****

**CSE439 Code Assignment**

**Name – Kazi Mahathir Rahman**

**ID – 23341066**

**Sec- 01**

**Faculty – MJA**

import cv2

import numpy as np

from matplotlib import pyplot as plt

image = cv2.imread("D:\CSE BRACU\CSE439-Machine-Vision\Assignment\Test\_image.png", cv2.IMREAD\_GRAYSCALE)

sobel\_x = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3)          # sobel x

sobel\_y = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3)          # sobel y

sobel\_45 = cv2.Sobel(image, cv2.CV\_64F, 1, 1, ksize=3)         # sobel 45

sobel\_135\_kernel = np.array([

    [-1, -1,  2],

    [-1,  2, -1],

    [ 2, -1, -1]

])

sobel\_135 = cv2.filter2D(image, cv2.CV\_64F, sobel\_135\_kernel)     # sobel 135

laplacian = cv2.Laplacian(image, cv2.CV\_64F, ksize=3)             # laplacian

sobel\_x = np.uint8(np.absolute(sobel\_x))

sobel\_y = np.uint8(np.absolute(sobel\_y))

sobel\_45 = np.uint8(np.absolute(sobel\_45))

sobel\_135 = np.uint8(np.absolute(sobel\_135))

laplacian = np.uint8(np.absolute(laplacian))

# Display the images

titles = ['Original', 'Sobel X', 'Sobel Y', 'Sobel 45', 'Sobel 135', 'Laplacian']

images = [image, sobel\_x, sobel\_y, sobel\_45, sobel\_135, laplacian]

for i in range(len(images)):

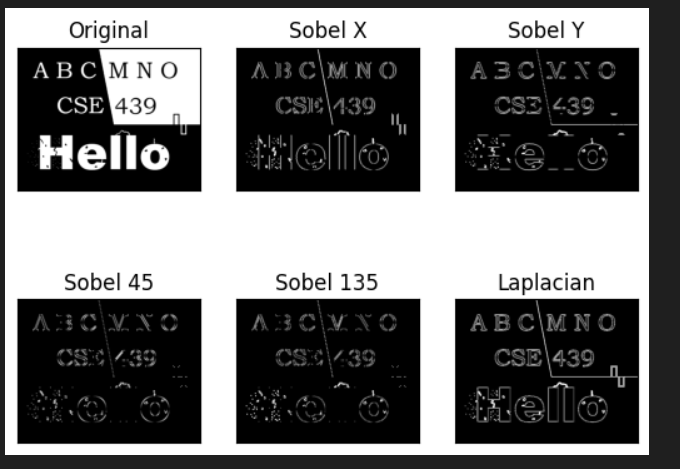
    plt.subplot(2, 3, i + 1)

    plt.imshow(images[i], cmap='gray')

    plt.title(titles[i])

    plt.xticks([]), plt.yticks([])

plt.show()

****

**Laplacian was better, why?**

The Laplacian operator stands out among edge detection algorithms due to its capability to precisely locate edges in an image, regardless of their orientation, making it highly effective in capturing fine details and subtle changes in intensity. Unlike Sobel operators, which are sensitive to noise and detect edges along specific directions, the Laplacian operator computes the second derivative of the image intensity, resulting in sharper edge localization and enhanced response to edge structures. This makes it particularly suitable for applications requiring high accuracy in edge detection, such as medical imaging or quality control in manufacturing. However, its sensitivity to noise and potential for producing responses to non-edge features must be considered when choosing the appropriate edge detection method for a given scenario.

kernel\_5x5 = np.ones((5, 5), dtype=np.uint8)

kernel\_9x9 = np.ones((9, 9), dtype=np.uint8)

# erosion and dilation

erosion\_5x5 = cv2.erode(image, kernel\_5x5, iterations=1)

dilation\_5x5 = cv2.dilate(image, kernel\_5x5, iterations=1)

erosion\_9x9 = cv2.erode(image, kernel\_9x9, iterations=1)

dilation\_9x9 = cv2.dilate(image, kernel\_9x9, iterations=1)

# opening and closing

opening\_5x5 = cv2.morphologyEx(image, cv2.MORPH\_OPEN, kernel\_5x5)

closing\_5x5 = cv2.morphologyEx(image, cv2.MORPH\_CLOSE, kernel\_5x5)

opening\_9x9 = cv2.morphologyEx(image, cv2.MORPH\_OPEN, kernel\_9x9)

closing\_9x9 = cv2.morphologyEx(image, cv2.MORPH\_CLOSE, kernel\_9x9)

plt.figure(figsize=(9, 7))

# First row: Erosion and Dilation (5x5)

plt.subplot(441), plt.imshow(image, cmap='gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(442), plt.imshow(erosion\_5x5, cmap='gray')

plt.title('Erosion (5x5)'), plt.xticks([]), plt.yticks([])

plt.subplot(443), plt.imshow(dilation\_5x5, cmap='gray')

plt.title('Dilation (5x5)'), plt.xticks([]), plt.yticks([])

# Second row: Erosion and Dilation (9x9)

plt.subplot(445), plt.imshow(image, cmap='gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(446), plt.imshow(erosion\_9x9, cmap='gray')

plt.title('Erosion (9x9)'), plt.xticks([]), plt.yticks([])

plt.subplot(447), plt.imshow(dilation\_9x9, cmap='gray')

plt.title('Dilation (9x9)'), plt.xticks([]), plt.yticks([])

# Third row: Opening and Closing (5x5)

plt.subplot(449), plt.imshow(image, cmap='gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(4,4,10), plt.imshow(opening\_5x5, cmap='gray')

plt.title('Opening (5x5)'), plt.xticks([]), plt.yticks([])

plt.subplot(4,4,11), plt.imshow(closing\_5x5, cmap='gray')

plt.title('Closing (5x5)'), plt.xticks([]), plt.yticks([])

# Fourth row: Opening and Closing (9x9)

plt.subplot(4,4,13), plt.imshow(image, cmap='gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(4,4,14), plt.imshow(opening\_9x9, cmap='gray')

plt.title('Opening (9x9)'), plt.xticks([]), plt.yticks([])

plt.subplot(4,4,15), plt.imshow(closing\_9x9, cmap='gray')

plt.title('Closing (9x9)'), plt.xticks([]), plt.yticks([])

plt.tight\_layout()

plt.show()

****

1. **Erosion (5x5)**: Erosion is a morphological operation that "erodes away" the boundaries of foreground objects in an image. It achieves this by moving a kernel over the image and replacing each pixel's value with the minimum value of its neighboring pixels covered by the kernel. This operation tends to shrink the boundaries of foreground objects. In the case of a 5x5 structuring element, the erosion operation will result in more pronounced thinning of object boundaries compared to smaller structuring elements.

2. **Dilation (5x5)**: Dilation is the opposite of erosion. It "dilates" the boundaries of foreground objects by replacing each pixel's value with the maximum value of its neighboring pixels covered by the kernel. Dilation tends to expand the boundaries of foreground objects. With a 5x5 structuring element, the dilation operation will lead to a thicker and more pronounced boundary of objects compared to smaller structuring elements.

3. **Erosion (9x9)**: Similar to the erosion operation with a 5x5 structuring element, but with a larger kernel size (9x9), the erosion operation here will cause more aggressive shrinking of the foreground object boundaries. This operation tends to remove finer details from the objects.

4. **Dilation (9x9)**: Similarly, dilation with a 9x9 structuring element will result in a more extensive expansion of foreground object boundaries compared to the 5x5 structuring element. It will create a thicker boundary around objects.

5. **Opening (5x5**): Opening is a combination of erosion followed by dilation. It is useful in removing noise from images while preserving the shape and size of the objects. The erosion operation removes small objects and "breaks" thin connections, and the subsequent dilation operation restores the size of the remaining objects. With a 5x5 structuring element, opening will effectively remove smaller noise while preserving the main structure of objects.

6. **Closing (5x5)**: Closing is the reverse of opening, consisting of dilation followed by erosion. It is effective in closing small holes in objects or in joining small breaks in the contours. With a 5x5 structuring element, closing will fill in small gaps or holes within objects.

7. **Opening (9x9)**: Similar to opening with a 5x5 structuring element, but with a larger kernel size (9x9), this operation will be more effective in removing noise and small details while preserving the main structures of objects.

8. **Closing (9x9)**: Similar to closing with a 5x5 structuring element, but with a larger kernel size (9x9), this operation will effectively fill in larger gaps or holes within objects.