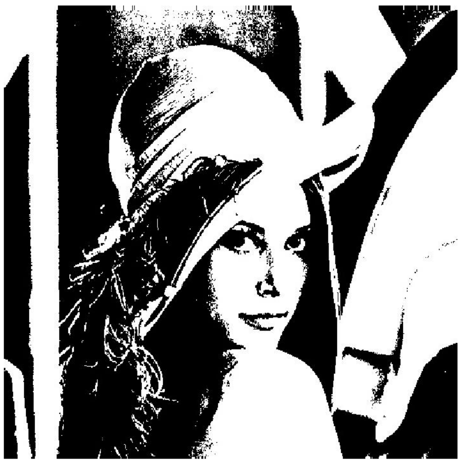
Computer Vision Homework 2

R08922079 資工所一 洪浩翔

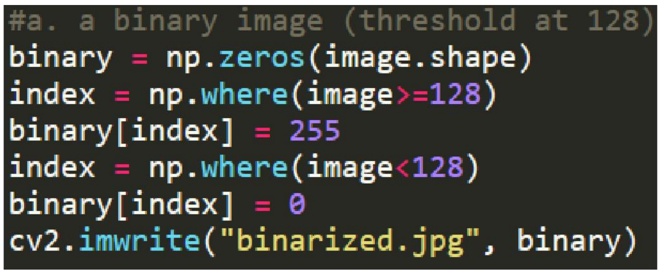
Part 1

(a) A binary image (threshold at 128)

Original Lena



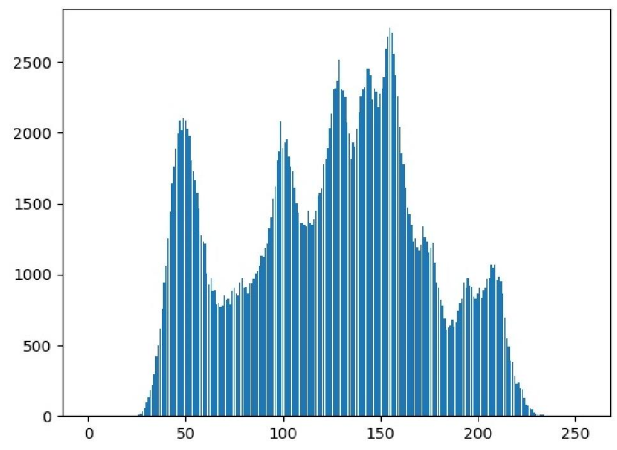
Binarized Lena



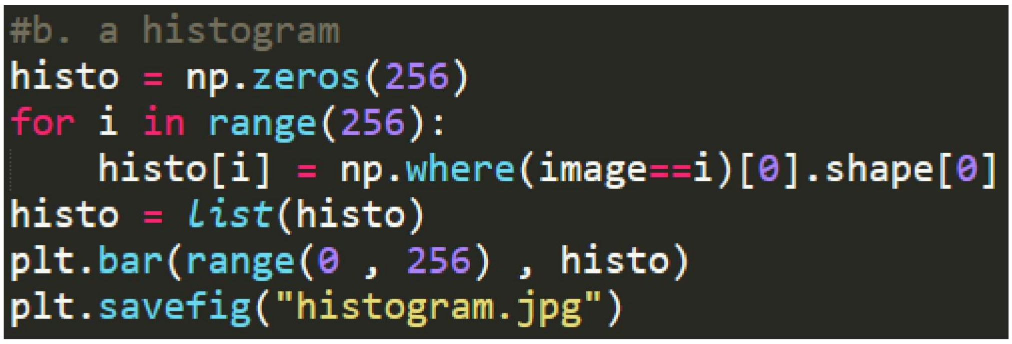
```
#a. a binary image (threshold at 128)
binary = np.zeros(image.shape)
index = np.where(image>=128)
binary[index] = 255
index = np.where(image<128)
binary[index] = 0
cv2.imwrite("binarized.jpg", binary)
```

Main code fragment

(b) A histogram



Histogram of Lena

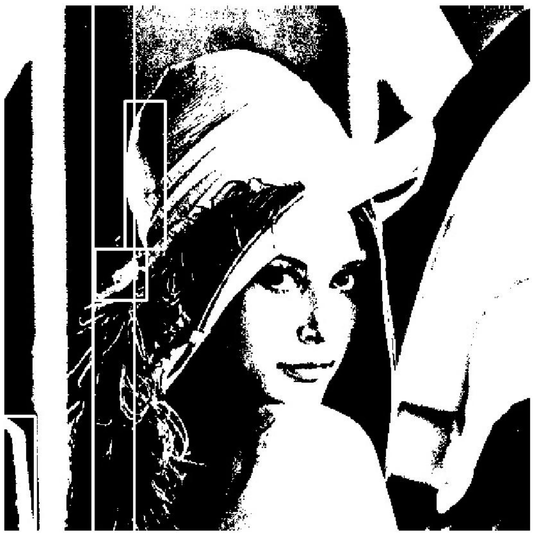


```
#b. a histogram
histo = np.zeros(256)
for i in range(256):
    histo[i] = np.where(image==i)[0].shape[0]
histo = list(histo)
plt.bar(range(0 , 256) , histo)
plt.savefig("histogram.jpg")
```

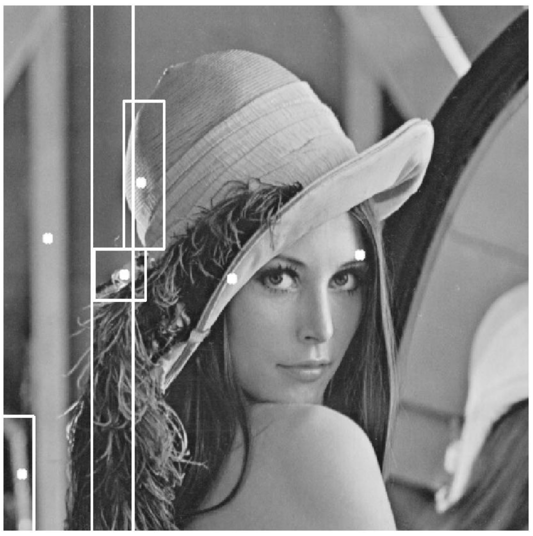
Main code fragment

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

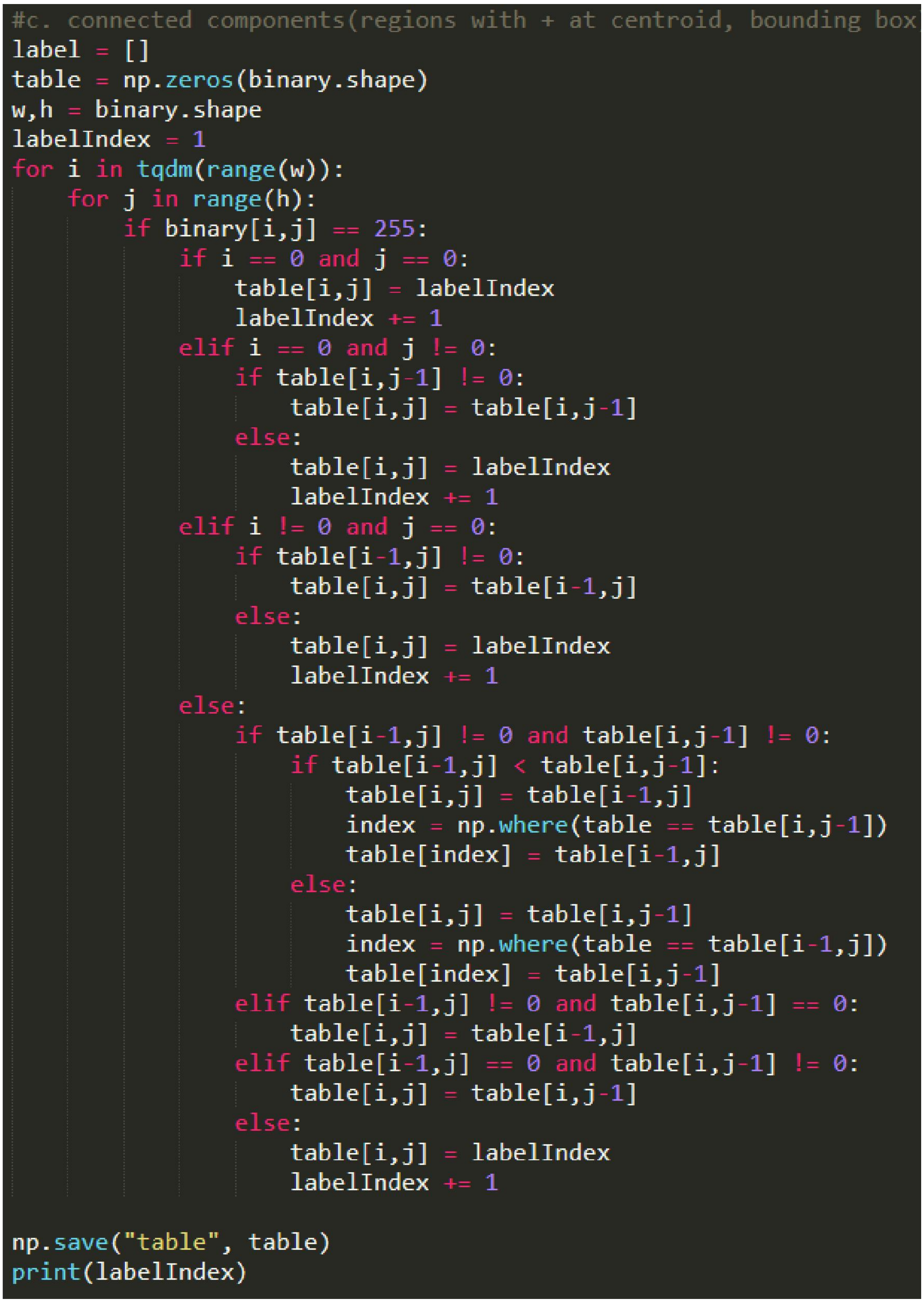
Mainly implement with classical connected components labeling algorithm, but I replace equivalent labels in runtime instead of at the end. Thus there is no need to remember label equivalences. Result is created with 4-connected.



Connected components on binarized Lena



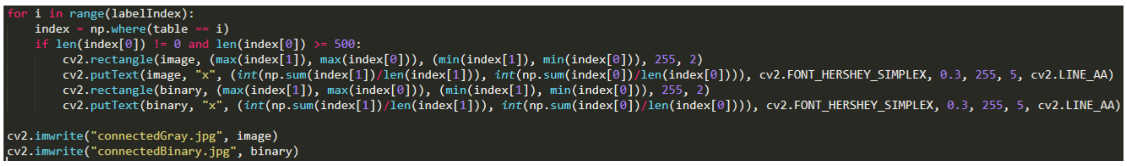
Connected components on grayscale Lena



```python
#c. connected components(regions with + at centroid, bounding box
label = []
table = np.zeros(binary.shape)
w,h = binary.shape
labelIndex = 1
for i in tqdm(range(w)):
    for j in range(h):
        if binary[i,j] == 255:
            if i == 0 and j == 0:
                table[i,j] = labelIndex
                labelIndex += 1
            elif i == 0 and j != 0:
                if table[i,j-1] != 0:
                    table[i,j] = table[i,j-1]
                else:
                    table[i,j] = labelIndex
                    labelIndex += 1
            elif i != 0 and j == 0:
                if table[i-1,j] != 0:
                    table[i,j] = table[i-1,j]
                else:
                    table[i,j] = labelIndex
                    labelIndex += 1
            else:
                if table[i-1,j] != 0 and table[i,j-1] != 0:
                    if table[i-1,j] < table[i,j-1]:
                        table[i,j] = table[i-1,j]
                        index = np.where(table == table[i,j-1])
                        table[index] = table[i-1,j]
                    else:
                        table[i,j] = table[i,j-1]
                        index = np.where(table == table[i-1,j])
                        table[index] = table[i,j-1]
                elif table[i-1,j] != 0 and table[i,j-1] == 0:
                    table[i,j] = table[i-1,j]
                elif table[i-1,j] == 0 and table[i,j-1] != 0:
                    table[i,j] = table[i,j-1]
                else:
                    table[i,j] = labelIndex
                    labelIndex += 1

np.save("table", table)
print(labelIndex)
```

Main code fragment for connected component computing



```python
for i in range(labelIndex):
    index = np.where(table == i)
    if len(index[0]) != 0 and len(index[0]) >= 500:
        cv2.rectangle(image, (max(index[1]), max(index[0])), (min(index[1]), min(index[0])), 255, 2)
        cv2.putText(image, "x", (int(np.sum(index[1])/len(index[1])), int(np.sum(index[0])/len(index[0]))), cv2.FONT_HERSHEY_SIMPLEX, 0.3, 255, 5, cv2.LINE_AA)
        cv2.rectangle(binary, (max(index[1]), max(index[0])), (min(index[1]), min(index[0])), 255, 2)
        cv2.putText(binary, "x", (int(np.sum(index[1])/len(index[1])), int(np.sum(index[0])/len(index[0]))), cv2.FONT_HERSHEY_SIMPLEX, 0.3, 255, 5, cv2.LINE_AA)

cv2.imwrite("connectedGray.jpg", image)
cv2.imwrite("connectedBinary.jpg", binary)
```

Main code fragment for bounding box and centroid computing