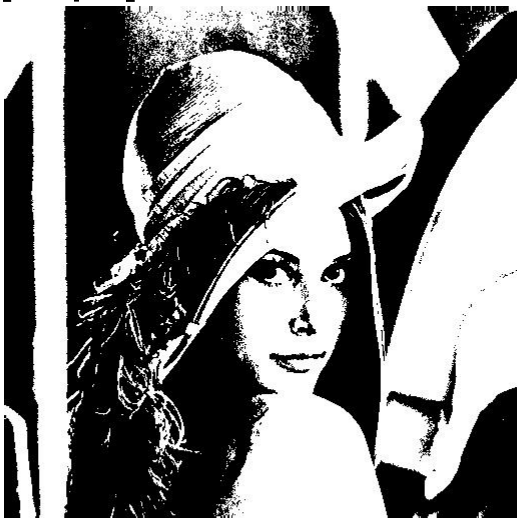
Computer Vision - Assignment 2

R09922A04 資工所人工智慧組 黃品硯

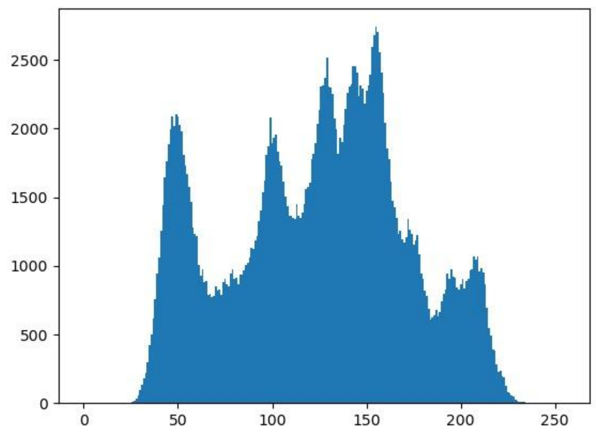
(a) a binary image (threshold at 128)

Iteratively check every pixel, if the pixel's value is greater or equal to 128, then set it's value to 255 otherwise set it to 0.	**[Output]** 

[Code]

```
for y in range(height):
    for x in range(width):
        if img[y, x] < 128:
            img[y, x] = 0
        else:
            img[y, x] = 255
```

(b) a histogram

1. Iteratively check every pixel's value and add one to the corresponding element in the list, pixel_count. 2. Plot the pixel_count.	**[Output]** 

[Code]

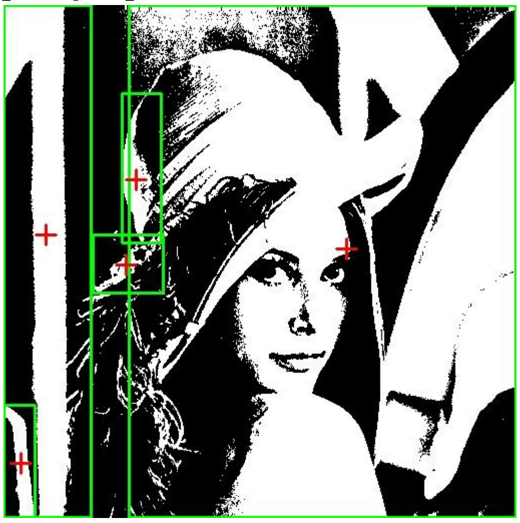
```
pixel_count = [0] * 256
for y in range(height):
    for x in range(width):
        pixel_count[img[y, x]] += 1

plt.bar(range(256), pixel_count, 1)
```

(c) connected components (regions with + at centroid, bounding box)

I'm using an 8-connected classical algorithm.

[Output]



[Code]

```
# first top-down
for y in range(height):
    for x in range(width):
        if img[y, x] == 255:
            labels[(y, x)] = min(find_neighbors(labels, y, x))

# build equivalence table by UnionFind
union_find = UnionFind()
for y in range(height):
    for x in range(width):
        if img[y, x] == 255:
            neighbors = find_neighbors(labels, y, x)
            if len(neighbors) > 1:
                sorted_neighbors = sorted(neighbors)
                min_neighbor = sorted_neighbors[0]
                for neighbor in sorted_neighbors[1:]:
                    union_find.union(min_neighbor, neighbor)
```

```python
label2pos = {}
for pos, label in labels.items():
    root_label = union_find.find(label)

    if root_label in label2pos:
        label2pos[root_label].append(pos)
    else:
        label2pos[root_label] = [pos]

color_img = cv2.merge([img, img, img])

# draw cross and bounding boxes
for label, positions in label2pos.items():
    if len(positions) >= 500:
        y, x = get_centroid(positions)
        draw_cross(color_img, int(y), int(x))
        draw_bounding_box(color_img, positions)
```