《计算机视觉》实验报告

姓名：汪江豪 学号：22121630

实验三

一. 任务 1

1、读取一张图片，将其转换为 HSV 空间

2、分离原图片 RGB 通道及转换后的图片 HSV 通道

3、 对 RGB 三个通道分别画出其三维图（提示：polt_sufface 函数）

a) 核心代码：

(1) 转换到 HSV 空间，并分离 BGR 和 HSV：

```
    img = cv.imread('img.jpg')
    img_hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
    h, s, v = cv.split(img_hsv) # Hue 色调, Saturation 饱和度, Value 亮度
    b, g, r = cv.split(img) # BGR 通道
```

(2) 对 RGB 三通道画出三维图：

```
# 绘制 RGB 三通道的三维图
def test1_2():
    '''调用 channels_3D 函数，绘制 RGB 三通道的三维图'''
    img = cv.imread('img.jpg')
    img = cv.resize(img, (0, 0), fx = 0.6, fy = 0.6) # 缩小图片

    # 将 BGR 图像转换为 RGB 图像
    img_rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
    # 分离 RGB 通道
    r, g, b = cv.split(img_rgb)

    # 创建网格
    h, w = img.shape[:2]
    x = np.arange(0, w, 1) # x 轴
    y = np.arange(0, h, 1) # y 轴
    x_mesh, y_mesh = np.meshgrid(x, y) # 创建网格

    # 创建 3D 图形
```

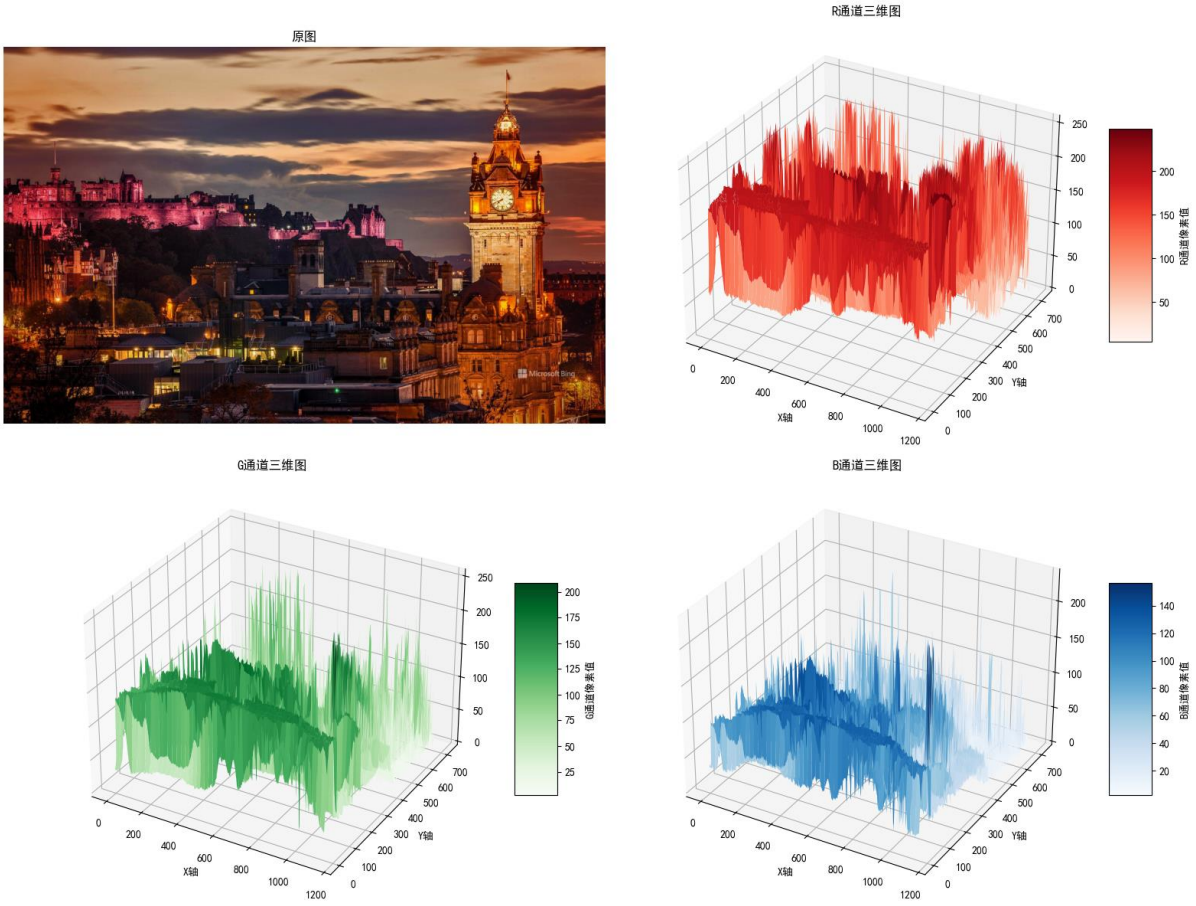
```
    fig = plt.figure(figsize=(16, 12))

    ax0 = fig.add_subplot(2,2,1)
    ax0.imshow(img_rgb) # 显示原图
    ax0.set_title('原图')
    ax0.axis('off')

    # R 通道
    ax1 = fig.add_subplot(2,2,2, projection='3d')
    surf1 = ax1.plot_surface(x_mesh, y_mesh, r, cmap='Reds')
    ax1.set_title('R 通道三维图')
    ax1.set_xlabel('X 轴')
    ax1.set_ylabel('Y 轴')

    # G 通道
    ax2 = fig.add_subplot(2,2,3, projection='3d')
    surf2 = ax2.plot_surface(x_mesh, y_mesh, g, cmap='Greens')
    ax2.set_title('G 通道三维图')
    ax2.set_xlabel('X 轴')
    ax2.set_ylabel('Y 轴')

    # B 通道
    ax3 = fig.add_subplot(2,2,4, projection='3d')
    surf3 = ax3.plot_surface(x_mesh, y_mesh, b, cmap='Blues')
    ax3.set_title('B 通道三维图')
    ax3.set_xlabel('X 轴')
    ax3.set_ylabel('Y 轴')

    # 为曲面添加颜色条
    fig.colorbar(surf1, ax=ax1, shrink=0.5, aspect=5, label='R 通道像素值')
    fig.colorbar(surf2, ax=ax2, shrink=0.5, aspect=5, label='G 通道像素值')
    fig.colorbar(surf3, ax=ax3, shrink=0.5, aspect=5, label='B 通道像素值')

    plt.tight_layout() # 自动调整子图参数
    plt.savefig('RGB_3D.png') # 保存图片
    plt.show() # 显示图形
```

b) 实验结果截图

(1) 分离 HSV 通道：

(2) 分离 BGR 通道：

(3) 画出 RGB 三通道的三维图：



c) 实验小结

图像有不同的通道表示方式，常见的是 RGB 也就是红绿蓝，还有 Hue 色调，Saturation 饱和度，Value 亮度。不同的通道空间常常用于不同领域的处理。RGB 三通道的三维图形象展示了各个通道，各个像素点的像素值。

二、任务 2

1. 读取彩色图像 home_color；
2. 画出灰度化图像 home_gray 的灰度直方图，并拼接原灰度图与结果图；
3. 画出彩色 home_color 图像的直方图，并拼接原彩色图与结果图，且与上一问结果放在同一个窗口中显示；
4. 画出 ROI（感兴趣区域 ）的直方图，ROI 区域为 x：50-100，y：100-200，将原图 home_color，ROI 的 mask 图，ROI 提取后的图及其直方图放在一个窗口内显示。

a) 核心代码：

(1) 灰度图与原图拼接：

```python
# 画出灰度图像及其直方图
def test2_1():
    img = cv.imread('home_color.png')
    # 计算直方图
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    hist, bins = np.histogram(img.flatten(), 256, [0, 256]) # 计算直方图

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(gray, cmap='gray')
    plt.title('home_gray')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.plot(hist, color='gray')
    plt.xlim([0, 256])
    plt.title('Histogram')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')
    plt.grid()

    plt.tight_layout()
    plt.savefig('gray_histogram.png') # 保存图片
    plt.show()
```

(2) 彩色图像与直方图

```python
# 计算 RGB 通道的直方图
def test2_2():
    img = cv.imread('home_color.png')
    # 计算 RGB 通道的直方图
    colors = ('b', 'g', 'r')
    plt.figure(figsize=(12, 6))

    # 左边显示彩色原图
    plt.subplot(1, 2, 1)
    plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))
    plt.title('home_color')
    plt.axis('off')

    # 右边显示彩色直方图
    plt.subplot(1, 2, 2)
    for i, color in enumerate(colors):
        hist = cv.calcHist([img], [i], None, [256], [0, 256])
        plt.plot(hist, color=color, label = f'{color.upper()}通道')
```

```
    plt.xlim([0, 256])
    plt.title('彩色直方图')
    plt.xlabel('像素值')
    plt.ylabel('频数')
    plt.grid()

    plt.tight_layout()
    plt.savefig('color_histogram.png') # 保存图片
    plt.show()
```

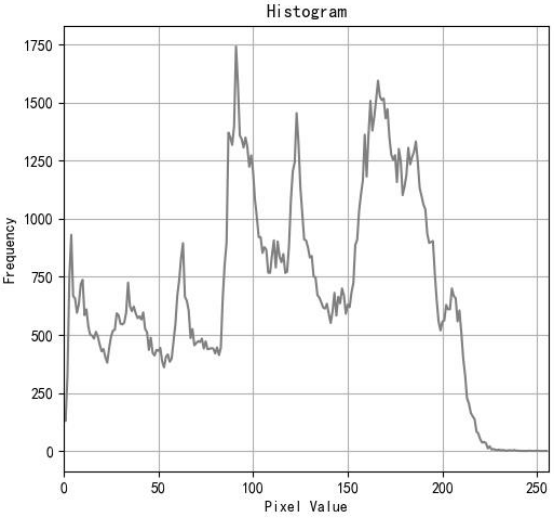
(3) ROI 与直方图

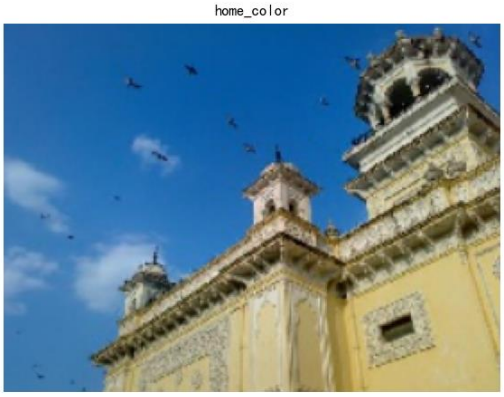
```
# 将四个子图放在一起，原图，ROI 的 mask 图，ROI 提取后的图及其直方图
def test2_3():
    img = cv.imread('home_color.png')
    # 定义 ROI 区域
    x_start, x_end = 50, 100
    y_start, y_end = 100, 200
    roi = img[x_start:x_end, y_start:y_end] # 提取 ROI 区域

    # 创建 mask
    mask = np.zeros(img.shape[:2], dtype=np.uint8) # 创建 mask
    mask[x_start:x_end, y_start:y_end] = 255 # 设置 ROI 区域为 255

    masked_img = cv.bitwise_and(img, img, mask=mask) # 应用 mask

    # 计算 ROI 区域的直方图
    colors = ('b', 'g', 'r')
    plt.figure(figsize=(12, 8))

    plt.subplot(2, 2, 1)
    plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))
    plt.title('home_color')
    plt.axis('off')

    plt.subplot(2, 2, 2)
    plt.imshow(mask, cmap='gray')
    plt.title('ROI Mask')
    plt.axis('off')

    plt.subplot(2, 2, 3)
    plt.imshow(cv.cvtColor(masked_img, cv.COLOR_BGR2RGB))
    plt.title('ROI 提取图')
    plt.axis('off')

    plt.subplot(2, 2, 4)
```

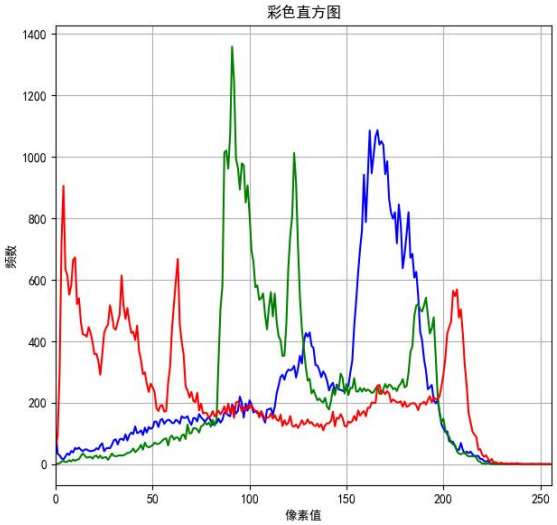
```
    for i, color in enumerate(colors):
        hist = cv.calcHist([roi], [i], None, [256], [0, 256])
        plt.plot(hist, color=color, label = f'{color.upper()}通道')
    plt.xlim([0, 256])
    plt.title('ROI 直方图')
    plt.xlabel('像素值')
    plt.ylabel('频数')
    plt.grid()
    plt.legend() # 显示图例

    plt.tight_layout()
    plt.savefig('roi_histogram.png') # 保存图片
    plt.show()
```

b) 实验结果截图

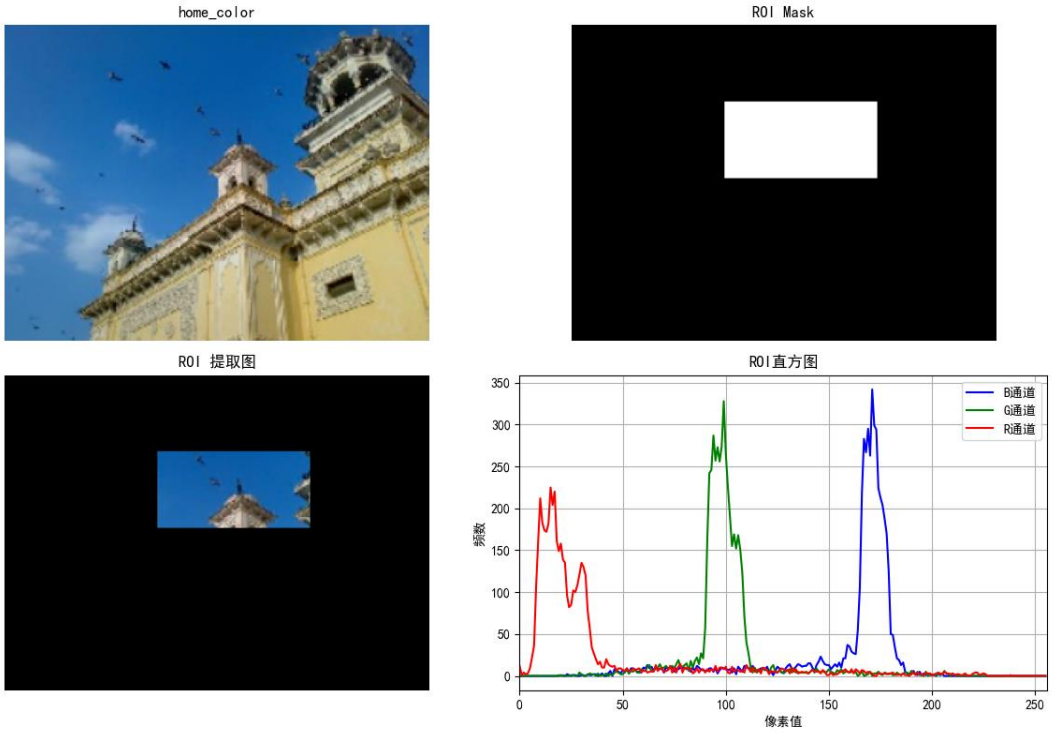
(1)灰度图和彩图的直方图：







(2)ROI 直方图：



c) 实验小结

图像的直方图反映了图像像素值的大题分布，以直方图为基础的图像处理计数有很多，包括直方图均衡化等。通过 mask 掩膜技术，能够提取到局部的像素值分布特征。

三、任务 3

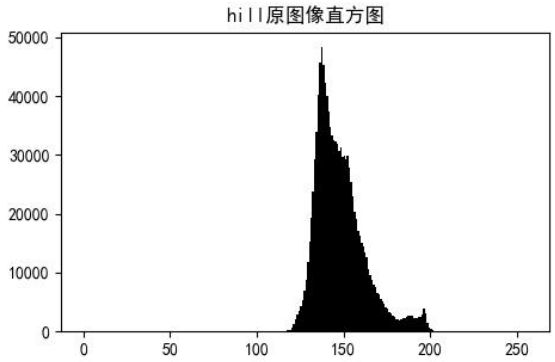
编程实现直方图均衡化，给出测试效果

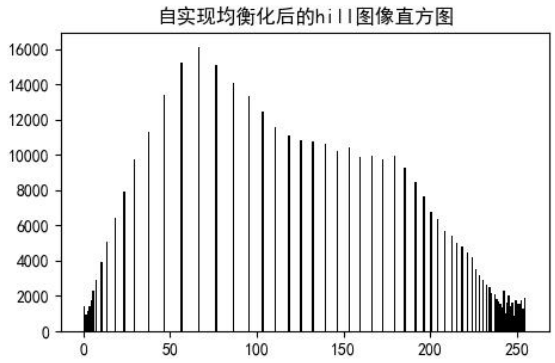
a) 核心代码

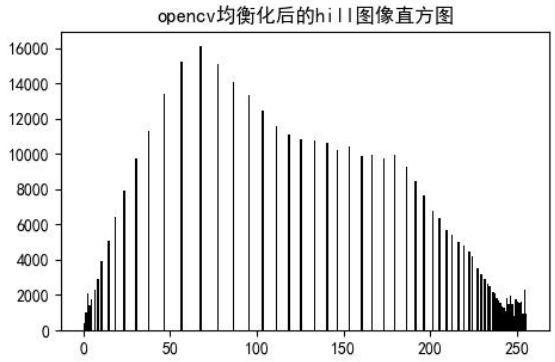
```
# 自实现直方图均衡化
def histogram_equalization(img):
    # 将图像转为灰度图
    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    # 计算灰度图像的直方图
    # hist 纵坐标数值数组，bins 是直方图的横坐标
    hist, bins = np.histogram(img_gray.flatten(), 256, [0, 256])
    # 计算累计分布函数
    cdf = hist.cumsum()  # 累积函数，表示每个灰度级的累计像素数量
    # 使用 cdf 对图像均衡化
    cdf_m = np.ma.masked_equal(cdf, 0)  # 掩盖 cdf 数组中的 0 值

    # 按权重映射到 0-255 之间
    cdf_m = (cdf_m - cdf_m.min()) / (cdf_m.max() - cdf_m.min()) * 255
```

```
    cdf = np.ma.filled(cdf_m, 0).astype("uint8") # 查找表

    # 将原图像的灰度级映射到均衡化后的灰度级
    img_res = cdf[img_gray]
    return img_res
```

b) 实验结果截图







c) 实验小结

直方图均衡化是个强大的工具。能将过暗或过亮的图片通过适当调整，展现出极好的显示效果，从而显示了图像的细节。其数学底层方法，是通过概率的累积分布函数，乘上灰度区间，将原来的灰度值映射成另一个灰度值，由于这个灰度值是由累积分布函数决定的，因此能保证图像整体上灰度分布均衡。