

## 電腦視覺 (一) Homework2

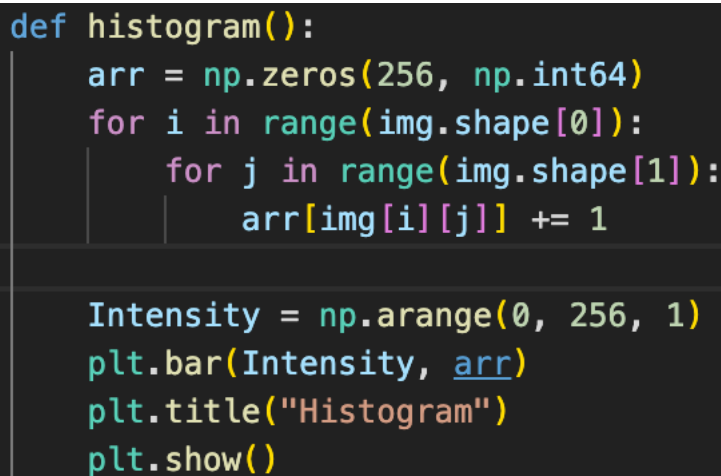
R11521701 程懷恩

### 1. a binary image (threshold at 128):

```
def binarize():  
    for i in range(0, img.shape[0]):  
        for j in range(0, img.shape[1]):  
            #print(img[i][j]) checker  
            if(img[i][j]>128):  
                img[i][j]=255  
            else:  
                img[i][j]=0  
    cv2.imwrite('binarized.bmp',img)
```

程式碼：以 128 為閾值過濾元素，大於 128 則改為 255；反之則改為 0。

### 2. a histogram :

```
def histogram():  
    arr = np.zeros(256, np.int64)  
    for i in range(img.shape[0]):  
        for j in range(img.shape[1]):  
            arr += 1  
  
    Intensity = np.arange(0, 256, 1)  
    plt.bar(Intensity, arr)  
    plt.title("Histogram")  
    plt.show()
```

程式碼：避免元素內無法存超過 255 的值，新陣列宣告 int64。讓 arr 陣列根

據讀到的值，讓相對應的陣列空間計數。(註：先執行 histogram 函式避免

受 binarize 影響)

### 3. connected components(regions with + at centroid, bounding box) :

前言：因為把功能都分段寫出來並組裝在一起後，發現運行時間實在太長，但又來不及做優化，若因為程式執行時間過長造成助教困擾真的很不好意思。

1. 先建立一個 image 物件，給予相應的成員，如：

當前座標(row, col)與當前 label 記數。

2. 定義 label 的相應功能，以 push 及 back 更改當前執行座標位置。

```
def push(self): ...
def back(self): ...
def label(self): ...
def label_item(self): #if recursive -> stack overflow, so using iteratively algorithm
    while(True):
        if(self.current_row == (self.arr.shape[0]-1) and self.current_col == (self.arr.shape[1]-1)):
            if(self.arr[self.current_row][self.current_col] == 0): #若最後一格為黑
                break
            else:
                self.label()
                break
        elif( self.arr[self.current_row][self.current_col] != 0): #如果pixel內為白
            self.label()
            self.push()
        else :
            self.push()
```

3. 定義置換標籤的功能，包含找出上方比自己小的元素...等等。然後組合這些功能函式後，由 top\_down()執行從左上走到右下; bottom\_up() 再從右下回到左上，重複 20 次左右，label 完成。

```

> #左上到右下函數功能
> def min_from_top(self): ~
> def min_from_left(self): ~
> #右下到左上函數功能
> def min_from_right(self): ~
> def min_from_bot(self): ~
> def min_body1(self): #第0列/第0行皆不適用
>     self.min_from_top()
>     self.min_from_left()
> def min_body2(self): #第511列/第511行皆不適用
>     self.min_from_right()
>     self.min_from_bot()
>
> def zero_checker(self): ~
> def top_down(self):
>     while(True):
>         if(self.current_row == (self.arr.shape[0]-1) and self.current_col == (self.arr.shape[1]-1)): #走到最後一格
>             self.min_body1()
>             break
>         elif(self.zero_checker() is True): self.push() #遇到0的格子do nothing, 直接進下一格
>         elif(self.current_row == 0): #第0列, 只能比左邊
>             if(self.current_col == 0 and self.current_col == 0):
>                 self.push()
>             else:
>                 self.min_from_left()
>                 self.push()
>         elif(self.current_col == 0): #第0行, 只能比上面
>             self.min_from_top()
>             self.push()
>         else: #中間區塊
>             self.min_body1()
>             self.push()

```

4. 標記完成後，為過濾不到 500pixels 的區域，令一個新陣列將大於面積

500 的標記存入。並使用擂台法演算法找出各個區塊的最大最小 xy 座

標。

```

for k in range(0,self.current_label):
    if(count.count(k)>500):
        region.append(k)
for h in range(len(region)):
    X_max=0
    X_min=99999999
    Y_max=0
    Y_min=99999999
    area = 0
    centroid_x = 0
    centroid_y = 0
    for i in range(0, self.arr.shape[0]):
        for j in range(0, self.arr.shape[1]):
            if(self.arr[i][j]==self.arrregion[h]):
                area+=1
                centroid_x += i*1
                centroid_y += j*1
                if(i > X_max):
                    X_max = i
                if(i < X_min):
                    X_min = i
                if(j > Y_max):
                    Y_max = j
                if(j < Y_min):
                    Y_min = j
    centroid_x = (centroid_x//area)
    centroid_y = (centroid_y//area)
    cv2.rectangle(self.img, (Y_min, X_min), (Y_max, X_max), (255,0,0) , 2)
    cv2.line(self.img, (centroid_y - 7, centroid_x), (centroid_y + 7, centroid_x), (0, 0, 255), 3)
    cv2.line(self.img, (centroid_y, centroid_x - 7), (centroid_y,centroid_x + 7), (0, 0, 255), 3)

```