# 滤波器实现

## 一、添加噪声

主要的噪声有高斯噪声、瑞利噪声、伽马噪声、指数噪声、均匀噪声、椒盐噪声。给一个图像加噪声主要使用matlab中的im2double函数对图像作归一化处理，然后根据相应的噪声分布函数生成噪声图像，把噪声图像加到原始图像中就可以了。高斯噪声和椒盐噪声的添加可以使用imnoise函数完成。

img= im2double(rgb2gray(imread('pic.jpg')));%im2double作归一化处理

sigma=25;%添加噪声的标准差

J\_Gaussian=imnoise(img,'gaussian',0,(sigma/255)^2); %添加均值为0，标准差为sigma高斯噪声

imwrite(J\_Gaussian,'gaussian.png');%输出高斯噪声图

J\_salt=imnoise(img,'salt & pepper',0.07); %添加7%的椒盐噪声

imwrite(J\_salt,'salt&pepper.png');

%生成瑞利噪声

[M,N]=size(img);

a=0;

b=0.08;

B=1;

N\_Ray1=a+b\*raylrnd(B,M,N);

%将噪声叠加到图像上

J\_rayl=img+N\_Ray1;

imwrite(J\_rayl,'rayl.png');

% 伽马噪声

a=0;

b=0.08;

A=1;

B=2;

N\_Gam=a+b\*gamrnd(A,B,[M,N]);

J\_Gamma=img+N\_Gam;

imwrite(J\_Gamma,'gamma.png');

% 指数噪声

a=0;

b=0.08;

mu=2;

N\_exp=a+b\*exprnd(mu,[M,N]);

J\_exp=img+N\_exp;

imwrite(J\_exp,'exp.png');

% 均匀分布噪声

a=0;

b=0.08;

A=0;

B=2;

N\_unif=a+b\*unifrnd(A,B,[M,N]);

J\_unif=img+N\_unif;

imwrite(J\_unif,'uniform.png');

subplot(2,3,1);

imshow(J\_Gaussian);

title('高斯噪声');

subplot(2,3,2);

imshow(J\_rayl,[]);

title('瑞利噪声');

subplot(2,3,3);

imshow(J\_Gamma,[]);

title('伽马噪声');

subplot(2,3,4);

imshow(J\_exp,[]);

title('指数噪声');

subplot(2,3,5);

imshow(J\_unif,[]);

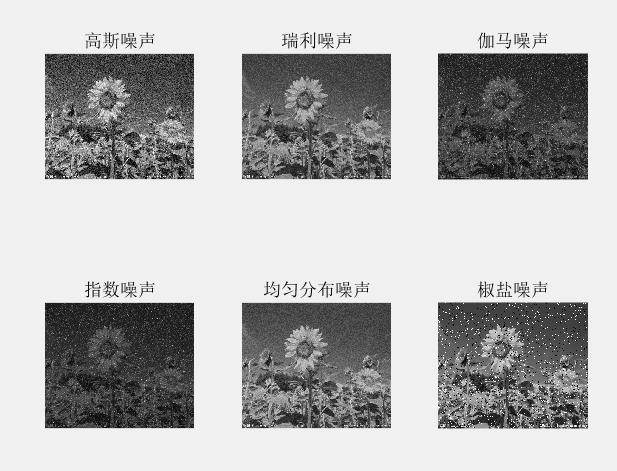
title('均匀分布噪声');

subplot(2,3,6);

imshow(J\_salt,[]);

title('椒盐噪声');

最终效果：



同时，对上述代码进行如下修改，可以实现给彩色图像加噪声

img= im2double(imread('pic.jpg'));%im2double作归一化处理

[M,N,count]=size(img);



## 二、滤波器的实现

### 2.1均值滤波器

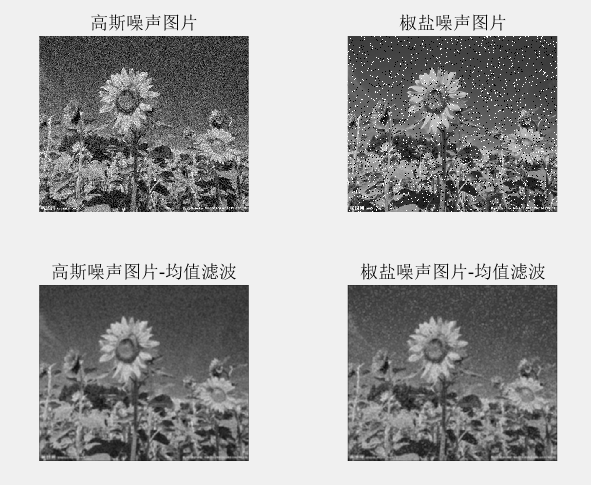
#### 2.1.1算术均值滤波器

算数均值滤波器是对周围的m\*n的像素点求平均值，作为这一点的输出，实现过程中定义了3\*3的滤波盒大小，以下是实现代码，为了清楚的描述算数均值滤波器的matlab实现，我只在文档中提供关键代码，首先，使用fspecial创建均值滤波器的算子，然后使用imfilter对图像进行滤波。Fdata就是滤波后的图像。

h=fspecial('average',5);%创建一个均值(5\*5)

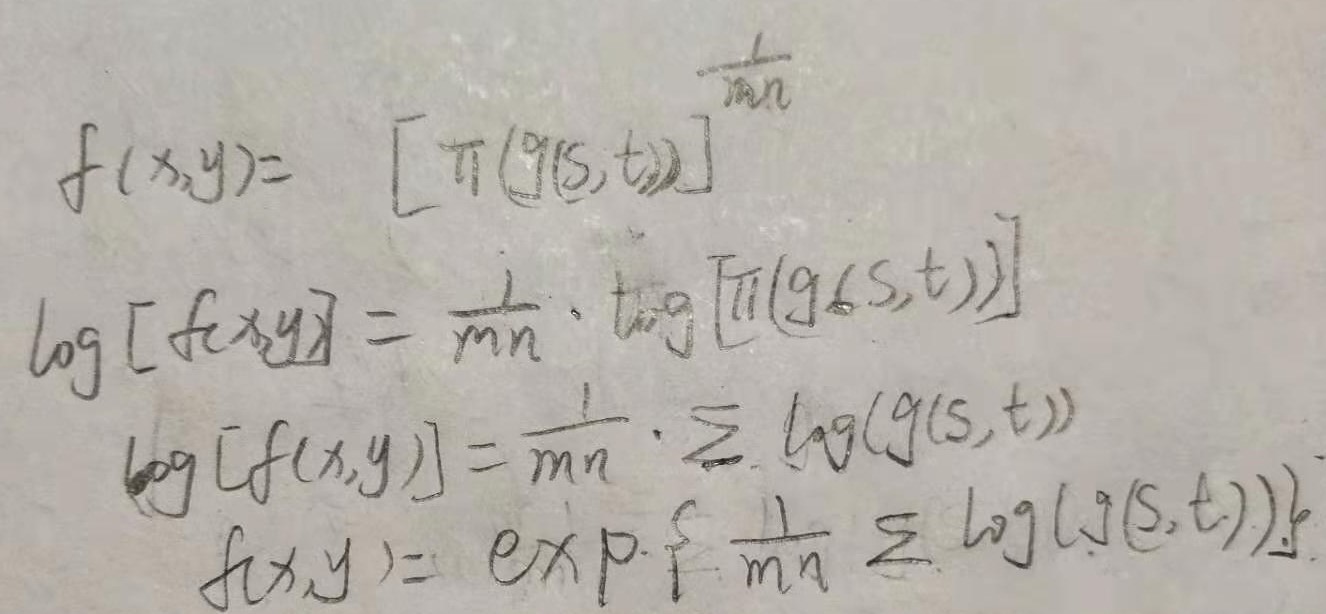
fdata=imfilter(gaussian,h);%前面是图片，后面是模板

以下是对加过高斯噪声、椒盐噪声的图片进行均值滤波之后得到的效果图片。详细代码在test02.m中。可以看到图像在滤波之后变得模糊了许多，但是也起到了不错的去噪效果，然而，考虑到这个去噪效果是以牺牲清晰度为代价的，在实际应用中，不常用这种方式进行滤波。



#### 2.1.2几何均值滤波器

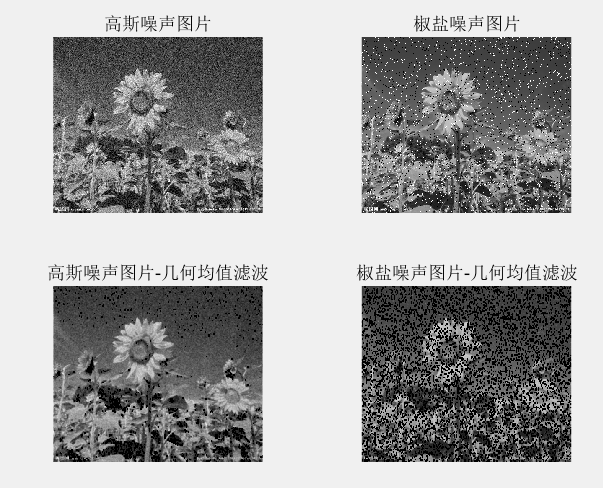
基于以上对算数均值滤波器的描述，对几何均值滤波器的公式进行如下变换，可以很容易根据算数均值滤波器实现几何均值滤波器。



因此，几何均值滤波器实现代码如下：

Result=exp(imfilter(log(img),fspecial('average',5)))；

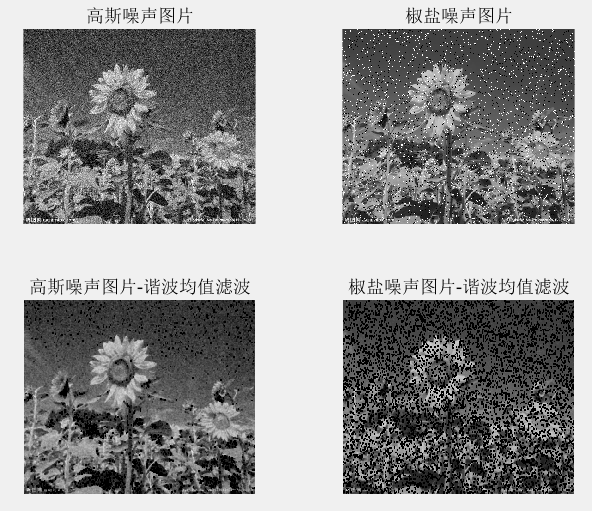
Result是滤波之后的图像，对原始img进行log操作之后采用均值滤波之后，再对结果进行exp操作，最后得到的结果就是几何均值滤波器的结果。如下是几何均值滤波器对椒盐噪声、高斯噪声的效果，图像相比算数均值滤波器清晰了许多，但是图像莫名多出了许多的黑点，可见效果不是很好。（详细代码在test03.m）



#### 2.1.3谐波均值滤波器

基于谐波滤波器的公式，可以根据算数均值滤波器变换实现该滤波器，详细代码如下：

Result=1./(imfilter(1./img,fspecial('average',5)))；下图是谐波滤波器对高斯噪声、椒盐噪声的去除前后效果对比，可以看到，它对于“盐” 噪声效果较好，但不适于“胡椒”噪声。它善于处理高斯噪声。代码在（test04.m）



#### 2.1.4逆谐波均值滤波器

这种滤波器适合减少或是在实际中消除椒盐噪声的影响。当*Q*值为正数时，滤波器用于消除“胡椒”噪声;当*Q*值为负数时，滤波器用于消除“盐”噪声。逆谐波的实现与前面的滤波器有些不同，但是还是可以用算数均值滤波器实现，但是需要两个算数均值滤波器，先对图像中的每个元素求Q+1次幂，然后求算数均值滤波器，得到一个图像矩阵，然后对原图像中每个元素求Q次幂，再求算数均值滤波器，得到一个图像矩阵，把两个对应元素相除之后，得到的图像矩阵就是逆谐波均值滤波器之后的效果，其主要代码如下，其中-1是Q+1,-2是Q。

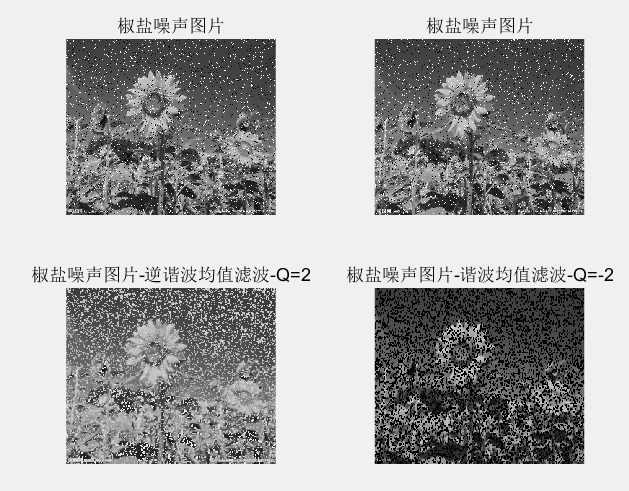
h=fspecial('average',3);%创建一个均值(5\*5)

fdata1=imfilter(salt.^-1,h);%前面是图片，后面是模板

fdata2=imfilter(salt.^-2,h);

fdata=fdata1./fdata2;

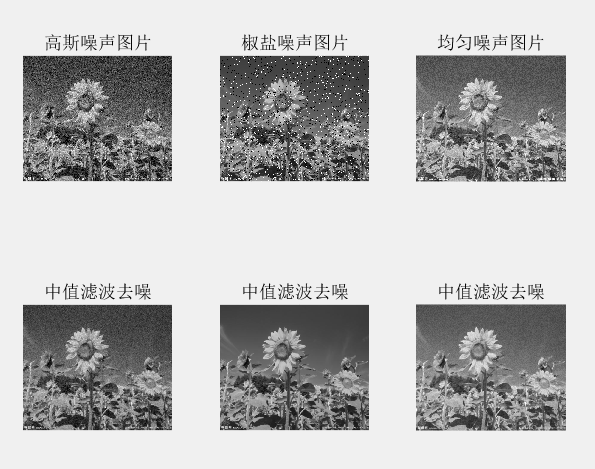
使用Q=-2和Q=2对椒盐噪声图像处理，对比结果可以看到，Q=2(正值)的时候，图像中黑色的胡椒噪声看不见了，但是盐噪声还在，而Q=-2的时候，正好相反，图像中的椒噪声很明显，基本没有去除，但是盐噪声已经被去除了。详细代码（test05.m）



### 2.2统计排序滤波器

#### 2.2.1中值滤波器

中值滤波器是取m\*n区域内的中间值，对于其实现可以用matlab中的函数medfilt2(img,[m n]),第一个参数是图像矩阵，第二个参数是m\*n的滤波盒，如下是中值滤波器对高斯噪声、椒盐噪声、均匀分布噪声的去噪效果，可以看到它对这三种噪声都有较好的去噪效果。详细代码（test06.png）,它的去噪效果可以通过其设计思想解释：因为噪声的出现，使该点像素比周围的像素亮(暗)许多，取中值一般不会取到噪声。



#### 2.2.2最大值和最小值滤波器

最大值和最小值滤波器是取周围像素点最大值或最小值，可以使用ordfilt2(X,order,domain)函数实现，这个函数对domain内的值进行排序，然后输出第order个值作为该点的值，排序是从小到大排序的，所以如果是3\*3的最小值滤波器，就用ordfilt2(X,1,ones(3,3)),如果是最大值滤波器就使用ordfilt2(X,9,ones(3,3))。先对高斯噪声、椒盐噪声、均匀分布噪声的图像进行最小值滤波，详细代码在test07.png中，以下是对比图



从图中可以观察到最小值滤波器使图像变得更暗了，它可以很好的去除了盐噪声，无法去除椒噪声。接下来观察最大值滤波器的效果（详细代码在test08.m）



从结果中可以看出最大值滤波器对盐噪声基本没有作用，反而扩大的盐噪声，而它很好的去除了椒噪声，它使图像整体看上去暗了许多，最大值和最小值滤波器对于去除特定的椒噪声和盐噪声效果比较好，对于别的噪声没有中值滤波器的效果好。

#### 2.2.3中点滤波器

中点滤波器是对m\*n的滤波盒中最大值和最小值取平均，对于3\*3的滤波盒，实现只需要(ordfilt2(X,9,ones(3,3))+ ordfilt2(X,1,ones(3,3)))/2就可以了，如下是高斯噪声、椒盐噪声、均匀分布噪声的图像用中点滤波器前后对比效果图（详细代码在test09.m）：



可以看到中点滤波器对高斯噪声、均匀分布噪声有较好的去除效果，但是，它对椒盐噪声效果不是很好，这是因为取最大值的时候很可能取到椒噪声，取最小值很可能取到盐噪声，那么结果就是噪声。

#### 2.2.4修正后的阿尔法均值滤波器

这个滤波器先对m\*n大小的像素点进行了排序，排序之后去除最小的d/2个灰度值和最大的d/2个灰度值，然后求平均值。其关键代码如下，先从图像(I,j)处去除一个len\_m,len\_n大小的矩阵，使用sort函数进行排序，然后取出中间的m\*n-d个数，最后求平均值，得到最终的像素值。

Block=I\_D\_pad(i-len\_m:i+len\_m,j-len\_n:j+len\_n);

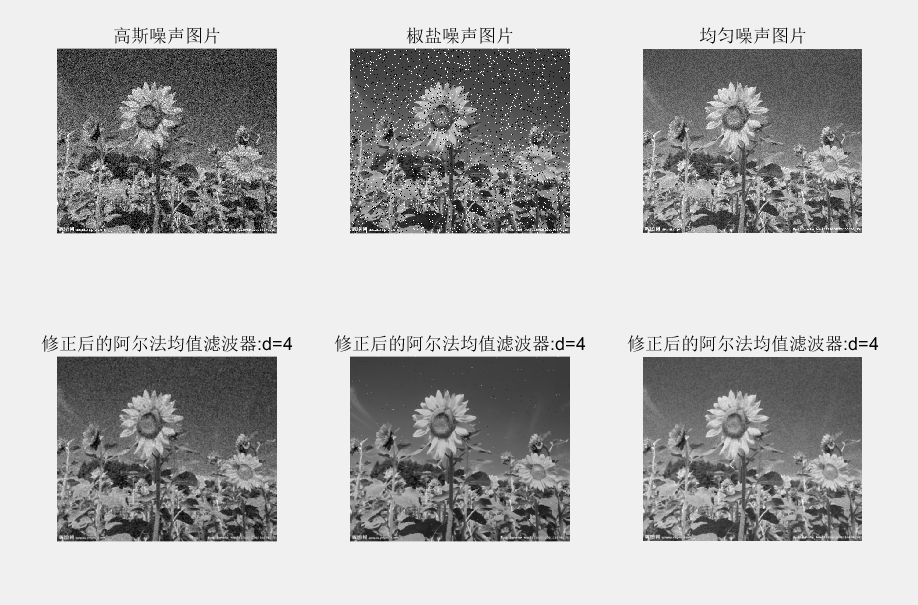
Block=sort(Block(:));

Block=Block(floor(d/2+1):floor(m\*n-d/2));

%计算矩阵的阿尔法均值

J\_Alpha(i-len\_m,j-len\_n)=sum(sum(Block))/(m\*n-d);

以下是修正后的阿尔法均值滤波器对高斯噪声、均匀分布噪声、以及椒盐噪声去除效果对比图，可以看到这种滤波器对这三种噪声都有较好的去除效果。代码在test10.m中



接下来观察这种滤波器对混合噪声的去除效果，它对高斯噪声和椒盐噪声的去除效果还是比较好的。（详细代码在test11.m）



### 2.3自适应滤波器

#### 2.3.1自适应局部降低噪声滤波器

自适应滤波器的实现过程中要涉及到的量有以下几个：原图像的像素值，滤波盒内各个像素值的局部方差，噪声的全局方差，这个值需要估计，我在这里设置它的值跟高斯噪声的方差相等，希望得到最好的去噪效果，滤波盒的均值。先取出图像滤波盒大小的矩阵，然后求局部方差，再求他们的平均值，最后根据公式求出该像素点的像素值，Eta\_total代表的是整体噪声的方差。

Block=I\_D\_pad(i-len\_m:i+len\_m,j-len\_n:j+len\_n);

Eta\_local=std2(Block).^2; %求局部方差

average=mean(mean(Block)); %求平均值

J\_Alpha(i-len\_m,j-len\_n)=t1(i-len\_m,j-len\_n)-(Eta\_total/Eta\_local)\*(t1(i-len\_m,j-len\_n)-average);

下面是对高斯噪声的图像去噪的结果。（详细代码在test12.m）



从结果中可以看到它有比较好的去噪效果。

#### 2.3.2 自适应中值滤波器

自适应中值滤波器算法工作在两个层次，定义为A层和B层，

A层：

A1=*z*med-*z*min

A2=*z*med-*z*max

如果A1>0且A2<0，转到B层

否则增大窗口尺寸

如果窗口尺寸≤*S*max重复A层，否则输出*z*xy

B层：

B1=*z*xy-*z*min

A2=*z*xy-*z*max

如果B1>0且B2<0，输出*z*xy重复A层，否则输出*z*med

通过上述算法描述实现滤波器，设置每次滤波盒迭代，每次找到满足条件A和B的元素位置，设置为原来的像素，满足A不满足B的设置为中值，直到所有的像素值被处理过。

function II=adp\_median(I,Smax)

%自适应中值滤波

if (Smax<=1)||(Smax/2==round(Smax/2))||(Smax~=round(Smax))%Smax约束，噪声密度越大，Smax应当设置越高

error('Smax must be an odd integer >1');

end

%初始化

II=I;

II(:)=0;

alreadyProcessed=false(size(I));

%迭代

for k=3:2:Smax

zmin=ordfilt2(I,1,ones(k,k),'symmetric');

zmax=ordfilt2(I,k\*k,ones(k,k),'symmetric');

zmed=medfilt2(I,[k k],'symmetric');

processUsingLevelB=(zmed>zmin)&(zmax>zmed)&(~alreadyProcessed);%需要转到B步骤的像素

zB=(I>zmin)&(zmax>I);

outputZxy=processUsingLevelB&zB;%满足步骤A，B的输出原值 对应的像素位置

outputZmed=processUsingLevelB&~zB;%满足A,不满足B的输出中值 对应的像素位置

II(outputZxy)=I(outputZxy);

II(outputZmed)=zmed(outputZmed);

alreadyProcessed=alreadyProcessed|processUsingLevelB;%处理过的像素

if all(alreadyProcessed(:))

break;

end

end

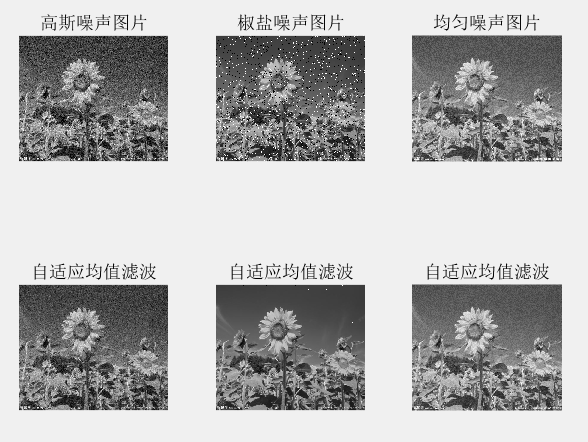
II(~alreadyProcessed)=I(~alreadyProcessed);%超过窗口大小没被处理的像素位置 输出原值

下面是自适应均值滤波器对高斯、椒盐、均匀噪声去除效果对比图

Smax=7



Smax=9



从图中可以看到它对椒盐噪声有比较好的去除效果，保留了图像的尖锐部分。通过对Smax设置不同的值，可以观察到，当Smax越大的时候，去除效果就会越好。

## 三、总结

通过对以上滤波器的实现，了解了空间滤波的相关应用，对不同滤波器的去除噪声的效果有了更加直观的认识。