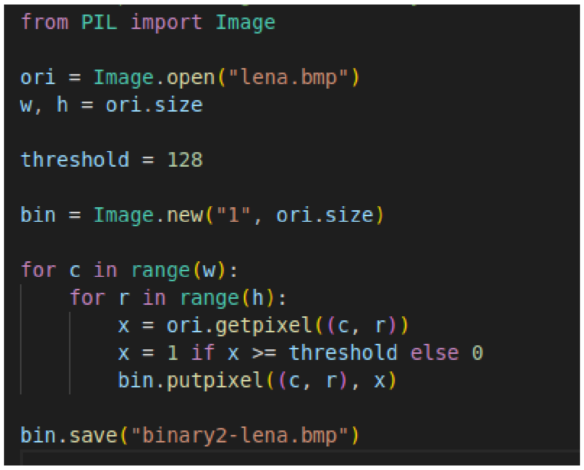
Computer Vision HW2 Report
蕭恩慈 / B07902095

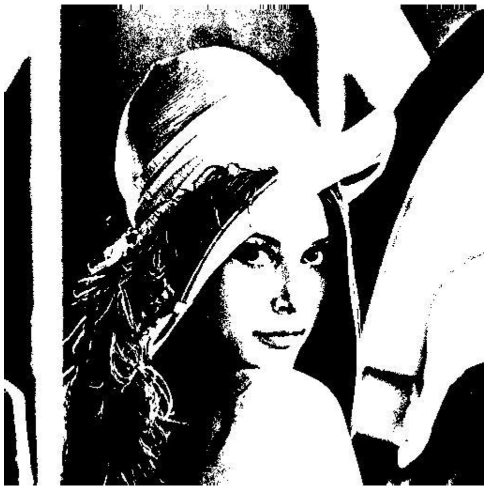
Part (a)
The task is to write a program that generates a binary image with threshold at 128.
For this task, I use Python as my program language. Using the PIL library, open the original image (lena.bmp) then get the width and height of the image and define the threshold value to 128. Then initialize a new image with the size of the original image, and "binary" format. "Chop" the original image into chunks of pixels, then turn its value (in accordance to threshold) pixel by pixel, before putting it into the new image (the image result). Lastly, save the image to the save file we wanted. For this part, I save it under the name *binary2-lena.bmp*

Final code:



```
from PIL import Image

ori = Image.open("lena.bmp")
w, h = ori.size

threshold = 128

bin = Image.new("1", ori.size)

for c in range(w):
    for r in range(h):
        x = ori.getpixel((c, r))
        x = 1 if x >= threshold else 0
        bin.putpixel((c, r), x)

bin.save("binary2-lena.bmp")
```

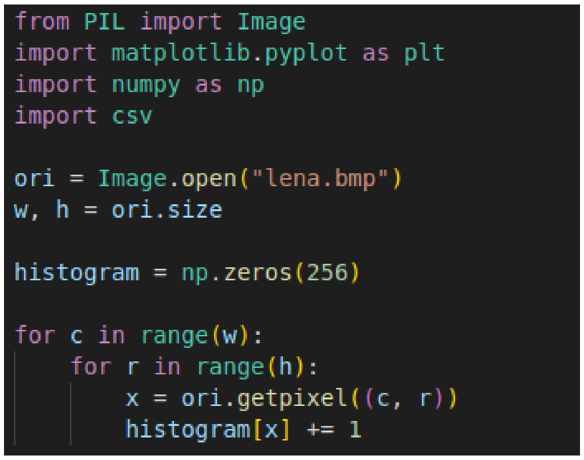
Result:



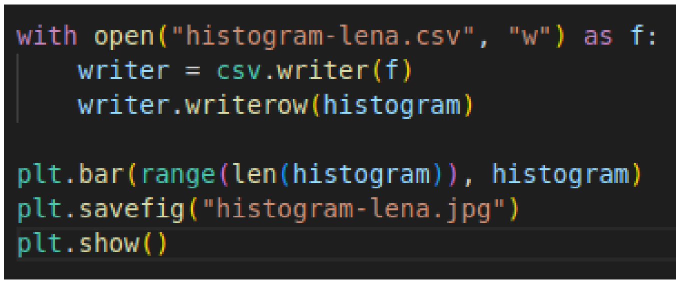
Note: At first I save the file as .jpg, but since the image result will be used later for part c (which I realize will not work if with jpg file type), I change it to .bmp
For further homework, I will save the result image as .bmp

Part (b)
The task is to write a program that makes a histogram based on an image.
For this task, I use Python as the program language. Using PIL, matplotlib.pyplot, numpy, and csv library, first open the original image (lena.bmp) and get the width and height from the image size, then initialize an array with zeros. After "chopping" the image into pixels, put it into the array, before saving it to a csv file. From the csv file, plot the histogram and save the result image. For this part, I save it under the name *histogram-lena.csv* for the csv file, and *histogram-lena.jpg* for the result image.

Final code:

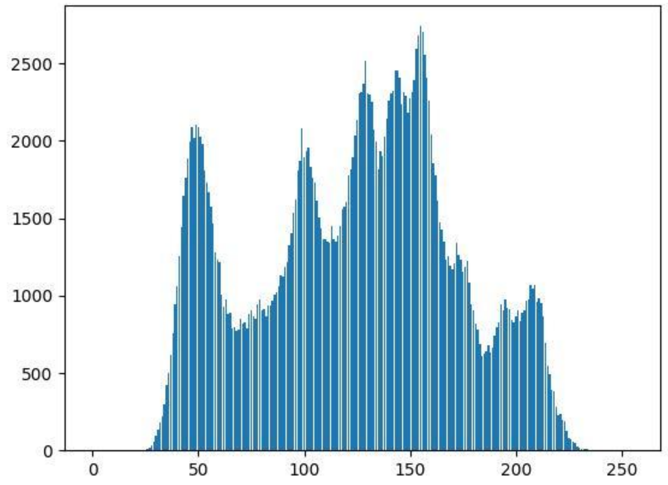


```
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import csv

ori = Image.open("lena.bmp")
w, h = ori.size

histogram = np.zeros(256)

for c in range(w):
    for r in range(h):
        x = ori.getpixel((c, r))
        histogram[x] += 1
```



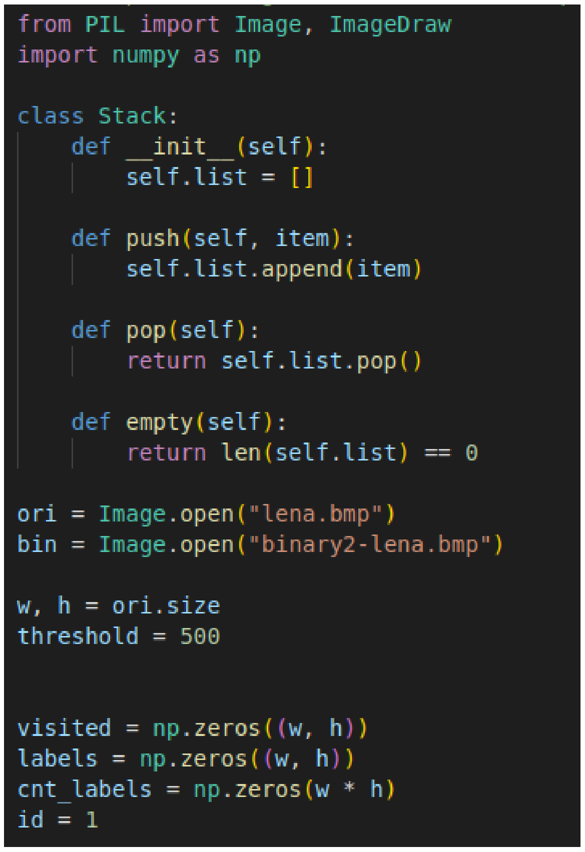
```
with open("histogram-lena.csv", "w") as f:
    writer = csv.writer(f)
    writer.writerow(histogram)

plt.bar(range(len(histogram)), histogram)
plt.savefig("histogram-lena.jpg")
plt.show()
```

Result:

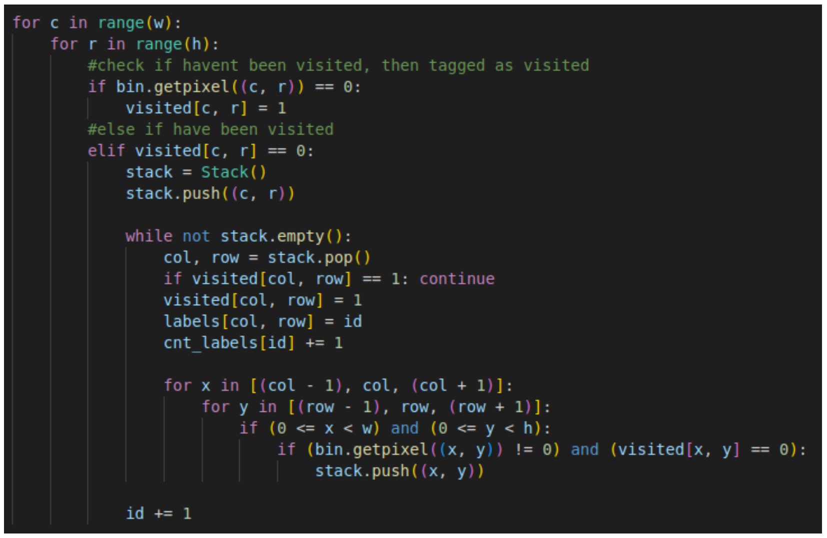


Part (c)
The task is to write a program that gives a determine regions with bounding box
For this task, I use Python as the program language using PIL and numpy library. To start, open the original image (lena.bmp) and get the width and height of the image, and define the threshold at 500. Also open the binary image from the result of part a (binary2-lena.bmp). Since so far my method is to process the image pixel by pixel, first I need to determine whether the "location" has been "visited" or not, I use 0 for visited, and 1 for unvisited as the indicator. For the unvisited, push the location into the created stack then check its "neighboring location" to determine if they are within the image range or not, also count how many pixels in the region label then record it.
To create the bounding box, look through each label then select only the region with at least 500 pixels.
After converting the image into RGB format, draw the line on the image then save the result image. (bounded-lena.jpg)

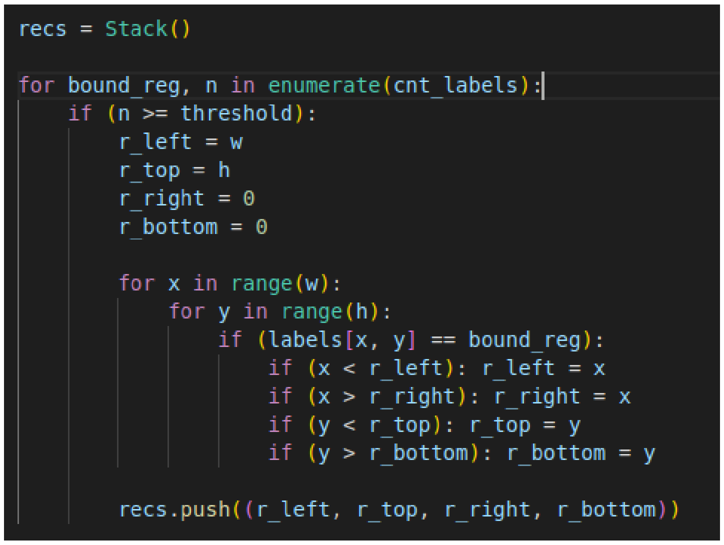
Final code:



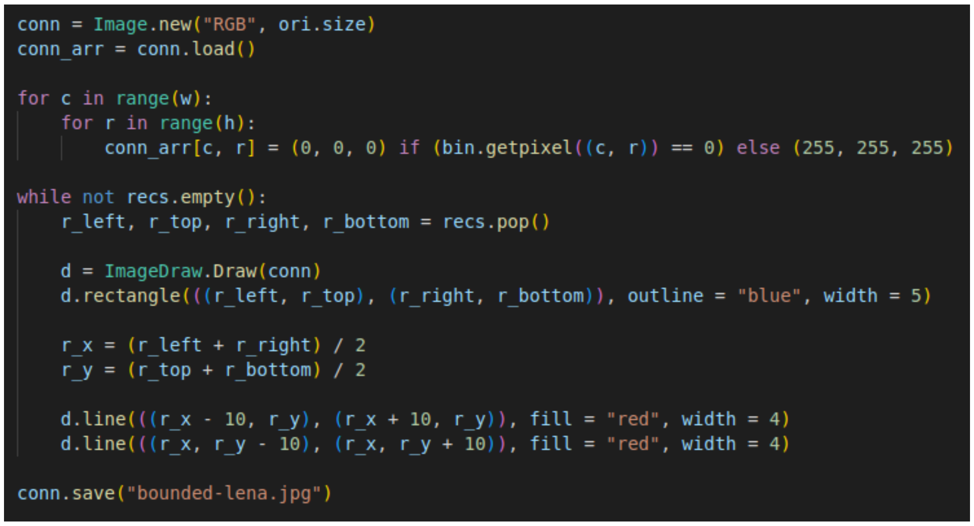
```python
from PIL import Image, ImageDraw
import numpy as np

class Stack:
    def __init__(self):
        self.list = []

    def push(self, item):
        self.list.append(item)

    def pop(self):
        return self.list.pop()

    def empty(self):
        return len(self.list) == 0

ori = Image.open("lena.bmp")
bin = Image.open("binary2-lena.bmp")

w, h = ori.size
threshold = 500


visited = np.zeros((w, h))
labels = np.zeros((w, h))
cnt_labels = np.zeros(w * h)
id = 1
```



```python
for c in range(w):
    for r in range(h):
        #check if havent been visited, then tagged as visited
        if bin.getpixel((c, r)) == 0:
            visited[c, r] = 1
        #else if have been visited
        elif visited[c, r] == 0:
            stack = Stack()
            stack.push((c, r))

            while not stack.empty():
                col, row = stack.pop()
                if visited[col, row] == 1: continue
                visited[col, row] = 1
                labels[col, row] = id
                cnt_labels[id] += 1

                for x in [(col - 1), col, (col + 1)]:
                    for y in [(row - 1), row, (row + 1)]:
                        if (0 <= x < w) and (0 <= y < h):
                            if (bin.getpixel((x, y)) != 0) and (visited[x, y] == 0):
                                stack.push((x, y))

            id += 1
```



```python
recs = Stack()

for bound_reg, n in enumerate(cnt_labels):
    if (n >= threshold):
        r_left = w
        r_top = h
        r_right = 0
        r_bottom = 0

        for x in range(w):
            for y in range(h):
                if (labels[x, y] == bound_reg):
                    if (x < r_left): r_left = x
                    if (x > r_right): r_right = x
                    if (y < r_top): r_top = y
                    if (y > r_bottom): r_bottom = y

        recs.push((r_left, r_top, r_right, r_bottom))
```



```python
conn = Image.new("RGB", ori.size)
conn_arr = conn.load()

for c in range(w):
    for r in range(h):
        conn_arr[c, r] = (0, 0, 0) if (bin.getpixel((c, r)) == 0) else (255, 255, 255)

while not recs.empty():
    r_left, r_top, r_right, r_bottom = recs.pop()

    d = ImageDraw.Draw(conn)
    d.rectangle(((r_left, r_top), (r_right, r_bottom)), outline = "blue", width = 5)

    r_x = (r_left + r_right) / 2
    r_y = (r_top + r_bottom) / 2

    d.line(((r_x - 10, r_y), (r_x + 10, r_y)), fill = "red", width = 4)
    d.line(((r_x, r_y - 10), (r_x, r_y + 10)), fill = "red", width = 4)

conn.save("bounded-lena.jpg")
```

Result: