数字图像处理第六次作业

班级：自动化 56

姓名：陈泊言

学号：2160504056

完成日期：4.2

摘要：

图像复原技术的主要目的是以预先确定的目标来改善图像，通常会涉及一个最佳准则来产生期望结果的最佳估计。本次实验首先介绍了图像退化/复原模型和两种典型的噪声模型——高斯噪声和椒盐噪声，然后用多种方法为图像去噪，并对比分析不同方法的区别；最后介绍了最小均方差滤波对运动引起的图像模糊的处理方法。

一、图像复原与重建

1.图像复原与图像增强之间的关系

图像复原与图像增强相似，主要目的是以预先确定的目标来改善图像。但图像增强是一个主管的过程，而图像复原主要是一个客观的过程。图像复原试图通过通过退化现象的某种先验知识来复原被退化的图像。因此复原技术是面向退化模型的，并且采用相反的过程进行处理，以便恢复出原始图像。例如对比度拉伸属于一种图像增强技术，给观看者提供更能够接受的图像；而去除图像的模糊则被认为是一个图像复原过程。

在图像复原的过程中，当退化仅仅是加性噪声时，空间处理就非常适用；而像图像模糊这样的退化在空域使用较小的滤波模板就很难恢复。在这种情况下基于不同优化准则的频域滤波有时是使用的。

2.图像退化/复原过程的模型

在这里图像的退化过程被建模为一个退化函数和一个加性噪声的形式，即

$$g(x,y)=h(x,y)\bigstar f(x,y)+\eta(x,y)$$

这里 f(x,y)f(x,y)是原始图像，h(x,y)h(x,y)是退化函数，η (x,y)η (x,y)是加性噪声，g(x,y)g(x,y)是产生的退化之后的图像，而星号表示两者的卷积。当然这个过程也可在频域内进行表示

$$G(u,v)=H(u,v)F(u,v)+N(u,v)$$

具体的过程可以用下图的形式表示

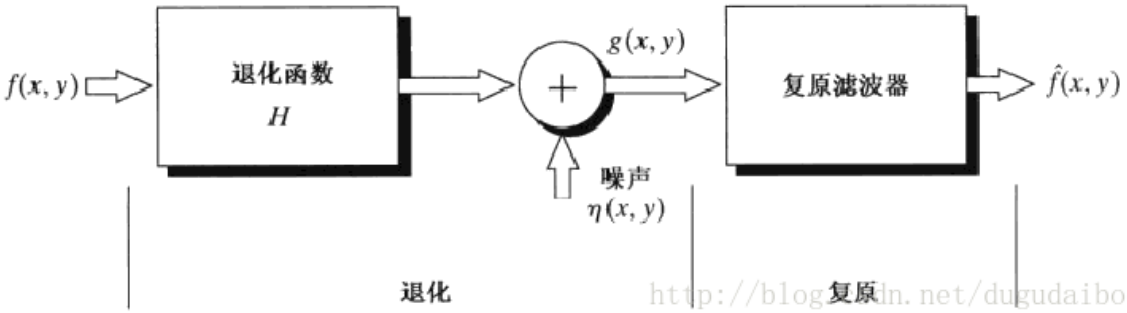
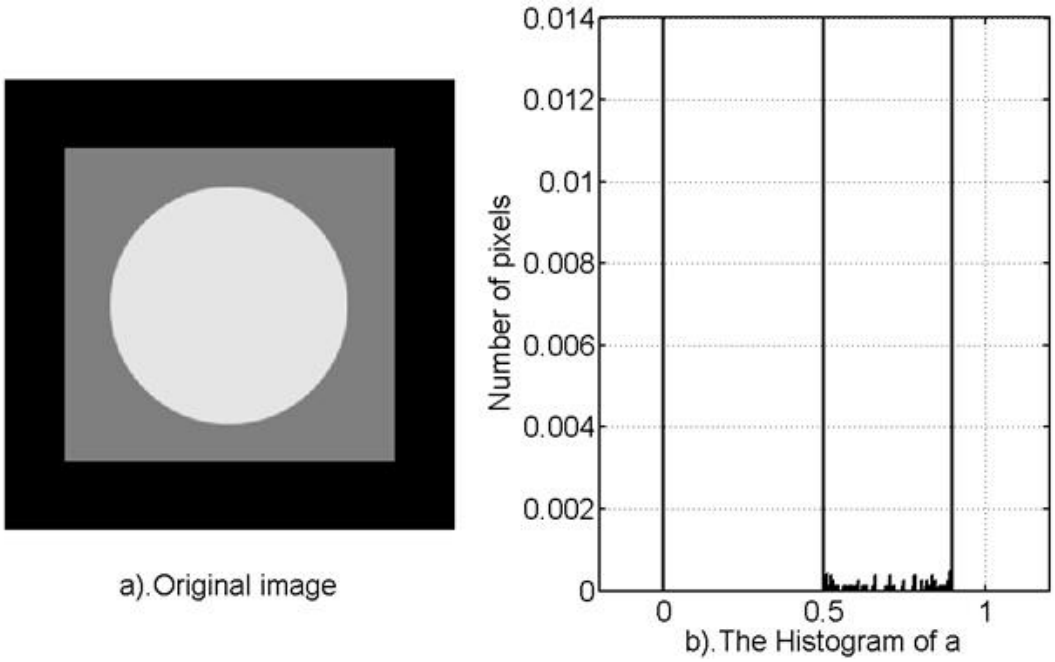


图 1 图像退化/复原过程的模型

而我们需要做的就是在已知噪声项和退化函数的基础上，估计f^(x,y)f^(x,y)，这就是希望得到的原始图像。具体来讲主要分为两个部分，一个部分是估计噪声项并将噪声项去掉；对图像进行去噪之后在对图像进行恢复。

二、典型噪声模型。

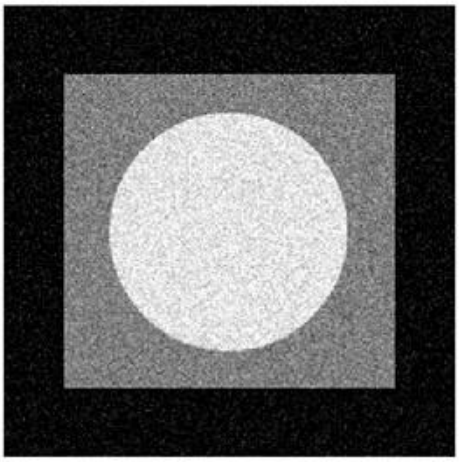
1. 评价用图像与其直方图



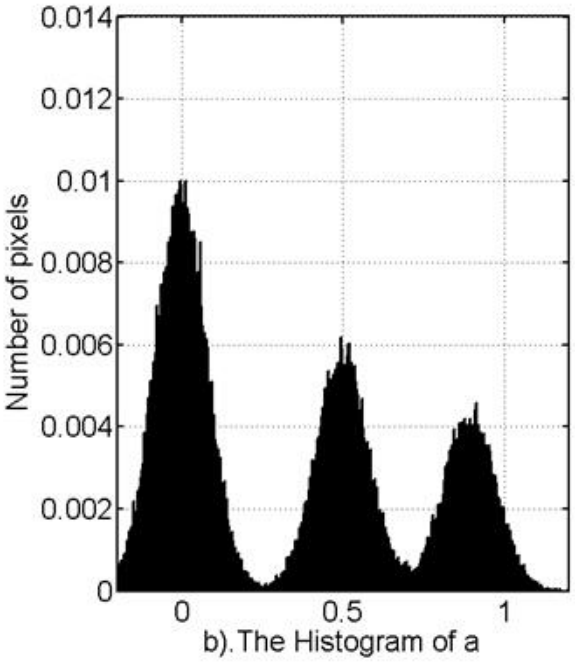
2. 高斯噪声

高斯噪声，也称为正态噪声，其统计特性服从正态分布。一种较为泛用的噪声模型。

$$p(z)=\frac{1}{\sqrt{2\pi}\sigma}e^{-(z-\bar{z})^2/2\sigma^2}$$



a).Ruselt of Gaussian noise



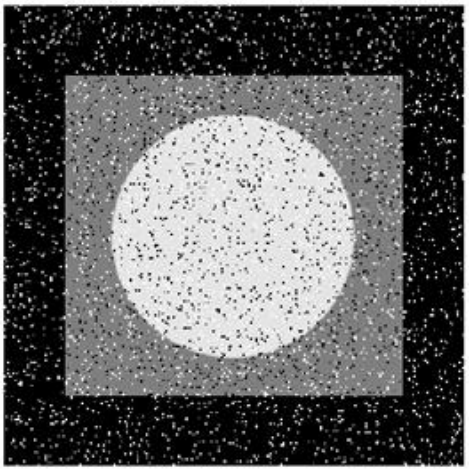
b).The Histogram of a

3.椒盐噪声

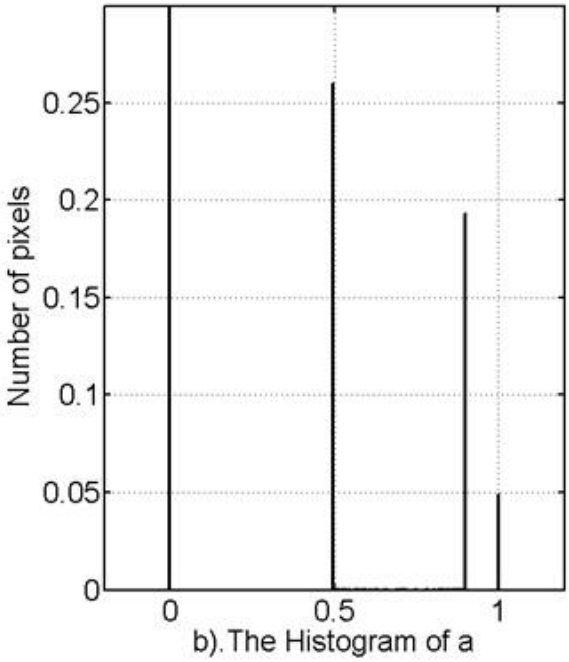
椒盐噪声也成为双脉冲噪声。在早期的印刷电影胶片上，由于胶片化学性质的不稳定和播放时候的损伤，会使得胶片表面的感光材料和胶片的基底欠落，在播放时候，产生一些或白或黑的损伤。事实上，这也可以归结为特殊的椒盐噪声。椒盐噪声的实现，需要一些逻辑判断。这里我们的思路是，产生均匀噪声，然后将超过阈值的点设置为黑点，或白点。

$$p(z)=\begin{cases}P_a, & z=a\\ P_b, & z=b\\ 1-P_a-P_b, & \text{其他}\end{cases}$$

其中盐表示亮点，椒表示暗点。



a).Ruselt of Salt & pepper noise



三、高斯噪声的产生与消除

1. 题目分析

在测试图像上产生高斯噪声 lena 图-需能指定均值和方差;并用多种滤波器恢复图像，分析各自优缺点；

根据上文的高斯噪声的公式，以及高斯噪声的原理：图像中每一点都存在噪声，但幅值是随机分布的。自行编写产生高斯噪声代码后，使用算术均值滤波，几何均值滤波，中值滤波器进行去噪并分析优缺点。

2.滤波器原理

1．算术均值滤波器

这是最简单的均值滤波器。令 SxySxy 表示中心点在(x,y)(x,y)处，大小为 m×n 的滤波器窗口。算术均值滤波器就是简单的计算窗口区域的像素均值，然后将均值赋值给窗口中心点处的像素：

$$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}}g(s,t)$$

可以去除均匀噪声和高斯噪声，但会对图像造成一定程度的模糊

2．几何均值滤波器

几何均值滤波器，求某个区域内的几何平均值。

$$\hat{f}(x,y)=\left[\prod_{(s,t)\in S_{xy}} g(s,t)\right]^{\frac{1}{mn}}$$

对于这个滤波器，该滤波器产生的去噪效果可以与算术平均值滤波器基本一样，但是可以更少的丢失细节。

3．中点滤波器

中点滤波器基本原理是把数字图像或数字序列中一点的值用该点的一个临域中各点值的最小值与最大值的和的均值替换。

用3*3模板说明如下：

P1	P2	P3
P4	P0	P5
P6	P7	P8

Fig.11 3*3邻域模版

$$P\min = Min(P1,P2\cdots P8) \qquad 2\text{-}(104)$$

$$P\max = Max(P1,P2\cdots P8) \qquad 2\text{-}(105)$$

$$P0=\frac{(P\min+P\max)}{2} \qquad 2\text{-}(106)$$

这种滤波器结合了顺序统计和求平均，对于高斯和均匀随机分布这类噪声有最好的效果。

Midpoint(A)=(max[A(x+i,y+j)]+min[A(x+i,y+j)])/2　(x,y)属于M

注：(x+i,y+j)是定义在图像上的坐标，(i,j)是定义在模板M上的坐标。M即为运算的模板。

它最适合处理随机分布噪声，如高斯噪声或均匀噪声。

3.实验结果

高斯噪声：　均值=0；　方差=0.08

原图像

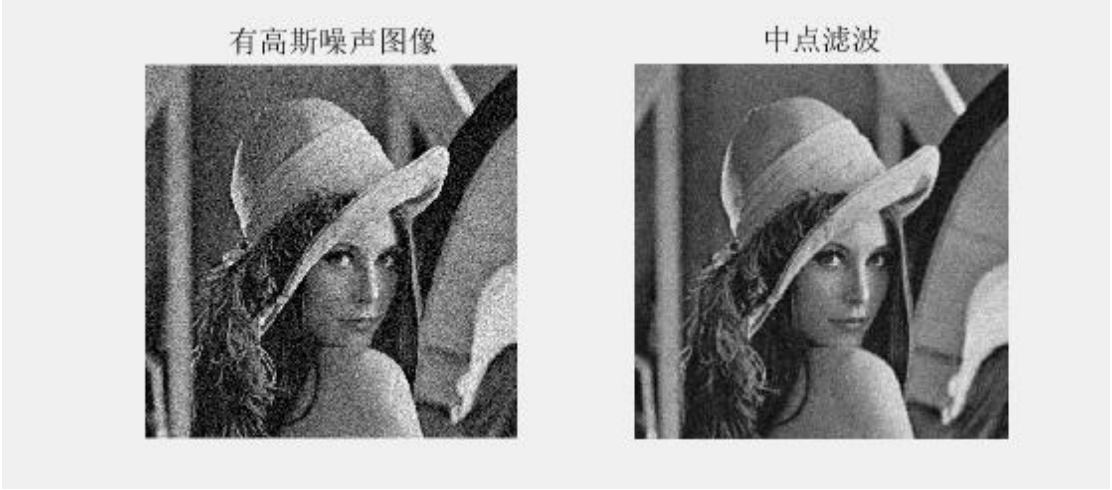
加高斯噪声后

有高斯噪声图像

算术均值滤波

有高斯噪声图像

几何算术均值滤波



四、椒盐噪声的产生与消除

1.题目分析

在测试图像 lena 图加入椒盐噪声（椒和盐噪声密度均是 0.1）；用学过的滤波器恢复图像；在使用反谐波分析 Q 大于 0 和小于 0 的作用；

根据椒盐噪声的公式与原理：噪声幅值基本相同，但出现位置随机，自行编写 matlab 程序，使用逆谐波均值滤波器，中值滤波器，最大，小值滤波器，对图像进行去噪处理并分析。

2.滤波器原理

1．逆谐波均值滤波器

逆谐波滤波器可以对应多种噪声，式子如下。

$$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

这个滤波器，可以通过 Q 值的变化，来获得一定的效果。当 Q 为正，这个滤波器可以去除胡椒噪声，当 Q 为负，这个滤波器可以去除盐粒噪声。

2．中值滤波

中值滤波器的想法很简单，如果一个信号是平缓变化的，那么某一点的输出值可以用这点的某个大小的邻域内的所有值的统计中值来代替。这个邻域在信号处理领域称之为窗（window）。窗开的越大，输出的结果就越平滑，但也可能会把我们有用的信号特征给抹掉。所以窗的大小要根据实际的信号和噪声特性来确定。

$$\hat{f}(x,y)=\underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\}$$

中值滤波器的应用十分普遍，因为对于某些类型的随机噪声，它们可提供良好的去噪能力，且与相同尺寸的线性平滑滤波器相比，引起的模糊更少。在存在单极或双极脉冲噪声的情况下，中值滤波尤其有效。

3．最大，小值滤波器

最大最小值滤波是一种比较保守的图像处理手段，与中值滤波类似，首先要排序周围像素和中心像素值，然后将中心像素值与最小和最大像素值比较，如果比最小值小，则替换中心像素为最小值，如果中心像素比最大值大，则替换中心像素为最大值。

最大值滤波器

$$\overline{f}(x,y)=\max_{(s,t)\in S_{xy}}\{g(s,t)\}$$

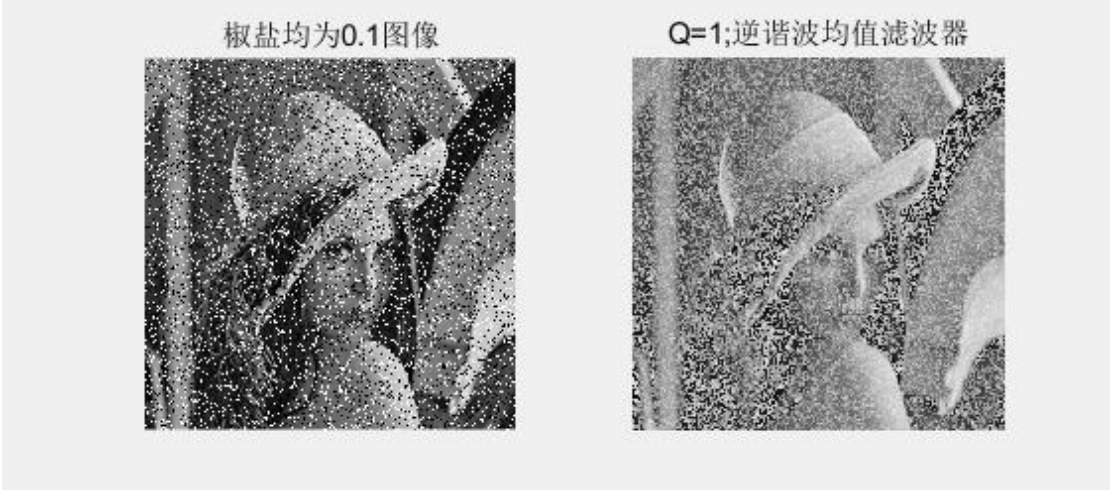
这种滤波器对发现图像中的最亮点非常有用，有效处理胡椒噪声。

最小值滤波器

$$\overline{f}(x,y)=\min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

这种滤波器对于发现图像的最暗点非常有用，有效降低盐粒噪声。

3. 实验结果



椒盐均为0.1图像

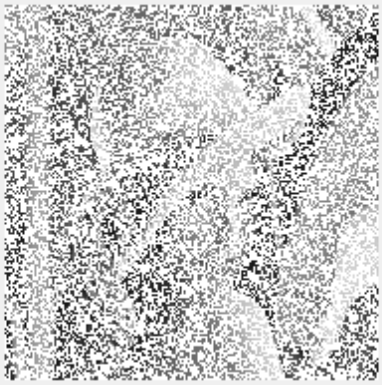
Q=-1;逆谐波均值滤波器

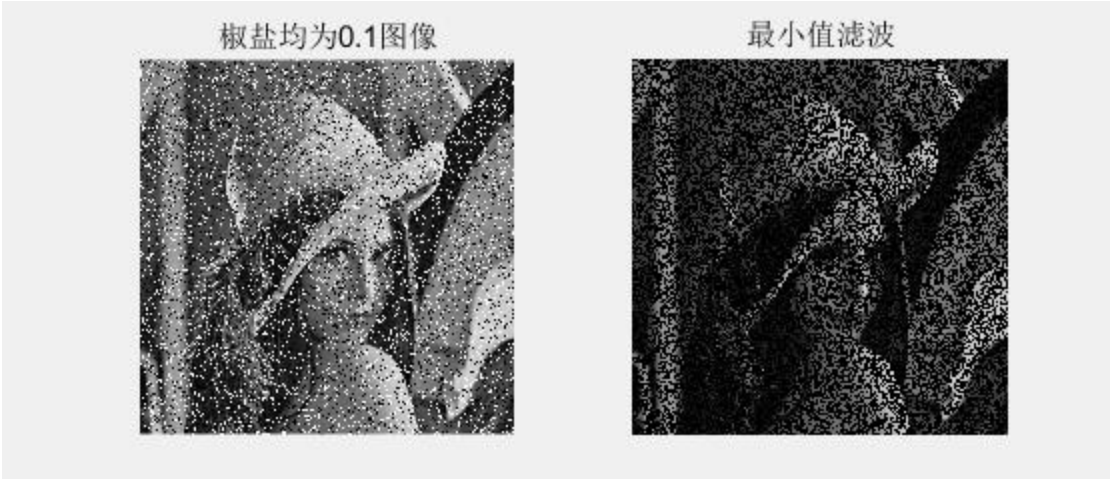
椒盐均为0.1图像

中点滤波

椒盐均为0.1图像

最大值滤波





4.实验分析

从实验结果可以得出，均值滤波器中的逆谐波均值滤波器适合处理脉冲信号，但是有缺点，及必须知道是暗噪声还是亮噪声，以便 Q 选择的正确的符号，如果 Q 选择错误会引起相反的效果，而且该滤波器不能处理同时有胡椒噪声和盐粒噪声的图像，因为只能根据 Q 的选择，识别一种亮度来滤波。而统计排序滤波器中中值滤波器对同时有胡椒，盐粒噪声的处理效果很好，但最大值和最小值滤波器与 Q 等于正和 Q 等于负的处理结果相同。

五、维纳滤波器

1.题目分析

(a) 实现模糊滤波器如方程 Eq. (5.6-11).

(b) 模糊 lena 图像：45 度方向，T=1；

(c) 再模糊的 lena 图像中增加高斯噪声，均值= 0 ，方差=10 pixels 以产生模糊图像；

(d)分别利用方程 Eq. (5.8-6)和(5.9-4)，恢复图像；并分析算法的优缺点

2.维纳滤波器原理

1. 退化函数

$$H(u,v)=\frac{1}{1+\left[\dfrac{D_0^2}{D_1(u,v)D_2(u,v)}\right]^n}$$

$$D_1(u,v)=[(u-M/2-u_0)^2+(v-N/2-v_0)^2]^{1/2}$$

$$D_2(u,v)=[(u-M/2+u_0)^2+(v-N/2+v_0)^2]^{1/2}$$

2．最小均方误差滤波

维纳滤波建立在图像和噪声都是随机变量的基础上，目标是找到未污染图像 f 的一个估计 f，使他们之间的均方误差最小。这种误差度量由下式给出下式

$$e^2 = E\{(f - f')^2\}$$

式中 E{}是参数的期望值。假设维纳滤波器的傅立叶变换表达式为 W(u,v)，那么上式可变为

$$\begin{aligned} e^2 &= E\{[F(u,v) - W(u,v)G(u,v)]^2\} \\ &= E\{[F(u,v) - W(u,v)(H(u,v)F(u,v) + N(u,v))]^2\} \end{aligned}$$

之后进行平方展

$$\begin{aligned} e^2 &= (1 - W(u,v)H(u,v))(1 - W(u,v)H(u,v)) * E\{|F(u,v)|^2\} \\ &-(1 - W(u,v)H(u,v))W(u,v) * E\{F(u,v)N*(u,v)\} \\ &-W(u,v)(1 - W(u,v)H(u,v)) * E\{N(u,v)F*(u,v)\} \\ &+W(u,v)W*(u,v)E\{|N(u,v)|^2\} \end{aligned}$$

之后我们假设噪声和独立信号无关

$$E\{F(u,v)N*(u,v)\} = E\{N(u,v)F*(u,v)\} = 0$$

并定义如下功率谱

$$\begin{aligned} S_\eta(u,v) &= |N(u,v)|^2 \text{，噪声的功率谱} \\ S_f(u,v) &= |F(u,v)|^2 \text{，未退化图像的功率谱} \end{aligned}$$

之后对 W(u,v)求导，并使导数为 0

$$\frac{de^2}{dW(u,v)} = W*(u,v)S_\eta(u,v) - H(u,v)(1 - W(u,v)H(u,v)) * S_f(u,v) = 0$$

之后对公式进行化简

$$W*(u,v)S_\eta(u,v) = H(u,v)(1 - W(u,v)H(u,v)) * S_f(u,va)$$

左右同时取共轭

$$W(u,v)S_\eta*(u,v) = H*(u,v)(1 - W(u,v)H(u,v))S_f*(u,v)$$

化简得

$$(S_\eta(u,v) + |H(u,v)|^2 S_f(u,v))W(u,v) = H*(u,v)S_f(u,v)$$

维纳滤波的基本公式是

$$\hat{F}(u,v)=[\frac{H^*(u,v)}{|H(u,v)|^2+P_N(u,v)/P_S(u,v)}]G(u,v)=[\frac{1}{H(u,v)}\cdot\frac{|H(u,v)|^2}{|H(u,v)|^2+P_N(u,v)/P_S(u,v)}]G(u,v)$$

其中H(u,v)表示退化函数，则维纳滤波退化为逆滤波（又称为理想的逆滤波器）。所以，逆滤波也可以认为是维纳滤波的一种特殊情况。而且，没有噪声情况下的逆滤波中，逆滤波可以很正常对退化图像进行复原。

但是在实际应用中，通 $P_N(u,v)/P_S(u,v)$ 常是是未知的，因此通常的做法是会采用一个常数值 $P_N(u,v)/P_S(u,v)$ K来代替，这时K就变成了一个可以调节的参数，于是维纳滤波的公式又可以写成：

$$\hat{F}(u,v)=[\frac{1}{H(u,v)}\cdot\frac{|H(u,v)|^2}{|H(u,v)|^2+k}]G(u,v)$$

3．约束最小二乘滤波

图像复原的方法较多，按大类可分为无约束复原和有约束复原（或称约束复原）两种，即无约束复原和有约束复原（constraint restoration）。无约束恢复是一种在图像恢复过程中不受其他条件限制的一种方法。典型方法是逆滤波法。此外，在图像恢复过程中，为了在数学上更容易处理，常常给复原加上一定的约束条件，并在这些条件下使某个准则函数最小化。这类方法叫做有约束恢复，其中典型的方法有维纳滤波法和约束最小平方滤波法。

无约束恢复是一种在图像恢复过程中不受其他条件限制的一种方法。在对 n 没有先验知识的情况下，需要寻找一个 f 的估计 fˆ，使得 Hfˆ在最小均方误差的意义下最接近 g，即要使 n 的范数最小。即

$$||n||^2=n^Tn=||g-H\hat{f}||^2=(g-H\hat{f})(g-H\hat{f})$$

最小化上式则可以推出：

$$\hat{f}=H^{-1}g$$

其约束条件为

$$C=\sum_{0}^{M-1}\sum_{0}^{N-1}[\nabla^2f(x,y)]^2$$

即

$$||G - H\hat{F}||_2^2 = ||N||_2^2$$

这里，F^是为退化图像的估计，N 为加性噪声，拉普拉斯算子∇2 在这里表示平滑程度。

这个最佳优化问题在频域域的解决表达式为：

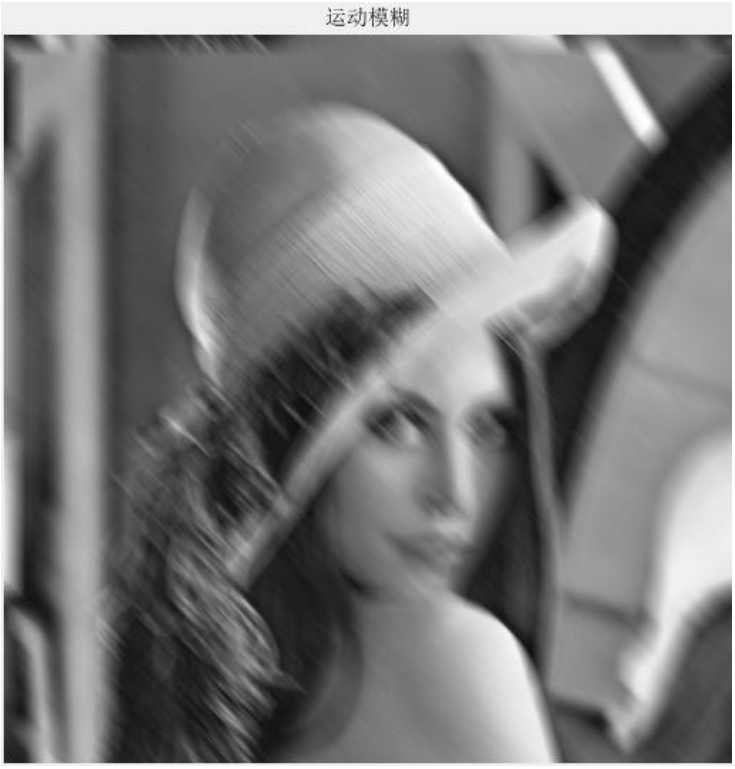
$$\hat{F} = \frac{\lambda H^* G}{\lambda H^* H + P^* P} = \left[\frac{H^*}{||H||_2^2 + \gamma||P||_2^2}\right] G$$

这里，P 是函数

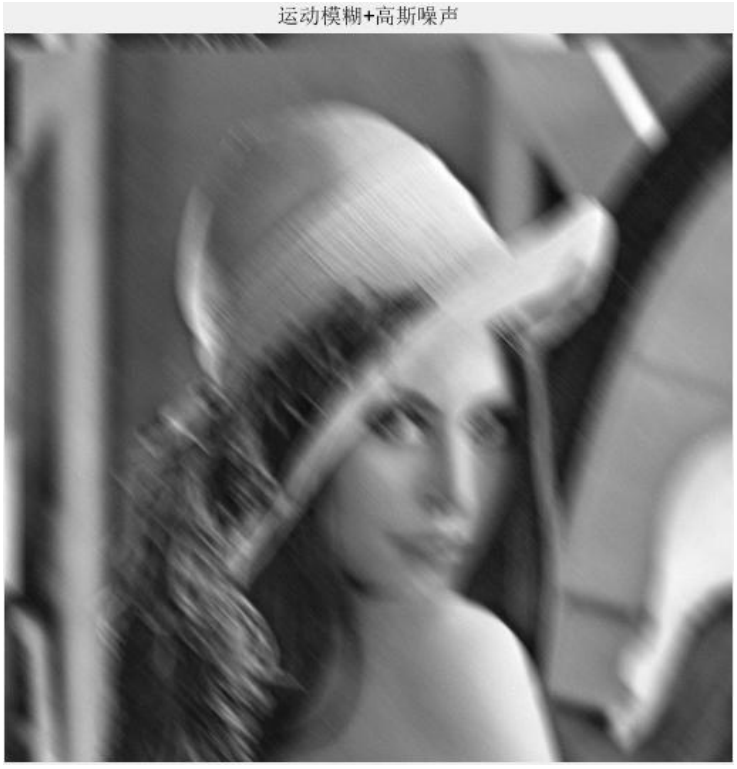
$$p = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

3.实验结果

1．模糊 lena 图像：45 度方向，T=1



2．在模糊的 lena 图像中增加高斯噪声，均值= 0 ，方差 0.01 以产生模糊图像。



3．分别利用方程 Eq. (5.8-6)和(5.9-4)，恢复图像；并分析算法的优缺点。

由以上处理可知，维纳滤波和约束最小二乘方滤波都可以做到消除一部分运动模糊和椒盐噪声，但是两者作用有一些差别，其中维纳滤波在消除噪声方面效果更好，约束最小方乘方滤波在消除运动模糊方面效果更好。

六、源代码

1.添加高斯噪声

```
I = imread('C:\Users\Administrator\Desktop\大三下\第六次作业
\lena.bmp');
[m,n]=size(I);
figure(1)
subplot(1,2,1);
imshow(I)
title('原图像');
e=0;d=0.08;
u1=rand(m,n);
u2=rand(m,n);
x=d*sqrt(-2*log(u1)).*cos(2*pi*u2)+e;
p=double(I)/255+x;
p=uint8(255*p);
subplot(1,2,2);
imshow(p);
```

```
title('加高斯噪声后');
```

2.中点滤波，算术均值滤波，几何均值滤波

```
n = 3;
template = ones(n);
[height, width] = size(p);
figure(1)
subplot(1,2,1);
imshow(p);
title('有高斯噪声图像');
x1 = double(p);
x2 = x1;
for i = 1:height-n+1
    for j = 1:width-n+1
        c = x1(i:i+n-1,j:j+n-1).*template;
        s = sum(sum(c));
        x2(i+(n-1)/2,j+(n-1)/2) = s/(n*n);
    end
end

g = uint8(x2);
subplot(1,2,2);
imshow(g);
title('算术均值滤波');

template = ones(n);
n=3;
[height, width] = size(p);
figure(1)
subplot(1,2,1);
imshow(p);
title('有高斯噪声图像');
x1 = double(p);
x2 = x1;
for i = 1:height-n+1
    for j = 1:width-n+1
        c = x1(i:i+n-1,j:j+n-1).*template;
```

```
        s = prod(prod(c));
        x2(i+(n-1)/2,j+(n-1)/2) = nthroot(s,n*n);
    end
end
g = uint8(x2);
subplot(1,2,2);
imshow(g);
title('几何算术均值滤波');

[height, width] = size(p);
n=3;
figure(1)
subplot(1,2,1);
imshow(p);
title('有高斯噪声图像');
x1 = double(p);
x2 = x1;
for i = 1:height-n+1
    for j = 1:width-n+1
        c = x1(i:i+n-1,j:j+n-1);
        e = c(1,:);
        for k = 2:n
            e = [e, c(k, :)];
        end
        tmp = 1/2*(max(e)+min(e));
        x2(i+(n-1)/2,j+(n-1)/2) = tmp;
    end
end
g = uint8(x2);
 subplot(1,2,2);
imshow(g);
title('中点滤波');
```

3.逆谐波均值滤波器消除椒盐噪音

```
[m,n] = size(I);
jiaoyan=I;
k1=0.1;
```

```
k2=0.9;
a1=rand(m,n)<k1;
a2=rand(m,n)>k2;
jiaoyan(a1)=0;
jiaoyan(a2)=255;
Q = -1;
J = wextend('2D','sym',jiaoyan,1);
G = zeros(m,n);
for i = 2:(m+1)
    for j = 2:(n+1)
        o = double(J(i-1:i+1,j-1:j+1));
        G(i-1,j-1) = (sum(sum(o.^(Q+1))))/(sum(sum(o.^Q)));
    end
end
G = uint8(G);
figure(1);
subplot(1,2,1);
imshow(jiaoyan);
title('椒盐均为 0.1 图像')
subplot(1,2,2);
imshow(G);
title('Q=-1;逆谐波均值滤波器')
```

4.最大值，最小值，中值滤波

```
[height,width] = size(I);
jiaoyan=I;
k1=0.1;
k2=0.9;
a1=rand(height,width)<k1;
a2=rand(height,width)>k2;
jiaoyan(a1)=0;
jiaoyan(a2)=255;
figure(1)
subplot(1,2,1);
imshow(jiaoyan);
title('椒盐均为 0.1 图像');
x1 = double(jiaoyan);
```

```
x2 = x1;
n=3;
for i = 1:height-n+1
    for j = 1:width-n+1
        c = x1(i:i+n-1,j:j+n-1);
        e = c(1,:);
        for k = 2:n
            e = [e, c(k, :)];
        end
        tmp = max(e);
        x2(i+(n-1)/2,j+(n-1)/2) = tmp;
    end
end
g = uint8(x2);
subplot(1,2,2);
imshow(g);
title('最大值滤波');

[height,width] = size(I);
jiaoyan=I;
k1=0.1;
k2=0.9;
a1=rand(height,width)<k1;
a2=rand(height,width)>k2;
jiaoyan(a1)=0;
jiaoyan(a2)=255;
figure(1)
subplot(1,2,1);
imshow(jiaoyan);
title('椒盐均为 0.1 图像');
x1 = double(jiaoyan);
x2 = x1;
n=3;
for i = 1:height-n+1
    for j = 1:width-n+1
        c = x1(i:i+n-1,j:j+n-1);
        e = c(1,:);
```

```
        for k = 2:n
            e = [e, c(k, :)];
        end
        tmp = min(e);
        x2(i+(n-1)/2,j+(n-1)/2) = tmp;
    end
end
g = uint8(x2);
 subplot(1,2,2);
imshow(g);
title('最小值滤波');

[height,width] = size(I);
jiaoyan=I;
k1=0.1;
k2=0.9;
a1=rand(height,width)<k1;
a2=rand(height,width)>k2;
jiaoyan(a1)=0;
jiaoyan(a2)=255;
figure(1)
subplot(1,2,1);
imshow(jiaoyan);
title('椒盐均为 0.1 图像');
x1 = double(jiaoyan);
x2 = x1;
for i = 1:height-n+1
    for j = 1:width-n+1
        c = x1(i:i+n-1,j:j+n-1);
        e = c(1,:);
        for k = 2:n
            e = [e, c(k, :)];
        end
        tmp =median(e);
        x2(i+(n-1)/2,j+(n-1)/2) = tmp;
    end
end
```

```
g = uint8(x2);
 subplot(1,2,2);
imshow(g);
title('中值滤波');
```

5.约束及非约束维纳滤波

```
MU = 0;
SIGMA = 10/(255*255);
I = imread('lena.bmp');
[m,n] = size(I);
M = 2*m;
N = 2*n;
a = 0.03;
b = 0.03;
T = 1;
K = 0.04;
gama = 0.001;
f = double(I);
F = fft2(f);
F = fftshift(F);
for u = 1:m
    for v = 1:n
        if (u-m/2)*a+(v-n/2)*b~=0
            H(u,v) =
T/(pi*((u-m/2)*a+(v-n/2)*b))*sin(pi*((u-m/2)*a+(v-n/2)*b))*exp(-sqrt(
-1)*pi*((u-m/2)*a+(v-n/2)*b));
        else
            H(u,v) = 1;
        end
        G(u,v) = F(u,v)*H(u,v);
    end
end
g = ifftshift(G);
g = ifft2(g);
g = 256.*g./(max(max(g)));
g = uint8(real(g));
figure(1)
```

```
imshow(g);
title('运动模糊');
R0 = normrnd(MU,sqrt(SIGMA),m,n);
R = uint8(R0*255);
Inoise0 = R+g;
Inoise = zeros(m,n);
for i = 1:m
    for j = 1:n
        if Inoise0(i,j)>=0&&Inoise0(i,j)<=255
            Inoise(i,j) = Inoise0(i,j);
        elseif Inoise0(i,j)<0
            Inoise(i,j) = 0;
        elseif Inoise0(i,j)>255
            Inoise(i,j) = 255;
        end
    end
end
Inoise = uint8(Inoise);
figure(2)
imshow(Inoise);
title('运动模糊+高斯噪声');
If = double(Inoise);
If = fft2(If);
If = fftshift(If);
for u = 1:m
    for v = 1:n
        F1(u,v) = (abs(H(u,v))^2/(H(u,v)*(abs(H(u,v))^2+K)))*If(u,v);
    end
end
f1 = ifftshift(F1);
f1 = ifft2(f1);
f1 = 255.*f1./(max(max(f1)));
f1 = uint8(real(f1));
figure(3)
imshow(f1);
title('无约束维纳滤波');
p = [0 -1 0;-1 4 -1;0 -1 0];
```

```
P = fft2(p,m,n);
P = fftshift(P);
for u = 1:m
    for v = 1:n
        F2(u,v) =
conj(H(u,v))*If(u,v)/(abs(H(u,v))^2+gama*abs(P(u,v))^2);
    end
end
f2 = ifftshift(F2);
f2 = ifft2(f2);
f2 = 255.*f2./(max(max(f2)));
f2 = uint8(real(f2));
figure(4);
imshow(f2);
```