**物联网第四章大作业实验报告**

富诗涵2019213493

目录

[物联网第四章大作业实验报告 1](#_Toc5618)

[1.数据源准备 1](#_Toc6070)

[2.实验环境 1](#_Toc10078)

[3.实验设计 2](#_Toc13117)

[4.实验过程 2](#_Toc12809)

[4.1对图像进行直方图均衡化操作 2](#_Toc5532)

[4.2自定义阈值将图像二值化处理 4](#_Toc13677)

[4.3对图像添加噪声后再进行平滑处理 5](#_Toc30479)

[4.4边缘检测 7](#_Toc3284)

[5.实验总结 9](#_Toc16818)

[6.代码备注 10](#_Toc28164)

[7.参考文献 10](#_Toc16384)

# 1.数据源准备

使用给定数据，Lena.bmp



# 2.实验环境

|  | Recommended |
| --- | --- |
| operating system | windows 10 |
| processor | Intel(R) Core(TM) i7-9750H |
| RAM | 16GB |
| storage | 32GB |
| running environment | Matlab R2019a |

# 3.实验设计

实验目的：掌握图像表示相关的基础知识，理解与图像相关的常见属性与参数信息，掌握常见图像噪声添加、图像平滑处理、图像边缘检测、常见的图像处理过程方法。

实验设计：使用matlab，完成对Lena.bmp的一系列操作。通过对比程序输出结果，分析理解图像相关的常见属性，以及不同噪声、不同算子、不同平滑处理的区别。

# 4.实验过程

## 4.1对图像进行直方图均衡化操作

直方图均衡化是一种简单有效的图像增强技术，通过改变图像的直方图来改变图像中各像素的灰度，主要用于增强动态范围偏小的图像的对比度。原始图像由于其灰度分布可能集中在较窄的区间，造成图像不够清晰。

例如，过曝光图像的灰度级集中在高亮度范围内，而曝光不足将使图像灰度级集中在低亮度范围内。采用直方图均衡化，可以把原始图像的直方图变换为均匀分布（均衡）的形式，这样就增加了像素之间灰度值差别的动态范围，从而达到增强图像整体对比度的效果。

直方图均衡化流程[1]：

1.获取现有图形的灰度图像值

2.划分灰度等级fi，计算每个像素对应灰度等级出现的次数，绘制变化前的直方图

3.对直方图进行归一化处理，计算各个灰度等级出现的频率p=ni/n

4.计算累计分布函数c=fi\*p,四舍五入取整获取新的分布的灰度等级gi

5.用映射关系修改原始灰度等级fi=gi，获得输出图像

首先尝试直接调用库函数，histeq。通过对比输出图片以及直方图分布，发现图片的对比度被增强了，且图片的直方图分布变得更加均衡。图1为使用库函数histeq后的图片及结果对比。

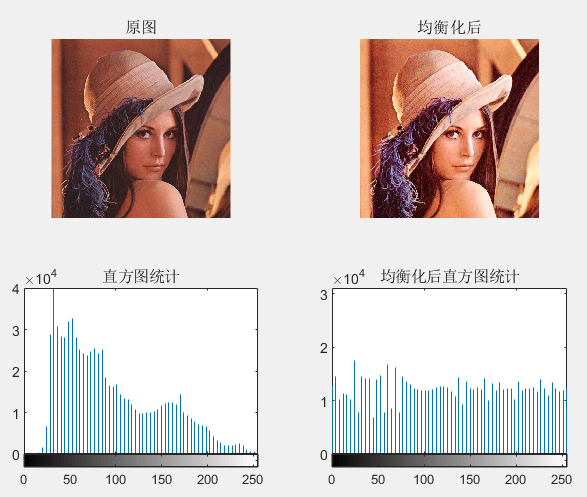


图 1 使用库函数histeq后的图片及结果对比

然后尝试自己编写histeq函数，具体过程如上文所述。图2为自己编写均衡函数后的图片及结果对比。

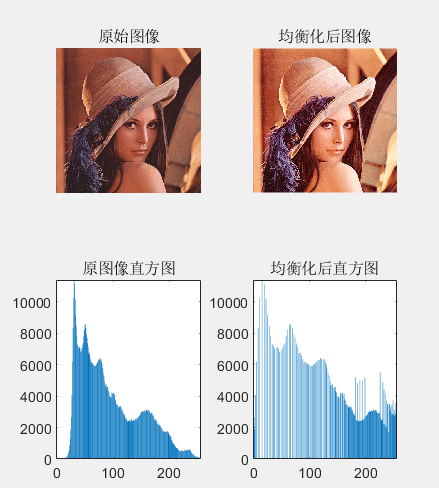


图 2 编写均衡函数后的图片及结果对比

其中，并没有直接取平均值，更像是直方图匹配。它们的相同点都是将原图像直方图按照一定的方式进行调整，但调整的方式不同，可以将直方图均衡化看作是直方图匹配在规定的直方图形态为均匀分布条件下的特例[2]。

## 4.2自定义阈值将图像二值化处理

在实验中发现，只有输入图片是灰的图时，才可以将其进行二值化处理，因此先将其转成灰度图、再使用自定义阈值进行图像二值化处理。

灰度化处理就是将一幅色彩图像转化为灰度图像的过程。彩色图像分为R，G，B三个分量，分别显示出红绿蓝等各种颜色，灰度化就是使彩色的R，G，B分量相等的过程。灰度值大的像素点比较亮（像素值最大为255，为白色），反之比较暗（像素最下为0，为黑色）。

图像灰度化的算法主要有以下3种[3]：

1.最大值法： R=G=B=max（R，G，B）(1)这种方法转换的灰度图亮度很高。

2.平均值法： R=G=B=(R+G+B)/3 (2)这种方法产生的灰度图像比较柔和。

3.加权平均值法，R=G=B=(WRR+WGG+WBB)/3 （3）由于人眼对绿色最为敏感，红色次之，对蓝色的敏感性最低，因此使将得到较易识别的灰度图像。一般时，得到的灰度图像效果最好。

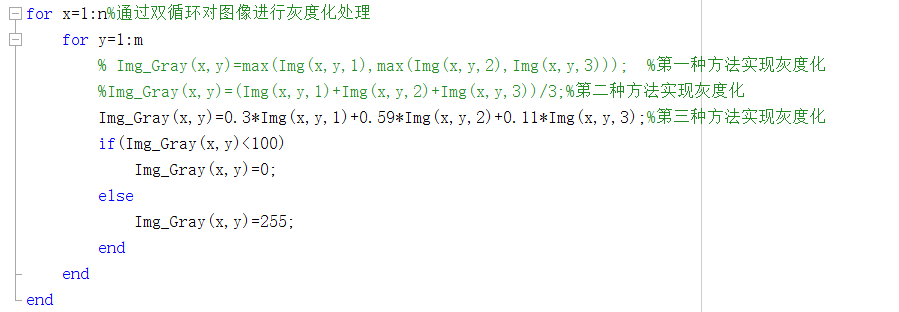


图 3 具体自定义实现灰度化的三种不同方法代码以及自定义二值化代码

表 1 不同方法得到灰度图结果比较

|  |  |  |  |
| --- | --- | --- | --- |
| 系统函数调用 | 最大值法 | 平均值法 | 加权平均值法 |
|  |  |  |  |

选择加权平均值法得到的灰度图作为后续对图像进行图像二值化处理的输入。

图像二值化有以下四种方法[4]：

1.简单粗暴判断法。对RGB彩色图像[灰度化](https://so.csdn.net/so/search?q=%E7%81%B0%E5%BA%A6%E5%8C%96&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/fu18946764506/article/details/_blank)以后，扫描图像的每个像素值，值小于127 的将像素值设为0(黑色)，值大于等于127的像素值设为255(白色)。该方法的好处是 计算量少速度快。缺点更多首先阈值为127没有任何理由可以解释，其次完全不考虑图 像的像素分布情况与像素值特征。

2.计算像素的平均值K。也是现在最常见的二值处理方法。扫描图像的每个像素值如像 素值大于K像素值设为255(白色)，值小于等于K像素值设为0(黑色)。使用平均值作 为二值化阈值的缺点：可能导致部分对象像素或者背景像素丢失，且二值化结果不能真 实反映源图像信息。

3.使用直方图方法来寻找二值化阈值，直方图是图像的重要特质，直方图方法选择二值

化阈值主要是发现图像的两个最高的峰，然后在阈值取值在两个峰之间的峰谷最低处。

本方法更加精准。

4.使用近似一维Means方法寻找二值化阈值[5]

其步骤如下：

1).获得一个初始化阈值T，可以自己设置或者根据随机方法生成。

2).根据阈值图每个像素数据P(n,m)分为对象像素数据G1与背景像素数据G2。(n为

行，m为列)

3).G1的平均值是m1, G2的平均值是m2

4).计算得到一个新的阈值T’= (m1 + m2)/2

5).回到第二步，用新的阈值继续分像素数据，重复循环步，直到计算出来的新阈值等 于上一次阈值。

实验中使用了第一种方法。

表 2 不同阈值的结果对比

|  |  |
| --- | --- |
| 阈值127 | 阈值100 |
|  |  |

通过对比结果发现，阈值越小，图片越”亮“，因为经过二值化后赋值为白色的点更多。

## 4.3对图像添加噪声后再进行平滑处理

噪声可以理解为灰度值的随机变化，即拍照过程中引入的一些不想要的像素点。噪声可分为椒盐噪声，高斯噪声，加性噪声和乘性噪声等[6]。本实验中选择的是高斯噪声，指服从高斯分布的一类噪声，通常是因为不良照明和高温引起的传感器噪声。通常在RGB图像中，显现比较明显。

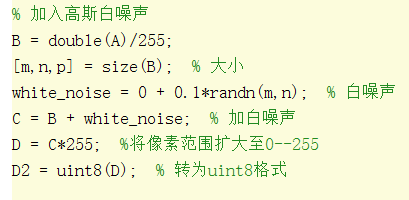


图 4 加入高斯白噪声代码



图 5 加入高斯噪声后的图像

在图像平滑处理方面，使用中值滤波。其原理为：将掩膜内所有像素点值进行排序，然后用中位数（中值）来作为当前像素点的值。中值滤波的效果一般最好，因为噪声点的像素值一般很大，所以噪声点像素值一般不会被选上，因此他不像均值处理那样会改变原始图像的像素值分布[7]。

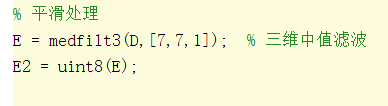


图 6 平滑处理



图 7 使用中值滤波进行平滑处理

## 4.4边缘检测

边缘检测的目的是检测识别出图像中亮度变化剧烈的像素点构成的集合。边缘检测可以减少源图像的数据量，剔除与目标不相干的信息，保留图像重要的结构属性。

图像的边缘指的是图像中像素灰度值突然发生变化的区域，如果将图像的每一行像素和每一列像素都描述成一个关于灰度值的函数，那么图像的边缘对应在灰度值函数中是函数值突然变大的区域。函数值的变化趋势可以用函数的导数描述。当函数值突然变大时，导数也必然会变大，而函数值变化较为平缓区域，导数值也比较小，因此可以通过寻找导数值较大的区域去寻找函数中突然变化的区域，进而确定图像中的边缘位置。

边缘检测分为以下三个类型[8]：

1.一阶微分为基础的边缘检测，通过计算图像的梯度值来检测图像的边缘，如Sobel算子、Prewitt算子、Roberts算子及差分边缘检测。

2.二阶微分为基础的边缘检测，通过寻求二阶导数中的过零点来检测边缘，如拉普拉斯算子、高普拉普拉斯算子、Canny算子边缘检测。

3.混合一阶与二阶微分为基础的边缘检测，综合利用一阶微分与二阶微分特征，如Marr-Hildreth边缘检测算子。

本实验中选择一阶微分为基础的边缘检测。

表 3 不同算子比较 [9]

|  |  |  |
| --- | --- | --- |
| roberts算子 | sobel算子 | prewitt算子 |
|  |  |  |

首先使用系统定义的prewitt算子。

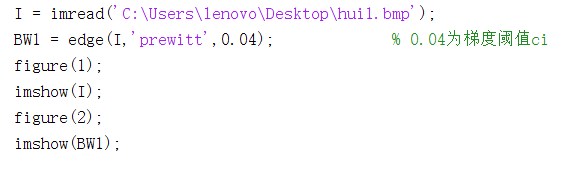


图 8 使用系统自带的prewitt算子



图 9 系统pretwitt结果

然后通过上述具体算子，编写自己的边缘检测函数。

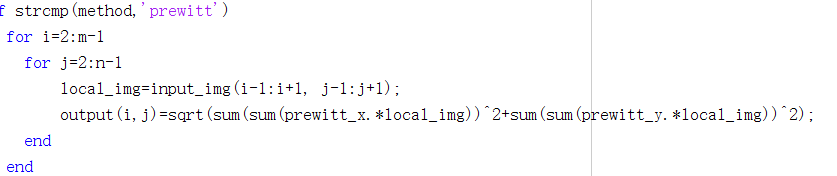


图 10 自定义prewitt函数

除此之外，将边缘检测与图像平滑相结合，使用高斯低通滤波平滑图像。因为在未进行此步处理时，图片细节过多，结果图片的检测出边缘过多，因为我们希望牺牲部分图像细节以获得更加舒适的边缘图片。



图 11 使用高斯滤波以及自定义算子prewitt结果

# 5.实验总结

在进行图像处理的过程中，学习了图像表示相关的基础知识，理解与图像相关的常见属性与参数信息，掌握了常见图像噪声添加、图像平滑处理、图像边缘检测、常见的图像处理过程方法。并尝试手动编写直方图均值处理、自定义阈值二值化以及边缘检测的代码，掌握了matlab在图像处理方面的使用以及见识到其功能库的强大。未来在图像处理方面，我们还有很长的路要走。

# 6.代码备注

6.1对图像进行直方图均衡化操作

Histeq.m 使用库函数

HistogramEqualization.m 使用自定义函数

6.2自定义阈值将图像二值化处理

Thresholding.m 灰度化+二值化

6.3对图像添加噪声后再进行平滑处理

noise\_smooth.m

6.4边缘检测

sys\_prewitt.m 使用系统算子prewitt

myEdgeDetection.m 自定义算子+图像平滑处理

# 7.参考文献

1. https://blog.csdn.net/JHFUSFLSJH/article/details/120662271
2. https://blog.csdn.net/weixin\_45818370/article/details/124128203
3. https://blog.csdn.net/Aidam\_Bo/article/details/89916109
4. [图像处理之常见二值化方法汇总\_fu18946764506的博客-CSDN博客\_图像二值化方法](https://blog.csdn.net/fu18946764506/article/details/85062154?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~aggregatepage~first_rank_ecpm_v1~rank_v31_ecpm-3-85062154-null-null.pc_agg_new_rank&utm_term=%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86%E4%BA%8C%E5%80%BC%E5%8C%96%E7%9A%84%E5%87%A0%E7%A7%8D%E6%96%B9%E6%B3%95&spm=1000.2123.3001.4430)
5. http://en.wikipedia.org/wiki/Thresholding\_(image\_processing)
6. https://blog.csdn.net/qq\_42112618/article/details/120798578
7. https://blog.csdn.net/weixin\_43327191/article/details/108951725
8. https://blog.csdn.net/wenhao\_ir/article/details/51743382
9. https://blog.csdn.net/qq\_44310495/article/details/111288118