
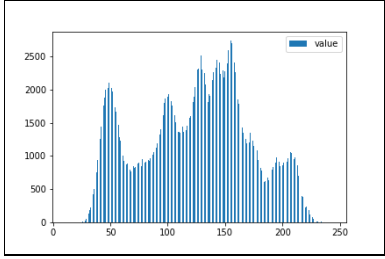



Ans

| (a) a binary image<br>(threshold at 128)  | (b) a histogram   | (c) connected components<br>(regions with • at centroid, bounding box)              |
|---|---|---|
|  |  |  |

說明

1. 圖片檔位於資料夾中
2. 本次作業使用 python，編譯器採用 spyder
3. Source Code 說明如下
4. 本次採用 8-connected components，並將長方形中心標記為實心圓

## Part (a)

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed Sep 11 19:33:55 2019
4
5  @author: vincent黃國郡
6  """
7  from PIL import Image
8  import numpy as np
9  import pandas as pd
10 import cv2
11
12 lena_pic = Image.open("lena.bmp")
13 original_lena_array = np.array(lena_pic)
14
15 # (a) a binary image (threshold at 128)
16 binarize_lena_array = original_lena_array.copy()
17 for ii in range(len(original_lena_array)):
18     for jj in range(len(original_lena_array[ii])):
19         if (original_lena_array[ii][jj] >= 128):
20             binarize_lena_array[ii][jj] = 255
21         else:
22             binarize_lena_array[ii][jj] = 0
23 Image.fromarray(binarize_lena_array.astype(np.uint8)).save('binarize_lena.bmp')
24

```

匯入各種 library

讀進來的 picture 存成矩陣形式

複製一個 binarize 矩陣，準備進行二級化處理

複製一個 binarize 矩陣，準備進行二級化處理

如果該 pixel 大於等於 128 則將該 pixel 調整為 255 反之設為 0

將 binarize\_lena\_array 設為 uint8 格式存成.bmp 檔

## Part (b)

```

25 # (b) a histogram
26 histogram = np.zeros([256])
27 for r in original_lena_array:
28     for pixel in r:
29         histogram[pixel] += 1
30
31 df = pd.DataFrame(histogram, columns=['value'])
32 df.to_csv('histogram.csv')
33 ax = df.plot.bar(y='value', rot=0)
34 ticks = ax.xaxis.get_ticklocs()
35 ticklabels = [l.get_text() for l in ax.xaxis.get_ticklabels()]
36 ax.xaxis.set_ticks(ticks[:50])
37 ax.xaxis.set_ticklabels(ticklabels[:50])
38 fig = ax.get_figure()
39 fig.savefig("histogram.png")

```

創一個要畫 histogram 統計數量的矩陣，並將  
每一個 pixel 的數量記錄起來，將之存進去

設定 dataframe，容易寫入 csv  
並且簡單明瞭

調整 x 軸標籤間距，並將畫完的  
Bar 圖存入圖片檔中

## Part (c)

```

41 # (c) connected components
42 class Stack:
43     def __init__(self):
44         self.list = []
45
46     def push(self, item):
47         self.list.append(item)
48
49     def pop(self):
50         return self.list.pop()
51
52     def isEmpty(self):
53         return len(self.list) == 0
54
55 thresholdRegionPixels = 500
56 height,width=original_lena_array.shape
57 visited = np.zeros([height,width])
58 labeledImageArray = np.zeros([height,width])
59 labelId=1
60 numberOfLabelDict={}
61 numberOfCurrentLabel=0

```

定義一個 stack  
可以方便等一下做位置之儲存動作

定義 500 門檻，以便挑出適合的長方形  
紀錄圖片長寬

初始化每一個像素等一下跑迴圈時，各自代表的 label

```

62 for ii in range(len(original_lena_array)):
63     for jj in range(len(original_lena_array[ii])):
64         if binarize_lena_array[ii, jj] == 0:
65             visited[ii, jj] = 1
66         elif visited[ii, jj] == 0:
67             stack = Stack()
68             stack.push((ii, jj))
69             while not stack.isEmpty():
70                 xx, yy = stack.pop()
71                 if visited[xx, yy] == 1:
72                     continue
73                 visited[xx, yy] = 1
74                 labeledImageArray[xx,yy]=labelId
75                 numberOfCurrentLabel+=1
76                 for uu in [xx-1,xx,xx+1]:
77                     for vv in [yy-1,yy,yy+1]:
78                         if(0<=uu<height)and(0<=vv<width):
79                             if (visited[uu,vv]==0)and (binarize_lena_array[uu,vv]!=0):
80                                 stack.push((uu,vv))
81             if (numberOfCurrentLabel>=thresholdRegionPixels):
82                 numberOfLabelDict[labelId]=numberOfCurrentLabel
83             numberOfCurrentLabel=0
84             labelId+=1

```

跑每一張原始的圖片，如果圖片中 pixel 值為 0 則直接將該 pixel 設定為拜訪過

如果不為 0，又沒拜訪過，則將該位置丟進去 stack 處理

如果 stack 裡還有東西，則依序 pop 出來以先進後出的方式

當每處理一個 stack 中元件都去掃描九宮格，並判斷是否有相連，如果有 pixel 像素不等於 0 並且或沒拜過再丟進去 stack 處理，等一次 stack 都空了表示整張圖該 label 相連的也結束，並進入下一回合

取出數量大於 500 的 label 並將之存入 numberOfLabelDict 裡面以利畫圖

```

89 rectangles={}
90 for key in numberOfLabelDict:
91     currentKeyX,currentKeyY=np.where(labeledImageArray == key)
92     point1=(min(currentKeyY),min(currentKeyX))
93     point2=(max(currentKeyY),max(currentKeyX))
94     middlePointY=int(average(currentKeyY))
95     middlePointX=int(average(currentKeyX))
96     middlePoint=(middlePointY,middlePointX)
97     rectangles[key]=[point1,point2,middlePoint]
98
99 connected_lena_array = np.zeros((height,width,3))
100 connected_lena_array[:, :,0]=binarize_lena_array
101 connected_lena_array[:, :,1]=binarize_lena_array
102 connected_lena_array[:, :,2]=binarize_lena_array
103
104 for points in rectangles:
105     cv2.rectangle(connected_lena_array, rectangles[points][0], rectangles[points][1], (255,0, 0), 2)
106     cv2.circle(connected_lena_array,rectangles[points][2], 8, (255, 0, 0), thickness=-1)
107 Image.fromarray(connected_lena_array.astype(np.uint8)).save('connected_lena.bmp')

```

掃描每一個要畫長方形 label 的位置，並將每個相同 label 的 pixel 位置取平均，算出重心

將單通道轉為三通道

畫長方形

畫實心圓

將最後的相連矩陣存入 connected\_lena.bmp