《计算机视觉》实验报告

姓名：冯俊佳 学号：23122721

实验 3

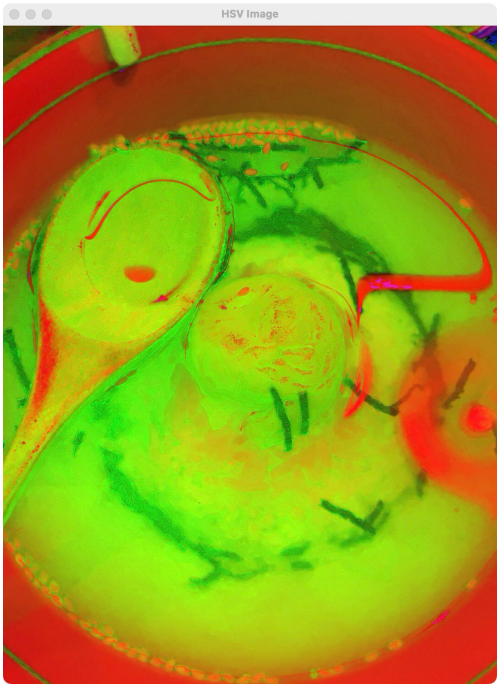
一. 任务 1

a) 核心代码：

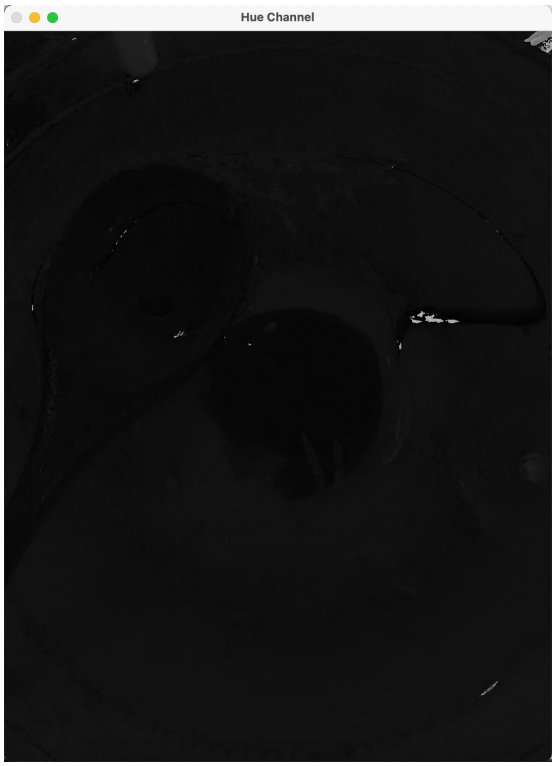
```
# 1.读取一张图片，将其转换为 HSV 空间
image = cv.imread("/Users/feng/Desktop/计算机视觉/Week3/W3_Experiment/test1.jpg")
# 转化为 HSV
HSV = cv.cvtColor(image, cv.COLOR_BGR2HSV)

# 2.分离原图片 RGB 通道及转换后的图片 HSV 通道
# 分离 RGB
B,G,R = cv.split(image)
# 分离 HSV
H,S,V = cv.split(HSV)

# 3.对 RGB 三个通道分别画出其三维图（提示：polt_surface 函数）
# 创建网格
x, y = np.meshgrid(np.arange(image.shape[1]), np.arange(image.shape[0]))
# image.shape[1]宽，shape[0]长
# 创建一个空白的图形窗口
fig = plt.figure(figsize=(9, 3))

# fig.add_subplot 添加子图 131:一行三列第一个图 projection:指定为三维
# plot_surface 绘制三维曲面图 参数：xyz，cmap:指定颜色映射
# 绘制红色通道的三维曲面
ax = fig.add_subplot(131, projection='3d') # 添加子图
ax.plot_surface(x, y, R, cmap='Reds') # 指定为红色映射
ax.set_title('Red Channel') # 设置标题

# 绘制绿色通道的三维曲面
ax = fig.add_subplot(132, projection='3d')
ax.plot_surface(x, y, G, cmap='Greens')
```

```
ax.set_title('Green Channel')

# 绘制蓝色通道的三维曲面
ax = fig.add_subplot(133, projection='3d')
ax.plot_surface(x, y, B, cmap='Blues')
ax.set_title('Blue Channel')

plt.show() # 显示
```

b) 实验结果截图

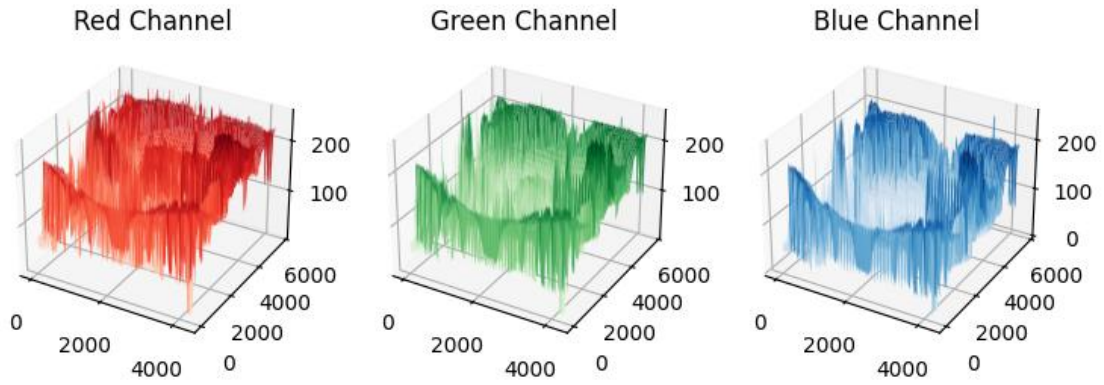



Red Channel


Hue Channel


Saturation Channel


Value Channel



c) 实验小结

本实验基本思想是使用 OpenCV 库读取图片，并将其转换为 HSV 色彩空间。然后，利用 split( )分离 RGB 和 HSV 通道，并利用 matplotlib 库绘制三维曲面图展示 RGB 通道的分布情况。收获包括对图像处理中颜色通道的理解以及使用 matplotlib 进行三维可视化的经验。

二. 任务 2

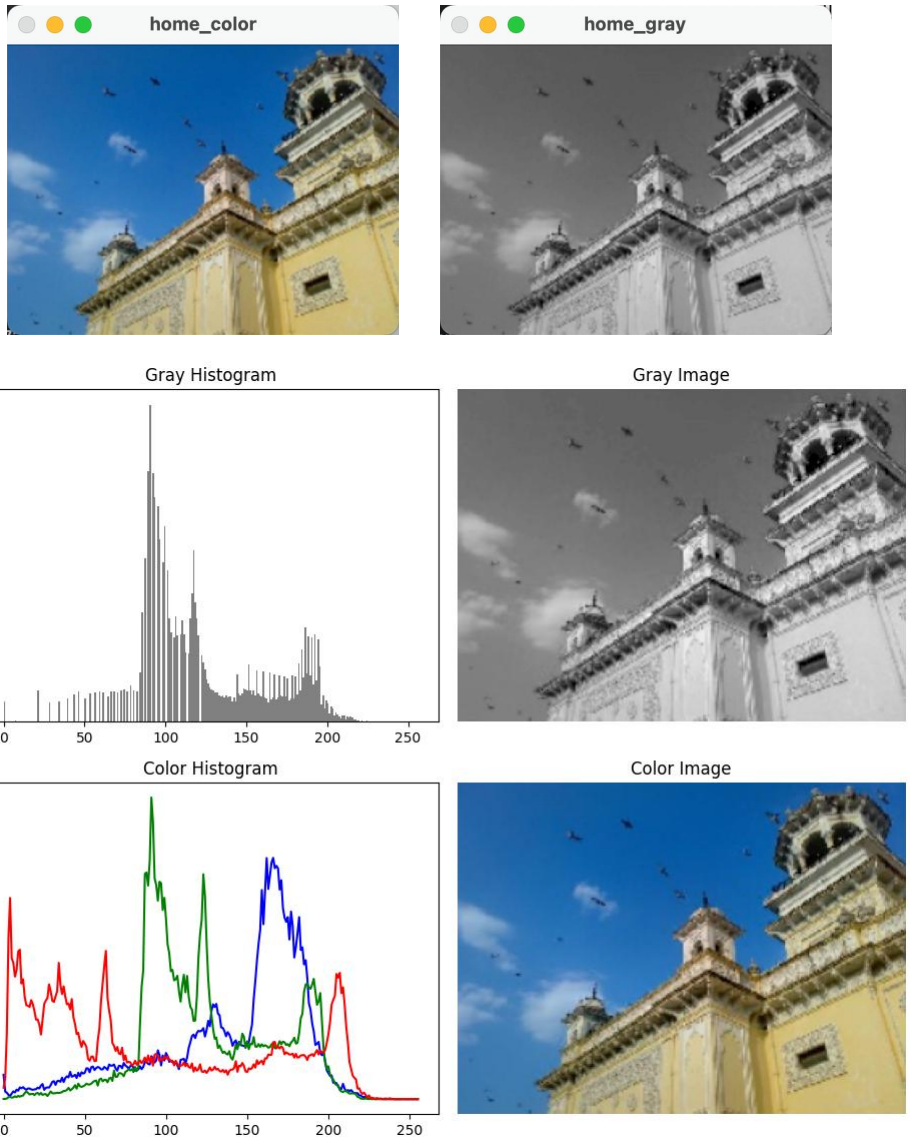
a) 核心代码：

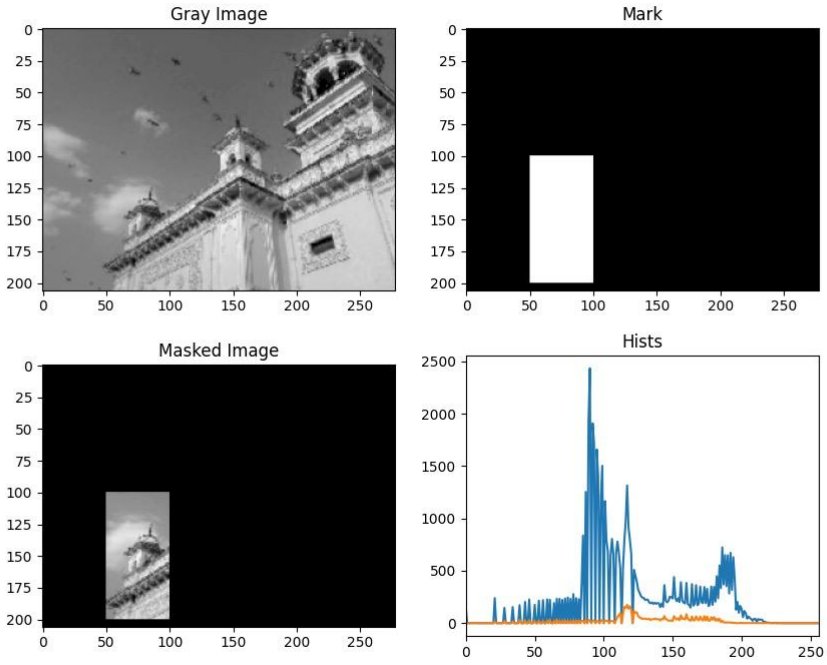
```
# 1.读取 home_color 图像和灰度图像
home_color = cv.imread("/Users/feng/Desktop/计算机视觉/Week3/W3_Experiment/home_color.jpg")
home_gray = cv.imread("/Users/feng/Desktop/计算机视觉/Week3/W3_Experiment/home_color.jpg", 0)

# 2&3. 绘制灰度直方图和彩色直方图，并将原图拼接在一起展示
# 第一行左边放灰度直方图，右边放灰度原图；第二行画 RGB 直方图和原图，以子图形式呈现；全部放在一个窗口里
# 创建画布
fig = plt.figure("Gray and Color", figsize=(10, 8))

# 灰度直方图
ax1 = fig.add_subplot(221)
ax1.hist(home_gray.ravel(), 256, [0, 256], color='gray')
ax1.set_title('Gray Histogram')

# 灰度图像
ax2 = fig.add_subplot(222)
ax2.imshow(home_gray, 'gray')
ax2.set_title('Gray Image')
```

```
ax2.axis('off')

# 彩色直方图
ax3 = fig.add_subplot(223)
colors = ('b', 'g', 'r')
for i, col in enumerate(colors):
    hist = cv.calcHist([home_color],[i], None, [256], [0, 256])
    ax3.plot(hist, color=col)
ax3.set_title('Color Histogram')

# 彩色图像
ax4 = fig.add_subplot(224)
ax4.imshow(cv.cvtColor(home_color, cv.COLOR_BGR2RGB))
ax4.set_title('Color Image')
ax4.axis('off')

plt.tight_layout()
plt.show()

# 4. 绘制 ROI 区域的直方图，并拼接原图、mask、ROI 区域图像
# 设置 ROI 区域（x: 50-100, y: 100-200）
x1, x2 = 50, 100
y1, y2 = 100, 200

# 创建掩码
mask = np.zeros(home_gray.shape, dtype=np.uint8)
mask[y1:y2, x1:x2] = 255  # 设为白色区域

# 掩码应用到灰度图上
masked_img = cv.bitwise_and(home_gray, home_gray, mask=mask)

# 直方图计算
hist_full = cv.calcHist([home_gray],[0],None, [256], [0, 256])
hist_mask = cv.calcHist([home_gray],[0],mask, [256], [0, 256])

# 创建窗口
fig2 = plt.figure("ROI and Hist", figsize=(10, 8))
ax1 = fig2.add_subplot(221) #添加子图
ax1.imshow(home_gray, 'gray') #灰度图
ax1.set_title('Gray Image') #设置标题

ax2 = fig2.add_subplot(222)
ax2.imshow(mask,'gray')  # 遮掩图
ax2.set_title('Mark')
```

```
63.
64. ax3 = fig2.add_subplot(223)
65. ax3.imshow(masked_img, 'gray') #遮掩图
66. ax3.set_title('Masked Image')
67.
68. ax4 = fig2.add_subplot(224)
69. ax4.plot(hist_full), plt.plot(hist_mask) #直方图
70. ax4.set_title('Hists')
71.
72. plt.xlim([0,256])
73. plt.show()
```

b) 实验结果截图





c) 实验小结

本次实验实现了对彩色图像和灰度图像的直方图分析，并且通过子图的方式将结果放在同一个窗口内展示。对于灰度图像，绘制了灰度直方图以及灰度图像本身；对于彩色图像，绘制了 RGB 直方图和彩色图像本身；实现了对感兴趣区域（ROI）的直方图分析，展示了原图、ROI 区域的 mask 图、ROI 提取后的图像以及它们的直方图。

收获是提升了对图像直方图分析的理解，将图像的通道分离开或者将图像变为灰度图，通过直方图的显示来更好的理解图像的组成，和不同通道的效果，以及为了能把这些图的原图、灰度图和各自的 hist 直方图放在一起比较，对 matplotlib 库中子图功能的运用有了更熟练的掌握。

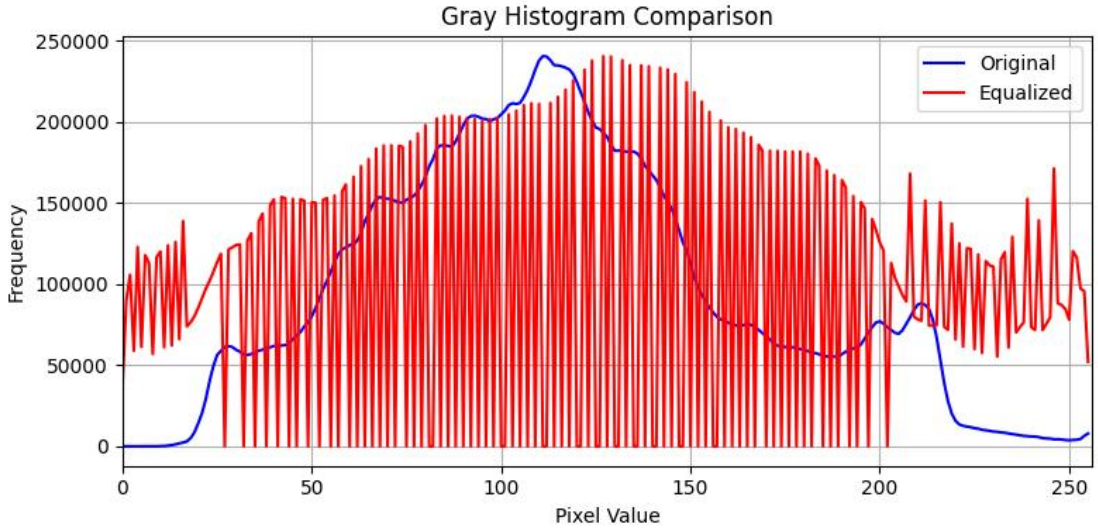
三. 任务 3

a) 核心代码：

```
1.# 1. 读取原图并转换为灰度图
2.image = cv.imread("/Users/feng/Desktop/计算机视觉/Week3/W3_Experiment/test1.jpg")
3.gray = cv.imread("/Users/feng/Desktop/计算机视觉/Week3/W3_Experiment/test1.jpg", 0)
4.
```

```
# 2. 直方图均衡化
equalized = cv.equalizeHist(gray)

# 3. 显示图像对比
cv.imshow("Original Gray", gray)
cv.imshow("Equalized Gray", equalized)

# 4. 显示直方图对比（用 matplotlib）
plt.figure(figsize=(8, 4))
plt.plot(cv.calcHist([gray], [0], None, [256], [0, 256]), color='blue', label='Original')
plt.plot(cv.calcHist([equalized], [0], None, [256], [0, 256]), color='red', label='Equalized')

plt.title('Gray Histogram Comparison')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.legend()
plt.xlim([0, 256])
plt.grid(True)
plt.tight_layout()
plt.show()
```

b) 实验结果截图



c) 实验小结

本实验实现了直方图均衡化，并展示了原始图像和直方图均衡化后的图像及其直方图。在直方图均衡化的过程中，图像的对比度被增强，细节得到了更好的呈现。

直方图均衡化的效果能够直观地从显示的图像和直方图上观察到。原始图像的直方图可能会有一些集中在特定区域的峰值，而均衡化后的直方图会更加平滑，并且覆盖整个灰度范围，从而提高了图像的对比度和视觉效果。经过原本图像和直方图的比较，从两个方面感受到直方图均衡化的效果，加深理解。