

**视听觉信号处理实验报告**

实验 一

题 目 图像的基本处理

学 院 未来技术学院

专 业 人工智能（视听觉信息处理）

学 号 7203610121

学 生 刘天瑞

任 课 教 师 姚鸿勋

哈尔滨工业大学计算机科学与技术学院

2022年秋季

说明

实验报告一般包含以下几个部分：实验内容，实验目的，实验设计、算法和流程，实验结果，结论，参考文献。

注意：不要把所有代码都粘贴到实验报告中，如确有必要，请粘贴少量关键代码即可，源代码单独提交！

Notes: Experimental report usually includes following sections: experiment content, experiment purpose, experiment design、algorithm and procedure， experimental results, conclusion, reference.

Do not copy and paste all source code into the report, if necessary, please paste some key codes. At the same time, please submit the total source code and the executive program.

**实验一 报告**

1. **实验内容（contents）**

1. 实现高斯噪声和椒盐噪声的添加；  
2. 实现直方图均衡化和归一化操作；  
3. 对实验内容1得到的加噪声后的图片实现中值滤波和均值滤波处理；  
4. 实现对图像的腐蚀与膨胀。

1. **实验目的（purposes）**

1. 掌握高斯噪声和椒盐噪声的添加；  
2. 掌握直方图均衡化和归一化操作；  
3. 掌握中值滤波和均值滤波；  
4. 掌握图像的腐蚀与膨胀运算。

1. **实验设计、算法和流程(Design, algorithm and procedure)**

1. 实验框架分析

首先本实验已经提供了一个较为完整的实验框架和主函数部分，只需补全实验功能（各算法函数的具体内容）即可，所以首先就需要了解整个实验框架的具体要求与接口规范，安装好PyQt5库，依此运行main.py。然后出现以下图像处理工具栏展示窗口：

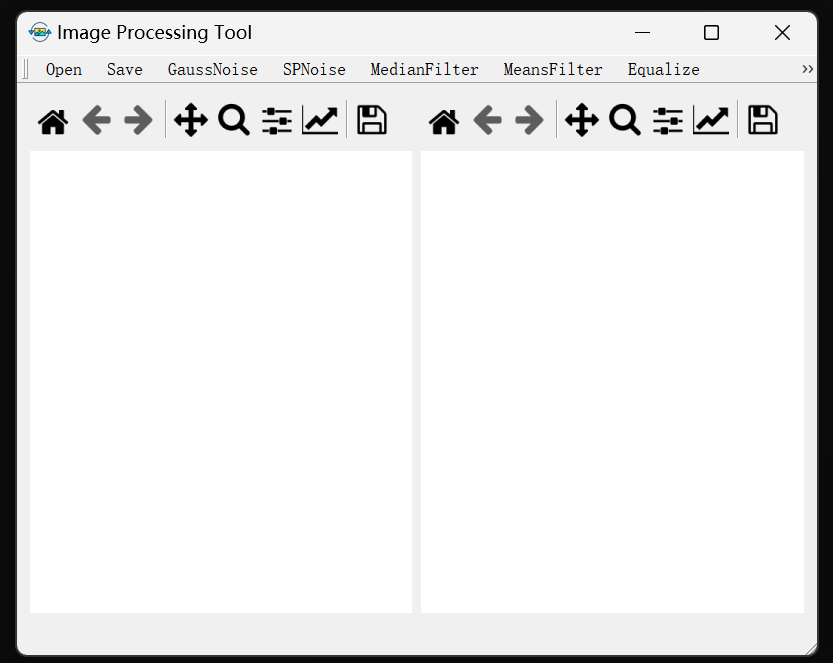


图1 Image Processing Tool可视化窗口

不断试错以及尝试后我掌握了一些功能：

Open用于打开提供的图像；

Save用于保存图片（值得注意的是所保存的图片会保留当前窗口的二维坐标）；

剩下8个按钮就是需要实现的算法功能。

然后再打开algorithm.py文件来查看各算法函数的参数输入与内容填充，分析各算法函数的图像均以像素矩阵的形式进行输入，但是在所提供的图像其中一些图片是灰度图，像素值是一维255，另外一些是RGB图，像素值为三维的255，因此需要对不同的图像进行接口适配。

1. 各算法函数功能的具体实现（包括算法介绍）

(i)高斯噪声：高斯噪声的添加较为简便，可以使用numpy库里的随机函数np.random.normal(0, 0.1, img.shape)来产生0到1范围内的高斯分布，其中函数调参中第一个参数为均值，第二个参数为方差，第三个参数为图像形状参数下的矩阵大小，最后将产生的噪声和原始图像叠加后再输出即可，如下图所示：

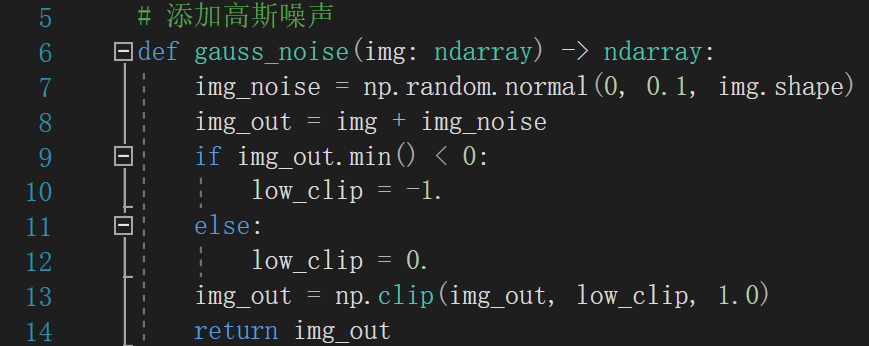


图2 高斯噪声算法具体运行代码

(ii)椒盐噪声：椒盐噪声的添加比高斯噪声稍微复杂一些，首先需要固定一个椒盐噪声比例系数不妨将其设置为0.05，然后对每个像素的值从高度到宽度均选择生成一个0到1范围内的随机值，此时我使用了numpy.random.randint（）函数来返回一个随机整型数进行黑白点噪声的添加，最后将其输出即可，如下图所示：

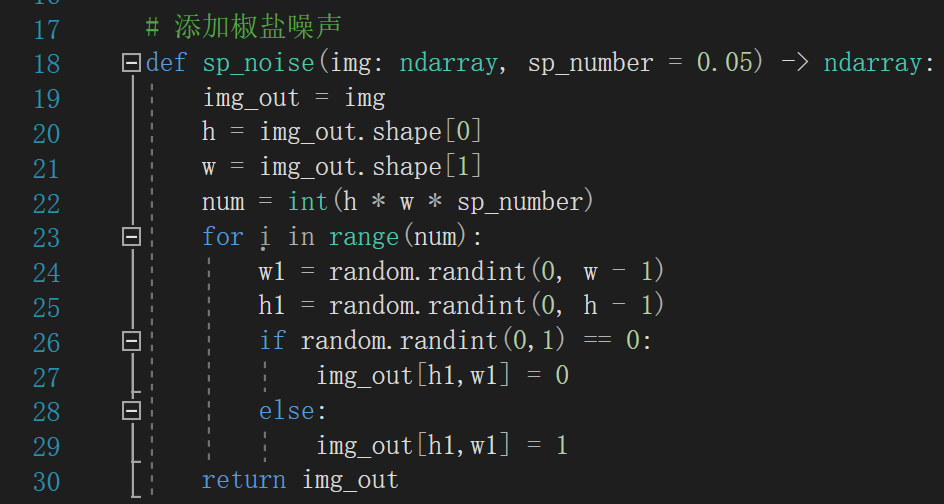


图3 椒盐噪声算法具体运行代码

(iii)中值滤波：中值滤波的实现需要将图像的每个像素用领域（以当前像素为中心的正方形区域）像素点的中值替代，这就要求查看各像素点周围像素点的情况，由于本次实验的要求包含尽量不使用已有的库函数，所以需要自己写一个窗口，再对其中数据进行具体操作处理。假设图像高度位h，宽度位w，每取一个像素点（分别0到h-2，0到w-2之间），我选择一个3\*3的窗口位，将每个像素点周围的8个像素点和它自己进行排序，然后将这个像素点的值设置为这9个像素点取中值即可，这里可以调用numpy库的median()函数，需要注意的是要适配RGB图像，如下图所示：

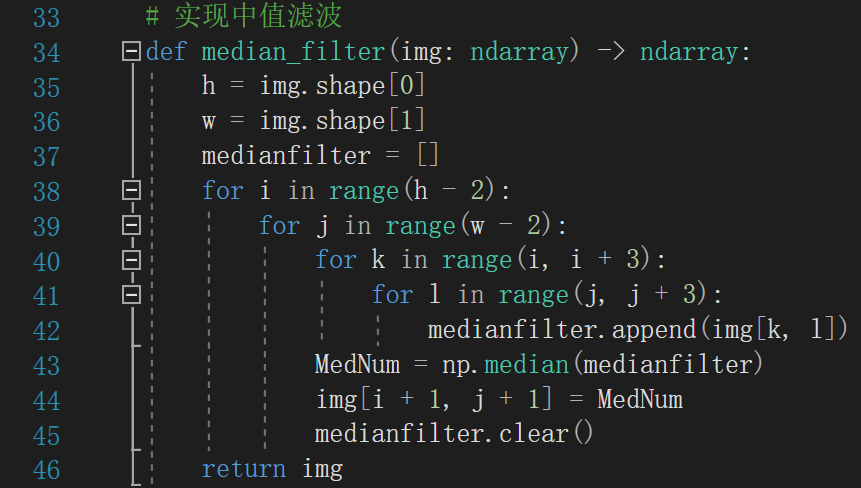


图4 中值滤波算法具体运行代码

(iv)均值滤波：均值滤波将图像的每个像素用领域（以当前像素为中心的正方形区域）像素点的均值替代，其中每个像素点的值都是其灰度。所以均值滤波和中值滤波差别只在于将像素点的值设置为九个像素的均值，只需将调用改为mean()函数，故不再赘述，如下图所示：

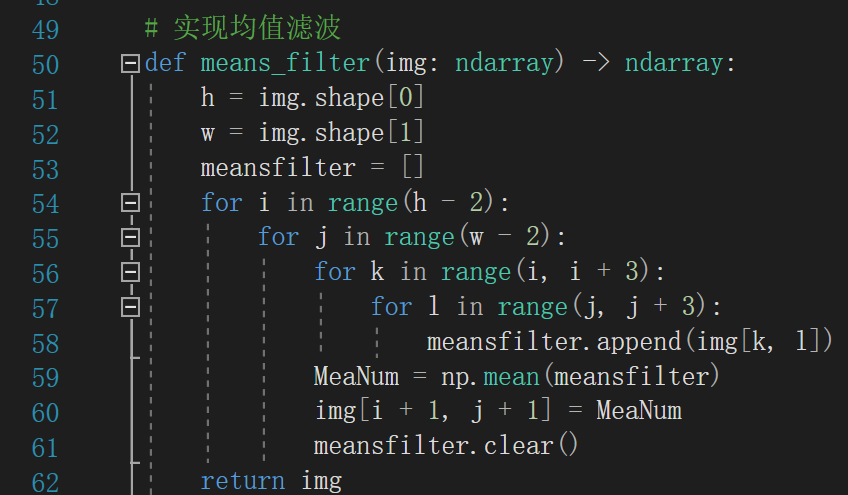


图5 均值滤波算法具体运行代码

(v)直方图均衡化：直方图均衡化的关键主要在于所输入的图像的值并非整型数，所以需要将其强制转化为unit8的0到255范围内的分布，再统计各个像素点的出现频率，最后直接可以采用直方图均衡化的公式，来获得其函数对应的值。直方图均值化操作可以分为如下5个步骤：

1.统计每个灰度的像素数量，默认规定是像素从0到255有256个划分等级。不妨初始化一个256维的数组，每一维对应该数组下标的灰度，当遍历这张图每个像素时，对应灰度的像素数量加一；

2.计算每个灰度像素占总像素数量的比值。得到灰度分布直方图。主要是为第三步做铺垫。求出每个点的概率值；

3.计算累计分布函数（可以类比概率分布函数）。从数学角度讲，就是做积分，计算每个灰度值前面所有灰度值像素数量之和；

4.计算映射函数，映射公式为(gmax-gmin)×C(f)+gmin+0.5，这里为计算后的灰度值（因为不一定为整数，所以需要统一向下取整），最后得到了一个256维度的数组，例如hist[m]=n则表示原先m灰度的像素全部映射为n灰度；

5.最后将下标为img[i, j]的像素灰度映射为n，这里重复遍历整张图像。

函数算法里的复杂度是O(n2)，如下图所示：

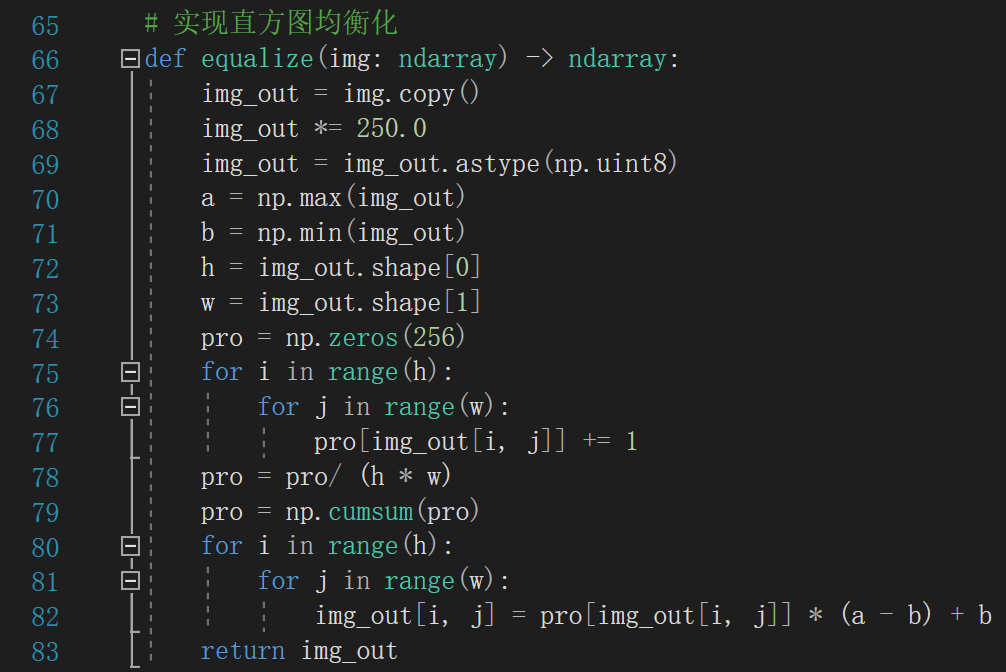


图6 直方图均衡化算法具体运行代码

(vi)直方图归一化：直方图归一化原理即让分布不均匀的图像像素点使其均匀分布于0到255范围内，其相对于均衡化操作来说较为简单，因为只需实现这个线性的对应关系即可，如下图所示：

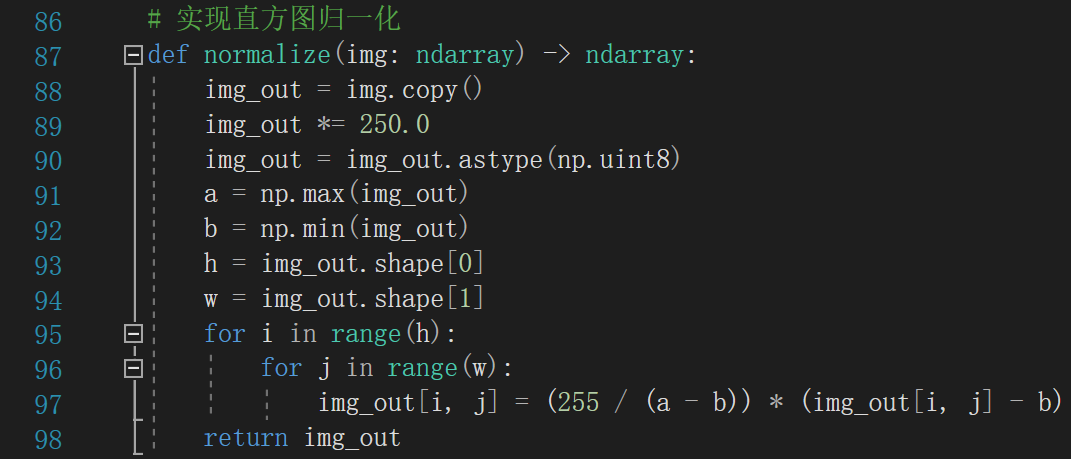


图7 直方图归一化算法具体运行代码

(vii)膨胀运算：膨胀算法需要设置核进行卷积运算，窗口和核进行卷积后，如果所得值小于所设阈值，则设置像素点的值，其实这里可以采用一个更为简单的算法：将3\*3的窗口位中的每一个像素点进行检查，若有白色像素点，便将其中心像素设置为白色，值得注意的是，前提是需要将输入的图像像素点矩阵进行二值化处理再进行图像复制，否则会得到全黑图像，如下图所示：

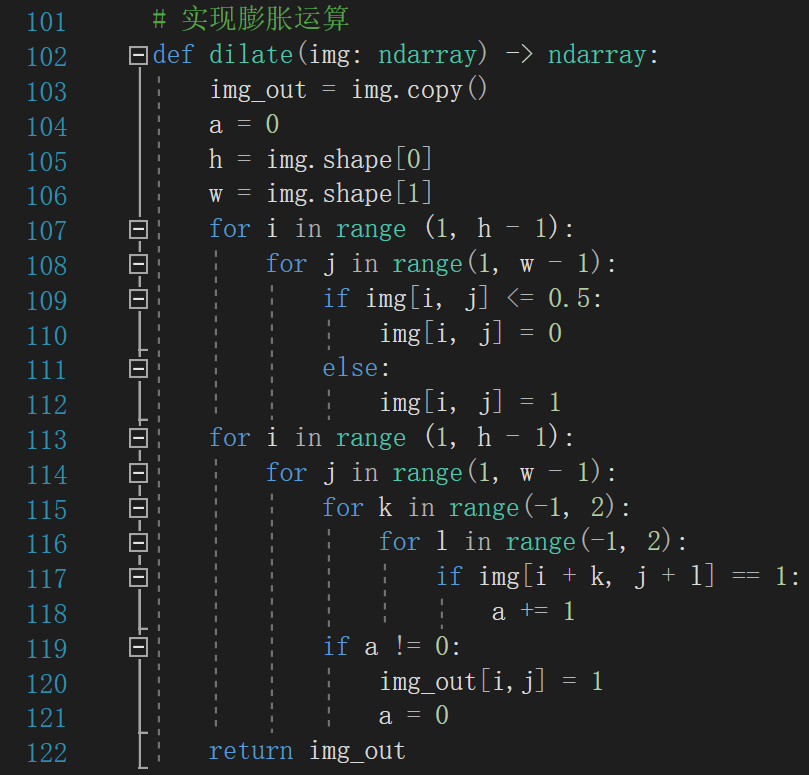


图8 膨胀运算具体运行代码

(viii)腐蚀运算：和膨胀算法同理，区别只是若有黑色像素点，便将其中心像素设置为黑色，其他实现不再赘述，如下图所示：

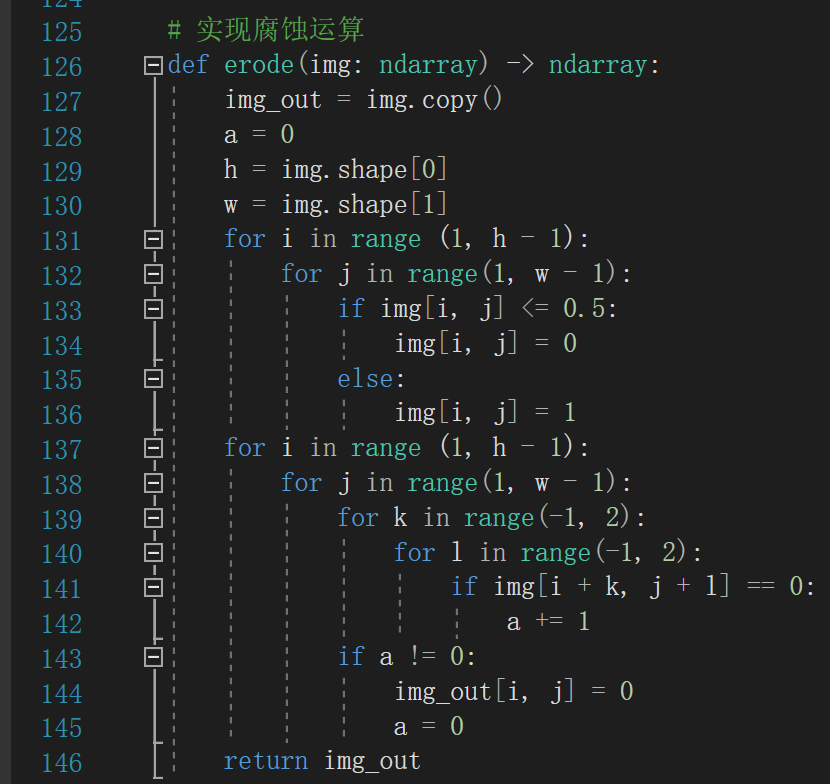


图9 腐蚀运算具体运行代码

1. **实验结果(results)**

(i)高斯噪声处理效果：

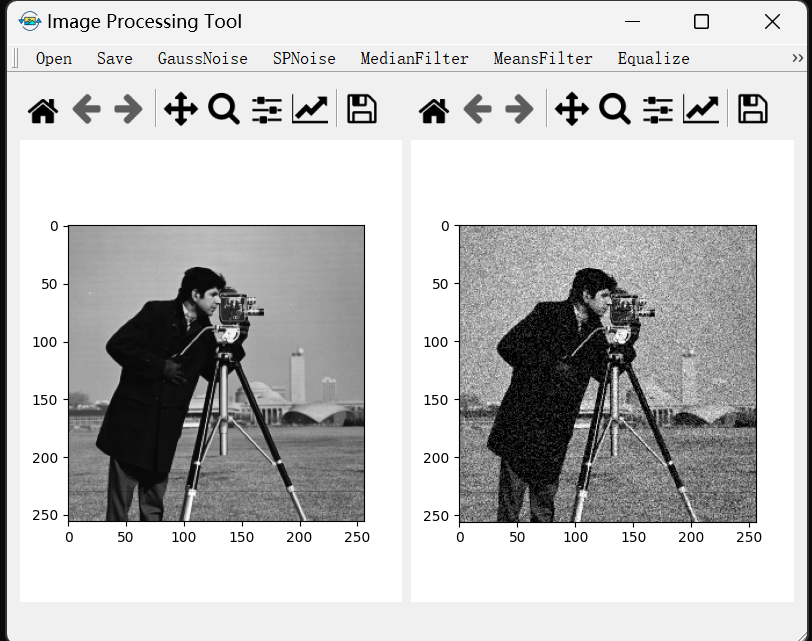
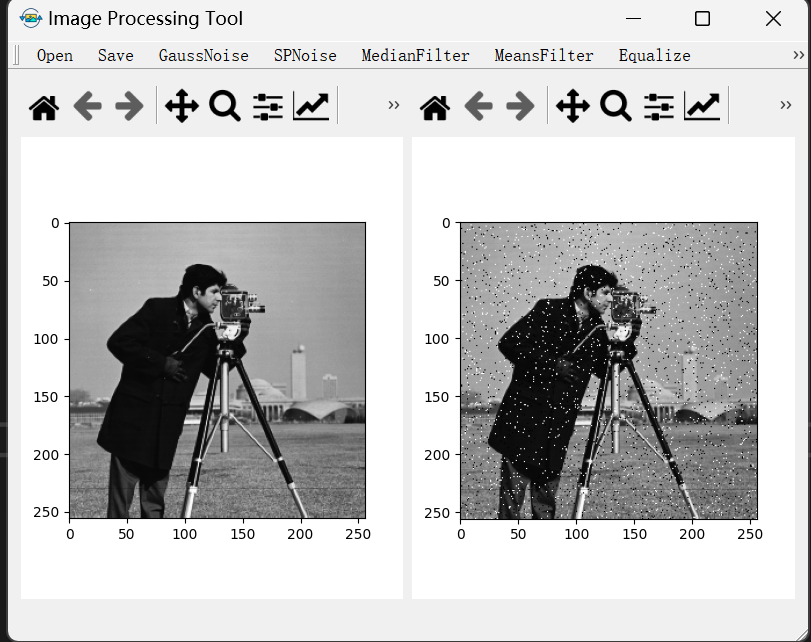


图10 高斯噪声处理效果

(ii)椒盐噪声处理效果：

图11 椒盐噪声处理效果

(iii)中值滤波处理椒盐噪声效果：

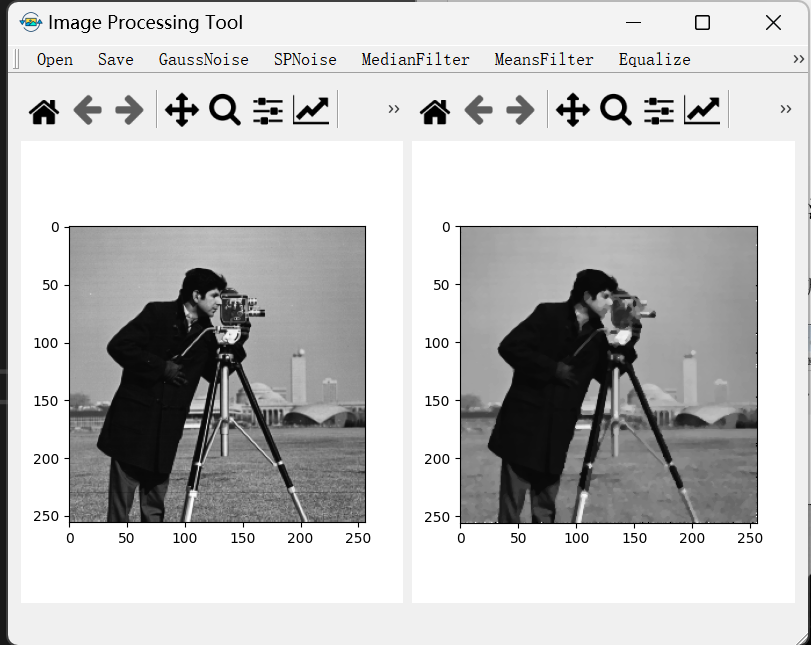


图12 中值滤波处理椒盐噪声效果、

(iv)中值滤波处理高斯噪声效果：

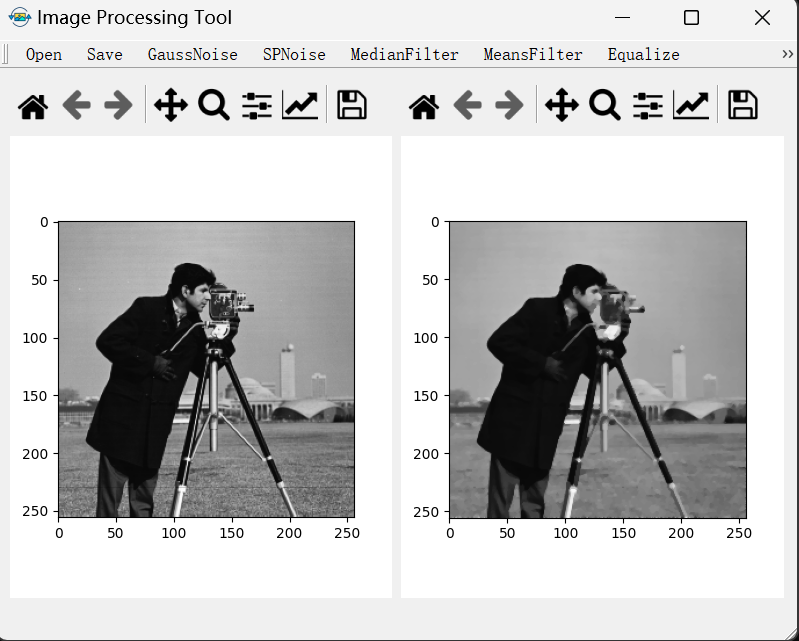


图13 中值滤波处理高斯噪声效果

(v)均值滤波处理高斯噪声效果：



图14 均值滤波处理高斯噪声效果

(vi)均值滤波处理椒盐噪声：

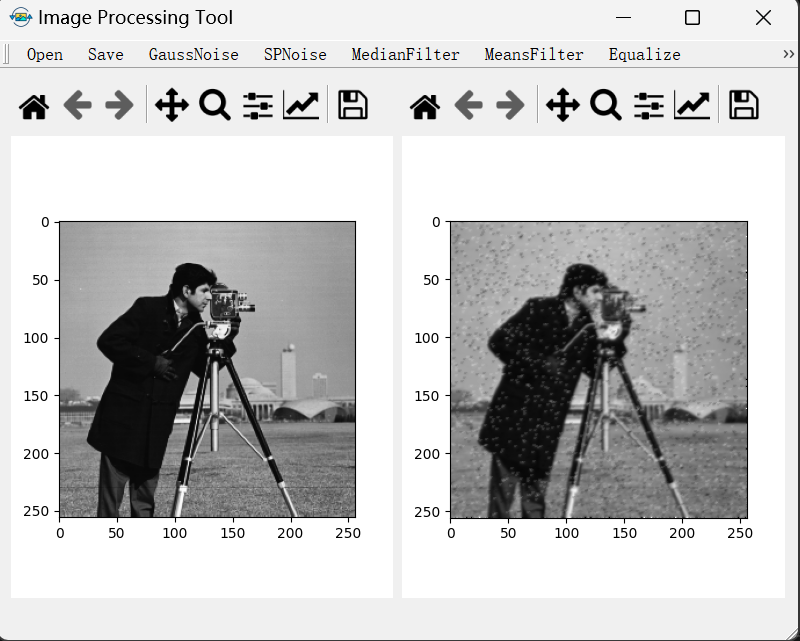


图15 均值滤波处理椒盐噪声

(vii)直方图均衡化处理效果：

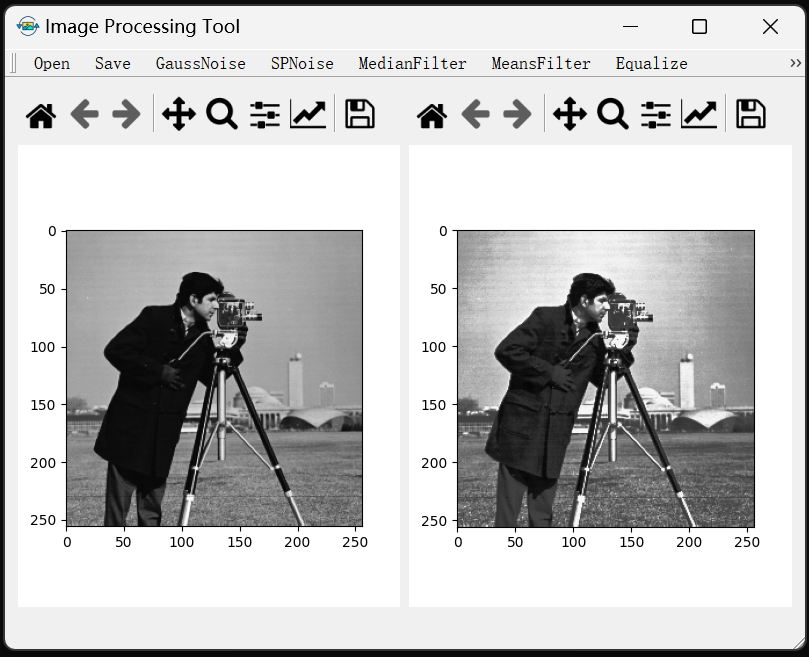


图16 直方图均衡化处理效果

(viii)直方图归一化处理效果：

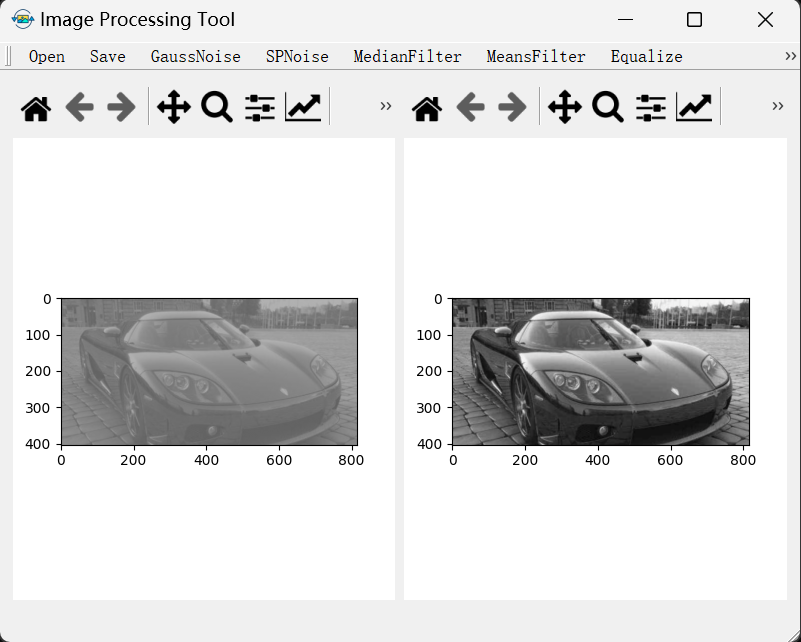


图17 直方图归一化处理效果

(ix)膨胀运算处理效果：

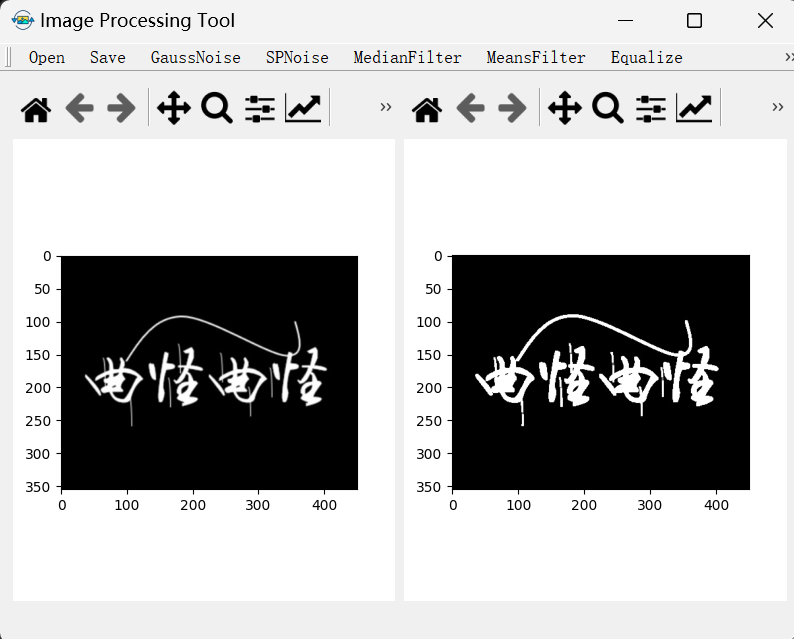


图18 膨胀运算处理效果

(x)腐蚀运算处理效果：

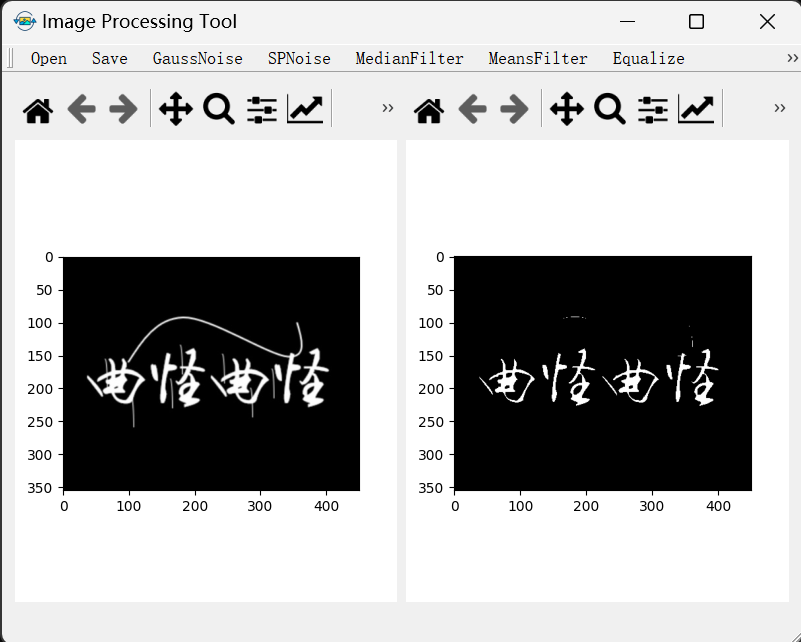


图19 腐蚀运算处理效果

1. **结论(conclusion)**

本次实验使我更加深入了解到了图像处理的一些具体有效的算法，也更加了解其是如何具体实现的。除此之外，在实际操作时我遇到的一些问题也值得深思，代码处理的一些接口规范也有助于避免遇到以上的问题。最后，我还做了如下总结：

* + - 1. 学会了对于图像基本的处理，包括图像的打开、保存、显示；
      2. 学会了图像空域增强算子的代码实现和应用，对什么情况运用什么算子已经有初步大致的了解和掌握，例如椒盐噪声最好的解决办法便是中值滤波；
      3. 学会了快速中值滤波算法，可以极大提高算法运行速度，降低时间复杂度；
      4. 学会了直方图均衡化和归一化的原理和步骤，是灰度级更加均匀达到图像信息增强的效果；
      5. 对比发现什么时候直方图均衡化效果比较好，什么时候效果并不是那么理想，同时对于直方图均衡化和归一化的实际运用有了初步掌握，例如医疗成像等等。

1. **参考文献(reference)**

[1] [【图像处理算法】直方图均衡化\_桂哥317的博客-CSDN博客\_直方图均衡化](https://blog.csdn.net/qq_15971883/article/details/88699218?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166727374316800180696969%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=166727374316800180696969&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-88699218-null-null.142%5ev62%5eopensearch_v2,201%5ev3%5eadd_ask,213%5ev1%5et3_esquery_v1&utm_term=%E7%9B%B4%E6%96%B9%E5%9B%BE%E5%9D%87%E8%A1%A1%E5%8C%96&spm=1018.2226.3001.4187) <https://blog.csdn.net/qq_15971883/article/details/88699218?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166727374316800180696969%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=166727374316800180696969&biz_id=0&utm_med>

[2] [数字图像处理---通俗理解腐蚀与膨胀\_alw\_123的博客-CSDN博客\_图像膨胀和腐蚀意义](https://blog.csdn.net/alw_123/article/details/83868878?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166727390916800180636521%2522%252C%2522scm%2522%253A%252220140713.130212432.pc%255Fall.%2522%257D&request_id=166727390916800180636521&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~pc_ctr_v1-5-83868878-null-null.142%5ev62%5eopensearch_v2,201%5ev3%5eadd_ask,213%5ev1%5et3_esquery_v1&utm_term=%E5%9B%BE%E5%83%8F%E8%85%90%E8%9A%80%E5%92%8C%E8%86%A8%E8%83%80&spm=1018.2226.3001.4187)

<https://blog.csdn.net/alw_123/article/details/83868878?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166727390916800180636521%2522%252C%2522scm%2522%253A%252220140713.130212432.pc%255Fall.%2522%257D&request_id=166727390916800180636521&biz_id=0&u>

[3] [图像滤波处理 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/344190267)

<https://zhuanlan.zhihu.com/p/344190267>

**注意：正文内容为小四，1.5倍行距**