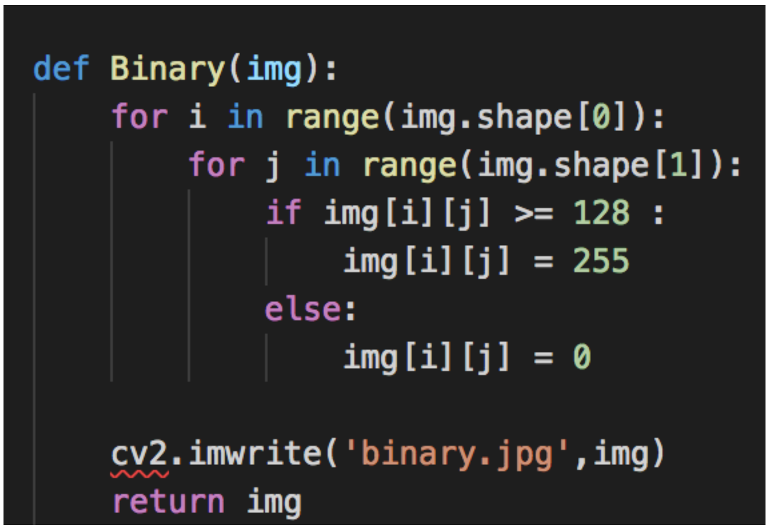
Computer and Robot Vision Homework 2
資工碩一 張家源 r07922102

使用語言： **Python**

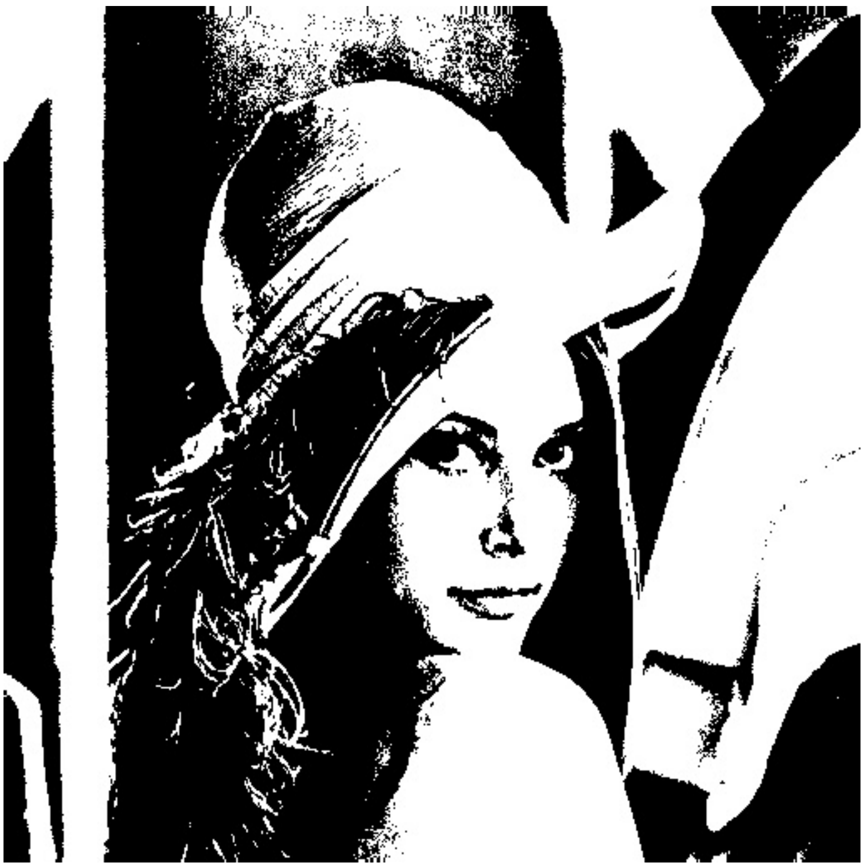
Part 1 : Binarize

對每個Pixel做Threshold檢查，大於等於128的部分全部換成255，反之設為0



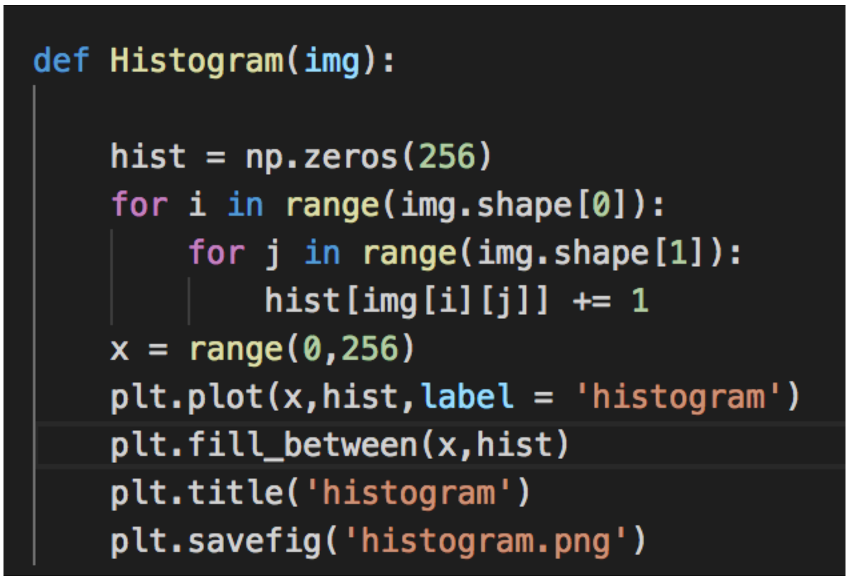
```python
def Binary(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j] >= 128 :
                img[i][j] = 255
            else:
                img[i][j] = 0

    cv2.imwrite('binary.jpg',img)
    return img
```

結果：



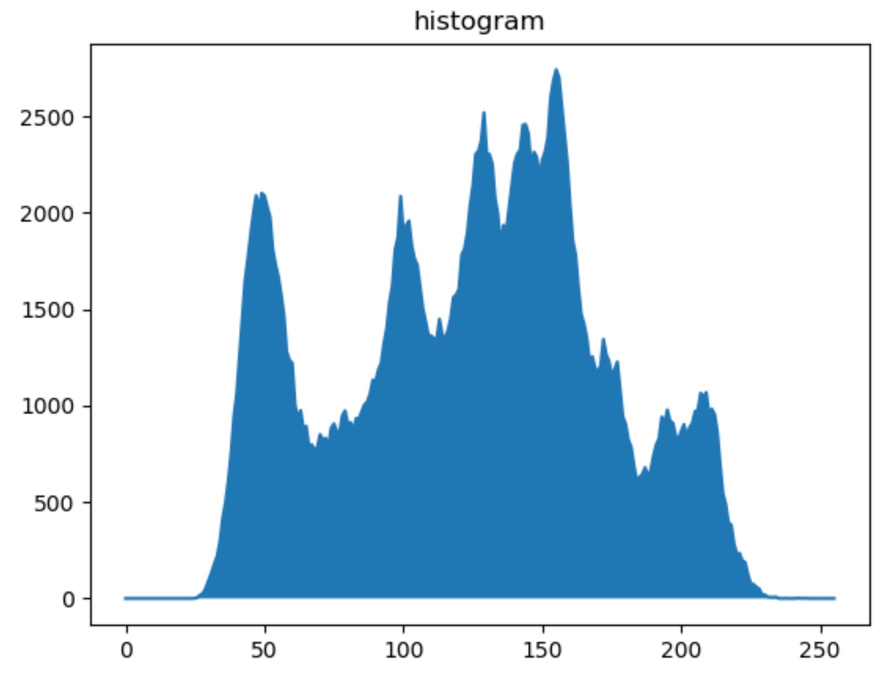
Part 2 : Histogram

建立一個list稱為hist，且大小為256(0-255)，預設value為0
針對每個Pixel做檢查，紀錄此Pixel出現的次數，並記錄在hist裡。



```python
def Histogram(img):

    hist = np.zeros(256)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            hist[img[i][j]] += 1
    x = range(0,256)
    plt.plot(x,hist,label = 'histogram')
    plt.fill_between(x,hist)
    plt.title('histogram')
    plt.savefig('histogram.png')
```

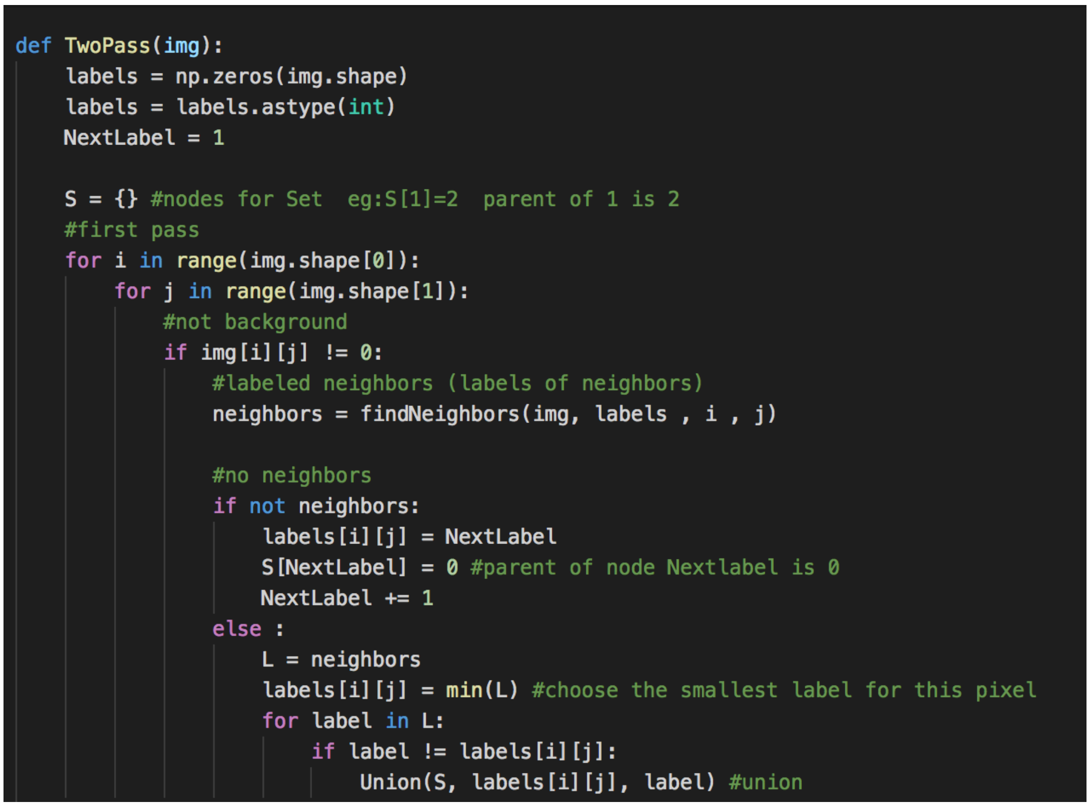
結果：



Part 3 : Connected Component

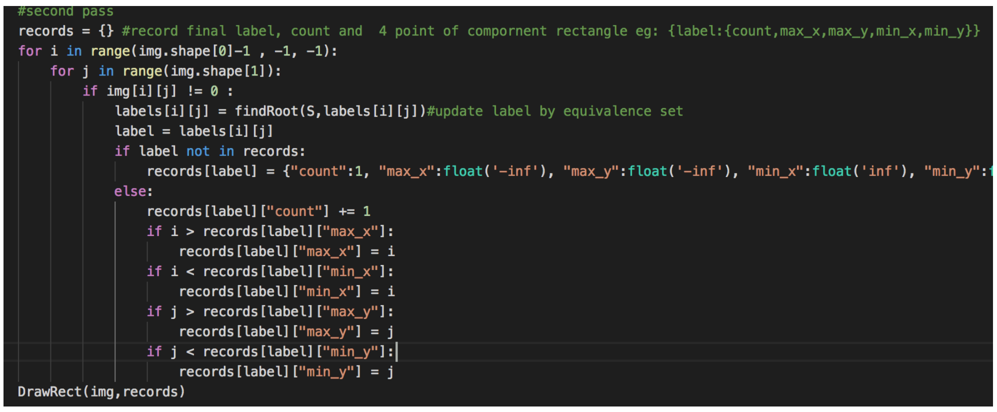
使用Algorithm: **The Classical Algorithm with 4-connected**
Equivalence classes利用disjoint set實作

First Pass:
labels:一個2d array，紀錄每個pixel目前的label，background設為0
NextLabel:目前用到的Label編號
S: 型態為Dict，存取所有的disjoint set node (key:# of label, value: parent of this label)
說明：從上至下、左到右，遇到foreground pixel時
case 1:無Neighbors，建立新的Set
case 2: 以neighbors最小的label來更新此pixel，並且union自己與所有neighbors



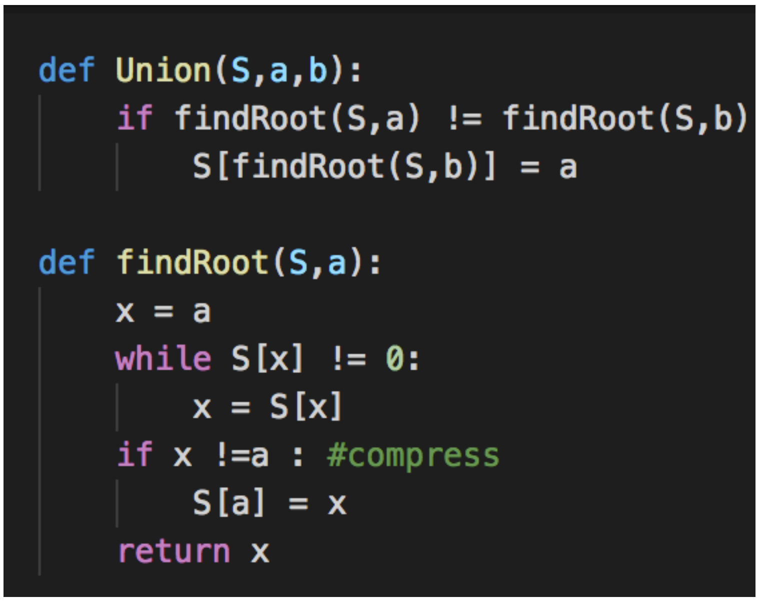
```
def TwoPass(img):
    labels = np.zeros(img.shape)
    labels = labels.astype(int)
    NextLabel = 1

    S = {} #nodes for Set  eg:S[1]=2  parent of 1 is 2
    #first pass
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            #not background
            if img[i][j] != 0:
                #labeled neighbors (labels of neighbors)
                neighbors = findNeighbors(img, labels , i , j)

                #no neighbors
                if not neighbors:
                    labels[i][j] = NextLabel
                    S[NextLabel] = 0 #parent of node Nextlabel is 0
                    NextLabel += 1
                else :
                    L = neighbors
                    labels[i][j] = min(L) #choose the smallest label for this pixel
                    for label in L:
                        if label != labels[i][j]:
                            Union(S, labels[i][j], label) #union
```

Second Pass:
records: 紀錄每個label出現的次數，以及框住此Component矩形的四點座標
說明：從下至上，由左至右，若pixel為foreground更新每個pixel對應的Equivalence class之最小值(即為Disjoint set之root)，同時記錄label出現的次數及Component座標，最後將Components畫出。



```
#second pass
records = {} #record final label, count and  4 point of compornent rectangle eg: {label:{count,max_x,max_y,min_x,min_y}}
for i in range(img.shape[0]-1 , -1, -1):
    for j in range(img.shape[1]):
        if img[i][j] != 0 :
            labels[i][j] = findRoot(S,labels[i][j])#update label by equivalence set
            label = labels[i][j]
            if label not in records:
                records[label] = {"count":1, "max_x":float('-inf'), "max_y":float('-inf'), "min_x":float('inf'), "min_y":
            else:
                records[label]["count"] += 1
                if i > records[label]["max_x"]:
                    records[label]["max_x"] = i
                if i < records[label]["min_x"]:
                    records[label]["min_x"] = i
                if j > records[label]["max_y"]:
                    records[label]["max_y"] = j
                if j < records[label]["min_y"]:
                    records[label]["min_y"] = j
DrawRect(img,records)
```

實作disjoint set:



```python
def Union(S,a,b):
    if findRoot(S,a) != findRoot(S,b)
        S[findRoot(S,b)] = a

def findRoot(S,a):
    x = a
    while S[x] != 0:
        x = S[x]
    if x !=a : #compress
        S[a] = x
    return x
```

結果: