《计算机视觉》实验报告

姓名：戴枫源 学号：19120199

实验三

一. 任务 1

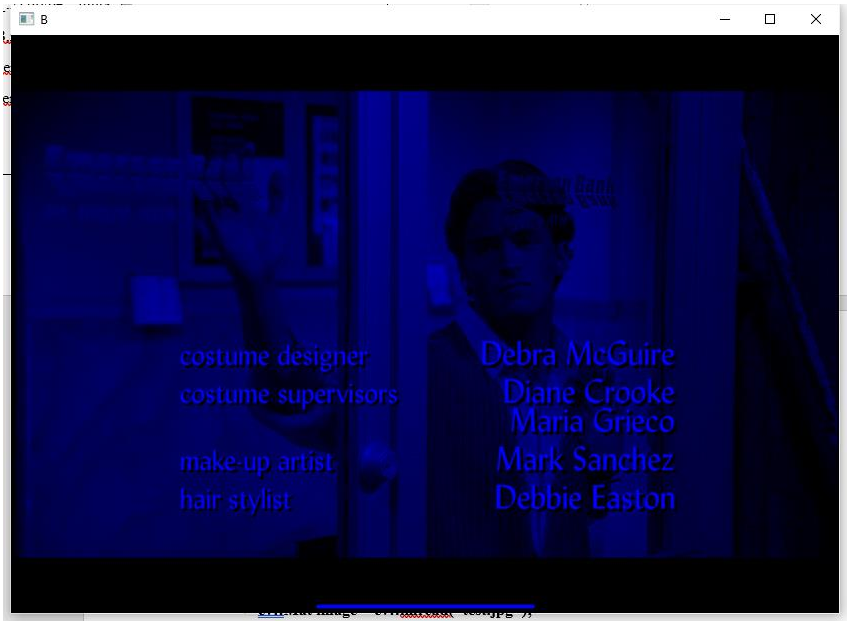
a) 核心代码：

如果是编程类的，贴出核心代码，及必要的注释

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 读取一张图片，将其转换为 HSV 空间
img = cv.imread('../chandler.PNG',-1)
img_hsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)
# cv.imshow("img",img_hsv)

# 分离原图片 RGB 通道及转换后的图片 HSV 通道
B,G,R = cv.split(img)
zeros = np.zeros(img.shape[:2],dtype="uint8")
cv.imshow("B",cv.merge([B,zeros,zeros]))
cv.imshow("G",cv.merge([zeros,G,zeros]))
cv.imshow("R",cv.merge([zeros,zeros,B]))

H,S,V = cv.split(img_hsv)
```

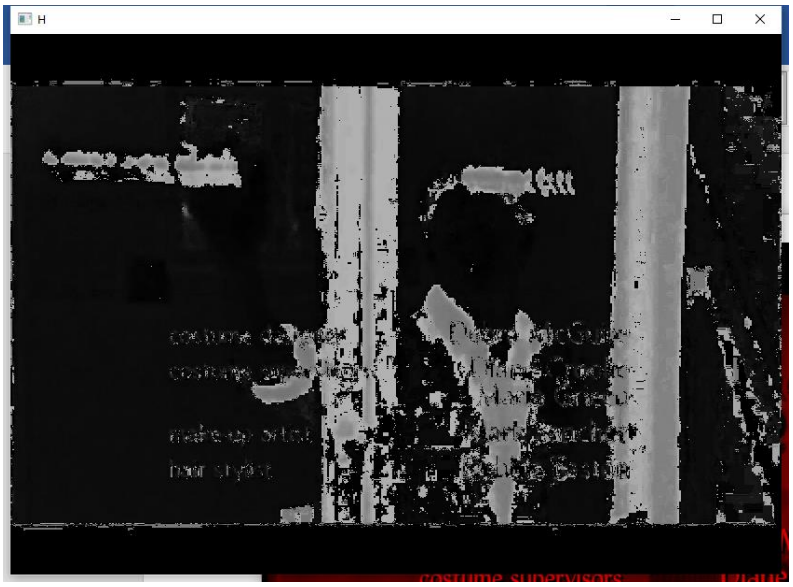
```
cv.imshow('H', H)
cv.imshow('S', S)
cv.imshow('V', V)

cv.waitKey()
cv.destroyAllWindows()

# 对 RGB 三个通道分别画出其三维图
fig = plt.figure(figsize=(12,8))
ax = Axes3D(fig)
zB = img[:,:,0]
zG = img[:,:,1]
zR = img[:,:,2]
x = np.arange(0,828,1)
y = np.arange(0,578,1)
X,Y = np.meshgrid(x,y)

# 这里以画出 B 通道的三维图为例
# 如果要画出 G 通道的三维图，只需修改下面的函数中第三个参数为 zG 即可,R 通道同理
ax.plot_surface(X,Y,zB,cmap=plt.get_cmap('rainbow'))
plt.title("B")
plt.show()
```

b) 实验结果截图


B
costume designer
Debra McGuire
costume supervisors
Diane Crooke
Maria Grieco
make-up artist
Mark Sanchez
hair stylist
Debbie Easton

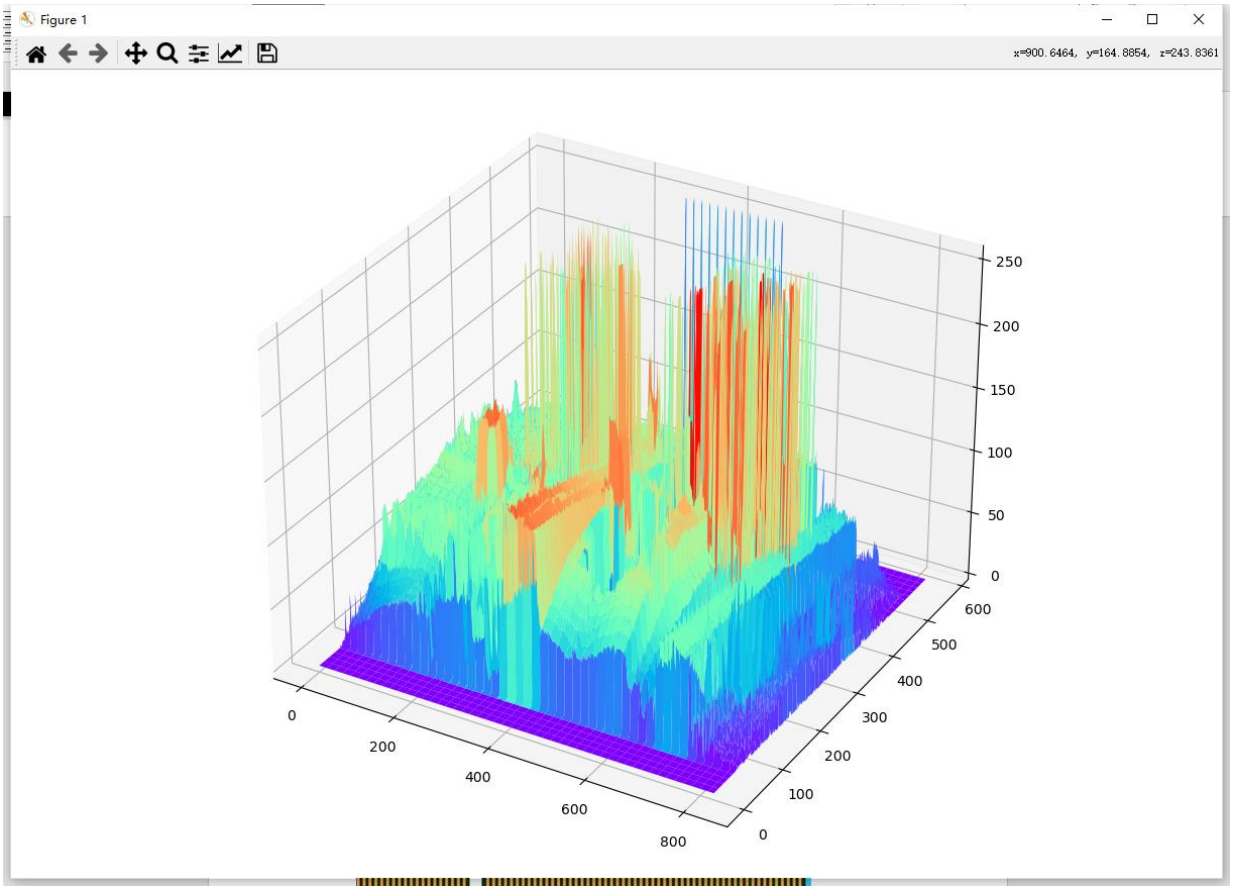

G
costume designer
Debra McGuire
costume supervisors
Diane Crooke
Maria Grieco
make-up artist
Mark Sanchez
hair stylist
Debbie Easton


R
costume designer
Debra McGuire
costume supervisors
Diane Crooke
Maria Grieco
make-up artist
Mark Sanchez
hair stylist
Debbie Easton




costume designer
Debra McGuire
costume supervisors
Diane Crooke
Maria Grieco
make-up artist
Mark Sanchez
hair stylist
Debbie Easton


costume designer
Debra McGuire
costume supervisors
Diane Crooke
Maria Grieco
make-up artist
Mark Sanchez
hair stylist
Debbie Easton



c) 实验小结

读取图片，并转换为 HSV 空间可以用 cv.cv2Color 从 BGR 转化为 HSV。分离原图片的 BGR 通道和 HSV 通道结合函数 cv.split 和 cv.merge 即可。最后画三维图需要用到 Axes3D 中的 plot_surface 方法，同时我也学到了 numpy 中的 meshgrid 函数，用于生成坐标便于画图。

二. 任务 2

a) 核心代码：

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt


# 读取彩色图像 home_color
home_color = cv.imread("../home_color.png",-1)
home_grey = cv.cvtColor(home_color,cv.COLOR_BGR2GRAY)
```

```
# 画出灰度化图像 home_gray 的灰度直方图，并拼接原灰度图与结果图
# ravel() 将高维数组压成一维
plt.hist(home_grey.ravel(),256)
plt.title('Histrogram of grey image')
plt.savefig('home_grey_hist.png')
plt.show()
home_grey_hist = cv.imread("home_grey_hist.png",0)
# （1）当输入矩阵是 uint8 类型的时候，此时 imshow 显示图像的时候，会认为输入矩阵的范
在 0-255 之间。
# （2）如果 imshow 的参数是 double 类型的时候，那么 imshow 会认为输入矩阵的范围在 0-1
#  所以一定要指明 dtype 为 uint8
home_grey_syn = np.zeros((home_grey.shape[0],home_grey.shape[1]*2),dtype='uint8')
home_grey_syn[:,:home_grey.shape[1]] = home_grey.copy()
home_grey_syn[:,home_grey.shape[1]:]
cv.resize(home_grey_hist,(home_grey.shape[1],home_grey.shape[0]),interpolation=cv.INTER_AREA

# 画出彩色 home_color 图像的直方图，并拼接原彩色图与结果图，且与上一问结果放在同一个
口中显示
color = ('b','g','r')
for id,bgrcolor in enumerate(color):
    plt.hist(home_color[:,:,id].flatten(),bins=256,density=True,color=bgrcolor,alpha=.7)
    plt.title('Histrogram of Color image')
plt.savefig('home_color_hist.png')
plt.show()
home_color_hist = cv.imread("home_color_hist.png",cv.IMREAD_COLOR)
home_color_syn
np.zeros((home_color.shape[0],home_color.shape[1]*2,home_color.shape[2]),dtype='uint8')
home_color_syn[:,:home_color.shape[1],:] = home_color
```

```
home_color_syn[:,home_color.shape[1]:,:]
cv.resize(home_color_hist,(home_color.shape[1],home_color.shape[0]),interpolation=cv.INTER_AR

ROI_area = np.zeros(home_color.shape,dtype='uint8')
ROI_area[50:100,100:200] = home_color[50:100,100:200]
ROI_mask = np.zeros(home_color.shape,dtype='uint8')
ROI_mask[:][:] = [255,255,255]
ROI_mask[50:100,100:200] = [0,0,0]

for id,bgrcolor in enumerate(color):
    plt.hist(ROI_area[:,:,id].flatten(),bins=256,density=True,color=bgrcolor,alpha=.7)
    plt.title('Histrogram of ROI_area')
plt.savefig('ROI_area.png')
plt.show()
ROI_hist = cv.imread("ROI_area.png",cv.IMREAD_COLOR)
ROI_hist
cv.resize(ROI_hist,(home_color.shape[1],home_color.shape[0]),interpolation=cv.INTER_AREA)
# cv.imshow("roi hist",ROI_hist)

original_mask = np.hstack((home_color,ROI_mask))
roi_and_hist = np.hstack((ROI_area,ROI_hist))
res = np.vstack((original_mask,roi_and_hist))
cv.imshow("res",res)

cv.imshow("home_grey_syn",home_grey_syn)
cv.imshow("home_color_syn",home_color_syn)
cv.waitKey()
cv.destroyAllWindows()
```

b) 实验结果截图

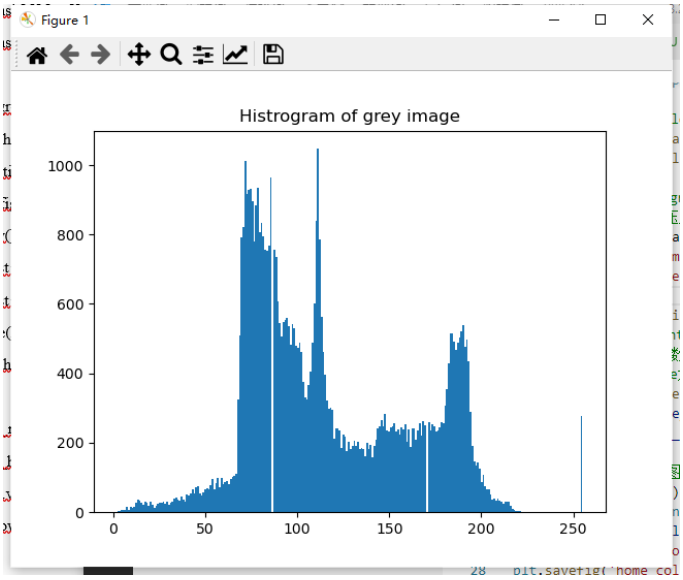


Figure 1
Histrogram of grey image
1000
800
600
400
200
0
0
50
100
150
200
250

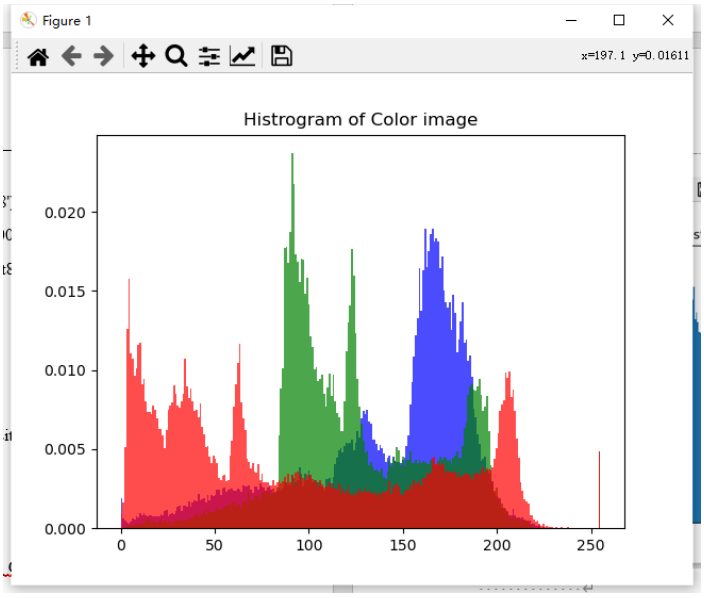


Figure 1
x=197.1 y=0.01611
Histrogram of Color image
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250

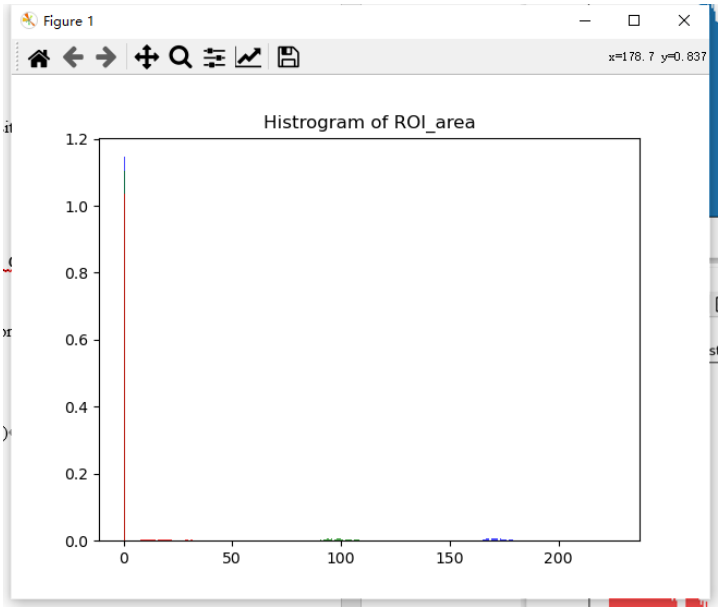
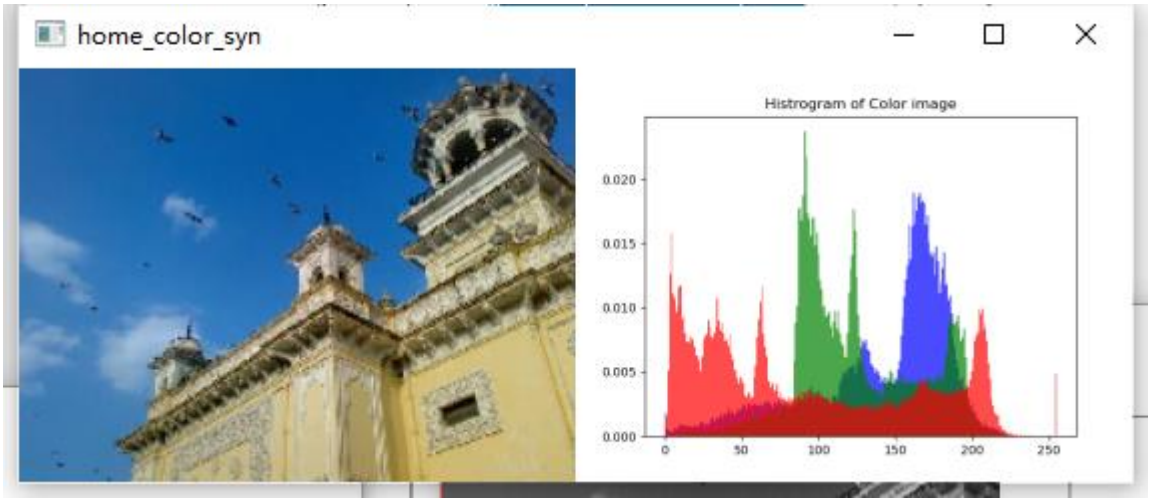
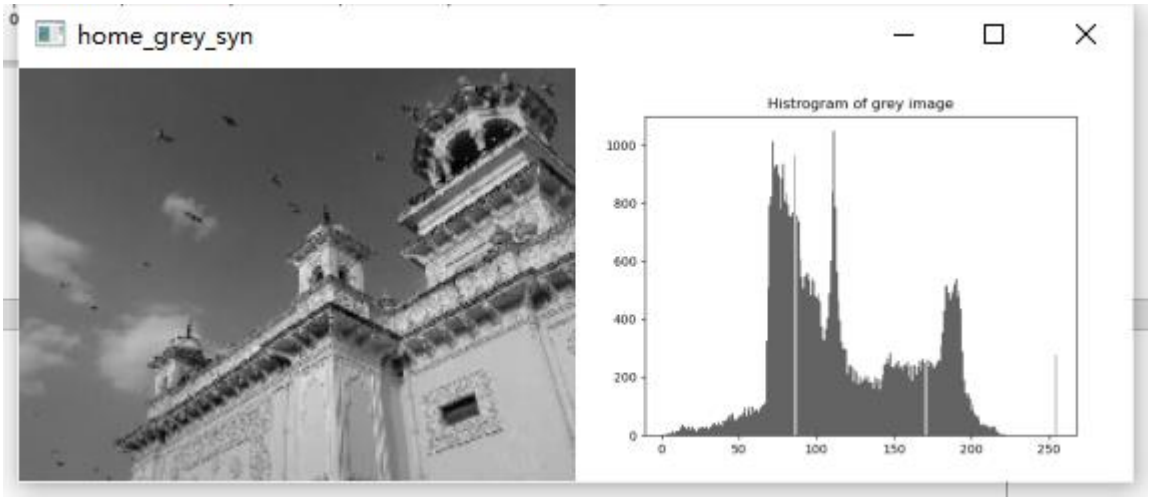
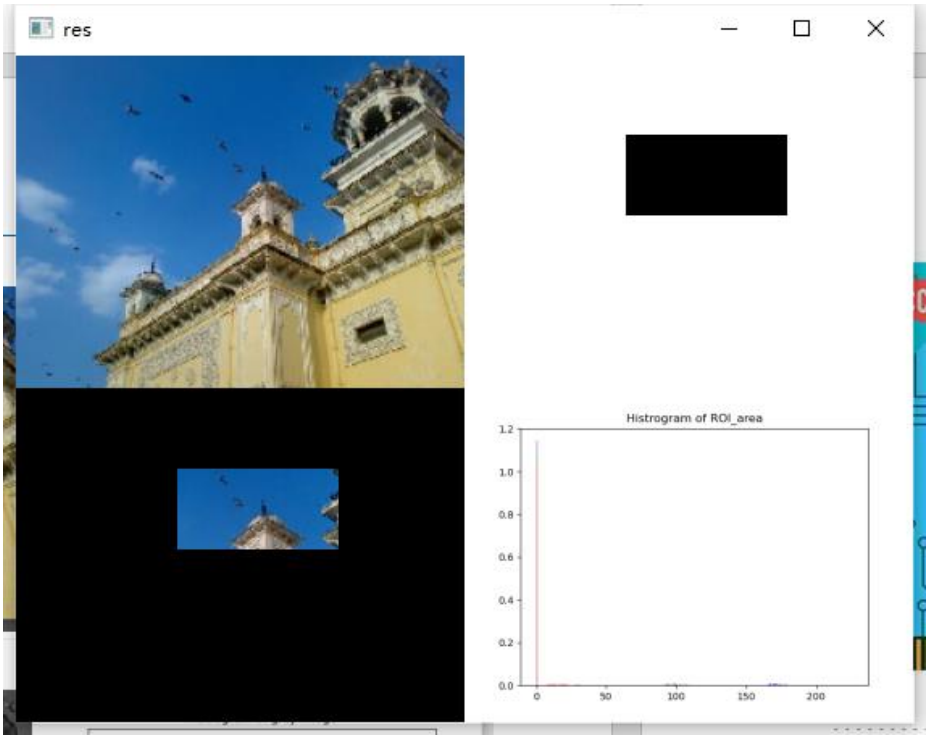


Figure 1
x=178.7 y=0.837
Histrogram of ROI_area
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200


home_color_syn
Histrogram of Color image
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250


home_grey_syn
Histrogram of grey image
1000
800
600
400
200
0
0
50
100
150
200
250



c) 实验小结

本次实验困扰我比较大的地方是我没有注意到 imshow 的一个特性：当输入矩阵的元素类型是 uint8 时，显示图像默认输入矩阵的范围是 0-255；输入矩阵元素的类型是 double 时，显示图像会认为输出矩阵的范围是 0-1。因此用 np.zeros 时，必须指定生成的元素类型是 uint8，不然通过 imshow 显示的图片会不正常。最后又学到了两个 numpy 中的函数：vstack 和 hstack。分别用于在行和列合并矩阵，这使得直接合并图像方便了很多。

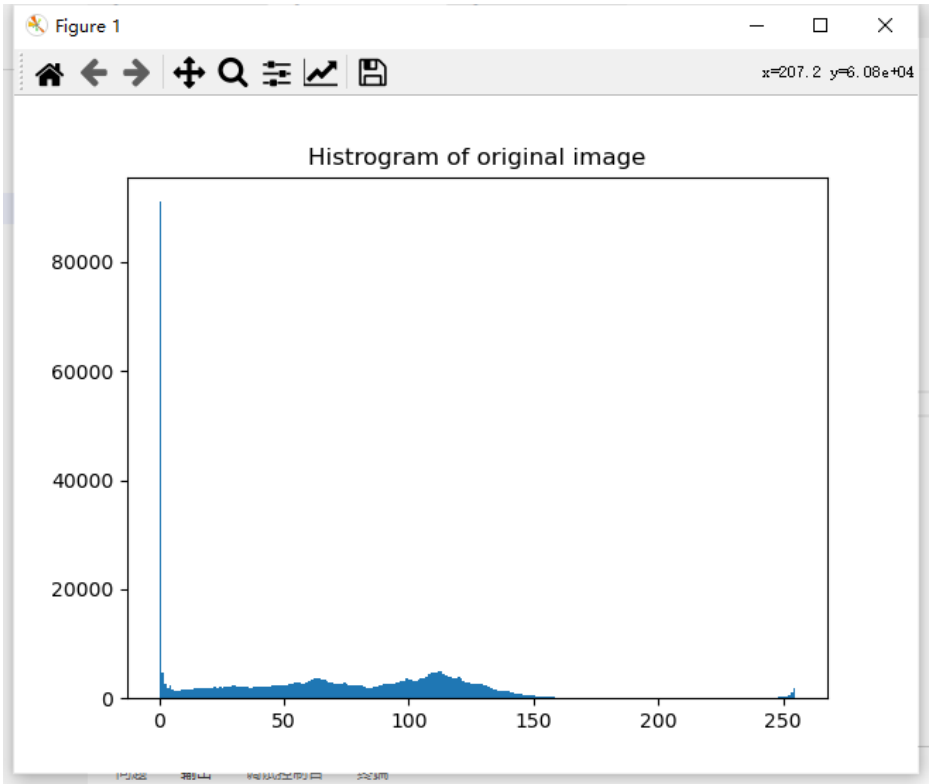
三. 任务 3

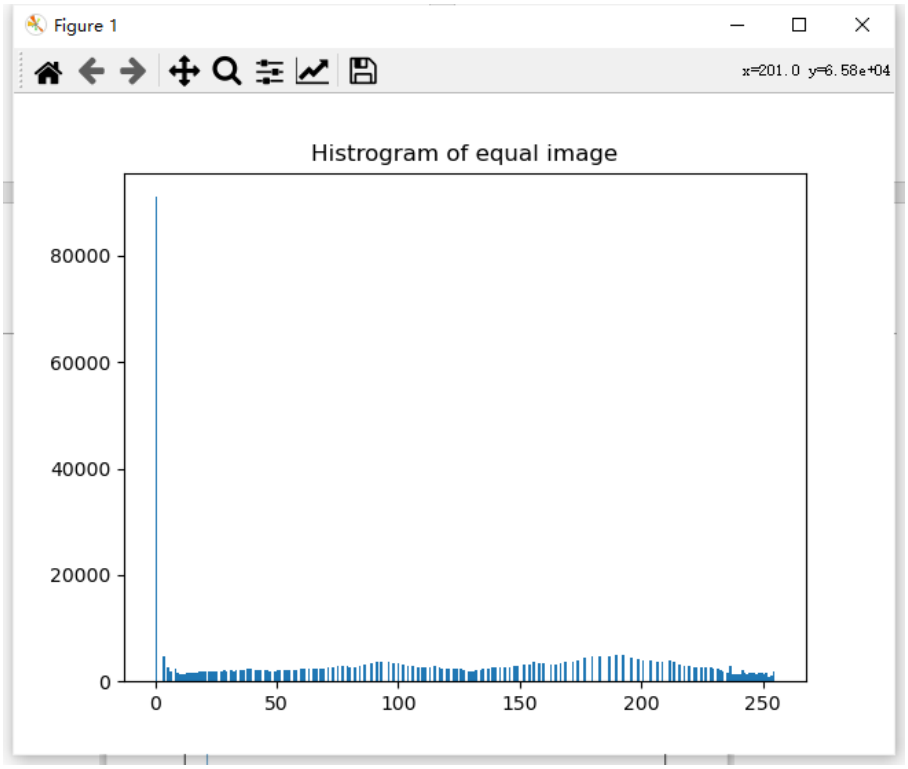
a) 核心代码：

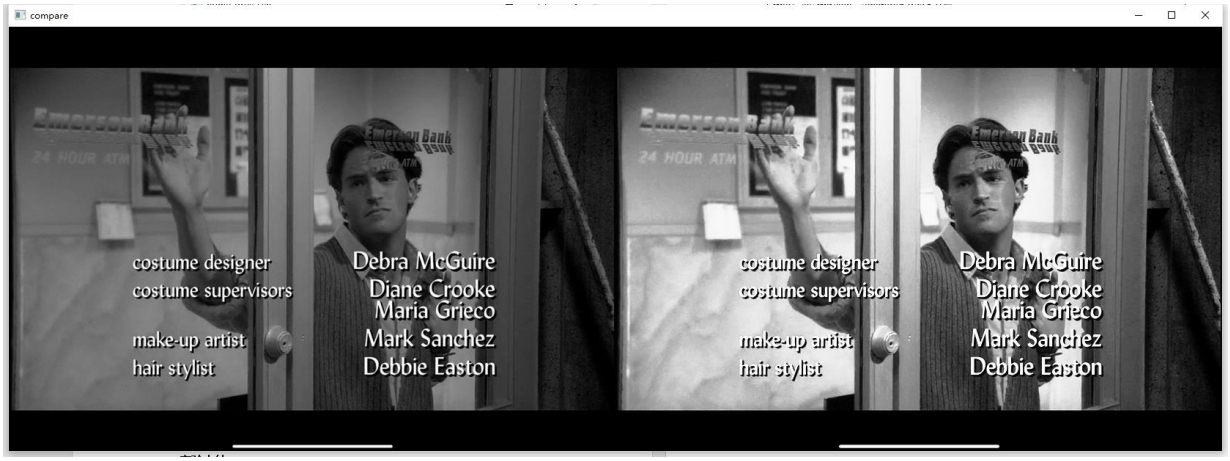
```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np


img = cv.imread('../chandler.PNG',0)


plt.hist(img.ravel(),256)
plt.title('Histrogram of original image')
plt.show()
```

```
# 直方图均衡化
equ = cv.equalizeHist(img)
plt.hist(equ.ravel(),256)
plt.title('Histrogram of equal image')
plt.show()

res = np.hstack((img,equ))
cv.imshow("compare",res)
cv.waitKey()
cv.destroyAllWindows()
```

b) 实验结果截图







c) 实验小结

关于直方图均衡化，我主要调用了 cv.equalizeHist 函数，通过实验图片也可以看到，均衡化后的直方图更接近均匀分布，两个图片对比后可以发现，均衡化后图片的对比度更强。