Practical No. 5

Aim :

Compare the results of any three edge detection algorithms on the same image dataset anddo the analysis of the result.

Objective : To Learn Classification Techniques for Image Segmentation.

Theory:

Image Segmentation

The first step in image analysis is to segment the image. Segmentation is the process to subdivide the image into its constituent parts or objects. The level to which subdivision is carried depends on the problem being solved.

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines,curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region are similar with respect to some characteristic or computed property, such as colour, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s). When
applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like marching cubes.

Segmentation algorithms for monochrome images are generally based on one of the two basic properties of grey level values:

1. Discontinuity: In this category, the approach is to partition the image based on abrupt changes in grey level.

2. Similarity: In this category, the approaches are based in thresholding, region growing, and region splitting and merging.

Edge Detection :

Edge detection is by far the most common approach for detecting meaningful discontinuities in gray level. The reason is that isolated points and the thin lines are not frequent occurrences in most practical applications. An edge is the boundary between two regions with relative distinct gray level properties.Note from the profile that an edge (transition from dark to light) is modeled as a smooth, rather than as anabrupt, change of gray level. This model reflects the fact that edges in digital images are generally slightly blurred as a result of sampling.

1.Gradient Operator :

The gradient of an image f (x,y) at the location (x,y) is defined as

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

Gradient vector always points in the direction of maximum rate of change off at co-ordinates (x,y). Magnitude of the vector is the main quantity in edge detection and given by,

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}.$$

Different Gradient Operators:

Use of different gradient operators for the following 3x3 grey level image is discussed below.

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

3x3 Grey Level Image

2.Roberts Operator:

One of the simplest way to implement first order partial derivative is to use Roberts cross-gradient operator. The mask for Roberts operator are given by,

-1	0

0	-1

0	-1

1	0

The two cross differences for x and y gradient components are given by equation,

$$Gx = Z9 - Z5 \text{ and } Gy = Z6 - Z8.$$

Then we can implement first order derivative ▼f at point Z5.

Mask of size 2x2 is awkward to implement because they don't have clear center. So generally, 3x3masks are used.

Sobel operator edge detection algorithm –

1. Read and load the input RGB image.

2. Convert the image to grayscale using the cv2.cvtColor () function.

3. Apply the Sobel operator to find the x and y gradients of the image using the cv2.Sobel()function.

4. Calculate the magnitude and angle of the gradients using the following formulae:

   •magnitude = np.sqrt(sobel_x**2 + sobel_y**2)

   •angle = np.arctan2(sobel_y, sobel_x)

1. Normalize the magnitude to the range [0, 255].

2. Create a binary image based on the gradient magnitudes.

3. Display the binary image with the detected edges using the cv2 library.

Canny edge detection algorithm –

1. Read and load the input RGB image.

2. Convert the image to grayscale using the cv2.cvtColor () function.

3. Apply Gaussian blur to reduce noise.

4. Use the Canny edge detector to find the edges by setting the upper and lower thresholds andusing the cv2.Canny () function.

5. Create a binary image using the obtained edges.

6. Display the binary image with the detected edges using the cv2 library.

Laplacian operator edge detection algorithm –

1. Read and load the input RGB image.

2. Convert the image to grayscale using the cv2.cvtColor () function.

3. Apply Gaussian blur to reduce noise.

4. Use the Laplacian operator edge detector to find the edges using the cv2.Laplacian () function.

5. Normalize the magnitude to the range [0, 255].

6. Create a binary image using the obtained edges.

7. Display the binary image with the detected edges using the cv2 library.

Conclusion:

We have successfully compare the results of edge detection algorithms on the image data set and analysis of the result.