**Contents**

Question 1 -------------------------------------------------------------------------------------------- 2

1. Image of code --------------------------------------------------------------------------------- 2

2. Code ---------------------------------------------------------------------------------------------- 3

3. Test ----------------------------------------------------------------------------------------------- 4

Question 2 -------------------------------------------------------------------------------------------- 5

1. Image of code --------------------------------------------------------------------------------- 5

2. Code ----------------------------------------------------------------------------------------------6

3. Test ----------------------------------------------------------------------------------------------- 7

Question 3 -------------------------------------------------------------------------------------------- 8

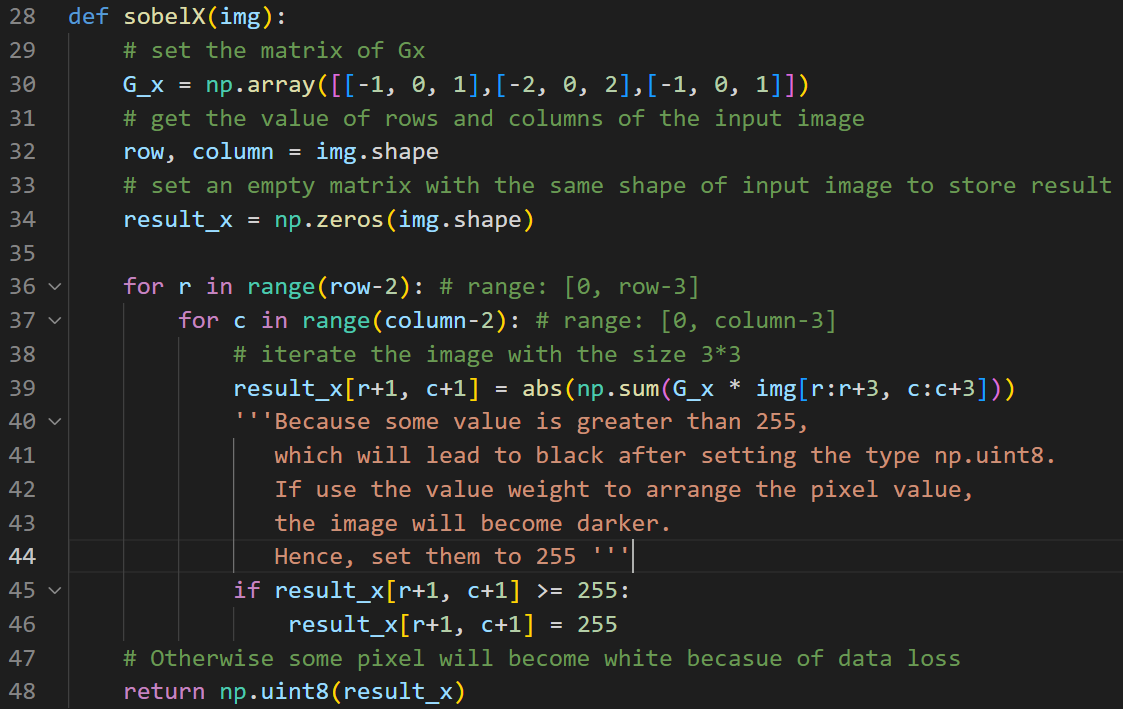
1. Image of code --------------------------------------------------------------------------------- 8

2. Code ----------------------------------------------------------------------------------------------9

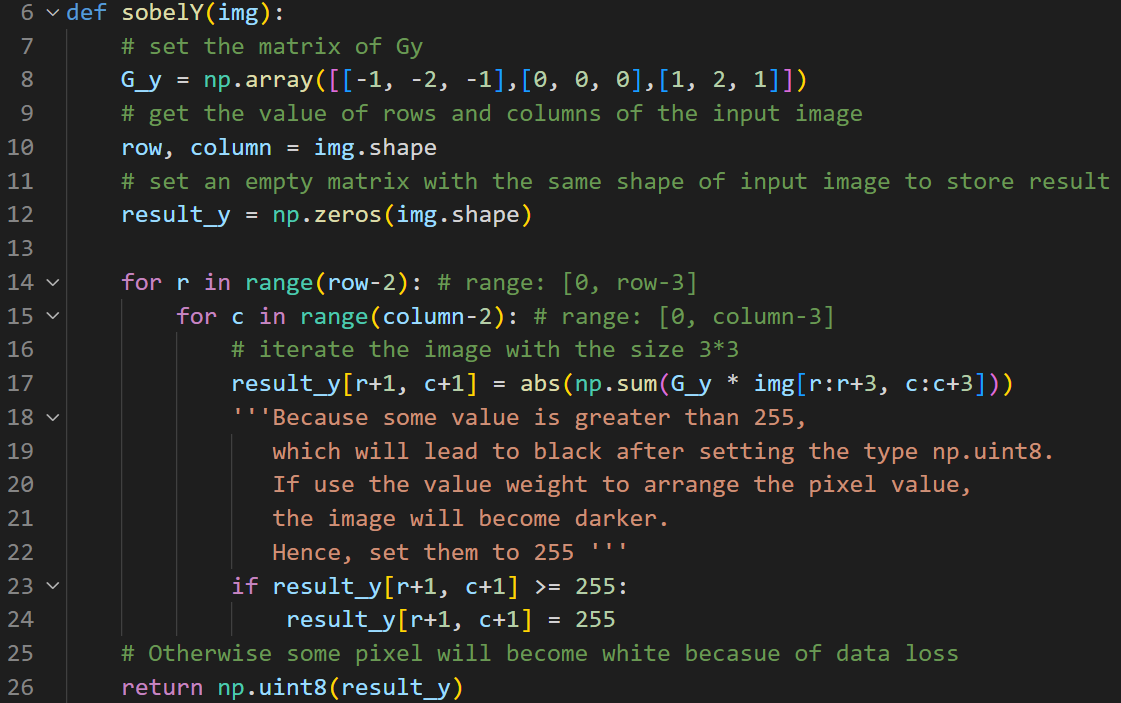
3. Test ----------------------------------------------------------------------------------------------- 10

**Question 1**

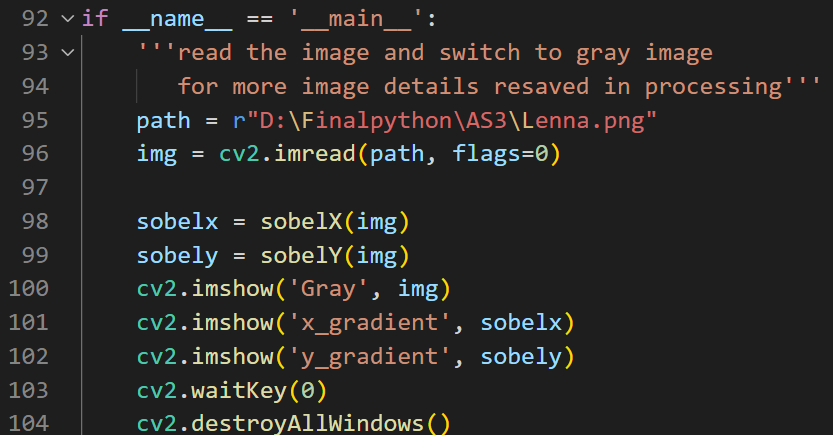
1. **Image of code for the better reading experience**
2. Function to calculate the gradient of X axis



1. Function to calculate the gradient of Y axis



1. Code for test the functions



1. **Code details**

from matplotlib import pyplot as plt

import numpy as np

import cv2

def sobelY(img):

# set the matrix of Gy

G\_y = np.array([[-1, -2, -1],[0, 0, 0],[1, 2, 1]])

# get the value of rows and columns of the input image

row, column = img.shape

# set an empty matrix with the same shape of input image to store result

result\_y = np.zeros(img.shape)

for r in range(row-2): # range: [0, row-3]

for c in range(column-2): # range: [0, column-3]

# iterate the image with the size 3\*3

result\_y[r+1, c+1] = abs(np.sum(G\_y \* img[r:r+3, c:c+3]))

'''Because some value is greater than 255,

which will lead to black after setting the type np.uint8.

If use the value weight to arrange the pixel value,

the image will become darker.

Hence, set them to 255 '''

if result\_y[r+1, c+1] >= 255:

result\_y[r+1, c+1] = 255

# Otherwise some pixel will become white becasue of data loss

return np.uint8(result\_y)

def sobelX(img):

# set the matrix of Gx

G\_x = np.array([[-1, 0, 1],[-2, 0, 2],[-1, 0, 1]])

# get the value of rows and columns of the input image

row, column = img.shape

# set an empty matrix with the same shape of input image to store result

result\_x = np.zeros(img.shape)

for r in range(row-2): # range: [0, row-3]

for c in range(column-2): # range: [0, column-3]

# iterate the image with the size 3\*3

result\_x[r+1, c+1] = abs(np.sum(G\_x \* img[r:r+3, c:c+3]))

'''Because some value is greater than 255,

which will lead to black after setting the type np.uint8.

If use the value weight to arrange the pixel value,

the image will become darker.

Hence, set them to 255 '''

if result\_x[r+1, c+1] >= 255:

result\_x[r+1, c+1] = 255

# Otherwise some pixel will become white becasue of data loss

return np.uint8(result\_x)

# Main function

if \_\_name\_\_ == '\_\_main\_\_':

'''read the image and switch to gray image

for more image details resaved in processing'''

path = r"D:\Finalpython\AS3\Lenna.png"

img = cv2.imread(path, flags=0)

sobelx = sobelX(img)

sobely = sobelY(img)

cv2.imshow('Gray', img)

cv2.imshow('x\_gradient', sobelx)

cv2.imshow('y\_gradient', sobely)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Test**

The figures below are the gray image of Lenna, x gradient and y gradient image after running the code.

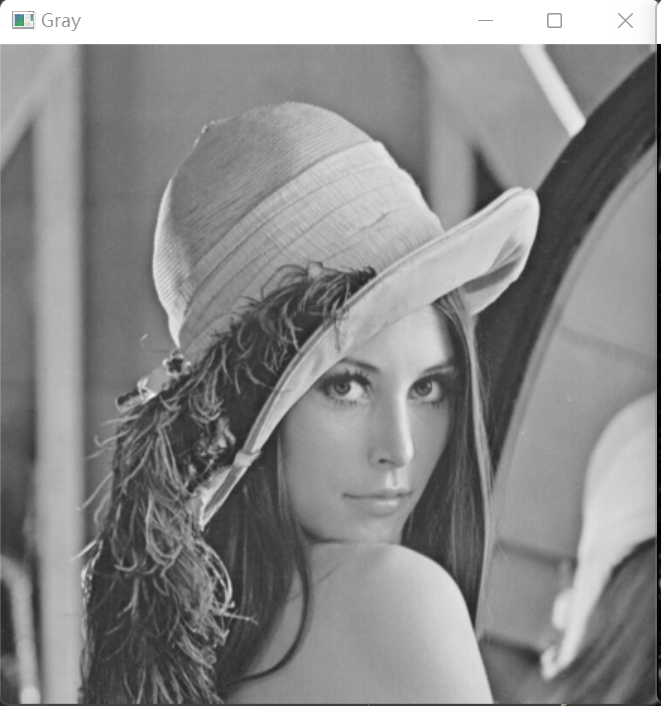


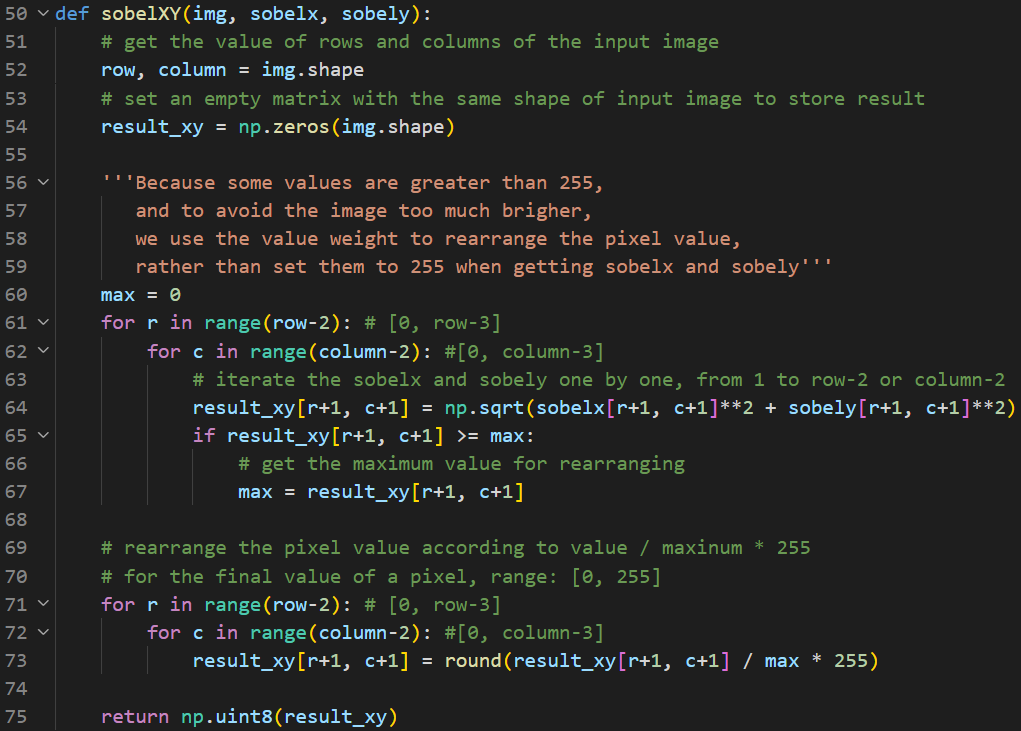
Figure 1: Gray level image of Lenna



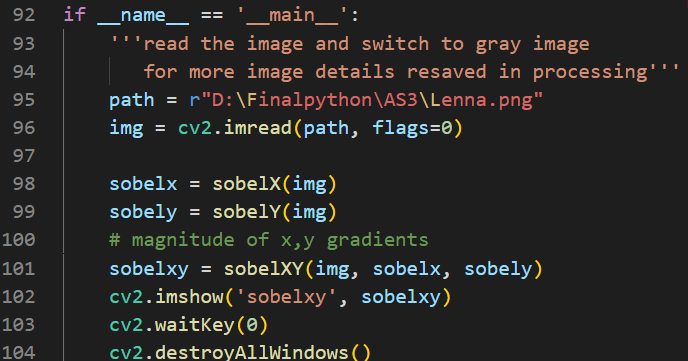
Figure 2: X gradient image (left) and Y gradient image (right)

**Question 2**

1. **Image of code for the better reading experience**
2. Function to calculate the magnitude of gradients for each pixel based on the x gradient and y gradient got from the two functions in the question 1



1. Main function



1. **Code details**

The code here only includes the function of calculating magnitude of x, y gradients and main function.

def sobelXY(img, sobelx, sobely):

# get the value of rows and columns of the input image

row, column = img.shape

# set an empty matrix with the same shape of input image to store result

result\_xy = np.zeros(img.shape)

'''Because some values are greater than 255,

and to avoid the image too much brigher,

we use the value weight to rearrange the pixel value,

rather than set them to 255 when getting sobelx and sobely'''

max = 0

for r in range(row-2): # [0, row-3]

for c in range(column-2): #[0, column-3]

# iterate the sobelx and sobely one by one, from 1 to row-2 or column-2

result\_xy[r+1, c+1] = np.sqrt(sobelx[r+1, c+1]\*\*2 + sobely[r+1, c+1]\*\*2)

if result\_xy[r+1, c+1] >= max:

# get the maximum value for rearranging

max = result\_xy[r+1, c+1]

# rearrange the pixel value according to value / maxinum \* 255

# for the final value of a pixel, range: [0, 255]

for r in range(row-2): # [0, row-3]

for c in range(column-2): #[0, column-3]

result\_xy[r+1, c+1] = round(result\_xy[r+1, c+1] / max \* 255)

return np.uint8(result\_xy)

# Main function

if \_\_name\_\_ == '\_\_main\_\_':

'''read the image and switch to gray image

for more image details resaved in processing'''

path = r"D:\Finalpython\AS3\Lenna.png"

img = cv2.imread(path, flags=0)

sobelx = sobelX(img)

sobely = sobelY(img)

# magnitude of x,y gradients

sobelxy = sobelXY(img, sobelx, sobely)

cv2.imshow('sobelxy', sobelxy)

cv2.waitKey(0)

cv2.destroyAllWindows()

# set the threshould value

thValue = 127

binary = binaryImg(thValue, sobely)

1. **Test**

After running the code,

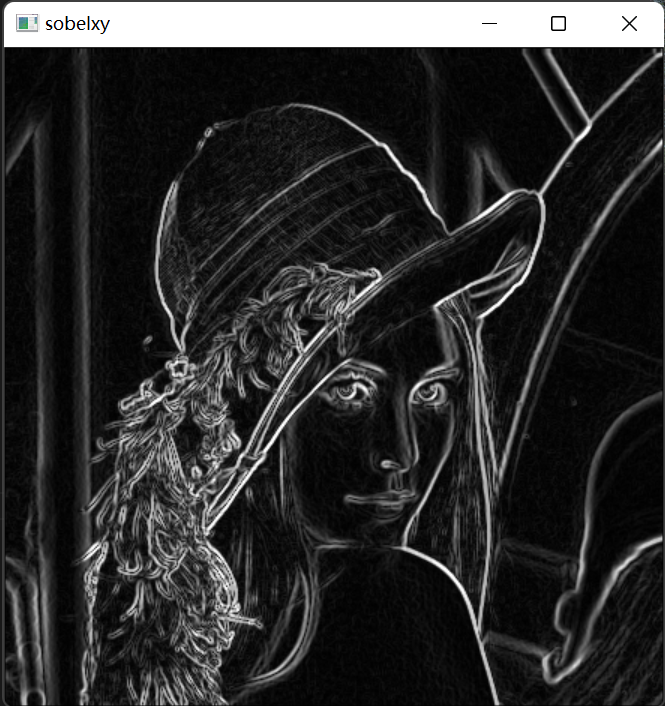
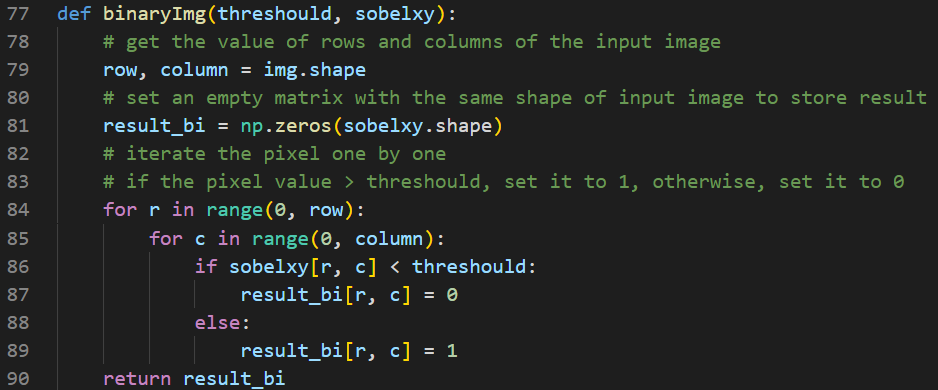


Figure 3: Edges of input image

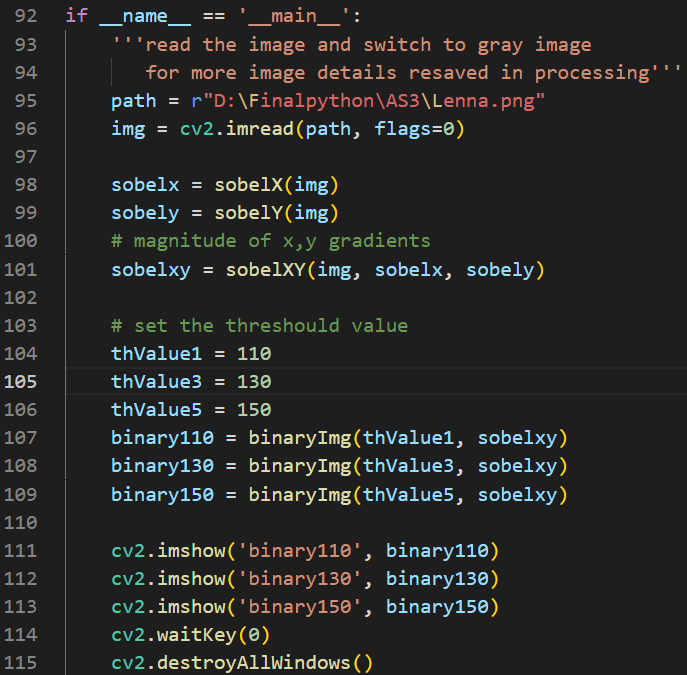
*Mention: The code should be run together with the two functions in the question 1, since the calculation of magnitude of gradients needs x, y gradients’ values.*

**Question 3**

1. **Image of code for the better reading experience**
2. Function to get the binary image based on the threshold given in the main function



1. Main function



1. **Code details**

The code here only includes the function to get the binary image based on the threshold given in the main function, and the complete main function to get the result which will be shown in the next part, *Test.*

def binaryImg(threshould, sobelxy):

# get the value of rows and columns of the input image

row, column = img.shape

# set an empty matrix with the same shape of input image to store result

result\_bi = np.zeros(sobelxy.shape)

# iterate the pixel one by one

# if the pixel value > threshould, set it to 1, otherwise, set it to 0

for r in range(0, row):

for c in range(0, column):

if sobelxy[r, c] < threshould:

result\_bi[r, c] = 0

else:

result\_bi[r, c] = 1

return result\_bi

if \_\_name\_\_ == '\_\_main\_\_':

'''read the image and switch to gray image

for more image details resaved in processing'''

path = r"D:\Finalpython\AS3\Lenna.png"

img = cv2.imread(path, flags=0)

sobelx = sobelX(img)

sobely = sobelY(img)

# magnitude of x,y gradients

sobelxy = sobelXY(img, sobelx, sobely)

# set the threshould value

thValue1 = 110

thValue3 = 130

thValue5 = 150

binary110 = binaryImg(thValue1, sobelxy)

binary130 = binaryImg(thValue3, sobelxy)

binary150 = binaryImg(thValue5, sobelxy)

cv2.imshow('binary110', binary110)

cv2.imshow('binary130', binary130)

cv2.imshow('binary150', binary150)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Test**

I set the threshold value as 110, 130, 150, which can show the difference quite clear. For example, as you can see the edges in the mouth area become fewer as the threshold value become larger.

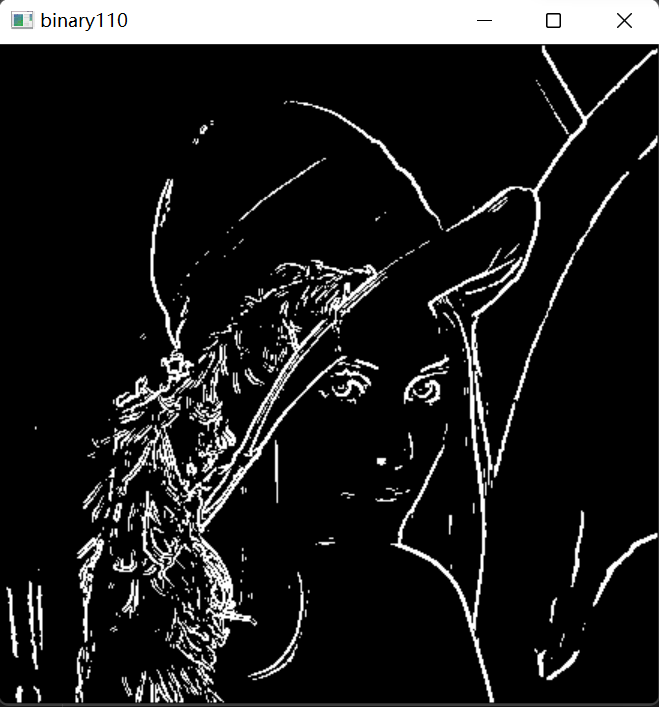
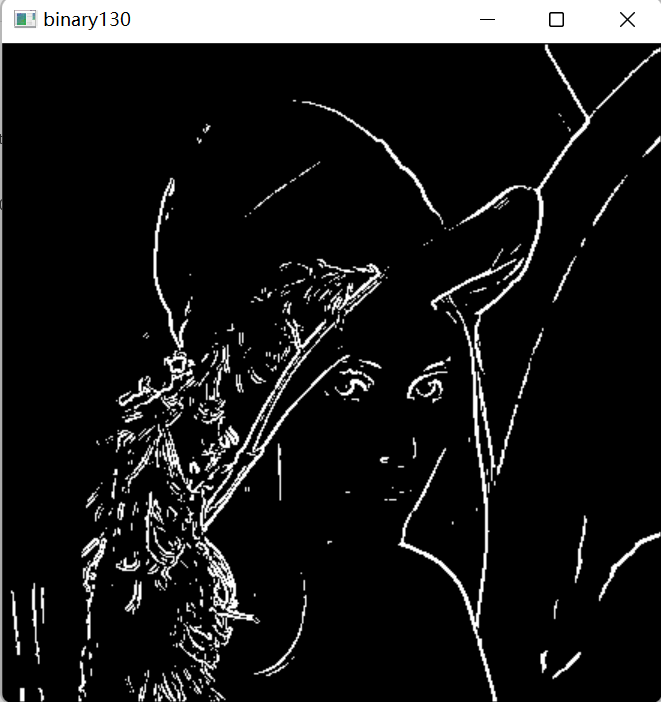
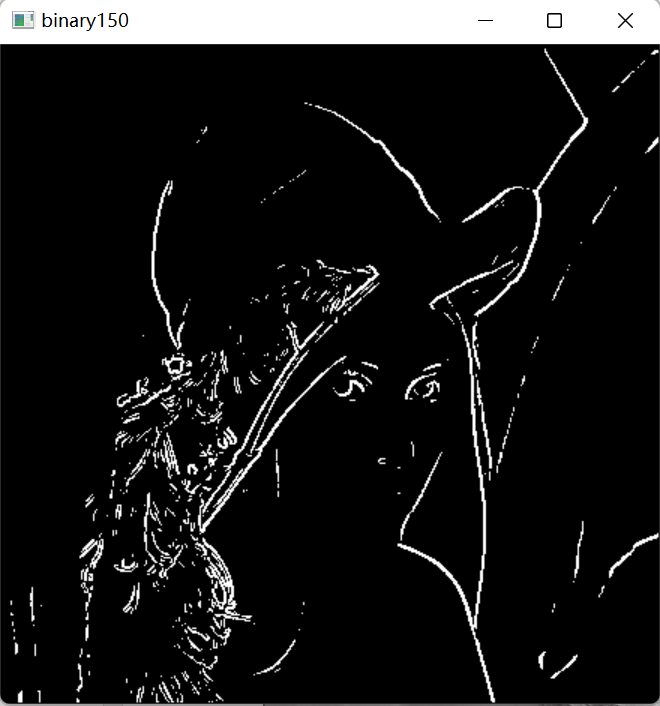
  

Figure 4: The binary images when threshold is 110, 130, 150

*Mention: The code should be run together with the two functions in the question 1 and the function in the question2.*