数字图像处理 作业一报告

姓名：王瀚霖
学号：181860092
邮箱：603102048@qq.com

一、实现细节

本次作业要求完成`Histogram_equalization.m`中的相关内容，来实现灰度图像直方图的均衡化。具体而言，需要实现函数`hist_equal`来对灰度图像进行直方图均衡化处理。这样，对于简单的灰度图像，直接调用该函数即可实现；对于彩色图像，框架代码给出的方法是分别对R、G、B三个通道进行直方图均衡化处理后合并；除此之外，本次作业中还尝试利用了`HSI`与`HSV`颜色模型对彩色图像进行处理。

- ### 灰度图像直方图均衡化函数`hist_equal`实现：

  灰度图像直方图均衡化函数`hist_equal`的内容就是通过实现一个变换函数，使得变换后图像的概率密度成为均匀分布。根据所学知识，对于有连续概率密度的图像，变化函数为：

  $$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

  对于离散的直方图，记n_j为灰度值为j的像素点的数目，则变化函数为：

  $$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$$

  本次作业中，我们要处理的是离散的直方图；同时，考虑到处理的灰度图像与彩色RGB图像一般均为256级灰度，所以本次实现的`hist_equal`函数处理的就是以`uint8`数据类型存储的图像(该想法可由`matlab`自带的库函数`im2uint8`得到保证。即对于传入的图像首先调用`im2uint8`函数来转化为`uint8`存储)。

  结合以上考虑及公式，可知要实现的`hist_equal`函数中，L取256；此外还需要统计得到$\sum_{j=0}^{k} n_j$(k=0, 1, 2……255)的值以及M、N的

值；得到以上数据后，只需要遍历整张图像，利用公式计算每一个像素点变化后的灰度并赋值即可。

根据以上思路，`hist_equal`函数首先需要将图片转化为`uint8`数据类型存储：

```
input_channel = im2uint8(input_channel);%统一按照8位整数来存
```

接着，需要统计$\sum_{j=0}^{k} n_j$(k=0, 1, 2……255)的值；为此，首先要统计不同灰度值的像素点个数，这一点可通过调用库函数`imhist`得到实现(与遍历统计的方法实现的效果一样)：

```
data = imhist(input_channel); %利用imhist函数得到各灰度值像素点的数目
```

这样，数组`data`就记录了不同灰度值像素点的数目。具体而言，`data(i)`记录了灰度值为`(i-1)`的像素点的数目(i = 1, 2, ……)；由此，从前到后遍历`data`数组，每次将前一项加到后一项上：

```
[length,t] = size(data);%得到data的长度length
    for i=2:length
        data(i) = data(i-1) + data(i);
        %累加，计算后data(i)表示灰度值小于等于i-1的像素点数总和
```

经上述处理，`data`就记录了我们需要的$\sum_{j=0}^{k} n_j$；接下来的M, N值获取十分简单，只需要调用库函数`size`即可得到输入图像的行、列数：

```
[r,c] = size(input_channel);
```

由此，我们得到了所需的全部数据。根据公式不难得出这里的计算公式为：

$$s_k = T(r_k) = [\frac{255}{r * c} * data(r_k + 1)]$$

PS:注意data(r_k)记录的是灰度值小于等于$r_k - 1$的像素点数目总和，而非小于等于r_k的像素点总和，因此用data($r_k + 1$);同时最后的结果要取整，这里利用`round`函数进行四舍五入；

因此，利用二重循环对输入图像进行遍历，代入上述式子进行计算，将得到的结果赋值给返回值`output2`：

```
    for i = 1:r
        for j = 1:c
            output2(i,j) = round(255 *
data(input_channel(i,j)+1) / (r*c));
            %二重循环代入公式，实现均衡化
        end
    end
```

由此，实现了直方图均衡化函数`hist_equal`。

- ## RGB图像处理实现：

1. **对R、G、B三个通道分别做直方图均衡化处理后合并：**

即框架代码中使用的处理方法；这种方法的处理思想较为简单。从运行结果来看，与原图片相比，均衡化后的图片亮度得到了调整，但颜色发生了比较严重的失真现象。具体的实现方法即分离出输入图像的三个通道后调用三次`hist_equal`进行均衡化，而后利用`cat`合并得到处理后图像。

2. **引入HSI颜色模型，将RGB图像转化为HSI，对I进行均衡化处理后转回RGB：**

在HSI颜色模型中，H表示色调，S表示饱和度，I表示亮度。与RGB相比，HSI比较符合人眼对于景物的感知。同时在HSI模型中，I分量与彩色信息无关。因此可以仅对I分量作直方图均衡化，保留H和S这两个与人感受颜色相关的不变，再进行三个分量的合并，完成彩色图像的直方图均衡化处理。

根据以上思路，我们要完成RGB到HSI转化的函数`rgb2hsi`与HSI转回RGB的函数`hsi2rgb`；再利用这两个函数与`hist_equal`实现目标。

`rgb2hsi`的实现：

RGB到HSI的转换公式如下图所示：

色调	饱和度	亮度
$H=\begin{cases}\theta, & G\geq B\\ 2\pi-\theta, & G<B\end{cases}$ $where\ \theta=\cos^{-1}\left\{\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$	$S=1-\frac{3\min(R,G,B)}{R+G+B}$	$I=\frac{R+G+B}{3}$

利用上述公式，实现`rgb2hsi`即可。要注意的细节有两点：一是要注意除数不可以为0；　　二是当强度I为0时，色调H、饱和度S无定义；当S=0时，色调H无定义。具体实现如下：

首先，提取R、G、B三个通道的数据：

```
function [H,S,I] = rgb2hsi(rgb)
    rgb = 2double(rgb);%首先转化为double类型存储
    r = rgb(:, :, 1);
    g = rgb(:, :, 2);
    b = rgb(:, :, 3); %提取三个通道数据
```

接着，计算角度θ，记作w：

```
    %利用转化公式进行计算
    x = 2 * r - g - b;
    y = 2 * sqrt((r-g).^2 + (r-b).*(g-b));
    zero = find(y == 0);
    y(zero) = eps; %避免除数为0
    w = acos(x./y);
```

计算H、S、I，并利用`cat`进行合并：

```
     H = w;
     change = find(b > g);
     H(change) = 2 * pi - w(change);
     % 对不同范围的H分别赋值
     sum = r + g + b;
     zero = find(sum == 0);
     sum(zero) = eps;%避免除数为0
     S = 1 - (3 * min(min(r, g), b) ./ sum);
     I = sum / 3;
     zero = find(S == 0);
     H(zero) = 0;%S=0时H无定义
     zero = find(I == (eps/3));%即之前sum == 0的位置
     H(zero) = 0;
     S(zero) = 0;%I=0时H、S均无定义
     I = uint8(sum / 3);%强度取整
end
```

`hsi2rgb`的实现：

HSI到RGB的转换公式如下图所示：

$$0 \le H < \frac{2\pi}{3}\text{时}: \begin{cases} B = I(1-S) \\ G = 3I - (R+B) \\ R = I[1 + \frac{ScosH}{cos(\frac{\pi}{3}-H)}] \end{cases}$$

$$\frac{2\pi}{3} \le H < \frac{4\pi}{3}\text{时}, H = H - \frac{2\pi}{3}: \begin{cases} R = I(1-S) \\ B = 3I - (R+B) \\ G = I[1 + \frac{ScosH}{cos(\frac{\pi}{3}-H)}] \end{cases}$$

$$\frac{4\pi}{3} \le H < 2\pi\text{时}, H = H - \frac{4\pi}{3}: \begin{cases} G = I(1-S) \\ R = 3I - (R+B) \\ B = I[1 + \frac{ScosH}{cos(\frac{\pi}{3}-H)}] \end{cases}$$

利用上述公式，实现`hsi2rgb`即可。具体实现如下：

首先，提取H、S、I三个通道的数据：

```
function [rgb] = hsi2rgb(hsi)
   H = hsi(:, :, 1);
   S = hsi(:, :, 2);
   I = hsi(:, :, 3); %提取三个通道数据
```

接着，建立R、G、B三个空矩阵，每个矩阵的大小与输入图片的前两维一致：

```
 [r,c] = size(H);
   R = zeros(r,c);
   G = R;
   B = G;
   %建立R、G、B三个空矩阵
```

代入公式，分区域计算R、G、B：

```
  %利用转化公式进行计算,对不同区域分别处理
for i = 1:r
    for j = 1:c
        if((H(i,j) < 4 * pi / 3) && (H(i,j) >= 0))
            B(i,j) = I(i,j) * (1 - S(i,j));
            G(i,j) = 3 * I(i,j) - (R(i,j) + B(i,j));
            R(i,j) = I(i,j) * (1 + S(i,j) * cos(H(i,j)) /
cos(pi / 3 - H(i,j)));
        end

        if((H(i,j) >= 2 * pi / 3) && (H(i,j) < 4 * pi / 3))
            H(i,j) = H(i,j) - 2 * pi / 3;
```

```
            R(i,j) = I(i,j) * (1 - S(i,j));
            B(i,j) = 3 * I(i,j) - (R(i,j) + B(i,j));
            G(i,j) = I(i,j) * (1 + S(i,j) * cos(H(i,j)) /
cos(pi / 3 - H(i,j)));
        end

        if((H(i,j) >= 4 * pi / 3) && (H(i,j) < 2 * pi))
            H(i,j) = H(i,j) - 4 * pi / 3;
            G(i,j) = I(i,j) * (1 - S(i,j));
            R(i,j) = 3 * I(i,j) - (R(i,j) + B(i,j));
            B(i,j) = I(i,j) * (1 + S(i,j) * cos(H(i,j)) /
cos(pi / 3 - H(i,j)));
        end
    end
  end
```

最后，将R、G、B转化回整数并进行合并：

```
            R=uint8(R);
            G=uint8(G);
            B=uint8(B);
            rgb = cat(3,R,G,B);
```

使用HSI方法均衡化函数`hist_equal_HSI`的实现：

利用上述实现子函数，完成HSI均衡化：

首先，将RGB函数转化为HSI，得到H、S、I三通道的数据：

```
    function [output3] = hist_equal_HSI(input_channel)

    [H,S,I] = rgb2hsi(input_channel);
```

保持H、S不变，对I进行直方图均衡化处理后进行合并；调用`hsi2rgb`转换回RGB图像；注意实现的`hist_equal`函数的输出是以`uint8`数据类型存储的，因此均衡化后需要利用`im2double`函数将均衡化后的输出转化为`double`类型存储，与H、S保持一致：

```
    I_new = double(hist_equal(I)); %对i作均衡化处理
    output3 = cat(3,H,S,I_new);
    output3 = hsi2rgb(output3);
```

3. 引入HSV颜色模型,将RGB图像转化为HSV，对V进行均衡化处理后转回RGB：

在HSV颜色模型中，H表示色调，S表示饱和度，I表示明度。HSV是另一种较为直观的颜色模型。同样的，可以先将RGB图像转化为HSV，对V进行直方图均衡化处理后合并转换回RGB。转换函数`rgb2hsv`与`hsv2rgb`均为`matlab`的库函数，无需实现。只需要利用库函数与`hist_equal`即可实现函数`hist_equal_HSV`，实现的方法思路与`hist_equal_HSI`基本一样，不再赘述。

二、结果

实验设置：

1、实现过程中调用了`imhist`函数，因此需要为`matlab`安装图像处理模块；

2、对不同的图片进行测试时，只需要修改`test_histeq`中`imread`的文件名即可；

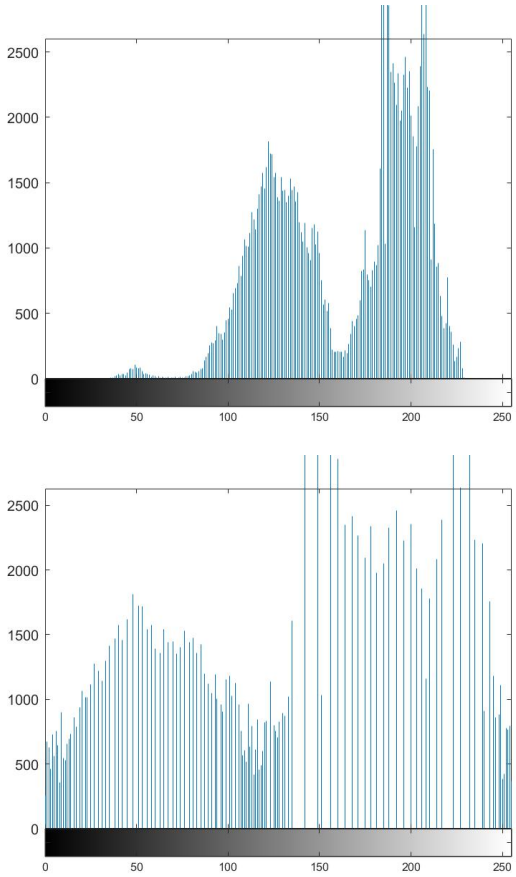
3、测试灰度图片时，运行结果会显示两张图片，依次为原图与直方图均衡化后的图像；测试彩色图片时，运行结果会显示四张图片，依次为：对RGB分别均衡化后合并的图像、利用HSI处理的图像、原图与利用HSV处理的图像。

实验结果：

1、灰度图像测试：

对`gray.jpg`进行测试结果对比如下：

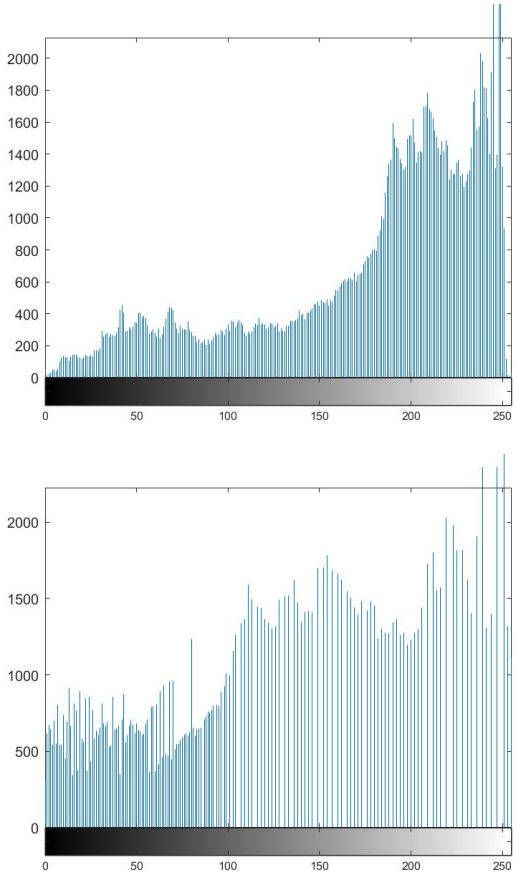
直方图对比如下图所示：



观察处理前后的图像，可以看到通过直方图均衡化，不同灰度的像素点个数较之前分布更加均衡，处理后图像的对比度有了明显的增强，从黑到白的渐变层次明显增多；与原图灰蒙蒙的效果相比，处理后的图像也变得更加清晰醒目，暗部细节有了更好的体现；

同样的，对`test_gray.jpg`进行测试结果对比如下：

直方图对比如下图所示：



观察可知，直方图均衡化处理效果同样很好。

2、彩色图像测试：

对`color.jpg`进行测试结果对比如下，其中第一张为原图，第二张为对RGB分别均衡化后合并的图像;第三张为利用HSI处理的图像，第四张为利用HSV处理的图像：

观察原图与经过不同方法处理后的图像，可以看到：原图是整体偏明亮的，但是经过不同方法处理后均出现了图像变暗的结果；其中对RGB分别均衡化后合并的方法得到的结果发生了比较严重的颜色失真，但云与烟雾的对比度得到了提升；HSI方法处理后图像稍微变暗，颜色略微失真，对比

度有所提高；HSV方法处理后对比度有所提升，但图像严重变暗，因而丢失了较多细节。综合来看，三种方法还属HSI方法效果最好，颜色没有严重失真，对比度也得到了提升；

`test.jpg`是一张雾气较重，对比度较低的风景图；对它进行测试结果对比如下(图片展示位置同上)：

观察原图与经过不同方法处理后的图像，可以看到：原图中雾气较大，远处景物细节不清楚，对比度低，经过不同方法处理后这一问题均得到了解决，图片的对比度均得到提高，细节也更多地展现了出来。但是，对RGB分别均衡化后合并的方法得到的结果仍然发生了轻微的颜色变化，这也使得图像内容显示得更加丰富；HSI方法处理后图像颜色的饱和度有所提高，色彩更加鲜艳；相比而言图像颜色略微失真；HSV方法处理后对比度有所提升，图像也有略微变暗的情况，但是整体上仍保持了原图的色彩。综合来看，三种方法还属HSV方法效果最好；

`test2.jpg`是作业一网站上的图片，整体与`color.jpg`风格较为相近；对它进行测试结果对比如下：

从结果来看，与`color.jpg`类似；效果最好的还是HSV方法；RGB方法颜色失真严重，HSI方法处理图像变得比较亮，不太符合原图；不同方法处理细节均有增加。

`test3.jpg`选取了一张整体偏暗，对比度适中的夜景图片；对它进行测试结果对比如下：

可以看到，不同方法处理后图像均变得亮了起来，细节也更加凸显出来。但HSI方法出现了颜色失真情况，可能与实现细节有关。综合来看，HSV效果最好。

综上所述，三种方法在提高图像对比度、增加细节而方面均有效果；但RGB方法容易造成严重的颜色失真，HSI与HSV方法相比之下更好。