实 验 报 告

实验 10

517030910374
郭嘉宋

目录:

注：在编程过程中我也遇到了很多困难和 Bug，还有很多思考了很久得出的结论与解释，以及一些需要注释的内容，都在下述“2. 实验过程”中，用加粗的 NOTE 加以了说明。

正文:

1. 实验准备

(1) 实验环境

本实验主要采用 Ubuntu 系统下的 python2.7 进行，使用 miniconda 环境，Ubuntu 系统安装在 VMware Workstation 提供的虚拟机中，同时使用 OpenCV、Numpy 等库对图像进行分析与后续处理。

(2) 实验目的

实验 10:

了解图像特征抽取，学习使用 OpenCV 和 Numpy 等库对图像进行分析，了解图像的颜色直方图、灰度图、梯度图等相关信息。

(3) 实验原理

实验 10:

图像的基本元素是“像素(pixel)”。一幅宽为 W，高为 H 的灰度图像在计算机中用一个 W×H 的矩阵存储。矩阵的每个元素是图像对应位置像素的灰度值，范围在 0 到 255 之间。灰度值 0 表示黑色，灰度值 255 表示白色。图像坐标系以左上角为原点，横向为 x 方向，纵向为 y 方向。

彩色图像的每个像素由红色(R)、绿色(G)、蓝色(B)三个颜色分量构成，从而能够呈现多种色彩。一幅宽为 W，高为 H 的彩色图像在计算机中用一个 W×H×3 的矩阵存储，其中最后一个维度表示 RGB 颜色空间。

假设 I(x, y)表示一幅灰度图像，则它 Y 方向的梯度定义为:

$$I_y(x, y) = \frac{\partial I(x, y)}{\partial y} = I(x, y+1) - I(x, y-1)$$

X 方向的梯度定义为:

$$I_x(x, y) = \frac{\partial I(x, y)}{\partial x} = I(x+1, y) - I(x-1, y)$$

梯度强度定义为:

$$M(x, y) = \sqrt{I_x(x, y)^2 + I_y(x, y)^2}$$

图像边界处的梯度无法根据上述梯度定义求出，并且边界的梯度的定义常常随不同应用场景有改变，如有的定义边界像素的梯度为零，有的定义边界像素的梯度等于它临近的非边界像素的梯度。本实验中涉及梯度的地方均不考虑

边界像素的梯度。

图像特征：至今为止,图像特征没有通用和精确的定义，图像特征的定义往往由问题或者应用类型决定。图像特征提取的总体目标是用尽可能少的数据量最大程度地描述图像信息。

图像直方图特征：图像的直方图特征是一种全局统计特征,描述了图像的某一数值特性的分布情况,反映了图像的大体风格。

颜色直方图：彩色图像 RGB 三种颜色分量在整张图像中所占的相对比例，反映了图像全局的主体色调。

灰度直方图：灰度图像灰度值的分布情况，反映了灰度图像的明暗程度。

梯度直方图：灰度图像的梯度强度分布情况，反映了图像的纹理的疏密程度。

2. 实验过程

(1) 实验 10

1.1 实验步骤

练习 1:

首先用 cv2 读入图片，再对 img 变量进行操作，如下图所示：

```
import cv2
import matplotlib.pyplot as plt
img=cv2.imread("images/img1.png",cv2.IMREAD_COLOR)
# B G R row before column row1 row2 row3...

# print img[:,:,0]
B=img[:, :, 0].ravel()
G=img[:, :, 1].ravel()
R=img[:, :, 2].ravel()
sum_B=0
sum_G=0
sum_R=0
```

NOTE：cv2 读下来是一个 3 维 list，使用数组操作取出对应分量即可，其中 ravel 函数是将 list 降维，从 2 维降到 1 维。

```
sum_B=0
sum_G=0
sum_R=0
for b1 in B:
    sum_B+=b1
for g1 in G:
    sum_G+=g1
for r1 in R:
    sum_R+=r1
sum_all=sum_B+sum_G+sum_R
sum_B/=float(sum_all)
sum_G/=float(sum_all)
sum_R/=float(sum_all)
RGB_LIST=[sum_B,sum_G,sum_R]
Xtips=['B','G','R']
```

如上图所示，最后分别 RGB 三个总和归一化即可。

接下来是可视化部分，较为容易，使用 plt 中的 bar 函数即可，如下图所示：

```
plt.bar(range(3),RGB_LIST,align="center",color=['B','G','R'],label=['B','G','R'])
# plt.hist(range(3),RGB_LIST,align="center",color=['B','G','R'],label=['B','G','R'])
plt.xticks(range(3),Xtips)
for xx, yy in zip(range(3),RGB_LIST):

    plt.text(xx, yy, str(yy), ha='center')

plt.show()
```

NOTE：最开始我是用 hist 画图，发现 hist 不支持每个直方图柱不同颜色，因此更改成 bar 画图即可指定颜色。

练习 2:

```
import cv2
import numpy
import matplotlib.pyplot as plt
img=cv2.imread("images/img1.png",cv2.IMREAD_GRAYSCALE)
gray=img.ravel()
# use ravel() to decrease the dimension


plt.hist(gray, bins=100,density=1,color='black')
# plt.hist(gray, bins=100,normed=1,color='black')
plt.title("Gray picture of img1")
plt.show()
```

如上图所示，同练习 1，在数据处理方面取出灰度使用 hist 画图即可，较容易。

练习 3:

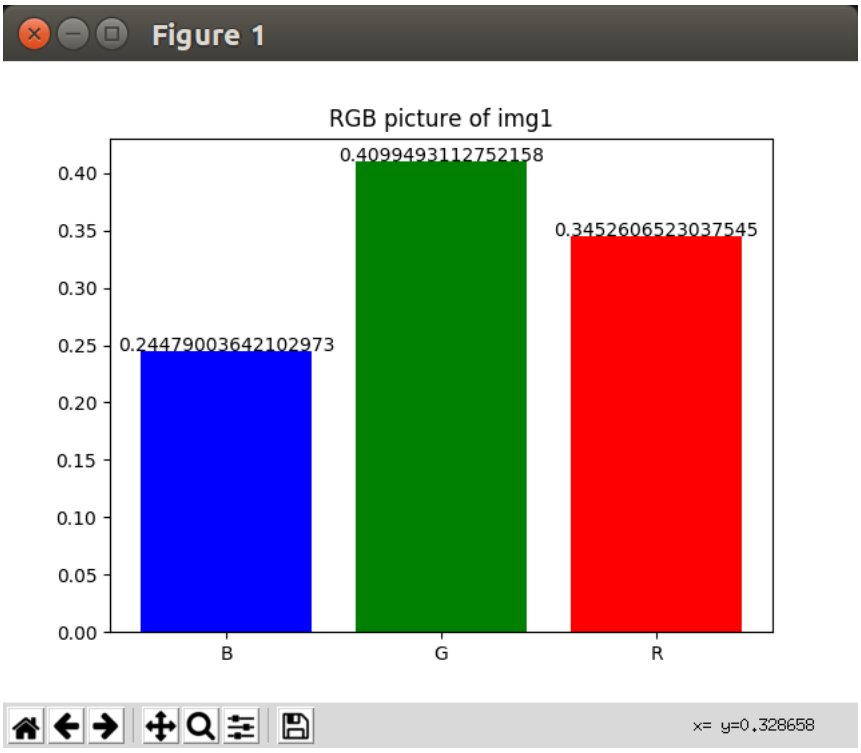
```
y=len(img)
x=len(img[0])
gradient={}
for t in range (361):
    gradient[t]=0
for k1 in range(1,x-1):
    for k2 in range(1,y-1):
        tmp=int(numpy.sqrt(pow(float(img[k2+1][k1])-float(img[k2-1][k1]),2)
                           +pow(float(img[k2][k1+1])-float(img[k2][k1-1]),2)))
        gradient[tmp]+=1

gradient_list=[]
sum=0
```

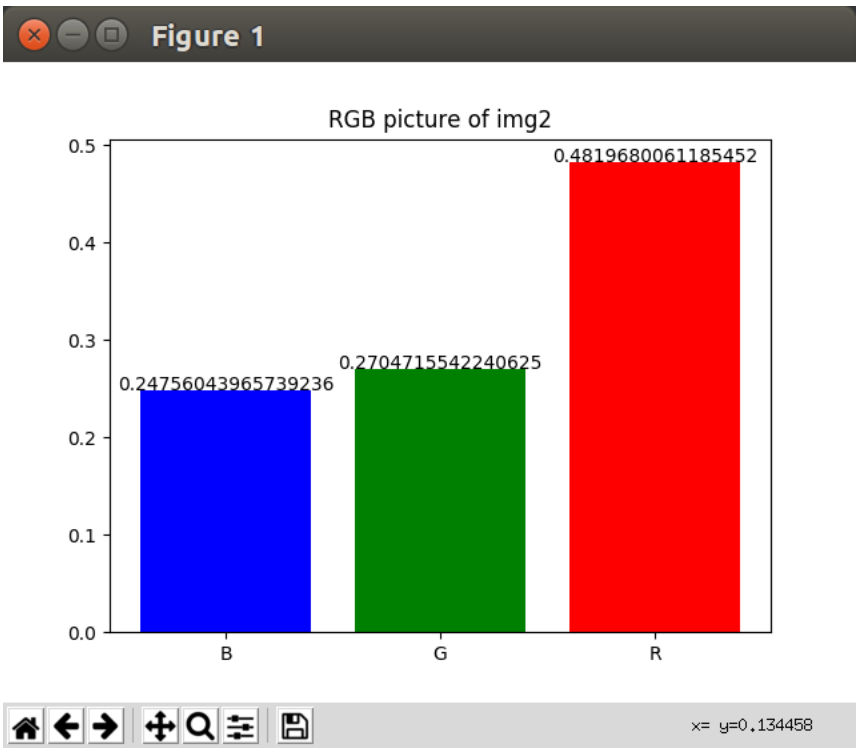
```
for t in range (361):
    gradient_list.append(gradient[t])
    sum+=gradient[t]
for t in range (361):
    gradient_list[t]/=float(sum)

plt.bar(range(361),gradient_list,align="center",color="black")
plt.title("Gragient of img1")
plt.show()
```

如上图所示，同练习 1、2，仅需要对参数作差即可，较容易，详见源码。

1.2 实验结果

练习 1:





练习 2:

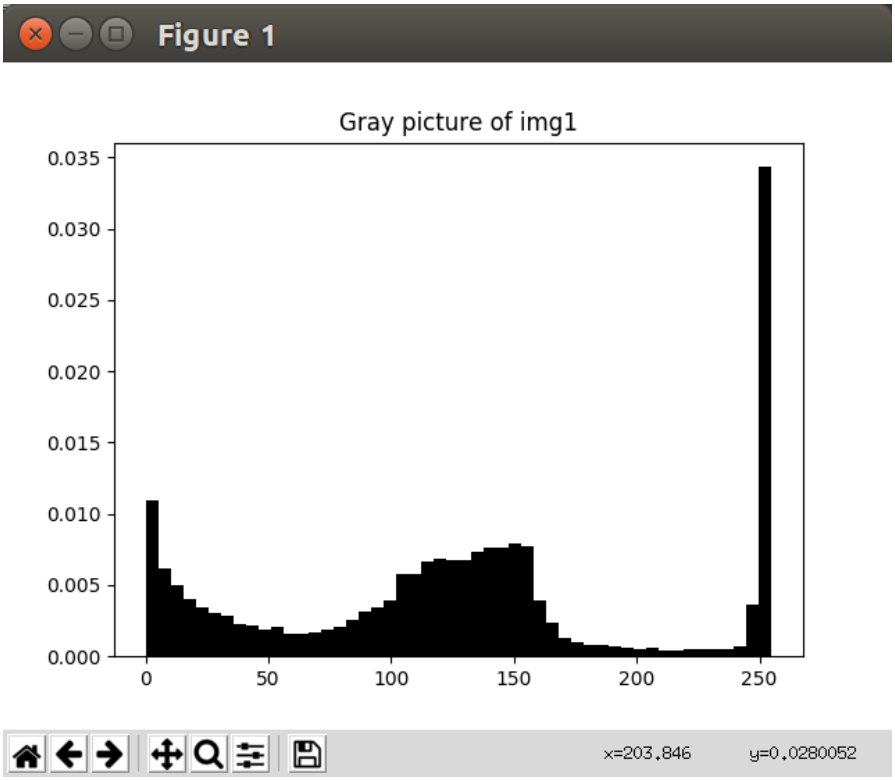


Figure 1
Gray picture of img1
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
x=203.846
y=0.0280052

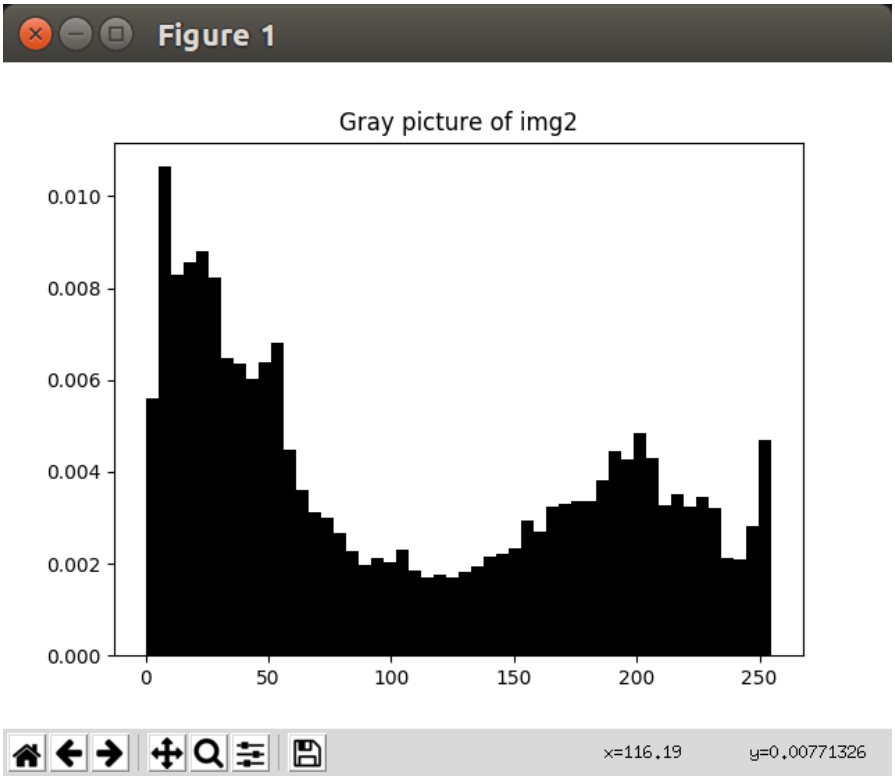


Figure 1
Gray picture of img2
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250
x=116.19
y=0.00771326

练习 3:

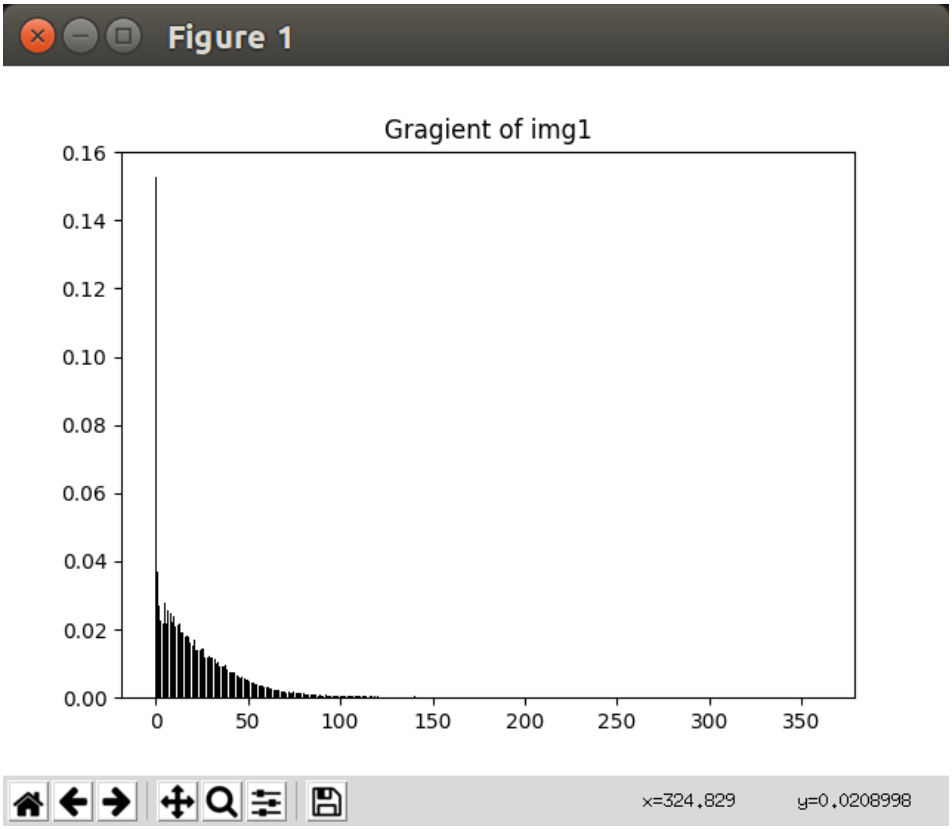


Figure 1
Gragient of img1
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250
300
350
x=324.829
y=0.0208998

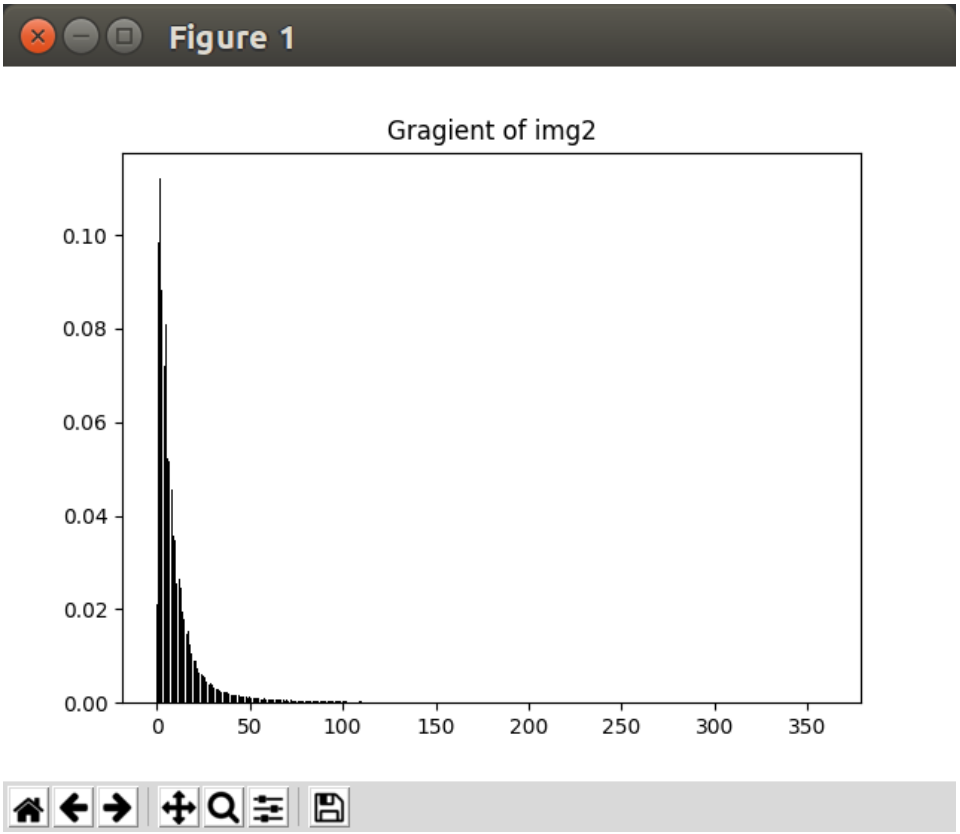


Figure 1
Gragient of img2
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250
300
350

NOTE： 如下图所示，我还制作了两幅图分别的 RGB 分量图，原理相同。

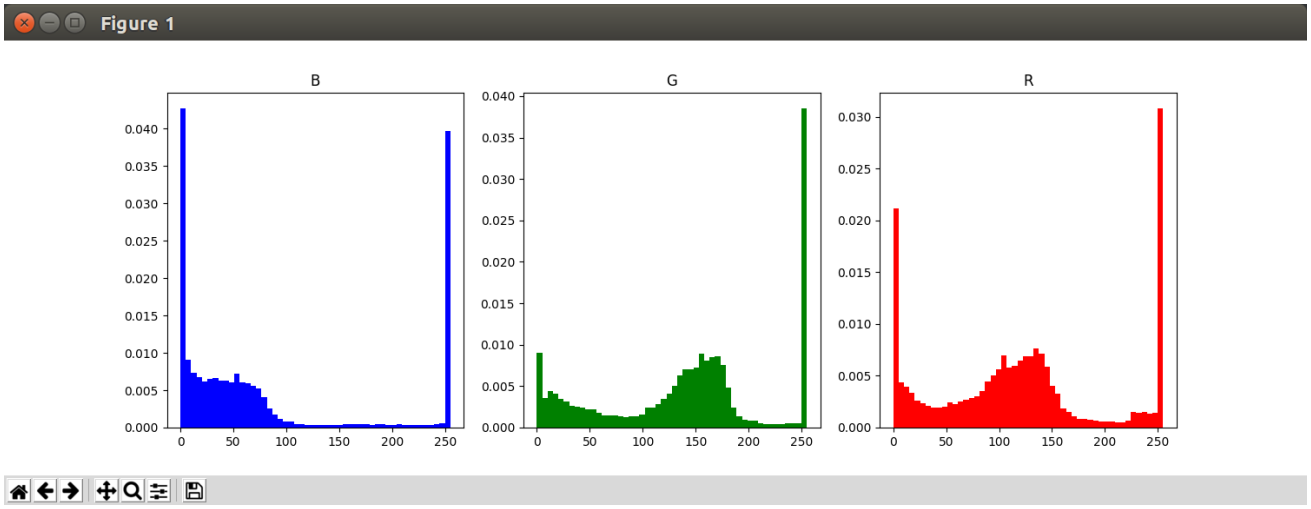


图 1

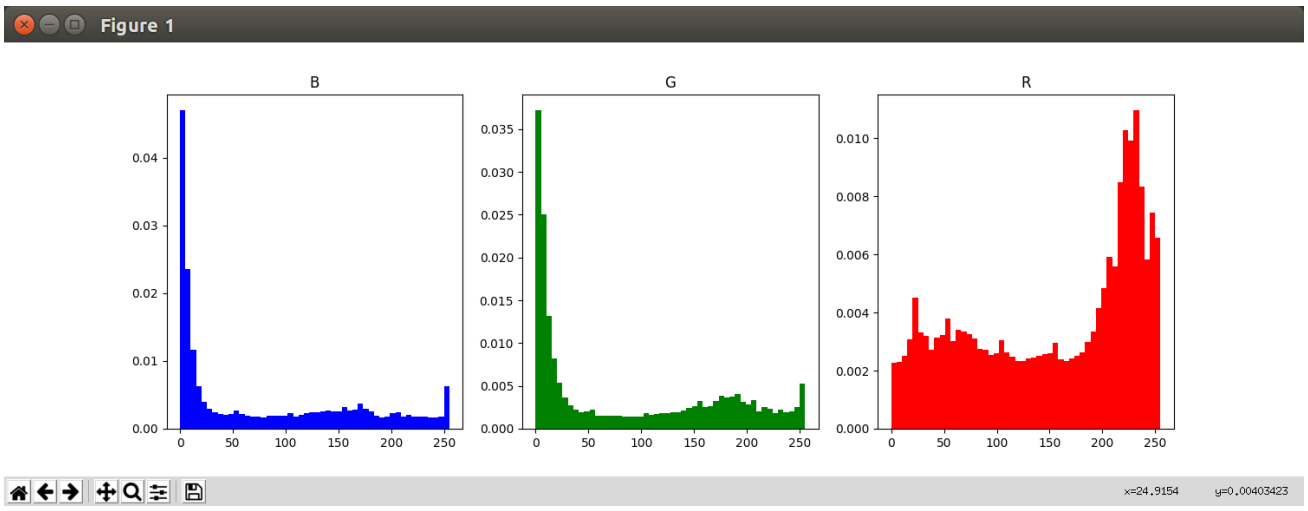


图 2

3. 实验总结

注：在编程过程中我也遇到了很多困难和 Bug，还有很多思考了很久得出的结论与解释，以及一些需要注释的内容，都在下述“2. 实验过程”中，用加粗的 NOTE 加以了说明。

此次试验让我对 OpenCV 的了解加深了，上学期计算导论使用过 OpenCV 制作过人脸识别，对 OpenCV 的有一定的认识，这次试验制作的比较快，希望之后的学习能对图像识别技术有更深的了解！
继续努力！

o(*≧▽≦)ツ！

517030910374

郭嘉宋