**一、实验内容**

介绍所做实验包括的主要内容

编程实现灰度图像的直方图均衡化算法

（1）使用Matlab实现，不能直接调用自带的histeq函数

（2）采用函数调用的方式：定义功能函数g=myhisteq(f)

（3）将结果与使用Matlab自带的histeq函数得到的结果进行比较

**二、实验原理与步骤**

写出实验中算法实现的基本思路与步骤

实验原理：

1. 直方图均衡化：将原图像进行灰度变换，使变换后图像的灰度直方图呈现均匀分布。直方图均衡化的基本思想是对图像中像素个数多的灰度级进行扩展，而对图像中像素个数少的灰度进行压缩，从而扩展像素的取值范围，提高对比度和灰度色调的变化，使图像更加清晰。直方图均衡化一来可以提高图像的对比度，二来可以把图像变换成像素值是几乎均匀分布的图像。

2.数字图像直方图均衡化变换公式：

其中：

灰度值变换函数

*L* 灰度级

归一化直方图

第k级灰度值（k = 0,1,2...,L-1）

变换过后的灰度值

根据上述公式我们可以总结直方图均衡化的步骤：

•1.统计各灰度级像素数目

•2.计算灰度概率分布（直方图）

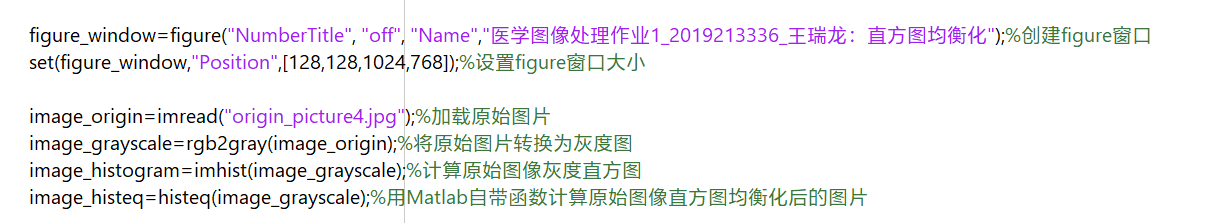
•3.计算累积概率分布（直方图）

•4.确定灰度映射关系（需要取整）

•5.灰度变换得到结果

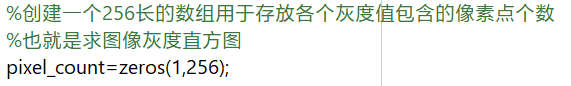
实验步骤：

1.读取图像

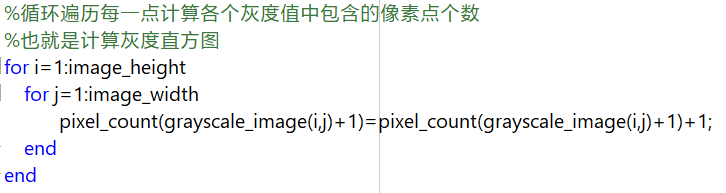


使用figure函数创建一个窗口，后续结果显示全在这个窗口中，方便进行对比。使用imread函数读取图像，并利用rgb2gray将图像从彩色图片转换为灰度图，并同时利用Matlab自带函数imhist和histeq函数得到原始图片的灰度图和直方图均衡化后的图像。

2.统计各个灰度级像素点数，计算灰度直方图

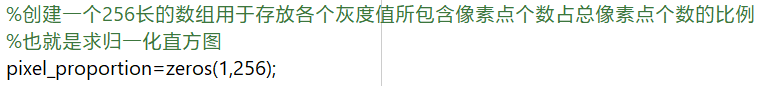


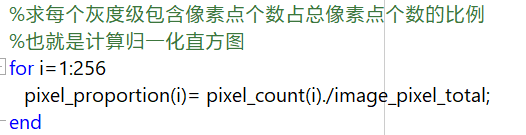
这里使用了zeros函数创建了一个一维数组，长度为256，也就是对应256个灰度级。



使用for循环一行一行遍历整个图像，得到灰度直方图，grayscale\_image(i,j)+1的原因是Matlab的数组索引是正整数从1开始。

3.求归一化直方图，得到灰度分布的概率密度

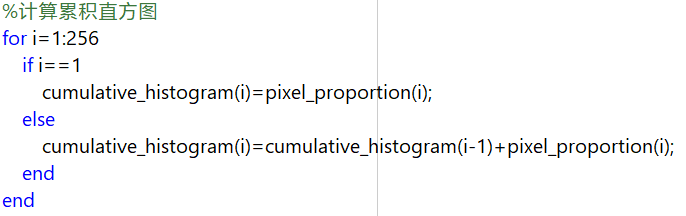




与之前相同，使用了zeros函数创建了一个一维数组，长度为256，并循环256次将上一步的直方图归一化，也就是得到灰度的概率密度函数，计算归一化直方图。

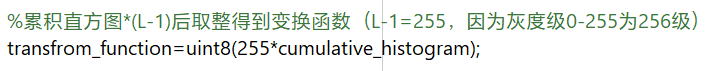
4．计算累积直方图





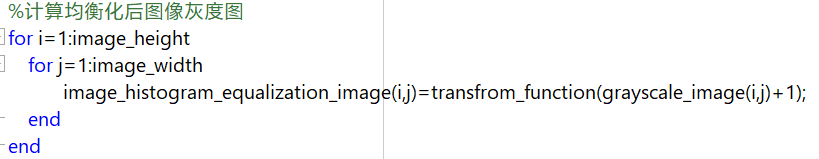
由数字图像直方图均衡化变换公式：我们可以知道，要得到变换关系式，我们需要对概率密度函数进行积分（求和），也就是要得到概率分布函数，由于是直方图是离散的，所以我们还是使用循环进行求和的方式，i等于1时cumulative\_histogram(1)只需要计算本身，i不等于1时cumulative\_histogram(i)= cumulative\_histogram(i-1)+pixel\_proportion(i),也就是上一个位置i-1累积直方图的值加上当前的灰度级的值，就可以只使用一个循环得到累积直方图。

5.对累积直方图进行取整，得到进行直方图均衡化的函数变换关系式



由于求出的 累积直方图\*255 得到的变换公式映射到的新的灰度值可能不是整数，所以需要取整，这里使用uint8显示转换为8位无符号整型，也就是向下取整，得到取整后的变换公式，映射到的新的灰度值都是整数。

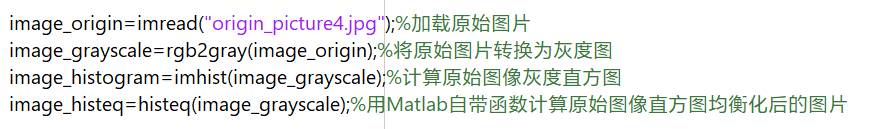
6.对原图像灰度图利用上一步得到的函数变换关系式进行变换，得到直方图均衡化后的图像

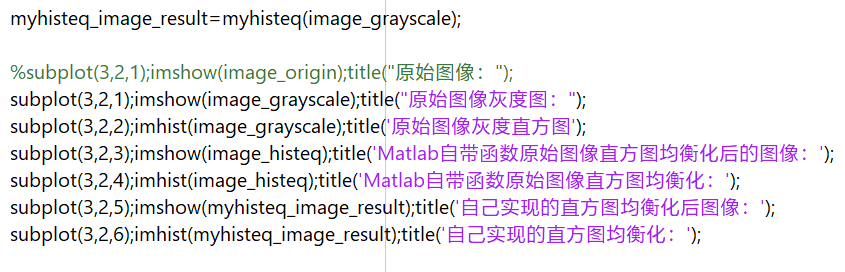


循环遍历整个图像将原始图像像素点的灰度值通过得到的变换函数转换为新的灰度值，实现灰度图像的直方图均衡化

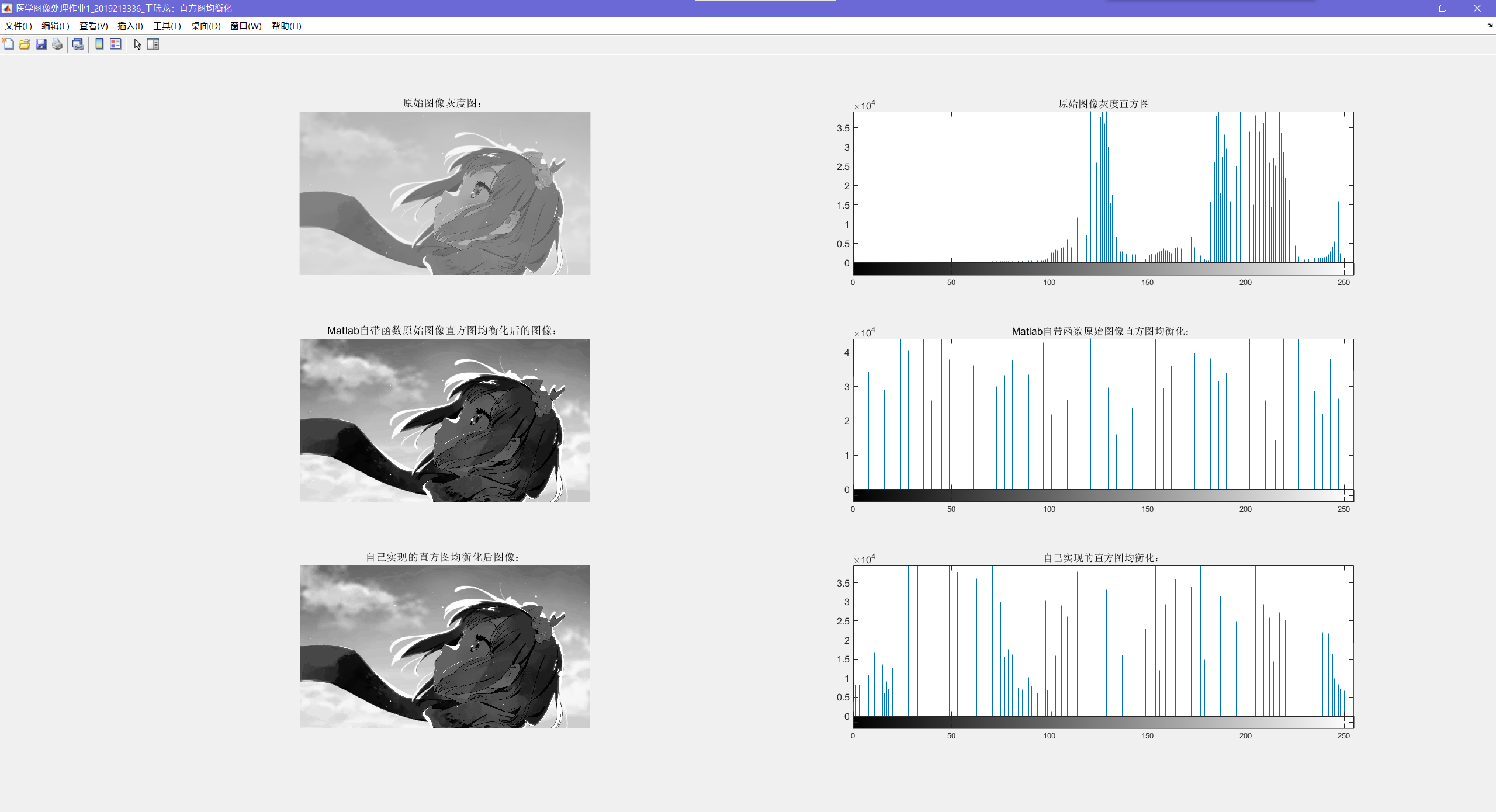
image\_histogram\_equalization\_image作为函数返回值返回。

7.显示结果





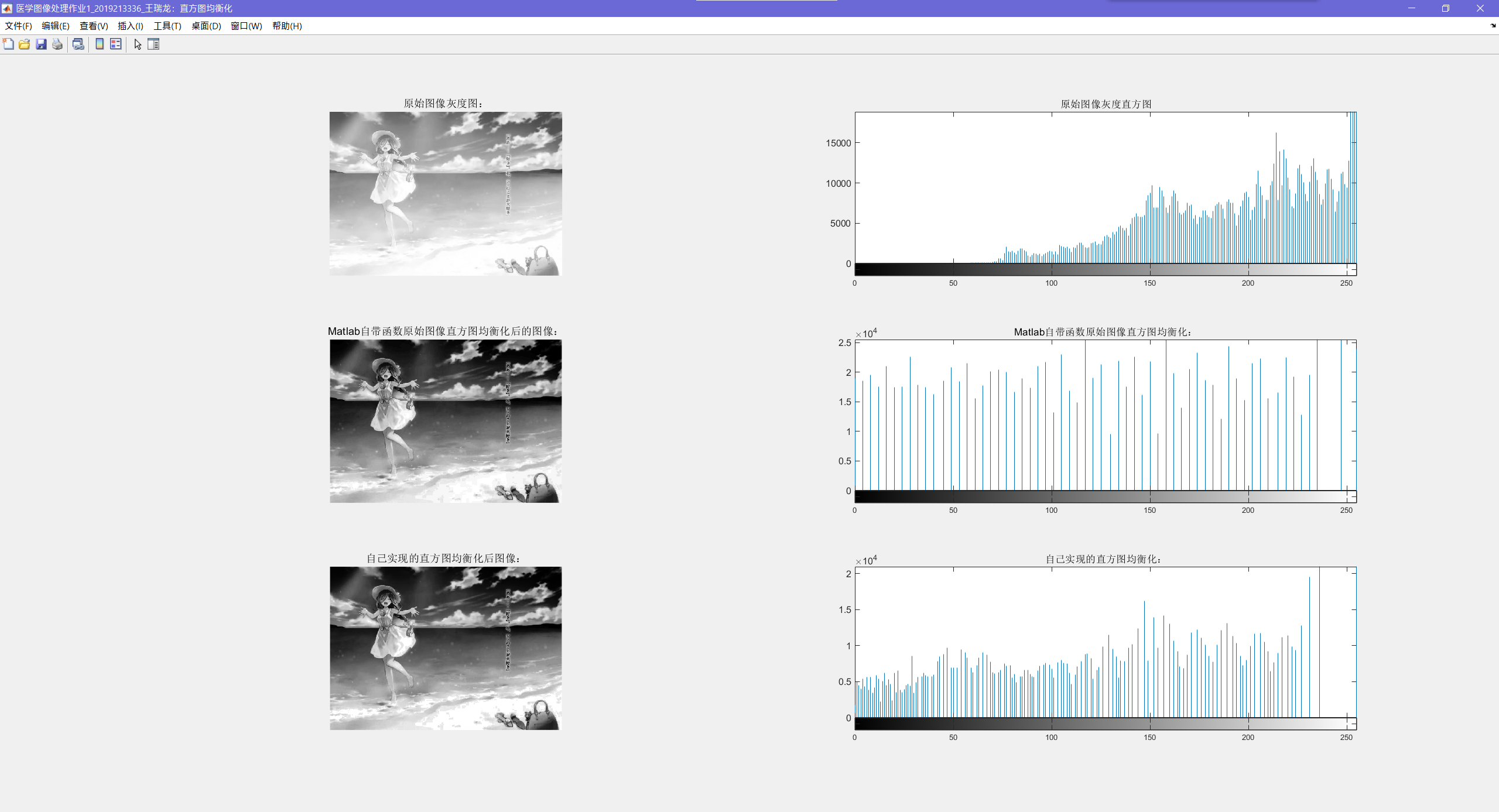
调用自己实现的直方图均衡化函数myhisteq，myhisteq\_image\_result为直方图均衡化后的图片结果，并将原始图片、Matlab自带函数均衡化后的图片、自己实现的直方图均衡化后的图片的灰度图和直方图进行显示，对比结果。

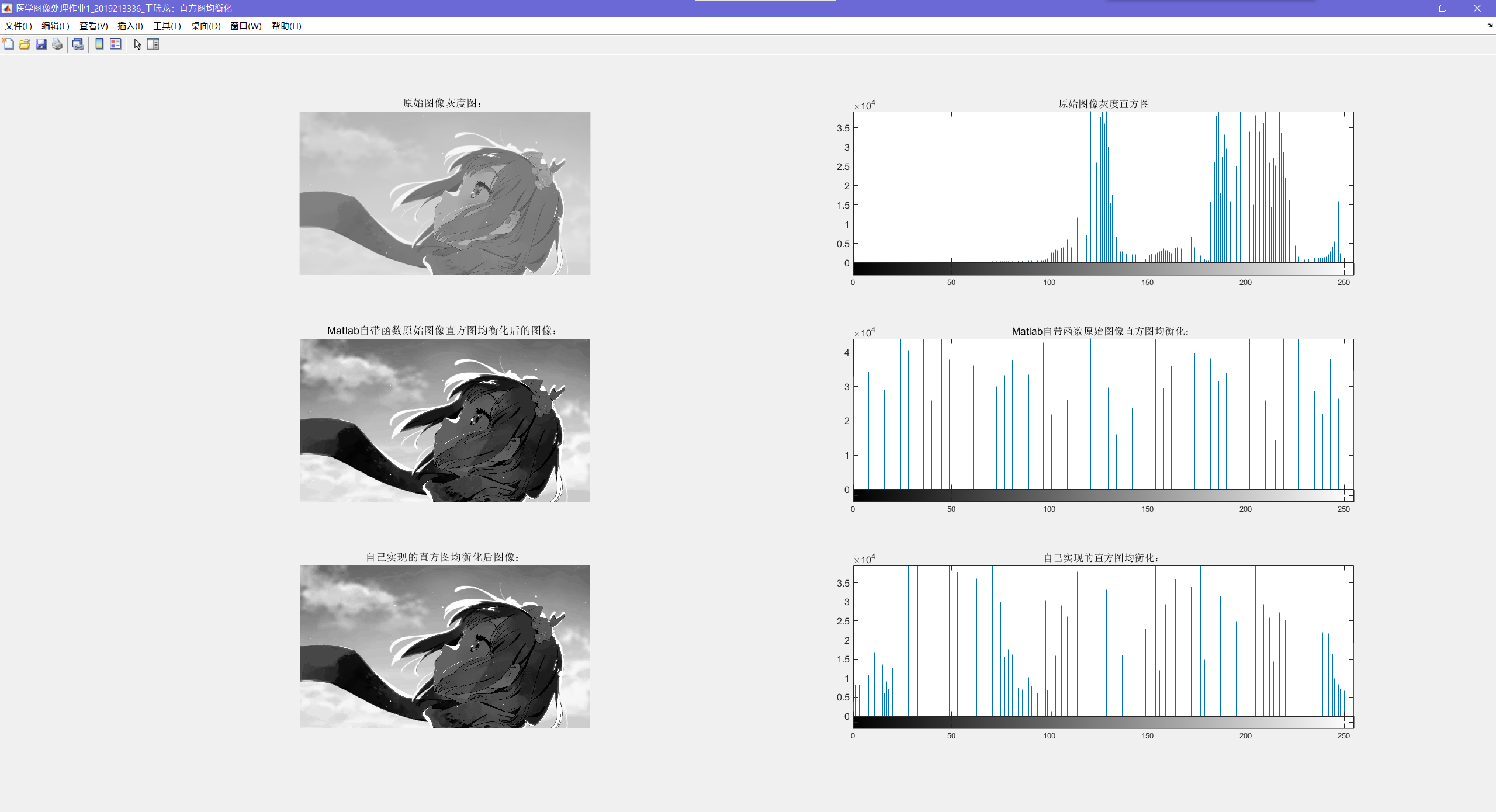


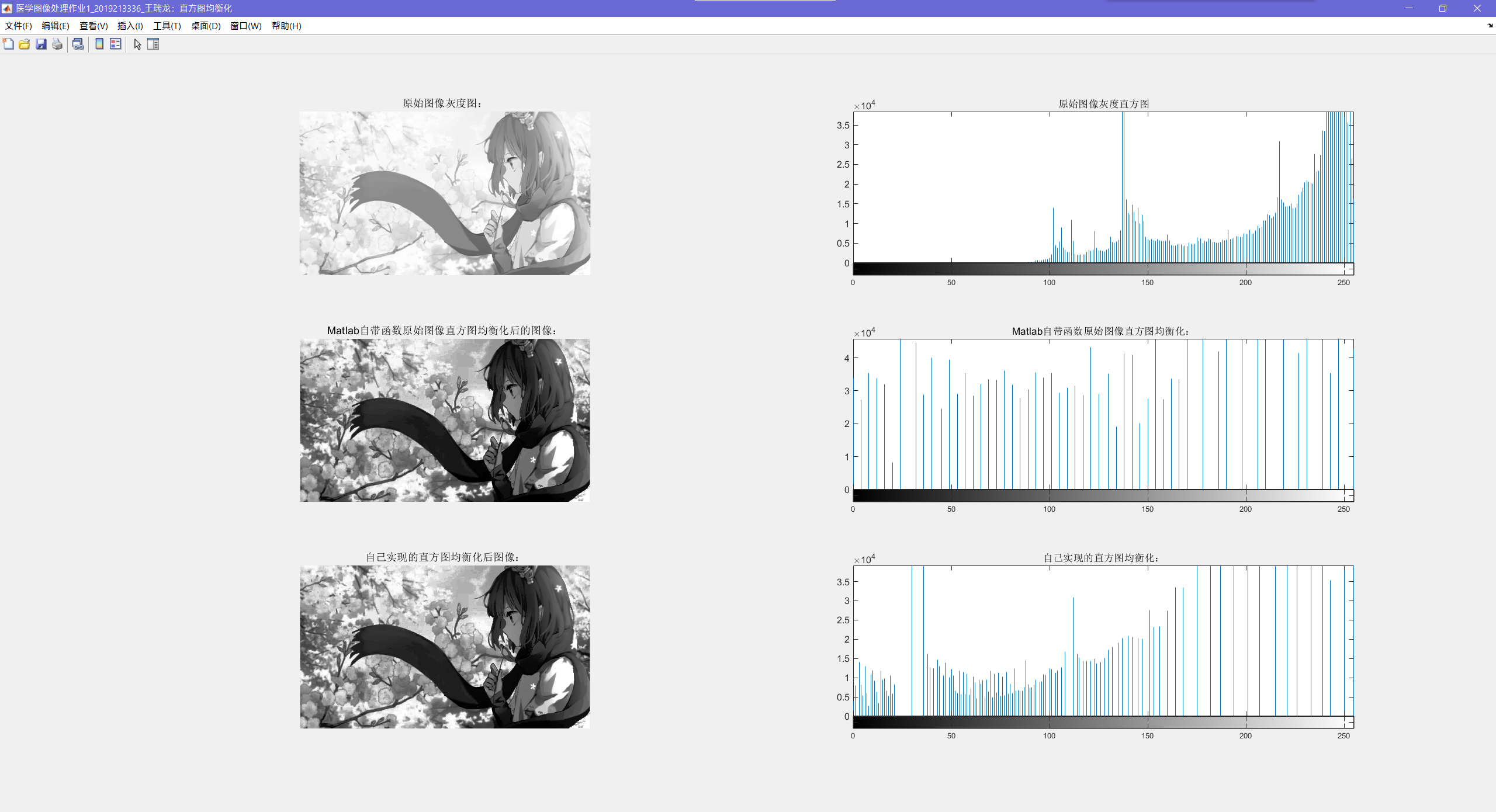
**三、实验结果与分析**

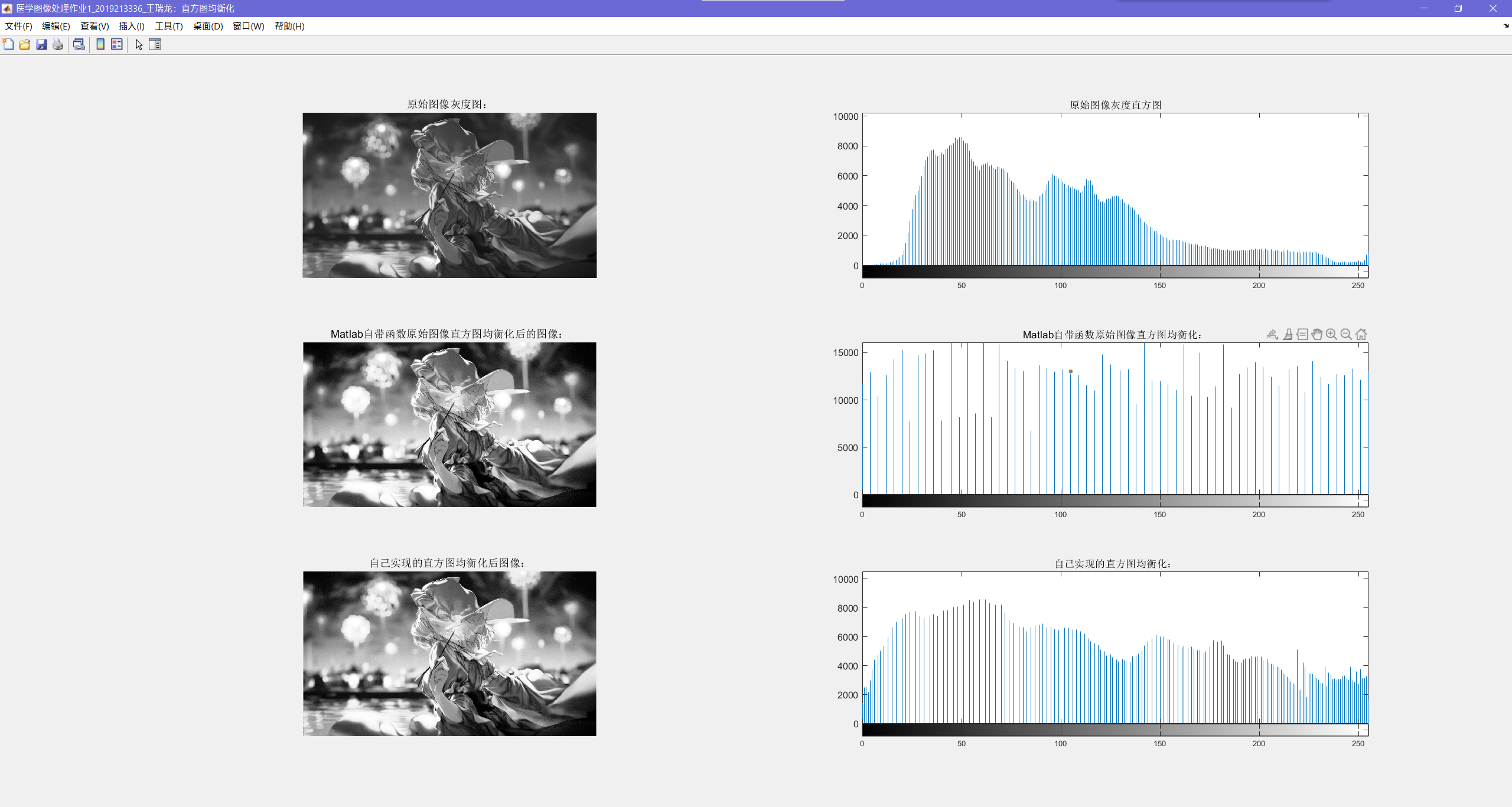
详细列出实验结果，并进行仔细分析，得出相关结论

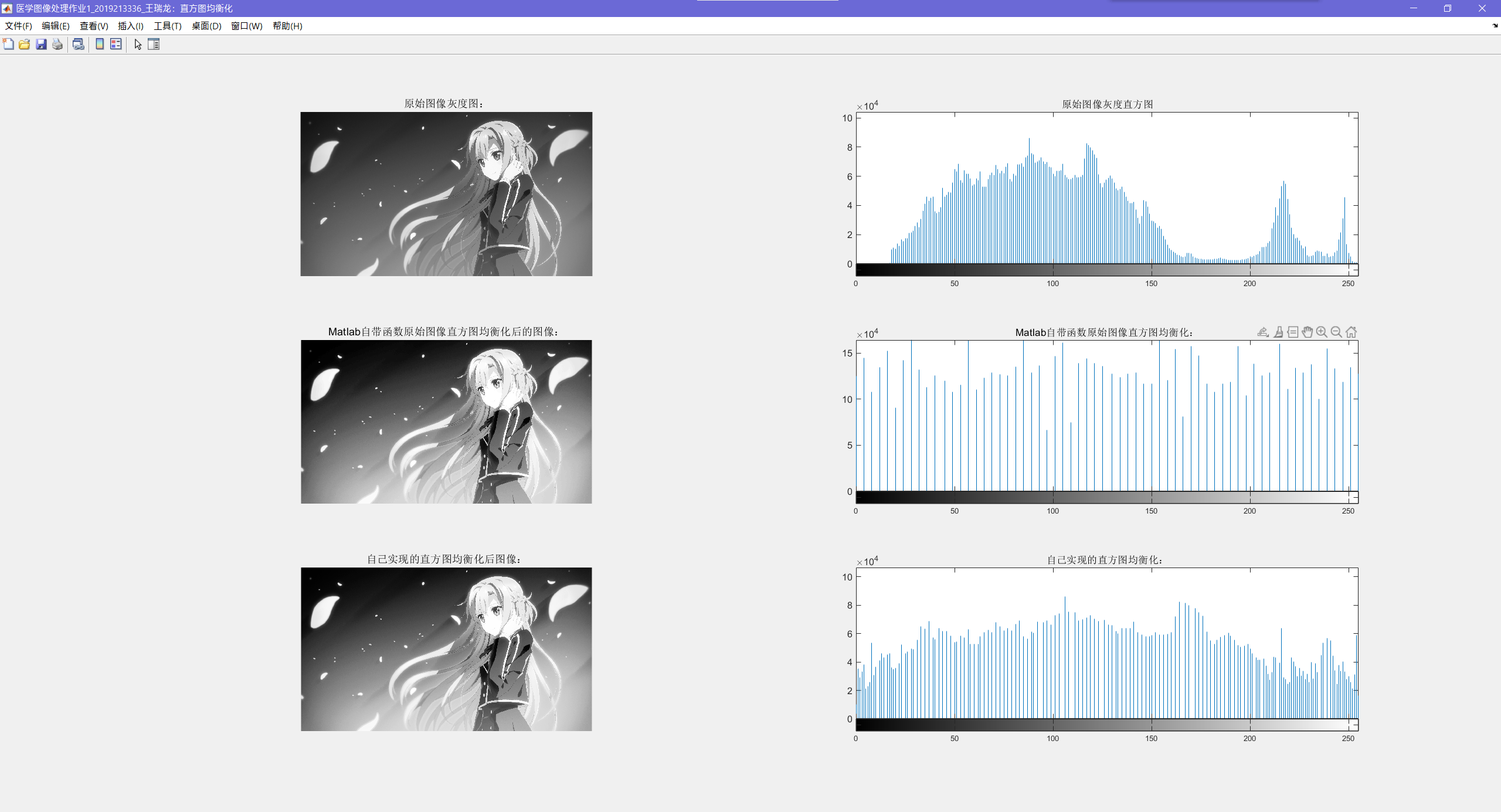
这里我选取了几张图片，来观察结果：









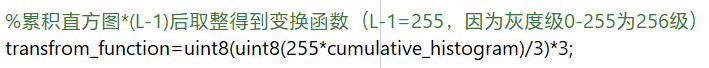


上述几副图片有偏亮的也有偏暗的的图片，直方图分布很不均匀，但经过直方图均衡化后的图片视觉效果得到明显改善，直方图分布比较均匀，对比度得到提升。

从上述几副图片的结果可以看出，自己实现的直方图均衡化灰度图和Matlab自带函数histeq处理后的灰度图视觉观感基本一致，不过从直方图来看，自己实现的直方图均衡化的直方图和Matlab自带函数histeq处理后的直方图还是有很明显去区别，表现在Matlab自带函数histeq处理后的直方图均衡化效果更好。

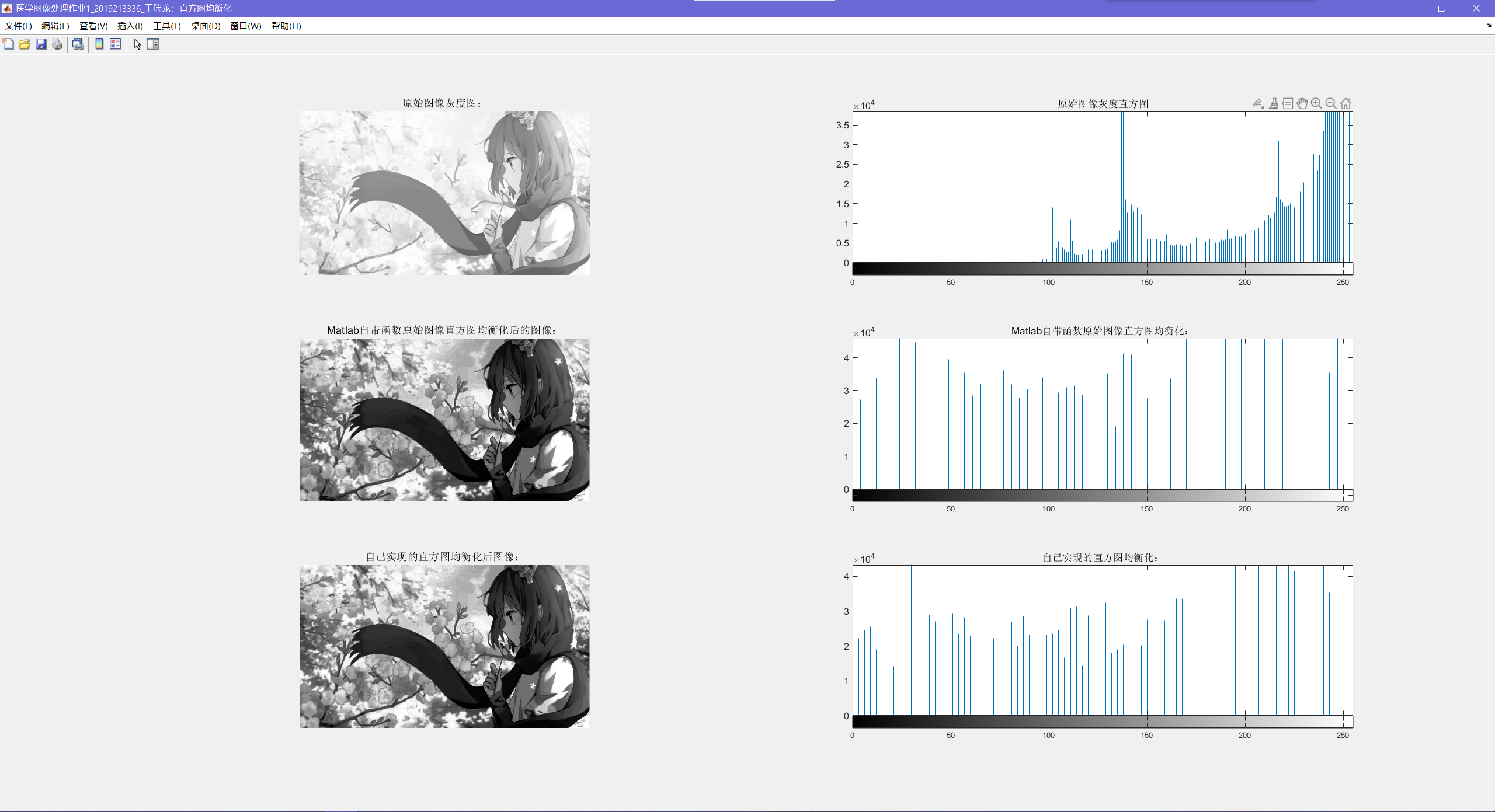
仔细观察可以发现，Matlab自带函数histeq处理后的直方图比较稀疏，而自己实现的比较密集，可以猜想，如果将相邻几个灰度级映射到一个灰度级，是不是可以实现比较稀疏的效果，直方图均衡化效果更好呢？

比如：数组{3，4，5，6，7，8，8，9，10} 我们可以先除一个数，假设为3，再向下取整，数组变为{1，1，1，2，2，2，3，3}，再乘一个数，假设为3，数组变为{3,3,3,6,6,6,9,9},可以看到，相邻3个数全都映射为了一个数，并且变的稀疏了，那么我们可以根据这个去改进我们的算法，



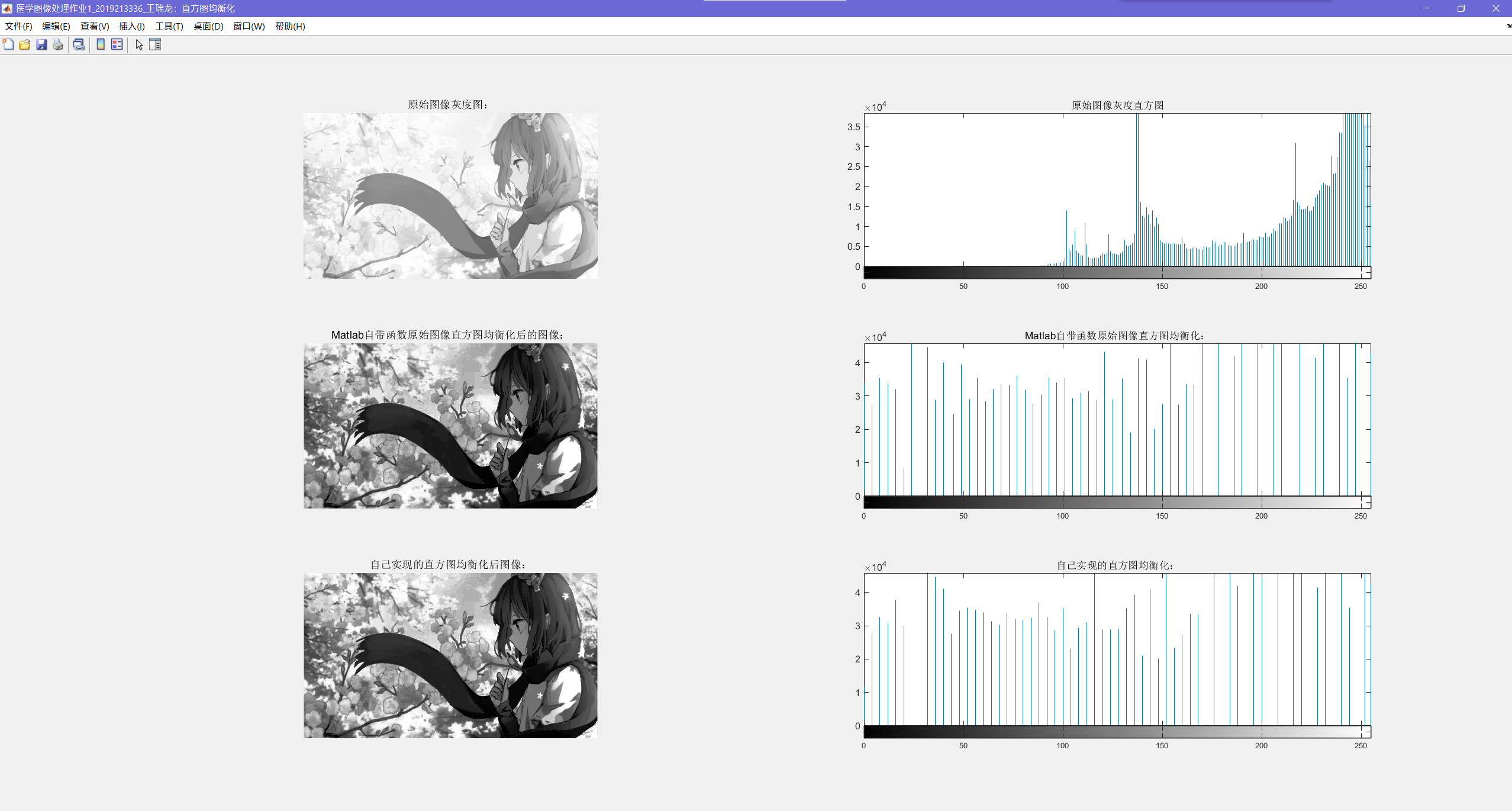
在求变换公式的时候我们对变换后的结果先乘以一个数，再除以一个数，这里我们假设取3，

结果如下：



可以看到，有非常明显的效果，与之前自己实现的直方图均衡化的效果相比，均衡化效果有了明显的提升，与Matlab自带函数histeq处理后的直方图相比，两者效果接近了不少。

我们可以继续增大，将3改为4，结果如下：



已经与Matlab自带函数histeq处理后的直方图相比已经非常接近了，可以发现这种做法可以改善直方图均衡化的效果。

结论：

1.偏亮或偏暗的以及直方图分布很不均匀的图片，经过直方图均衡化后的图片视觉效果得到明显改善，灰度直方图分布比较均匀，对比度得到提升。

2.自己实现的myhisteq可以实现直方图均衡化，但与Matlab自带函数histeq相比处理的结果存在差异，但通过相邻几个灰度级映射到同一个灰度级的做法可以明显改善直方图均衡化的效果，并且可以达到与Matlab自带函数histeq 大致相同的效果。