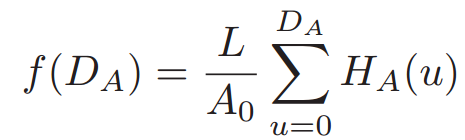
15331346 严晓珊

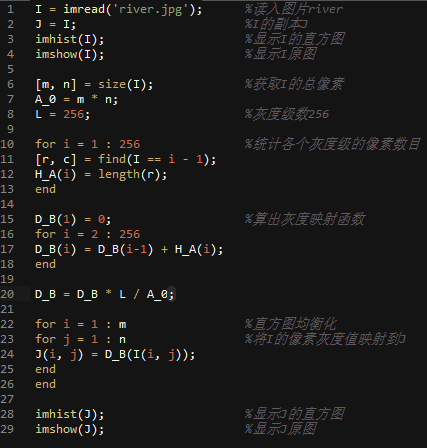
**直方图均衡**

1.算法描述：

**使用的是命令行**。直方图均衡化是通过增加局部的对比度而不影响整体的对比度，使得亮度可以更好地在直方图上分布。利用灰度级空间的总像素数目相同，将像素的灰度值进行映射。由于是在离散的情况下，所以H\_A(u)直接选择像素个数而不是频率。先读入原图river.jpg，在显示直方图和原图之后进行处理。首先通过size函数得出宽度m和高度n的像素，相乘得到总像素A\_0。然后开始统计不同灰度值的像素数目，用矩阵H\_A来记录。接着套用公式，获得矩阵D\_B。最后将I的副本J中的每一个像素的灰度值通过D\_B进行映射，获得新的图像J。依次显示J的直方图和原图。

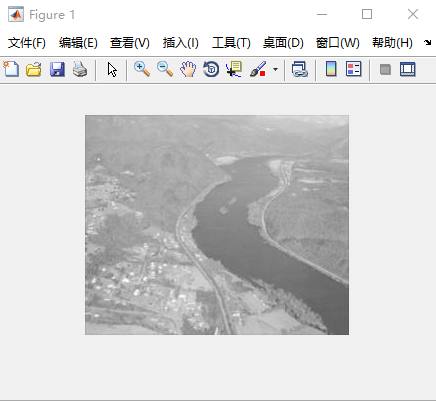
2.代码：都是命令行上面打的

|  |
| --- |
| I = imread('river.jpg'); %读入图片river  J = I; %I的副本J  imhist(I); %显示I的直方图  imshow(I); %显示I原图  [m, n] = size(I); %获取I的总像素  A\_0 = m \* n;  L = 256; %灰度级数256  for i = 1 : 256 %统计各个灰度级的像素数目  [r, c] = find(I == i - 1); %下标是D\_A+1  H\_A(i) = length(r);  end  D\_B(1) = H\_A(1); %算出灰度映射函数  for i = 2 : 256  D\_B(i) = D\_B(i-1) + H\_A(i);  end  D\_B = D\_B \* L / A\_0;  for i = 1 : m %直方图均衡化  for j = 1 : n %将I的像素灰度值映射到J  J(i, j) = D\_B(I(i, j)+1); %下标是D\_A+1  end  end  imhist(J); %显示J的直方图  imshow(J); %显示J原图 |

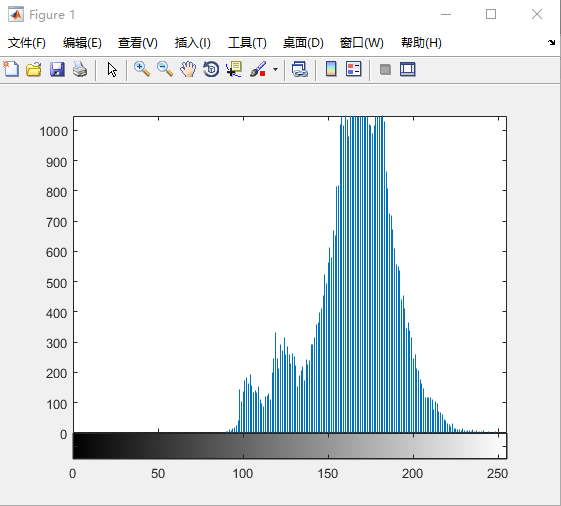


3.截图

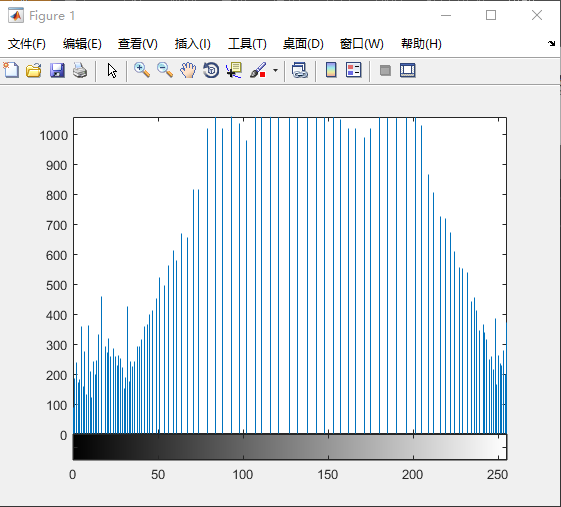
（1）I的原图：图片细节不清晰，局部对比度不强，亮度分布有问题



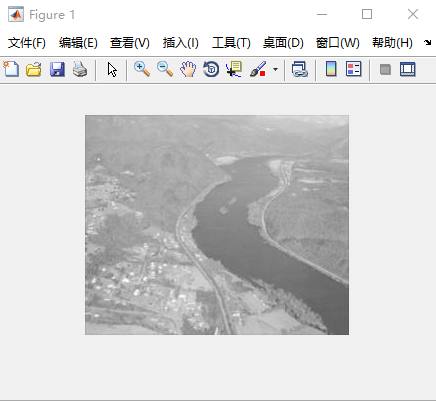
（2）I的直方图：在150-200区间比较集中，造成细节看不清楚



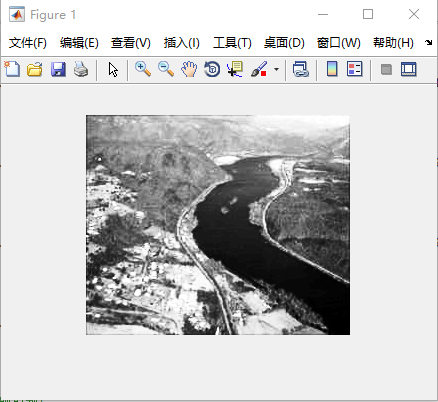
（3）J的直方图：均衡化后直方图变得分散，没有特别集中在某一小区间

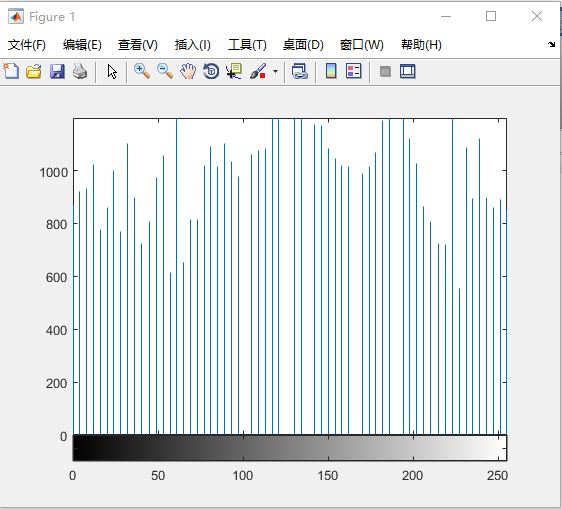
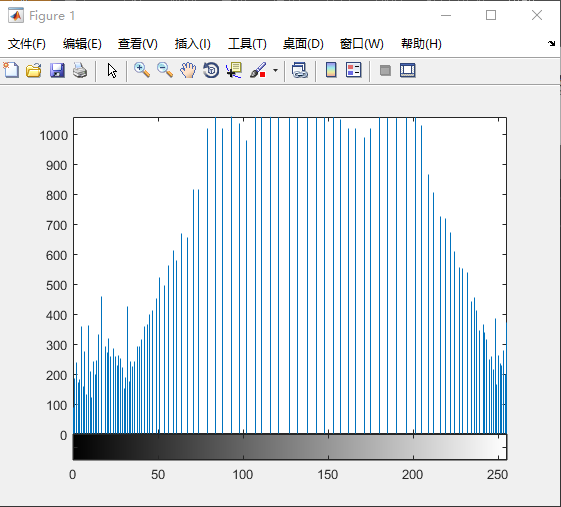


（4）J图原图：局部对比度增强，亮度分布更均匀，图片过渡细节更清楚

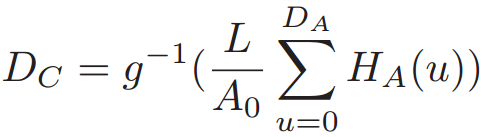
（5）matlab自带的直方图均衡化函数histeq处理后的效果：就图片效果来看和（4左）差不多，不过看起来颜色过渡上比（4左）好一点。但是直方图效果来看就差不少。histeq的直方图分布会均匀很多，而（3右）中50之前和250以后的部分还是比较少像素的。

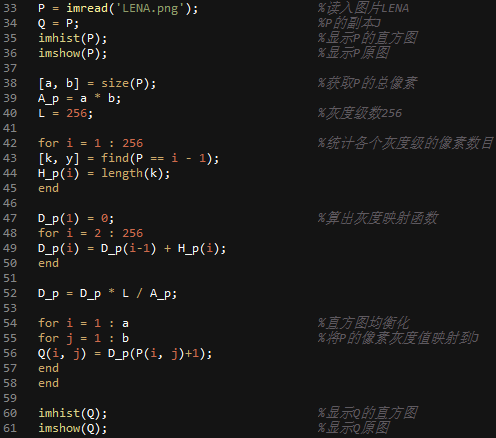
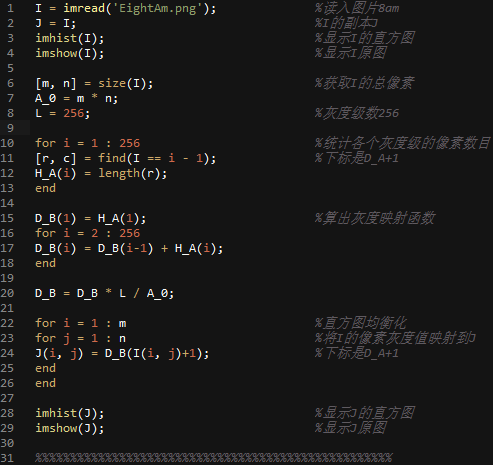
**直方图匹配**

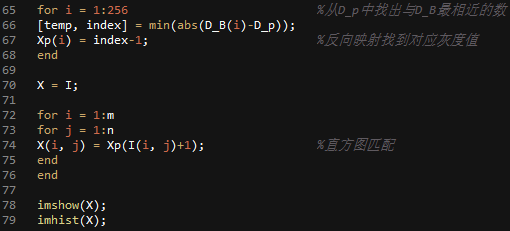
1.算法描述：

分别将EightAM.png和LENA.png两个图进行直方图均衡，利用上述算法，得到J图和Q图。利用灰度值映射反函数，通过近似EightAM和LENA的直方图均衡后的D\_B，从而获得从EightAM到LENA中像素的灰度值的映射关系。因为EightAM和LENA的总像素值还有直方图都不一样，直方图均衡出来的结果D\_B和D\_p也是不一样的，不能直接在D\_p中找到与D\_B相等的数，然后选下标。所以只能取与D\_B最相近的数，当做是相等，然后再反向映射找到相应的下标index。因为下标是从1开始的，而灰度值是从0开始的，所以要用到index-1，并将index-1存进数组Xp里面。而在最后直方图匹配的时候，在Xp里面选映射值时就要重新在下标位置加1，因为下标必须是正整数。X图即EightAM利用LENA直方图匹配后的图像。

2.代码：都是命令行上面打的

|  |
| --- |
| I = imread('EightAm.png'); %读入图片8am  J = I; %I的副本J  imhist(I); %显示I的直方图  imshow(I); %显示I原图  [m, n] = size(I); %获取I的总像素  A\_0 = m \* n;  L = 256; %灰度级数256  for i = 1 : 256 %统计各个灰度级的像素数目  [r, c] = find(I == i - 1); %下标是D\_A+1  H\_A(i) = length(r);  end  D\_B(1) = H\_A(1); %算出灰度映射函数  for i = 2 : 256  D\_B(i) = D\_B(i-1) + H\_A(i);  end  D\_B = D\_B \* L / A\_0;  for i = 1 : m %直方图均衡化  for j = 1 : n %将I的像素灰度值映射到J  J(i, j) = D\_B(I(i, j)+1); %下标是D\_A+1  end  end  imhist(J); %显示J的直方图  imshow(J); %显示J原图  P = imread('LENA.png'); %读入图片LENA  Q = P; %P的副本J  imhist(P); %显示P的直方图  imshow(P); %显示P原图  [a, b] = size(P); %获取P的总像素  A\_p = a \* b;  L = 256; %灰度级数256  for i = 1 : 256 %统计各个灰度级的像素数目  [k, y] = find(P == i - 1);  H\_p(i) = length(k);  end  D\_p(1) = 0; %算出灰度映射函数  for i = 2 : 256  D\_p(i) = D\_p(i-1) + H\_p(i);  end  D\_p = D\_p \* L / A\_p;  for i = 1 : a %直方图均衡化  for j = 1 : b %将P的像素灰度值映射到J  Q(i, j) = D\_p(P(i, j)+1);  end  end  imhist(Q); %显示Q的直方图  imshow(Q); %显示Q原图  for i = 1:256 %从D\_p中找出与D\_B最相近的数  [temp, index] = min(abs(D\_B(i)-D\_p));  Xp(i) = index-1; %反向映射找到对应灰度值  end  X = I;  for i = 1:m  for j = 1:n  X(i, j) = Xp(I(i, j)+1); %直方图匹配  end  end  imshow(X);  imhist(X); |



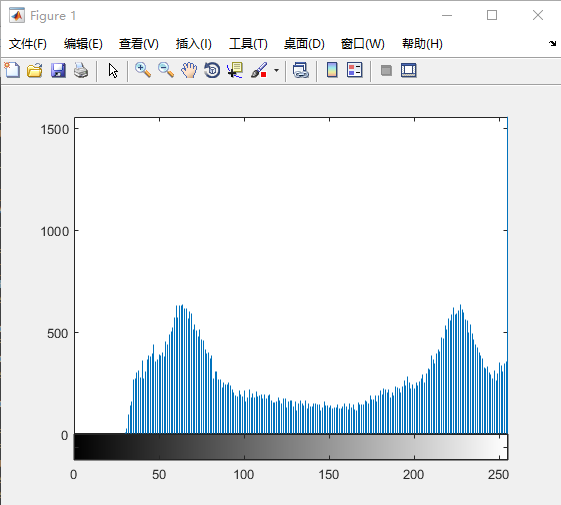


3.截图

（1）EightAM原图：大部分地方都太亮了，细节看不清楚



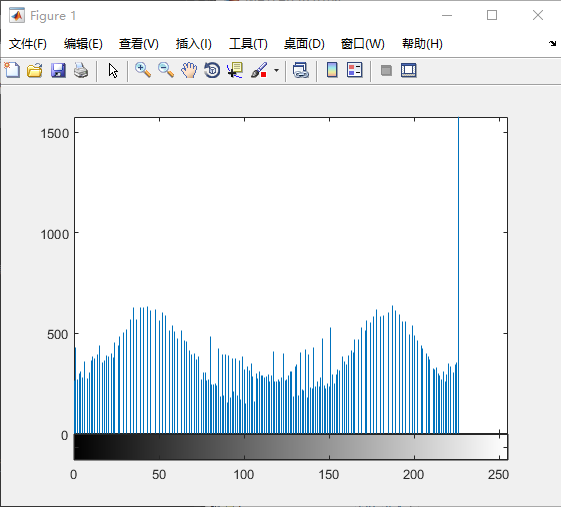
（2）EightAM直方图：低灰度值像素很少，细节丢失，75和225最多。



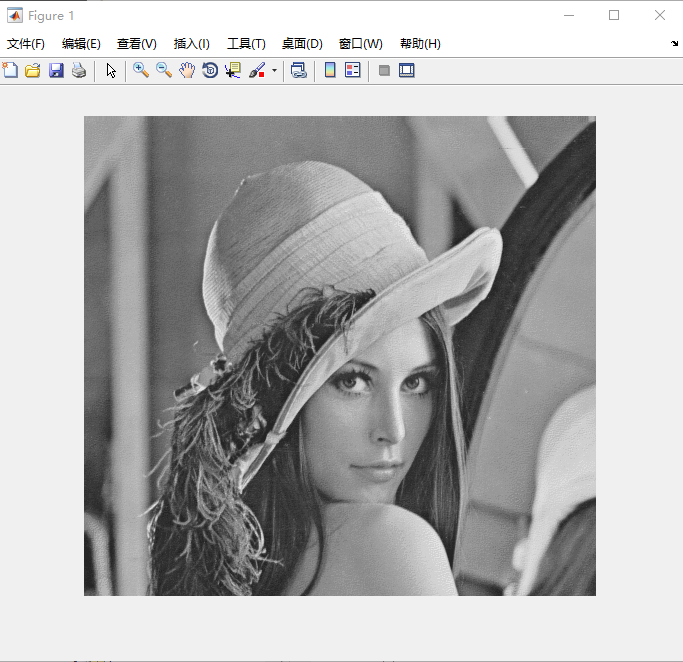
（3）EightAM直方图均衡后图：局部亮度进行了调节，细节显出来了



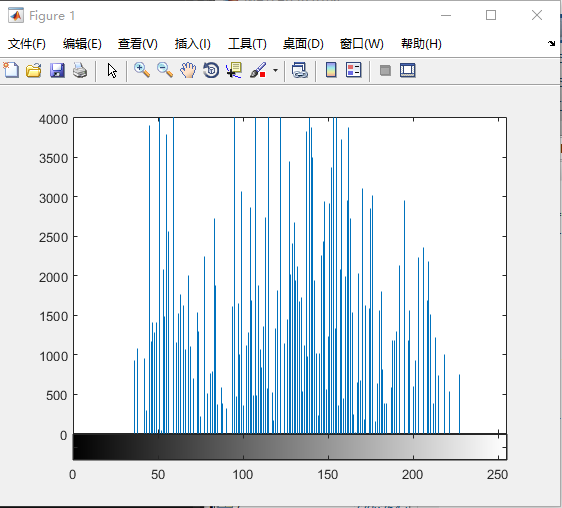
（4）EightAM直方图均衡后直方图：低灰度值有了像素的分布，总体均匀



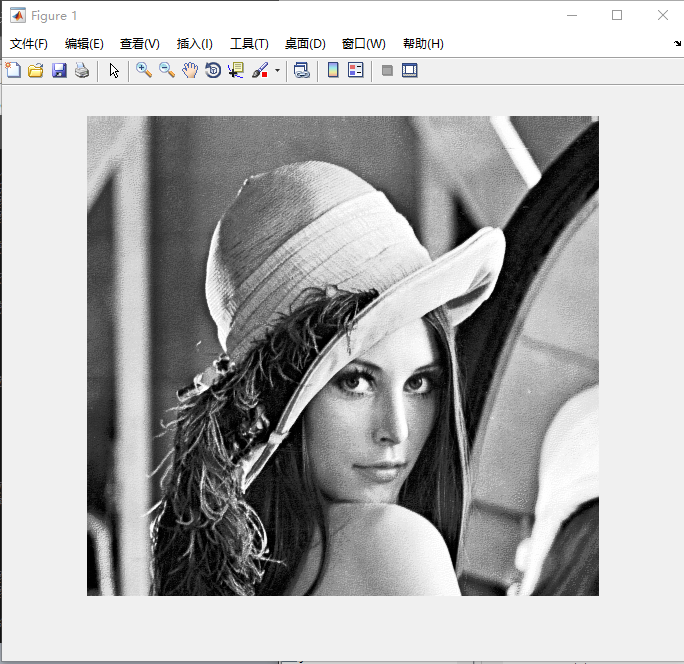
（5）LENA原图：细节不算很清楚，亮度有点暗沉，有点模糊



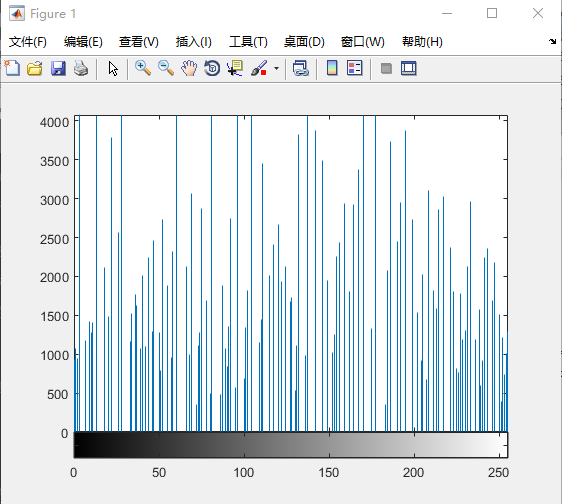
（6）LENA原图直方图：灰度值像素基本上是均匀的，除了某小部分区间



（7）LENA直方图均衡化图：比起原图，过渡细节更清楚了，亮度更柔和



（8）LENA直方图均衡化直方图：各个区间都有像素分布，细节更清楚



（9）EightAM直方图匹配图：图1为直方图匹配后的图，图2原图，图3参考图像。图2的背景是比较亮的，而图3的背景是比较暗的，图1的背景结合了图3的背景，变得比较暗。图1的衣服反光没有图2这么明显，继承了图3的暗沉。



（10）EightAM直方图匹配直方图：图1直方图匹配直方图，图2原图直方图，图3参考图像直方图。图1的直方图是图2和图3的融合版，50灰度值附近继承了图3，100-150继承了图3补充了一些像素，而200过后的灰度值作了舍弃。

