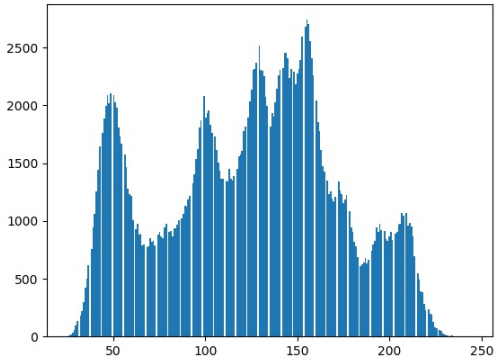
Computer vision homework2

學號：R07922162 姓名：胡嘉祐 系級：資工碩一

Result:

1.threshold at 128

2. Histogram



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

How to make it

1.threshold at 128

principle code:

```
#create threshold 128 image
img = cv2.imread('lena.bmp')
Histogram= []
H_list = []
for i  in range (0,256):
    H_list.append(0)
x= [[0 for j in range(len(img[0]))] for i in range(len(img))]
for i in range (0,512):
    for j in range (0,512):
        H_list[img[i][j][0]]+=1
        Histogram.append(img[i][j][0])
        if(img[i][j][0]<128):
            x[i][j]=0
            img[i][j][0]=0
            img[i][j][1]=0
            img[i][j][2]=0
        else:
            x[i][j]=1
            img[i][j][0]=255
            img[i][j][1]=255
            img[i][j][2]=255
cv2.imwrite('lena_threshold128.jpg',img)
##also create x = img for bounding_box
```

簡易的二元分類，將所有大於等於 128 的值，設成 255 ，小於 255 的值設成 0 ，同時建一個 二元矩陣 0 for 0 ,1 for 255，之後輸出圖片

2. Histogram

```
#create histogram
import matplotlib.pyplot as plt
plt.hist(Histogram,bins=256)
plt.savefig('Histgram.jpg')
#end create histogram
```

在 1.中已經建了 H_list，計算所有的值各有多少，再利用 plt.hist 直接輸出

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

1.main code:

```
#create connected component and bounding box
img2 =cv2.imread('lena_threshold128.jpg')

z= iterativeCCA (x) #for connected component
list_num=[]
list_num_sum=[]
list_num_big=[]
num=0
num_sum=0
for i in range (0,512):
    for j in range (0,512):
        same_num_flag=0
        for k in range(0,num):
            if(z[i][j]==list_num[k]):
                list_num_sum[k]+=1
                same_num_flag=1
                break
        if(same_num_flag==0):
            list_num.append(z[i][j])
            list_num_sum.append(1)
            num+=1
for i in range(0,num):
    if(list_num_sum[i]>=500):
        list_num_big.append(list_num[i])
        num_sum+=1
```

```
x_min=[]
x_max=[]
y_min=[]
y_max=[]
for i in range (0,len(list_num_big)):

    x_min.append(512)
    x_max.append(0)
    y_min.append(512)
    y_max.append(0)
for k in range(0,len(list_num_big)):
    for i in range (0,512):
        for j in range(0,512):
            if(z[i][j]==list_num_big[k]):
                x_min[k]=min(x_min[k],i)
                x_max[k]=max(x_max[k],i)
                y_min[k]=min(y_min[k],j)
                y_max[k]=max(y_max[k],j)
for k in range(0,len(list_num_big)):
    if(list_num_big[k]==0):
        continue
    draw_rectangle(img2,x_min[k],y_min[k],x_max[k],y_max[k]) #bounding box
    draw_central(img2,x_min[k],y_min[k],x_max[k],y_max[k]) #central
#end of connected components and bounding box
cv2.imwrite('connected.jpg',img2)
```

2. connected component algorithm + find min neighbor (4-connected)

```
############create label2 for neighbor check
num+=1
label2 =  [[0 for j in range(col+2)] for i in range(row+2)]
for i in range (0,row+2):
    label2[0][i]=num
for i in range (1,row+1):
    label2[i][0]=num
    for j in range (1,col+1):
        if(label[i-1][j-1]==0):
            label2[i][j]=num
        else:
            label2[i][j]=label[i-1][j-1]
    label2[i][col+1]=num
for i in range (0,row+2):
    label2[row+1][i]=num
change=True
while(change ==True):
    change=False
    for i in range(1,row):
        for j in range(1,col):
            if(label2[i][j]!=num):
                M=icca_min_neighbors_4connected(label2,i,j)
                if(label2[i][j]!=M):
                    change=True
                    label2[i][j]=M
    for i in range(row,0,-1):
        for j in range(col,0,-1):
            if(label2[i][j]!=num):
                M=icca_min_neighbors_4connected(label2,i,j)
                if(label2[i][j]!=M):
                    change=True
                    label2[i][j]=M
####return label2 to label1
for i in range (0,row):
    for j in range (0,col):
        if(label2[i+1][j+1]==num):
            label[i][j]=0
        else:
            label[i][j]=label2[i+1][j+1]

return label
```

使用 4-connected 的方式再加上 iterative connected component algo 實作，neighbor 則是直接找四個方向的 min 值。