实验 8-opencv 图像特征提取

姓名：王煌基　　学号：519030910100　　班级：F1903004

一、实验概览

基于对彩色、灰色数字图像在计算机内部的表示、存储方式的认知，利用 python 语言以及适合的环境配置实现对彩色、灰色数字图像的如颜色、灰度、梯度的特征提取，计算并画出相应的直方图。

二、实验环境

1. 个人笔记本电脑
2. 操作系统：windows10 专业版
3. 使用软件：Visual Studio Code；Docker Desktop

三、实验过程

练习一：将 img1.jpg、img2.jpg 和 img3.jpg 三幅图像以彩色图像方式读入，并计算颜色直方图；

【解】

要提取图像的特征，首先需要了解数字图像在计算机内部的表示方式。简单来说，一幅图像在计算机中的表示是利用矩阵存储的，每个最小单元就是像素，且图片的具体某一个点的矩阵值就是其颜色的具体数值，这里我通过 test.py 文件来进行比较深刻的理解（输出结果为右图）：

```python
import cv2
img_bgr = cv2.imread('images/img1.jpg')
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
print(img_rgb)
```

```
[[ 26 103  51]
 [ 25 102  50]
 [ 25 102  50]
 ...
 [ 43 120  66]
 [ 40 117  63]
 [ 37 114  60]]]
```

可以发现，一张彩色图的每个像素点的 R、G、B 的值是不同的，通过不同的像素点 RGB 值综合之后形成了一张完整的图片。因此，我们想要计算颜色直方图，只需要按列的方式将每个点的具体颜色值累加，再除以总的颜色值，得到的就是该颜色的相对比例，即：

$$\text{某颜色的总能量：}E(c)=\Sigma_{x=0}^{W-1}\Sigma_{y=0}^{H-1}I(x,y,c)\text{，且相对比例：}H(c)=\frac{E(c)}{\Sigma_{i=0}^{2}E(i)}$$

通过这个思路，我们可以较为轻松地得到相应代码(color.py)：

```python
import cv2
from matplotlib import pyplot as plt
filename = 'img1'
in_img = cv2.imread('images/'+filename+'.jpg')
image = cv2.cvtColor(in_img, cv2.COLOR_BGR2RGB)
```

```
# 记 0
blue, green, red = 0, 0, 0
# 宽 len(image),长 len(image[0])
for i in range(0,len(image)):
    for j in range(0,len(image[0])):
        # 读入的图片按照 r,g,b 的顺序
        blue += image[i][j][2]
        green+= image[i][j][1]
        red  += image[i][j][0]
# 将最后的总量相加
tot = blue + green + red
# 将具体比例存入 y 向量中
y = [blue*1.0/tot, green*1.0/tot, red*1.0/tot]
# 画出柱状图，其中横坐标为 b,g,r，纵坐标为 b,g,r 所占的比例
plt.bar(x=['blue','green','red'],height=y,width=1.0,color=['b','g','r'])
# color 参数传入颜色列表，可以在一幅图中显示不同颜色
# 保存图片
plt.savefig(filename+'.png')
plt.show()
```

通过这样的代码实现，我们就可以完成我们的练习一。

练习二：将 img1.jpg、img2.jpg 和 img3.jpg 三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；

【解】

与练习一相似的，为了加深对灰色数字图像在计算机中表示的理解，我通过 test.py 来加深印象：

```
import cv2
img_bgr = cv2.imread('images/img1.jpg')
img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
print(img_gray)
```

```
[[171 164 121 ... 196 196 196]
 [163 158 120 ... 196 196 196]
 [148 146 119 ... 196 196 196]
 ...
 [ 65  70  77 ...  90  92  93]
 [ 69  71  75 ...  80  76  72]
 [ 74  73  73 ...  91  88  85]]
```

可以发现，与彩色图不同的是，由于少了 RGB 的限制之后，image 的维度是二维的，意味着我们调用起来就更加简洁了。

根据特征提取等性质的研究，可以发现灰度图的灰度性质满足：

灰度值为i的像素个数 $\mathbf{N}(i) = \Sigma_{x=0}^{W-1}\Sigma_{y=0}^{H-1} I(x,y) == i ? 1 : 0$，**且相对比例：** $H(i) = \frac{N(i)}{\Sigma_{j=0}^{255} N(j)}$

利用同样的方式，我们可以轻松地得到灰度图的实现代码：

```
import cv2
from matplotlib import pyplot as plt
filename = 'img1'
in_img = cv2.imread('images/'+filename+'.jpg')
image = cv2.cvtColor(in_img, cv2.COLOR_BGR2GRAY)
# 创建 0~255 的灰度值像素点数
```

```
gray = [0]*256
tot  = 0
for i in range(0,len(image)):
    for j in range(0,len(image[0])):
        # 直接将 image[i][j]点的灰度像素值加到我们的 gray 上
        gray[image[i][j]] += 1
        tot += 1
# 建立具体比例，存入 y 中
y = [i*1.0/tot for i in gray]
plt.bar(x=range(0,256,1),height=y,width=1)
plt.savefig(filename+'gray.png')
plt.show()
```

从而我们的灰度直方图得到解决。

对于梯度直方图，此时的处理需要注意边界问题（防止越界的情况出现），所以对于水平的 x 我们的取值范围是 1 至 W-2，对于竖直的 y 我们的取值范围是 1 至 H-2，根据特征提取等性质的研究，可以发现灰度图的梯度性质满足：

$$\textbf{灰度值为i的x方向梯度满足：}\mathbf{I_x}=\frac{\partial \boldsymbol{I}(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{x}}=\boldsymbol{I}(\boldsymbol{x}+\mathbf{1},\boldsymbol{y})-\boldsymbol{I}(\boldsymbol{x}-\mathbf{1},\boldsymbol{y})\in[-\mathbf{255},\mathbf{255}]$$

$$\textbf{灰度值为i的y方向梯度满足：}\mathbf{I_Y}=\frac{\partial \boldsymbol{I}(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{y}}=\boldsymbol{I}(\boldsymbol{x},\boldsymbol{y}+\mathbf{1})-\boldsymbol{I}(\boldsymbol{x},\boldsymbol{y}-\mathbf{1})\in[-\mathbf{255},\mathbf{255}]$$

$$\textbf{梯度强度满足：}\mathbf{M(x,y)}=\sqrt{\boldsymbol{I_x}(\boldsymbol{x},\boldsymbol{y})^{\mathbf{2}}+\boldsymbol{I_y}(\boldsymbol{x},\boldsymbol{y})^{\mathbf{2}}}\in[\mathbf{0},\mathbf{255}\sqrt{\mathbf{2}}]\approx[\mathbf{0},\mathbf{360}]$$

$$\textbf{均分M到361个区间有：}\mathbf{B(x,y)}=\boldsymbol{i}\ \ \boldsymbol{if}\ \boldsymbol{i}\leq\boldsymbol{M}(\boldsymbol{x},\boldsymbol{y})<\boldsymbol{i}+\mathbf{1},\mathbf{0}\leq\boldsymbol{i}\leq\mathbf{360}$$

最后，只需要统计第 i 个区间的 B(x,y)的数目，我们就可以画出该灰度图的梯度直方图：

```
Ix, Iy = np.zeros((len(image),len(image[0]))), np.zeros((len(image),len(image[0])))
M, B = np.zeros((len(image),len(image[0]))), np.zeros((len(image),len(image[0])))
N = [0]*361
tot = 0
for i in range(1,len(image)-1):
    for j in range(1,len(image[0])-1):
        # Ix 是 x 方向的梯度，Iy 即 y 方向的
        Ix[i][j] = int(image[i+1][j])-int(image[i-1][j])
        Iy[i][j] = int(image[i][j+1])-int(image[i][j-1])
        # M 为梯度强度
        M[i][j]=(Ix[i][j]**2+Iy[i][j]**2)**0.5
        # B 实际上没有用处，这里只是为了清楚表示
        B[i][j]=math.floor(M[i][j])
        # N 记录实际的 i 区间内的值
        N[math.floor(M[i][j])] += 1
        tot += 1
y = [i*1.0/tot for i in N]
```

从而我们的练习二得以解决。

四、实验问题及解决

问题一：如何判断输入图片的 R、G、B 的维度？

【解】

查阅了相关资料后发现，opencv2 与普通图片的颜色存储方式是不同的，前者为 BGR 后者为 RGB，为了防止由于这样的问题导致颜色判定时出错，我通过设定纯色图来判断其输出的方式确定 R、G、B 的维度位置（\test_images\img4-6.jpg）：

通过打开不同的 img 文件，输出它们的读取结果来判断维度的具体颜色即可：

```
1. import cv2
2. img_bgr = cv2.imread('images/img4.jpg')
3. print(img_bgr)
```

输出结果（从左往右分别是 img4.jpg，img5.jpg，img6.jpg）：

```
[[254   0   0]
 [254   0   0]
 [254   0   0]
 ...
 [254   0   0]
 [254   0   0]
 [254   0   0]]]
```

```
[[  1 255   0]
 [  1 255   0]
 [  1 255   0]
 ...
 [  1 255   0]
 [  1 255   0]
 [  1 255   0]]]
```

```
[[  0   0 254]
 [  0   0 254]
 [  0   0 254]
 ...
 [  0   0 254]
 [  0   0 254]
 [  0   0 254]]]
```

可以很容易发现，读入的图片颜色维度应该是 BGR。

问题二：PPT 中柱状图上可以出现相应的数据，怎么实现这个功能？

【解】

通过查阅相关资料后，我发现可以利用 pyplot 中的 text 函数来进行相应的文本添加：

```
1. plot = plt.bar(x=['blue','green','red'],height=y,width=1.0,color=['b','g','r'])#color 参数传入颜色列表，可以在一幅图中显示不同颜色
2. for rect in plot:
3.     height = rect.get_height()
4.     plt.text(rect.get_x() + rect.get_width() / 2, height,  '%f'%height, ha="center", va="bottom")
```

从而我们可以实现柱状图上出现相应数据的功能。

问题三：在计算 Ix，Iy 梯度的时候，出现了 overflow encountered in ubyte_scalars 的报错问题，如何解决这个问题？

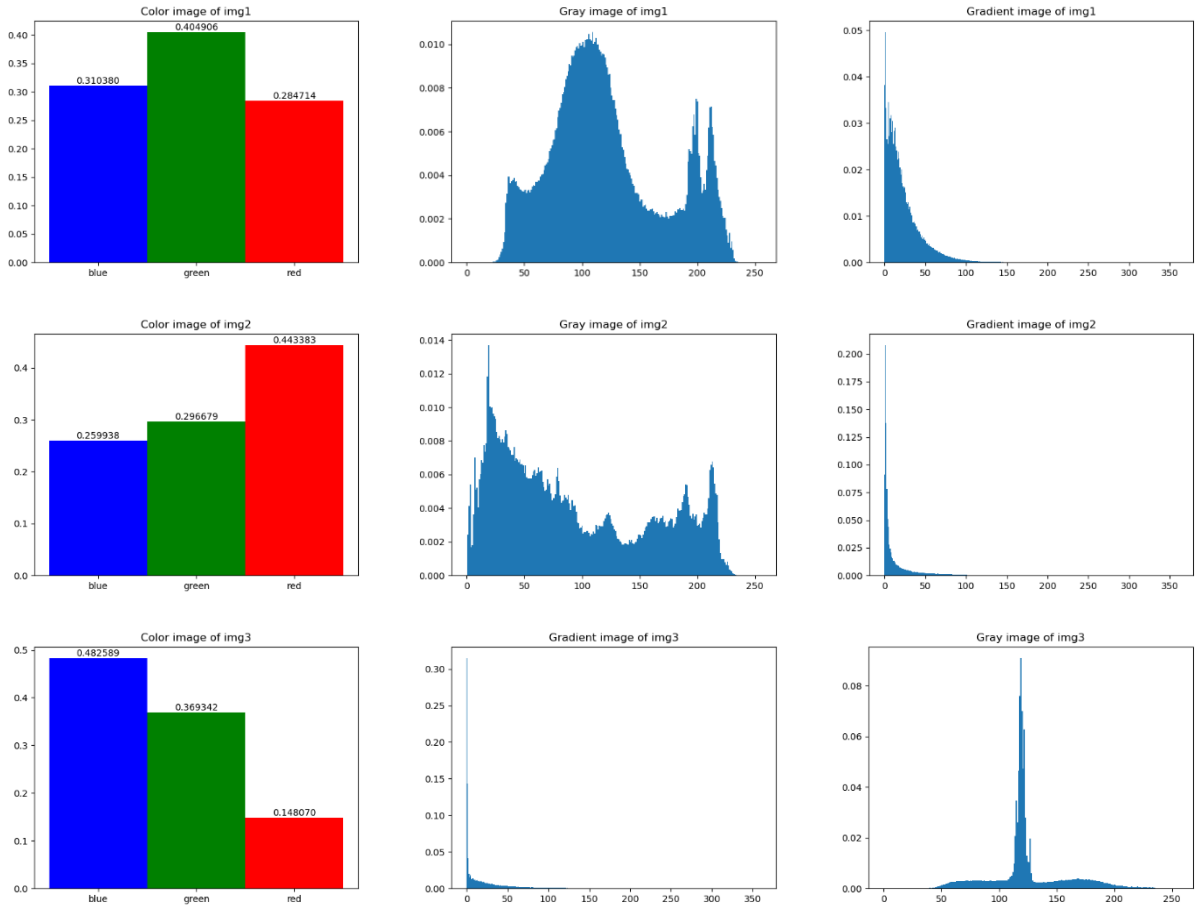
【解】

在处理图像计算梯度的时候，由于涉及到两幅图像像素值之间的加减运算，并且图像像素值是 ubyte 类型，ubyte 类型数据范围为 0~255。因此如果运算时出现负值或超出 255，则会抛出异常，此处的解决方法是强行将 ubyte 类型转换成 int 类型来避免这样的错误。

```
1.          Ix[i][j] = int(image[i+1][j])-int(image[i-1][j])
2.          Iy[i][j] = int(image[i][j+1])-int(image[i][j-1])
```

五、实验结果

从上往下依次是 img1.jpg、img2.jpg、img3.jpg 的颜色直方图、灰度直方图、梯度直方图：



【为了确保实验的准确性，我将 PPT 内出现的图片也一并进行了测试，保证了准确度，具体可见\result 下的 png 文件】

六、实验体会

本次接触的 opencv 是令我既熟悉又陌生的，熟悉在于在大一上的《计算导论》课程大作业中我们利用 opencv 进行了人脸的图像识别以及一定的操作，但是那毕竟是比较大的，对于细节上的操作并不像本次实验那么细致，所以在本次实验中我对数字图像在计算机中的存储方式的理解进一步加深了，并且能够提取一定的图像特征以便之后实验的应用。

七、拓展思考

1. 示例代码中的 cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB) 的作用是什么？

将 BGR 编码的图片转为 RGB 的编码。

2. 如何使得 pyplot 正确显示灰度图的颜色？

通过查阅相关资料，发现 pyplot 可以利用 cmap 来限定输出的图像的属性，因此可以通过限定 cmap=’gray’的方式来正确显示灰度图的颜色。

```
1. import cv2
2. from matplotlib import pyplot as plt
3. filename = 'img2'
4. in_img = cv2.imread('images/'+filename+'.jpg')
5. #将 BGR 转为灰度图
6. image = cv2.cvtColor(in_img, cv2.COLOR_BGR2GRAY)
7. plt.title('Gray image of '+filename)
8. # 显示图像
9. # 用 gray 灰度图显示
10. plt.imshow(image, cmap='gray')
11. plt.show()
```

运行结果：