Experiment-3

1) **Objective:** To enhance and detect edges in images by applying Sobel, Prewitt, and Laplacian filters, emphasizing significant transitions in pixel intensity.

2) **Software used:** **google collab(python)**

3) **Theory:**

1. Sobel Operator

The **Sobel filter** is a first-order derivative operator that computes an approximation of the gradient of the image intensity function. It emphasizes edges by detecting changes in intensity in both the horizontal (**Gx**) and vertical (**Gy**) directions.

Mathematical Representation:

The Sobel operator applies two convolution kernels to an image:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The gradient magnitude is then calculated as:

$$G = \sqrt{G_x^2 + G_y^2}$$

and the gradient direction (θ) is given by:

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Advantages:

- Reduces noise due to the weighted sum of surrounding pixels.
- More accurate than the Prewitt operator due to larger weight in the center.

2. Prewitt Operator

The **Prewitt filter** is similar to the Sobel operator but uses a simpler averaging method. It is also a first-order derivative operator used to compute edges by detecting intensity changes.

Mathematical Representation:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Like the Sobel operator, the gradient magnitude and direction are computed similarly.

Advantages:

- Simpler computation than the Sobel operator.
- Useful for detecting edges in environments where computational efficiency is a priority.

3. Laplacian Operator

The **Laplacian filter** is a second-order derivative operator that detects edges by finding regions of rapid intensity change. Unlike Sobel and Prewitt, which work with gradients in a specific direction, the Laplacian detects edges by measuring intensity changes in all directions.

Mathematical Representation:

A commonly used Laplacian kernel is:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Alternatively, another variant is:

$$L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Since the Laplacian is a second-order derivative, it is sensitive to noise. To reduce noise sensitivity, the Laplacian of Gaussian (LoG) is often used, where a Gaussian filter smooths the image before applying the Laplacian.

Since the Laplacian is a second-order derivative, it is sensitive to noise. To reduce noise sensitivity, the Laplacian of Gaussian (LoG) is often used, where a Gaussian filter smooths the image before applying the Laplacian.

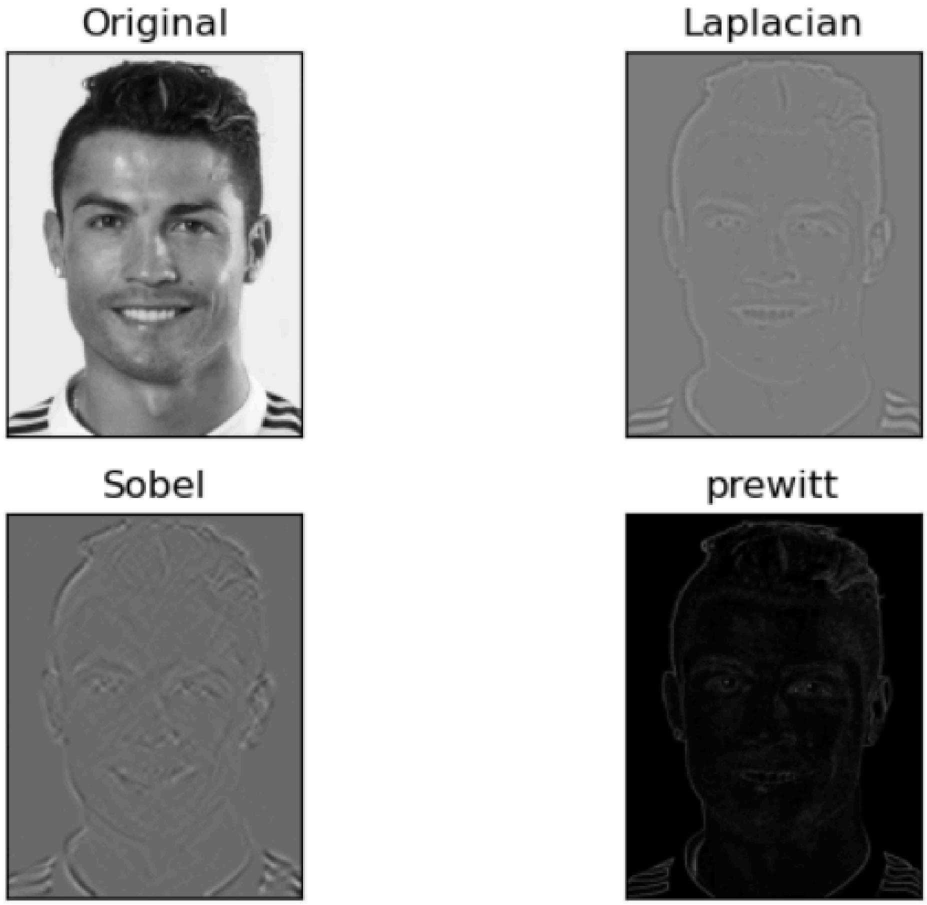
Advantages:

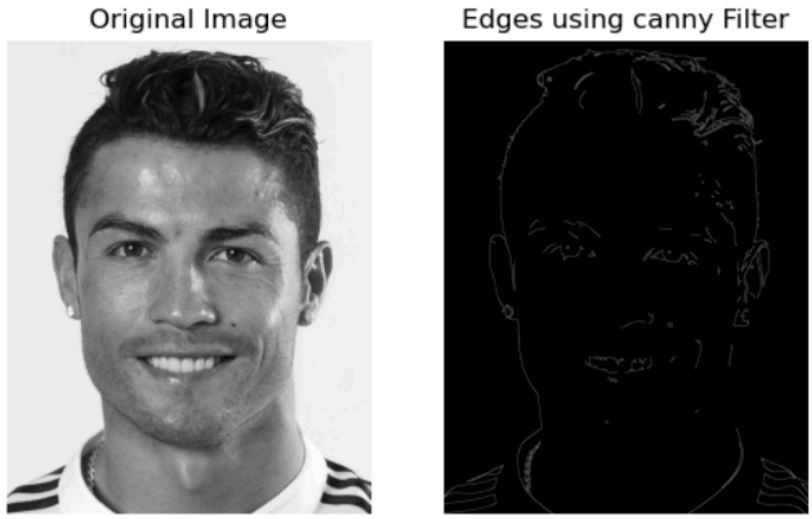
- Detects edges in all directions.
- Provides better localization of edges compared to first-order derivative methods.

4) Code:

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
from skimage import io, filters, color

img = cv.imread('C:\\Users\\DSE LAB 13\\Downloads\\ronaldo3.jpg',
cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with
os.path.exists()"

blurred_image = cv.GaussianBlur(img, (55, 55), 0)
laplacian = cv.Laplacian(blurred_image,cv.CV_64F)
sobel = cv.Sobel(blurred_image,cv.CV_64F,1,1,ksize=3)
prewitt = filters.prewitt(img)
canny = cv.Canny(blurred_image, threshold1=100, threshold2=200)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,3),plt.imshow(sobel,cmap = 'gray')
plt.title('Sobel'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,4),plt.imshow(prewitt,cmap = 'gray')
plt.title('prewitt'), plt.xticks([]), plt.yticks([])

plt.show()
```

5) Output:


Original
Laplacian
Sobel
prewitt


Original Image
Edges using canny Filter