EC7212: COMPUTER VISION AND IMAGE PROCESSING

TAKE HOME ASSIGNMENT 2

NAME	: ABERUWAN R. M. M. P.
REG. NO.	: EG/2020/3797
SEMESTER	: 07
DATE	: 27/06/2025

Contents

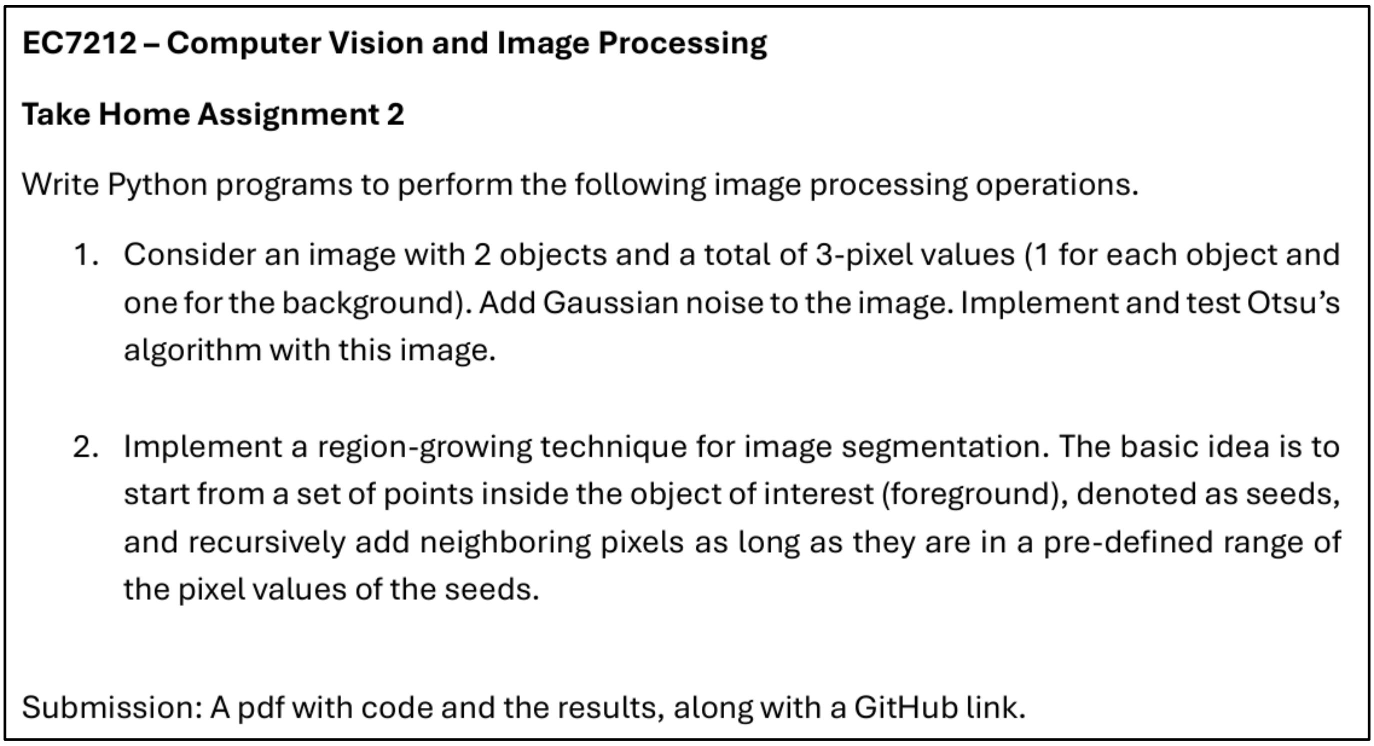
List of Figures

1 Introduction

Computer vision and image processing are important fields that help computers work with images and understand what they show. Image processing is about changing or improving images, like making them clearer or removing noise. Computer vision goes a step further by teaching computers to recognize and understand the objects in an image, just like humans do. In this assignment, we focus on two main image processing tasks using Python:

1. First, we add Gaussian noise to a simple image with two objects and a background. Gaussian noise is a type of random variation in pixel values that often happens during image capture or transmission. After adding the noise, we use Otsu's algorithm to automatically find the best threshold to separate the objects from the background. Otsu's method does this by analyzing the image's pixel values and finding a threshold that best splits the image into two groups: foreground and background.
2. Second, we implement a region-growing technique for image segmentation. This method starts from some seed points inside the object of interest and gradually adds neighboring pixels that are similar to the seeds. The process continues until the whole object is segmented. Region growing is useful because it can find objects with complex shapes and works well even when the image has some noise.

By completing these tasks, we learn how to use basic image processing and computer vision techniques to segment and analyze images.



EC7212 – Computer Vision and Image Processing

Take Home Assignment 2

Write Python programs to perform the following image processing operations.

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.

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.

Submission: A pdf with code and the results, along with a GitHub link.

Figure 1.1: Take Home Assignment 2

2 Task 1: Add Gaussian Noise and Apply Otsu's Algorithm

2.1 Objective

The goal of this task is to:

- Take a real image of fruits (apple and orange)
- Add artificial noise to make it look grainy
- Use Otsu's automatic method to separate the fruits from the background

- Show how well this method works on noisy images

2.2 Methodology

We followed these steps:

1. Downloaded an image of an apple and orange from the internet
2. Added Gaussian noise to make the image grainy (using random pixel variations)
3. Applied Otsu's algorithm which automatically finds the best threshold to:
    - Separate bright areas (fruits) from dark areas (background)
    - Convert the image to black-and-white (binary)
4. Displayed three images side-by-side for comparison:
    - Original clean image
    - Noisy version
    - Final thresholded result

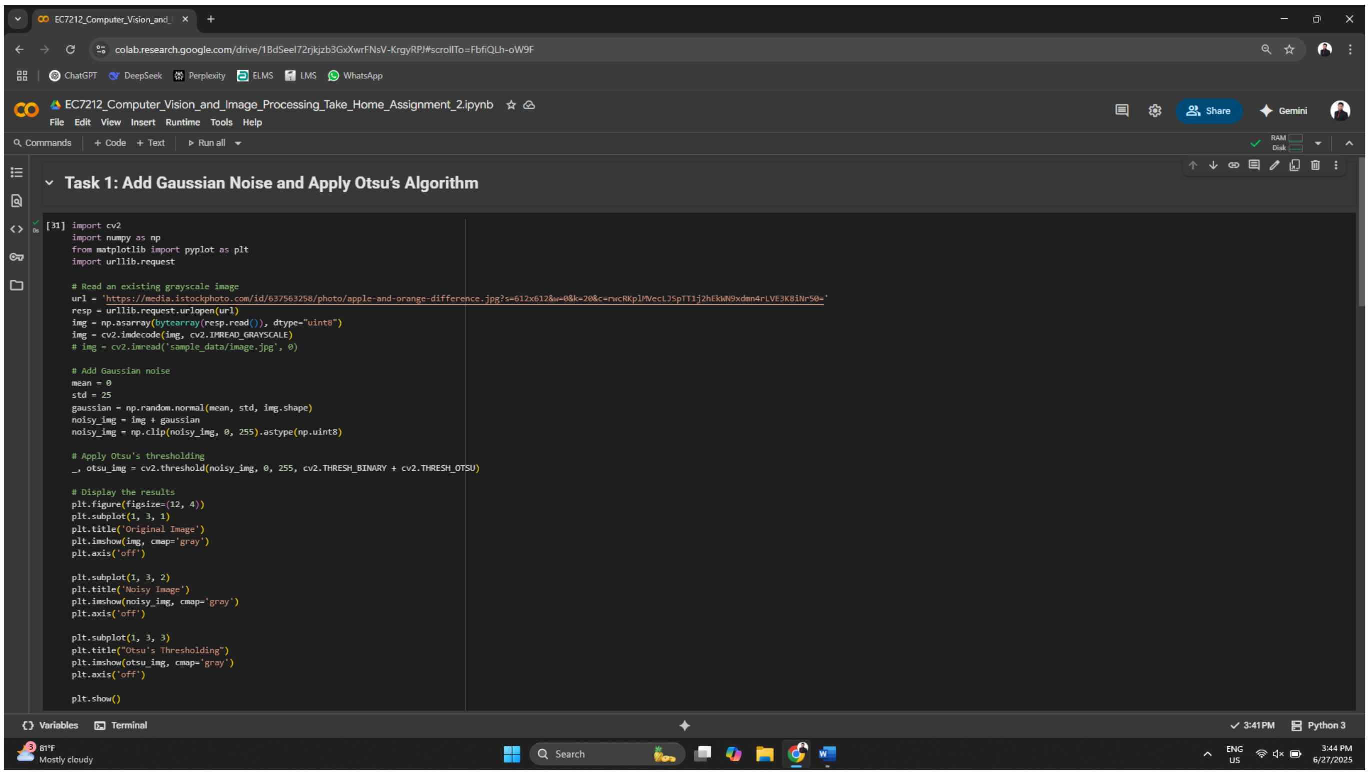
2.3 Results

The output shows:

1. Original image: Clear picture of fruits with dark background
2. Noisy image: Grainy version where details are harder to see
3. Otsu's result: Clean separation where:
    - Fruits appear white
    - Background appears black
    - Edges of fruits remain clear despite the noise

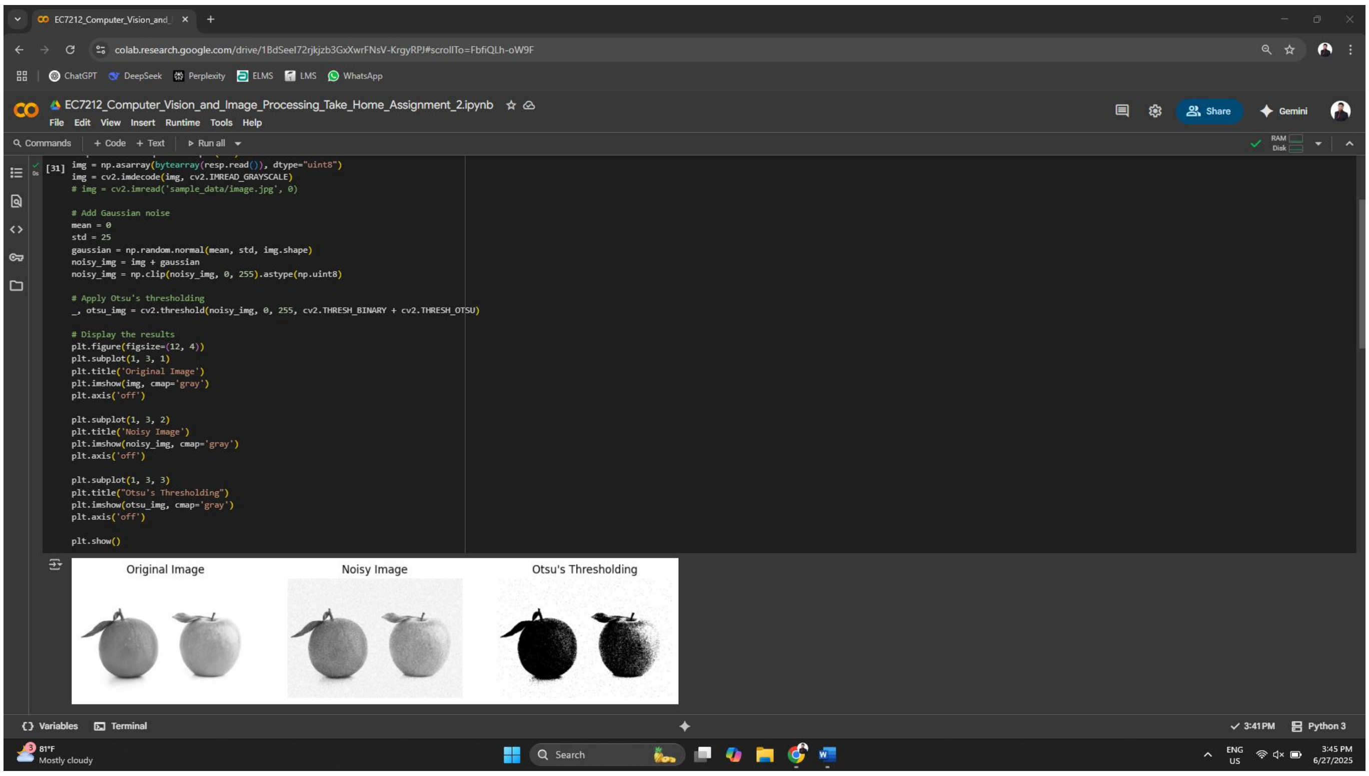
This demonstrates that Otsu's method can effectively separate objects from background even when the image has significant noise. The automatic threshold selection worked well for this fruit image with distinct objects and background.

2.4 Code and Output


EC7212_Computer_Vision_and_Image_Processing_Take_Home_Assignment_2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
Variables
Terminal

Task 1: Add Gaussian Noise and Apply Otsu's Algorithm

```
[31] import cv2
     import numpy as np
     from matplotlib import pyplot as plt
     import urllib.request

     # Read an existing grayscale image
     url = 'https://media.istockphoto.com/id/637563258/photo/apple-and-orange-difference.jpg?s=612x612&w=0&k=20&c=rwcRKplMVecLJSpTT1j2hEkWN9xdmn4rLVE3K8iNr50='
     resp = urllib.request.urlopen(url)
     img = np.asarray(bytearray(resp.read()), dtype="uint8")
     img = cv2.imdecode(img, cv2.IMREAD_GRAYSCALE)
     # img = cv2.imread('sample_data/image.jpg', 0)

     # Add Gaussian noise
     mean = 0
     std = 25
     gaussian = np.random.normal(mean, std, img.shape)
     noisy_img = img + gaussian
     noisy_img = np.clip(noisy_img, 0, 255).astype(np.uint8)

     # Apply Otsu's thresholding
     _, otsu_img = cv2.threshold(noisy_img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

     # Display the results
     plt.figure(figsize=(12, 4))
     plt.subplot(1, 3, 1)
     plt.title('Original Image')
     plt.imshow(img, cmap='gray')
     plt.axis('off')

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

     plt.subplot(1, 3, 3)
     plt.title("Otsu's Thresholding")
     plt.imshow(otsu_img, cmap='gray')
     plt.axis('off')

     plt.show()
```


EC7212_Computer_Vision_and_Image_Processing_Take_Home_Assignment_2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Original Image
Noisy Image
Otsu's Thresholding
Variables Terminal
Python 3

```
[31] img = np.asarray(bytearray(resp.read()), dtype="uint8")
     img = cv2.imdecode(img, cv2.IMREAD_GRAYSCALE)
     # img = cv2.imread('sample_data/image.jpg', 0)

     # Add Gaussian noise
     mean = 0
     std = 25
     gaussian = np.random.normal(mean, std, img.shape)
     noisy_img = img + gaussian
     noisy_img = np.clip(noisy_img, 0, 255).astype(np.uint8)

     # Apply Otsu's thresholding
     _, otsu_img = cv2.threshold(noisy_img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

     # Display the results
     plt.figure(figsize=(12, 4))
     plt.subplot(1, 3, 1)
     plt.title('Original Image')
     plt.imshow(img, cmap='gray')
     plt.axis('off')

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

     plt.subplot(1, 3, 3)
     plt.title("Otsu's Thresholding")
     plt.imshow(otsu_img, cmap='gray')
     plt.axis('off')

     plt.show()
```

Figure 2.1: Implementation and Results for Task 1

3 Task 2: Region Growing for Image Segmentation

3.1 Objective

The goal of this task is to:

- Use region growing to separate the apple and orange from the background
- Start from seed points inside each fruit
- Grow regions based on pixel similarity
- Create a segmented image showing just the fruits

3.2 Methodology

We followed these steps:

1. Used the same fruit image as in Task 1 for consistency
2. Selected seed points inside each fruit:
   - (100, 100) for the apple
   - (150, 150) for the orange
3. Set a similarity threshold of 20 (pixels within ±20 intensity of seed is included)
4. Implemented region growing that:
   - Starts from seed points
   - Checks all 8 neighboring pixels
   - Adds similar pixels to the region
   - Continues until no more similar pixels are found

3.3 Results

The output shows:

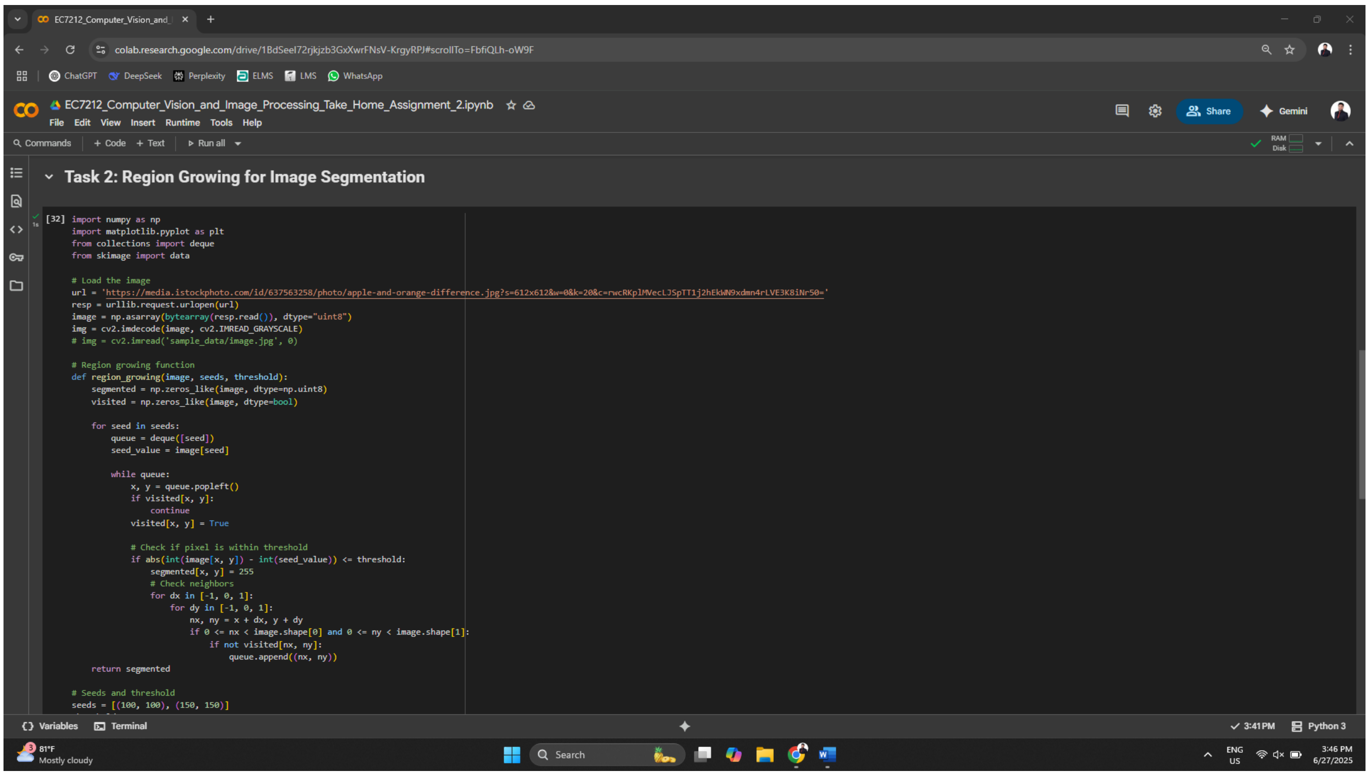
1. Original image: The starting picture with both fruits
2. Segmented result:
   - White areas show detected fruit regions

- Black areas represent the background
- Both fruits are clearly separated from background
- The segmentation follows the shape of each fruit

This demonstrates how region growing can effectively segment objects when:

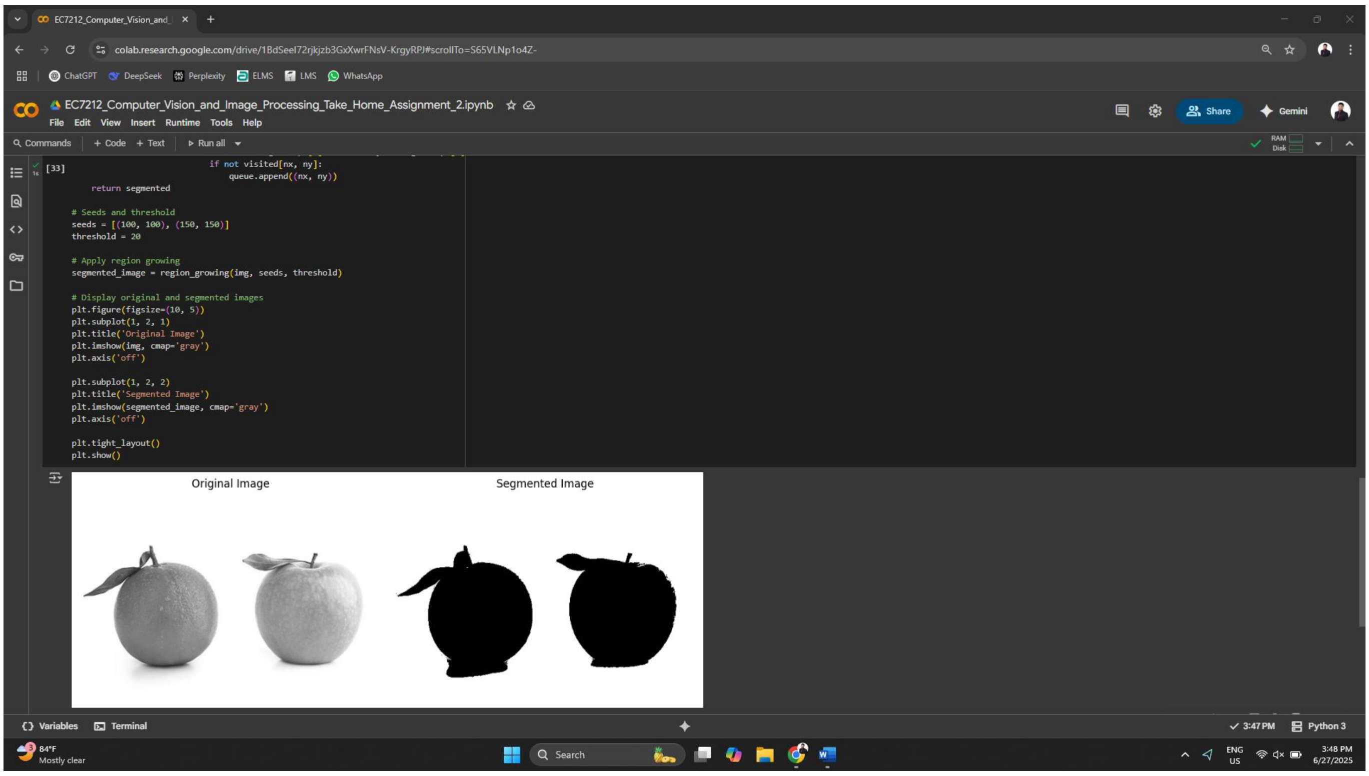
- Good seed points are chosen inside the objects
- A proper similarity threshold is selected
- Objects have distinct pixel values from the background

3.4 Code and Output



EC7212_Computer_Vision_and_Image_Processing_Take_Home_Assignment_2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Task 2: Region Growing for Image Segmentation
Variables Terminal

```
[32] import numpy as np
     import matplotlib.pyplot as plt
     from collections import deque
     from skimage import data

     # Load the image
     url = 'https://media.istockphoto.com/id/637563258/photo/apple-and-orange-difference.jpg?s=612x612&w=0&k=20&c=rwcRKplMVecLJSpTT1j2hEkWN9xdmn4rLVE3K8iNr50='
     resp = urllib.request.urlopen(url)
     image = np.asarray(bytearray(resp.read()), dtype="uint8")
     img = cv2.imdecode(image, cv2.IMREAD_GRAYSCALE)
     # img = cv2.imread('sample_data/image.jpg', 0)

     # Region growing function
     def region_growing(image, seeds, threshold):
         segmented = np.zeros_like(image, dtype=np.uint8)
         visited = np.zeros_like(image, dtype=bool)

         for seed in seeds:
             queue = deque([seed])
             seed_value = image[seed]

             while queue:
                 x, y = queue.popleft()
                 if visited[x, y]:
                     continue
                 visited[x, y] = True

                 # Check if pixel is within threshold
                 if abs(int(image[x, y]) - int(seed_value)) <= threshold:
                     segmented[x, y] = 255
                     # Check neighbors
                     for dx in [-1, 0, 1]:
                         for dy in [-1, 0, 1]:
                             nx, ny = x + dx, y + dy
                             if 0 <= nx < image.shape[0] and 0 <= ny < image.shape[1]:
                                 if not visited[nx, ny]:
                                     queue.append((nx, ny))

         return segmented

     # Seeds and threshold
     seeds = [(100, 100), (150, 150)]
```



EC7212_Computer_Vision_and_Image_Processing_Take_Home_Assignment_2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Original Image
Segmented Image
Variables Terminal

```
[33]                             if not visited[nx, ny]:
                                     queue.append((nx, ny))

         return segmented

     # Seeds and threshold
     seeds = [(100, 100), (150, 150)]
     threshold = 20

     # Apply region growing
     segmented_image = region_growing(img, seeds, threshold)

     # Display original and segmented images
     plt.figure(figsize=(10, 5))
     plt.subplot(1, 2, 1)
     plt.title('Original Image')
     plt.imshow(img, cmap='gray')
     plt.axis('off')

     plt.subplot(1, 2, 2)
     plt.title('Segmented Image')
     plt.imshow(segmented_image, cmap='gray')
     plt.axis('off')

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

Figure 3.1: Implementation and Results for Task 2

4 Conclusion

In this assignment, we learned how to use two important image processing techniques: Otsu’s thresholding and region growing. By working with a real image of fruits, we saw how adding noise can make image analysis harder, but Otsu’s method can still separate objects from the background automatically. We also used region growing to segment the apple and orange by starting from points inside the fruits and expanding to similar pixels. This showed us that region growing works well when we choose good starting points and set the right threshold.

Overall, these tasks helped us understand how computers can find and separate objects in images, even when the images are noisy or complex. Both methods are useful for many real-world applications in computer vision and image processing.

5 Submission

GitHub Link:

https://github.com/MalithPramoditha/EC7212_Computer_Vision_and_Image_Processing_Take_Home_Assignment_2

Google Colab Link:

https://colab.research.google.com/drive/1BdSeeI72rjkjzb3GxXwrFNsV-KrgyRPJ?usp=sharing