Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology

Department of Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar		
Class: TY	Division: B	Roll No: 372028
Semester: V	Academic Year: 2023-2024	
Subject Name & Code: Image Processing: ADUA31205(B)		
Title of Assignment: Perform edge based and region-based segmentation.		
Date of Performance: 27-09-2023	Date of Submission: 04-10-2023	

ASSIGNMENT NO. 6

Name: Siddhesh Dilip Khairnar
Roll No: 372028
PRN No: 22110398

Page No.
Date

DIP Assignment no 6

Aim: Perform edges based & region-based segmentation.

Learning objective: →

1. Learn fundamental concept of image segmentation, which involves dividing an image into region or object.
2. Explore region based segmentation method, including thresholding, region growing, watershed segmentation, and mean-shift clustering.
3. Learn how to choose the most suitable segmentation technique based on the specific characteristic and goals of the image processing task.

Theory: →

Edge-based and region based segmentation are techniques used in image processing to partition an image into meaningful regions or object. Here's a brief explanation of each method: →

1) **Edge-Based segmentation:** →

Edge-based segmentation focuses on identifying boundaries or edges edge or an image. Edges are abrupt changes in intensity or color in the image. Common techniques for edge-based segmentation include

A) **Gradient-Based method:** → These method detect edges by computing the gradient of the image. The canny edge detector is a well-known example.

b) **Laplacian of Gaussian:** This method applies a Gaussian filter followed by the Laplacian operator to highlight edges.

c) **Sobel and prewitt Operator:** These operator perform convolution on the image to detect edges.

2. Region-Based segmentation :

Region-based segmentation aim to group together pixels that share similar characteristic, such as color or intensity, to form meaningful regions or object. Common techniques for region-based segmentation include :→

A. Thresholding : This is a simple method where you select a threshold value, and pixel with intensity values above or below the threshold are assigned to different regions.

B. Region Growing : This method starts with a seed pixels and grows a region by adding neighboring pixels that meet certain similarity criteria.

C. Watershed segmentation : This method treats the image as a topographic map, & regions are formed where watershed meets.

D. Mean-shift Clustering : It's a clustering technique used for region-based segmentation, often in the context of color & texture analysis.

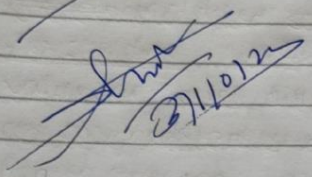
The choice b/w edge-based and region-based segmentation depends on the specific requirements of the image processing task. Edge-based method are useful when you want to detect object boundaries, while region-based methods are suitable for segmenting object based on their internal characteristics.

To perform these segmentation methods, you would typically use image processing libraries or software, such as opencv or MATLAB, & choose the appropriate algorithm based on the characteristics of the image and your specific goals.

Page No.
Date

Conclusion: →

We explored two fundamental image segmentation techniques: → edge-based segmentation & region based segmentation. These method are crucial in the field of image processing & computer vision, enabling the partitioning of image into meaningful regions or objed.



Program Code:

```
import cv2
import numpy as np

# Reading the input image
img = cv2.imread("C:/Users/asus/Downloads/download.jpeg", 0)

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5, 5), np.uint8)

# The first parameter is the original image,
# the kernel is the matrix with which the image is
# convolved, and the third parameter is the number
# of iterations, which will determine how much
# you want to erode/dilate a given image.
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)

cv2.imshow('Input', img)
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)
cv2.waitKey(0)

# Threshold the image
ret, img = cv2.threshold(img, 127, 255, 0)

# Step 1: Create an empty skeleton
size = np.size(img)
skel = np.zeros(img.shape, np.uint8)

# Get a Cross Shaped Kernel
element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))

# Repeat steps 2-4
while True:
    # Step 2: Open the image
    opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, element)

    # Step 3: Subtract open from the original image
    temp = cv2.subtract(img, opening)

    # Step 4: Erode the original image and refine the skeleton
    eroded = cv2.erode(img, element)
    skel = cv2.bitwise_or(skel, temp)
    img = eroded.copy()
```

```
    # Step 5: If there are no white pixels left
    # i.e., the image has been completely eroded, quit the loop
    if cv2.countNonZero(img) == 0:
        break

# Displaying the final skeleton
cv2.imshow("Skeleton", skel)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

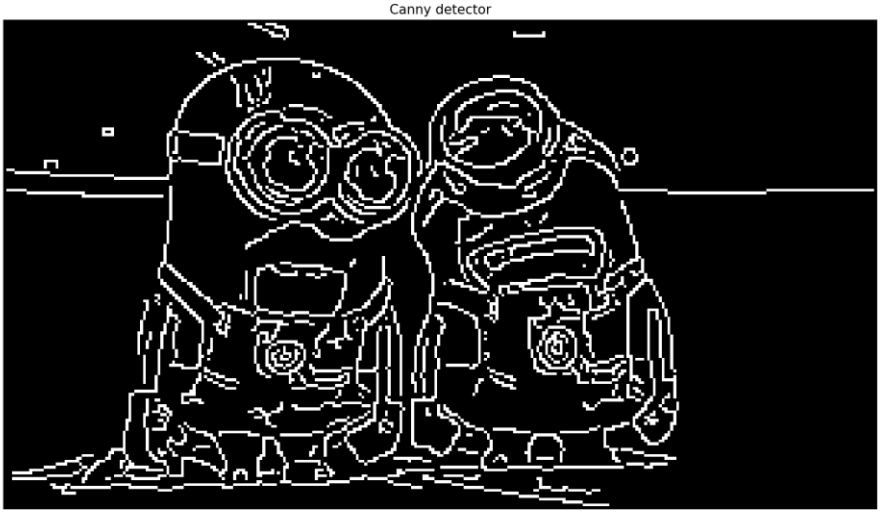
Figure 3

elevation_map

x=207.6 y=86.9
[0.269]

Figure 2

Canny detector



x=155.2 y=63.4
[0.000]