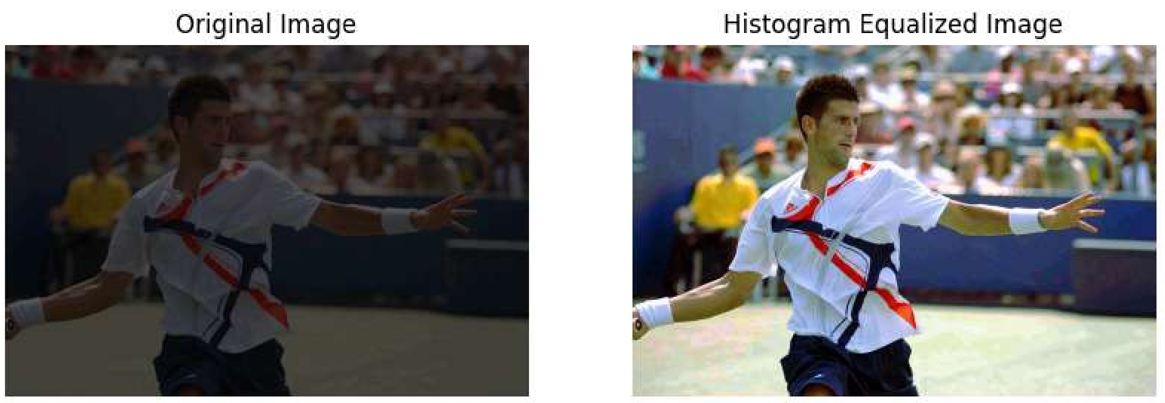
hwk11:彩色图像处理实践

对图像rgb三个通道分别进行直方图均衡

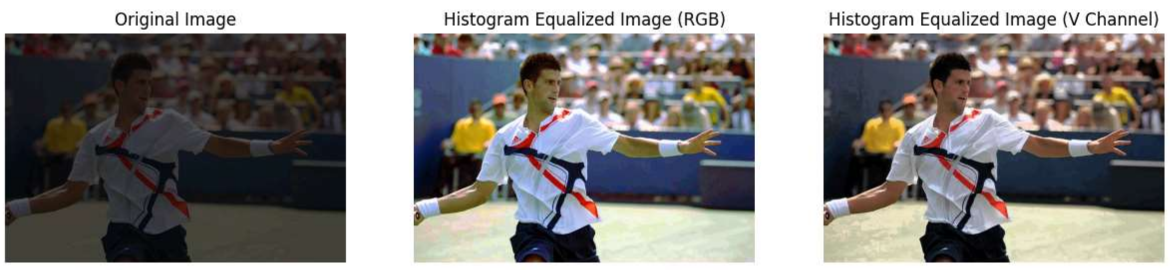
In [23]:
```python
import numpy as np
import matplotlib.pyplot as plt
from skimage import io

image = io.imread('novak-dark.jpg')

#直方图均衡化函数
def histogram_equalization(channel):
    #直方图
    hist, bins = np.histogram(channel.flatten(), 256, [0, 256])
    #计算CDF
    cdf = hist.cumsum()
    cdf_normalized = cdf * hist.max() / cdf.max()

    #使用线性插值找到新的像素值
    cdf_m = np.ma.masked_equal(cdf, 0)
    cdf_m = (cdf_m - cdf_m.min()) * 255 / (cdf_m.max() - cdf_m.min())
    cdf = np.ma.filled(cdf_m, 0).astype('uint8')

    return cdf[channel]

#对RGB三个通道分别进行直方图均衡化
r_equalized = histogram_equalization(image[:, :, 0])
g_equalized = histogram_equalization(image[:, :, 1])
b_equalized = histogram_equalization(image[:, :, 2])

#合并通道
equalized_image_rgb = np.stack((r_equalized, g_equalized, b_equalized), axis=2)

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(equalized_image_rgb)
plt.title('rgb Equalized Image')
plt.axis('off')
plt.show()
```



对HSV空间中的V通道进行均衡

In [28]:
```python
from skimage import io, color

# 合并均衡化后的通道
equalized_image_rgb = np.stack((r_equalized, g_equalized, b_equalized), axis=2)

# 将图像转换到HSV空间
hsv_image = color.rgb2hsv(image)

# 对V通道进行直方图均衡化
v_equalized = histogram_equalization((hsv_image[:, :, 2] * 255).astype(np.uint8)
hsv_image[:, :, 2] = v_equalized / 255.0

# 将图像转换回RGB空间
equalized_image_hsv = color.hsv2rgb(hsv_image)


# 显示原始图像和均衡化后的图像
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].imshow(image)
axes[0].set_title('Original Image')
axes[0].axis('off')

axes[1].imshow(equalized_image_rgb)
axes[1].set_title('Histogram Equalized Image (RGB)')
axes[1].axis('off')

axes[2].imshow(equalized_image_hsv)
axes[2].set_title('Histogram Equalized Image (V Channel)')
axes[2].axis('off')
```

Out[28]:
```
(-0.5, 1075.5, 719.5, -0.5)
```


Original Image
Histogram Equalized Image (RGB)
Histogram Equalized Image (V Channel)

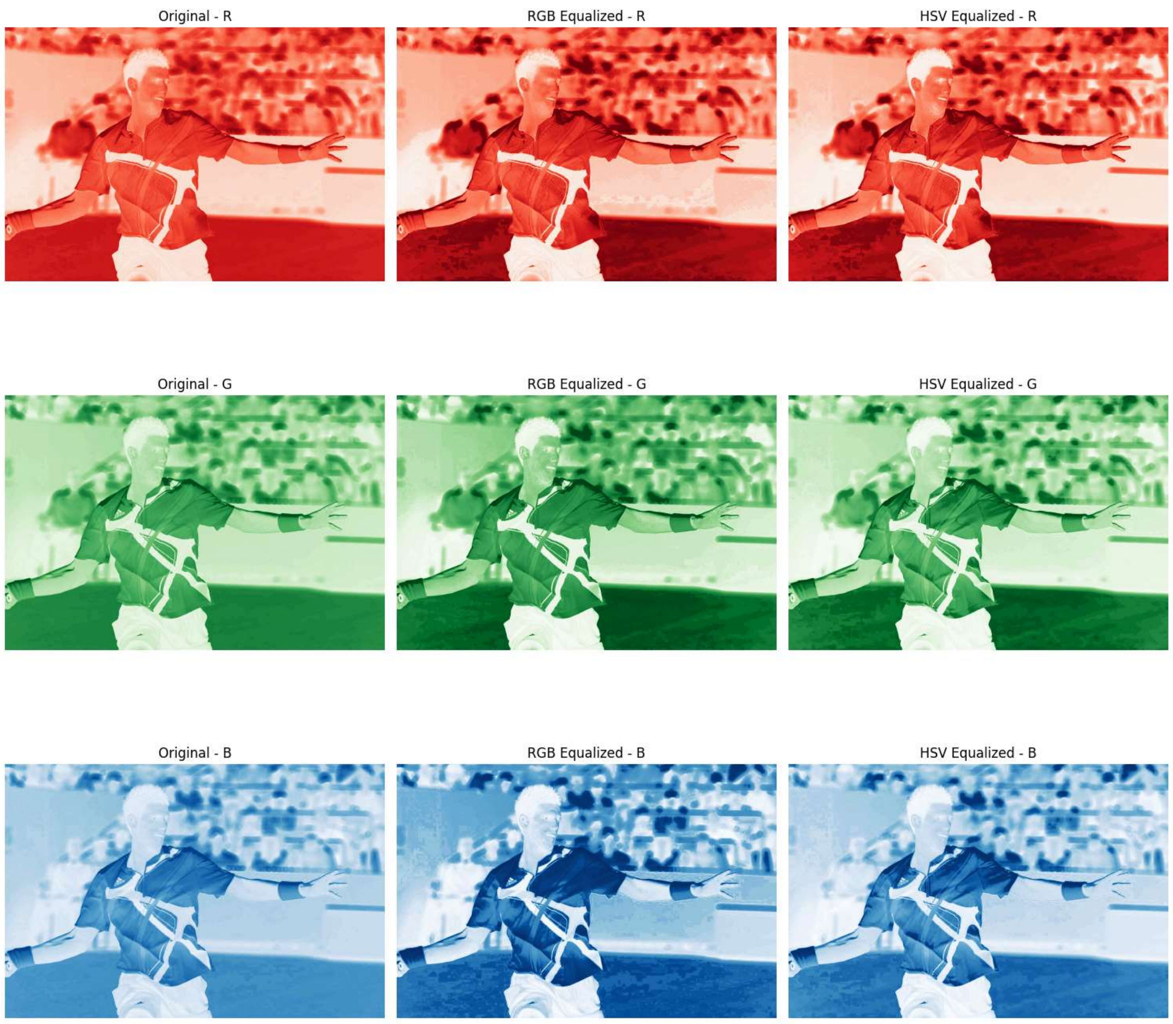
可以看到，HSV方法均衡对图像的处理效果更好，对RGB三个通道独立地进行直方图均衡后再合成一张图片，图中人物的肤色有些偏绿，整张图色调有点偏蓝。

这是因为RGB均衡没有考虑色彩之间的关联，每个通道独立均衡可能会引入颜色失真。

HSV中的**V通道直接描述亮度**，H和S则表示色相和饱和度。均衡时只用调整V通道而不影响另外两个独立的通道，因此能有效增强亮度细节而不破坏颜色。

In [ ]:

```python
# 对两张不同处理的图片进行RGB通道对比
import matplotlib.pyplot as plt
# 绘制对比图
def plot_channels(image, title, position):
    R, G, B = cv2.split(image)
    plt.subplot(3, 3, position)
    plt.imshow(R, cmap="Reds")
    plt.title(f"{title} - R")
    plt.axis("off")

    plt.subplot(3, 3, position + 3)
    plt.imshow(G, cmap="Greens")
    plt.title(f"{title} - G")
    plt.axis("off")

    plt.subplot(3, 3, position + 6)
    plt.imshow(B, cmap="Blues")
    plt.title(f"{title} - B")
    plt.axis("off")

plt.figure(figsize=(15, 15))

# 原图通道
plot_channels(image, "Original", 1)

# RGB均衡通道
plot_channels(equalized_image_rgb, "RGB Equalized", 2)

# HSV均衡通道
plot_channels(equalized_image_hsv, "HSV Equalized", 3)

plt.tight_layout()
plt.show()
```

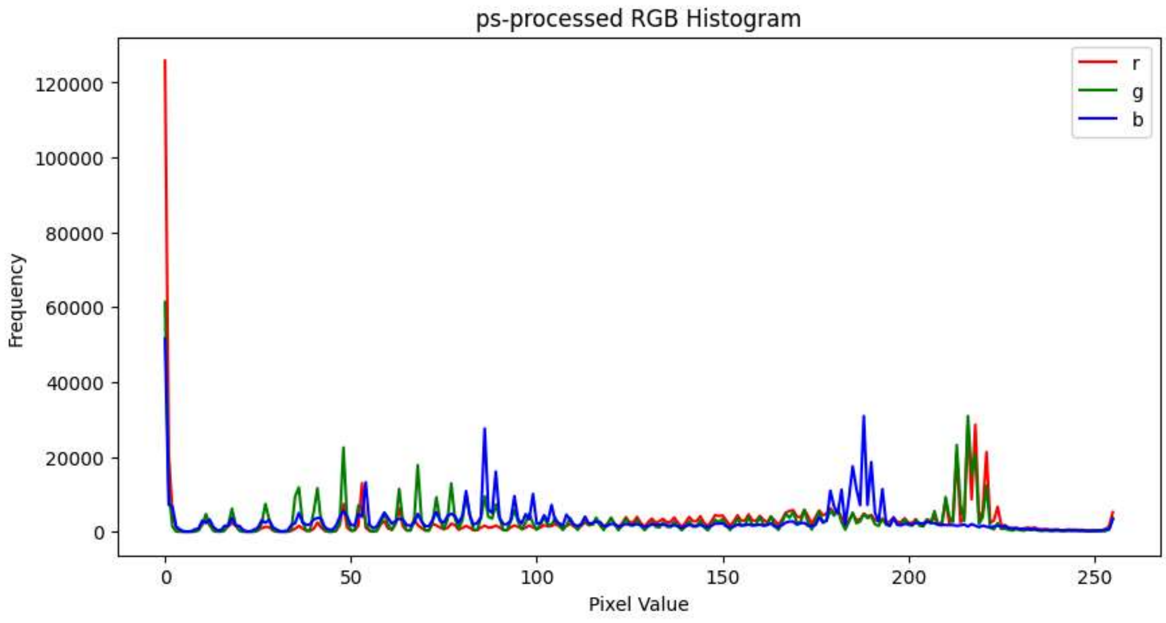


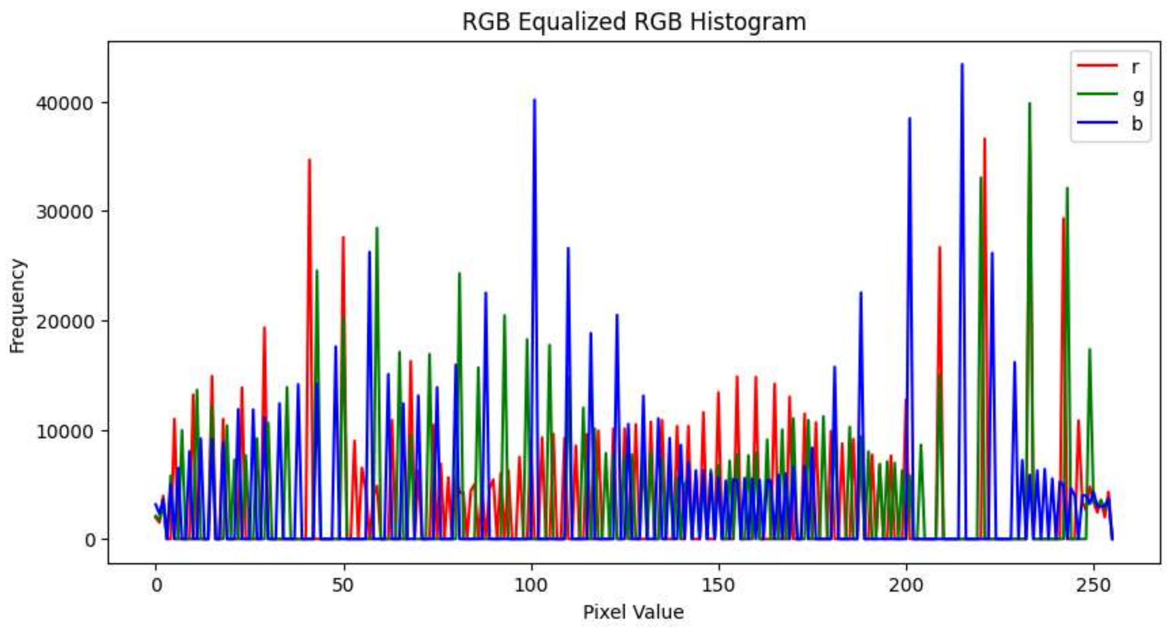
可以看出，HSV方法均衡的对于原图像的细节保留的更好（尤其是蓝色通道比较明显）

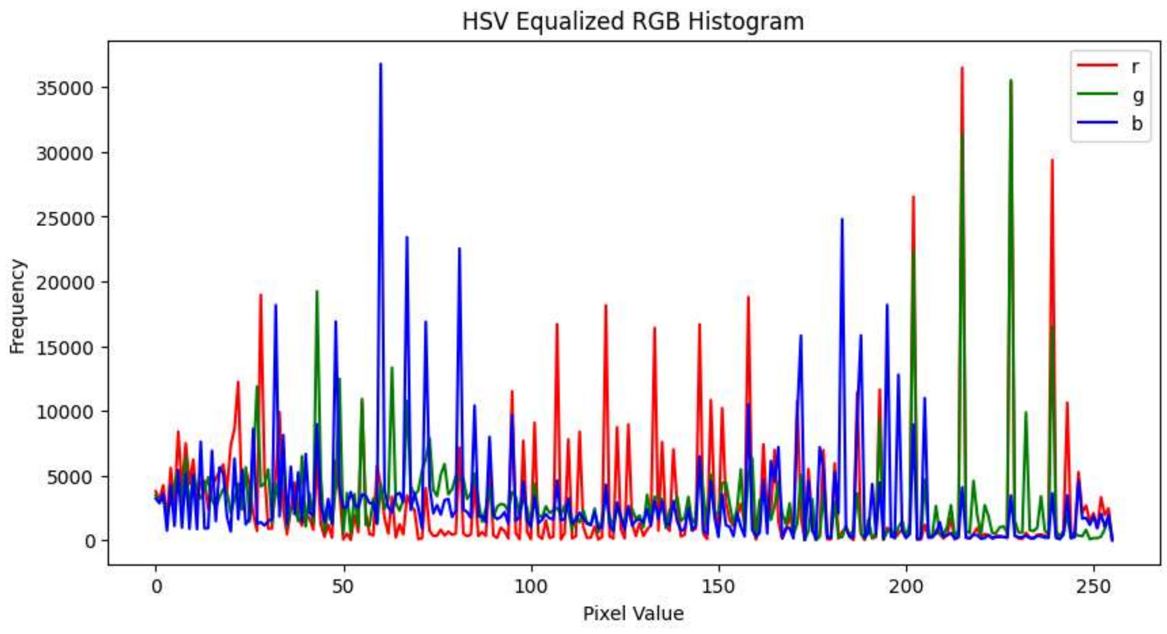
展示ps处理图片并对比rgb通道曲线

In [45]:

```
image = io.imread('novak-dark.jpg')
img_ps = io.imread('novak-ps.jpg')

# RGB通道直方图分析
def plot_rgb_histograms(image, title):
    image_8u = (image * 255).astype(np.uint8) if image.dtype == np.float64 else
    colors = ('r', 'g', 'b')
    plt.figure(figsize=(10, 5))
    for i, color in enumerate(colors):
        hist, bins = np.histogram(image_8u[:, :, i].flatten(), bins=256, range=[
        plt.plot(hist, color=color)
    plt.title(f'{title} RGB Histogram')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')
    plt.legend(colors)
    plt.show()

#展示ps处理后的图像
plt.imshow(img_ps)
plt.axis('off')

# 分别绘制原始图像和处理后图像的RGB通道直方图
plot_rgb_histograms(img_ps, "ps-processed")
plot_rgb_histograms(equalized_image_rgb, "RGB Equalized")
plot_rgb_histograms(equalized_image_hsv, "HSV Equalized")
```







很明显，HSV方法处理的图像rgb曲线与ps处理的更相近，尤其是在value为200附近的通道曲线较为明显。