电工导 C 第九次实验报告

姓名: 宋士祥　　学号:521030910013　　班级:F2103001

2022 年 11 月 13 日

1　实验概览

本次实验简单学习了图像的相关知识OpenCV 的相关知识。

对于图像的表现，我们有灰度图像和彩色图像两种情形。对于灰度图像而言，图像只由一个 $W \times H$ 的灰度值矩阵组成。W 为构成宽度的像素数目，H 为构成高度的像素数目。灰度值的取值范围为 $[0, 255]$，数值越高意味着颜色越淡。显然，0 代表黑色，255 代表白色。而对于彩色图像而言，其由一个 $W \times H \times 3$ 的高维矩阵构成，分别由红色 (R)、绿色 (G)、蓝色 (B) 各自组成其中的颜色能量。特别地，对于彩色图像与灰色图像的转化，我们有一常用公式：

$$\text{Gray} = 0.299R + 0.587G + 0.114B$$

对于图像的分析，我们一般采用图像直方图的方式来反应他们的特征。对于图像直方图，本次实验我们需要分别得到图像的灰度直方图、灰度梯度直方图和颜色直方图。颜色直方图反应图像的主体色调，灰度直方图反应灰度值的分布情况，而

OpenCV 是一个跨平台的图像分析库。由一系列 C 函数和少量 C++ 类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

2　实验环境

本次实验采用所需的实验环境如下：

- Docker 中的sjtumic/ee208 镜像
- Python3（使用 VSCode 编译）
- numpy 扩展。
- OpenCV 环境。(均在 SJTUEE208 镜像中给出)

3　绘制颜色直方图

3.1　问题重述与代码说明

对于一个彩色图片，其由一个 $W \times H \times 3$ 的高维矩阵构成。一个颜色直方图，就是来衡量三种颜色在图片中所占的比重。

我们知道，一个颜色的总能量 $E(c) = \sum_{i=0}^{W-1}\sum_{j=0}^{H-1} I(x,y,c)$。因此我们容易得到每一种颜色的占比：

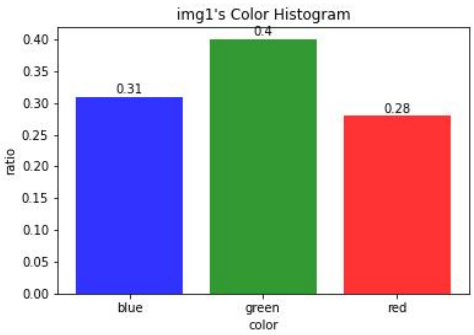
$$H(c) = \frac{E(c)}{\sum_{i=0}^{2} E(i)}$$

我们采取的方式是直接对某个部分涉及蓝色、绿色、红色的强度直接求和，然后比较大小。由于OpenCV 的存储为 $W \times H \times 3$ 的形式，因此我们不能直接使用sum 函数。对应的代码如下：

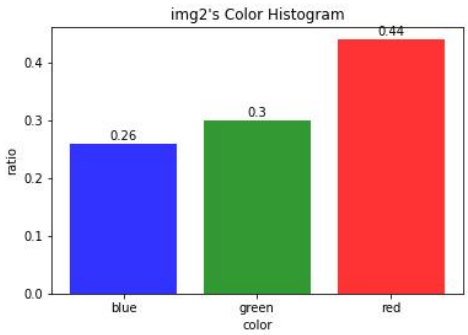
```
    img = cv2.imread('images/img{}.jpg'.format(index),cv2.IMREAD_COLOR)
    blue, green, red = 0, 0, 0
    # 记录数值RGB
    for i in range(len(img)):
        for j in range(len(img[0])):
            blue  += img[i][j][0]
            green += img[i][j][1]
            red   += img[i][j][2]
    # 计算比例并保留两位小数
    Sum = sum((blue, green, red))
    color_list = [round(blue/ Sum,2), round(green/ Sum,2), round(red/ Sum,2)]
```

对于直方图的绘制，我们直接利用matplot 自带的bar 进行处理。考虑这三个函数功能一致，我们直接将这一内容封装到一个函数中。

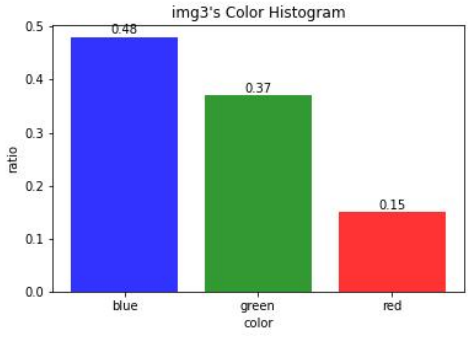
3.2 结果展示



(a) img1



(b) img2



(c) img3

3.3 问题探究

3.3.1 Matplotlib 中 bar 和 hist 的区别

我们知道，利用Matplotlib 的bar 函数与hist 函数均可绘制直方图。我们这里采用bar 函数的原因是为了颜色的区分（hist 函数不支持这一操作）。

3.3.2 如何在图片上方添加数字

我们采用添加text 的方式（也即图片注释）的方式对图片进行打标签的处理。需要注意的是，需要控制标签与 bar 之间的距离。考虑各数值取值在 $[0,1]$ 之间，为了图片的美观，我们一般不宜取值过大。本次实验取值为 0.01。

4 绘制灰度直方图

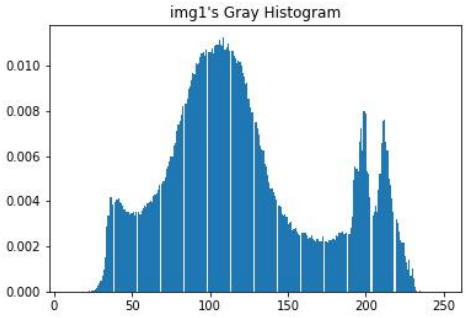
4.1 问题重述与代码说明

灰度图像 $I(x,y)$ 的灰度直方图定义为各灰度值像素数目的相对比例。

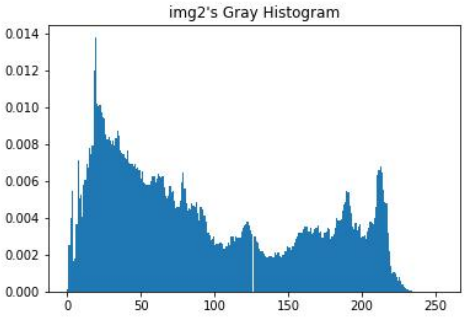
利用numpy 的ravel() 函数，我们可以很轻松的将二维矩阵降到一唯。同时利用matplotlib 自带的hist 函数，我们可以直接算出其对应比例。相关代码如下：

```
def make_gray_histogram(index):
    img = cv2.imread('images/img{}.jpg'.format(index),cv2.IMREAD_GRAYSCALE)
    ravel_img = img.ravel()
    #将灰度图转化为一维数组 并获取直方图
    plt.hist(ravel_img, 256, density=True)
    plt.title('img{}\'s Gray Histogram'.format(index))
    #显示直方图
    plt.savefig("gray-hist/img{}.jpg".format(index))
    plt.show()
```

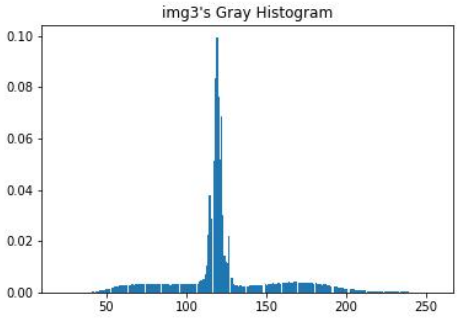
4.2 结果展示



(d) img1



(e) img2



(f) img3

4.3 问题探究

关于各部分比例的计算

我们知道，最终的结果表示为各部分灰度值的比例。`hist` 函数中自带一个计算比例的方法。而若我们自己写一个类似的方法，由于数量级过小，python 会自动置零。

5 灰度梯度直方图的计算

5.1 问题重述与代码说明

灰度梯度直方图是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。纹理越简单，梯度值越靠近 0，说明图片纹理越简单。

我们有：

$$\nabla I(x,y) = \left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y} \right)$$

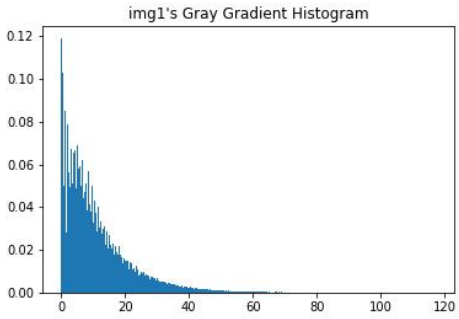
我们有：

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

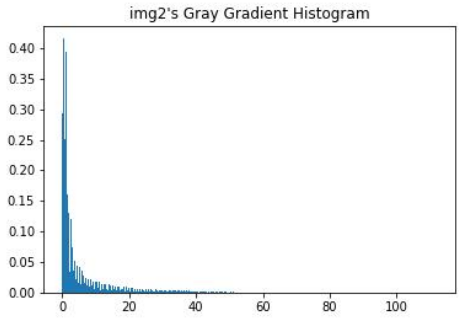
在numpy 中，`gradient` 函数可以自动求梯度，因此我们直接调用这一函数，并采取与第四部分相类似的内容即可。相关代码展示如下：

```
def make_gray_gradient(index):
    img = cv2.imread('images/img{}.jpg'.format(index),cv2.IMREAD_GRAYSCALE)
    img_gradient = np.gradient(img)
    img_gra = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            img_gra[i][j] = math.sqrt(pow(img_gradient[0][i][j],2) + pow(img_gradient[1]
                                                                [i][j],2))
    plt.hist(img_gradient.ravel(), 360 , density=True)
    plt.title('img{}\'s Gray Gradient Histogram'.format(index))
    #显示直方图
    plt.savefig("gradient/img{}.jpg".format(index))
    plt.show()
```

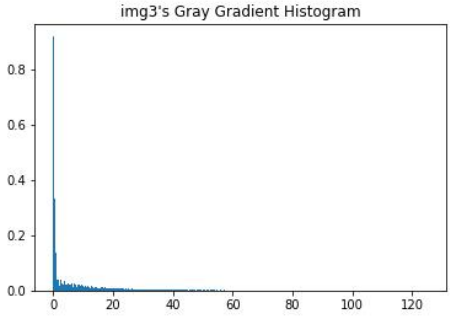
5.2 结果展示



(g) img1



(h) img2



(i) img3

5.3 问题分析

如果手搓一个求梯度算法呢？

如果我们尝试自己写一个算法函数，一方面，我们需要调用前后左右的变量共 8 次，并需要判断是否越界等。算法复杂度 $\Theta(H \times W)$，外加判断等因素，需要执行的程序条数过多。外加python 本身计算速度并不快，因此调用numpy 的梯度算法是一个比较合理的做法。

6 拓展思考

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

这是由于，cv2 在读取图片时，是按照 BGR 的通道顺序读取，而我们在转回原图像时，读取通道顺序为 RGB。如果不引入该函数，原本的蓝色通道和红色通道会反置，这样会导致原本的红色图像偏蓝等情况出现。

Ref:https://blog.csdn.net/x1131230123/article/details/119761336

6.2 如何使得 pyplot 正确显示灰度图的颜色？

通过查阅资料得知，pyplot 在调用灰度图像时会默认调用绿色通道，使得最终颜色发绿。为了避免此类问题的出现，我们需要强制其使用灰色通道进行读取，只需在imshow 函数中添加cmap='gray' 的参数即可。

Ref:https://blog.csdn.net/qq_42951560/article/details/124355446

6.3 对颜色直方图的进一步改进

本次实验中我们只需要反映出各个图像三种颜色的比例即可，但是这样会损失一部分信息。而我们希望能读取出更多的信息。

一个比较好的做法是利用`split()` 函数，将三个通道的颜色数据分离到三个图像中，分别生成三个图像的直方图，这样能反映出更多的信息。

6.4 彩色图像转灰度

在本文第 1 章中已经介绍了彩色图像转灰度图像的公式。这一公式是根据人的亮度感知系统调节出来的参数，是目前普遍采用的标准化公式。