt.imshow(gray\_img, cmap='gray')

for sp in seed\_points:

    plt.plot(sp[1], sp[0], 'ro')

plt.axis('off')

plt.subplot(1, 3, 3)

plt.title("Region Grown Segment")

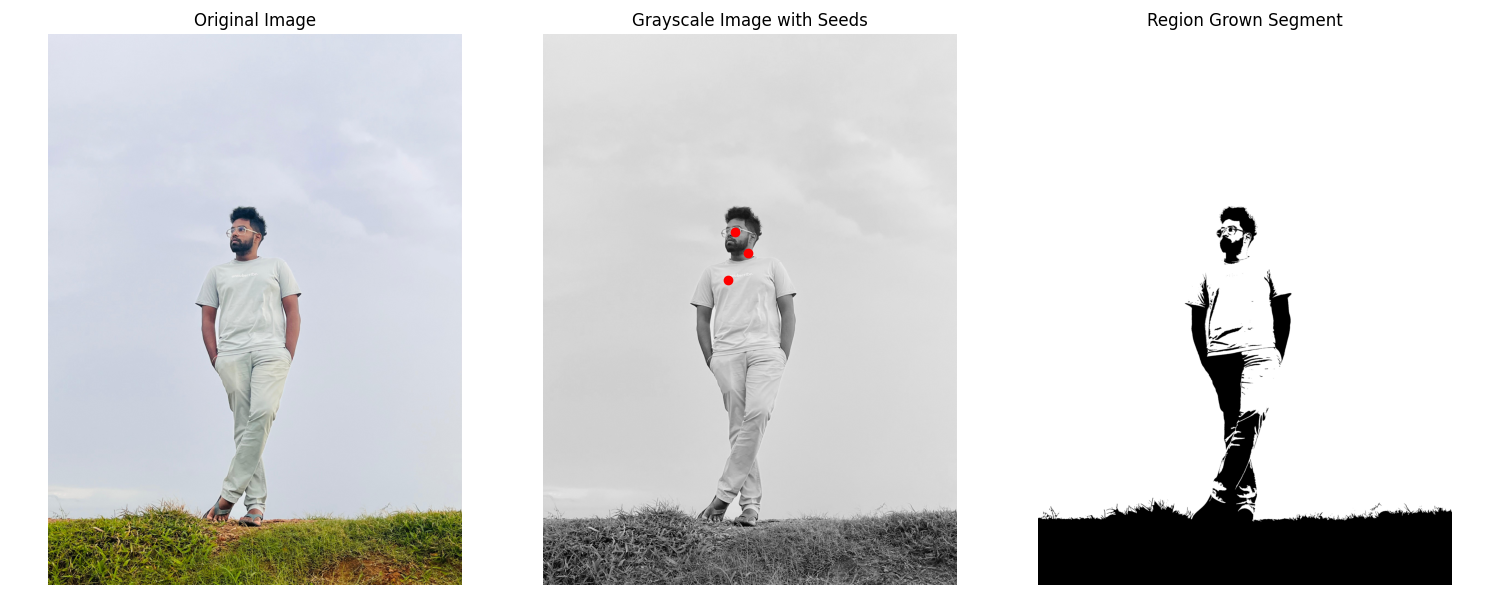
plt.imshow(segmented, cmap='gray')

plt.axis('off')

plt.tight\_layout()

plt.show()

## Output of Task 2



# GitHub Link