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。

2. 实验环境

Python 3.12.4，使用OpenCV、Matplotlib和Numpy库

3. 解决思路

3.1. 练习一

- 解决思路：将image文件夹与py文件置于同一目录下，获取待读入图像的路径`'images/' + img`，然后使用 `cv2.imread(img_path, cv2.IMREAD_COLOR)`方法以彩色图像的方式迭代读入三幅图像；再使用NumPy 切片的方法分离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：

```
images = ['img1.jpg', 'img2.jpg', 'img3.jpg']  # List the images
# Using Numpy slicing
def color_split(img):
    b = img[:, :, 0]  # Blue
    g = img[:, :, 1]  # Green
    r = img[:, :, 2]  # Red
    return b, g, r

for img in images:
    img_path = 'images/' + img   # Get the path
    # Color Histogram
    img_color = cv2.imread(img_path, cv2.IMREAD_COLOR)
    b, g, r = color_split(img_color)
    # Calculate color energy using Numpy
    blue_energy = np.sum(b)
    green_energy = np.sum(g)
    red_energy = np.sum(r)
    color_list = [blue_energy, green_energy, red_energy]
    total = sum(color_list)
    color_ratio = []
    for energy in color_list:
        ratio = float(energy / total)
        color_ratio.append(ratio)
    plt.bar(['Blue', 'Green', 'Red'], color_ratio, color=['blue', 'green', 'red'])
```

3.2. 练习二

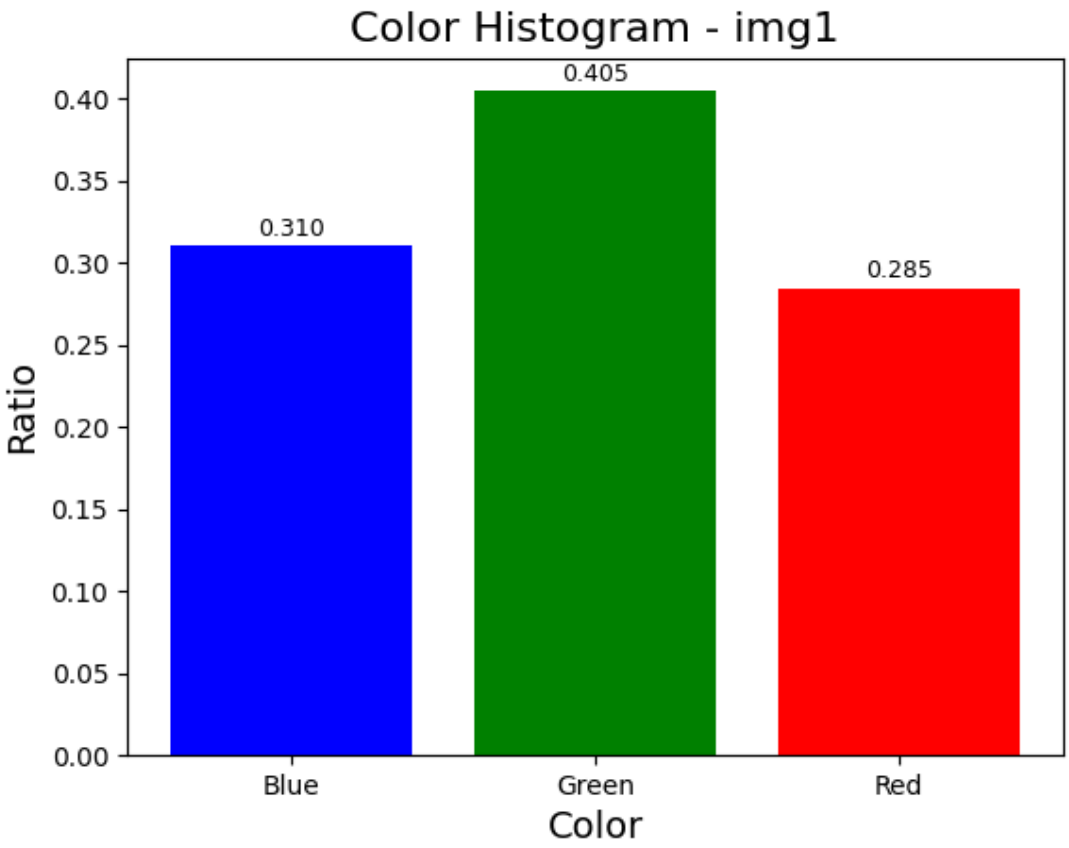
- 解决思路：同练习一读取图像，然后使用`cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)`方法以灰度图像方式读入三幅图像，再使用matplotlib中的直方图函数`plt.hist(img_gray.ravel(), bins=256, range=(0, 256), color='black')`计算并绘制灰度直方图，其中`ravel()`方法用于将二维灰度图像数组转换为一维数组以便于计算直方图。对于梯度直方图，我们使用Sobel算子（即`cv2.Sobel()`方法）对灰度图像进行梯度计算，得到梯度强度，然后用类似计算灰度直方图的方法计算并绘制梯度直方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

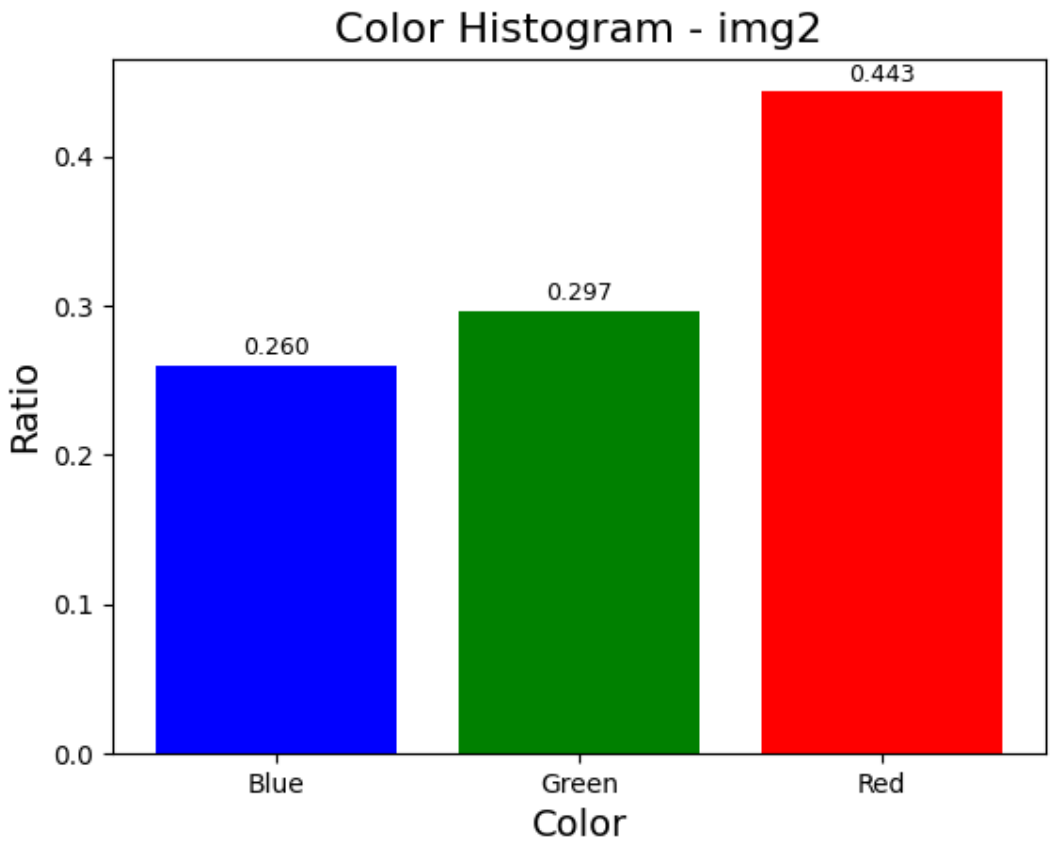
```
for img in images:
    # Read the image in grayscale
    img_gray = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

    plt.hist(img_gray.ravel(), bins=256, range=(0, 256), color='black', density=True)

    # 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)

    plt.hist(grad_magnitude.ravel(), bins=361, range=(0, 360), color='black',
density=True)
```

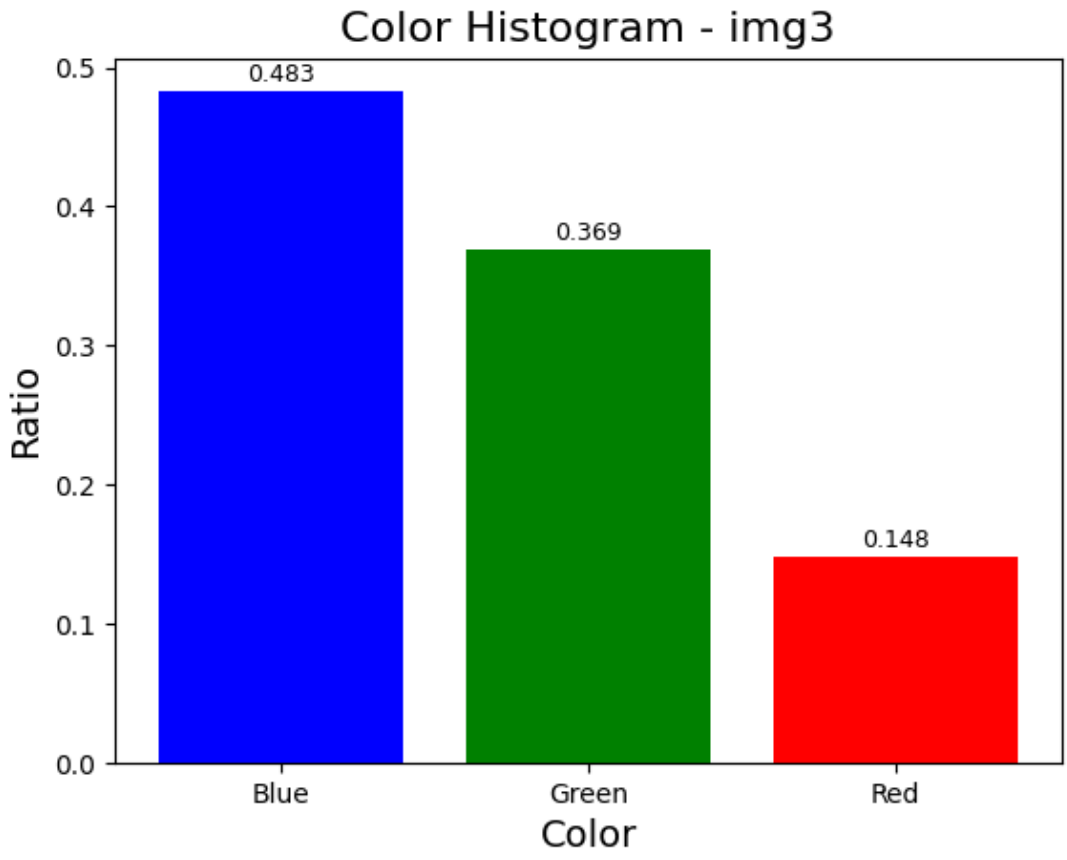
4. 代码运行结果

4.1. 练习一

得到的三幅颜色直方图为：

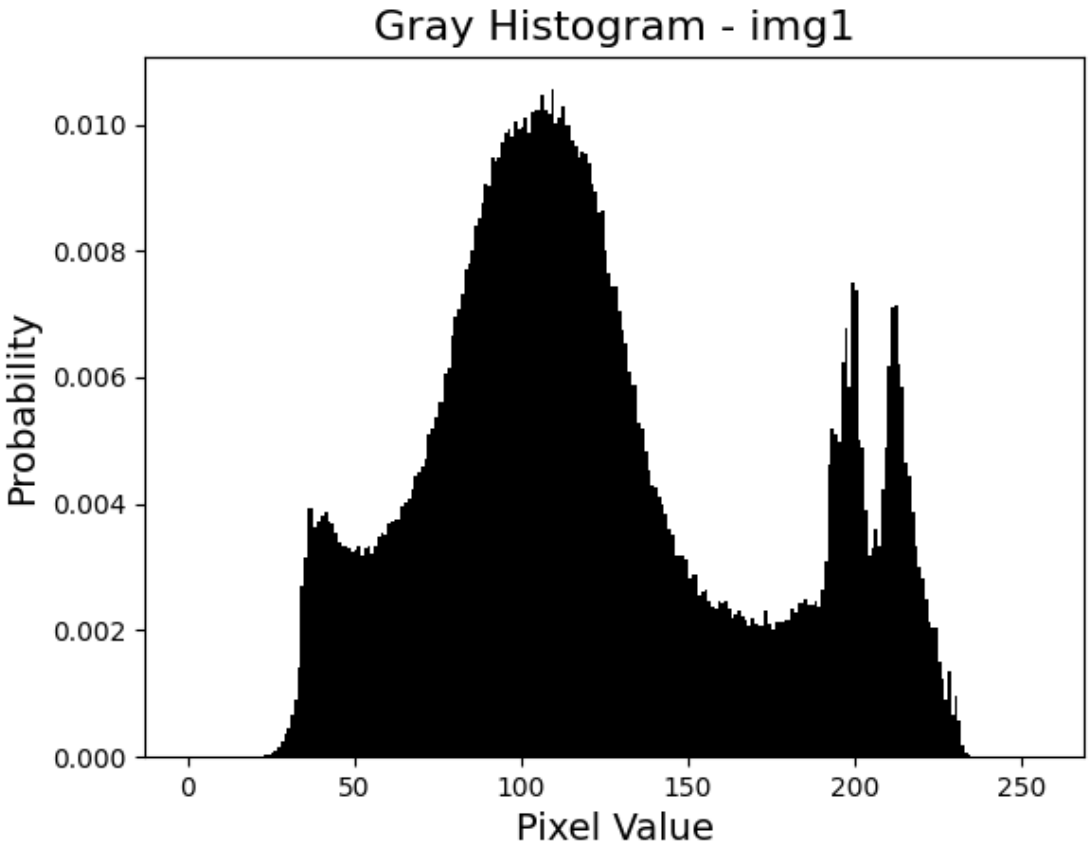


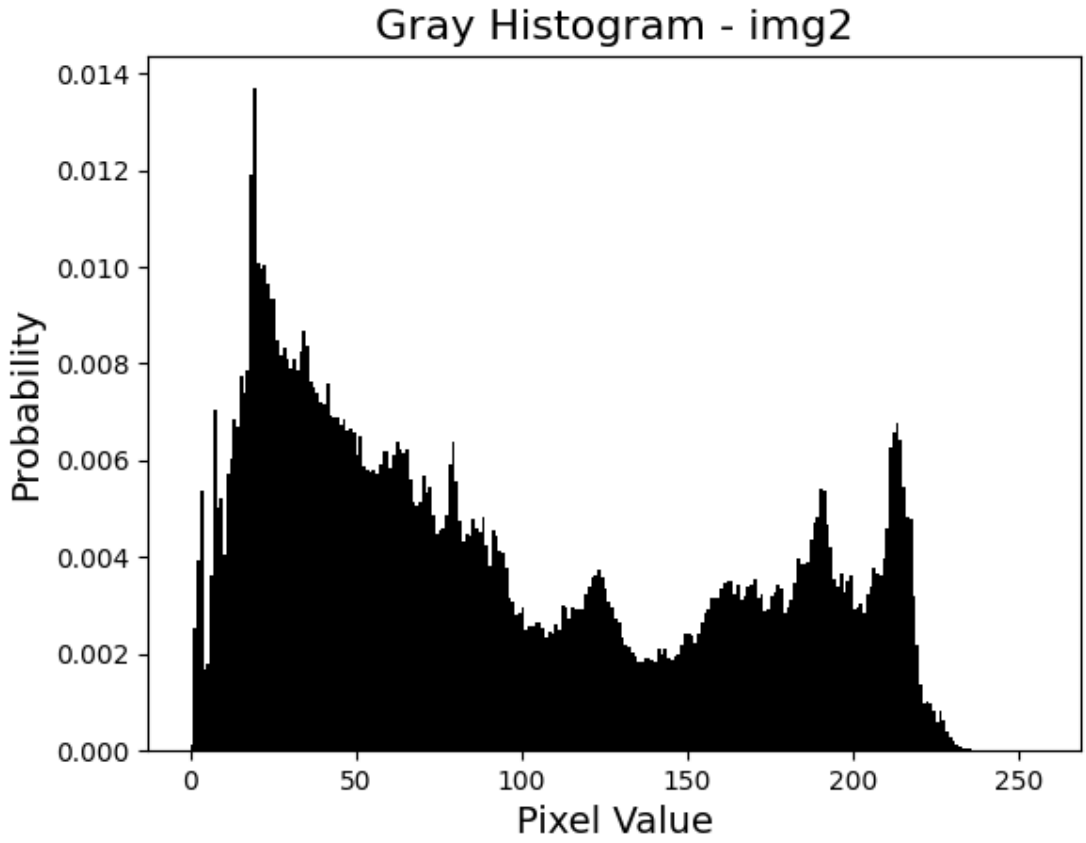

Color Histogram - img2
0.443
0.297
0.260
0.4
0.3
0.2
0.1
0.0
Ratio
Blue
Green
Red
Color

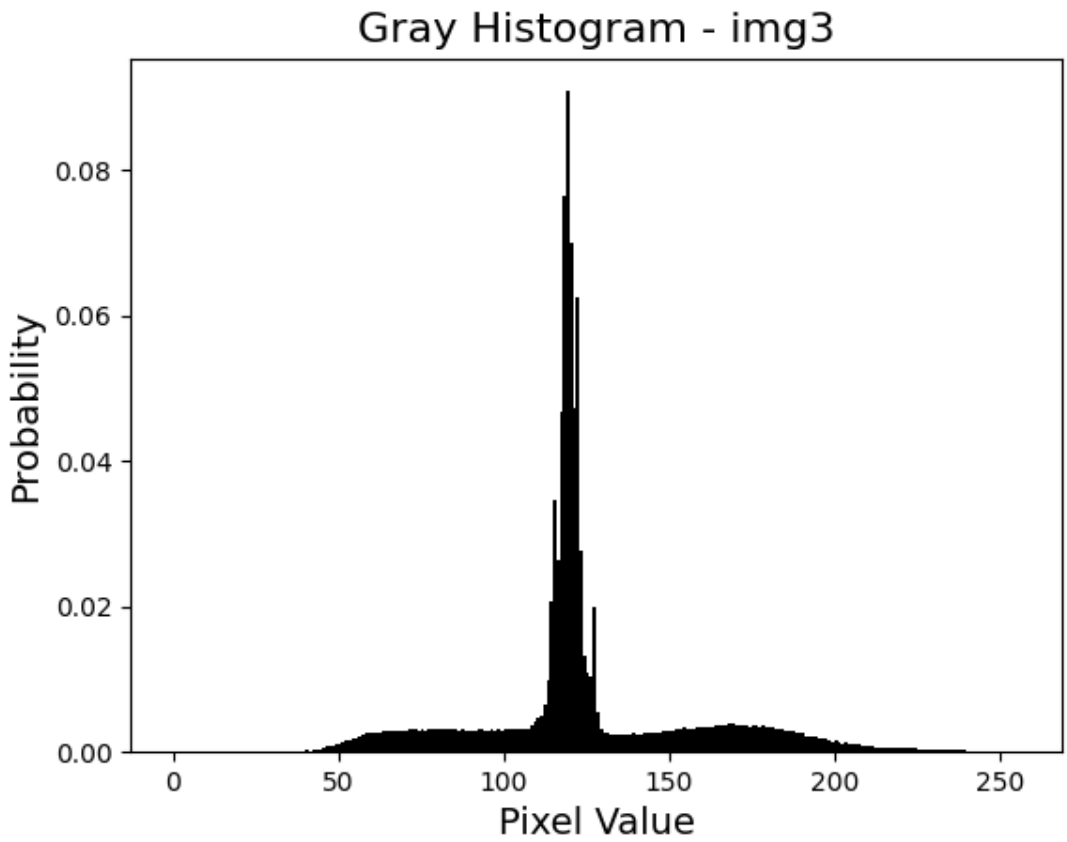

Color Histogram - img3
0.483
0.369
0.148
0.5
0.4
0.3
0.2
0.1
0.0
Ratio
Blue
Green
Red
Color

4.2. 练习二

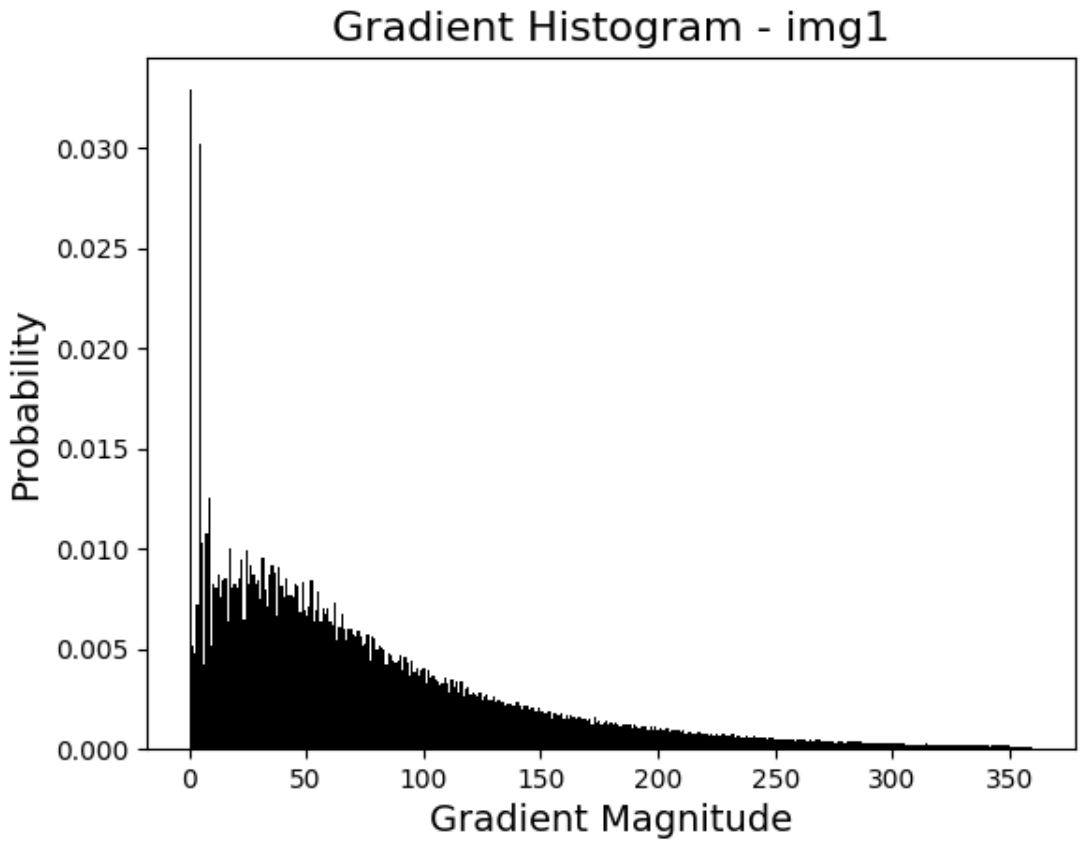
得到的三幅灰度直方图为：

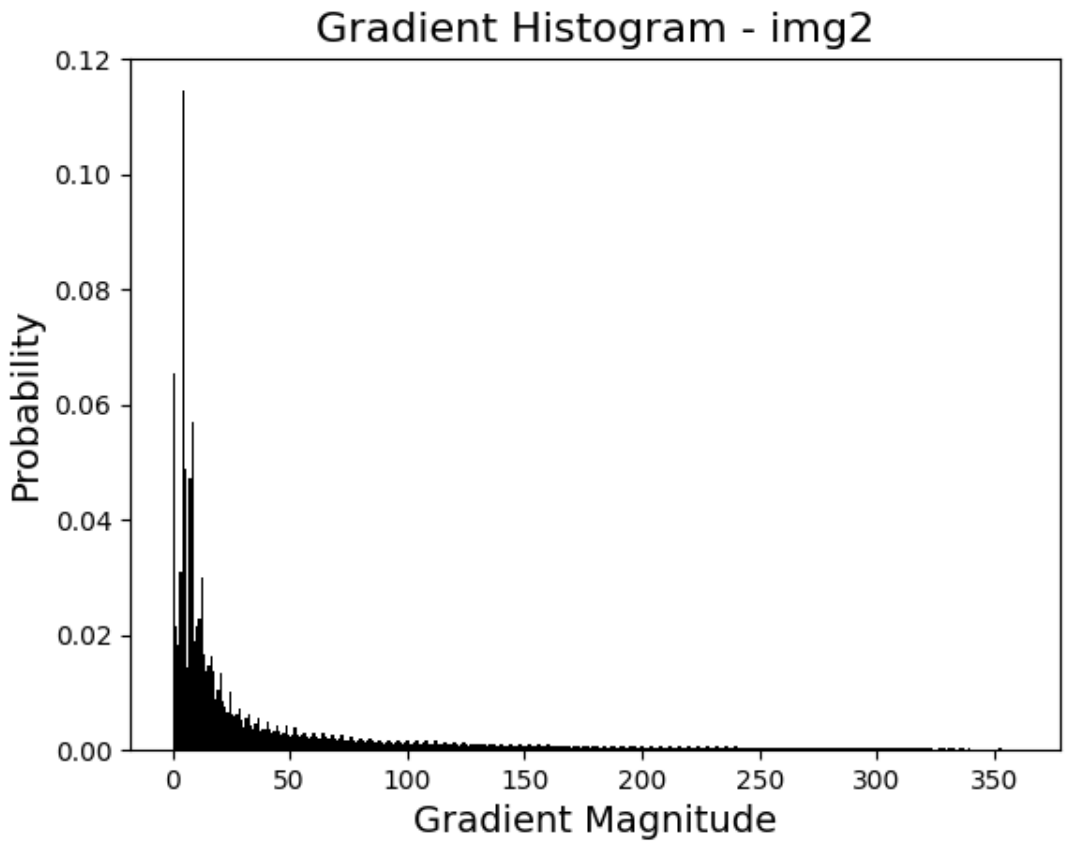


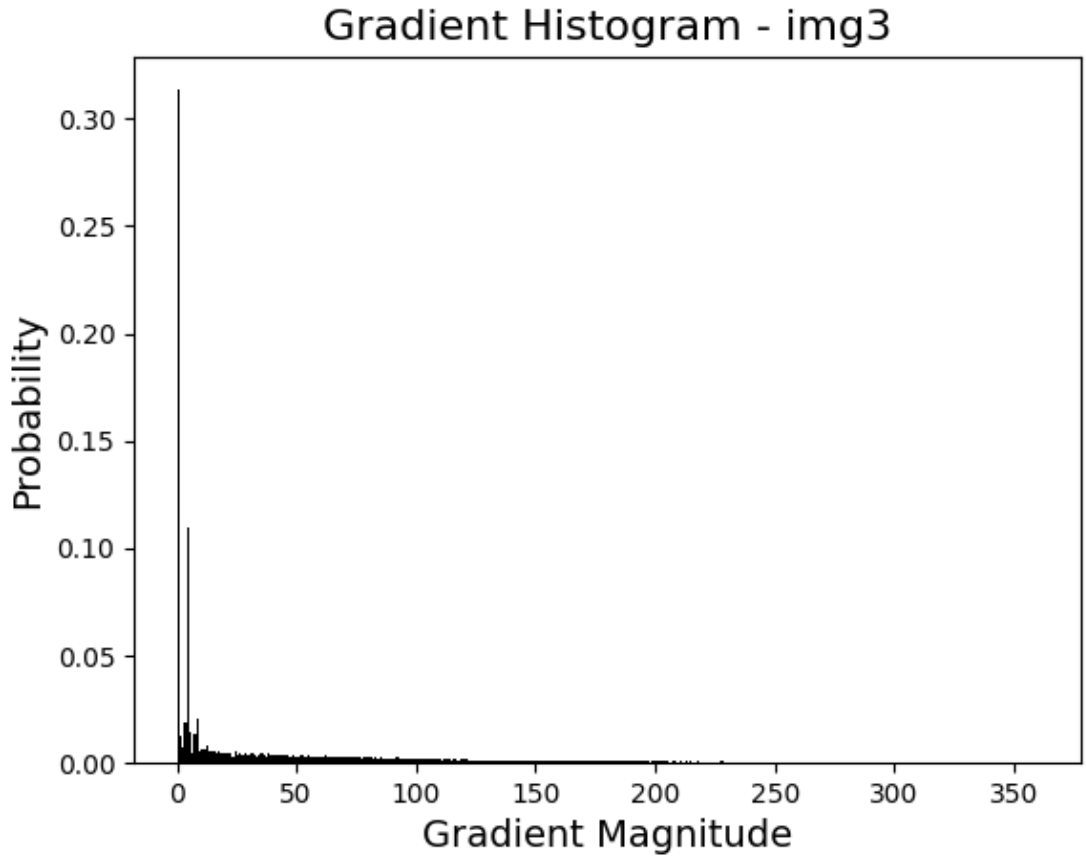




得到的三幅梯度直方图为：




Gradient Histogram - img2
Probability
Gradient Magnitude
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250
300
350


Gradient Histogram - img3
Probability
Gradient Magnitude
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
50
100
150
200
250
300
350

5. 实验结果分析与思考

5.1. 练习一结果分析

- 从三幅颜色直方图可以看出：
    - **图像1**：绿色分量的能量占比最高，蓝色分量略高于红色分量，因此整体色调偏绿色。这与图像中实际呈现的绿色视觉效果相一致。
    - **图像2**：红色分量的能量占比最高，绿色分量略高于蓝色分量，因此整体色调偏红，符合图像"夕阳西下"所呈现的暖色调特征。
    - **图像3**：蓝色分量的能量占比最高，绿色分量略高于红色分量，因此整体色调以蓝色为主，辅以部分绿色，红色最少。这与图像中蓝色主导的视觉特征相吻合

5.2. 练习二结果分析

(1) 灰度直方图分析

- **图像1**：灰度分布较为均匀，但在灰度值约 100 与 200 处存在明显集中，整体灰度主要分布在中段区域，表明图像整体画面较为明亮。
- **图像2**：灰度分布相对均匀，但主要集中在低灰度区域，同时在中高灰度段也有一定分布，说明图像在部分区域较为灰暗，而其他区域则偏明亮，存在明显的区域差异。
- **图像3**：灰度值几乎完全集中在 100~200 的中段区间，其中在 140~150 范围附近分布最为集中，表明图像整体较亮，且明暗对比不明显，画面层次较为单一。

(2) 梯度直方图分析

- **图像1**：梯度强度分布较为分散，范围宽，峰值较低，说明图像包含较多的细节和纹理，梯度分布较均匀，画面内容复杂。
- **图像2**：梯度强度集中在低值区域，说明图像整体较为平滑，边缘和细节较少，以大面积平坦区域为主。
- **图像3**：梯度分布高度集中在最小值附近，且峰值极高，说明图像几乎完全平坦，缺乏显著的边缘或细节变化，与图像的实际视觉特征相符。

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?

   在 OpenCV 中:

   `cv2.imread()` 读取的图像数据默认采用 **BGR（Blue–Green–Red）** 通道顺序，而 Matplotlib（`plt.imshow`）等大多数可视化库使用的是 **RGB（Red–Green–Blue）** 通道顺序。

   如果直接用 `plt.imshow(img_bgr)` 显示 OpenCV 读取的图像，红色和蓝色通道会对调，导致图像颜色失真。

   因此，`cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)` 的作用就是将图像的颜色空间从 **BGR 转换为 RGB**，保证在 Matplotlib 或其他使用 RGB 格式的库中显示时，图像颜色与原始图像保持一致。

2. 如何使得pyplot正确显示灰度图的颜色?

   如果 `img_gray` 是由 `OpenCV` 读的图像（`cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)`），那么它的本质是二维数组，`plt.imshow()` 默认会认为是彩色图。

   要正确显示灰度图的颜色，需要显式指定`colormap`，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

- 起初读取图像的路径时出现错误，对图形进行处理的函数报错`TypeError: 'NoneType' object is not subscriptable`，发现没有正确读取图片，将图片文件夹与代码文件置于同一目录下后解决。
- 一开始使用OpenCV-python中的函数来计算并绘制颜色直方图时，直接使用了`cv2.calchist()`函数，并没有画出想要得到的直方图，之后经过仔细学习之后，发现应先分离每个颜色分量，再对其进行求和，然后才能得到每种颜色的相对比例。
- 最初绘制灰度直方图与梯度直方图时没有归一化，使`plt.hist`中参数`density=True`解决。

6.2. 经验总结

使用未明确功能的函数可能出现错误结果，在使用前应先查阅官方文档和示例代码，或向人工智能大模型询问，清楚函数的用途和其中各参数含义后再进行使用。