**Implementing 3 edge detection algorithms in MATLAB.**

**Theory -**

Edge detection is a fundamental process in computer vision, image processing, and machine vision that involves locating the boundaries between regions of varying intensity within an image. These boundaries, referred to as edges, represent discontinuities in the image brightness where there is a sharp change in pixel values. Edge detection serves several purposes, including image segmentation, feature extraction, and understanding the underlying structures in images.

Various mathematical methods exist for edge detection, ranging from simple gradient operators like the Sobel, Prewitt, and Roberts operators to more sophisticated approaches like the Canny edge detector.

Other popular edge detection techniques include the Laplacian of Gaussian (LoG), which combines the advantages of Gaussian smoothing and the Laplacian operator to reduce noise while enhancing edges, and the Hessian matrix, which provides information about both the presence and orientation of edges.

Edge detection plays a crucial role in numerous applications, including medical imaging, automatic traffic control, face recognition, and fingerprint recognition .

**3 Edge Detection Algorithms are Follows :**

**1 . Sobel Edge Detection**

Sobel edge detection is a widely used method for computing the gradient of image intensities at each pixel, thereby estimating the strength and direction of edges within an image. The Sobel operator utilizes two separate 3x3 kernels—one for the horizontal gradient and another for the vertical gradient—which are applied to the original image to produce estimates of the gradient magnitudes *Gx*​ and *Gy*​ .



These kernels are applied to the image to generate two new images representing the estimated horizontal (*Gx*​) and vertical (*Gy*​) gradients. By taking the square root of the sum of squares of these gradients, the total gradient magnitude *G* can be computed:



This approach allows the identification of edges, as regions with significant gradient magnitudes indicate abrupt changes in intensity. Additionally, the angle of the gradient can provide insight into the orientation of edges .

**Code :**

**% Read the image**

I = imread('C:/Users/DELL/Downloads/ImageEdge.jpg');

**% Convert the image to grayscale if it's not already**

if size(I, 3) == 3

I\_gray = rgb2gray(I);

else

I\_gray = I;

end

**% Sobel operator for edge detection**

hx = [-1 0 1; -2 0 2; -1 0 1];

hy = hx';

**% Convolve the image with the Sobel kernels**

Gx = conv2(double(I\_gray), hx, 'same');

Gy = conv2(double(I\_gray), hy, 'same');

**% Compute the gradient magnitude**

gradient\_magnitude = sqrt(Gx.^2 + Gy.^2);

**% Thresholding**

threshold = 100; % You may need to adjust this threshold

edges\_binary = gradient\_magnitude > threshold;

**% Display the original image and the detected edges**

figure;

subplot(1, 2, 1);

imshow(I);

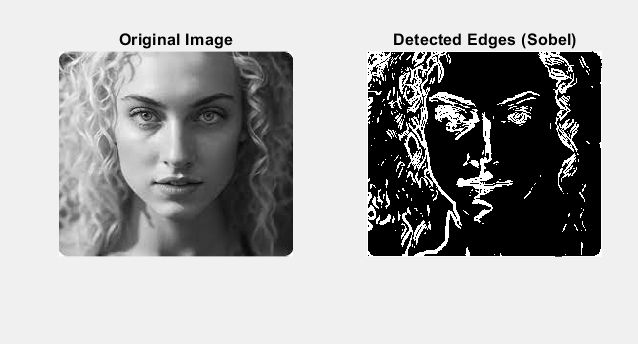
title('Original Image');

subplot(1, 2, 2);

imshow(edges\_binary);

title('Detected Edges (Sobel)');

**OUTPUT :**

****

**2 . Prewitt Edge Detection**

Prewitt edge detection is a technique used in computer vision for detecting edges in digital images. It's based on convolving the image with a set of predefined kernels to approximate the gradient of the image intensity function. The Prewitt operator calculates the gradient approximation in both the horizontal and vertical directions separately and then combines the results to find edge magnitudes and orientations.

Here's how Prewitt edge detection works:

1. **Gradient Approximation**: Prewitt edge detection uses two 3x3 convolution kernels: one for detecting vertical edges and another for horizontal edges. These kernels are:

Vertical Edge Detection Kernel (Gy):

-1 0 1 ;-1 0 1 ;-1 0 1

Horizontal Edge Detection Kernel (Gx):

-1 -1 -1; 0 0 0; 1 1 1

These kernels are applied to the image to compute approximations of the horizontal and vertical gradients.

1. **Convolution**: The image is convolved with these two kernels separately. Convolution involves sliding the kernels over the image and computing the weighted sum of pixel intensities at each position. This results in two gradient images: one representing the horizontal gradient (Gx) and the other representing the vertical gradient (Gy).
2. **Gradient Magnitude and Direction**: At each pixel, the gradient magnitude (edge strength) can be calculated using the Euclidean distance between the horizontal and vertical gradients:

**Gradient Magnitude = sqrt(Gx^2 + Gy^2)**

Additionally, the gradient direction (edge orientation) can be calculated using:

**Gradient Direction = arctan(Gy / Gx)**

Overall, Prewitt edge detection is a simple yet effective method for detecting edges in images. However, it may be sensitive to noise, and tuning the threshold parameter is crucial for obtaining desirable results .

**Code :**

**% Read the image**

I = imread('ImageEdge.jpg');

**% Convert the image to grayscale if it's not already**

if size(I, 3) == 3

I\_gray = rgb2gray(I);

else

I\_gray = I;

end

**% Prewitt operator for edge detection**

hx = [-1 0 1; -1 0 1; -1 0 1];

hy = hx';

**% Convolve the image with the Prewitt kernels**

Gx = conv2(double(I\_gray), hx, 'same');

Gy = conv2(double(I\_gray), hy, 'same');

**% Compute the gradient magnitude**

gradient\_magnitude = sqrt(Gx.^2 + Gy.^2);

**% Thresholding**

threshold = 100; % You may need to adjust this threshold

edges\_binary = gradient\_magnitude > threshold;

**% Display the original image and the detected edges**

figure;

subplot(1, 2, 1);

imshow(I);

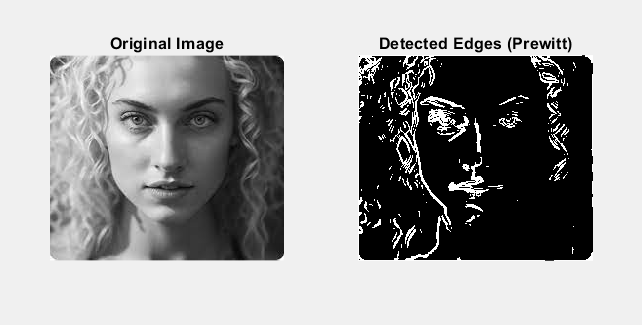
title('Original Image');

subplot(1, 2, 2);

imshow(edges\_binary);

title('Detected Edges (Prewitt)');

**OUTPUT :**

****

**3 . Canny Edge Detection**

Canny edge detection is a widely used edge detection algorithm in computer vision that works in multiple stages to produce smoother, thinner, and cleaner images than Sobel and Prewitt filters. The algorithm involves five main steps:

1. Convert the image to grayscale and remove noise using a Gaussian filter.
2. Calculate the gradient magnitude and direction using the Sobel operator in the horizontal and vertical directions.
3. Perform non-maximum suppression to remove unwanted pixels and retain only the local maxima in the gradient direction.
4. Apply thresholding to remove pixels below a certain threshold and retain pixels above a certain threshold.
5. Perform hysteresis tracking to make a pixel strong if any of the 8 neighboring pixels are strong.

**Code :**

**% Read the image**

I = imread('ImageEdge.jpg');

**% Convert the image to grayscale if it's not already**

if size(I, 3) == 3

I\_gray = rgb2gray(I);

else

I\_gray = I;

end

**% Perform Canny edge detection**

edges\_canny = edge(I\_gray, 'Canny');

**% Display the original image and the detected edges**

figure;

subplot(1, 2, 1);

imshow(I\_gray);

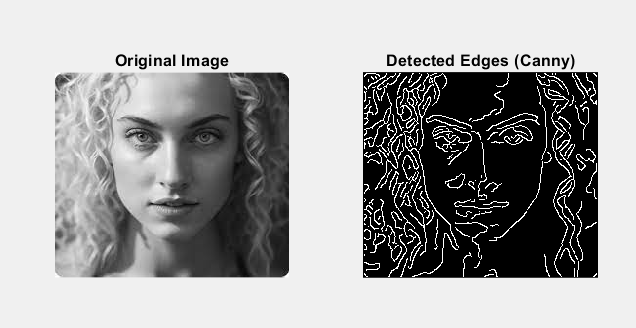
title('Original Image');

subplot(1, 2, 2);

imshow(edges\_canny);

title('Detected Edges (Canny)');

**OUTPUT :**

****

**Observations :**

After implementing the Sobel, Prewitt, and Canny edge detection algorithms, I observe the following :

1. **Sobel Edge Detection:**
   * The Sobel edge detection algorithm is relatively fast and simple to implement.
   * It is less sensitive to noise compared to the Prewitt operator.
   * It provides a good approximation of the gradient magnitude and direction.
   * However, it may produce thicker edges compared to the Canny edge detection algorithm.
2. **Prewitt Edge Detection:**
   * The Prewitt edge detection algorithm is similar to the Sobel edge detection algorithm.
   * It is also relatively fast and simple to implement.
   * However, it may be more sensitive to noise compared to the Sobel operator.
   * It may also produce thicker edges compared to the Canny edge detection algorithm.
3. **Canny Edge Detection:**
   * The Canny edge detection algorithm is more complex and computationally expensive compared to the Sobel and Prewitt edge detection algorithms.
   * However, it produces smoother, thinner, and cleaner edges compared to the Sobel and Prewitt edge detection algorithms .
   * It involves multiple stages, including smoothing, gradient calculation, non-maximum suppression, thresholding, and hysteresis tracking .
   * It is less sensitive to noise and provides a more accurate approximation of the gradient magnitude and direction .

In general, the choice of edge detection algorithm depends on the specific application and the desired trade-off between speed, accuracy, and complexity. The Sobel and Prewitt edge detection algorithms are good choices for simple applications where speed and simplicity are important, while the Canny edge detection algorithm is a better choice for more complex applications where accuracy and noise reduction are important.