CV HW2 Report

林義聖
B03902048

1 Introduction

I use *Python* as my programming language and *Pillow* as my Image Library. Also, I use *matplotlib* as the auxiliary program to draw the histogram from the data.

Figure 1: original lena picture

2 Binarized the Image

I use the following steps to binarize *lena.bmp* with threshold set to 128.

1. Read lena.bmp in as Python *list*.
2. For every pixels, compare its intensity to the threshold.
3. if its intensity is higher, draw it as white, else draw it as black.

Listing 1: Binarize the image

```
threshold = 127
black, white = 0, 255
for i in range(height * width):
    output_data[i] = white if pixel_data[i] > threshold else black
```

Figure 2: binarized lena picture

3 Draw the Histogram

I use the following steps to calculate the histogram of *lena.bmp*.

1. Read lena.bmp in as Python *list*.
2. For every pixels, accumulate its intensity.
3. Call *matplotlib.pyplot.bar* to draw the histogram.

Listing 2: Accumulating the intensity

```
1 histo_data = [0] * 255
2 for i in range(height * width):
3     histo_data[ pixel_data[i] ] += 1
```

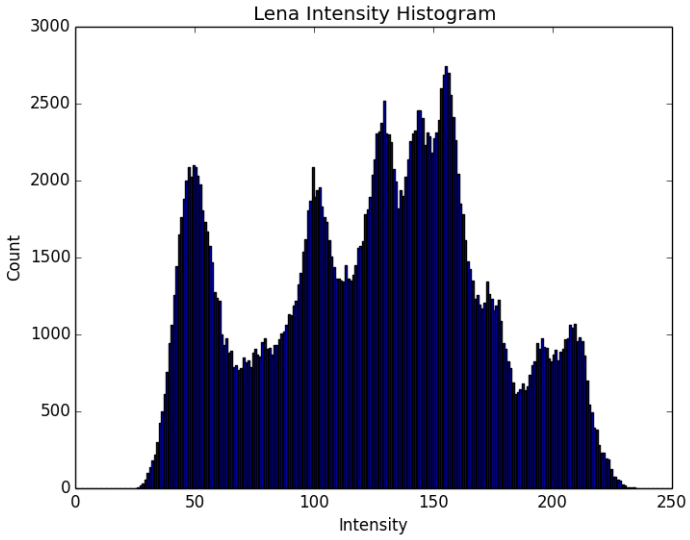

Figure 3: histogram of lena picture

4 Finding the Connected Components

I use the following steps to find the connected components and draw bounding box on the picture.

1. Take lena-binarized.bmp as input image, and read it in as Python *list*.
2. For every pixels (actually, only white pixel), give it a label first, then use *BFS* to expand all connected pixels around it (8-connected).
3. Count the numbers of pixels of each labels.
4. Draw the bounding box (using function *PIL.ImageDraw.draw.rectangle* to draw rectangles on output picture) around the group of connected components with size larger than 500.

Listing 3: Finding connected components

```
# 8−connected neighbor
nei_offset = [(−1,−1),(0,−1),(1,−1),(−1,0),(0,1),(−1,1),(1,0),(1,1)]
for y in range(height):
    for x in range(width):
        if label_map[y*width+x] == 0 and data_seq[y*width+x] == 255:
            curr_label = get_label() # assign a new label
            curr_color = data_seq[y*width+x]

            state_q = []
            label_map[y * width + x] = −1 # assign a temporary label
            state_q.append((x,y)) # push into state queue

            while len(state_q) != 0:
                tmp_x, tmp_y = state_q.pop(0) # pop out from queue and assign label
                label_map[tmp_y * width + tmp_x] = curr_label

                for offset in nei_offset:
                    off_x = tmp_x + offset[0]
                    off_y = tmp_y + offset[1]
                    if 0 <= off_x < width and 0 <= off_y < height and \
                        label_map[off_y*width+off_x] == 0 and \
                        data_seq[off_y*width+off_x] == curr_color:
                        label_map[off_y * width + off_x] = −1
                        state_q.append((off_x,off_y)) # push into state queue
```

Listing 4: Drawing the bounding box

```
for i in range(1, len(count_label)):
    if count_label[i] >= THRESHOLD:
        # finding the boundary of the area of this label
        topVal = min(label_array[i], key=lambda x: x[1])[1]
        bottomVal = max(label_array[i], key=lambda x: x[1])[1]
        leftVal = min(label_array[i], key=lambda x: x[0])[0]
        rightVal = max(label_array[i], key=lambda x: x[0])[0]

```

```
        # draw bounding box and centroid with ImageDraw class
        sum_x, sum_y = 0, 0
        for j in range(len(label_array[i])):
            sum_x += label_array[i][j][0]
            sum_y += label_array[i][j][1]
        avg_x = int(sum_x / len(label_array[i]))
        avg_y = int(sum_y / len(label_array[i]))
        rcolor = (255, 0, 0)
        draw.rectangle([(leftVal,topVal),(rightVal,bottomVal)], outline=rcolor)
        draw.line([(avg_x-5,avg_y-5),(avg_x+5,avg_y+5)], fill=rcolor)
        draw.line([(avg_x+5,avg_y-5),(avg_x-5,avg_y+5)], fill=rcolor)
```

Figure 4: lena picture with bounding box

5 How to Use

There are three programs in my HW2.

1. *binarize-128.py*
2. *calc-histogram.py*
3. *find-connected-components.py*

You just need to enter command in this format: "*program [input image name] [output image name]*" to use it. For example, `./binarize-128.py lena.bmp output.bmp`.