Course Name:	Digital Signal & Image Processing Laboratory	Semester:	VI
Date of Performance:	09 / 04 / 2025	Batch No.:	B - 2
Faculty Name:	Dr. Om Goswami	Roll No.:	16014022050
Faculty Sign & Date:		Grade/Marks:	___ / 25

Experiment No.: 10

Title: Implementation of the different following Edge detection operators

Objective:

To learn and understand following edge detection operators.

i. Prewitt
ii. Sobel
iii. Robert
iv. Laplacian

COs to be achieved:

CO4: Utilize image processing in edge detection, image restoration and segmentation

Materials Required: MATLAB software

Books/ Journals/ Websites referred:

1. http://www.mathworks.com/support/
2. www.math.mtu.edu/~msgocken/intro/intro.html.
3. R. C.Gonsales R.E.Woods, “Digital Image Processing”, Second edition, Pearson Education
4. S.Jayaraman, S Esakkirajan, T Veerakumar “Digital Image Processing “Mc Graw Hill.
5. S.Sridhar,”Digital Image processing”, oxford university press, 1st edition."

Theory

Image segmentation can be achieved in two ways,

1. Segmentation based on discontinuities of intensity.

2. Segmentation based on similarities in intensity.

Edge information in an image is found by looking at the relationship a pixel has with its neighbourhoods. If a pixel’s gray-level value is similar to those around it, there is probably not an edge at that point. If a pixel’s has neighbors with widely varying gray levels, it may present an edge point.

Edge Detection Methods:

Many are implemented with convolution mask and based on discrete approximations to differential operators. Differential operations measure the rate of change in the image brightness function.Some operators return orientation information. Other only return information about the existence of an edge at each point.

Sobel Operator

The operator consists of a pair of 3×3 convolution kernels as shown in Figure 1. One kernel is simply the other rotated by 90°.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

Typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

which is much faster to compute. The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \arctan(Gy/Gx)$$

Robert's cross operator:

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point. The operator consists of a pair of 2×2 convolution kernels as shown in Figure. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

An approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

Which is much faster to compute? The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan(Gy/Gx) - 3\pi/4$$

Prewitt's operator:

Prewitt operator is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Laplacian of Gaussian:

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian Smoothing filter in order to reduce its sensitivity to noise. The operator normally takes a single gray level image as input and produces another gray level image as output.

The Laplacian *L(x,y)* of an image with pixel intensity values *I(x,y)* is given by:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Three commonly used small kernels are shown in Fig below

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

−1	2	−1
2	−4	2
−1	2	−1

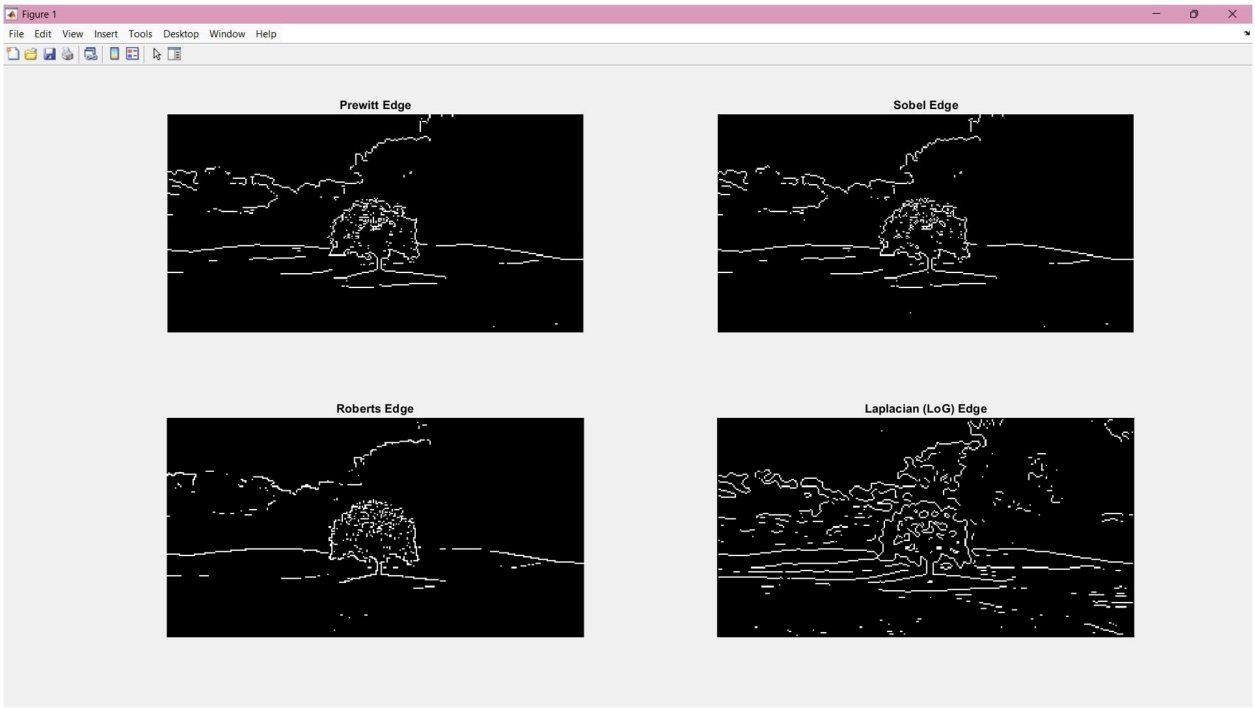
Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian Smoothed before applying the Laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

Write Algorithm and Matlab commands used:

```
I = imread('tree.jpg');

I_gray = rgb2gray(I);

edge_prewitt = edge(I_gray, 'Prewitt');
edge_sobel   = edge(I_gray, 'Sobel');
edge_roberts = edge(I_gray, 'Roberts');
edge_laplacian = edge(I_gray, 'log');
figure;
subplot(2,2,1), imshow(edge_prewitt), title('Prewitt Edge');
subplot(2,2,2), imshow(edge_sobel), title('Sobel Edge');
subplot(2,2,3), imshow(edge_roberts), title('Roberts Edge');
subplot(2,2,4), imshow(edge_laplacian), title('Laplacian (LoG) Edge');
```



Conclusion:

Edge detection operators like Prewitt, Sobel, Roberts, and Laplacian effectively highlight object boundaries by detecting intensity changes. These techniques are fundamental in image analysis for feature extraction and object recognition.

Post Lab Question:

1. **Explain the need of LOG operator.**

   The LoG operator is used for edge detection by combining Gaussian smoothing and the Laplacian operator. Noise can cause false edges during edge detection, especially with second-order derivatives like the Laplacian. To overcome this, LoG first smooths the image using a Gaussian filter to reduce noise, then applies the Laplacian to detect edges by identifying regions where the intensity changes sharply (zero-crossings of the second derivative).

   Why LoG is Needed:
   - Reduces the effect of noise by smoothing before detecting edges.
   - Detects edges as zero-crossings in the second derivative.
   - Suitable for detecting fine and detailed edges in noisy images.

2. **Explain the technique of thresholding for segmentation.**

   Thresholding is a simple image segmentation technique that separates objects from the background based on pixel intensity values. It converts a grayscale image into a binary image (black & white) by selecting a threshold value (T). Pixels with intensity above T are set to white (object), and those below T are set to black (background).

   Types of Thresholding:
   - Global Thresholding: A single threshold value is applied to the whole image.
   - Adaptive Thresholding: Different thresholds are calculated for different regions.
   - Otsu's Method: Automatically calculates the optimal threshold by minimizing intra-class variance.

   Uses of Thresholding:
   - Separating foreground from background.
   - Preprocessing for object detection, shape analysis, OCR, etc.

Signature of faculty in-charge with Date: