Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)

AIM:

THEORY:

1. Region Growing

The objective of segmentation is to partition an image into regions. One way to do is to by finding the regions directly. When we are segmenting based on regions, we are essentially finding similarities. In edge detection, we find differences. The basic approach is to start with a set of "seed" points, and from these grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed. Properties can be: ranges of intensity, color, texture, shape, model etc.
Region growth should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as intensity values, texture, and color are local in nature and do not take into account the "history" of region growth. Additional criteria that can increase the power of a region-growing algorithm utilize the concept of size,likeness between a candidate pixel and the pixels grown so far (such as a comparison of the intensity of a candidate and the average intensity of the grown region), and the shape of the region being grown.

Let:

f(x,y): input image
S(x,y): seed array containing 1's at the locations of seed points and 0's elsewhere
Q: a predicate to be applied at each location (x, y)
Arrays f and S are assumed to be of the same size

1. Find all connected components in $S(x, y)$ and reduce each connected component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.
2. Form an image f_Q such that, at each point (x, y), $f_Q(x,y) = 1$ if the input image satisfies a given predicate, Q, at those coordinates, $f_Q(x,y) = 0$ and otherwise. The predicate can use a threshold T for the same:

$$Q = \begin{cases} \text{TRUE} & \text{if the absolute difference of intensities} \\ & \text{between the seed and the pixel at } (x, y) \text{ is } \leq T \\ \text{FALSE} & \text{otherwise} \end{cases}$$

3. Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are L-connected to that seed point. (L could be 4, 8 or m)
4. Label each connected component in g with a different region label (e.g.,integers or letters). This is the segmented image obtained by region growing

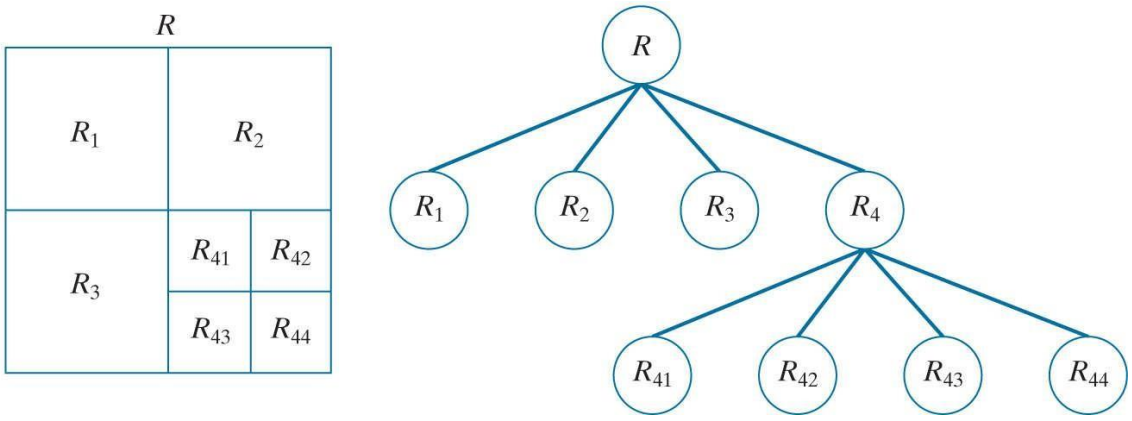
Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)

2. Region Splitting and Merging

An alternative is to subdivide an image initially into a set of disjoint regions and then merge and/or split the regions in an attempt to satisfy the conditions of segmentation stated in region growing

Let *R* represent the entire image region and select a predicate *Q*. One approach for segmenting *R* is to subdivide it successively into smaller and smaller quadrant regions so that, for any region. We start with the entire region, *R*. If we divide the image into quadrants. If *Q* is FALSE for any quadrant, we subdivide that quadrant into sub-quadrants, and so on. This splitting technique has a convenient representation in the form of so-called *quadtrees*; that is, trees in which each node has exactly four descendants

See a partitioned image and its corresponding quadtree:



1. Split into four disjoint quadrants any region R_i for which $Q(R_i)$ = FALSE
2. When no further splitting is possible, merge any adjacent regions R_j and R_k and for which $Q(R_j \cup R_k) = \text{TRUE}$
3. Stop when no further merging is possible.

Lab Assignments to complete in this session

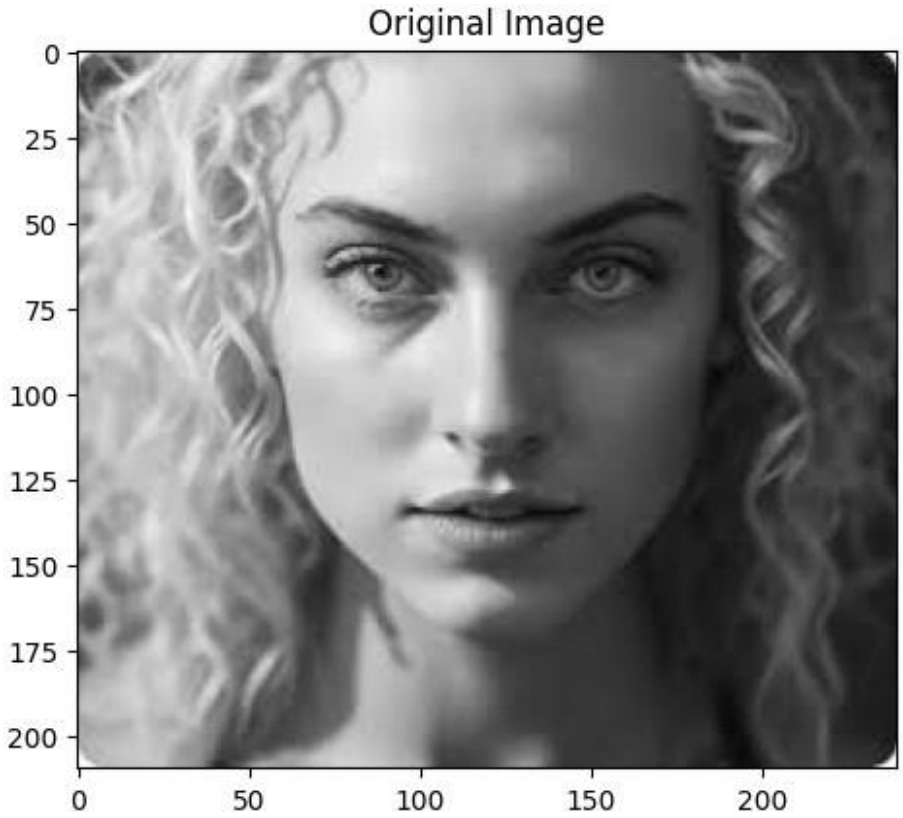
Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)

Problem Statement: Develop a Python program utilizing the OpenCV library to manipulate images from the Fashion MNIST digits dataset. The program should address the following tasks:

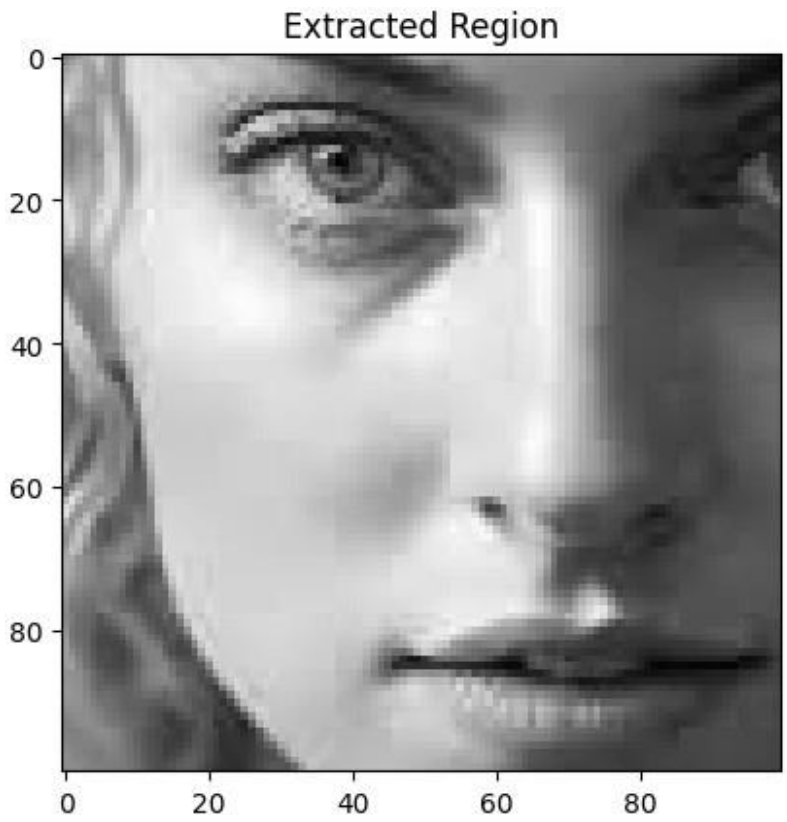
1. Importing libraries
2. Read random image(s) from the MNIST fashion dataset.
3. **Dataset Link:** Digit MNIST Dataset
4. Extract a region of the input image depending on a start position and a stop condition.
5. The input should be a single channel 8 bits image and the seed a pixel position (x, y).
6. The threshold corresponds to the difference between outside pixel intensity and mean intensity of region.
7. In case no new pixel is found, the growing stops.
8. Output a single channel 8 bits binary (0 or 255) image. Extracted region is highlighted in white – REGION GROWING
9. For REGION MERGING AND SPLITTING, divide the whole image into regions if the threshold predicate does not match
10. When no further division is possible, merge regions if their unions satisfy the predicate.
11. Provide outputs of both #8 and #10

The solution to the operations performed must be produced by scratch coding without the use of built in OpenCV methods.

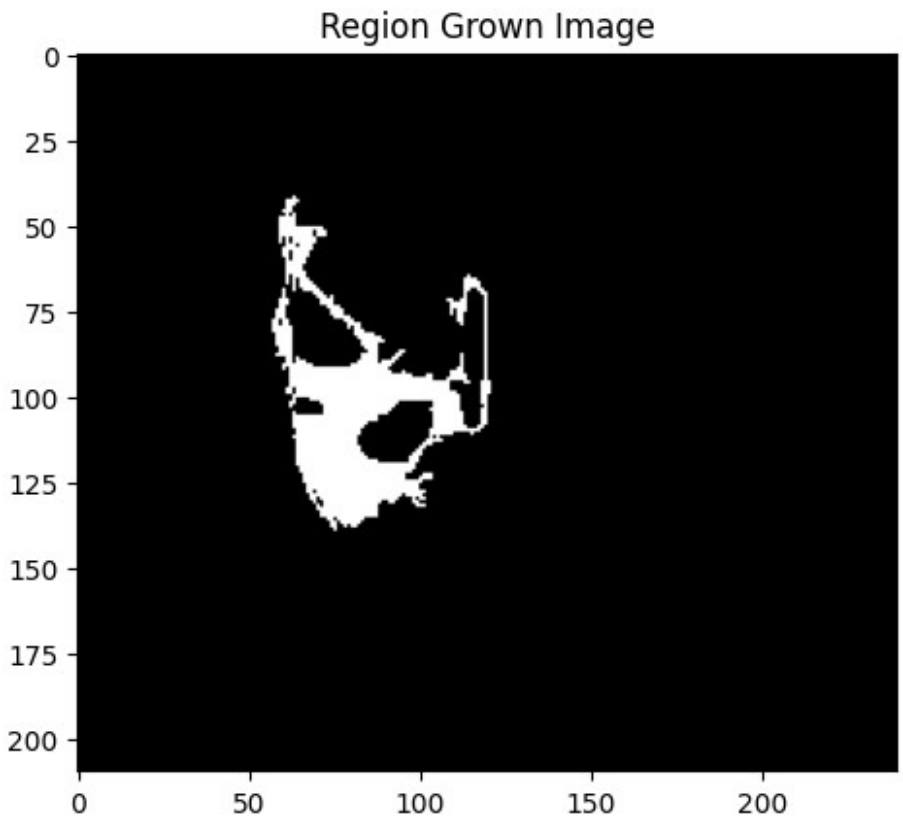
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('/content/grayscale.jpeg', 0)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.show()
```



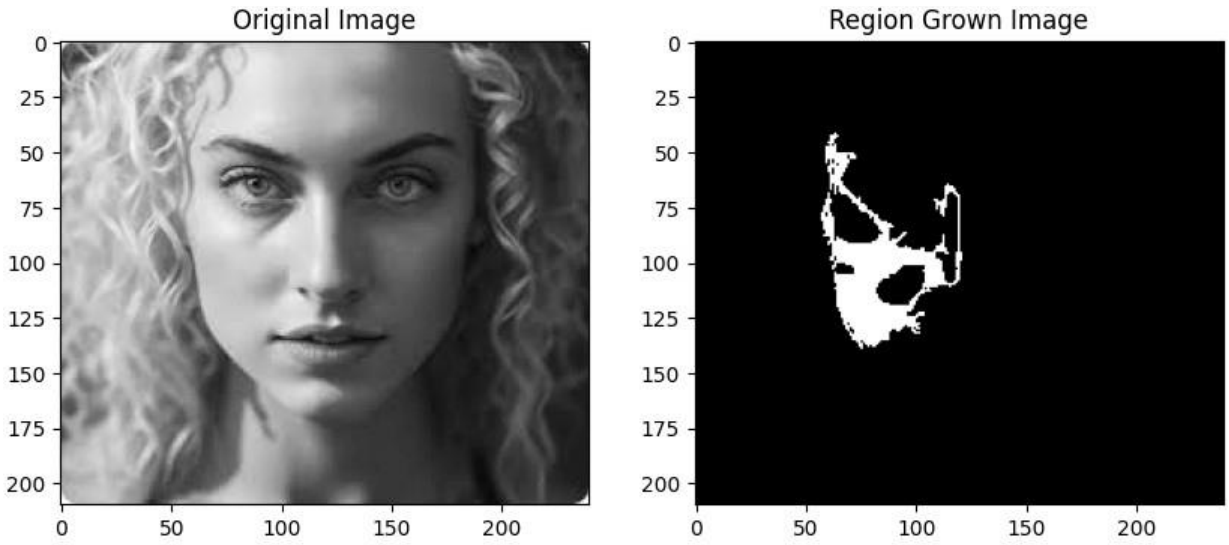

```
def extract_region(img, x_start, y_start, width, height):
    return img[y_start:y_start+height, x_start:x_start+width]

region = extract_region(image, 50, 50, 100, 100)
plt.imshow(region, cmap='gray')
plt.title("Extracted Region")
plt.show()
```



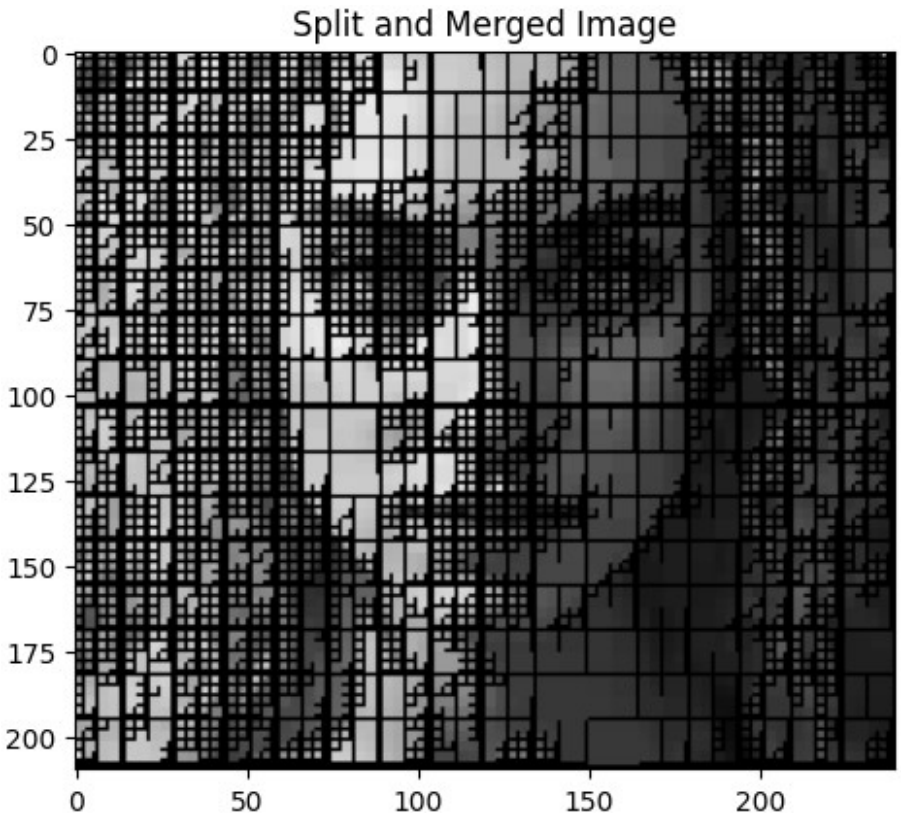

```
def region_growing(img, seed, threshold):
    h, w = img.shape
    segmented = np.zeros_like(img)
    mean_intensity = img[seed]
    pixels = [seed]

    while len(pixels) > 0:
        new_pixels  =  []
        for x, y in pixels:
            if segmented[y, x] == 0 and abs(int(img[y, x]) -
mean_intensity) <= threshold:
                segmented[y, x] = 255
                for nx, ny in [(x-1,y), (x+1,y), (x,y-1), (x,y+1)]:
                    if 0 <= nx < w and 0 <= ny < h:
                        new_pixels.append((nx, ny))
        pixels = new_pixels
    return segmented

seed = (100, 100)
result = region_growing(image, seed, 10)
plt.imshow(result, cmap='gray')
plt.title("Region Grown Image")
plt.show()
```



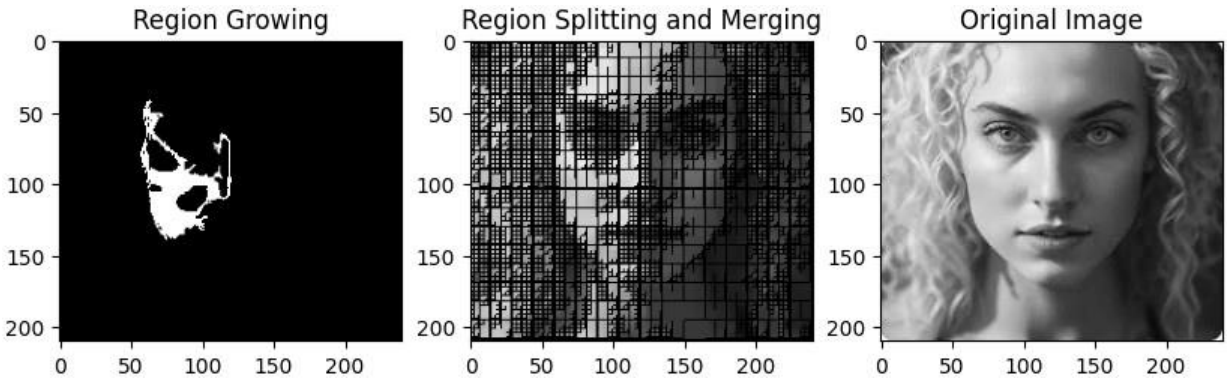

```
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
axes[0].imshow(image, cmap='gray')
axes[0].set_title("Original Image")

axes[1].imshow(result, cmap='gray')
axes[1].set_title("Region Grown Image")

plt.show()
```




```
def split_and_merge(img, threshold):
    h, w = img.shape
    segmented = np.zeros_like(img)

    def split_region(x_start, y_start, width, height):
        region = img[y_start:y_start+height, x_start:x_start+width]
        mean_intensity = np.mean(region)
        if np.max(region) - np.min(region) > threshold:
            if width > 1 and height > 1:
                split_region(x_start, y_start, width // 2, height //
2)
                split_region(x_start + width // 2, y_start, width //
2, height // 2)
                split_region(x_start, y_start + height // 2, width //
2, height // 2)
                split_region(x_start + width // 2, y_start + height //
2, width // 2, height // 2)
        else:
            segmented[y_start:y_start+height, x_start:x_start+width] =
mean_intensity

    split_region(0, 0, w, h)
    return segmented

result_split = split_and_merge(image, 10)
plt.imshow(result_split, cmap='gray')
plt.title("Split and Merged Image")
plt.show()
```




```
fig, axes = plt.subplots(1, 3, figsize=(10, 5))
axes[0].imshow(result, cmap='gray')
axes[0].set_title("Region Growing")

axes[1].imshow(result_split, cmap='gray')
axes[1].set_title("Region Splitting and Merging")

axes[2].imshow(image, cmap='gray')
axes[2].set_title("Original Image")

plt.show()
```



Application

The project leverages advanced image processing techniques, specifically region-growing and region-splitting/merging, to identify and highlight relevant areas within grayscale images. This approach is valuable in many applications where localized feature extraction is needed, such as medical imaging, object detection in computer vision, and quality inspection in industrial automation. By implementing these operations without relying on OpenCV's built-in functions, the solution deepens the understanding of underlying image processing principles and algorithms. This knowledge can be applied to a wide range of computer vision tasks requiring custom segmentation or detailed region analysis.

Conclusion

The project effectively demonstrates the application of region-growing and region-splitting/merging techniques, highlighting the benefits and challenges associated with each. Region-growing was successfully used to extract and highlight a specific area based on an initial seed position and a defined threshold, providing insights into connected regions with similar pixel intensities. Region-splitting and merging allowed for flexible partitioning of the image into smaller regions and merging them based on threshold conditions. This dual approach reinforces the versatility of these segmentation methods, showing that they can adapt to images of varying complexity and characteristics. Overall, this project illustrates the effectiveness of custom image segmentation techniques and reinforces fundamental skills in image processing, data handling, and computational algorithm design.