第九次实验报告

学号：518030910308
姓名：刘文轩

一、实验准备

1、实验环境介绍

操作系统：ubuntu 14.04
语言：Python 2
IDE：Pycharm 2019.2.3

2、实验目的

2.1 了解数字图像在计算机中的储存
2.2 了解颜色直方图、灰度直方图、梯度直方图的概念以及计算方法

3、实验思路

3.1 安装 OpenCV 后，简单了解绘制直方图相关函数的使用参数
3.2 将 img1.png 和 img2.png 两幅图像以彩色图像方式读入，并计算 RGB 三种颜色所占的比例，然后画出对应图像
3.3 将 img1.png 和 img2.png 两幅图像以灰度图像方式读入，并调用函数直接获得灰度直方图，自行计算得到梯度直方图

二、实验过程

1、颜色直方图

1.1 ex1.py 的设计

我们首先获得各个颜色分量的总能量，在使用 cv2 读入图片后，使用如下代码即可，注意，np.sum()会将后面参数中所有的值全部相加，我们在此通过它获得对应颜色通道的总能量：

```
B = np.sum(src[:,:,0])
G = np.sum(src[:,:,1])
R = np.sum(src[:,:,2])
```

获得了各个颜色的能量之后，计算各个颜色能量的比例十分简单，重要的是如何将对应的图像呈现出来，我们通过

```
plt.bar(X,Y,width=1,color=['b','g','r'])
```

将对应的颜色的图像设置为它们对应的颜色，再通过

```
for xx,yy in zip(X,Y):
    plt.text(xx,yy+0.005,str(yy),ha='center')
```

为它们加上对应的值的标签。
完整代码如下：

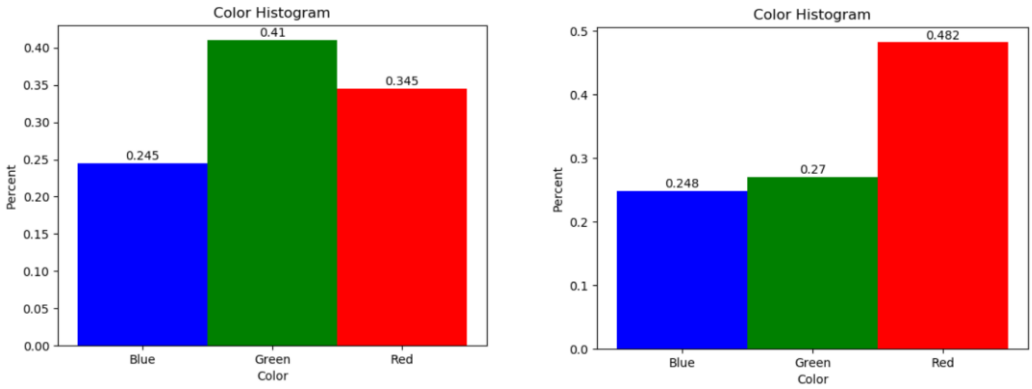
```
#encoding:utf-8
import cv2
```

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

src = cv2.imread('img1.png')
'''''
cv2.imshow("src", src)
cv2.waitKey(0)
cv2.destroyAllWindows()
'''

B = np.sum(src[:,:,0])
G = np.sum(src[:,:,1])
R = np.sum(src[:,:,2])
total = float(B+G+R)
print B/total
percent_B = round(B/total,3)
percent_G = round(G/total,3)
percent_R = round(R/total,3)
print (percent_B,percent_G,percent_R)
X = ['Blue','Green','Red']
Y = [percent_B,percent_G,percent_R]


plt.title("Color Histogram")#图像的标题
plt.xlabel("Color")#X 轴标签
plt.ylabel("Percent")#Y 轴标签
plt.bar(X,Y,width=1,color=['b','g','r'])
for xx,yy in zip(X,Y):
    plt.text(xx,yy+0.005,str(yy),ha='center')
plt.show()
```

1.2 运行结果

我们的 img1.png 和 img2.png 分别如下：

它们的颜色直方图分别如下（左图为 img1.png，右图为 img2.png）：



2、 灰度直方图

2.1 ex2_1.py 的设计

我们首先将图片以灰色图像读入：

```
1.  src = cv2.imread('img1.png',0)
```

$imread()$函数中的“0”这个参数表示以灰色图像读入，非常简洁明了。

绘图部分的核心代码如下：

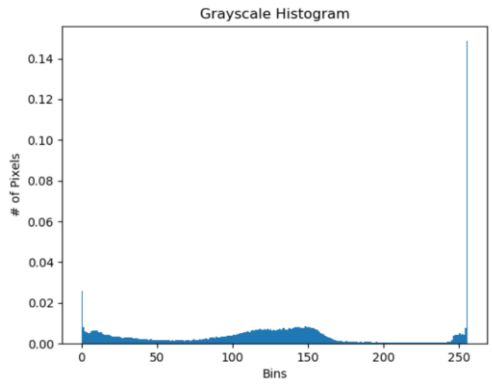
```
1.  plt.hist(src.ravel(), 256,range=[0, 256],density=1)
```

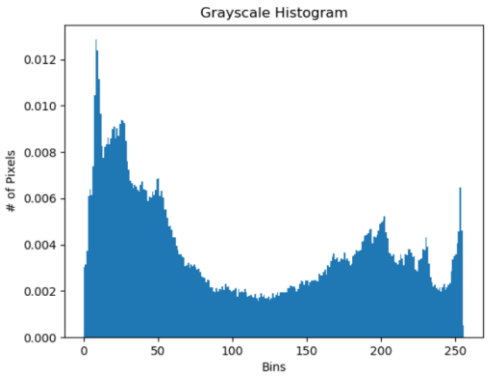
其中$src.ravel()$将多维数组转化为一维数组，256表示柱状元素的个数，$range$表示数据的变化范围，$density = 1$将数据归一化，便于我们接下来的展示比例，完整代码如下：

```
1.  #encoding:utf-8
2.  import cv2
3.  import numpy as np
4.  import matplotlib.pyplot as plt
5.
6.  src = cv2.imread('img1.png',0)
7.
8.  plt.title("Grayscale Histogram")#图像的标题
9.  plt.xlabel("Bins")#X 轴标签
10. plt.ylabel("# of Pixels")#Y 轴标签
11. plt.hist(src.ravel(), 256,range=[0, 256],density=1)
12. plt.show()
```

2.2 运行结果

还是上方所示的 img1.png 和 img2.png，生成的灰度直方图分别如下图所示，灰度直方图反映了图像的明暗程度：





3、 梯度直方图

3.1 ex2_2.py 的设计

梯度强度的范围为：

$$0 \le M(x,y) \le 255\sqrt{2} \approx 360.625$$

把梯度强度均匀分成 361 个区间，(x, y)处的像素所在区间为：

$$B(x,y) = i, \text{if } \ i \le M(x,y) < i+1, 0 \le i \le 360$$

在这里，我直接运用公式：

$$N(i) = \sum_{x=1}^{W-2} \sum_{y=1}^{H-2} B(x,y) == i \,?1:0$$

来计算落在第i个区间的总的像素个数。

包含核心计算部分的完整代码如下：

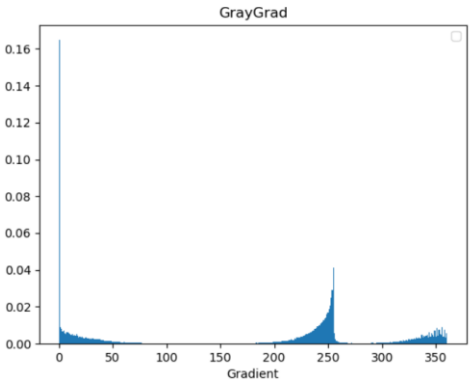
```
#!/usr/bin/env python
import cv2
import numpy
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties

img = cv2.imread("img2.png",0)

list = []

H = len(img)
W = len(img[0])

for j in range(1, H - 1):
    for k in range(1, W - 1):
        Ix = (img[j][k + 1]) - (img[j][k - 1])
        Iy = (img[j + 1][k]) - (img[j - 1][k])
        I = (Ix ** 2 + Iy ** 2) ** 0.5
```

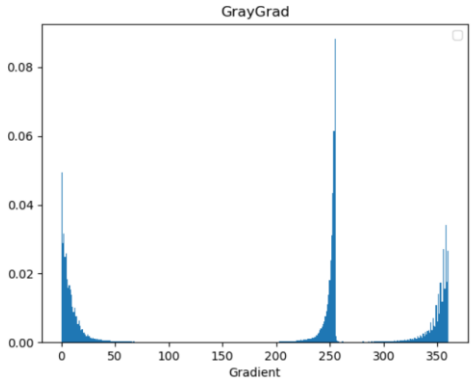
```
        list.append(I)

number = 360

plt.hist(list,360,density=1)

plt.legend()

plt.xlabel("Gradient")
plt.title("GrayGrad")

plt.show()
```

需要注意的是，区间为$1 <= x <= W-2, 1 <= y <= H-2$，因为不对最外围像素做梯度操作!

3.2 运行结果

还是上方所示的 img1 和 img2，生成的梯度直方图分别如下图所示，梯度直方图反映了图像的纹理复杂程度：





三、实验总结

1、实验概述

本次实验的主要任务，可以总结为了解、图像在数字计算机的储存，并且绘制相关的颜色直方图、灰度直方图、梯度直方图。

2、感想总结

在这次的实验中，学会了的东西有很多，其中最重要的就是提高了自己处理问题、收集相关资料、解决问题的能力，这在我们将来的学习和工作生活中都是很重要的。而具体细化开来，在本次实验中：

2.1 了解了颜色直方图、灰度直方图、梯度直方图的概念以及计算方法

2.2 学会了如何使用 cv2 以及 numpy 来读取图像并绘制直方图

3、创新点

在使得图像归一化，获取对应像素点在总区间内的比例时，我并未使用简单的读入数值然后再手动计算的方法，这样一方面可能会不够准确，另一方面也不够智能，而是通过绘图

时加入如下的代码：

```
1.  plt.hist(src.ravel(), 256,range=[0, 256],density=1)
```

由$density = 1$来使得图像自动归一化，这样过程既更加简便，得到的结果也很准确，确实十分方便。

而且原先在颜色直方图中，直接得到的图像上并不直接显示数据，这样子不利于我们获得最准确的结果，我通过：

```
1.  for xx,yy in zip(X,Y):
2.      plt.text(xx,yy+0.005,str(yy),ha='center')
```

来为每一个柱形图形加上对应的数值。

4、遇到的问题

在运行获得梯度直方图的程序时，会遇到如下的警告：

```
1.  RuntimeWarning: overflow encountered in ubyte_scalars
```

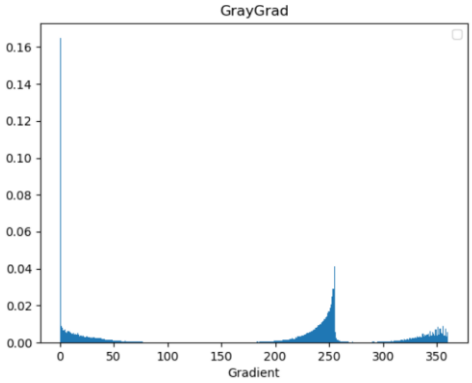
这是因为在梯度的计算公式中，需要将对应的像素相加减，这意味着可能得到的结果会溢出（如变成负值或者超过 255），我们可以通过将：

```
1.  Ix = (img[j][k + 1]) - (img[j][k - 1])
2.  Iy = (img[j + 1][k]) - (img[j - 1][k])
```

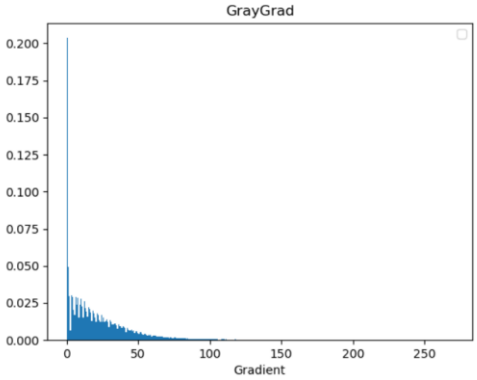
中的变量的类型强制转换为 int 类型来避免这个 warning 的出现：

```
1.  Ix = int(img[j][k + 1]) - int(img[j][k - 1])
2.  Iy = int(img[j + 1][k]) - int(img[j - 1][k])
```

然而这会导致原来的图像由：



变为：



由图可见，200 之后的数据损失了很多，这样子我认为时得不偿失的，因而我选择不改变原先的代码。