Computer Vision HW2

R07921010 余奇安

1.Binary Image (Threshold at 128)

I define function bin to check every pixels after getting image by cv.imread(“Lena.jpg”,0)

0 means gray scale, 1 for color, -1 for unchanged.

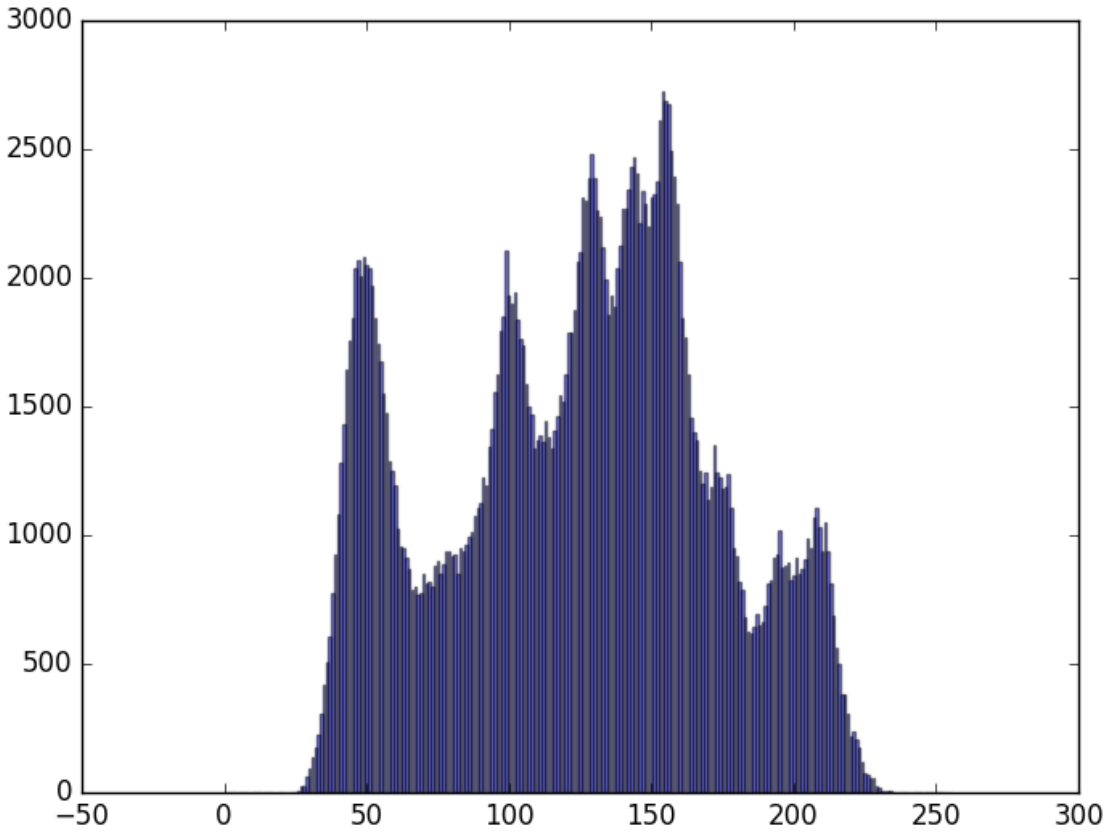
Main code:

```
img_gray=cv.imread("Lena.jpg",0)#0 for gray scale, 1 for color,-1 Unchanged

def bin (img):
    row=img.shape[0]
    col=img.shape[1]
    for i in range(row):
        for j in range(col):
            if img[i][j]<128:
                img[i][j]=0
            else:
                img[i][j]=255

    #cv.imwrite("thresold_128.png",img)
    return img
```

2. A histogram
I use matplotlib to do the histogram
I generate a list x range from 0 to 255, another list p to count the number of pixel value.
Get the pixel value from image, and store it to the respect index of list p
For example:
If I get pixel value 13 from the img[i][j], p[13] will be added by 1.

Plot the histogram, x list as x axle, p as y axle. X for pixel value, Y for the mount of pixel value.

```
def histogram(img):
    p=[0]*256
    x=[i for i in range(256)]
    #print (x)
    for i in range(row):
        for j in range(col):
            p[img[i][j]]+=1
    plt.bar(x,p,align="center",alpha=0.5)
    plt.savefig("histogram.png")
    plt.show()
```



3. Find the connected components
I ignored + at centroid, just plot the bounding box
I use **4-connected** to define connection.
My algorithm to find connected components is **iterative algorithm**,which is the most simple and understandable.

Define findmin function to make the process more clear.
Initialize the image with the increasing integer from left to right, up to down.

```
def findmin(label,r1,c1,r2,c2):
    if label[r2][c2]!=0:
        ans=min(label[r1][c1],label[r2][c2])
    else:
        ans=label[r1][c1]
    return ans
def connected(img):
    row=img.shape[0]
    col=img.shape[1]

    im=bin(img)
    nim=np.zeros((row,col),dtype=np.int)
    #df3=pd.DataFrame(nim)
    #df3.to_csv("binary.csv")

    label=1
    for i in range(row):
        for j in range(col):
            if im[i][j]==255:
                nim[i][j]=label
                label+=1
```

The algorithm should be stopped after no pixels change the value.
At first, iterative algorithm check the label value from left to right, up to down.
The pixel should be compared with the left and up pixels, and change to the smallest value except for 0.

```
check=1
while check==1:
    check=0
    for i in range(row):#from left to right,up to down
        for j in range(col):
            temp=0
            if i==0 and j-1>=0:
                if nim[i][j]!=0 and nim[i][j-1]!=0:
                    if nim[i][j]!=nim[i][j-1]:
                        check=1
                    nim[i][j]=min(nim[i][j],nim[i][j-1])

            elif j==0 and i-1>=0:
                if nim[i][j]!=0 and nim[i-1][j]!=0:
                    if nim[i][j]!=nim[i-1][j]:
                        check=1
                    nim[i][j]=min(nim[i][j],nim[i-1][j])

            elif i>0 and j>0:
                if nim[i][j]!=0 :
                    temp=findmin(nim,i,j,i-1,j)
                    temp2=min(temp,nim[i][j-1])
                    if temp2!=0:
                        temp=temp2
                    if nim[i][j]!=temp:
                        check=1
                    nim[i][j]=temp
```

Second, bottom up process.
In this process, comparing every pixel to right and down pixels from right to left, down to up.

```
    for i in range(1,row+1):
        for j in range(1,col+1):
            temp=0
            if i==1 and j>1:
                if nim[-i][-j]!=0 and nim[-i][-j+1]!=0:
                    if nim[-i][-j]!=nim[-i][-j+1]:
                        check=1
                    nim[-i][-j]=min(nim[-i][-j],nim[-i][-j+1])

            elif j==1 and i>1:
                if nim[-i][-j]!=0 and nim[-i+1][-j]!=0:
                    if nim[-i][-j]!=nim[-i+1][-j]:
                        check=1
                    nim[-i][-j]=min(nim[-i][-j],nim[-i+1][-j])

            elif j>1 and i>1:
                if nim[-i][-j]!=0:
                    temp=findmin(nim,-i,-j,-i+1,-j)
                    temp2=min(temp,nim[-i][-j+1])
                    if temp2!=0:
                        temp=temp2
                    if nim[-i][-j]!=temp:
                        check=1
                    nim[-i][-j]=temp
```

Finally, calculate the area of every connected region. Discard the region under 500.
plot the bounding box.

```
    area=[]
    for i in range(row):
        for j in range(col):
            if nim[i][j]!=0:
                area.append(nim[i][j])
    #print(im)
    for item in set(area):
        if area.count(item)>500:
            x=[]
            y=[]
            #print(x,y)
            for i in range(row):
                for j in range(col):
                    if nim[i][j]==item:
                        x.append(j)
                        y.append(i)
            print(min(x),min(y),max(x),max(y),item,area.count(item))
            cv.rectangle(img,(min(x),min(y)),(max(x),max(y)),(55,25,155),4)

    print (nim)
    cv.imwrite("connected.png",img)
    return nim
```

Result:
Get 5 connected region, which are

(0, 0, 87, 511, 1, 18370)
(125, 0, 511, 511, 73, 105966)
(0, 399, 31, 511, 107925, 1493)
(89, 237, 139, 287, 66942, 645)
(118, 94, 150, 237, 26204, 2010)

(x1,y1,x2,y2,label,number of label)