数字图像处理实验一

161220135 - 吴德亚- 3524214204@qq.com-15996232665

一：灰度图像处理

首先计算每种灰度值（0 ~ 255）占该图片的比例，再累计求和乘以灰度值得到每一种像素映射的结果。

$$\int_0^{\nu_B} H_B(D)dD = \frac{D_B}{L} \times A_0 = \frac{f(D_A)}{L} \times A_0$$

A0是总像素个数　　L是灰度级个数

代码如下：

```
function [output2] = hist_equal(input_channel)  % 通道直方图均衡化
    output2 = input_channel;
    [p_length,p_width] = size(input_channel);  %获取输入矩阵的长，宽
    mapping = getmap(input_channel);              % 调用函数getmap( 如下文 )获取映射数组
    for i = 1:p_length
        for j = 1:p_width
            output2(i,j) = mapping(input_channel(i,j)+1);  % 映射
        end
    end
end
```

hist_equal 函数调用**getmap** 获取映射数组，然后按照输入通道进行映射得到**ouput2**均衡化后的通道。

```
function[mapping] = getmap(input_channel)          % 获取通道的映射关系
    %you should complete this sub-function
    [p_length,p_width] = size(input_channel);
    num_count = zeros(256);      % 计算每个灰度值的像素个数
    for i=1:p_length
```

```
            for j=1:p_width
                gray_value = input_channel(i,j);
                num_count(gray_value+1) = num_count(gray_value+1)+1;
            end
        end
        prob_count = zeros(256);
        dest_count = zeros(256);   % 存储映射表
        for i = 1:256
            prob_count(i) = num_count(i) / (p_length * p_width);
            prob_sum = 0;
            for j = 1:i
                prob_sum = prob_sum + prob_count(j);  % 概率求和
            end
            dest_count(i) = prob_sum * 255;   % 共有255个灰度级
        end
        mapping = fix(dest_count);     % 截尾取整
    end
end
```

输入图片（左），输出图片（右）：

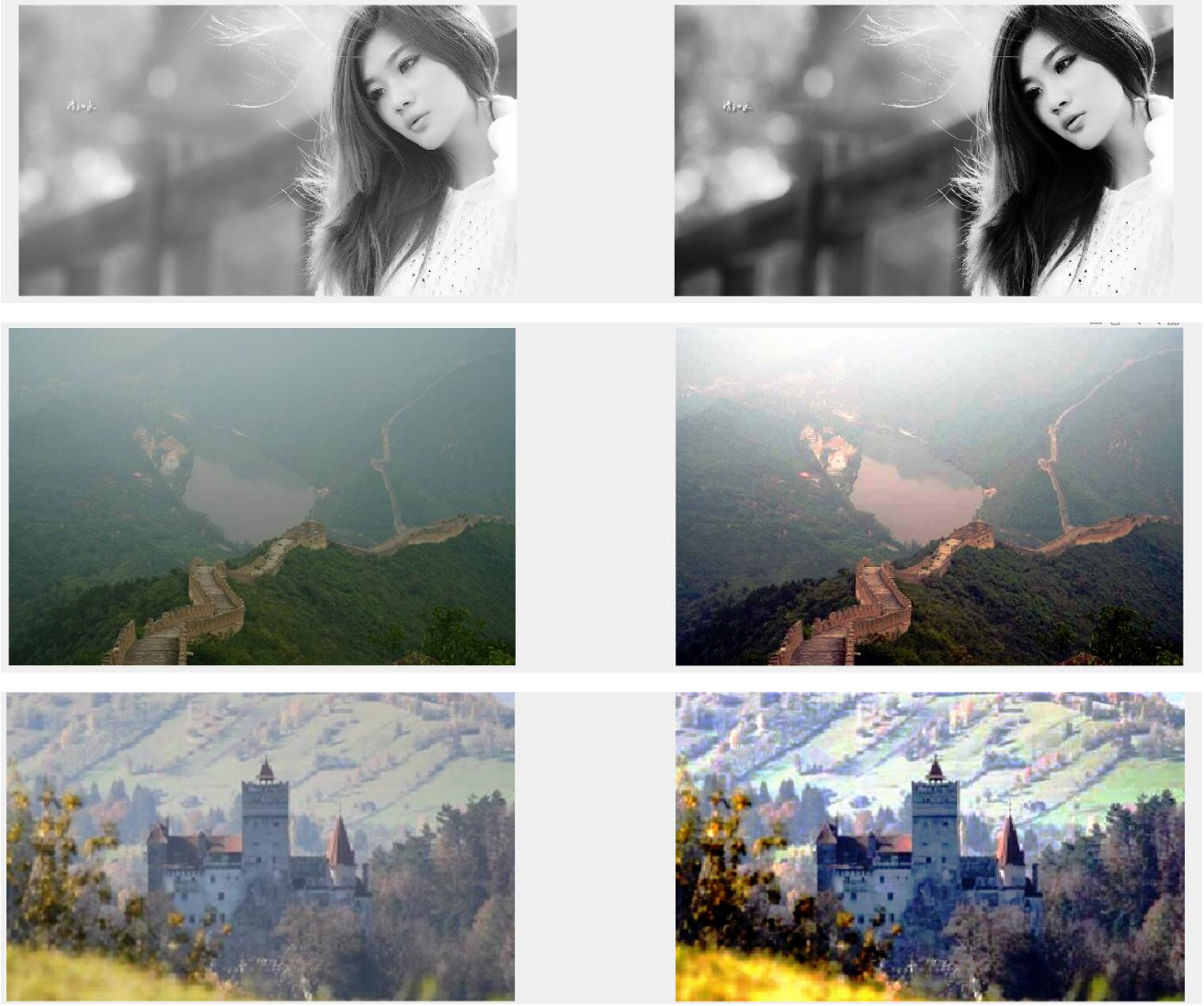

二：彩色图像处理

方法一：

直接按照灰度函数中的直方图均衡化的方法即可。

```
function [color_output] = color_equal(input_image)
        %方法一：直接对每个通道进行直方图均衡化
         r=input_image(:,:,1);
         v=input_image(:,:,2);
         b=input_image(:,:,3);
         r = hist_equal(r);  % 调用均衡化图像
         v = hist_equal(v);
         b = hist_equal(b);
         color_output = cat(3,r,v,b);
end
```

输入图（左），输出图（右）：



从以上三幅图片可以看出，对R, G, B三个通道分别做直方图均衡化可以达到增强图像的效果。

方法二:

现将RGB图片转换成HSV图像。然后再将s 或 v 值域0~1映射到0 ~ 255，利用直方图均衡化做均衡，最后再转换到0~1区间内。

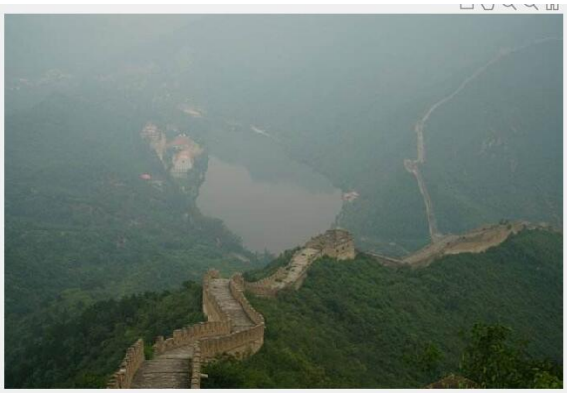
```
        hsv_image = rgb2hsv(input_image);
        h = hsv_image(:,:,1);
        s = hsv_image(:,:,2);
        v = hsv_image(:,:,3);
        v = hsv_hist_equal(v);  %色相
        s = hsv_hist_equal(s);  %亮度
        color_output = hsv2rgb(h,s,v);     % 转换成HSV后，再转成RGB
```

hsv_hist_equal函数调用上述实现的hist_equal函数。

```
    function[output2] = hsv_hist_equal(input_channel)
        output2 = input_channel;
        [p_length,p_width] = size(input_channel);
```

```
        channel_one = uint8(zeros(p_length,p_width));
        for i = 1:p_length
            for j = 1:p_width
                channel_one(i,j) = input_channel(i,j)*255;  % 映射到0 ~ 255
            end
        end
        k = hist_equal(channel_one);    %调用单通道处理函数
        for i = 1:p_length
            for j = 1:p_width
                output2(i,j) = (double(k(i,j)))/255;  % 映射到0~1
            end
        end
    end
```

输入图像（左），输出图像（右）：

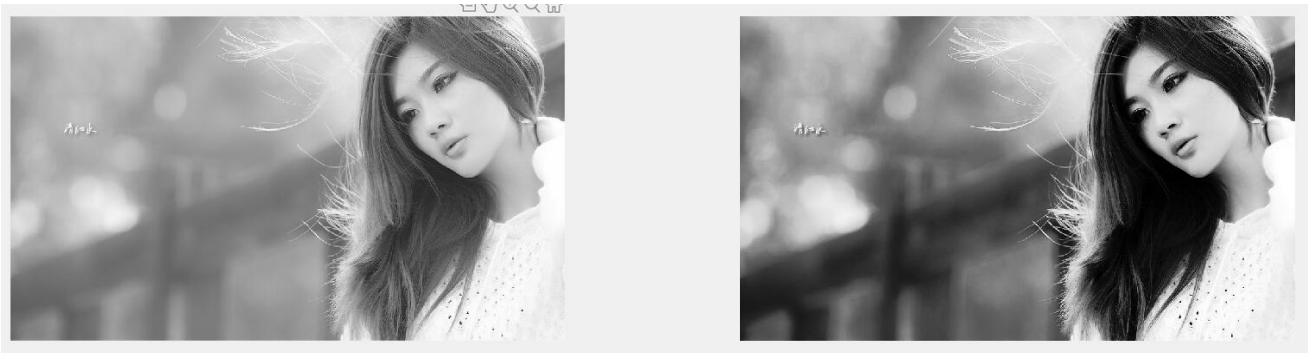



可以看到通过对色相（S),亮度(V)的均衡化，可以使颜色更突出（图片中上部分的山峰，建筑）。

三：问题和解决

（1）在运用（RGB -- HSV) 方法解决没有彩色（仅黑白）图片处理时，不能进行S(色相)的均衡化，否则会出现色相均衡化导致的其他颜色出现（例如红色）。


故，直接直方图均衡化或者不对S进行处理。



(2) 在代码书写过程中忽略了 **double** 和 **(uint8)**强制类型转换，导致运行结果和预期相差较大,最终排查找到了错误。