电类工程学导论 C 实验报告 10

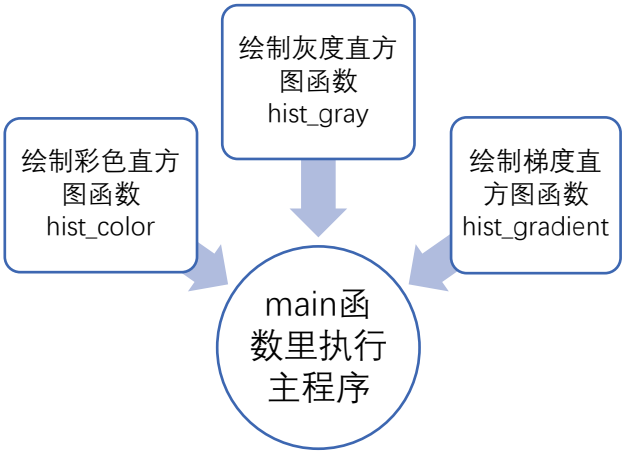
518030910406　郑思榕

一、实验准备

1. 实验环境介绍
   1) 环境：在 windows 系统中进行本次实验
   2) 语言： python 3.7
   3) 工具：本实验主要使用了开源计算机视觉库 openCV、处理大型多维矩阵的 python 库 numpy 和用作绘图的 python 库 matplotlib
2. 实验目的
   1) 将 img1.png 和 img2.png 两幅图像以彩色图像方式读入，并计算颜色直方图;
   2) 将 img1.png 和 img2.png 两幅图像以灰度图像方式读入，并计算灰度直方图和梯度直方图。
3. 实验原理

   本实验通过读取图片的所有像素点形成大型的像素矩阵，对像素矩阵进行矩阵运算得到需要的统计信息，使用绘图库的函数绘制直方图。

二、实验过程



1. ex1

```
def hist_color(src):
    image = cv2.imread(picture_prefix+src, cv2.IMREAD_COLOR)
    sum = np.sum(image.reshape(-1, 3), axis=0)
    sum = sum / np.sum(image)
    plt.title('Color Histogram of '+src.upper())
    plt.bar(['blue', 'green', 'red'], sum, width=0.3, color=['royalblue', 'springgreen', 'red'])
    plt.draw()
    plt.waitforbuttonpress(0)
    plt.close('all')
```

hist_color 函数的参数是图片文件名，文件路径写为全局变量 picture_prefix = 'E:\EE(C)\lab10\\'，所以 picture_prefix+src 即是完整的文件名。以 cv2.IMREAD_COLOR 彩色方式读入，形成三维像素矩阵 image。image 的第 i 行 j 列为图片第 i 行 j 列的彩色像素点，是一个列表，里面分别放该像素点蓝、绿、红的颜色能量。

```
[[  2   0   1]
 [  0   1   0]
 [  0   6   0]
 ...
 [ 63 153 124]
 [ 59 153 119]
 [ 57 153 116]]
```

image.reshape(-1,3)将该三维像素矩阵重新排列成右图矩阵，该函数的 3 表示形成列数为 3 的矩阵，-1 表示行数自适应。新矩阵每行的三个量分别是蓝绿红的颜色能量。使用 numpy 库里的函数 sum 进行矩阵加法，axis=0 表示做加法的方

向，这里具体含义是把所有行里的三个分量相应加起来，最终形成一个元素数量为三的列表。列表里的三个元素分别是蓝、绿、红在图片中所有能量之和。为了得到颜色分布直方图，我们还需要用 sum = sum / np.sum(image)转换为蓝绿红在图片中能量的相对比例。

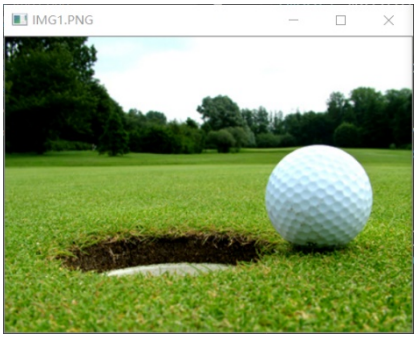
最后进行画图。下面这句代码用 matplotlib 库创建直方图。第一个参数是 x 轴上直方图中各个

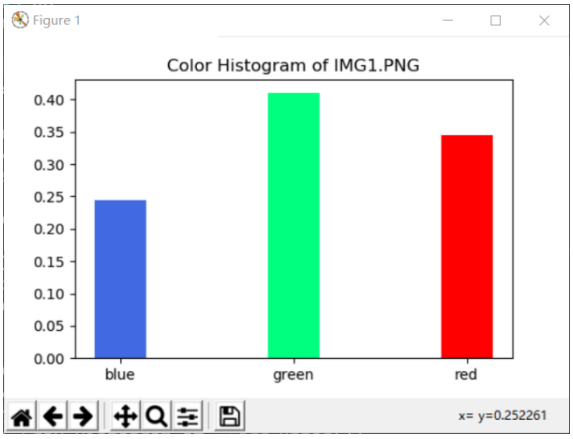
```
plt.bar(['blue', 'green', 'red'], sum, width=0.3, color=['royalblue', 'springgreen', 'red'])
```

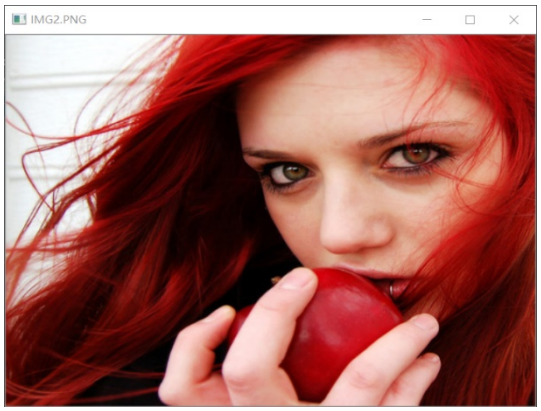
矩形的名称，sum 是各个矩形的高度，color 是各个矩形的颜色。上面三个参数都是长度为 3 的列表，元素一一对应。width 是矩阵的宽度。然后使用 plt.draw()进行直方图的绘制。

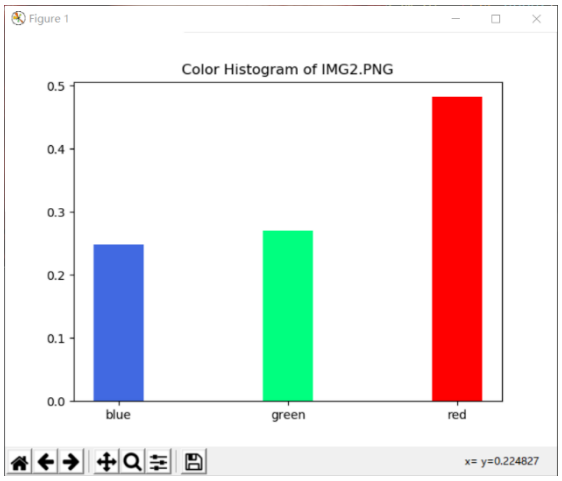
下面是我的创新点。每次展现出直方图后都要点击鼠标关闭窗口非常麻烦，不如按键盘任意键关闭窗口。因此在展示直方图后，用 plt.waitforbuttonpress(0)暂停等待按任意键继续，按任意键后执行 plt.close('all')关闭所有窗口。

成果展示：









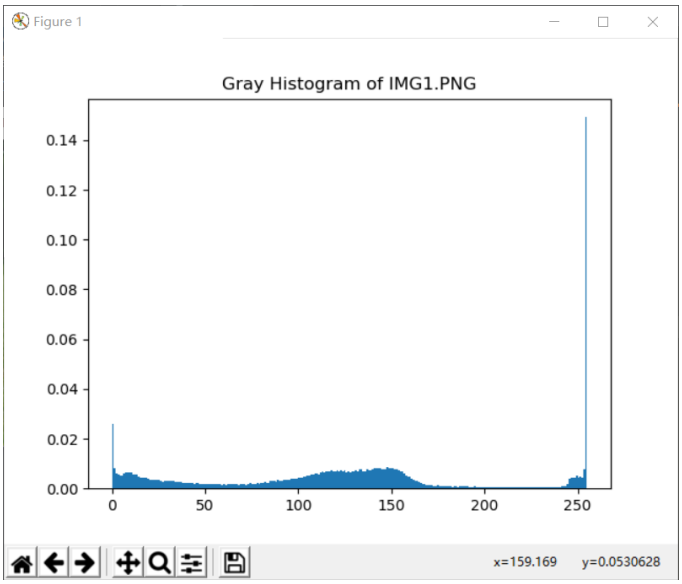
2. ex2

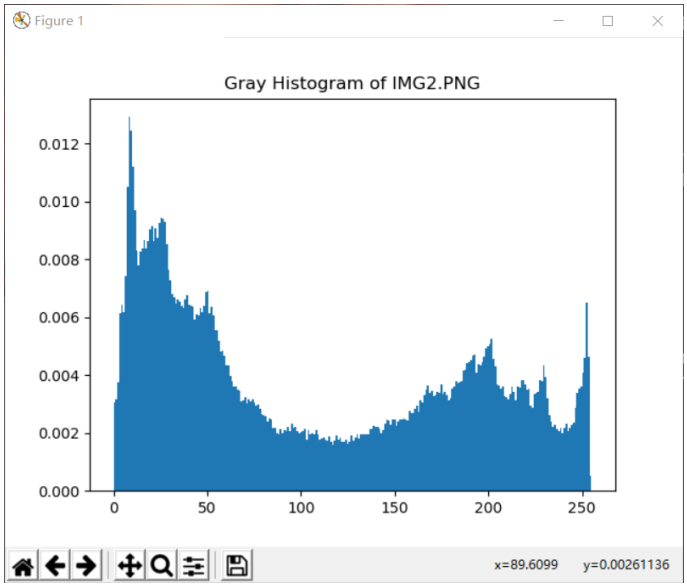
➢ 绘制灰度直方图

```
def hist_gray(src):
    image = cv2.imread(picture_prefix+src, cv2.IMREAD_GRAYSCALE)
    onedimention = image.ravel()
    plt.title('Gray Histogram of '+src.upper())
    plt.hist(x=onedimention, bins=256, density=1)
    plt.draw()
    plt.waitforbuttonpress(0)
    plt.close('all')
```

对图片使用 cv2.IMREAD_GRAYSCALE 灰度图的方式读入。因为此时 image 只有二维，求灰度直方图时只要统计像素矩阵 imag 里各个灰度值的像素数量再绘图即可。这次使用 matplotlib 里的函数 hist 进行绘制。因为 hist 函数的 x 参数是直方图中各个矩形的高度，只支持一维矩阵，所以在 hist 之前先用 ravel()函数将二维的 image 矩阵拆解为一维矩阵 onedimention。hist 的 bins 参数是 x 轴上划分的区间数，即直方图中矩形的个数。因为像素点的范围是 8 位二进制数，最大是 256，因此划分区间数 bins=256。但此时画出的直方图仍不是我们想要的，因为纵轴表示的不是各个灰度值的出现频率而是各个灰度值的累计出现数量。所以使用 density=1 使得整个直方图面积为 1，此时得到的纵轴即是灰度值的出现频率。

下面展示的分别是 img1、img2 的灰度直方图：





➢ 绘制梯度直方图

```
def hist_gradient(src):
    image = cv2.imread(picture_prefix+src, cv2.IMREAD_GRAYSCALE)
    image = image.astype('int32')
    Ix = (image[:,2:]-image[:,:-2])[1:-1,:]
    Iy = (image[2:,:]-image[:-2,:])[:,1:-1]
    gradient_matrix = np.sqrt(Ix**2+Iy**2)
    onedimention = gradient_matrix.ravel()
    plt.title('Gradient Histogram of '+src.upper())
    plt.hist(x=onedimention, bins=256, density=1)
    plt.draw()
    plt.waitforbuttonpress(0)
    plt.close('all')
```

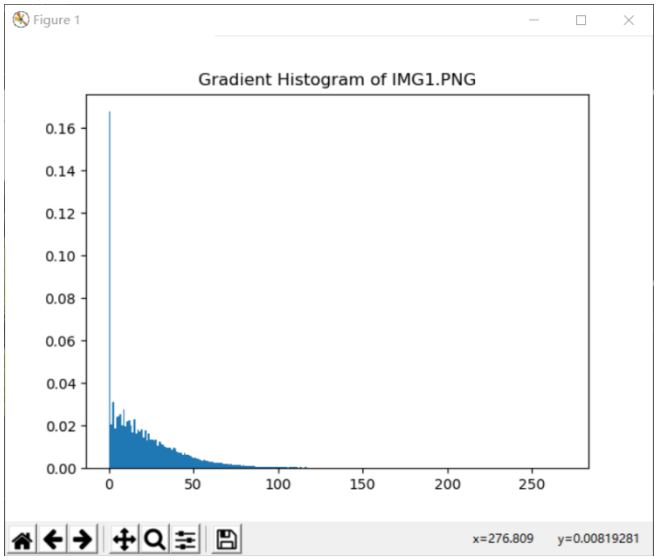
因为要得到各个像素点的梯度需要加减运算，而 imread()函数得到的矩阵默认是 8 位整数，用来加减运算容易越界而报错。因此使用 image.astype('int32')来将 8 位整数转化为 32 位整数。

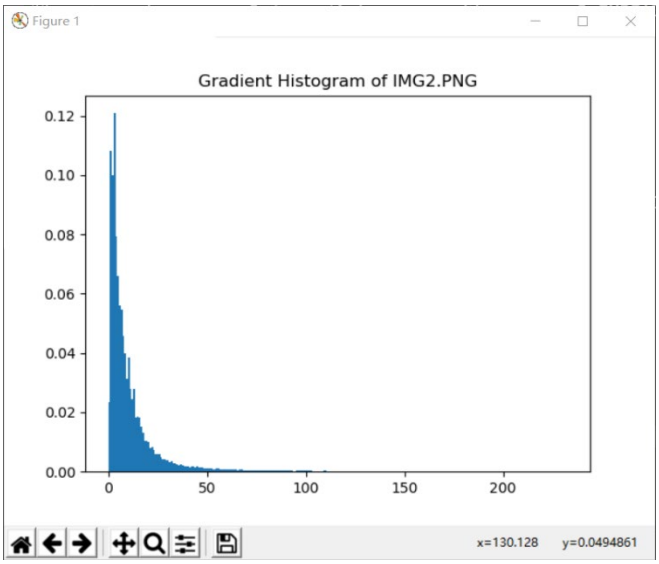
接下来是关键步骤：进行梯度的计算。由于非边缘像素点的梯度计算公式如右图公式，而 image 的[2:-2][2:-2]切片即是所有非边缘像素点，因此使用

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

`Ix = (image[:,2:]-image[:,:-2])[1:-1,:]`来获得第一行到倒数第二行间所有像素点的 X 方向梯度分量值，用 `Iy = (image[2:,:]-image[:-2,:])[:,1:-1]` 来获得第一列到倒数第二列间所有像素点的 Y 方向梯度分量值。然后根据像素点梯度值的定义，使用 numpy 库的函数 sqrt 函数进行 Ix 和 Iy 的平方和运算。最后用 ravel 函数将二维矩阵转变为一维矩阵，即可得到绘制梯度直方图所需的统计数据。后面的绘图阶段与绘制灰度直方图一致，不再赘述。

下面展示的分别是 img1、img2 的梯度直方图：





三、实验总结

1. 实验概述

本实验根据数字图像处理的基本知识，运用 opencv、numpy、matplotlib 等 python 库，实现图片的展示以及彩色直方图、灰度直方图和梯度直方图的绘制和展示

2. 实验心得

本实验我收获良多，主要有以下几点：

1) 学会了数字图像处理的基本知识，如像素的梯度、灰度直方图和彩色直方图
2) 学会了 opencv 和 numpy 的部分操作
3) 学会了如何运用 matplotlib 进行

3. 实验创新点

1) 添加了按任意键关闭图片窗口的功能。

在展示直方图后，用 plt.waitforbuttonpress(0)暂停等待按任意键继续，按任意键后执行 plt.close('all')关闭所有窗口。

2) 针对绘制三种直方图的功能相应设计三个函数，结构清晰代码简洁。

最后，衷心感谢实验中老师和各位助教的帮助!