**Report on**

Project 1: Image Edge Detection

Name: Maliha Jannat (马里哈)

Student ID: W2110805009

Edge detection simplifies the image data while preserving the structural properties, making further analysis easier an essential technique in image processing. It is used to identify points in an image where intensity changes abruptly. These points often correspond to boundaries of objects in the image.

We are going to solve the program using Python high-level programing language in Visual Studio code. To perform operation related to Computer Vision, arrays, matrixes and plot, we have to download 3 libraries using terminal:

opencv-python, numpy and matplotlib

PS C:\Users\maliha\Desktop\Project 1> pip install opencv-python numpy matplotlib

And need to import those libraries into the code:

import cv2

import numpy as np

import matplotlib.pyplot as plt

* 1. Filters:

Apply the Sobel filter, which are commonly used for edge detection:

sobel\_horizontal = np.array([[-1, 0, 1],

                             [-2, 0, 2],

                             [-1, 0, 1]])

sobel\_vertical = np.array([[-1, -2, -1],

                           [ 0,  0,  0],

                           [ 1,  2,  1]])

Load and preprocessing images with grayscale for simplicity, as edge detection works best on intensity values:

image\_license\_plate = cv2.imread('license\_plate.png', cv2.IMREAD\_GRAYSCALE)

image\_lena = cv2.imread('lena.jpg', cv2.IMREAD\_GRAYSCALE)

Applying the filters on both pictures by apply\_filter() function with horizontal filter to detect horizontal edges and vertical filter to detect vertical edges:

def apply\_filter(image, kernel):

    return cv2.filter2D(image, -1, kernel)

horizontal\_license\_plate = apply\_filter(image\_license\_plate, sobel\_horizontal)

vertical\_license\_plate = apply\_filter(image\_license\_plate, sobel\_vertical)

horizontal\_lena = apply\_filter(image\_lena, sobel\_horizontal)

vertical\_lena = apply\_filter(image\_lena, sobel\_vertical)

To display the result initializing fig and axes size and call the methods:

fig, axes = plt.subplots(2, 3, figsize=(12, 8))

axes[0, 0].imshow(image\_license\_plate, cmap='gray')

axes[0, 0].set\_title('License Plate (Original)')

axes[0, 1].imshow(horizontal\_license\_plate, cmap='gray')

axes[0, 1].set\_title('License Plate (Horizontal Edges)')

axes[0, 2].imshow(vertical\_license\_plate, cmap='gray')

axes[0, 2].set\_title('License Plate (Vertical Edges)')

axes[1, 0].imshow(image\_lena, cmap='gray')

axes[1, 0].set\_title('Lena (Original)')

axes[1, 1].imshow(horizontal\_lena, cmap='gray')

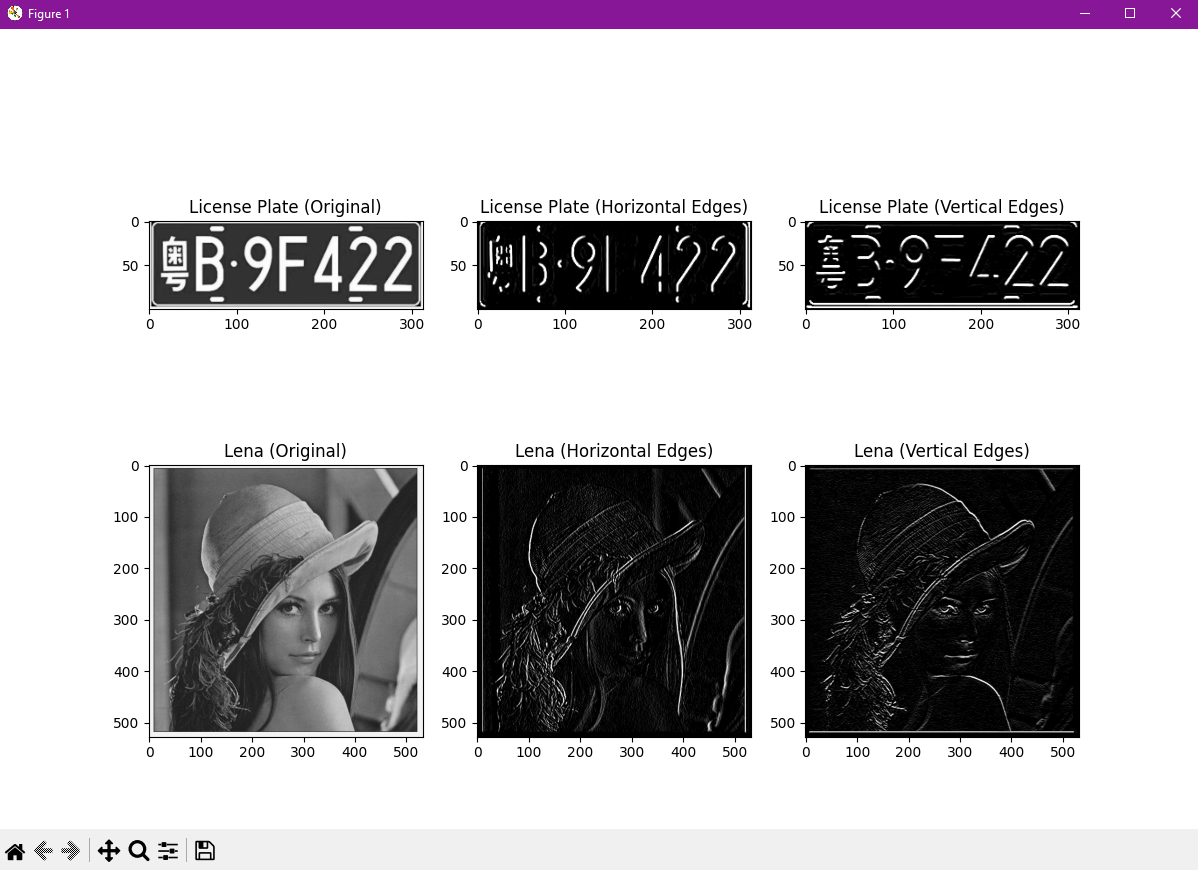
axes[1, 1].set\_title('Lena (Horizontal Edges)')

axes[1, 2].imshow(vertical\_lena, cmap='gray')

axes[1, 2].set\_title('Lena (Vertical Edges)')

plt.show()

Run the code and get the results:



* 1. Sobel edge detector:

Initialize image array with 2 pictures and threshold for binarization:

images = ["license\_plate.png", "lena.jpg"]

threshold = 100

Create the function with image array and threshold arguments which loads and preprocess image array with grayscale, apply horizontal and vertical Sobel filter, calculate gradient magnitude and binarization. Returns original image, gradient magnitude image and binary image:

def sobel\_edge\_detection(image\_path, threshold):

    img = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

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

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

    gradient\_magnitude = np.sqrt(sobel\_x\*\*2 + sobel\_y\*\*2)

    gradient\_magnitude = np.uint8(np.clip(gradient\_magnitude, 0, 255))

    \_, binary\_edges = cv2.threshold(gradient\_magnitude, threshold, 255, cv2.THRESH\_BINARY)

    return img, gradient\_magnitude, binary\_edges

Create the loop which applies sobel edge detection for each image in the image array and displays the result:

for image\_path in images:

    original, gradient, edges = sobel\_edge\_detection(image\_path, threshold)

    plt.figure(figsize=(12, 4))

    plt.subplot(1, 3, 1)

    plt.title('Original Image')

    plt.imshow(original, cmap='gray')

    plt.axis('off')

    plt.subplot(1, 3, 2)

    plt.title('Gradient Magnitude')

    plt.imshow(gradient, cmap='gray')

    plt.axis('off')

    plt.subplot(1, 3, 3)

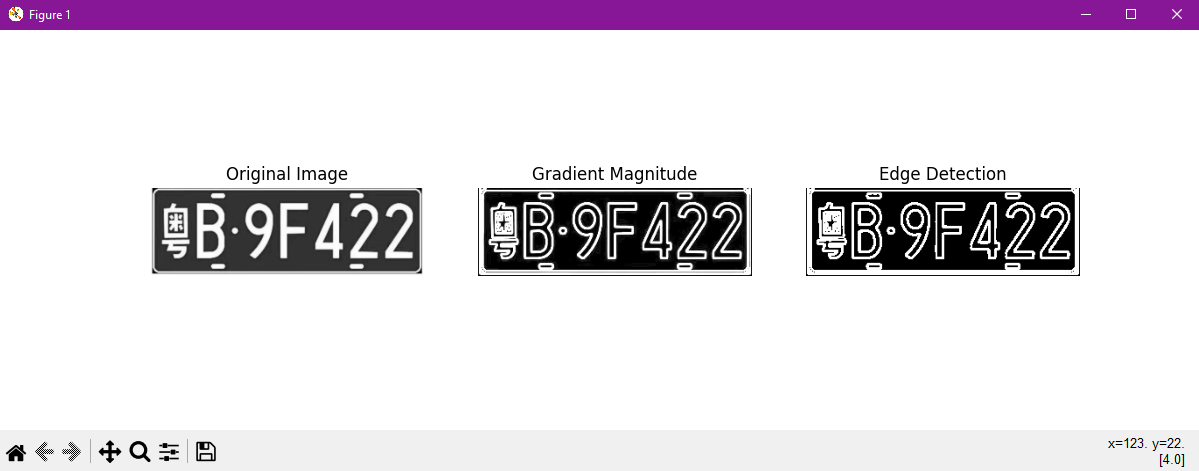
    plt.title('Edge Detection')

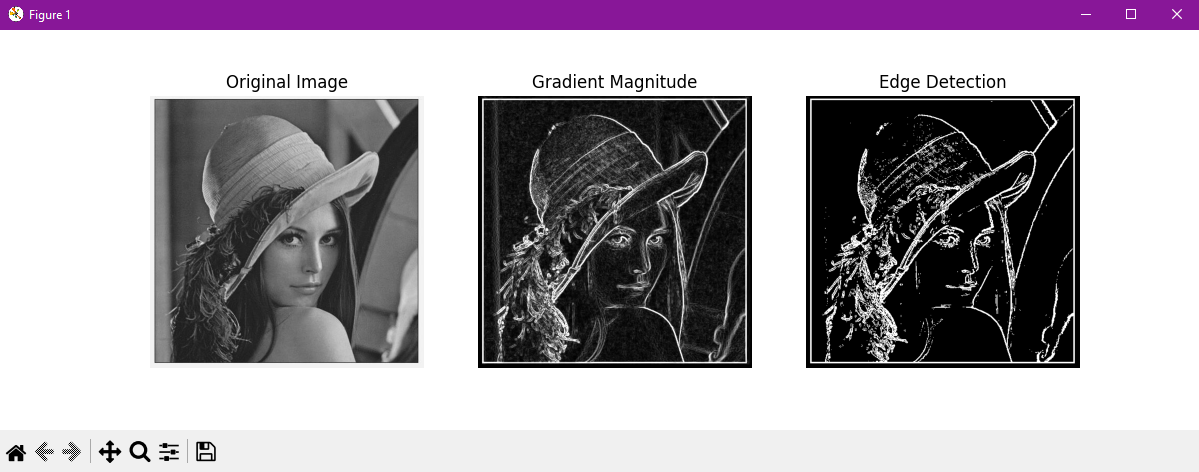
    plt.imshow(edges, cmap='gray')

    plt.axis('off')

    plt.show()

Run the code and get the results:





* 1. Edge detection using library functions:

Create 2 variables – 1st variable is loaded original “lena” picture and 2nd is converted to grayscale “lena” picture.

lena = io.imread('lena.jpg')

gray\_lena = color.rgb2gray(lena)

Initialize and apply different edge detectors –

* Sobel edge detection:

sobel\_edges = cv2.Sobel(np.float32(gray\_lena), cv2.CV\_64F, 1, 1, ksize=3)

* Prewitt edge detection:

prewitt\_kernel\_x = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])

prewitt\_kernel\_y = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])

prewitt\_edges\_x = cv2.filter2D(gray\_lena, -1, prewitt\_kernel\_x)

prewitt\_edges\_y = cv2.filter2D(gray\_lena, -1, prewitt\_kernel\_y)

prewitt\_edges = np.sqrt(prewitt\_edges\_x\*\*2 + prewitt\_edges\_y\*\*2)

* Roberts edge detection:

roberts\_kernel\_x = np.array([[1, 0], [0, -1]])

roberts\_kernel\_y = np.array([[0, 1], [-1, 0]])

roberts\_edges\_x = cv2.filter2D(gray\_lena, -1, roberts\_kernel\_x)

roberts\_edges\_y = cv2.filter2D(gray\_lena, -1, roberts\_kernel\_y)

roberts\_edges = np.sqrt(roberts\_edges\_x\*\*2 + roberts\_edges\_y\*\*2)

* Canny edge detection:

canny\_edges = cv2.Canny(np.uint8(gray\_lena\*255), 100, 200)

Displays the results by running the code:

plt.figure(figsize=(10, 10))

plt.subplot(2, 2, 1)

plt.imshow(sobel\_edges, cmap='gray')

plt.title('Sobel Edge Detection')

plt.subplot(2, 2, 2)

plt.imshow(prewitt\_edges, cmap='gray')

plt.title('Prewitt Edge Detection')

plt.subplot(2, 2, 3)

plt.imshow(roberts\_edges, cmap='gray')

plt.title('Roberts Edge Detection')

plt.subplot(2, 2, 4)

plt.imshow(canny\_edges, cmap='gray')

plt.title('Canny Edge Detection')

plt.show()

