机器视觉技术 第四次实验

实验题目 1——直方图均衡化

实验目的：

创建一幅在整个亮度范围内具有相同亮度分布的图像。

实验原理：

找到一个单调的像素亮度变换 q=T(p)，使输出直方图 G(q)在整个输出亮度范围[q0,qk]内是均匀的。增强了靠近直方图极大值附近的亮度的对比度，减小了极小值附近的对比度。

实验步骤：

1. 对于 k(256)个亮度级、大小为 M×N 的图像，创建长为 k 的数组 H，初始化为 0。
2. 形成图像直方图 H。
3. 形成累计直方图 Hc，其中 Hc[0]=H[0]，Hc[p]=Hc[p-1]+H[p]，p=1,2,……,k-1。
4. 设置 T[p]=round((k-1) Hc[p]/MN)， p=1,2,……,k-1。
5. 重新扫描图像，根据查找表获得变换结果。

程序代码：

实验结果显示：

测试图片：

结果 1：

结果 2：

实验分析总结：

以下是通过得到的结果图像：

通过对比以上三个结果。

实验题目 2——线性滤波

实验目的：

对含有噪音的图像进行线性滤波。

实验原理：

选取掩模，用其遍历图像。掩模覆盖区域的中心位置，在输出图像中替换为该区域的均值。最终去掉掩模无法覆盖的边缘区域，得到线性滤波后的图像。

实验步骤：

1. 确定掩模大小和掩模中心。
2. 在掩模内将像素点求均值，作为掩模中心的新像素值（通过归一化实现）。
3. 遍历图像。

程序代码：

```
#include <iostream>
#include <fstream>
#include "opencv2/opencv.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdio.h>

using namespace cv;
using namespace std;

Mat convolution(Mat img, Mat kernel) {
    // 获得尺寸
    auto Row = img.rows;
    auto Col = img.cols;
    auto row = kernel.rows;
    auto col = kernel.cols;

    // 转化
    img.convertTo(img, CV_32F);
    normalize(img, img, 0, 1, NORM_MINMAX, CV_32F);

    // 返回值初始化
    Mat Filter2D(Row, Col, CV_32F);
    Filter2D.setTo(Scalar(0));

    // 进行卷积
    for (int i = row / 2; i < Row - row / 2; i++) {
        for (int j = col / 2; j < Col - col / 2; j++) {
            for (int m = 0; m < row; m++) {
                for (int n = 0; n < col; n++) {
                    Filter2D.at<float>(i, j) += img.at<float>(i - col / 2 + m, j - row
/ 2 + n) * kernel.at<float>(m, n);
                }
            }
        }
    }

    // 归一化
    normalize(Filter2D, Filter2D, 0, 255, NORM_MINMAX, CV_8U);
```

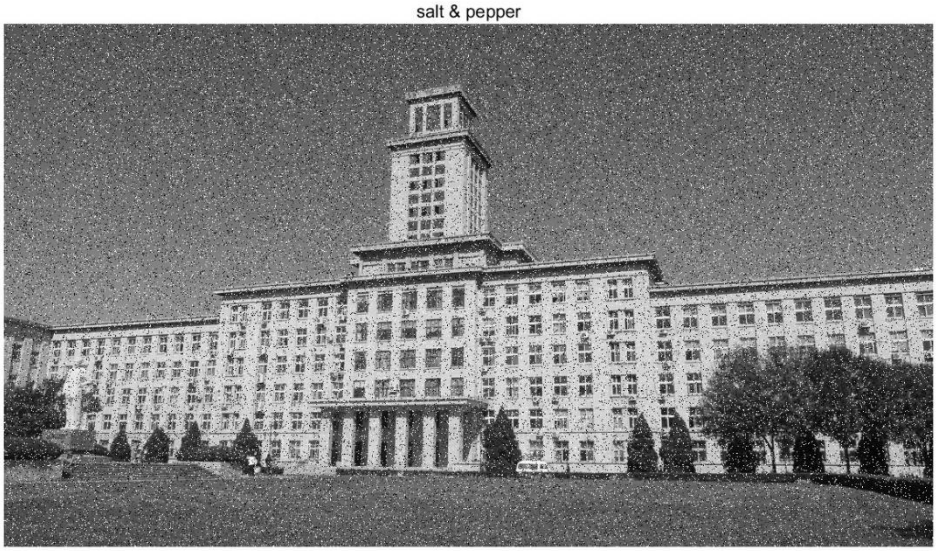
```
    return Filter2D;
}

Mat myFilter2D(Mat img)
{
    // 定义卷积核
    Mat kernel = (Mat_<float>(3, 3) <<
        1, 1, 1,
        1, 1, 1,
        1, 1, 1);

    // 进行卷积
    Mat Filter2D = convolution(img, kernel);

    // 返回线性滤波结果
    return Filter2D;
}

void main()
{
    Mat input = imread("testimg.jpg");

    // 彩色图转为灰度图
    Mat gray;
    cvtColor(input, gray, COLOR_BGR2GRAY);

    // 线性滤波，用'replicate'填充边界
    Mat Filter2D = myFilter2D(gray);

    imshow("input", input);
    imshow("Filter2D", Filter2D);
    waitKey(0);
}
```

实验结果显示：

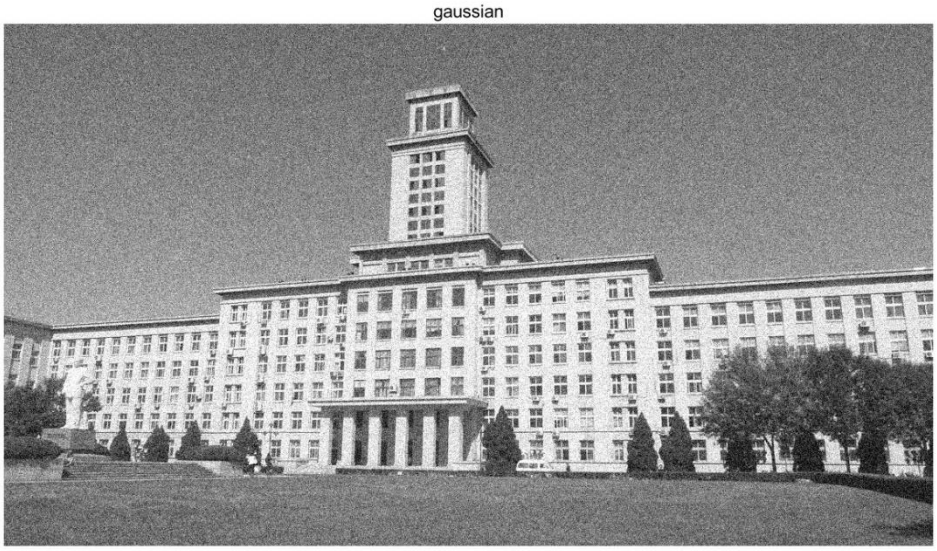
以下先通过 Matlab 调用函数生成了对应的椒盐噪声和高斯噪声图像，题目三也是使用了这两张图像作为测试对象。

测试图片：

椒盐噪声：

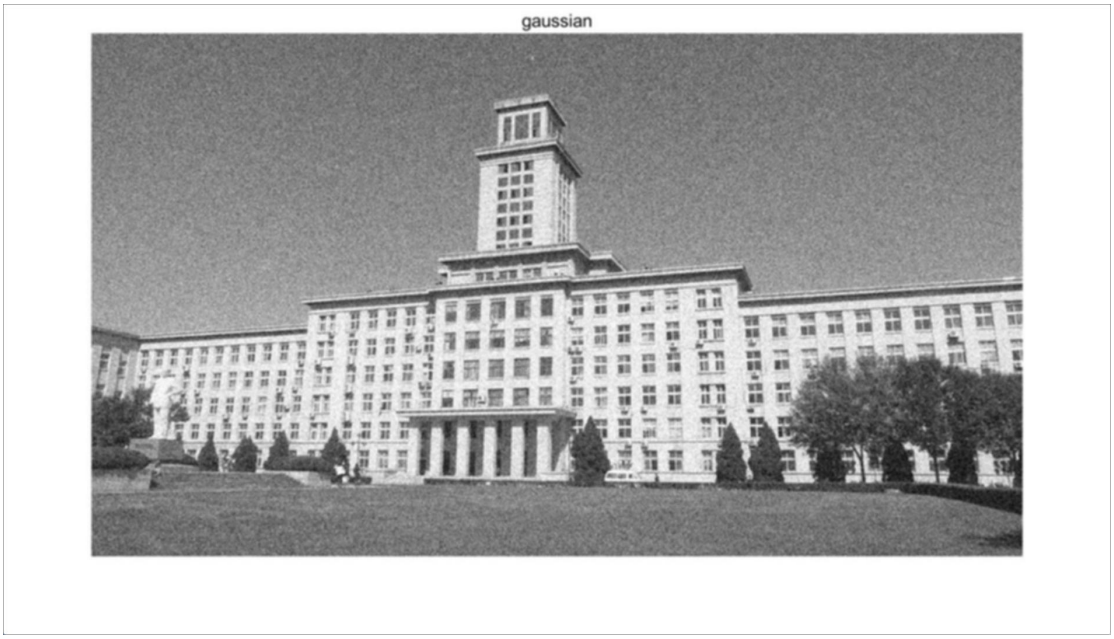


高斯噪声：



结果（椒盐噪声）：

结果（高斯噪声）：



实验分析总结：

通过对比以上两个结果，可以发现，简单的线性滤波对于噪声的处理效果一般（没有下面的中值滤波好）。而线性滤波对于高斯噪声的滤波效果要强于椒盐噪声。

为保证输入输出图像大小一致，输出图像的外围留了一圈没有进行卷积，因为初始化为 0，所以会有黑边。

实验题目 3——中值滤波

实验目的：

对含有噪音的图像进行中值滤波。

实验原理：

选取 n（n 为奇数）维方阵掩模，用其遍历图像。掩模覆盖区域的中心位置，在输出图像中替换为该区域的中值。最终去掉掩模无法覆盖的边缘区域，得到中值滤波后的图像。

实验步骤：

1. 确定掩模大小和掩模中心。
2. 在掩模内将像素点按亮度值大小排序。
3. 选择序列的中间值作为掩模中心的新像素值。
4. 遍历图像。

程序代码：

main.m

```
clc;clear all;close all;

% 读入图像
```

```
img=imread('testimg.jpg');

% 彩色图转为灰度图
img=rgb2gray(img);

% 加噪音
in = imnoise(img, 'salt & pepper', 0.1);

% 中值滤波
Medfilt2_result = MyMedfilt2(in);

% 输出图像
imshow(img);
title('img');
figure;
imshow(in);
title('salt & pepper');
figure;
imshow(Medfilt2_result,[]);
title('Medfilt2_result');
```

MyMedfilt2.m

```
% 定义函数 MyMedfilt2，参数为二值图像 img
function Medfilt2_result=MyMedfilt2(img)

% 获取 img 的行数 rows 和列数 cols
[rows, cols] = size(img);

% 3x3 掩模
img_0 = zeros(rows, cols);
for i=2:rows - 1
    for j=2:cols - 1

A=[img(i-1,j-1),img(i-1,j),img(i-1,j+1),img(i,j-1),img(i,j),img(i,j+1),img(
i+1,j-1),img(i+1,j),img(i+1,j+1)];
        img_0(i,j)=median(A);
    end
end

% 返回值
Medfilt2_result=img_0(2:rows - 1,2:cols - 1);

end
```

实验结果显示：
测试图片：
测试对象为 Matlab 调用函数生成的椒盐噪声和高斯噪声图像，见上一题目。
结果（椒盐噪声）：

结果（高斯噪声）：

实验分析总结：
以下是通过 Matlab 调用“medfilt2(in,[a a], 'symmetric')”函数得到的结果图像：
对椒盐噪声中值滤波：
a=2：

if1

a=3：

if2

对高斯噪声中值滤波：

a=2：

if1

a=3：

if2

实验分析总结：

通过对比以上结果，可以看出，中值滤波针对椒盐噪声的滤波能力比较强，滤波效果好，而相比之下，对高斯噪声的滤波效果较差。针对相同噪声，中值滤波的掩模越大，效果越好。