本科生实验报告

学生姓名：　丁晓琪

学生学号：　22336057

专业名称：　计科

一：直方图均衡

此处直方图均衡直接参照第一次编程实验的直方图均衡

输入：需要直方图均衡的矩阵，矩阵取值为0-255的整数

输出：均衡后的矩阵，取值范围为0-255的整数

```python
#归一化处理，计算直方图
def calculate_histogram(channel_matrix):
    histogram=[0]*256
    width=channel_matrix.shape[1]
    height=channel_matrix.shape[0]
    sum=width*height
    for x in range(0,height):
        for y in range(0,width):
            gray_value=channel_matrix[x][y]
            histogram[int(gray_value)]+=1
    for i in range(0,len(histogram)):
        histogram[i]/=sum
    return histogram


#直方图均衡
def histogram_equalization(channel_matrix):
    histogram=calculate_histogram(channel_matrix)
    #histogram=np.histogram(channel_matrix,256)
    cdf=[0]*256 #累计均衡分布
    cdf[0]=histogram[0] #初始化
    width=channel_matrix.shape[1]
    height=channel_matrix.shape[0]
    for i in range(1,len(cdf)):
        cdf[i]=cdf[i-1]+histogram[i]

    after_matrix=np.zeros((height,width))
```

```
    for x in range(0,height):
        for y in range(0,width):
                gray_values=channel_matrix[x][y]
                sk=(255)*cdf[gray_values]
                after_matrix[x][y]=int(sk)

    return after_matrix
```

二：HSI空间直方图均衡

1. RGB空间转HSI空间

- 理论：注意下面公式中RGB的取值范围为0-1（做过归一化处理）。S,I的取值范围为[0,1], θ的取值范围为（角度）[0,360] (弧度上[0,π]) , H的取值范围为（弧度上）[0,2π]（后续需要做归一化处理）

色调H分量由夹角求出

$$H = \begin{cases} \theta & B \le G \\ 360 - \theta & B > G \end{cases}$$

$$\theta = \arccos\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-G)(G-B)\right]^{\frac{1}{2}}} \right\}$$

饱和度S分量

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right]$$

亮度I分量

$$I = \frac{1}{3}(R+G+B)$$

- 实现：参照上述公式实现, 注意有些地方为了防止除0错误加上了极小值(1e-10)

```
def RGB_TO_HSI(rgb_image):
    # 归一化rgb值
    rgb_image = rgb_image.astype(np.float32)/255.0
    # 分离通道
    R,G,B = rgb_image[:, :, 0], rgb_image[:, :, 1], rgb_image[:, :, 2]
    # I空间
    I=(R+G+B)/3.0
    # S空间
    min_rgb=np.minimum(np.minimum(R, G),B)
    S=1-(3/(R+G+B+1e-10))*min_rgb
    # H空间 0-1
    theta=np.arccos(0.5*((R-G)+(R-B))/np.sqrt((R-G)**2+(R-B)*(G-B)+1e-
10))
    mask=(B<=G)  # 掩码，B<=G为1
    H=np.where(mask,theta/(2*np.pi),(2*np.pi-theta)/(2*np.pi))
    # 合并
    hsi_image=np.stack((H,S,I),axis=-1)
    return hsi_image
```

2. HSI空间转为RGB空间

- 理论：注意输入的HSI值的取值范围都是[0,1], 计算得到的RGB取值范围也是0-1

已知HSI值在[0,1]内，求同一区间对应的RGB值，适用的公式取决于H存在的不同扇区:

RG扇区 $0^0 \le H < 120^0$

$$R = I\left[1 + \frac{S\cos H}{\cos(60^0 - H)}\right] \quad G = 3I - (R+B) \quad B = I(1-s)$$

GB扇区 $120^0 \le H < 240^0 \quad H = H - 120^0$

$$G = I\left[1 + \frac{S\cos H}{\cos(60^0 - H)}\right] \quad B = 3I - (R+G) \quad R = I(1-s)$$

BR扇区 $240^0 \le H < 360^0 \quad H = H - 240^0$

$$B = I\left[1 + \frac{S\cos H}{\cos(60^0 - H)}\right] \quad R = 3I - (G+B) \quad G = I(1-s)$$

- 实现：参照上述公式实现

```python
def HSI_TO_RGB(hsi_image):
    H,S,I = hsi_image[:, :, 0], hsi_image[:, :, 1], hsi_image[:, :, 2]

    R=np.zeros_like(H)
    G=np.zeros_like(H)
    B=np.zeros_like(H)
    #0<=H<120（角度）
    idx1=(0<=H)&(H<120/360)
    B[idx1]=I[idx1]*(1 - S[idx1])
    R[idx1]=I[idx1]*(1+
(S[idx1]*np.cos(H[idx1]*2*np.pi)/np.cos((np.pi/3)-(H[idx1]*2*np.pi))))
    G[idx1]=3*I[idx1]-(R[idx1]+B[idx1])
     #120<=H<240
    idx2=(120/360<=H)&(H<240/360)
    H_region2=H[idx2]-120/360
    R[idx2]=I[idx2]*(1-S[idx2])
    G[idx2]=I[idx2]*(1+
(S[idx2]*np.cos(H_region2*2*np.pi)/np.cos((np.pi/3)-
(H_region2*2*np.pi))))
    B[idx2]=3*I[idx2]-(R[idx2]+G[idx2])

    #240<=H<360
    idx3=(240/360<=H)&(H<1)

    H_region3=H[idx3]-240/360
    G[idx3]=I[idx3]*(1-S[idx3])
    B[idx3]=I[idx3]*(1+
(S[idx3]*np.cos(H_region3*2*np.pi)/np.cos((np.pi/3)-
(H_region3*2*np.pi))))
    R[idx3]=3*I[idx3]-(G[idx3]+B[idx3])

    RGB_image=np.stack((R, G, B), axis=-1)
    RGB_image=(RGB_image*255).astype(np.uint8)
    return RGB_image
```

3. 对I空间直方图均衡：

- 步骤：
  - RGB空间转为HSI空间
  - 先将取值范围为（0-1）的I空间映射成0-255的整数
  - 对映射后的I空间直方图均衡
  - 均衡后重新归一化，并组合回HSI空间
  - HSI空间转为RGB表示
- 实现：

```
def HSI_histogram_equalization(rgb_image):
      # 转到RGB空间
      hsi_image=RGB_TO_HSI(rgb_image)
      H, S, I = hsi_image[:, :, 0], hsi_image[:, :, 1], hsi_image[:, :,
2]
      print(np.max(S))
      # I空间映射到0-255
      I_8bit=np.clip(np.round(I*255),0,255).astype(np.uint8)
      # I空间直方图均衡化
      I_equalized_8bit=histogram_equalization(I_8bit)
      # 重新归一化
      I_equalized_normalized=I_equalized_8bit.astype(np.float32)/255.0
      # 重新组合为HSI空间
      hsi_equalized=np.stack((H, S, I_equalized_normalized), axis=-1)
      # 转到RGB空间
      RGB_hsi_equalized=HSI_TO_RGB(hsi_equalized)
      plt.subplot(1,4,3)
      plt.imshow(RGB_hsi_equalized)
      plt.axis('off')
      plt.title('HSI_histogram_equalization(image)')
```

三：RGB空间直方图均衡

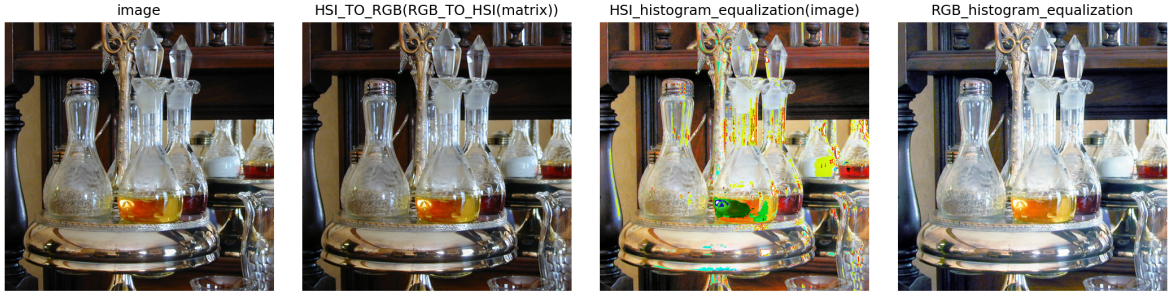
- 步骤：分别对RGB三通道直方图均衡，均衡后重新组合
- 实现：

```
def RGB_histogram_equalization(rgb_matrix):
    red_channel = rgb_matrix[:, :, 0]   # 红色通道
    green_channel = rgb_matrix[:, :, 1] # 绿色通道
    blue_channel = rgb_matrix[:, :, 2]  # 蓝色通道
    # 对每个通道直方图均衡
    after_red_channel=histogram_equalization(red_channel)
    after_green_channel=histogram_equalization(green_channel)
    after_blue_channel=histogram_equalization(blue_channel)
    # 防止超出0-255范围（要加不然会曝光）
    after_red_channel = np.clip(after_red_channel, 0,
255).astype(np.uint8)
    after_green_channel = np.clip(after_green_channel, 0,
255).astype(np.uint8)
    after_blue_channel = np.clip(after_blue_channel, 0,
255).astype(np.uint8)
```

```
    # 组合
    combined_image=np.stack((after_red_channel, after_green_channel, after_blue_channel), axis=-1)
    plt.subplot(1,4,4)
    plt.imshow(combined_image)
    plt.axis('off')
    plt.title('RGB_histogram_equalization')
```

四：实验结果

效果：



- HSI_TO_RGB(RGB_TO_HSI(matrix))：为了检验RGB和HSI空间的相互转换的实现是否正确，RGB图像先转为HSI空间表示再转为RGB空间后的图像和原图像一致，则实现正确
- HSI_histogram_equalization(image): 对I空间（图像平均灰度空间）做直方图均衡后，偏暗的地方变得明亮，个别地方细节更明显，图像对比度更强。但是出现图像色彩失真的现象，可能是由于直方图均衡时的误差造成
- RGB_histogram_equalization(image): 图像整体变亮，对比度增强。但是色彩上出现失真，原图整体是偏棕色，直方图均衡后整体偏白，失去自然色彩。