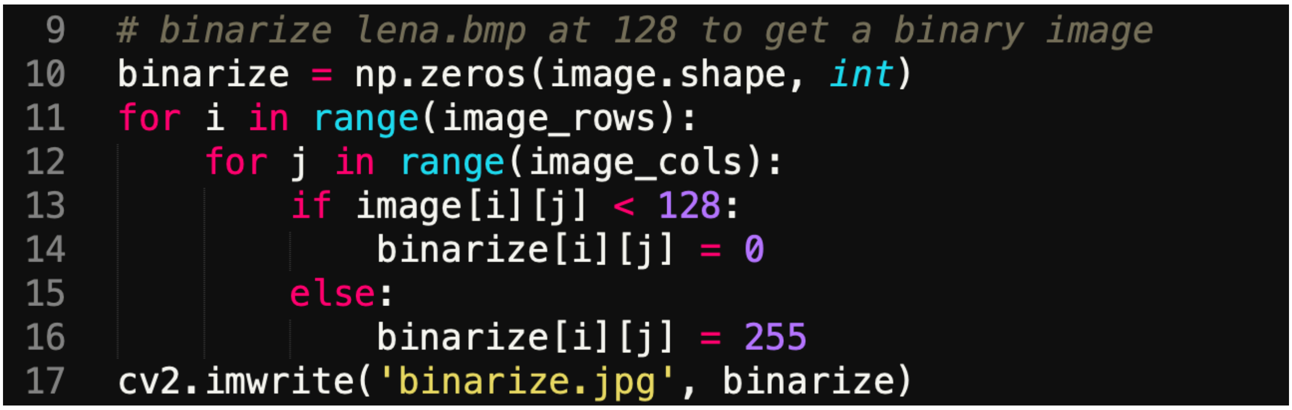
Computer Vision — Homework 2

B05902118 陳盈如

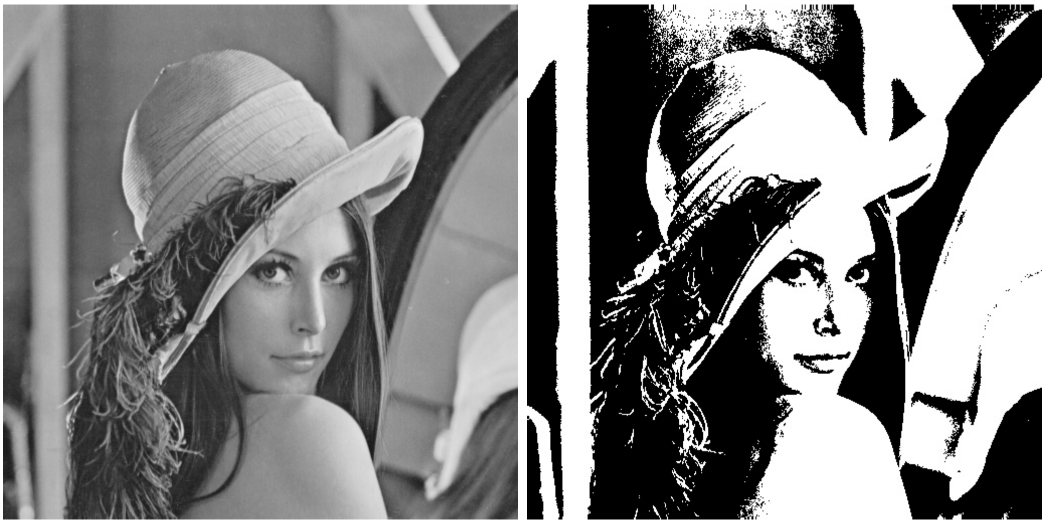
September 29, 2020

1 A binary image (threshold at 128)



```python
# binarize lena.bmp at 128 to get a binary image
binarize = np.zeros(image.shape, int)
for i in range(image_rows):
    for j in range(image_cols):
        if image[i][j] < 128:
            binarize[i][j] = 0
        else:
            binarize[i][j] = 255
cv2.imwrite('binarize.jpg', binarize)
```

Actually, the picture is already grayscale, so we don't need to transfer the color into grayscale again. Then, identify the value. If it is less than 128, change it to black (pixel = 0), else, change it to white (pixel =255).

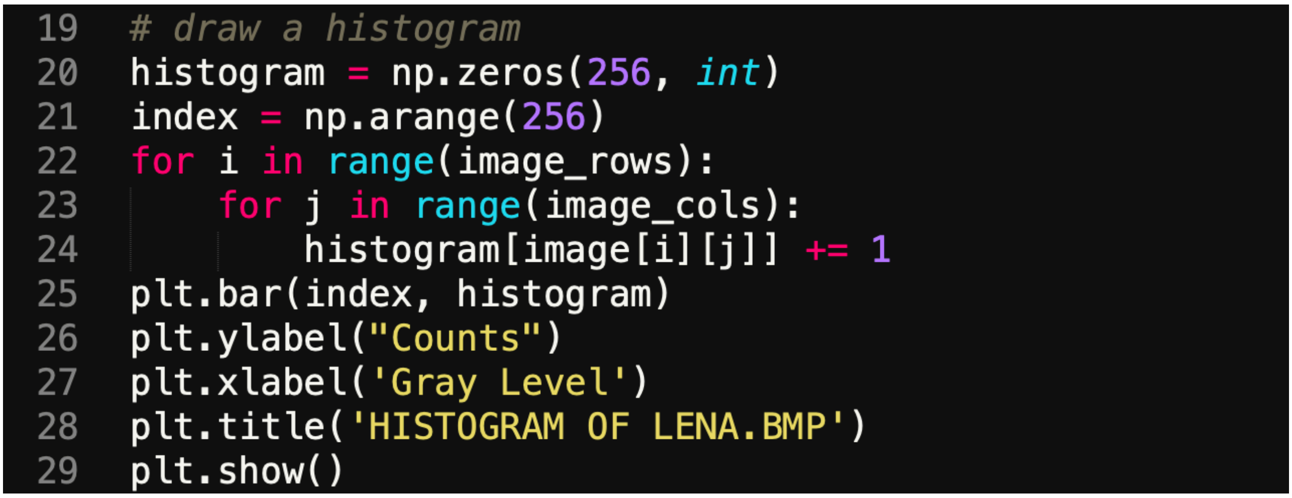


(a) Original

(b) Result

Figure 1: binarize at threshold 128

2 A histogram



```
# draw a histogram
histogram = np.zeros(256, int)
index = np.arange(256)
for i in range(image_rows):
    for j in range(image_cols):
        histogram[image[i][j]] += 1
plt.bar(index, histogram)
plt.ylabel("Counts")
plt.xlabel('Gray Level')
plt.title('HISTOGRAM OF LENA.BMP')
plt.show()
```

Create an array and make it size 256. This array represents the number of each gray value from 0 to 255 on *lena.bmp*. Check each pixel's gray value on *lena.bmp* and make that gray value's count +1.

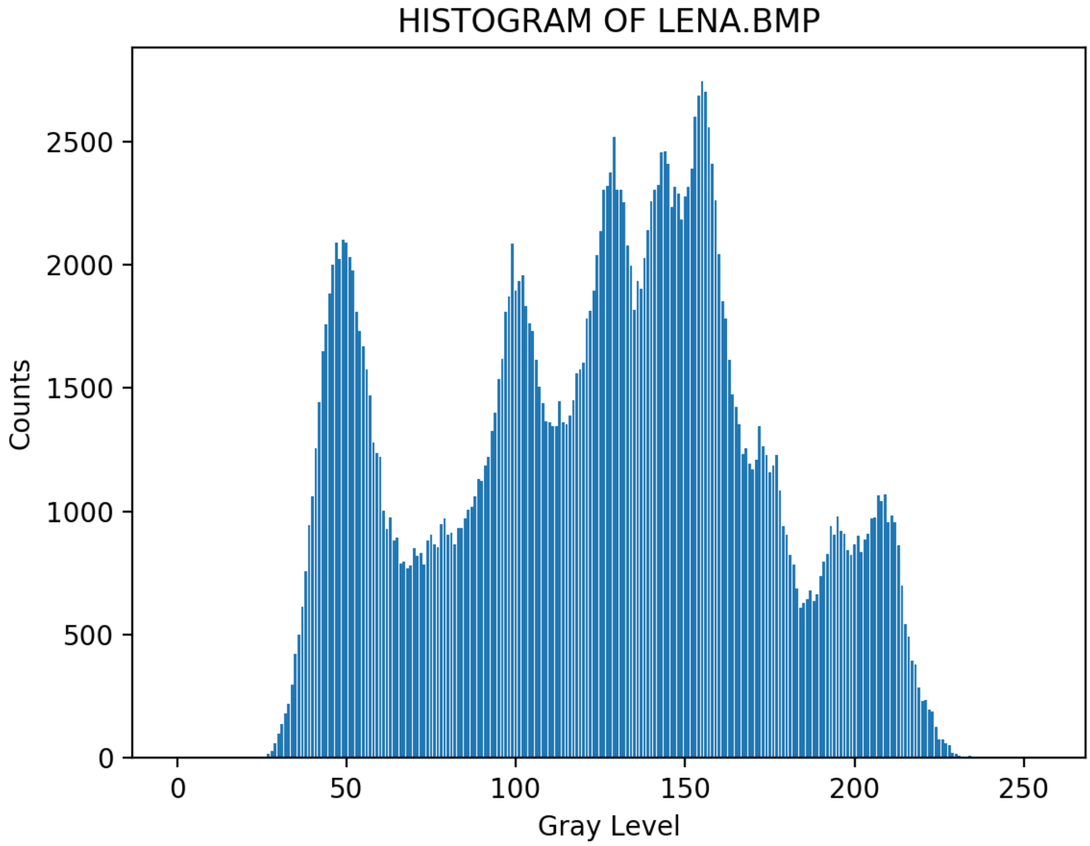
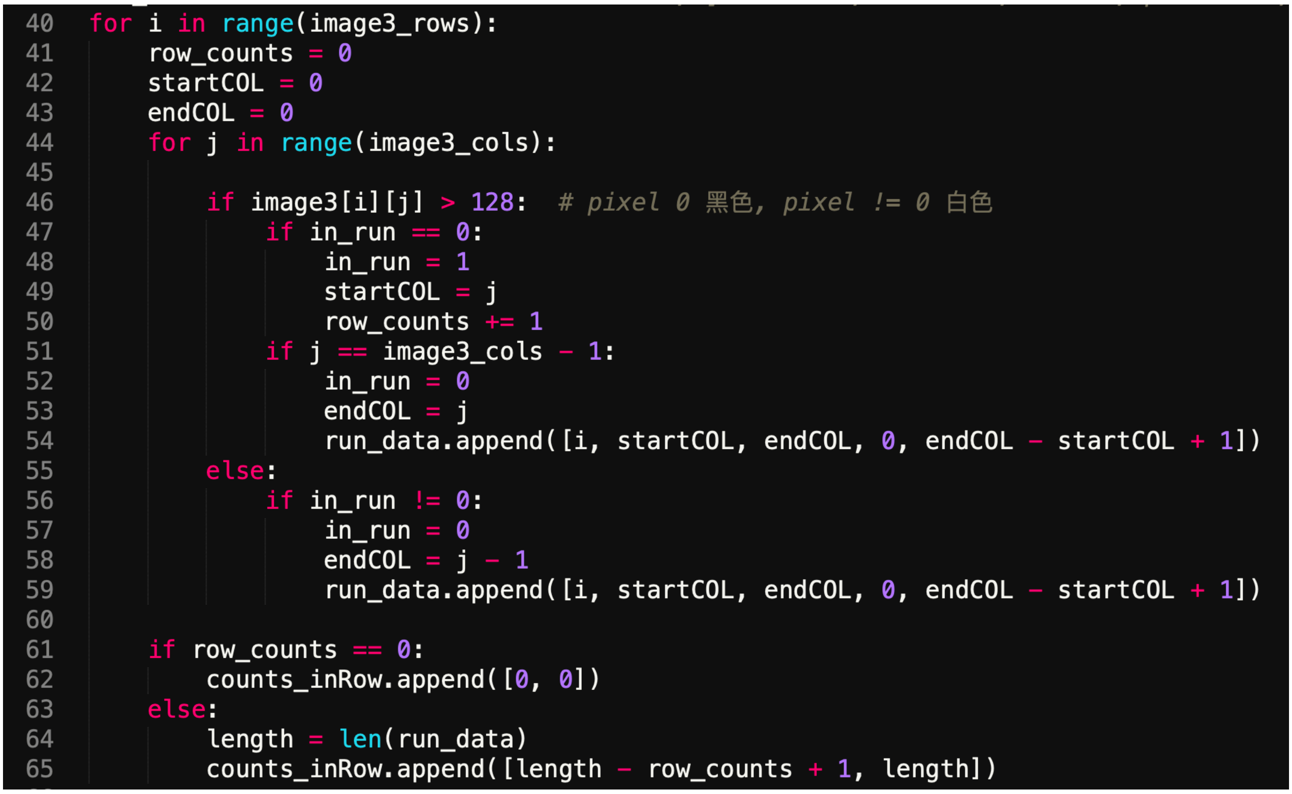


Figure 2: Histogram

3 Connected components (regions with + at centroid, bounding box)

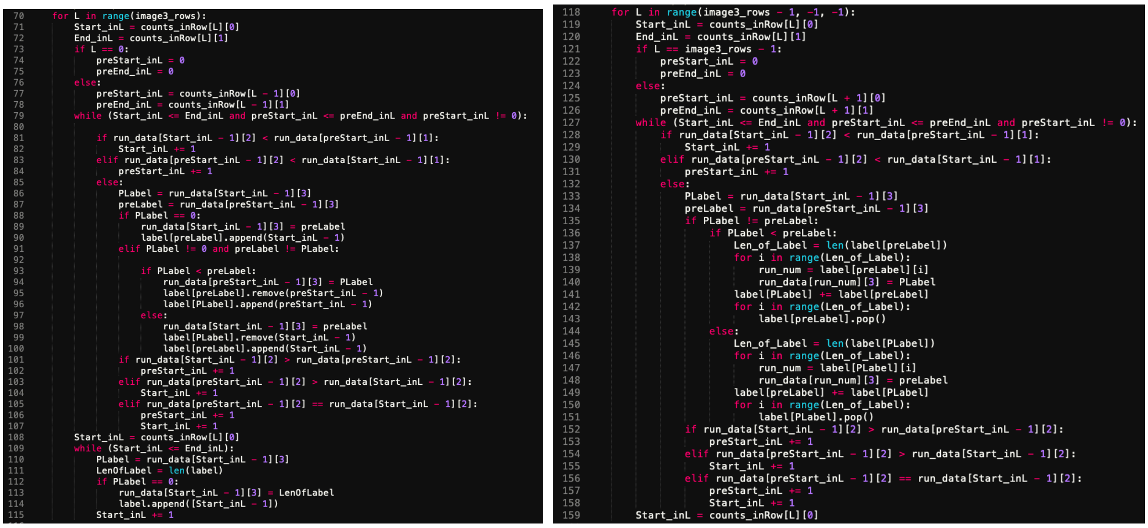


```python
for i in range(image3_rows):
    row_counts = 0
    startCOL = 0
    endCOL = 0
    for j in range(image3_cols):

        if image3[i][j] > 128:  # pixel 0 黑色, pixel != 0 白色
            if in_run == 0:
                in_run = 1
                startCOL = j
                row_counts += 1
            if j == image3_cols - 1:
                in_run = 0
                endCOL = j
                run_data.append([i, startCOL, endCOL, 0, endCOL - startCOL + 1])
        else:
            if in_run != 0:
                in_run = 0
                endCOL = j - 1
                run_data.append([i, startCOL, endCOL, 0, endCOL - startCOL + 1])

    if row_counts == 0:
        counts_inRow.append([0, 0])
    else:
        length = len(run_data)
        counts_inRow.append([length - row_counts + 1, length])
```

The first loop is responsible for recording two lists, *counts_inRow* and *run_data*.

counts_inRow: [0]: start of the run, [1]: end of the run

run_data: [0]: this run on which row, [1]: this run starting from which column, [2]: ending from which column, [3]: label, [4]: the number of pixels in this run



```python
for L in range(image3_rows):
    Start_inL = counts_inRow[L][0]
    End_inL = counts_inRow[L][1]
    if L == 0:
        preStart_inL = 0
        preEnd_inL = 0
    else:
        preStart_inL = counts_inRow[L - 1][0]
        preEnd_inL = counts_inRow[L - 1][1]
    while (Start_inL <= End_inL and preStart_inL <= preEnd_inL and preStart_inL != 0):

        if run_data[Start_inL - 1][2] < run_data[preStart_inL - 1][1]:
            Start_inL += 1
        elif run_data[preStart_inL - 1][2] < run_data[Start_inL - 1][1]:
            preStart_inL += 1
        else:
            PLabel = run_data[Start_inL - 1][3]
            preLabel = run_data[preStart_inL - 1][3]
            if PLabel == 0:
                run_data[Start_inL - 1][3] = preLabel
                label[preLabel].append(Start_inL - 1)
            elif PLabel != 0 and preLabel != PLabel:

                if PLabel < preLabel:
                    run_data[preStart_inL - 1][3] = PLabel
                    label[preLabel].remove(preStart_inL - 1)
                    label[PLabel].append(preStart_inL - 1)
                else:
                    run_data[Start_inL - 1][3] = preLabel
                    label[PLabel].remove(Start_inL - 1)
                    label[preLabel].append(Start_inL - 1)
            if run_data[Start_inL - 1][2] > run_data[preStart_inL - 1][2]:
                preStart_inL += 1
            elif run_data[preStart_inL - 1][2] > run_data[Start_inL - 1][2]:
                Start_inL += 1
            elif run_data[preStart_inL - 1][2] == run_data[Start_inL - 1][2]:
                preStart_inL += 1
                Start_inL += 1
    Start_inL = counts_inRow[L][0]
    while (Start_inL <= End_inL):
        PLabel = run_data[Start_inL - 1][3]
        LenOfLabel = len(label)
        if PLabel == 0:
            run_data[Start_inL - 1][3] = LenOfLabel
            label.append([Start_inL - 1])
        Start_inL += 1
```

(a) Top-down pass

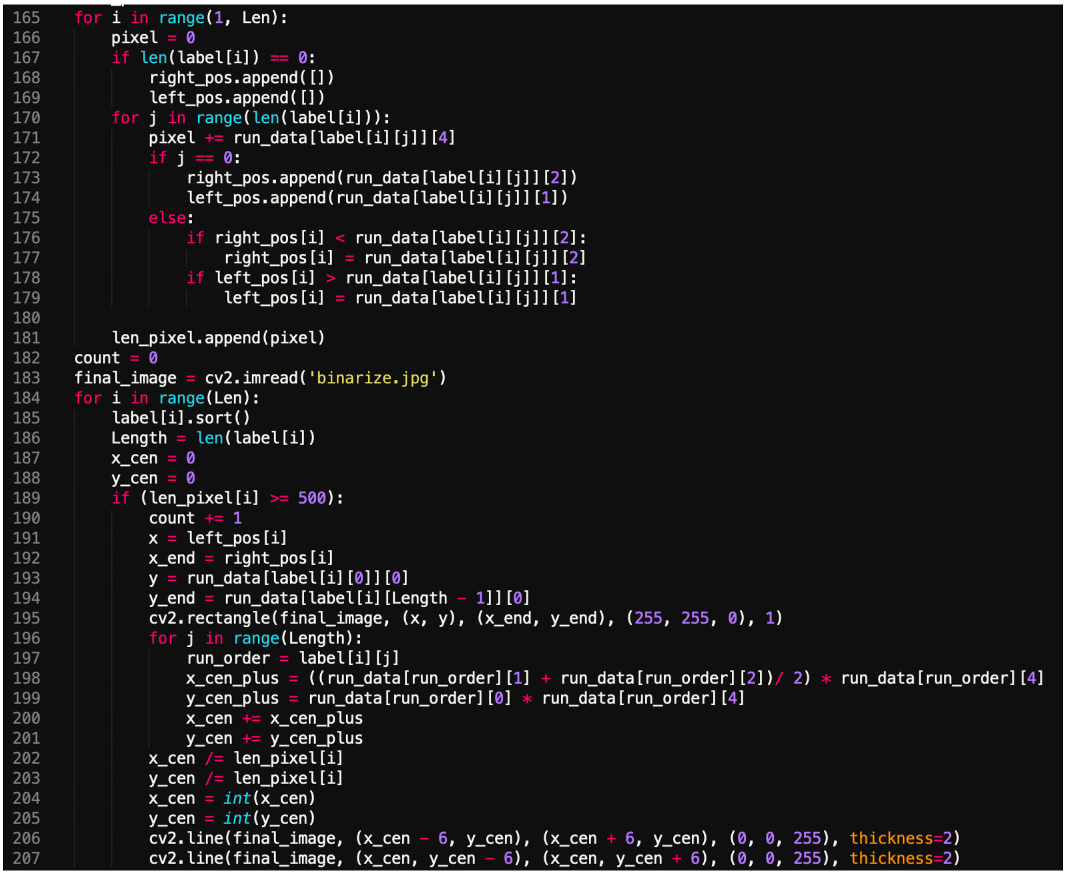
```python
for L in range(image3_rows - 1, -1, -1):
    Start_inL = counts_inRow[L][0]
    End_inL = counts_inRow[L][1]
    if L == image3_rows - 1:
        preStart_inL = 0
        preEnd_inL = 0
    else:
        preStart_inL = counts_inRow[L + 1][0]
        preEnd_inL = counts_inRow[L + 1][1]
    while (Start_inL <= End_inL and preStart_inL <= preEnd_inL and preStart_inL != 0):
        if run_data[Start_inL - 1][2] < run_data[preStart_inL - 1][1]:
            Start_inL += 1
        elif run_data[preStart_inL - 1][2] < run_data[Start_inL - 1][1]:
            preStart_inL += 1
        else:
            PLabel = run_data[Start_inL - 1][3]
            preLabel = run_data[preStart_inL - 1][3]
            if PLabel != preLabel:
                if PLabel < preLabel:
                    Len_of_Label = len(label[preLabel])
                    for i in range(Len_of_Label):
                        run_num = label[preLabel][i]
                        run_data[run_num][3] = PLabel
                    label[PLabel] += label[preLabel]
                    for i in range(Len_of_Label):
                        label[preLabel].pop()
                else:
                    Len_of_Label = len(label[PLabel])
                    for i in range(Len_of_Label):
                        run_num = label[PLabel][i]
                        run_data[run_num][3] = preLabel
                    label[preLabel] += label[PLabel]
                    for i in range(Len_of_Label):
                        label[PLabel].pop()
            if run_data[Start_inL - 1][2] > run_data[preStart_inL - 1][2]:
                preStart_inL += 1
            elif run_data[preStart_inL - 1][2] > run_data[Start_inL - 1][2]:
                Start_inL += 1
            elif run_data[preStart_inL - 1][2] == run_data[Start_inL - 1][2]:
                preStart_inL += 1
                Start_inL += 1
    Start_inL = counts_inRow[L][0]
```

(b) Bottom-up pass

Figure 3: Run Length Implementation

Implement the third method on PowerPoint – Run Length Implementation. Maintain *label* list at the same time when tracing top-down pass and bottom-up pass steps.

label: Its index represents label number, and content represents the orders of the runs, which are in this label. For example: *label=[[nothing], [0, 2, 3], [1]]*. It means run #0, run #2, run #3 have the same label, label#1, and run #1 has label#2.



```python
for i in range(1, Len):
    pixel = 0
    if len(label[i]) == 0:
        right_pos.append([])
        left_pos.append([])
    for j in range(len(label[i])):
        pixel += run_data[label[i][j]][4]
        if j == 0:
            right_pos.append(run_data[label[i][j]][2])
            left_pos.append(run_data[label[i][j]][1])
        else:
            if right_pos[i] < run_data[label[i][j]][2]:
                right_pos[i] = run_data[label[i][j]][2]
            if left_pos[i] > run_data[label[i][j]][1]:
                left_pos[i] = run_data[label[i][j]][1]

    len_pixel.append(pixel)
count = 0
final_image = cv2.imread('binarize.jpg')
for i in range(Len):
    label[i].sort()
    Length = len(label[i])
    x_cen = 0
    y_cen = 0
    if (len_pixel[i] >= 500):
        count += 1
        x = left_pos[i]
        x_end = right_pos[i]
        y = run_data[label[i][0]][0]
        y_end = run_data[label[i][Length - 1]][0]
        cv2.rectangle(final_image, (x, y), (x_end, y_end), (255, 255, 0), 1)
        for j in range(Length):
            run_order = label[i][j]
            x_cen_plus = ((run_data[run_order][1] + run_data[run_order][2])/ 2) * run_data[run_order][4]
            y_cen_plus = run_data[run_order][0] * run_data[run_order][4]
            x_cen += x_cen_plus
            y_cen += y_cen_plus
        x_cen /= len_pixel[i]
        y_cen /= len_pixel[i]
        x_cen = int(x_cen)
        y_cen = int(y_cen)
        cv2.line(final_image, (x_cen - 6, y_cen), (x_cen + 6, y_cen), (0, 0, 255), thickness=2)
        cv2.line(final_image, (x_cen, y_cen - 6), (x_cen, y_cen + 6), (0, 0, 255), thickness=2)
```

Threshold at 500, figure out the label whose area is larger than 500, use a bounding box to frame the smallest area of it, and mark the centroid of this area.

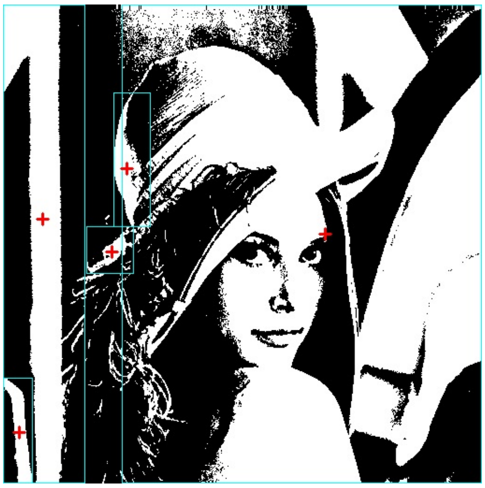


Figure 4: Result