SOMAIYA
VIDYAVIHAR UNIVERSITY

K. J. Somaiya College of Engineering, Mumbai-77

Batch: A-4 **Roll No.:** 16010122151

Experiment No. 9

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implement following Edge detection operators (Prewitt, Sobel, Robert, and Laplacian).

Objective: To learn and understand different edge detection operators.

Expected Outcome of Experiment:

CO	Outcome
CO4	Design & implement algorithms for digital image enhancement, segmentation & restoration.

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."

Pre Lab/ Prior Concepts:

Image segmentation can be achieved in two ways,

1. Segmentation based on discontinuities of intensity.
2. Segmentation based on similarities in intensity.

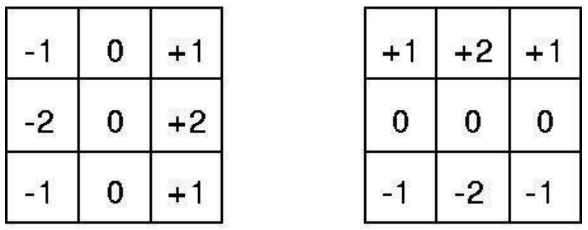
Edge information in an image is found by looking at the relationship a pixel has with its neighbourhoods. If a pixel's gray-level value is similar to those around it, there is probably not an edge at that point. If a pixel's has neighbors with widely varying gray levels, it may present an edge point.

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.

Implementation Details:

Write Algorithm and Matlab commands used:

ALGORITHM:

1. Load and Preprocess the Image

- Step 1: Read the input image and convert it to grayscale using `rgb2gray`.
- Step 2: Initialize the image dimensions (`img_width` and `img_height`) based on the grayscale image.

2. Extend the Image by Adding Border

- Step 3: Create an extended version of the grayscale image with a border of zeros (padding).
- Step 4: Copy the original image into the center of the extended image, leaving the border as zeros for boundary handling.

3. Apply Robert Edge Detection

- Step 5: Define the two 2x2 Robert edge detection kernels:
    - `robert_w1 = [1 0; 0 -1]`
    - `robert_w2 = [0 1; -1 0]`
- Step 6: For each pixel in the image (excluding the borders):
    - Apply the Robert operator (`robert_w1` and `robert_w2`) to calculate the gradient in the x and y directions (gx and gy).
    - Compute the magnitude of the gradient using a manual square root calculation.
- Step 7: Store the edge magnitude in the `robert_edge` array.

4. Apply Prewitt Edge Detection

- Step 8: Define the two 3x3 Prewitt edge detection kernels:

  - `prewitt_w1 = [1 1 1; 0 0 0; -1 -1 -1]`
  - `prewitt_w2 = [-1 0 1; -1 0 1; -1 0 1]`
- Step 9: For each pixel in the image (excluding the borders):
  - Apply the Prewitt operator (`prewitt_w1` and `prewitt_w2`) using convolution to calculate the gradients in the x and y directions (gx and gy).
  - Compute the magnitude of the gradient (edge strength).
- Step 10: Store the edge magnitude in the `prewitt_edge` array.

5. Apply Sobel Edge Detection

- Step 11: Define the two 3x3 Sobel edge detection kernels:
  - `sobel_w1 = [-1 0 1; -2 0 2; -1 0 1]`
  - `sobel_w2 = [-1 -2 -1; 0 0 0; 1 2 1]`
- Step 12: Use MATLAB's `conv2` function to convolve the grayscale image with the Sobel kernels (`sobel_w1` and `sobel_w2`) to calculate the gradients in the x and y directions (`sobel_gx` and `sobel_gy`).
- Step 13: Compute the magnitude of the gradient using `sqrt(gx^2 + gy^2)` to obtain the edge strength.

- Step 14: Store the result in the `sobel_edge` array.

6. Apply Laplace Edge Detection (Laplacian of Gaussian)

- Step 15: Define the Laplace kernel:
    - `laplace_w = [0 1 0; 1 -4 1; 0 1 0]`
- Step 16: Use `conv2` to convolve the grayscale image with the Laplace kernel.
- Step 17: Store the result in the `laplace_edge` array.

7. Display the Results

- Step 18: Display the following images in a subplot:
    1. The original grayscale image.
    2. The result of the Robert edge detection.
    3. The result of the Prewitt edge detection.
    4. The result of the Sobel edge detection.
    5. The result of the Laplace edge detection.

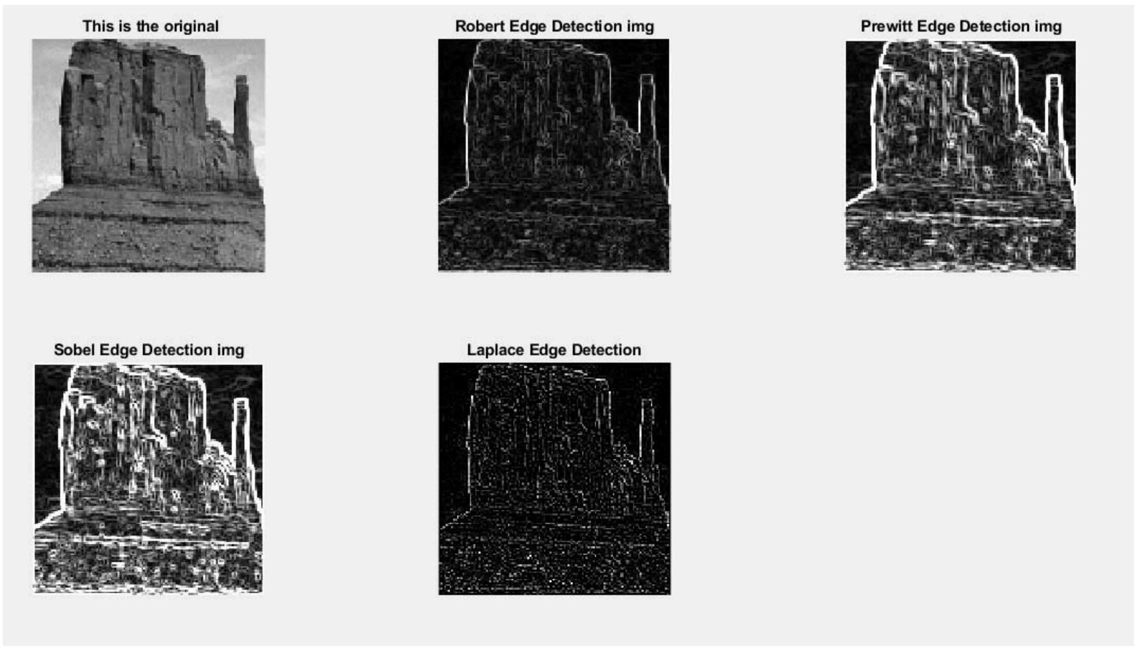
COMMANDS:

```
image = imread("rockbmp.jpg");

gray_image = rgb2gray(image); img_width = 0;
img_height = 0;

img_width = length(gray_image(:,1));

img_height = length(gray_image(1,:));

extended_image = [];

for i = 1:(img_width + 2)

   extended_image(i, :) = zeros(1, img_height + 2);

end

for i = 2:(img_width + 1)

   for j = 2:(img_height +

   1)

       extended_image(i, j) = gray_image(i-1, j-1);

   en

d end

robert_w1 = [1 0; 0 -1];

robert_w2 = [0 1; -1 0];

robert_edge = [];

for i = 1:img_width

   robert_edge(i, :) = zeros(1, img_height);

end

for row = 2:img_width+1

   for col =

   2:img_height+1

                    gx   =   robert_w1(1,1)*extended_image(row,col)

+ robert_w1(1,2)*extended_image(row,col+1)

+

robert_w1(2,1)*extended_image(row+1,col)

+ robert_w1(2,2)*extended_image(row+1,col+1);

                    gy   =   robert_w2(1,1)*extended_image(row,col)

+ robert_w2(1,2)*extended_image(row,col+1)

+

robert_w2(2,1)*extended_image(row+1,col)

+ robert_w2(2,2)*extended_image(row+1,col+1);


manual_sqrt = 0;

sum_of_squares = gx^2 +

gy^2; x = sum_of_squares;
```

```
guess = x / 2;
epsilon = 1e-6;
while abs(guess^2 - x) > epsilon
   guess = (guess + x / guess) / 2;
end
manual_sqrt = guess;
robert_edge(row-1, col-1) = manual_sqrt;
   end
end
prewitt_w1 = [1 1 1; 0 0 0; -1 -1 -1];
prewitt_w2 = [-1 0 1; -1 0 1; -1 0 1];
prewitt_edge = zeros(img_width, img_height);
for i = 1:img_width
   prewitt_edge(i, :) = zeros(1, img_height);
end
for row = 2:img_width+1
   for col =
   2:img_height+1
             gx = sum(sum(prewitt_w1 .* extended_image(row-1:row+1,
col-1:col+1)));
             gy = sum(sum(prewitt_w2 .* extended_image(row-1:row+1,
col-1:col+1)));
       prewitt_edge(row-1, col-1) = sqrt(gx.^2 + gy.^2);
   en
d end
sobel_w1 = [-1 0 1; -2 0 2; -1 0 1];
sobel_w2 = [-1 -2 -1; 0 0 0; 1 2 1];
sobel_gx = conv2(double(gray_image), sobel_w1,
'same'); sobel_gy = conv2(double(gray_image),
sobel_w2, 'same'); sobel_edge = sqrt(sobel_gx.^2 +
sobel_gy.^2); laplace_w = [0 1 0; 1 -4 1; 0 1 0];
laplace_edge = conv2(double(gray_image), laplace_w, 'same');
figure;
subplot(2, 3, 1);
imshow(gray_image);
title(' This is the original');
subplot(2, 3, 2);
imshow(uint8(robert_edge));
```

```
title('Robert Edge Detection img');
subplot(2, 3, 3);
imshow(uint8(prewitt_edge));
title('Prewitt Edge Detection img');
subplot(2, 3, 4);
imshow(uint8(sobel_edge));
title('Sobel Edge Detection img');
subplot(2, 3, 5);
imshow(uint8(laplace_edge));
title('Laplace Edge Detection');
```

OUTPUT:



Conclusion:- We successfully implemented all 4 edge detection filters using MATLAB and completed the experiment.

Date: 02/04/2025 **Signature of faculty in-charge**

Post Lab Descriptive Questions

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

The logarithmic (LOG) operator is widely used in digital signal processing (DSP) and digital image processing to enhance data interpretation, simplify complex relationships, and improve computational efficiency. Here's how it specifically applies:

1. **Dynamic Range Compression**
   - In image processing, intensity values in an image often span a wide range. For example, in medical images like X-rays or CT scans, pixel intensities can vary significantly.
   - Applying the **LOG operator** compresses these intensities, ensuring subtle details in darker regions (low-intensity values) become more visible without overwhelming high-intensity areas.
   - Similarly, in **digital signal processing (DSP)**, audio signals or seismic data can have a large dynamic range. Using a **logarithmic scale** simplifies their analysis and visualization.
2. **Feature Enhancement**
   - The **LOG operator enhances small differences** in lower intensity or amplitude regions.
   - This is critical for applications like **texture analysis in images** or **extracting weak signals** in noisy environments.
3. **Frequency Analysis (Fourier Transform Applications)**
   - In DSP, spectral analysis often involves very large ranges of frequency or amplitude values.
   - By applying a **logarithmic transformation**, the spectral data becomes easier to interpret visually.
   - This transformation compresses the high-frequency components while retaining detail in the low-frequency components.
4. **Edge Detection (Laplacian of Gaussian)**
   - In image processing, the **Laplacian of Gaussian (LOG) operator** is used for detecting edges, which are regions of rapid intensity changes.
   - This is vital for **segmentation tasks**, such as identifying object boundaries or features in images.

- Example applications include **segmenting anatomical structures in medical imaging** and **detecting edges in satellite images**.

2. Explain the technique of thresholding for segmentation.

Thresholding separates objects from their background by comparing pixel intensity levels to a specific threshold value.

- **Global Thresholding**
  - A single threshold value is applied to the entire image.
  - Example: In medical imaging, this helps isolate areas like **tumors in MRI scans** by setting an intensity threshold.
- **Adaptive Thresholding**
  - Used when image illumination is uneven (e.g., shadows or varying lighting conditions).
  - Adaptive techniques compute thresholds dynamically for different regions in the image, enabling better segmentation accuracy.
  - Example: **Detecting handwritten text** in scanned documents.
- **Otsu's Method**
  - An automated method that calculates an optimal threshold value based on image histograms.
  - It maximizes the separation between two classes (foreground and background).
  - Example: Used in **fingerprint recognition** applications.

2. Thresholding in Signal Processing

- **Peak Detection**
  - Thresholding can isolate significant peaks in time-series signals.
  - Example: **Identifying heartbeats in ECG signals** or detecting **seismic events** in earthquake monitoring.

- **Noise Removal**
    - By setting a threshold, low-amplitude noise in a signal can be suppressed while preserving meaningful signal components.