# 《计算机视觉》实验报告

**姓名：刘远航 学号：22121883**

## 实验三

### 一. 任务 1

#### a) 实验内容

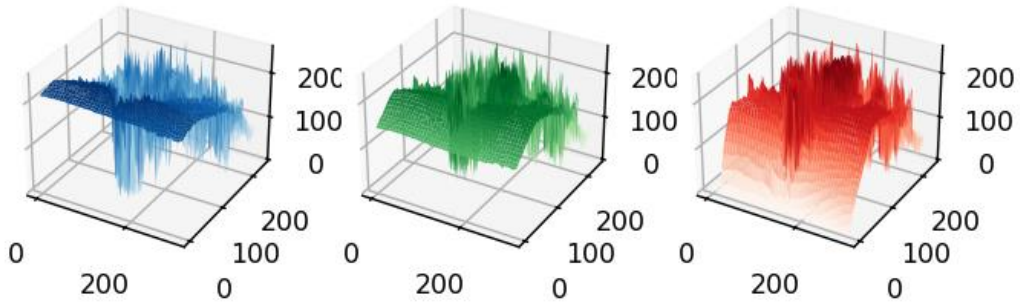
1. 读取一张图片，将其转换为 HSV 空间
2. 分离原图片 RGB 通道及转换后的图片 HSV 通道
3. 对 RGB 三个通道分别画出其三维图（提示：polt_sufface 函数）

#### b) 核心代码：

```
# 读取图片
image = cv2.imread('image.jpg')
# 将图片转换为 HSV 空间
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
# 分离原图 RGB 通道
b, g, r = cv2.split(image)
# 分离 HSV 图像的通道
h, s, v = cv2.split(hsv_image)
# 获取图像尺寸
height, width = image.shape[:2]
x = np.linspace(0, width - 1, width)
y = np.linspace(0, height - 1, height)
X, Y = np.meshgrid(x, y)
# 绘制 RGB 三维图
fig = plt.figure()
# 绘制蓝色通道的三维图
ax1 = fig.add_subplot(131, projection='3d')
ax1.plot_surface(X, Y, b, cmap='Blues')
# 绘制绿色通道的三维图
ax2 = fig.add_subplot(132, projection='3d')
ax2.plot_surface(X, Y, g, cmap='Greens')
# 绘制红色通道的三维图
ax3 = fig.add_subplot(133, projection='3d')
```

```
ax3.plot_surface(X, Y, r, cmap='Reds')
plt.show()
```

**c) 实验结果截图**




**d) 实验小结**

通过任务一了解了 RGB 与 HSV 的区别，HSV 更适用于图像处理，RGB 更适用于工业，通过绘制三维图对图片有了更直观的了解。

## 二. 任务 2

**a) 实验内容**

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；

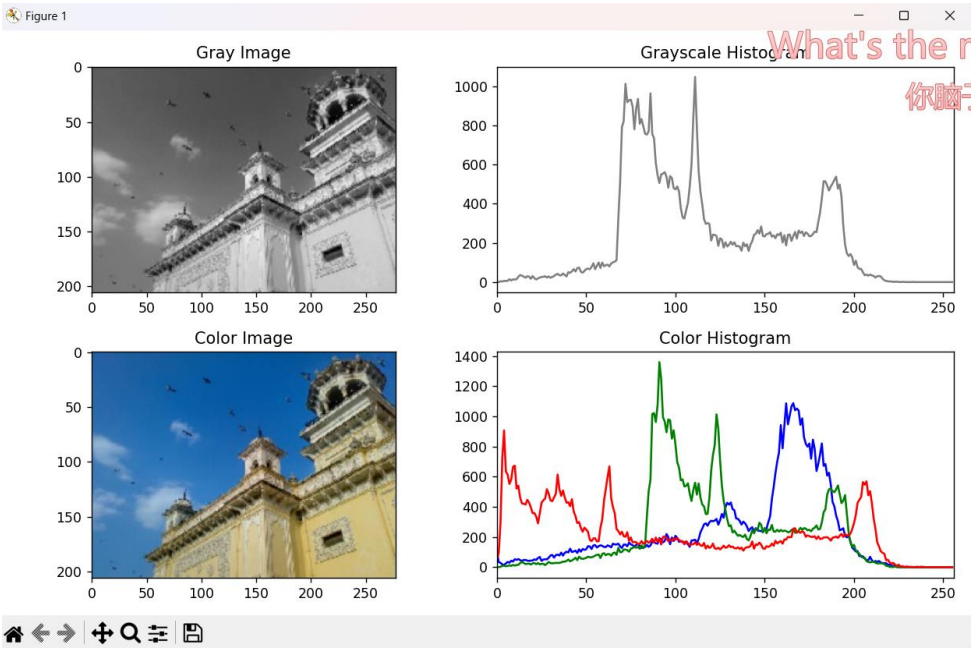
4、画出 ROI（感兴趣区域 ）的直方图，ROI 区域为 x：50-100，y：100-200，将原图 home_color，ROI 的 mask 图，ROI 提取后的图及其直方图放在一个窗口内显示。

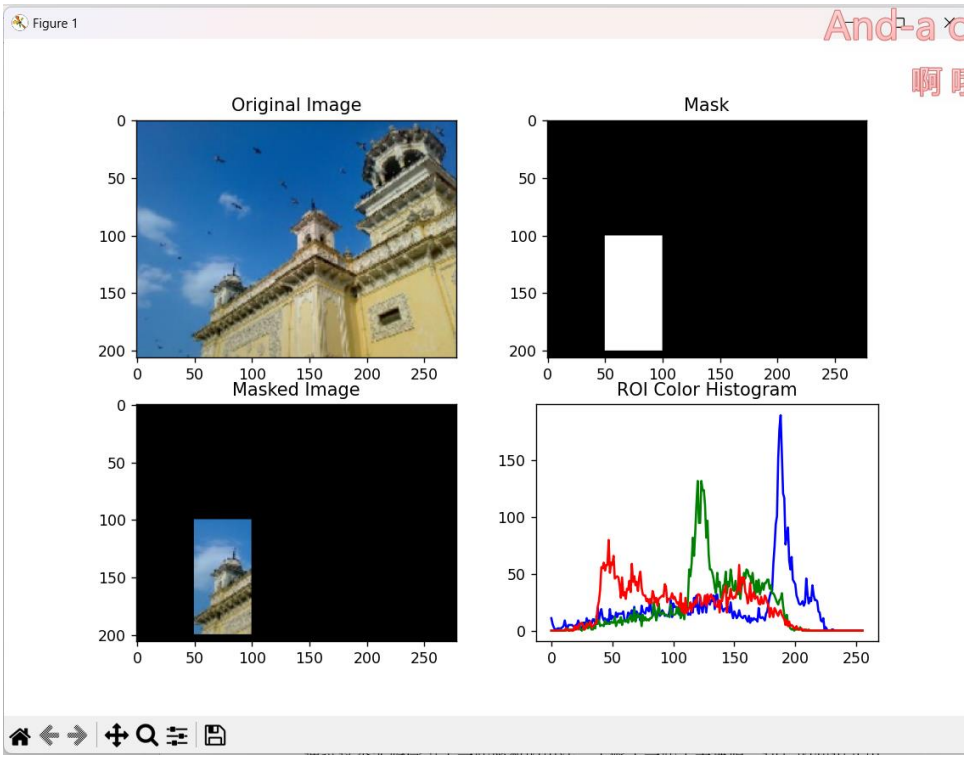
**b) 核心代码**

```
# 读取彩色图像
home_color = cv2.imread('image3.png')
# 1. 灰度化图像
home_gray = cv2.cvtColor(home_color, cv2.COLOR_BGR2GRAY)
# 2. 画出灰度化图像的直方图
hist_gray = cv2.calcHist([home_gray], [0], None, [256], [0, 256])
```

```
# 3. 画出彩色图像的直方图
colors = ('b', 'g', 'r')
hist_color = []
for i, color in enumerate(colors):
        hist = cv2.calcHist([home_color], [i], None, [256], [0, 256])
        hist_color.append(hist)
#  绘制灰度直方图
plt.figure(figsize=(10, 6)) ,plt.subplot(2, 2, 1),plt.imshow(home_gray, cmap='gray')
plt.title('Gray Image')
plt.subplot(2, 2, 2), plt.plot(hist_gray, color='gray')
plt.title('Grayscale Histogram'),plt.xlim([0, 256])
#  绘制彩色直方图
plt.subplot(2, 2, 3)
plt.imshow(cv2.cvtColor(home_color, cv2.COLOR_BGR2RGB))
plt.title('Color Image')
plt.subplot(2, 2, 4)
for i, color in enumerate(colors):
        plt.plot(hist_color[i], color=color)
plt.title('Color Histogram')
plt.xlim([0, 256])
plt.tight_layout()
plt.show()
#ROI
roi = home_color[100:200, 50:100]
mask = np.zeros(home_color.shape[:2], np.uint8)
mask[100:200, 50:100] = 255
masked_img = cv2.bitwise_and(home_color, home_color, mask=mask)

plt.figure(figsize=(12, 8)), plt.subplot(221)
plt.imshow(cv2.cvtColor(home_color, cv2.COLOR_BGR2RGB)), plt.title('Original
Image')
plt.subplot(222), plt.imshow(mask, 'gray'), plt.title('Mask')
plt.subplot(223)
plt.imshow(cv2.cvtColor(masked_img, cv2.COLOR_BGR2RGB)), plt.title('Masked
Image')
color_hist_roi = []
for i, col in enumerate(['b', 'g', 'r']):
        hist = cv2.calcHist([roi], [i], mask[100:200, 50:100], [256], [0, 256])
        color_hist_roi.append(hist)
plt.subplot(224)
for i, col in enumerate(['b', 'g', 'r']):
        plt.plot(color_hist_roi[i], color=col)
plt.title('ROI Color Histogram')
plt.show()
```

**c) 实验结果截图**




**d) 实验小结**

通过这个实验学习了直方图的画法以及什么是直方图，对掩膜切片也有了更深的认识，在画直方图的时候也可以在三个通道画，对图像处理认识的更深了一些。

## 三. 任务 3

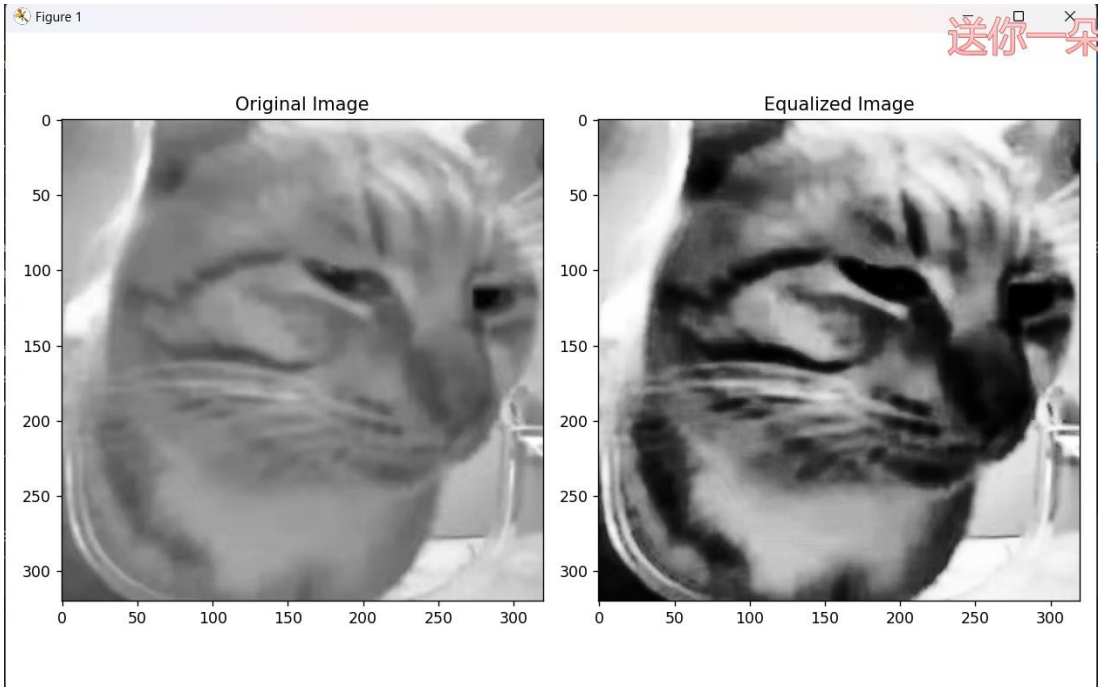
**a) 实验内容**

1.编程实现直方图均衡化，给出测试效果

**b) 核心代码**

```
def histogram_equalization(image):
    # 计算图像的直方图
    hist, bins = np.histogram(image.flatten(), 256, [0, 256])

    # 计算累积分布函数
    cdf = hist.cumsum()
    cdf_normalized = cdf * hist.max() / cdf.max()

    # 使用累积分布函数进行均衡化
    cdf_masked = np.ma.masked_equal(cdf, 0)
    cdf_masked = (cdf_masked - cdf_masked.min()) * 255 / (cdf_masked.max() -
cdf_masked.min())
    cdf_equalized = np.ma.filled(cdf_masked, 0).astype('uint8')

    # 将均衡化后的值映射回原始图像
    equalized_image = cdf_equalized[image]

    return equalized_image
```

**c) 实验结果截图**




**d) 实验小结**

明白了什么叫直方图均衡化，是通过调整像素大小来使整个图像的像素分布更均匀，通过编程了解了均衡化的原理。