جَـامعة
المَـنارة
MANARA UNIVERSITY

Manara university
Engineering college
Faculty of Informatics engineering
Computer vision
2025 – 2026

Lab session 7:
Edge detection (Sobel and Canny)

Tasks:

1. Read (lena.jfif) image in the gray scale.

2. Define Hx as:

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

3. Define Hy as:

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

4. Filter the image using convolution operation with Hx and store the result in Ix array.
5. Filter the image using convolution operation with Hy and store the result in Iy array.
6. Create the Magnitude image using the following formula:

$$M = \sqrt{Ix^2 + Iy^2}$$

7. Create the orientation image using the following formula:

$$A = arctan2(Iy, Ix)$$

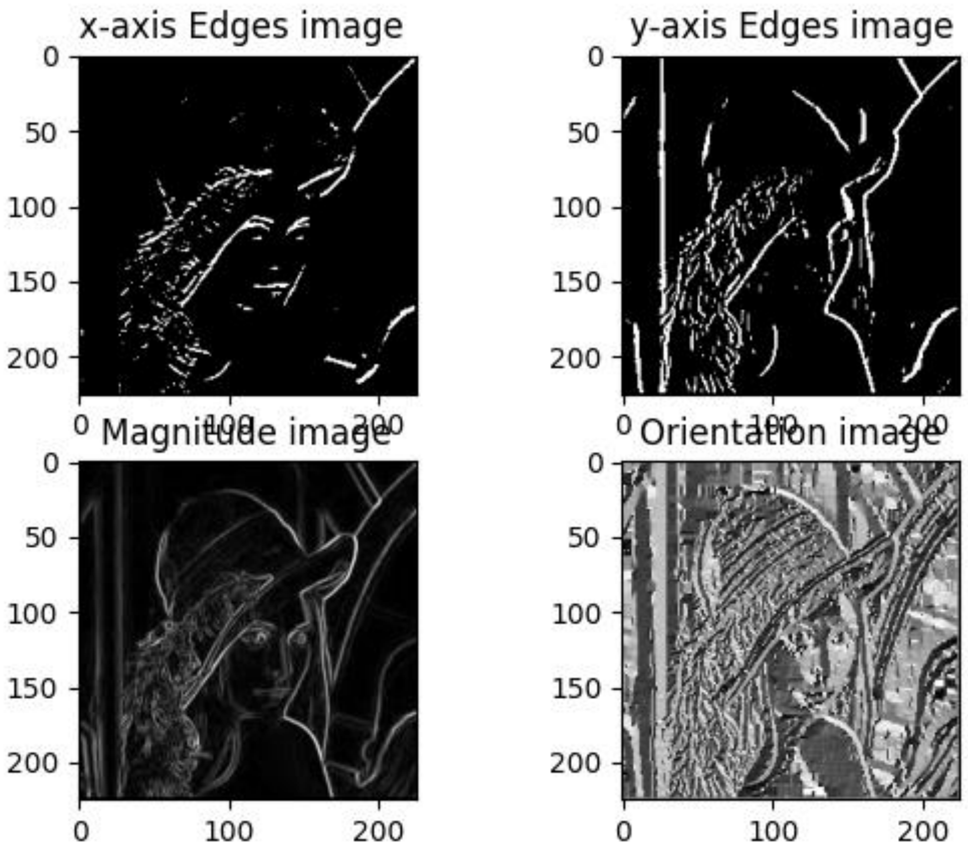
8. Apply a suitable threshold th = 100 on Ix and Iy to get Ixt and Iyt.
9. Display the results (Ixt. Iyt, M, A).

Needed Syntaxes:

M = np.sqrt(Ix*Ix + Iy*Iy)

A = np.atan2(Iy, Ix)

Results:


x-axis Edges image
y-axis Edges image
Magnitude image
Orientation image
0
50
100
150
200
0
100
200

Code:

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
def convolution(img, mask):
    r1, c1 = img.shape
    r2, c2 = mask.shape
    convolved_img = np.zeros((r1, c1), dtype=float)

    for i in range(r2//2,r1-r2//2):
        for j in range(c2//2,c1-c2//2):
            window = img[i-r2//2:i+r2//2+1, j-c2//2:j+c2//2+1]
            convolved_img[i, j] = np.sum( window * mask )
    return convolved_img

path =r' \\'
img = cv.imread(path+'lena.jfif', 0)
Hx = np.array( [[1, 2, 1], [0, 0, 0], [-1, -2, -1]] )
Hy = np.array( [[1, 0, -1], [2, 0, -2], [1, 0, -1]] )

Ix = convolution(img, Hx)
Iy = convolution(img, Hy)
M = np.sqrt(Ix*Ix + Iy*Iy)
```

```
A = np.atan2(Iy, Ix)


r, c = img.shape
Ixt = np.zeros((r, c))
Iyt = np.zeros((r, c))
th = 150


for i in range(r):
    for j in range(c):
        if Ix[i, j] > th:
            Ixt [i, j] = 255
        if Iy[i, j] > th:
            Iyt[i, j] = 255


fig, axis = plt.subplots(2, 2)
axis[0][0].set_title('x-axis Edges image ')
axis[0][0].imshow(Ixt, cmap='gray')


axis[0][1].set_title('y-axis Edges image')
axis[0][1].imshow(Iyt, cmap='gray')


axis[1][0].set_title('Magnitude image')
axis[1][0].imshow(M, cmap='gray')
```

```
axis[1][1].set_title('Orientation image')
axis[1][1].imshow(A, cmap='gray')
plt.show()
```

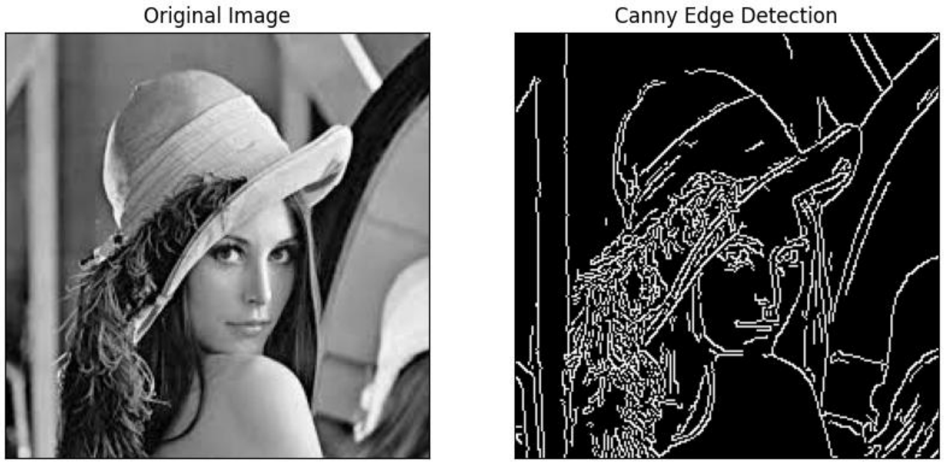
Canny edge detector :

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

# Load the image in grayscale
img = cv.imread('image.jpg', cv.IMREAD_GRAYSCALE)
edges = cv.Canny(img, 100, 200)

plt.figure(figsize=(10, 5))
plt.subplot(121), plt.imshow(img, cmap='gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(122), plt.imshow(edges, cmap='gray')
plt.title('Canny Edge Detection'), plt.xticks([]), plt.yticks([])
plt.show()
```



Original Image

Canny Edge Detection