《数字图像处理》实验报告

姓名：汪江豪 学号：22121630

实验 5

一. 任务 1

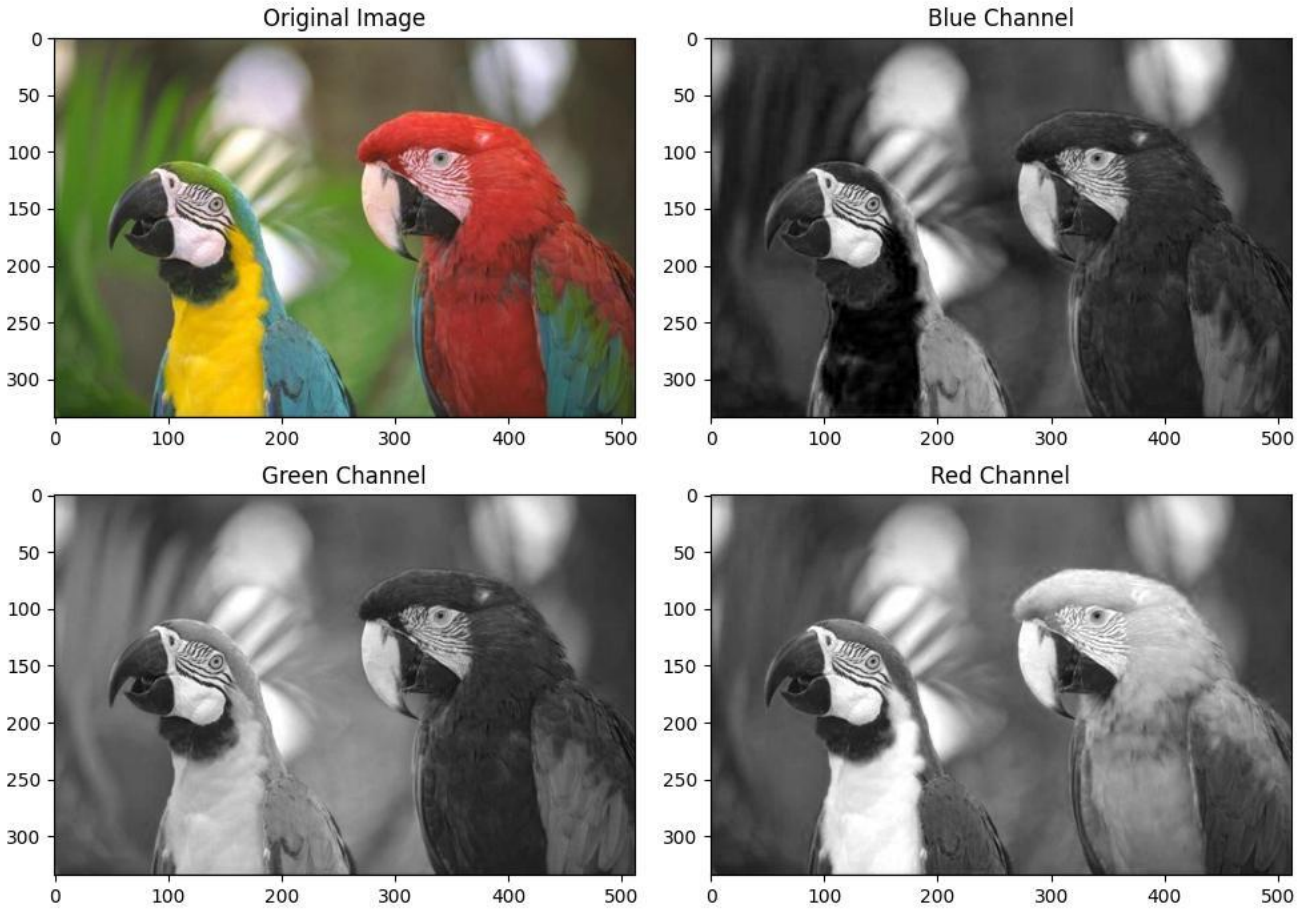
将彩色图像'araras.jpg 的 RGB 3 个通道用灰度图显示出来

a) 核心代码：

```
import cv2 as cv
import matplotlib.pyplot as plt

# 读取图片
img = cv.imread("./araras.jpg")

# 分离 RGB 三个通道
b, g, r = cv.split(img)

# 将每个通道转换为灰度图
gray_b = cv.cvtColor(cv.merge([b, b, b]), cv.COLOR_BGR2GRAY)
gray_g = cv.cvtColor(cv.merge([g, g, g]), cv.COLOR_BGR2GRAY)
gray_r = cv.cvtColor(cv.merge([r, r, r]), cv.COLOR_BGR2GRAY)

# 显示原图和灰度图
plt.figure(figsize=(10, 7))

plt.subplot(2, 2, 1)
plt.title("Original Image")
plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))

plt.subplot(2, 2, 2)
plt.title("Blue Channel")
plt.imshow(gray_b, cmap="gray")

plt.subplot(2, 2, 3)
plt.title("Green Channel")
plt.imshow(gray_g, cmap="gray")
```

```
plt.subplot(2, 2, 4)
plt.title("Red Channel")
plt.imshow(gray_r, cmap="gray")

# 自动调整合适大小
plt.tight_layout()
plt.savefig("task1.jpg")
plt.show()
```

b) 实验结果截图



c) 实验小结

可以看到,在将 R,G,B 三个通道用灰度图显示出来后,对于原图红色部分,在红色通道的灰度图中,相应部分会更亮一些,而蓝色和绿色通道中红色相应部分就会暗一些.这加深了我对于彩色图像 RGB 数值的理解,同时也能发现不同色彩之间的关系,比如黄色会和绿色和红色更相关一些,而与蓝色更不相关.

二. 任务 2

实现彩色图像的直方图均衡化（可以用 OpenCV 自带函数）

方法 1：对 RGB 通道分别做直方图均衡化再合成

方法 2：转换到 HSV 空间，仅对亮度分量 V 用直方图均衡化，再转换回 RGB

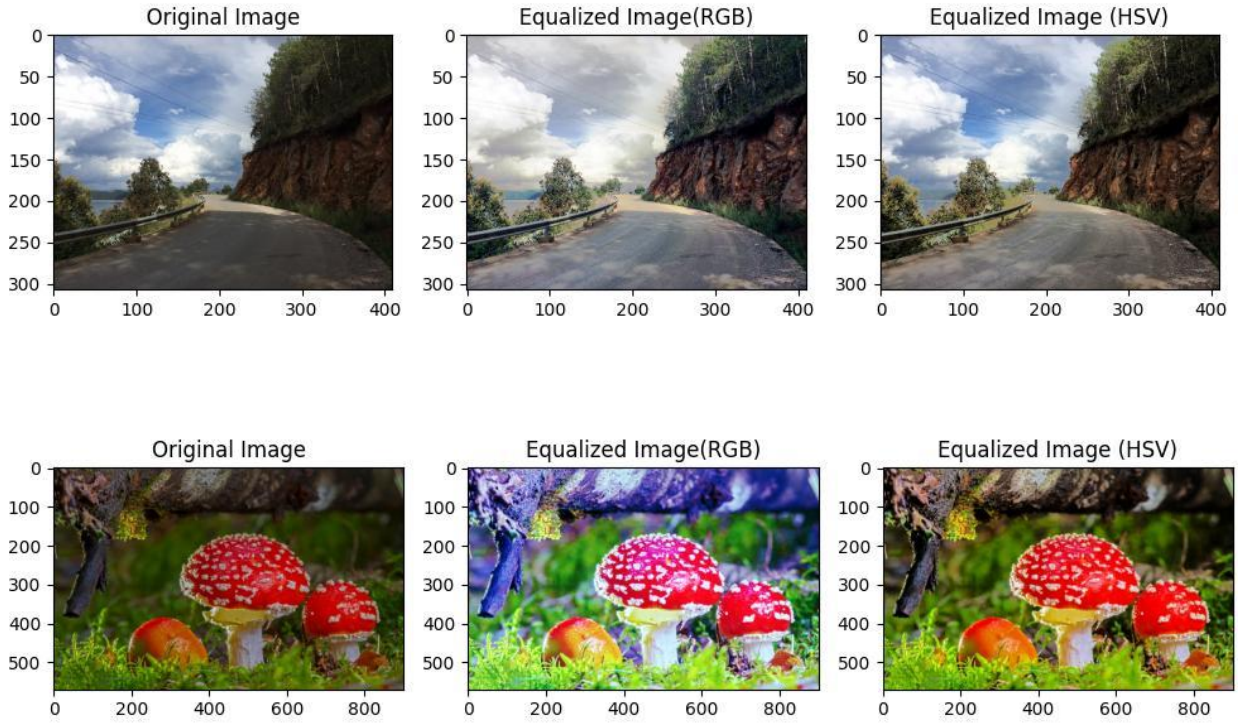
用 sky，mushroom 两个图片来对比两个方法的结果

a) 核心代码：

```python
import cv2 as cv
import matplotlib.pyplot as plt

# 彩色图像的直方图均衡化
def img_eq_rgb(img):
    # 分离 RGB 三个通道
    b, g, r = cv.split(img)

    # 对每个通道进行直方图均衡化
    b = cv.equalizeHist(b)
    g = cv.equalizeHist(g)
    r = cv.equalizeHist(r)
    # 合并三个通道
    equalized_img = cv.merge([b, g, r])
    return equalized_img

def img_eq_hsv(img):
    # 转换到 HSV 空间
    hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

    # 对亮度分量 V 进行直方图均衡化，三个维度分别表示高、宽和通道
    hsv[:, :, 2] = cv.equalizeHist(hsv[:, :, 2])

    # 转换回 RGB 空间
    equalized_img = cv.cvtColor(hsv, cv.COLOR_HSV2BGR)
    return equalized_img

# 显示原图和直方图均衡化后的图
def show(img, res_rgb, res_hsv):
    plt.figure(figsize=(10, 7))
    plt.subplot(1, 3, 1)
    plt.title("Original Image")
    plt.imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))
    plt.subplot(1, 3, 2)
    plt.title("Equalized Image(RGB)")
    plt.imshow(cv.cvtColor(res_rgb, cv.COLOR_BGR2RGB))
    plt.subplot(1, 3, 3)
    plt.title("Equalized Image (HSV)")
    plt.imshow(cv.cvtColor(res_hsv, cv.COLOR_BGR2RGB))
    plt.tight_layout()
    plt.savefig("task2_mushroom.jpg")
```

```
    plt.show()


# 读取图片
img1 = cv.imread("./sky.bmp")
img2 = cv.imread("./mushroom.png")

res1_rgb = img_eq_rgb(img1)
res1_hsv = img_eq_hsv(img1)
show(img1, res1_rgb, res1_hsv)

res2_rgb = img_eq_rgb(img2)
res2_hsv = img_eq_hsv(img2)
show(img2, res2_rgb, res2_hsv)
```

b) 实验结果截图



c) 实验小结

采用 RGB 三种通道分别均衡化后,再合成,会发现与原图有较大色差,而在 HSV 空间中,仅对亮度分量 V 进行直方图均衡化后,图像整体的亮度会变得更加均衡.色差也不会有明显变化,适合调整过亮或过暗的图片.

三. 任务 3

编程将 green 图片中的人物抠出，并融合到 tree 图片中，力求融合结果自然

a) 核心代码：

```
import cv2 as cv
import numpy as np

# 读取图片
img = cv.imread("./green.png")
bg = cv.imread("./tree.jpg")

# 将green 转换为HSV 颜色空间
hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)

# 定义绿色的范围,三个值分别是H、S、V，表示色调、饱和度、亮度
lower_green = np.array([30, 100, 0])
upper_green = np.array([85, 255, 255])

# 创建掩膜，选择绿色背景
mask = cv.inRange(hsv, lower_green, upper_green)

# 反转掩膜,选择人物
mask_inv = cv.bitwise_not(mask)

# 使用掩膜将人物从 green 中抠出
person = cv.bitwise_and(img, img, mask=mask_inv)

# 将tree 调整为与green 相同大小，宽、高
bg_resized = cv.resize(bg, (img.shape[1], img.shape[0]))

# 使用掩膜将背景从 tree 中抠出
bg_masked = cv.bitwise_and(bg_resized, bg_resized, mask=mask)

result = cv.add(person, bg_masked)

cv.imwrite("task3.jpg", result)
cv.imshow("result", result)
cv.waitKey(0)
cv.destroyAllWindows()
```

b) 实验结果截图

c) 实验小结

在抠图过程中,我选择在 HSV 空间中,从色调,饱和度和亮度三个维度对绿色进行选择,三个数值需要反复调整,否则会有较多绿色残余.为了更好地融合进背景图,可以将背景图变换到与 green 图片相同尺寸,再将背景图中人物部分的像素抠除,再将人物融合进背景图,从而达到更加自然的效果.