数字图像处理

pro03-02 直方图均衡

信息与电子工程学院　信息工程

2023 年 3 月 26 日

1　实验任务

(1) 编写计算图像直方图的程序；
(2) 实现第 3.3.1 节中讨论的直方图均衡技术；
(3) 下载图 3.8(a)，并对其进行直方图均衡；

2　算法设计

2.1　计算直方图

获取原始图像信息，并转化为灰度图像，获得图像的信息矩阵。

构造长为 256 的零向量，用双重循环遍历图像每一点像素值，以像素值为零向量下标进行累加，即可得到灰度值范围为 0-255 的点数统计。

2.2　直方图均衡

有两种方法可以实现。

第一种是，根据 $s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \quad k = 0, 1, 2, \ldots, L-1$ 对应的函数映射，改变原始图像每一点对应的像素值，在 matlab 中可以直接利用 histeq 函数实现。

第二种是，计算得到灰度的概率分布积累函数 $cdf_x(i) = \sum_{j=0}^{i} p_x(j)$，对积累分布函数进行均衡化 $cdf_y(i) = T[cdf_x(i)]$，T(x) 是一个转化函数，定义 T 为 $cdf_y(i) = \text{round}\left[\frac{cdf(v)-cdf_{\min}}{M*N-cdf_{\min}} \times (\text{ Graylevel } -2)\right] + 1$，再利用新的 cdf 值对原始图像进行归一化 $\text{Image}_{\text{equal}}(i,j) = cdf_y(cdf_x(Image(i,j)))$，与原始方法相比更改了映射函数。

3　代码实现

可以用两种代码方式实现，采用 T(x) 做为转化函数，代码如下：

```
clear;clc;close all;
pic_ori = imread('spine.tif'); % 读取原始图像
size = size(pic_ori);
% 若图像是rbg的，则转化为灰度图
if( numel(size) > 2 )
  pic_ori = rgb2gray(pic_ori);
  size = size(pic_ori);
end
height = size(1);
width = size(2);
gray_level = 256;
```

```
% 获取灰度值频数分布
P = zeros(gray_level,1);
for i = 1:height
  for j = 1:width
    gray_value = pic_ori(i,j)+1;
    P(gray_value) = P(gray_value) + 1;
  end
end

% 获得灰度值累积分布
cdf = zeros(gray_level,1);
cdf(1) = P(1);
cdf_min = 0;
for i = 2:gray_level
  cdf(i) = cdf(i-1) + P(i);
  if(cdf_min == 0 && cdf(i) > 0)
    cdf_min = cdf(i);
  end
end

% 对灰度值累积分布进行转化
cdf_equal = zeros(gray_level, 1);
for i = 1:gray_level
  cdf_equal(i) = round( (cdf(i)-cdf_min) / (height * width - cdf_min) * (gray_level - 1) ) + 1;
end

% 计算图像像素点新的灰度值
pic_equal = pic_ori;
for i = 1:height
  for j = 1:width
    pic_equal(i,j) = cdf_equal( pic_equal(i,j) + 1 );
  end
end

% 获取均衡后的灰度值频数分布
P_equal = zeros(gray_level,1);
for i = 1:height
  for j = 1:width
    gray_value = pic_equal(i,j)+1;
    P(gray_value) = P(gray_value) + 1;
  end
end
figure(1);
subplot(121);imshow(pic_ori);title('原图')
subplot(122);imshow(pic_equal);title('均衡化后');
figure(2);
subplot(121);imhist(pic_ori);title('原图像直方图');
subplot(122);imhist(pic_equal);title('均衡化后直方图');
```

直接使用函数 histeq，代码如下：

```
clear;clc;close all;
% 读取图像
pic_ori=imread('spine.tif');
% 直方图均衡
pic_hist = histeq(image);
figure(1);
subplot(1,2,1);imshow(pic_ori);title('原图');
subplot(1,2,2);imshow(pic_hist);title('均衡化后');
figure(2);
subplot(1,2,1);imhist(pic_ori);title('原图像直方图');
```

```
subplot(1,2,2);imhist(pic_hist);title('均衡化后直方图');
```

4 实验结果

导入图 3.8(a)，使用 T(x) 作为转化函数，得到的原图像和直方图均衡后图像以及它们的直方图如下。从图像看，均衡后图像细节更明显，对比度更高。从直方图看，减少了灰度值低的像素，总体上更加平均，趋势与原直方图类似，依次递减。

图 1: code1: 原图与直方图均衡后图像

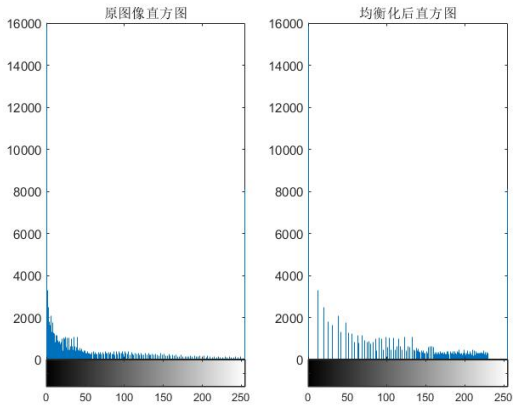

图 2: code1: 原图与直方图均衡后直方图

导入图 3.8(a)，使用 histeq 函数，得到的原图像和直方图均衡后图像以及它们的直方图如下。从图像看，均衡化后图像细节更加明显，整体颜色更加明亮，但在黑色背景部分出现了一些噪点。从直方图看，直方图均衡化后总体上更加平均，但因为原直方图绝大部分像素值集中在 0 附近，所以直方图均衡后不能覆盖到像素值低的部分，高的部分可以正常近似平均。

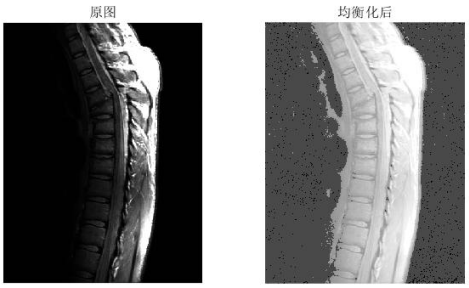

图 3: code2: 原图与直方图均衡后图像

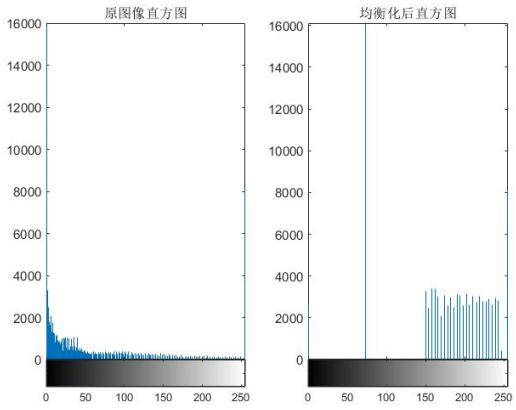

图 4: code2: 原图与直方图均衡后直方图

5 总结

本次实验的关键在于直方图的计算与直方图均衡的算法，直方图计算比较简单，只需要遍历图像的所有像素；直方图均衡也可以直接采用 matlab 的函数进行处理，但是最终结果并不理想。原始方法其实对这种像素值主要集中在 0 附近的直方图处理并不理想。之后参考了网上的另一算法，改变了映射关系，可以看到比较好地削减了某一像素值过高的情况。通过这次实验，我熟悉了数字图像处理操作的基础代码，因为是第一次接触图像处理，学习到了很多基本的 matlab 数字图像处理的函数。