Lab1-OpenCV

1. 实验概览

1.1. 实验目的

1. 学习数字图像在计算机中的表示和存储
2. 学习图像特征的提取
3. 学习OpenCV-python的使用
4. 实验过程简介：

   本次实验使用OpenCV-python及numpy库中的函数分别对图像进行彩色读入与灰色读入，并计算彩色图像的颜色直方图、灰色图像的灰度直方图和梯度直方图。最后使用matplotlib库绘制并保存这几种图像。

3. 解决思路

3.1. 练习一

- 解决思路：先使用os库获取待读入图像的绝对路径，然后使用`cv2.imread('path/to/img', cv2.IMREAD_COLOR)`方法以彩色图像的方式迭代读入三幅图像；再使用`cv2.split()`方法分离BGR三个通道的颜色分量，并用numpy库中`np.sum()`方法对颜色分量求和，并得到每个分量的相对比例，用matplotlib库中`plt.bar()`函数绘制条形图（直方图）。核心代码如下：

```
# Get the current directory and image path
current_working_dir = os.path.dirname(__file__)
imgs_path = os.path.join(current_working_dir, 'images')

# Image file names to process
images = ['img1.jpg', 'img2.jpg', 'img3.jpg']

for img in images:
    # Get the base name of the image
    base_img_name = os.path.splitext(img)[0]

    # Read the image in color
    img_color = cv2.imread(os.path.join(imgs_path, img), cv2.IMREAD_COLOR)

    # Calculate each color channel energy and total energy
    blue_channel, green_channel, red_channel = cv2.split(img_color)
    blue_energy = np.sum(blue_channel)
    green_energy = np.sum(green_channel)
    red_energy = np.sum(red_channel)
    total_energy = blue_energy + green_energy + red_energy

    # Calculate each color's relative energy ratio
    blue_energy_ratio = float(blue_energy / total_energy)
    green_energy_ratio = float(green_energy / total_energy)
    red_energy_ratio = float(red_energy / total_energy)

    # Plot the color energy ratios in histogram
    energy_ratios = [blue_energy_ratio, green_energy_ratio, red_energy_ratio]
```

```
    plt.bar(['Blue', 'Green', 'Red'], energy_ratios, color=['blue', 'green',
'red'])
```

3.2. 练习二

- 解决思路：同练习一读取图像，然后使用`cv2.imread('path/to/img', cv2.IMREAD_GRAYSCALE)`方法以灰度图像方式读入三幅图像，然后使用matplotlib中的直方图函数`plt.hist(img_gray.ravel(), bins=256, range=(0, 256), color='black')`计算并绘制灰度直方图，其中`ravel()`方法用于将二维灰度图像数组转换为一维数组以便于计算直方图。对于梯度直方图，这里使用Sobel算子（即使用`cv2.Sobel()`方法）对灰度图像进行梯度计算，并得到梯度强度，然后类似计算灰度直方图的方法计算并绘制梯度直方图。核心代码展示如下：

```
for img in images:
    # Read the image in grayscale
    img_gray = cv2.imread(os.path.join(imgs_path, img), cv2.IMREAD_GRAYSCALE)

     # Use matplotlib.pyplot to get and plot gray histogram
    plt.hist(img_gray.ravel(), bins=256, range=(0, 256), color='black')

    # Calculate gradient histogram using Sobel operator
    grad_x = cv2.Sobel(img_gray, cv2.CV_64F, 1, 0, ksize=3)
    grad_y = cv2.Sobel(img_gray, cv2.CV_64F, 0, 1, ksize=3)
    grad_magnitude = np.sqrt(grad_x**2 + grad_y**2)

    # Plot gradient histogram using matplotlib.pyplot
    plt.hist(grad_magnitude.ravel(), bins=361, range=(0, 360), color='black')
```

4. 代码运行结果

4.1. 练习一

得到的三幅颜色直方图为：

4.2. 练习二

得到的三幅灰度直方图为：

得到的三幅梯度直方图为：

5. 实验结果分析与思考

5.1. 练习一结果分析

- 从三幅颜色直方图中可以看出：
    - 图像1绿色分量的能量的相对比例最大，蓝色分量的相对比例略大于红色分量；因此图像1整体色调应呈绿色，与图像的视觉特征相符合。

- 图像2的红色分量的能量的相对比例最大，绿色分量能量的相对比例略大于蓝色分量；因此图像2整体色调应呈红色，与图像夕阳西下的视觉特征相符合。
- 图像3的蓝色分量的能量的相对比例最大，绿色分量能量的相对比例略大于红色分量；因此图像3整体色调应成蓝色，辅以绿色，红色的部分最少，与图像3的视觉特征相符合。

5.2. 练习二结果分析

- 从三幅灰度直方图中可以看出：
  - 图像1的灰度分布比较均匀，但是在灰度值为100及200左右有灰度值的集中分布，即灰度值主要分布在中段区域，说明图像1的画面较为明亮。
  - 图像2的灰度分布也较为均匀，灰度主要分布在灰度值较小的区域，但是在中段区域与高段区域也有较多的分布，说明图像2在某些区域的画面比较灰暗，其他区域的画面较为明亮。
  - 图像3的灰度基本上分布在灰度值100~200之间，即集中分布在中段区域，且在140~150范围左右有非常集中的分布，说明图像3的画面较为明亮，且明暗程度几乎没有差别。
- 从三幅梯度直方图中可以看出：
  - 图像1的梯度强度分布较为分散，整体范围较宽，峰值较低。因此图像可能包含丰富的细节和纹理，梯度分布相对均匀，可能是内容较为复杂。
  - 图像2的梯度强度分布集中在低值区域，图像可能较平滑，边缘和细节较少，主要由平坦区域组成。
  - 图像3的梯度分布极度集中在最小值附近，峰值极高。图像可能非常平坦，几乎没有显著的边缘或变化，与图像特征符合。

5.3. 结果综合分析

1. **图像1**：整体色调成绿色，同时也有稍少量的蓝色和红色色调，其画面较为明亮，内容比较复杂，包含丰富的细节和纹理。
2. **图像2**：整体色调成红色，在某些区域比较明亮，在其他区域比较灰暗，亮度分布较为均匀，图像比较平滑，细节比较少。
3. **图像3**：整体色调成蓝色，也有一些绿色和红色，图像较为明亮，且明暗程度相差不大，图像几乎没有显著的变化，非常平坦。

5.4. 思考题

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

   `cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)` 的作用是将图像的颜色空间从 **BGR** 转换为 **RGB**。

   在 OpenCV 中，`cv2.imread` 默认会以 BGR（蓝-绿-红）顺序加载图像，而 Matplotlib 显示图像时使用的是 RGB（红-绿-蓝）颜色顺序。如果不进行这种转换，图像的颜色在 Matplotlib 中显示会出现偏差，比如红色和蓝色互换。

   通过 `cv2.cvtColor` 进行 BGR 到 RGB 的转换，可以确保图像颜色在 Matplotlib 中正确显示。

2. 如何使得pyplot正确显示灰度图的颜色?

   为了使 `pyplot` 正确显示灰度图的颜色，可以在调用 `plt.imshow()` 时指定参数 `cmap='gray'`，这样 Matplotlib 会将图像按照灰度图的正确方式显示。

6. 实验感想

6.1. 遇到的问题

- 一开始使用OpenCV-python中的函数来计算颜色直方图时，错误地使用了`cv2.calchist()`函数，导致画出的直方图并不是想要得到的直方图，之后经过仔细学习之后，发现应先分离每个颜色分量，在对其进行求和，然后才能得到每种颜色的相对比例。

6.2. 经验总结

- 盲目使用未明确功能的函数可能导致错误结果，应先查阅官方文档和示例代码，搞清楚函数的用途和参数含义后再进行使用。