《数字图像处理》
第7讲 图像恢复

冯建江

清华大学 自动化系

2017.11.9

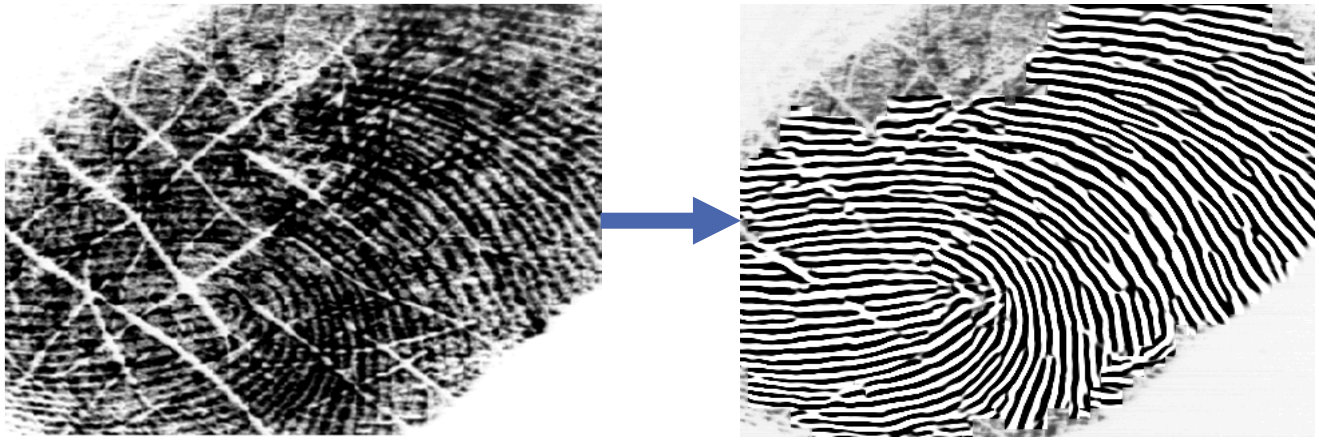
内 容

- 图像退化模型
- 只有噪声时的图像恢复
  - 周期性噪声
  - 随机噪声
- 考虑退化函数的图像恢复
  - 逆滤波
  - 维纳滤波

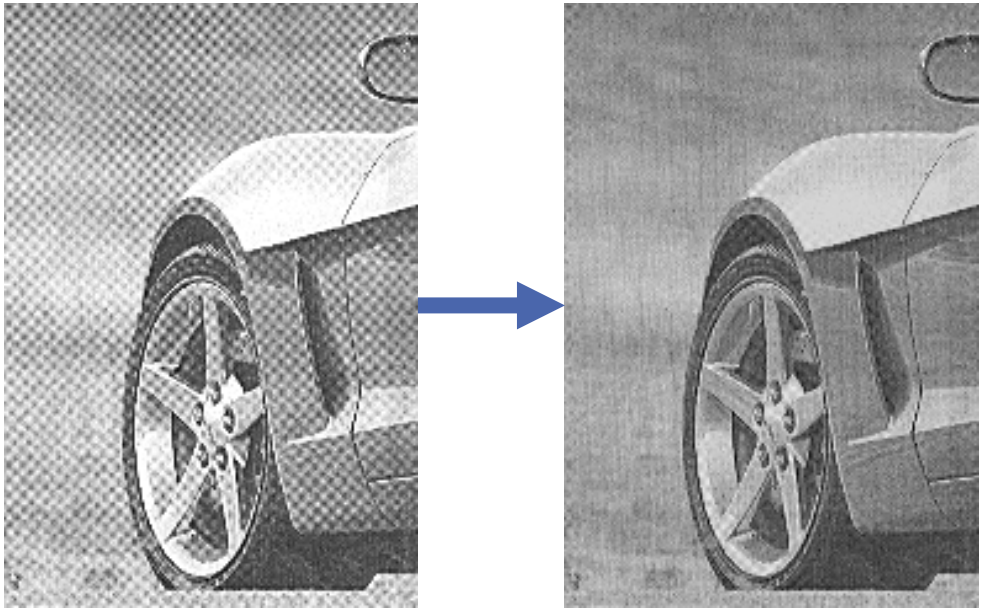
图像增强与图像恢复

- 图像增强：改善图像质量，便于人观看或自动分析
- 图像恢复：认为图像在某种情况下退化了，根据相应的退化模型和知识，恢复原始图像
- 不同之处
  - 增强带有主观性，评价指标是人主观评价或后续算法性能
  - 恢复比较客观，评价标准是与原始图像的差异
- 相似之处
  - 可以改善图像质量
  - 有些滤波方法是相同的，例如空域滤波（均值滤波、中值滤波）、频域滤波（选择性滤波）
- 增强的含义更泛，可以包含恢复

图像增强与图像恢复



- 处理的目的不是为了得到指纹本来的样子
- 归为增强



- 处理的目的是为了得到照片本来的样子
- 归为恢复，更为准确

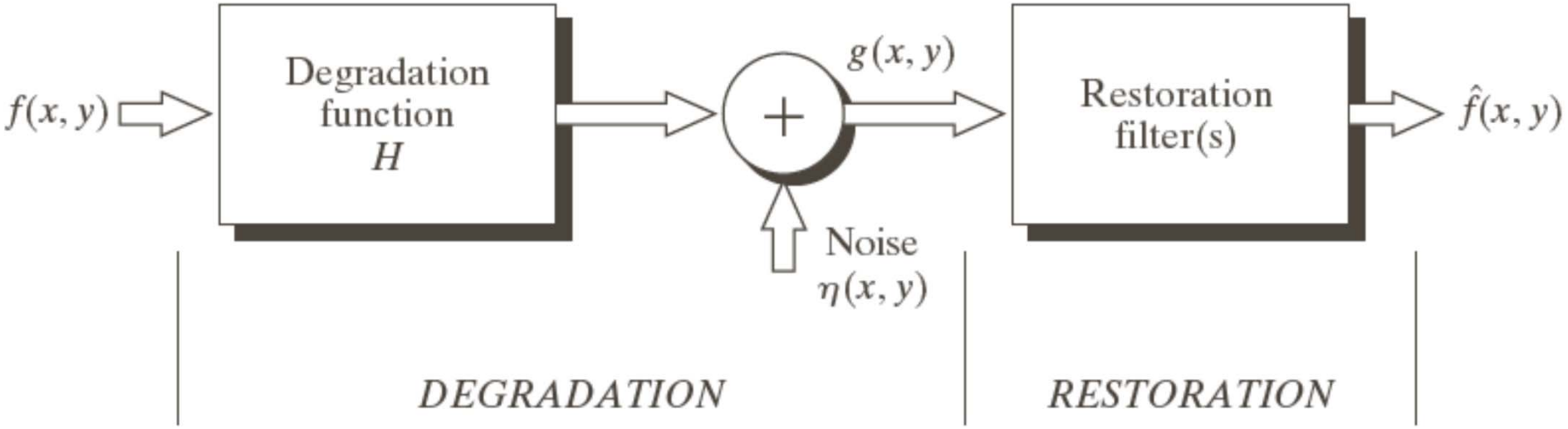
图像退化模型（Degradation Model）

为实际的图像退化建立一个简单的图像退化模型。

图像退化模型：退化系统H和加性噪声$\eta(x,y)$联合作用到原始图像$f(x,y)$上，得到退化图像$g(x,y)$。

$$g(x,y) = H[f(x,y)] + \eta(x,y)$$

图像恢复：已知退化图像$g(x,y)$，关于退化系统和噪声的某些知识，估计$\hat{f}(x,y)$，使其尽可能接近$f(x,y)$



图像退化模型

- 我们只考虑线性、位置不变的退化系统。
- 这类退化系统的空域模型为：

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

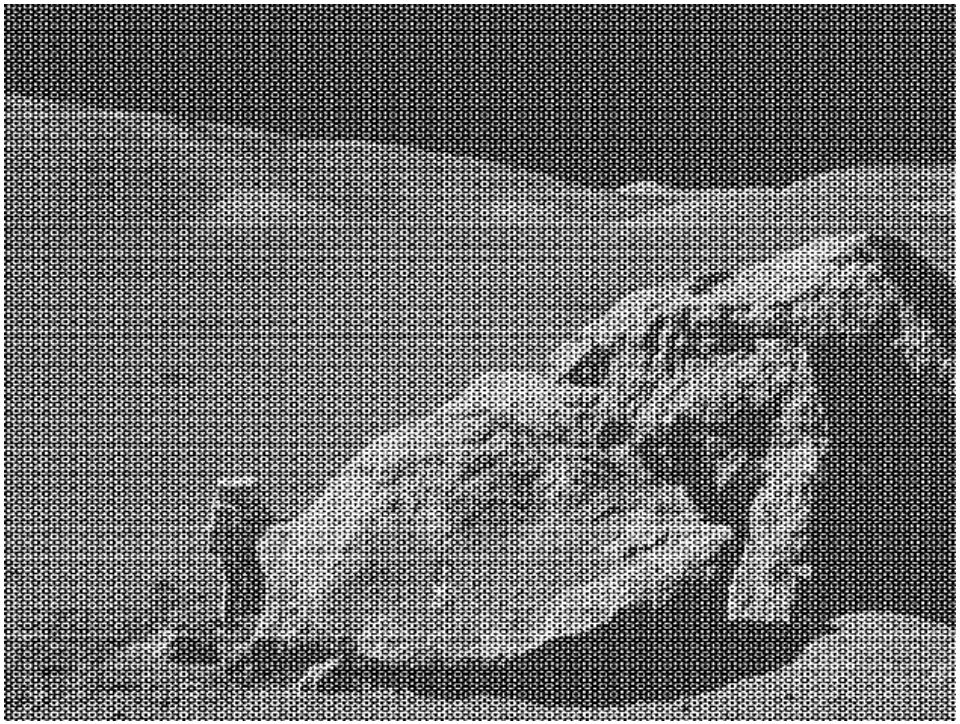
- 按照卷积定理，其频域表示为：

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

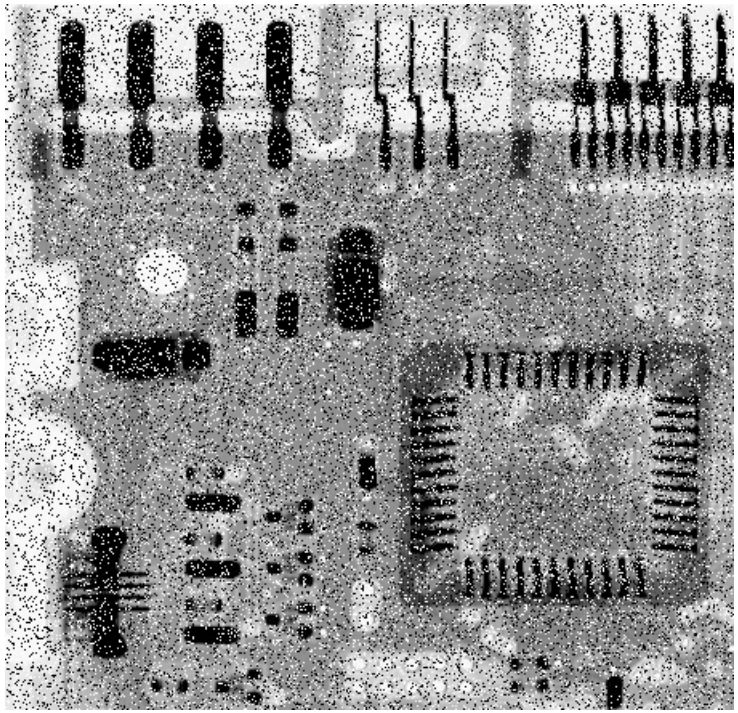
- 下面，先考虑只有噪声的情况，即$H(u,v)$为1
- 之后再考虑$H(u,v)$不为1、且有噪声的情况

噪声

- 图像成像和传输过程中都有可能出现噪声
- 我们研究两种噪声：周期性噪声和随机噪声
- 去除周期性噪声：利用选择性滤波
- 去除随机噪声：分析其统计特性，利用相应去除方法



周期性噪声



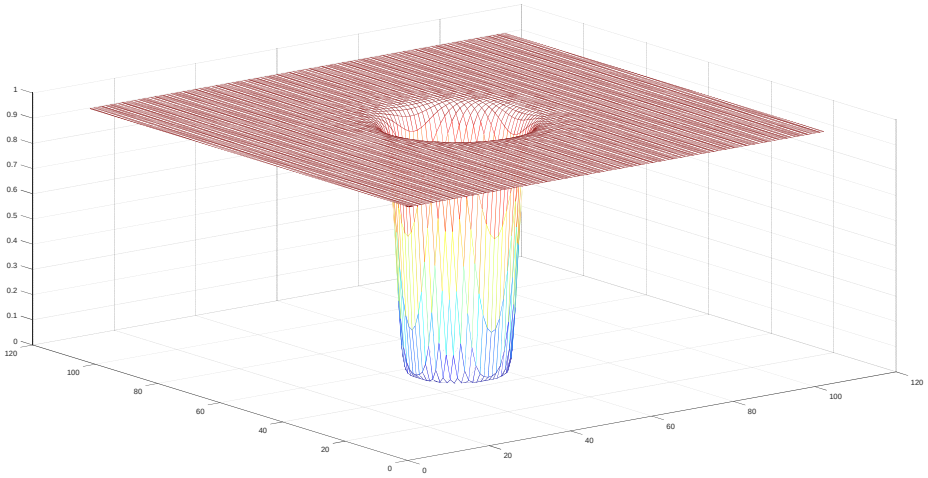
随机噪声

内 容

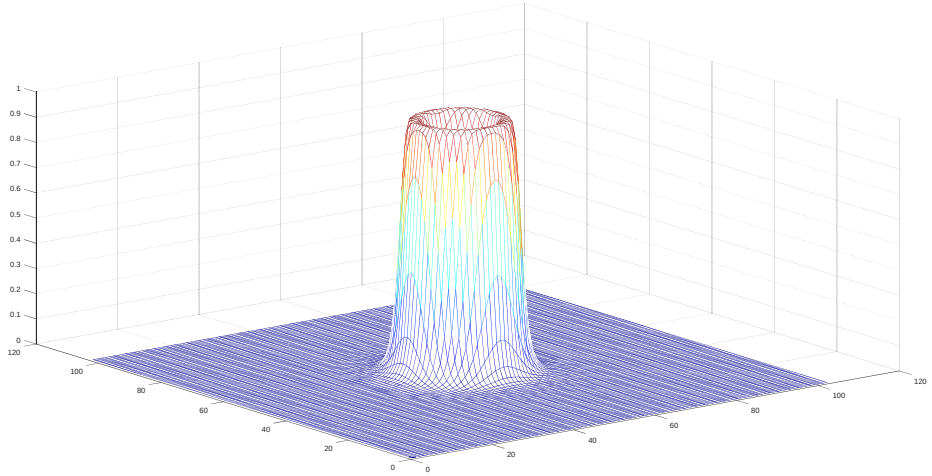
- 图像退化模型
- 只有噪声时的图像恢复
  - 周期性噪声
  - 随机噪声
- 考虑退化函数的图像恢复
  - 逆滤波
  - 维纳滤波

周期性噪声的去除

- 我们在频域对周期性噪声进行分析，设计选择性滤波器，将其去除
- 选择性滤波器包括上节课介绍的带通、带阻、陷波滤波器



带阻



带通

陷波滤波器（notch filter）

- 陷波滤波器阻断（或通过）指定频率附近的频率成分。
- 陷波阻断滤波器由成对的高通滤波器组成

$$H_{\mathrm{NR}}(u,v)=\prod_{k=1}^{Q}H_k(u,v)H_{-k}(u,v)$$

$H_k(u,v)$和$H_{-k}(u,v)$分别为中心位于(u_k,v_k)和$(-u_k,-v_k)$的高通滤波器。成对是考虑到图像的DFT是共轭对称的。

- 例如，巴特沃斯陷波阻断滤波器（notch reject filter）：

$$H_{\mathrm{NR}}(u,v)=\prod_{k=1}^{3}\frac{1}{1+[D_{0k}/D_k(u,v)]^{2n}}\frac{1}{1+[D_{0k}/D_{-k}(u,v)]^{2n}}$$

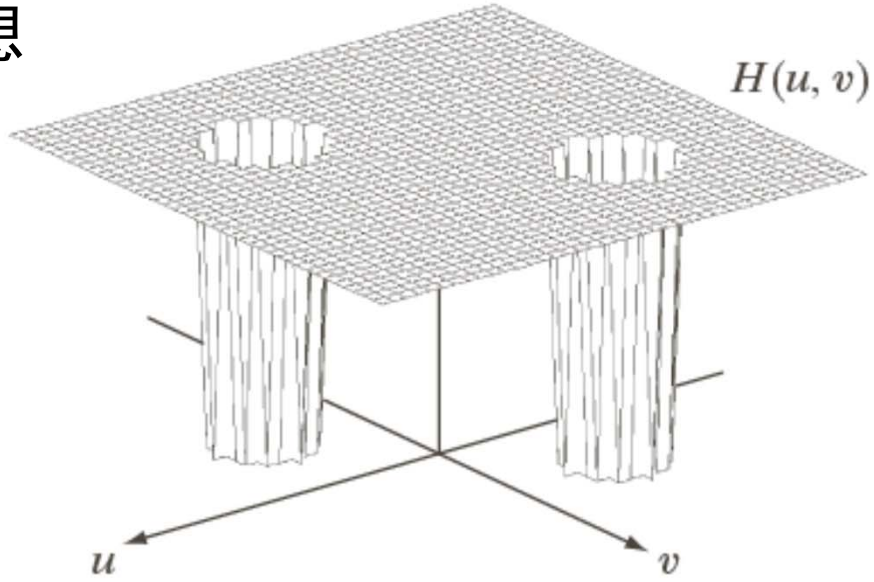
$$D_k(u,v)=\sqrt{(u-P/2-u_k)^2+(v-Q/2-v_k)^2}$$

$$D_{-k}(u,v)=\sqrt{(u-P/2+u_k)^2+(v-Q/2+v_k)^2}$$

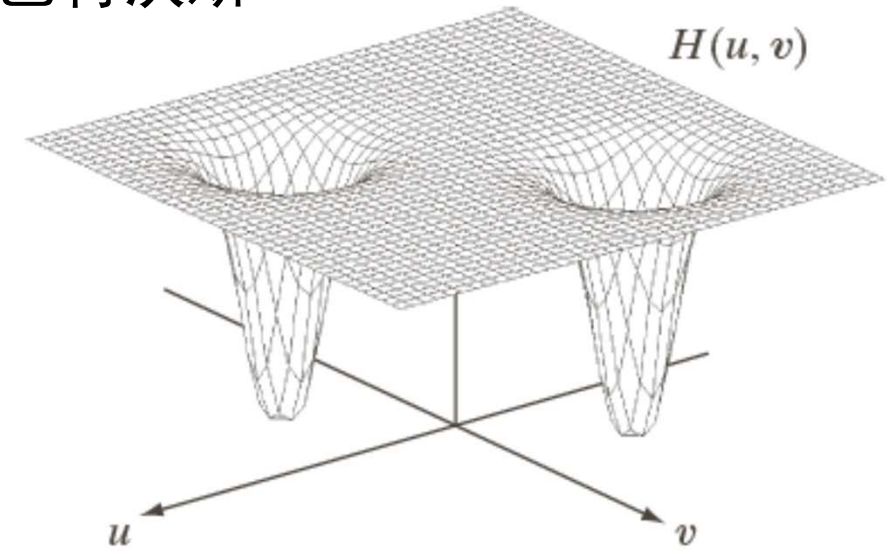
$$H_{\mathrm{NP}}(u,v)=1-H_{\mathrm{NR}}(u,v)$$

3种陷波阻断滤波器

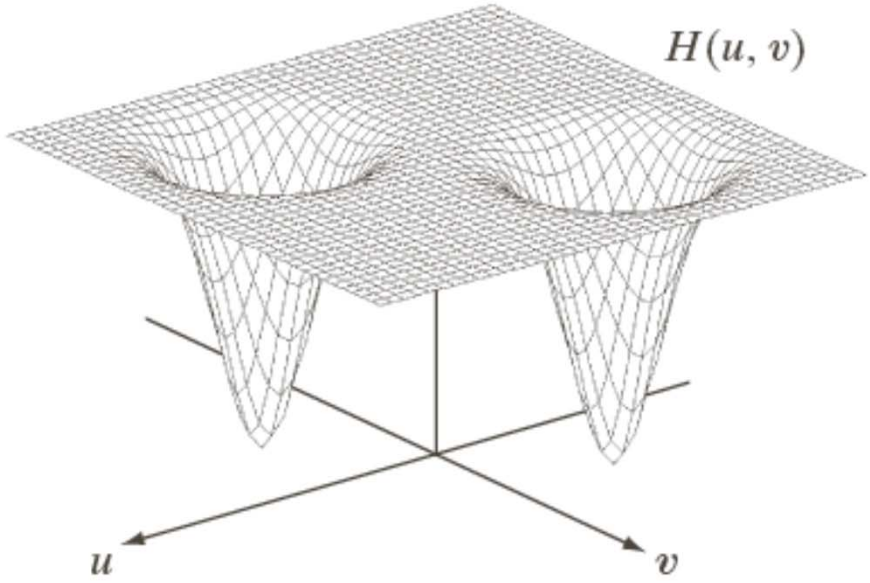
理想



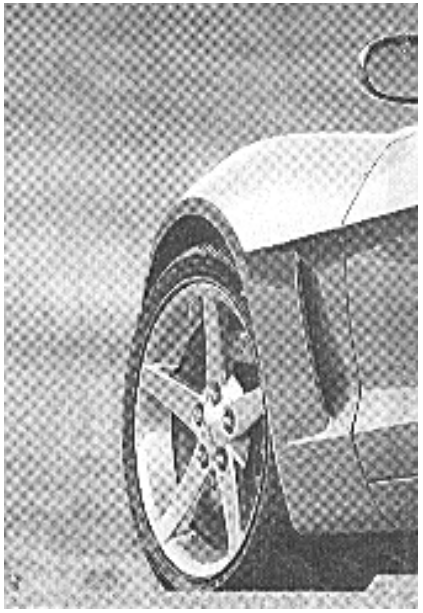
巴特沃斯

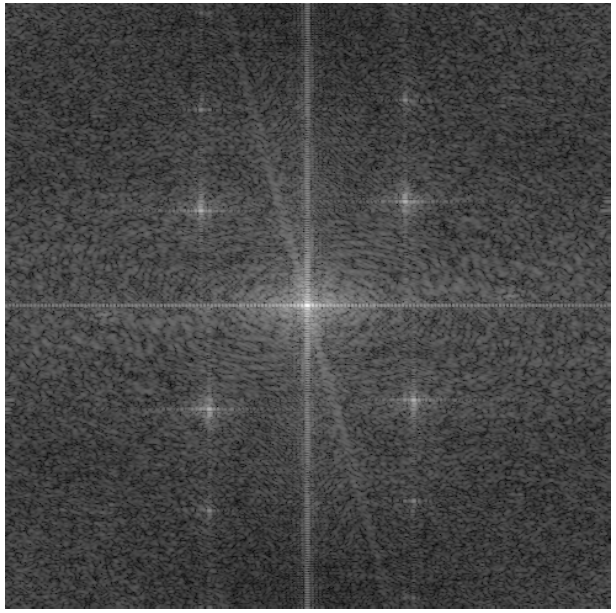


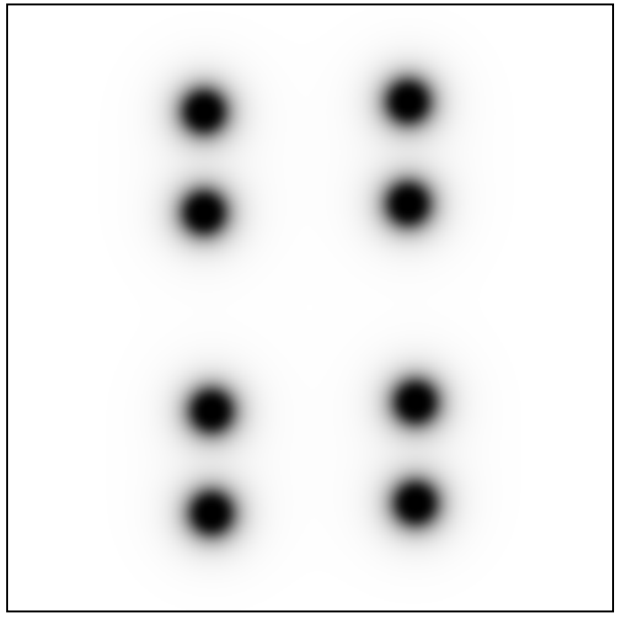
高斯

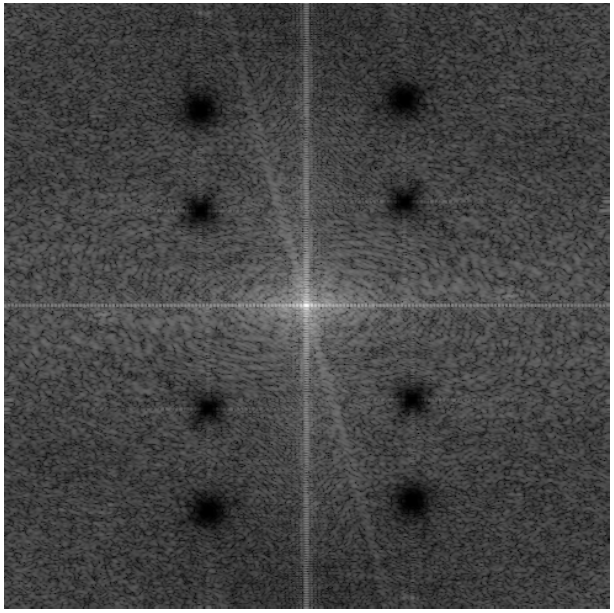


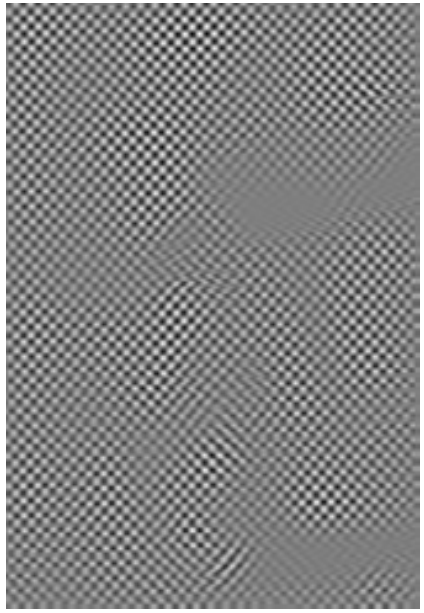
陷波滤波器的应用









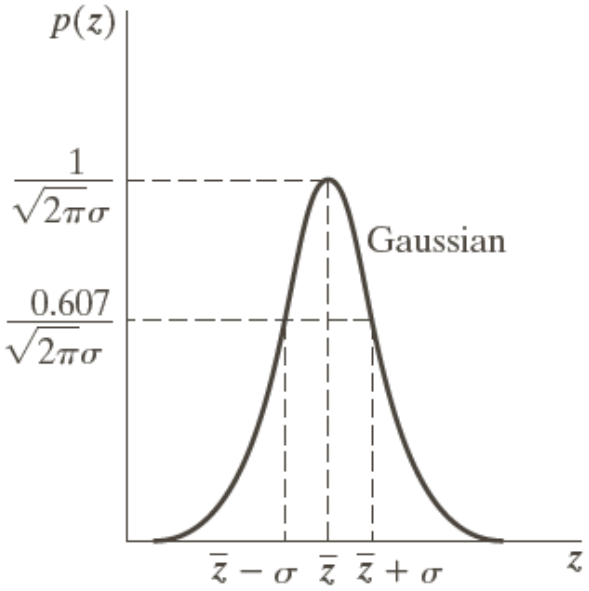


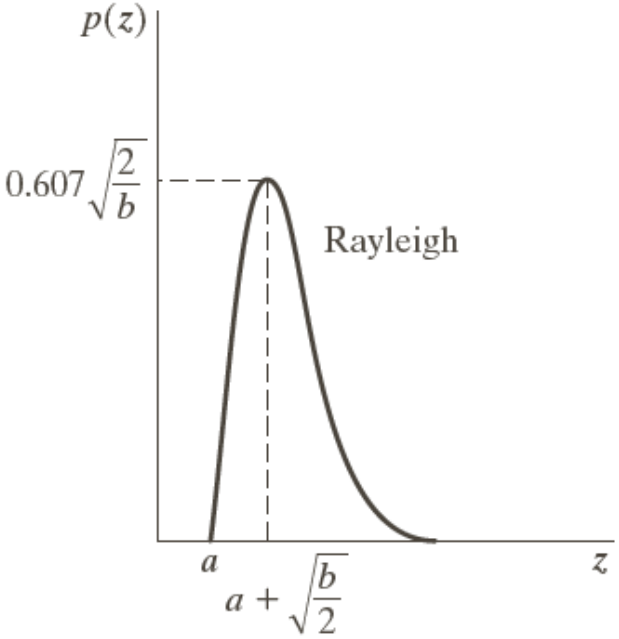
```
%ex0423_notch
f = imread('..\data\Fig0464(a)(car_75DPI_Moire).tif');
f = im2double(f);
[M,N] = size(f);
P = max(2*[M N]);% Padding size.
F = fftshift(fft2(f,P,P));

close all
figure(1),imshow(f,[]);
figure(2),imshow(log(1+abs(F)),[]);

% Creat 4 pairs of notch reject filters
p = [327 163; 327 80; 333 324; 333 406];% locations of maxima, found by imtool
H = ones(P,P);
[DX, DY] = meshgrid(1:P);
D0 = 20;
for k = 1:4
    Dk1 = sqrt((DX-p(k,1)).^2+(DY-p(k,2)).^2);
    Dk2 = sqrt((DX-P-2+p(k,1)).^2+(DY-P-2+p(k,2)).^2);
    H1 = 1./(1+(D0./Dk1).^(2*n));
    H2 = 1./(1+(D0./Dk2).^(2*n));
    H = H.*H1.*H2;
end
figure(3),imshow(H,[]);

% Filtering
G = H.*F;
g = real(ifft2(ifftshift(G)));
g = g(1:M,1:N);

figure(4),imshow(log(1+abs(G)),[]);
figure(5),imshow(g,[]);
```

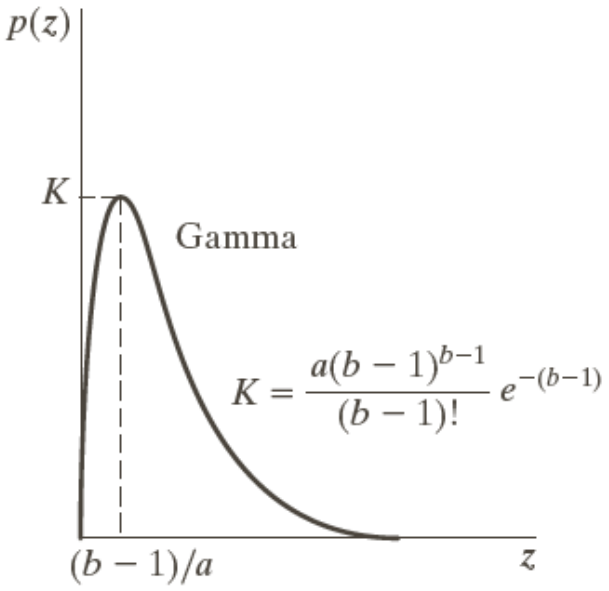
内 容

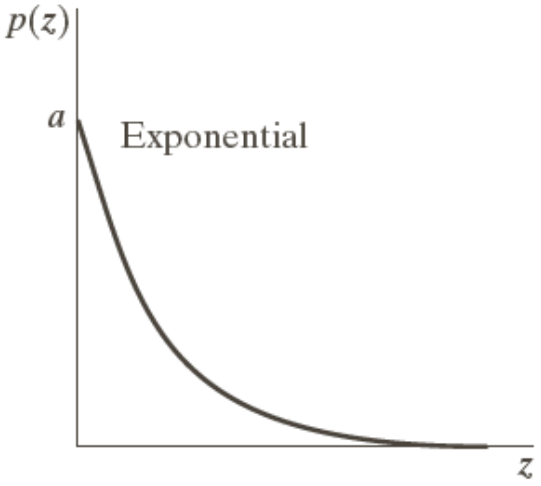
- 图像退化模型
- 只有噪声时的图像恢复
  - 周期性噪声
  - 随机噪声
- 考虑退化函数的图像恢复
  - 逆滤波
  - 维纳滤波

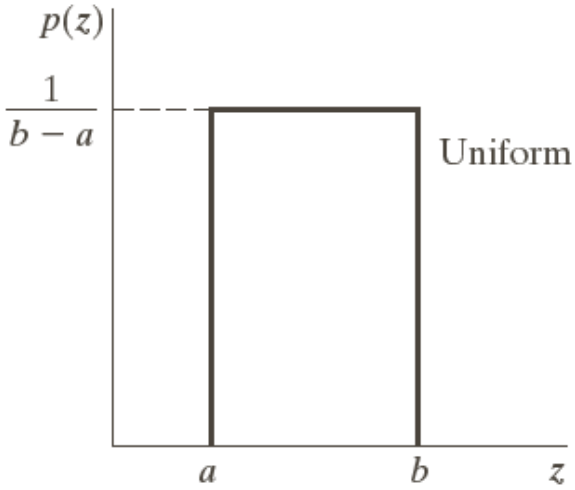
常见随机噪声的PDF

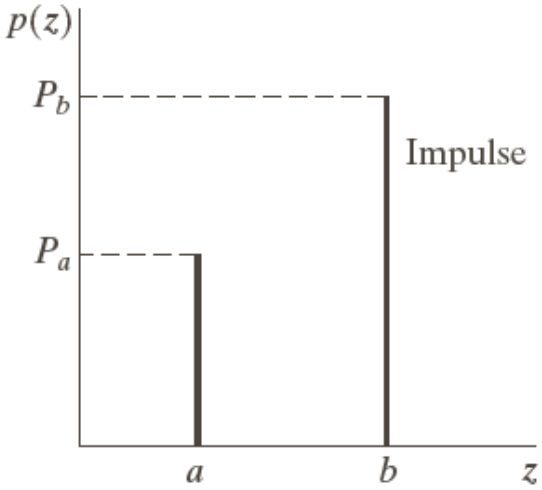

p(z)
1/√(2πσ)
0.607/√(2πσ)
Gaussian
z̄ − σ
z̄
z̄ + σ
z


p(z)
0.607√(2/b)
Rayleigh
a
a + √(b/2)
z


p(z)
K
Gamma
K = a(b − 1)^(b−1)/(b − 1)! e^(−(b−1))
(b − 1)/a
z


p(z)
a
Exponential
z

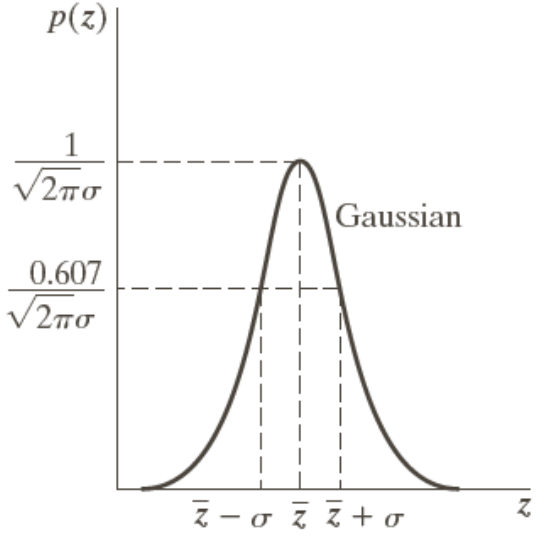

p(z)
1/(b − a)
Uniform
a
b
z


p(z)
P_b
P_a
Impulse
a
b
z

几种常见噪声（1）

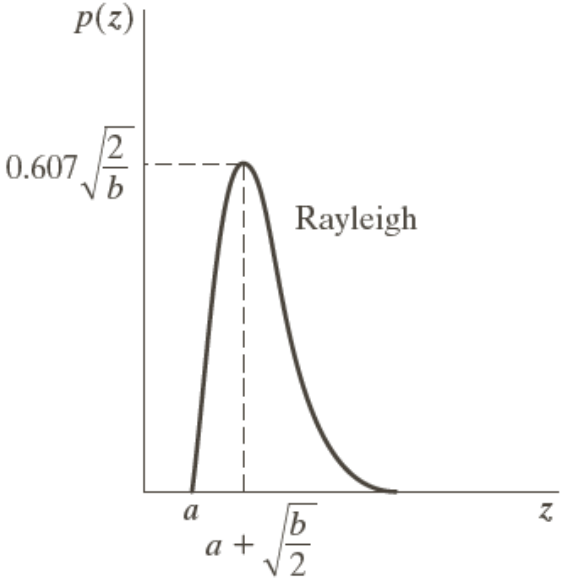
- 高斯噪声随机变量z的PDF

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



- 瑞利（Rayleigh）噪声随机变量z的PDF

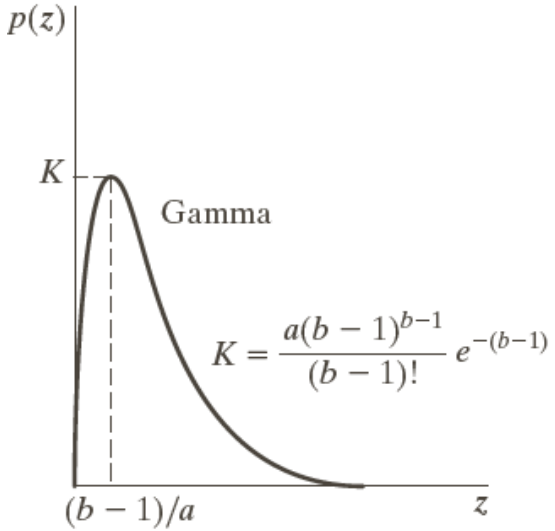
$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{for } z \geq a \\ 0 & \text{otherwise} \end{cases}$$



几种常见噪声（2）

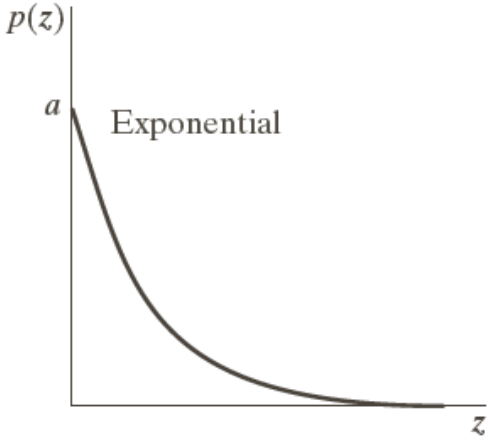
- 爱尔朗（Erlang）噪声随机变量z的PDF

$$p(z) = \begin{cases} \dfrac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



- 指数噪声随机变量的PDF

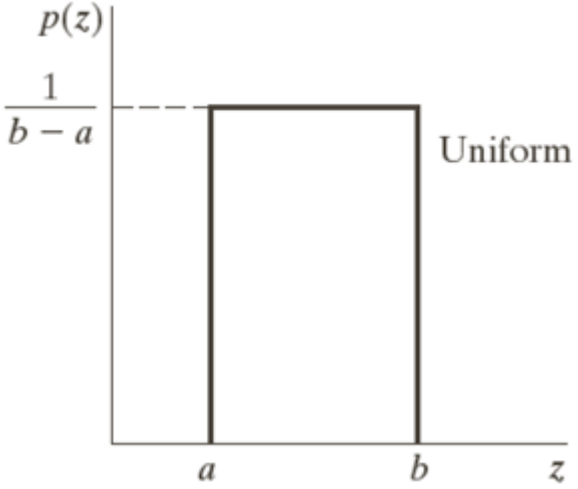
$$p(z) = \begin{cases} ae^{-az} & \text{for } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



几种常见噪声（3）

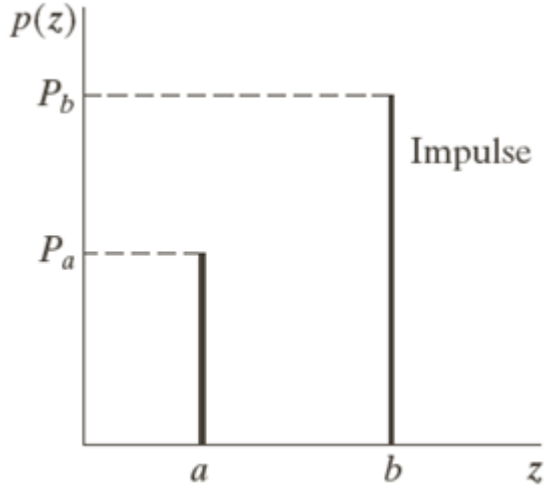
- 均匀噪声随机变量z的PDF

$$p(z)=\begin{cases}\dfrac{1}{b-a} & \text{for } a\le z\le b\\ 0 & \text{otherwise}\end{cases}$$



- 脉冲噪声（也叫椒盐噪声）随机变量z的PDF

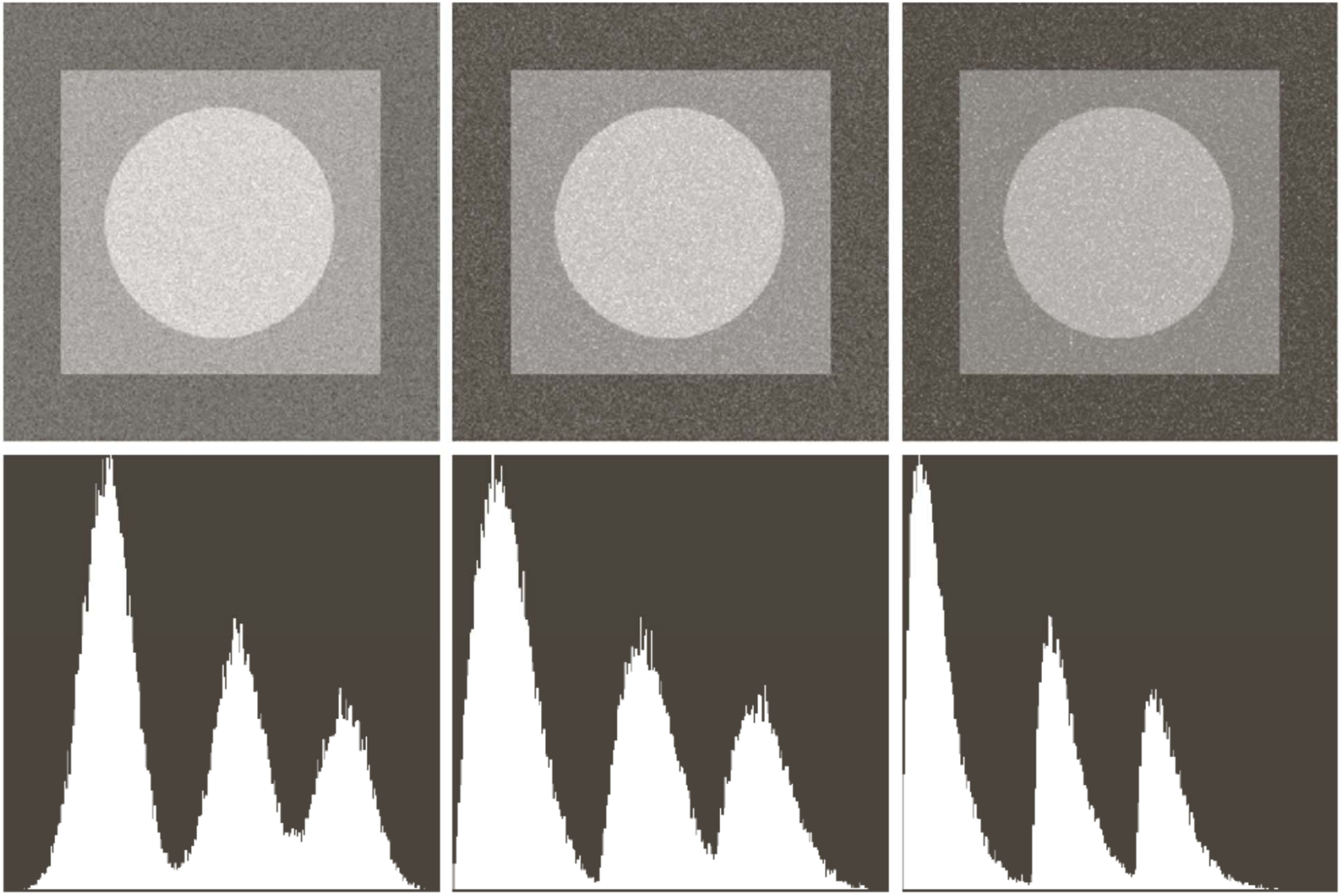
$$p(z)=\begin{cases}P_a & \text{for } z=a\\ P_b & \text{for } z=b\\ 0 & \text{otherwise}\end{cases}$$



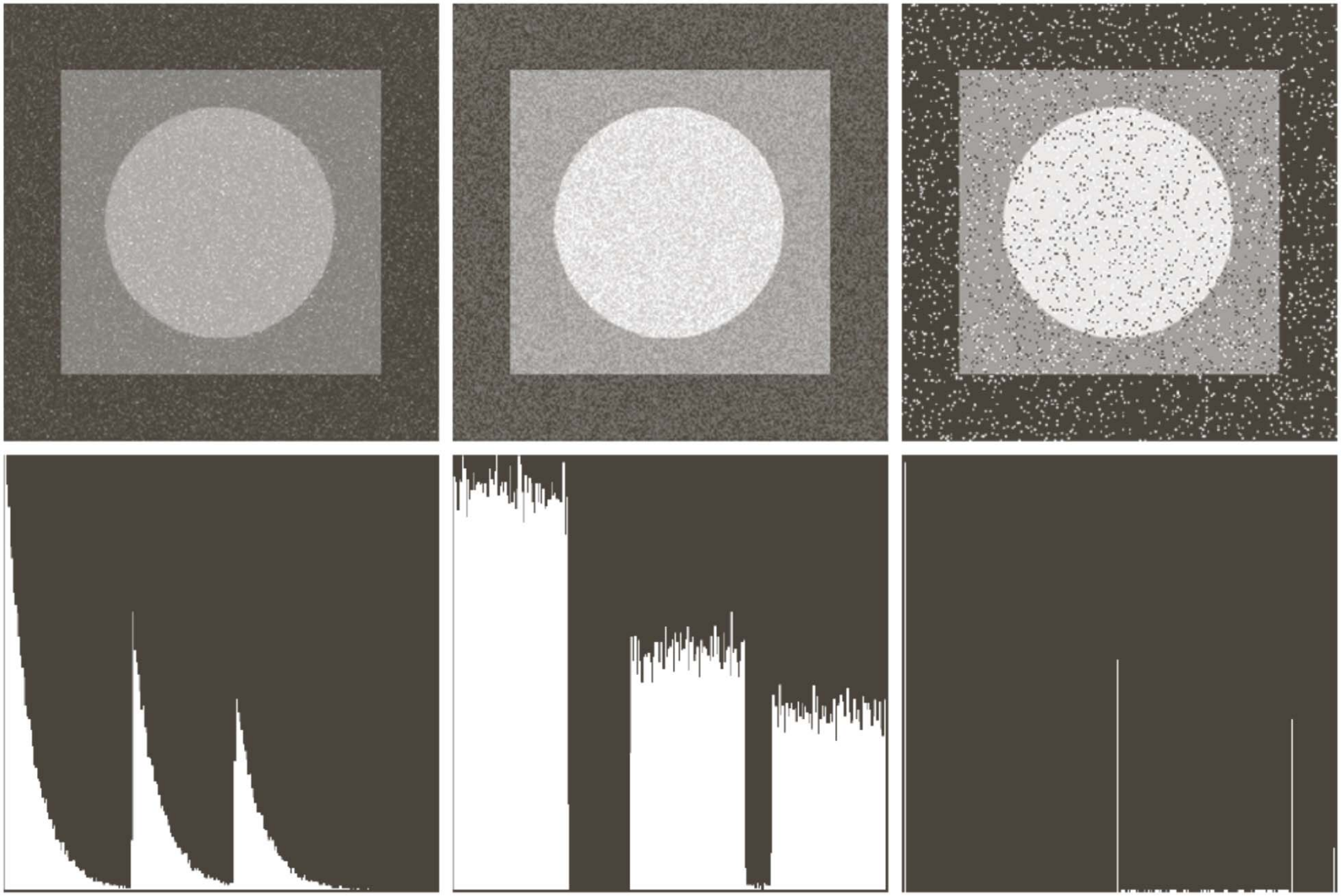
合成噪声

- 为了测试恢复算法，我们需要原始图像和退化图像；但实际应用中往往只有退化图像。
- 为了测试，我们在理想图像上叠加合成噪声得到退化图像。
- MATLAB已有均匀分布和高斯分布随机变量的生成器（rand和randn）
- 如何利用均匀分布生成器，生成任意给定分布的随机变量？
- 直方图匹配（histogram matching）的技术

用该测试图像演示几种加性噪声的效果



Gaussian Rayleigh Gamma

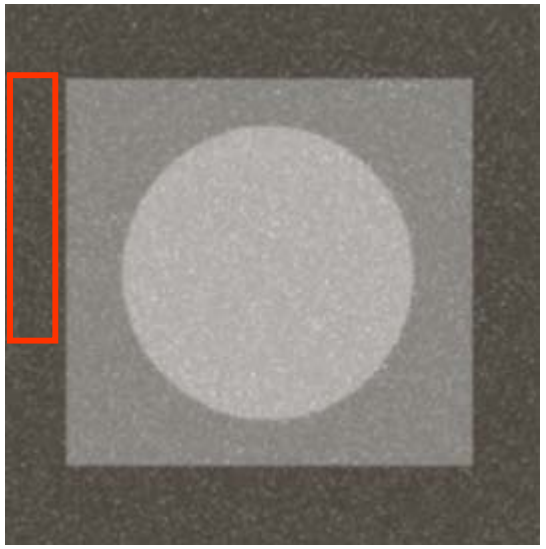


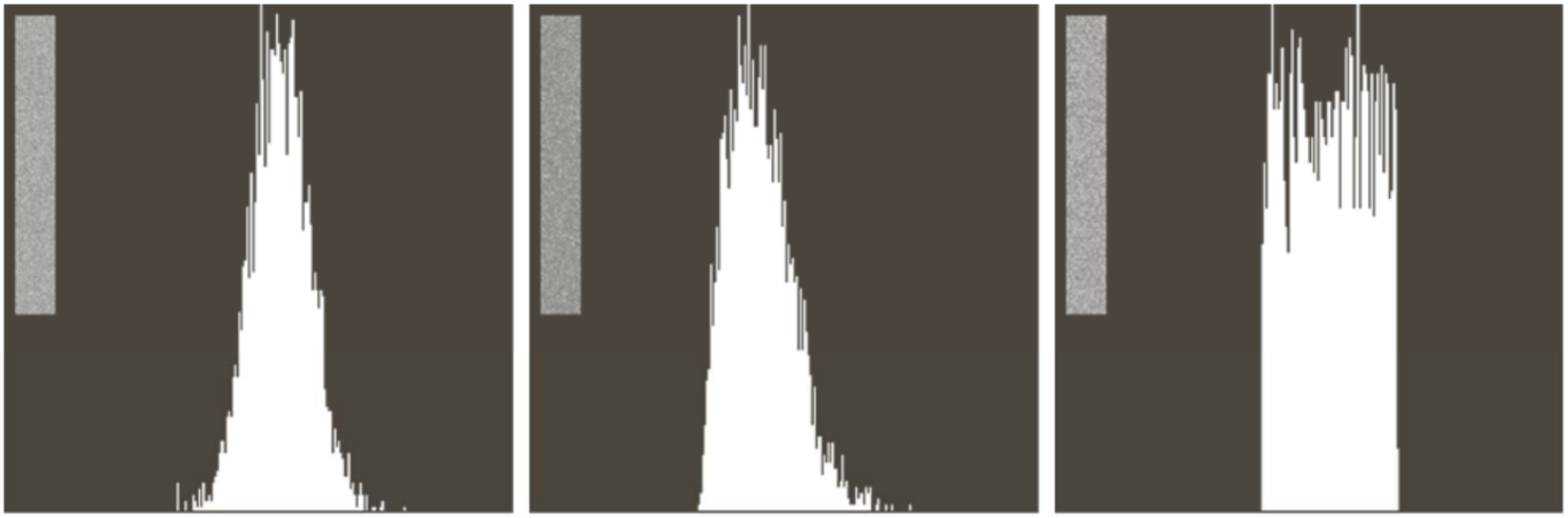
Exponential Uniform Salt & Pepper

噪声参数估计

从给定的退化图像估计噪声参数：

- 选择灰度恒定的区域计算直方图
- 从直方图形状选择合适的PDF形式
- 计算样本均值、方差，计算PDF参数





a b c

FIGURE 5.6 Histograms computed using small strips (shown as inserts) from (a) the Gaussian, (b) the Rayleigh, and (c) the uniform noisy images in Fig. 5.4.

空域滤波

- 均值滤波器
- 顺序统计滤波器
- 自适应滤波器

均值滤波器

- 算术均值滤波器（arithmetic mean filter）

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

- 几何均值滤波器（geometric mean filter）

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

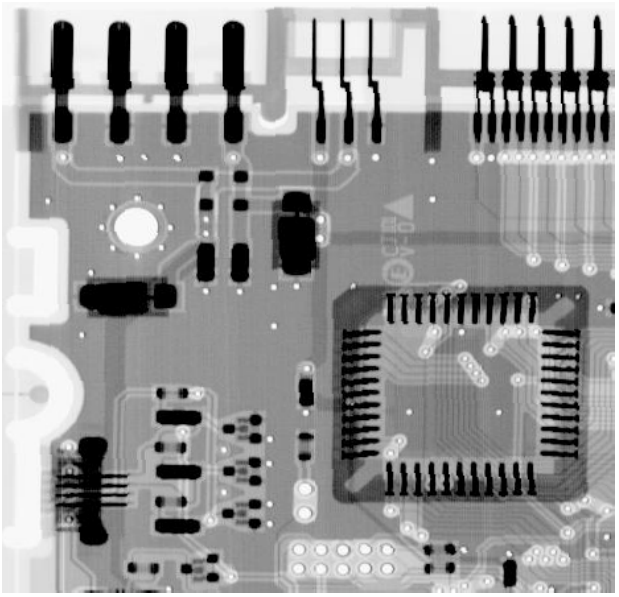
- 逆调和均值滤波器（contraharmonic mean filter）

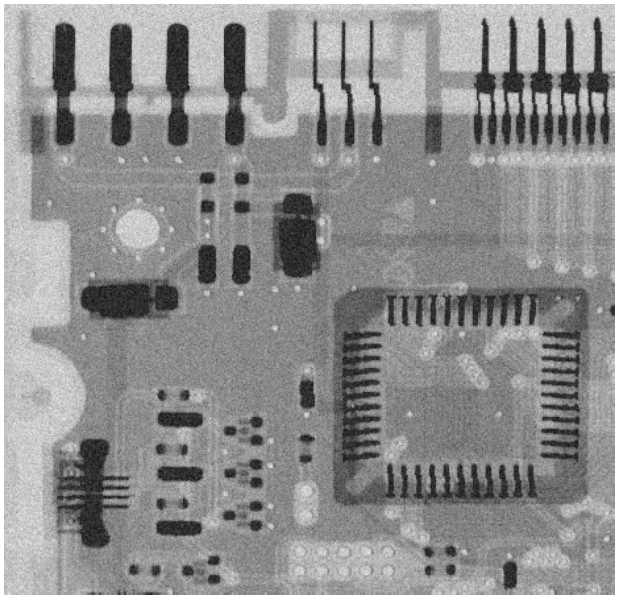
$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

$Q = 0$，算术均值； $Q > 0$，去除椒噪声； $Q < 0$，去除盐噪声

均值滤波器的应用（1）

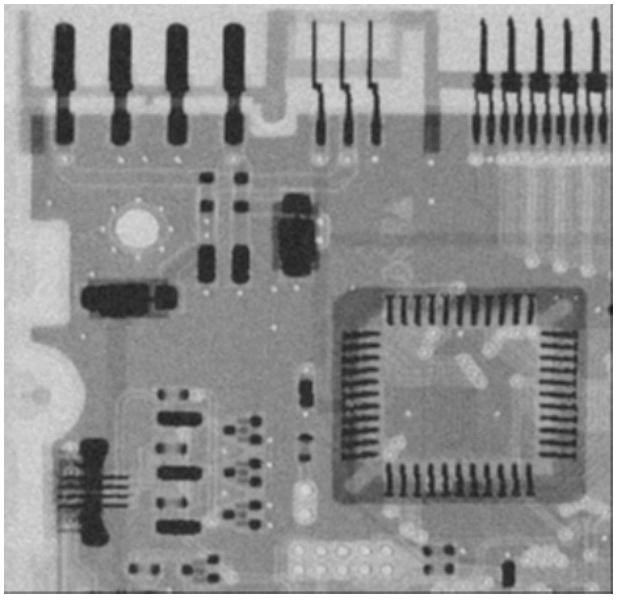
原图

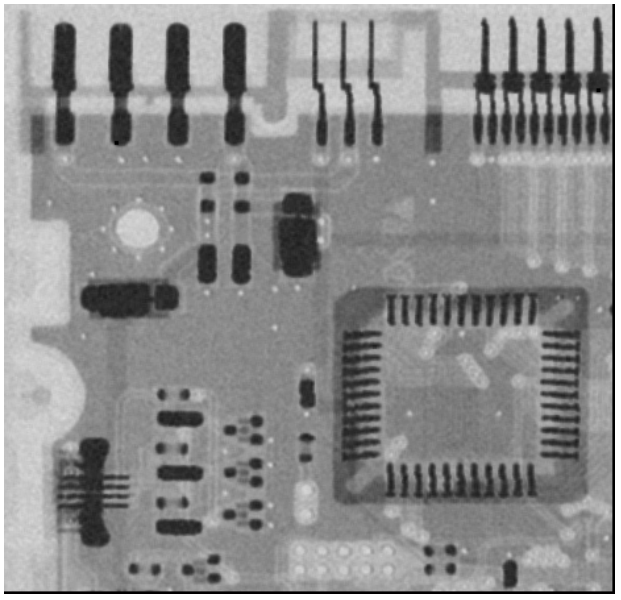




原图加高斯噪声

算术均值滤波器

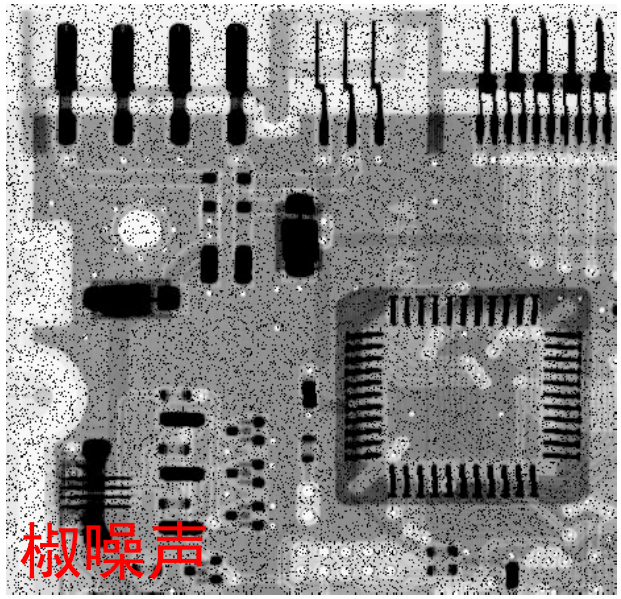


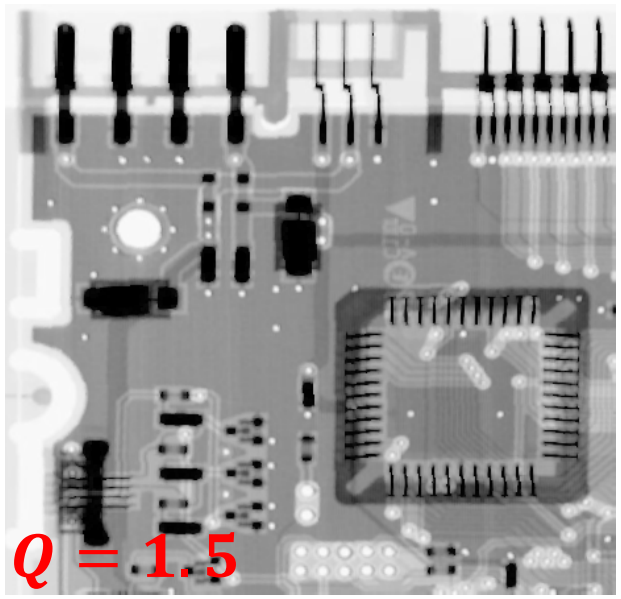


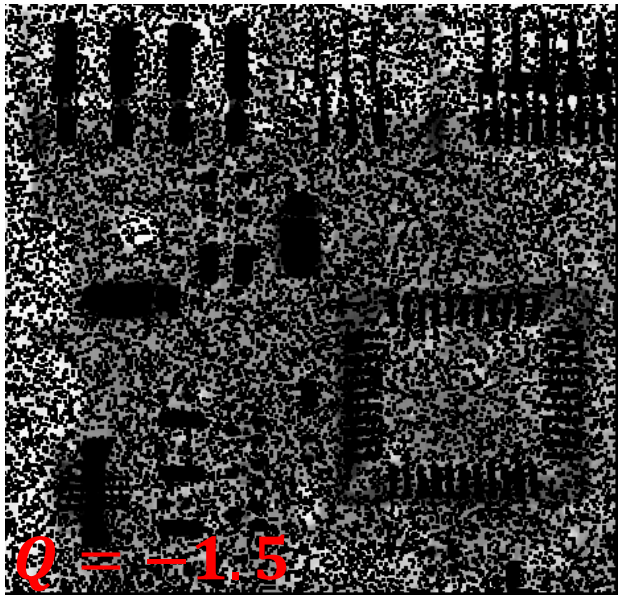
几何均值滤波器

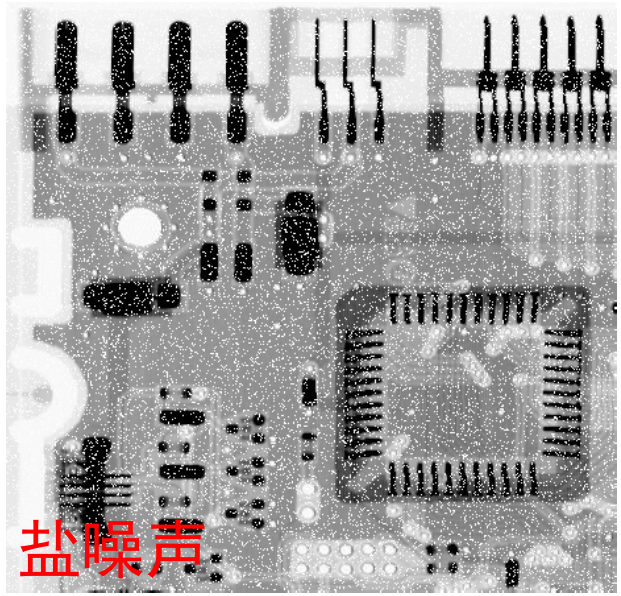
```
% fig0507_MeanFilters, P346
f = imread('..\data\Fig0507(a)(ckt-board-orig).tif');
f = im2double(f);
g = imread('..\data\Fig0507(b)(ckt-board-gauss-var-400).tif');
g = im2double(g);

% arithmetic mean
h_arithmetic = ones(3,3)/9;
f1 = imfilter(g,h_arithmetic);

% geometric mean
fun1 = @(x) power(prod(x(:)),1/9);
f2 = nlfilter(g,[3 3],fun1);

figure(1),clf,
ax(1) = subplot(2,2,1);imshow(f);
ax(2) = subplot(2,2,2);imshow(g);
ax(3) = subplot(2,2,3);imshow(f1);
ax(4) = subplot(2,2,4);imshow(f2);
linkaxes(ax);
```

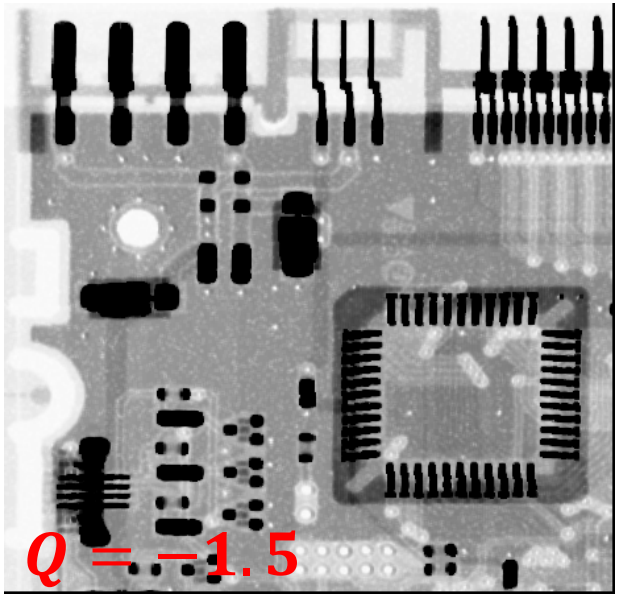
均值滤波器的应用（2）：逆调和均值滤波器

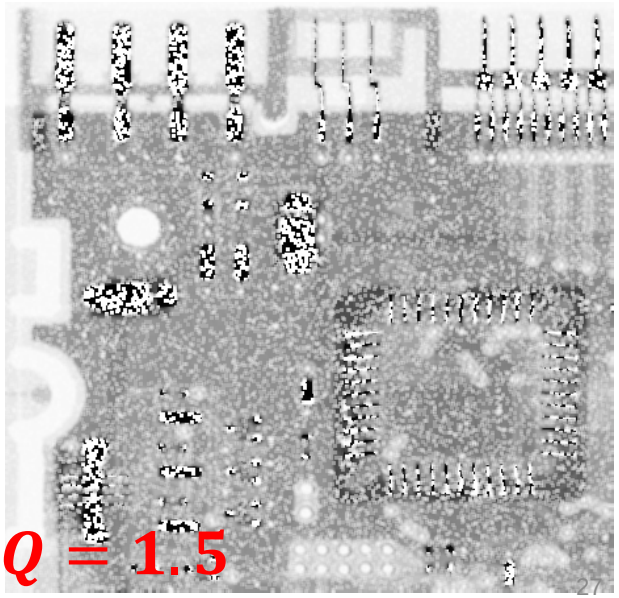












```
% fig0508_MeanFilters, P347

% contraharmonic mean
g1 = imread('..\data\Fig0508(a)(circuit-board-pepper-prob-pt1).tif');
g1 = im2double(g1);
g2 = imread('..\data\Fig0508(b)(circuit-board-salt-prob-pt1).tif');
g2 = im2double(g2);

Q1 = 1.5;
fun1 = @(x) sum(x(:).^(Q1+1))/(eps+sum(x(:).^Q1));
Q2 = -1.5;
fun2 = @(x) sum(x(:).^(Q2+1))/(eps+sum(x(:).^Q2));

f1 = nlfilter(g1,[3 3],fun1);
f2 = nlfilter(g1,[3 3],fun2);

f3 = nlfilter(g2,[3 3],fun2);
f4 = nlfilter(g2,[3 3],fun1);

figure(1),clf,
ax(1) = subplot(2,3,1);imshow(g1);
ax(2) = subplot(2,3,4);imshow(g2);
ax(3) = subplot(2,3,2);imshow(f1);
ax(4) = subplot(2,3,3);imshow(f2);
ax(5) = subplot(2,3,5);imshow(f3);
ax(6) = subplot(2,3,6);imshow(f4);
linkaxes(ax);
```

排序滤波器

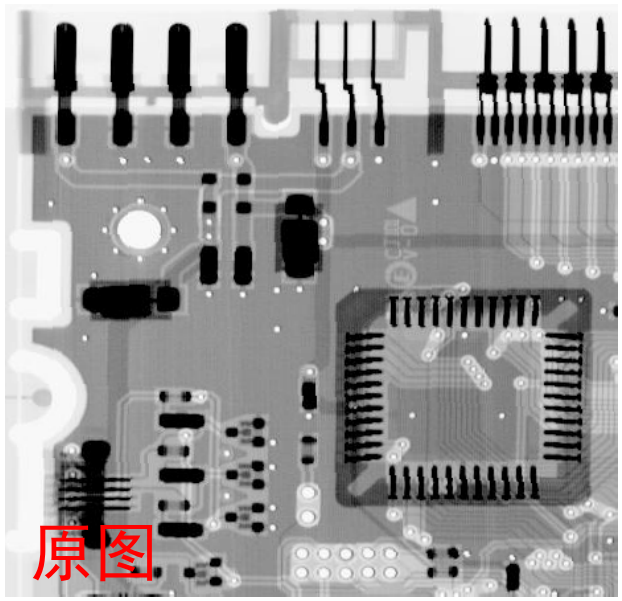
排序滤波器：对输入图像(x,y)邻域像素值进行排序，根据排序结果确定输出值。

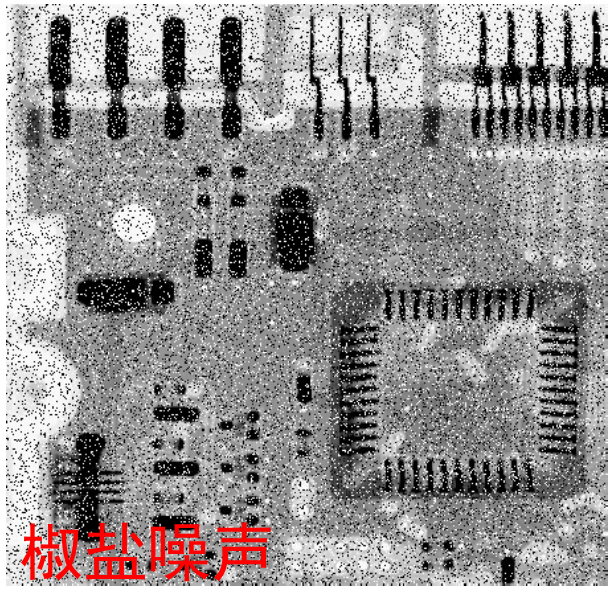
- 中值滤波器：$\hat{f}(x,y)=\text{median}_{(s,t)\in S_{xy}}\{g(s,t)\}$
- 最大值滤波器：$\hat{f}(x,y)=\max_{(s,t)\in S_{xy}}\{g(s,t)\}$
- 最小值滤波器：$\hat{f}(x,y)=\min_{(s,t)\in S_{xy}}\{g(s,t)\}$
- 中点滤波器：

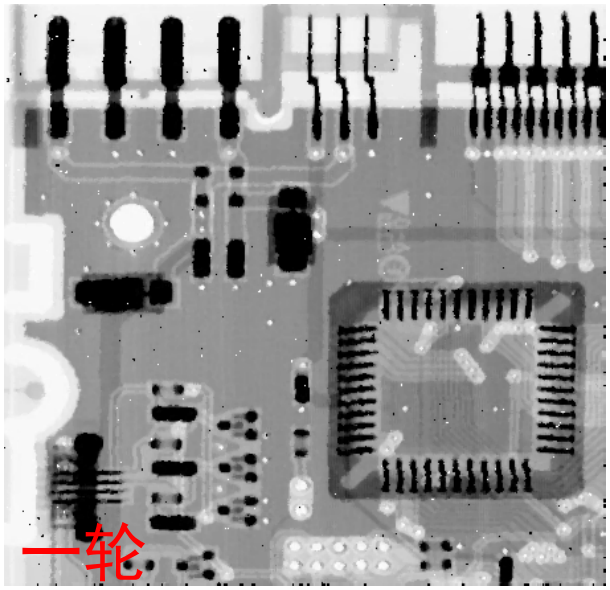
$$\hat{f}(x,y)=\frac{1}{2}\left(\max_{(s,t)\in S_{xy}}\{g(s,t)\}+\min_{(s,t)\in S_{xy}}\{g(s,t)\}\right)$$

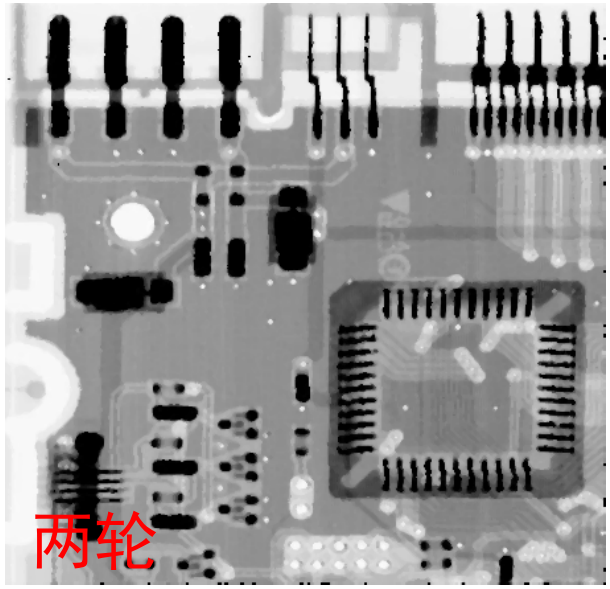
- 剪切均值滤波器：去掉$d/2$个最小值和$d/2$个最大值，取其余值的平均值。它对混合噪声有效（如椒盐噪声和高斯噪声）

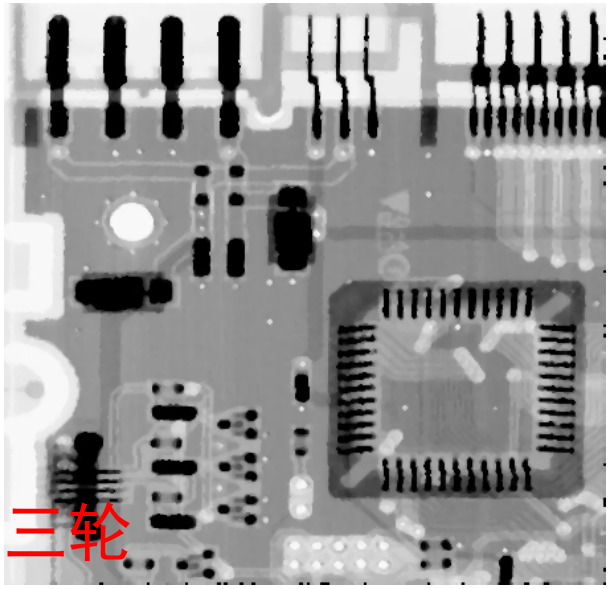
排序滤波器的应用（1）：中值滤波器

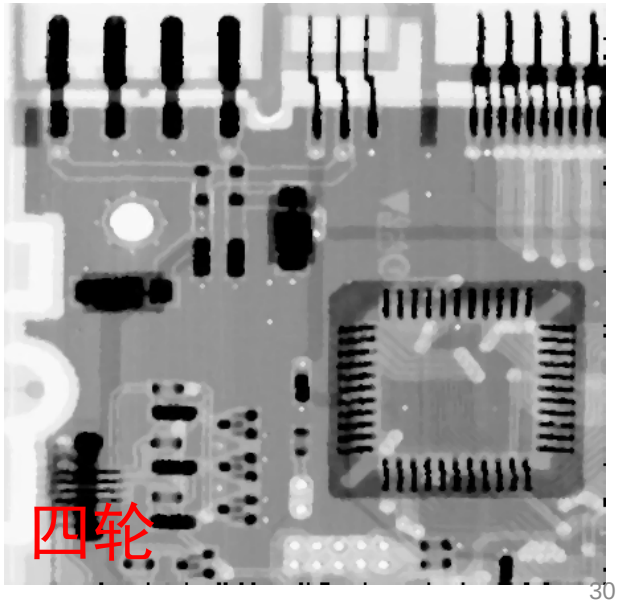








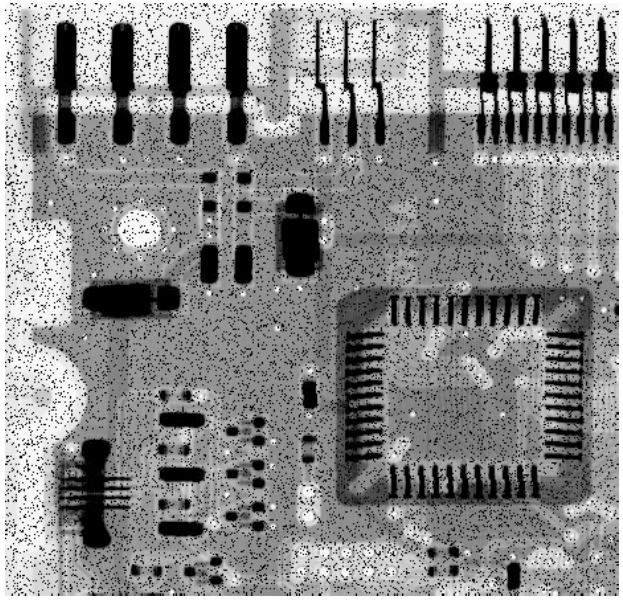


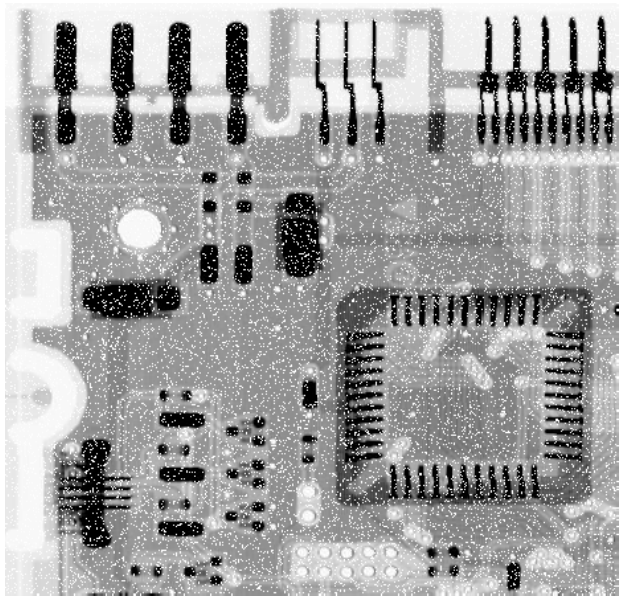


```
% fig0510_MedianFilter, P350

f = imread('..\data\Fig0507(a)(ckt-board-orig).tif');
f = im2double(f);
g = imread('..\data\Fig0510(a)(ckt-board-saltpep-prob.pt05).tif');
g = im2double(g);

fun = @(x) median(x(:));
f1 = nlfilter(g,[3 3],fun);
f2 = nlfilter(f1,[3 3],fun);
f3 = nlfilter(f2,[3 3],fun);
f4 = nlfilter(f3,[3 3],fun);

figure(1),clf,
ax(1) = subplot(2,3,1);imshow(f);
ax(2) = subplot(2,3,2);imshow(g);
ax(3) = subplot(2,3,3);imshow(f1);
ax(4) = subplot(2,3,4);imshow(f2);
ax(5) = subplot(2,3,5);imshow(f3);
ax(6) = subplot(2,3,6);imshow(f4);
linkaxes(ax);
```

排序滤波器的应用（2）

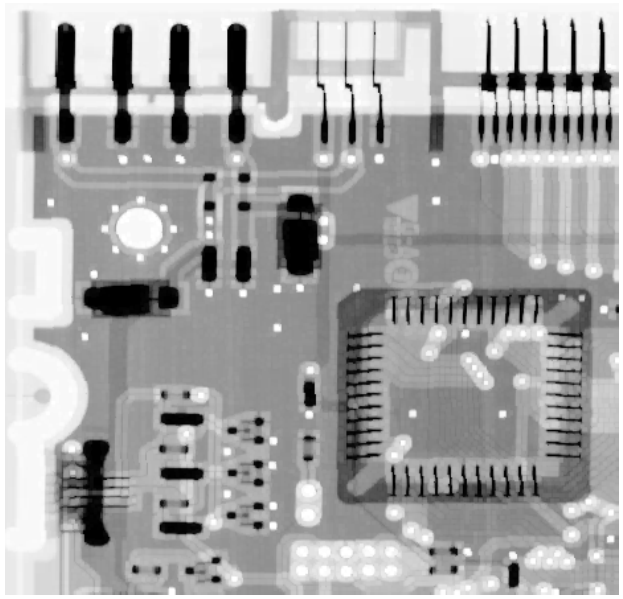
椒噪声

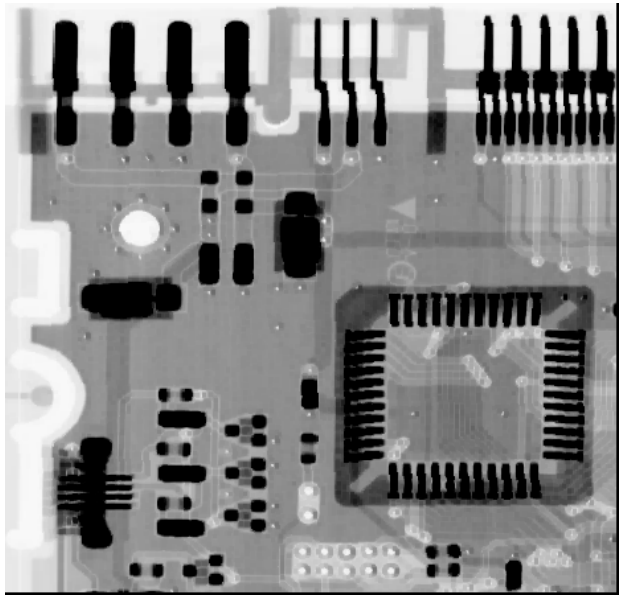




盐噪声

最大值滤波



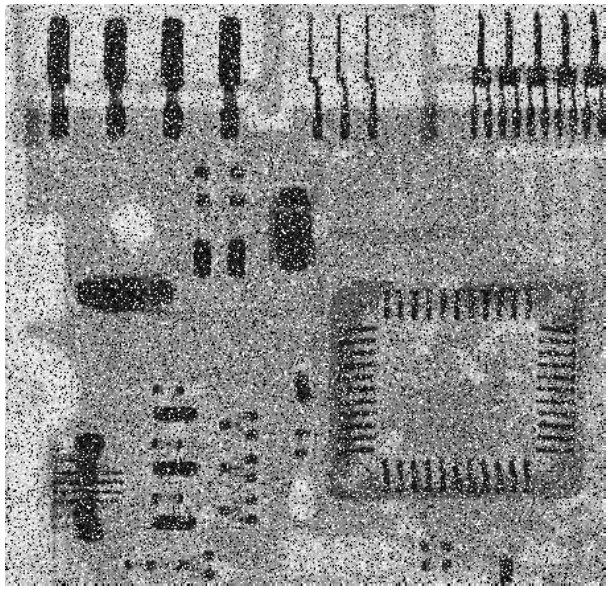


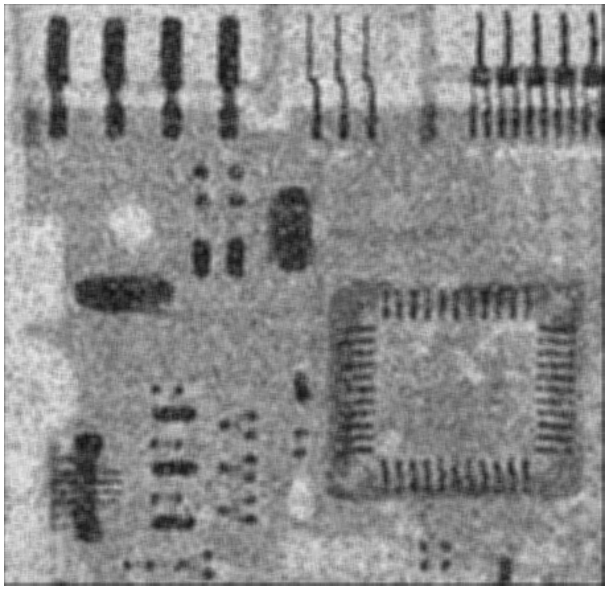
最小值滤波

```
% fig0511_MaxMinFilter, P350

g1 = imread('..\data\Fig0508(a)(circuit-board-pepper-prob-pt1).tif');
g1 = im2double(g1);
g2 = imread('..\data\Fig0508(b)(circuit-board-salt-prob-pt1).tif');
g2 = im2double(g2);

fun1 = @(x) max(x(:));
fun2 = @(x) min(x(:));
f1 = nlfilter(g1,[3 3],fun1);
f2 = nlfilter(g2,[3 3],fun2);

figure(1),clf,
ax(1) = subplot(2,2,1);imshow(g1);
ax(2) = subplot(2,2,2);imshow(g2);
ax(3) = subplot(2,2,3);imshow(f1);
ax(4) = subplot(2,2,4);imshow(f2);
linkaxes(ax);
```

排序滤波器的应用（3）

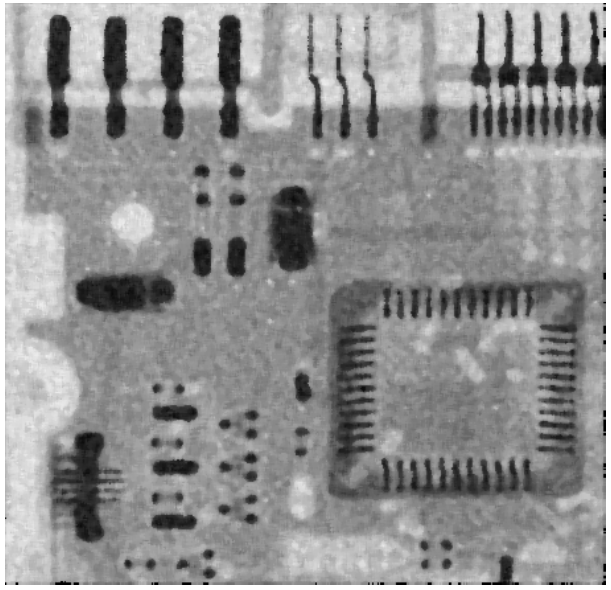
椒盐+均匀噪声

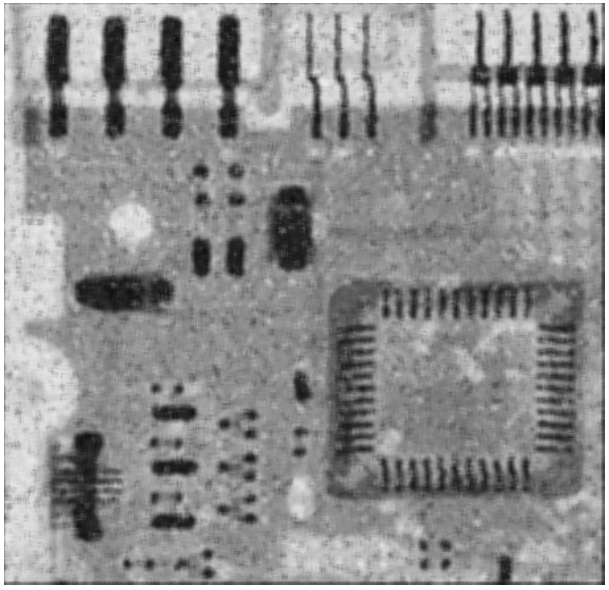




算术均值

中值滤波





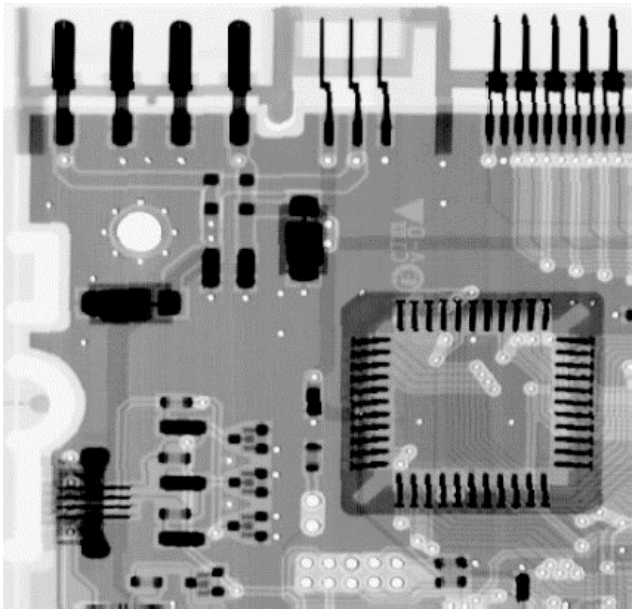
剪切均值滤波

```
% fig0512_TrimmedMeanFilter, P351
f = imread('..\data\Fig0507(a)(ckt-board-orig).tif');
f = im2double(f);
g = imread('..\data\Fig0512(b)(ckt-uniform-plus-saltpepr-prob-pt1).tif');
g = im2double(g);

hsize = 5;
% arithmetic mean
h_arithmetic = ones(hsize,hsize)/(hsize*hsize);
f1 = imfilter(g,h_arithmetic);

% geometric mean
fun1 = @(x) power(prod(x(:)),1/(hsize*hsize));
f2 = nlfilter(g,[hsize hsize],fun1);

% median
fun2 = @(x) median(x(:));
f3 = nlfilter(g,[hsize hsize],fun2);

% Trimmed Mean Filter
f4 = TrimmedMeanFilter(g,hsize,6);

figure(1),clf,
ax(1) = subplot(2,3,1);imshow(f),title('Original image');
ax(2) = subplot(2,3,2);imshow(g),title('Noisy image');
ax(3) = subplot(2,3,3);imshow(f1),title('arithmetic mean');
ax(4) = subplot(2,3,4);imshow(f2,[]),title('geometric mean');
ax(5) = subplot(2,3,5);imshow(f3),title('median');
ax(6) = subplot(2,3,6);imshow(f4),title('trimmed mean');
linkaxes(ax);
```

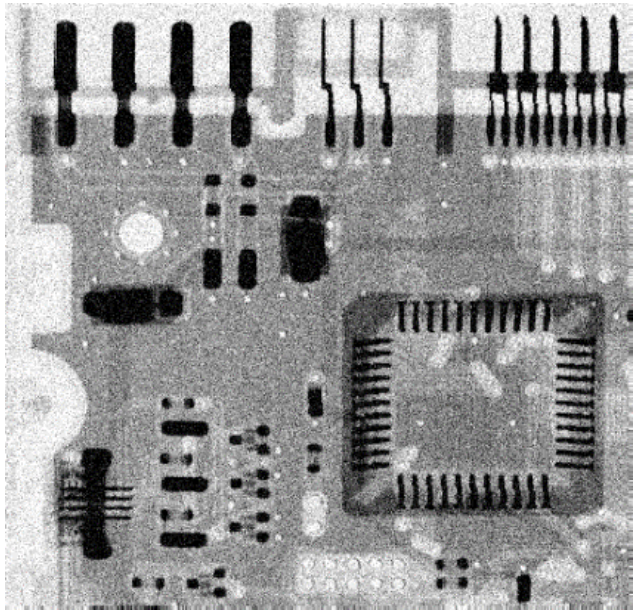
自适应均值滤波器

前面介绍的几种均值滤波器对图像的所有位置一视同仁，没有考虑图像不同位置可能具有不同的特征。

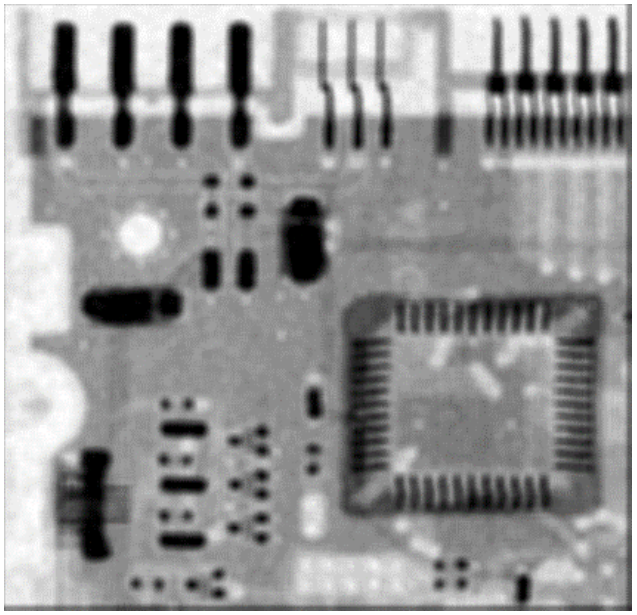
Original image



Noisy image



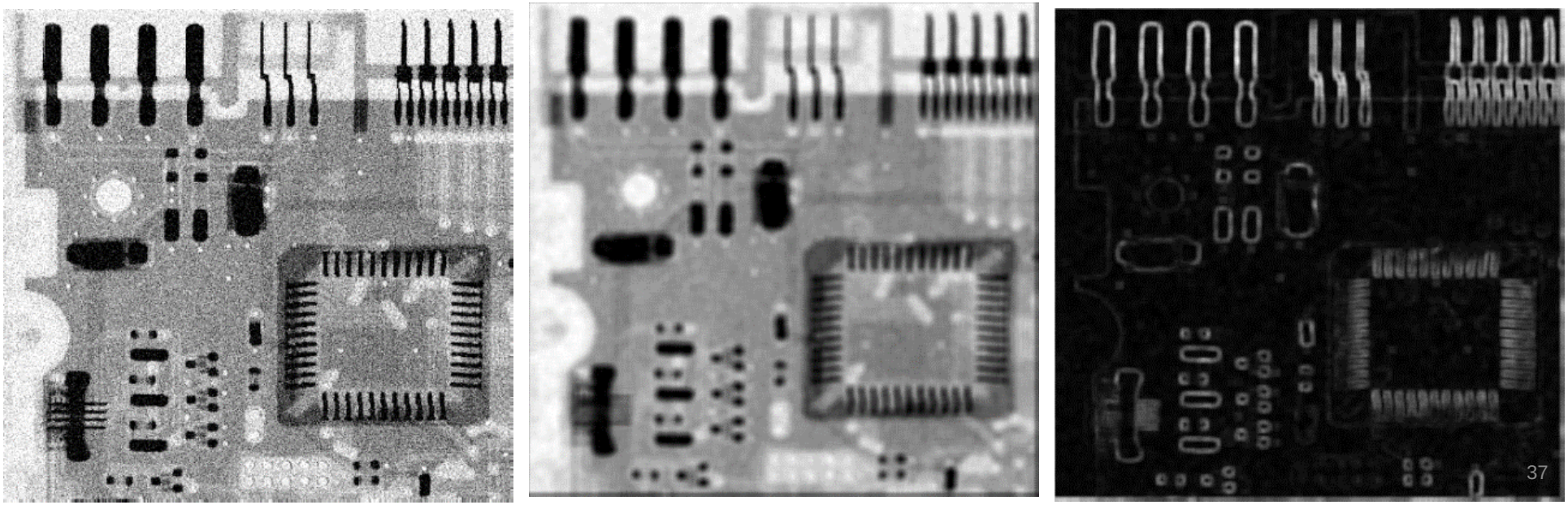
Arithmetic mean



自适应均值滤波器

- “自适应”：算法根据图像局部特征来调整行为
- 图像的局部均值和方差是基本的特征
- 自适应均值滤波器在(x,y)处的输出取决于4个量：退化图像像素值$g(x,y)$，噪声方差$\sigma_\eta{}^2$，局部均值m_L，局部方差$\sigma_L{}^2$。
- 局部均值和方差是在(x,y)的邻域S_{xy}内计算。
- 噪声方差是给定的，或通过某种方法估计得到

degraded image　　local mean　　local variance



自适应均值滤波器

自适应均值滤波器应该具有以下特点：

- 如果噪声方差$\sigma_\eta{}^2 = 0$，应输出$g(x,y)$
- 如果局部方差$\sigma_L{}^2$很大（例如远大于噪声方差），输出值应比较接近$g(x,y)$
- 如果局部方差$\sigma_L{}^2$很小（例如等于噪声方差），输出值应等于局部均值m_L

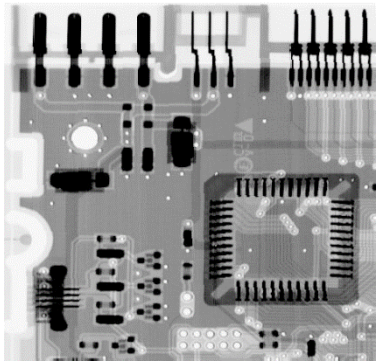
换句话说，输出值等于$g(x,y)$和m_L的加权平均，而权重取决于局部方差与噪声方差的比值。

下面公式具有以上特点

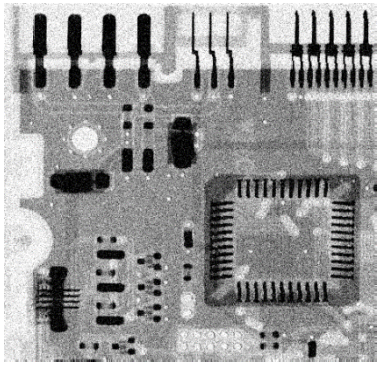
$$\hat{f}(x,y) = \left(1 - \frac{\sigma_\eta{}^2}{\sigma_L{}^2}\right) g(x,y) + \frac{\sigma_\eta{}^2}{\sigma_L{}^2} m_L$$

一般，$\sigma_\eta{}^2 \leq \sigma_L{}^2$. 如果$\sigma_\eta{}^2 > \sigma_L{}^2$时，可令其为1.

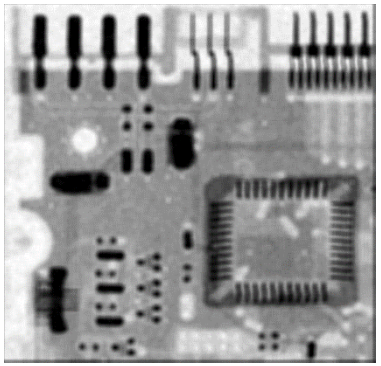
Original image



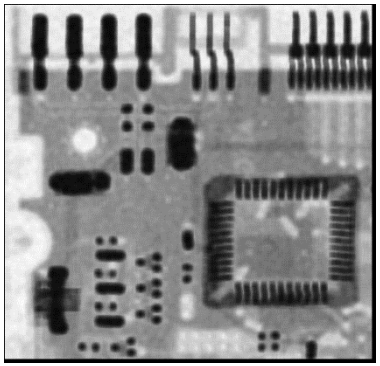
Noisy image



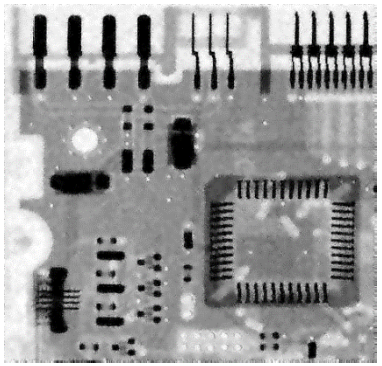
arithmetic mean



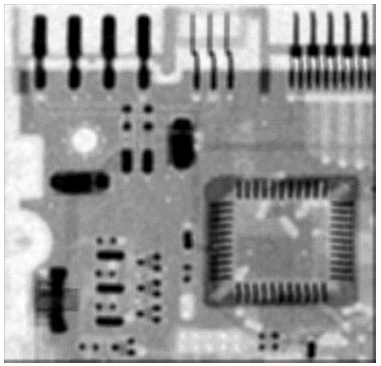
geometric mean



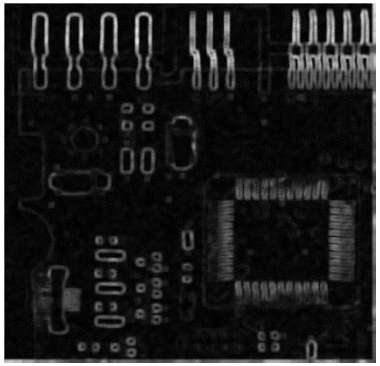
adaptive mean



local mean

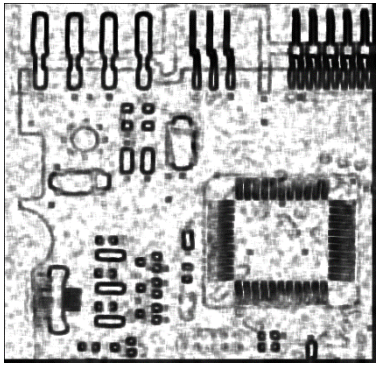


local variance



mask

weight



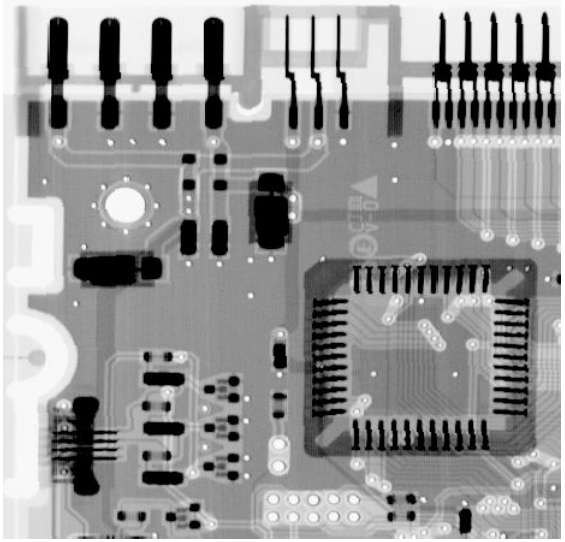
```
% fig0513_AdaptiveMean, P353
clear all
f = imread('..\data\Fig0507(a)(ckt-board-orig).tif');
f = im2double(f);
var_noise_real = 1000/(255*255);
var_noise = var_noise_real;
% var_noise = var_noise_real-0.01;
% var_noise = var_noise_real+0.02;
noise = randn(size(f,1),size(f,2))*sqrt(var_noise_real);
g = f + noise;
hsize = 7;
% arithmetic mean
h_arithmetic = ones(hsize,hsize)/(hsize*hsize);
f1 = imfilter(g,h_arithmetic);
% geometric mean
fun1 = @(x) power(prod(x(:)),1/(hsize*hsize));
f2 = nlfilter(g,[hsize hsize],fun1);
% adaptive mean
% Compute local mean and local variance
fun1 = @(x) mean(x(:));
mean_local = nlfilter(g,[hsize hsize],fun1);
fun2 = @(x) var(x(:));
var_local = nlfilter(g,[hsize hsize],fun2);
mask = var_local<var_noise;
w = var_noise./(var_local+eps);
w(mask) = 1;
f3 = (1-w).*g + w.*mean_local;

figure(1),clf,
ax(1) = subplot(3,3,1);imshow(f),title('Original image');
ax(2) = subplot(3,3,2);imshow(g),title('Noisy image');
ax(3) = subplot(3,3,3);imshow(f1),title('arithmetic mean');
ax(4) = subplot(3,3,4);imshow(f2,[]),title('geometric mean');
ax(5) = subplot(3,3,5);imshow(f3),title('adaptive mean');
ax(6) = subplot(3,3,6);imshow(mean_local),title('local mean');
ax(7) = subplot(3,3,7);imshow(var_local,[]),title('local variance');
ax(8) = subplot(3,3,8);imshow(mask,[]),title('mask');
ax(9) = subplot(3,3,9);imshow(w,[]),title('weight');
linkaxes(ax);
```

自适应中值滤波器

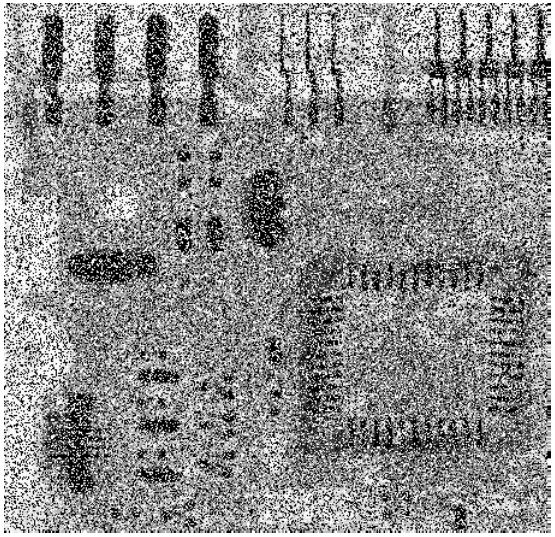
- 中值滤波器：将当前像素替换为邻域像素的中值
- 当椒盐比例较小时（例如< 0.2），小邻域（例如3×3）内的中值往往是未被破坏的原始图像值。中值滤波很有效。
- 但当椒盐比例较大时，中值滤波器面临两难：
  - 小邻域内，中值很可能是椒或者盐，滤波无效
  - 增大邻域，可避免将椒盐选为中值，但会导致图像失真

椒盐比例高时的中值滤波

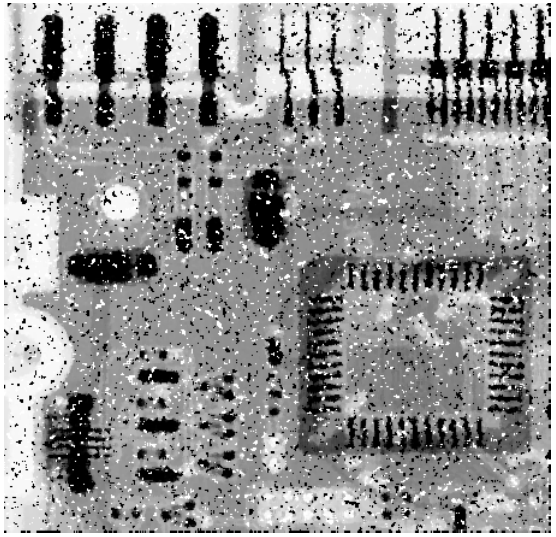
Original image



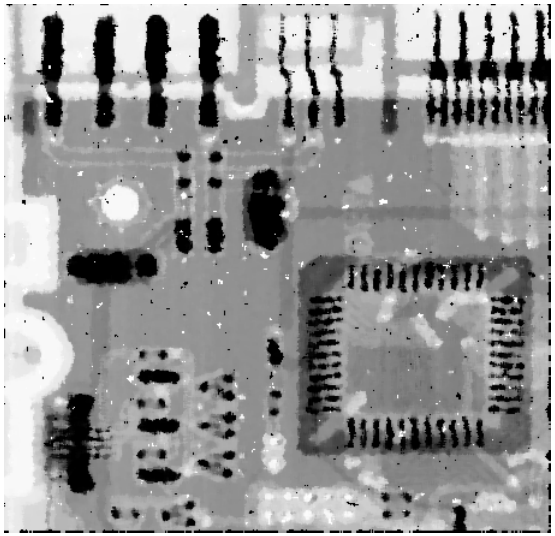
Noisy image



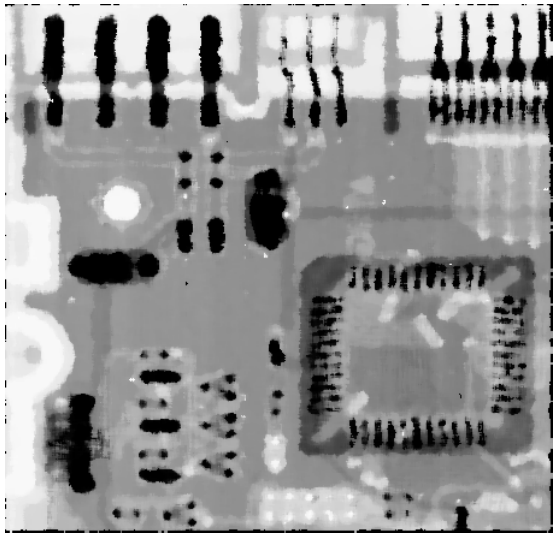
median 3x3



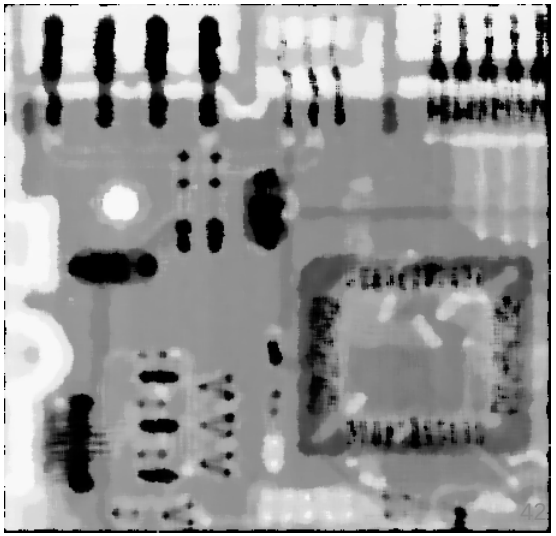
median 5x5



median 7x7



median 9x9



```
% fig0514_Median, P356
clear all
f = imread('..\data\Fig0507(a)(ckt-board-orig).tif');
g = imread('..\data\Fig0514(a)(ckt_saltpep_prob_pt25).tif');

fun1 = @(x) median(x(:));
f1 = nlfilter(g,[3 3],fun1);
f2 = nlfilter(g,[5 5],fun1);
f3 = nlfilter(g,[7 7],fun1);
f4 = nlfilter(g,[9 9],fun1);

figure(1),clf,
ax(1) = subplot(2,3,1);imshow(f),title('Original image');
ax(2) = subplot(2,3,2);imshow(g),title('Noisy image');
ax(3) = subplot(2,3,3);imshow(f1),title('median 3x3');
ax(4) = subplot(2,3,4);imshow(f2),title('median 5x5');
ax(5) = subplot(2,3,5);imshow(f3),title('median 7x7');
ax(6) = subplot(2,3,6);imshow(f4),title('median 9x9');
linkaxes(ax);
```

自适应中值滤波器

- 自适应中值滤波器的2个思想：
  - 从小到大尝试不同大小的邻域，优先在小邻域完成滤波
  - 如无必要，不改变输入图像值
- 算法：对每个像素执行如下两个阶段的操作

Stage A:

Set window size to 3×3
Compute $z_{\text{med}}, z_{\text{min}}, z_{\text{max}}$
if $z_{\text{med}} > z_{\text{min}}$ and $z_{\text{med}} < z_{\text{max}}$
  Go to Stage B
end
Increase the window size
if window size<S_{max}
  Repeat Stage A
else
  Output z_{med}
end

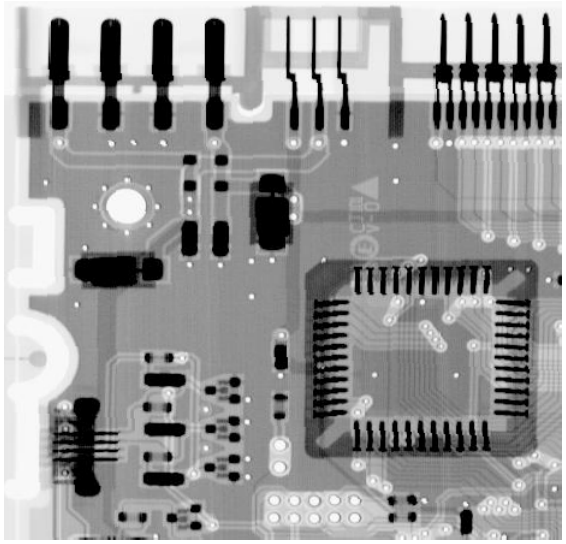
Stage B:

if $z_{\text{xy}} > z_{\text{min}}$ and $z_{\text{xy}} < z_{\text{max}}$
  Output z_{xy}
else
  Output z_{med}
end

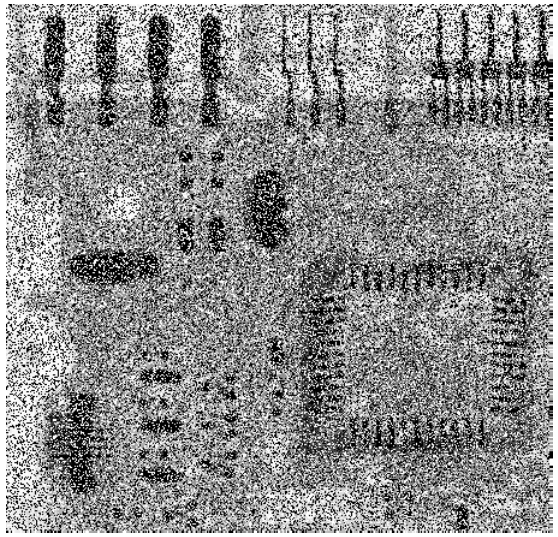
z_{min}：当前邻域内最小值
z_{max}：当前邻域内最大值
z_{med}：当前邻域内中值
z_{xy}：(x, y)处像素值
S_{max}：最大邻域

自适应中值滤波器

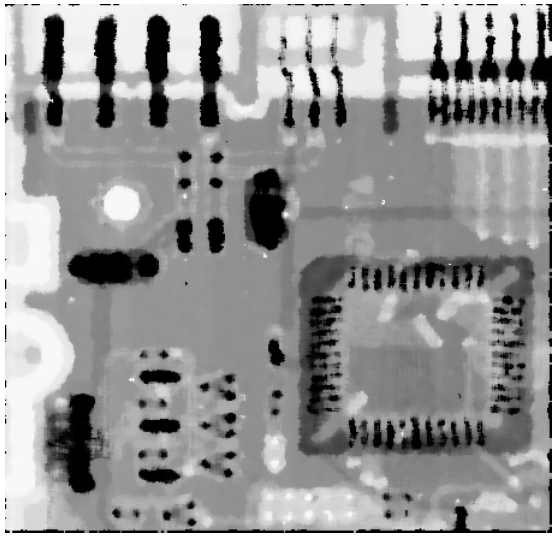
Original image



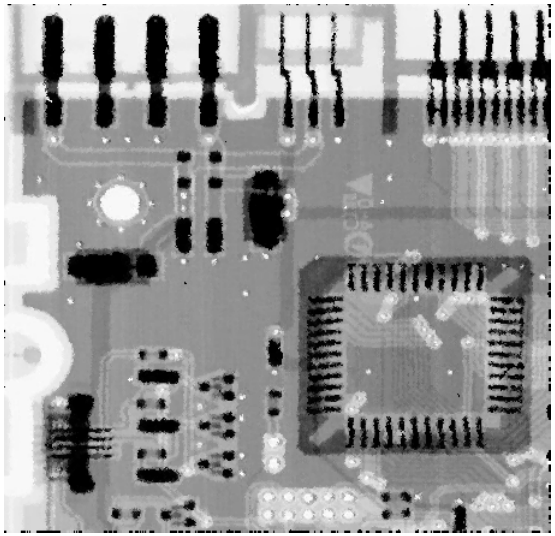
Noisy image



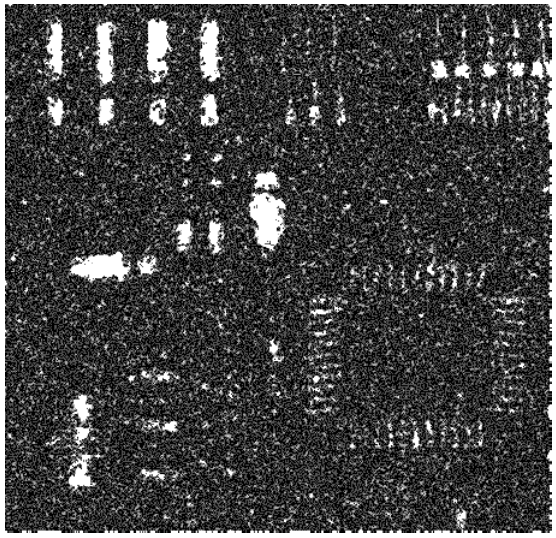
median 7x7



adaptive median 7x7



flag



```
% fig0514_AdaptiveMedian, P356
f = imread('..\data\Fig0507(a)(ckt-board-orig).tif');
g = imread('..\data\Fig0514(a)(ckt_saltpep_prob_pt25).tif');

local_median = cell(1,3);
local_max = cell(1,3);
local_min = cell(1,3);
fun_median = @(x) median(x(:));
fun_max = @(x) max(x(:));
fun_min = @(x) min(x(:));

sizes = [3 5 7];
for k = 1:3
    local_median{k} = nlfilter(g,[sizes(k) sizes(k)],fun_median);
    local_max{k} = nlfilter(g,[sizes(k) sizes(k)],fun_max);
    local_min{k} = nlfilter(g,[sizes(k) sizes(k)],fun_min);
end

f_adaptive = local_median{3};
flag = 3*ones(size(g));
for m = 1:size(g,1)
    for n = 1:size(g,2)
        for k = 1:3
            if local_median{k}(m,n)>local_min{k}(m,n) && local_median{k}(m,n)<local_max{k}(m,n)
                if g(m,n)>local_min{k}(m,n) && g(m,n)<local_max{k}(m,n)
                    f_adaptive(m,n) = g(m,n);
                    flag(m,n) = 0;
                else
                    f_adaptive(m,n) = local_median{k}(m,n);
                    flag(m,n) = k;
                end
                break
            end
        end
    end
end

figure(1),clf,
ax(1) = subplot(2,3,1);imshow(f),title('Original image');
ax(2) = subplot(2,3,2);imshow(g),title('Noisy image');
ax(3) = subplot(2,3,3);imshow(local_median{3}),title('median 7x7');
ax(4) = subplot(2,3,4);imshow(f_adaptive),title('adaptive median 7x7');
ax(5) = subplot(2,3,5);imshow(flag,[]),title('flag');
linkaxes(ax);
```

内 容

- 图像退化模型
- 只有噪声时的图像恢复
  - 周期性噪声
  - 随机噪声
- 考虑退化函数的图像恢复
  - 逆滤波
  - 维纳滤波

退化函数

- 退化模型的空域表示为：

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

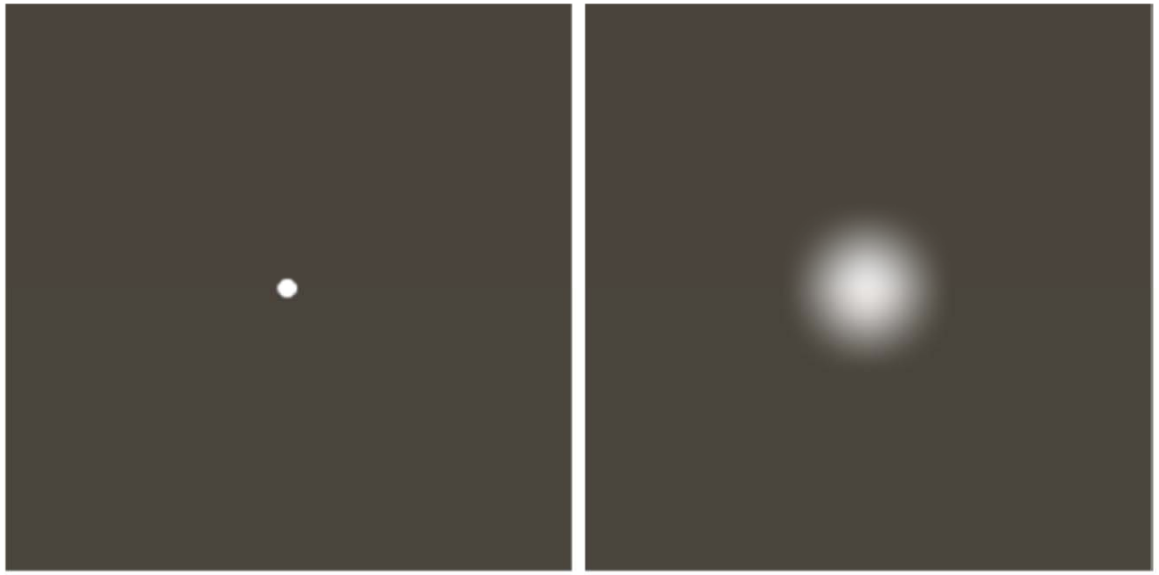
- 按照卷积定理，其频域表示为：

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

- 脉冲响应函数 $h(x,y)$ 也叫点扩散函数（point spread function），是对点光源的成像结果
- 退化函数$H(u,v)$也叫光学转移函数（optical transfer function）

a b

FIGURE 5.24 Degradation estimation by impulse characterization. (a) An impulse of light (shown magnified). (b) Imaged (degraded) impulse.



退化函数（1）

- 大气湍流是大气中一种不规则的随机运动，对光的传播产生影响，使成像模糊

- 退化模型为

$$H(u,v) = e^{-k\left(u^2+v^2\right)^{5/6}}$$

- 形式像高斯低通滤波器；k越大，带宽越窄

original

$k = 0.0025$

$k = 0.001$

$k = 0.00025$

退化函数（2）

- 运动模糊：快门打开和关闭之间，物体和相机发生比较大的相对运动
- h = fspecial('motion', len, theta)

1. Construct an ideal line segment with the desired length and angle, centered at the center coefficient of h.
2. For each coefficient location (i,j), compute the nearest distance between that location and the ideal line segment.
3. h = max(1 - nearest_distance, 0);
4. Normalize h:h = h/(sum(h(:)))

模拟运动模糊

```
% SimulateMotionBlur
I = imread('cameraman.tif');
figure(1),clf,subplot(2,2,1),imshow(I);
H = fspecial('motion',10,0);
MotionBlur = imfilter(I,H,'replicate');
subplot(2,2,2),imshow(MotionBlur);
H = fspecial('motion',20,90);
MotionBlur = imfilter(I,H,'replicate');
subplot(2,2,3),imshow(MotionBlur);
H = fspecial('motion',30,135);
MotionBlur = imfilter(I,H,'replicate');
subplot(2,2,4),imshow(MotionBlur);
```

逆滤波（Inverse Filtering）

退化模型的频域表示：

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

逆滤波：

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

由于噪声$N(u,v)$未知，无法精确恢复$F(u,v)$

如果没有噪声，可以精确恢复$F(u,v)$吗？

退化函数为16×16均值滤波，无噪声

为防止除0，$\hat{F}(u,v) = R(u,v)G(u,v)$

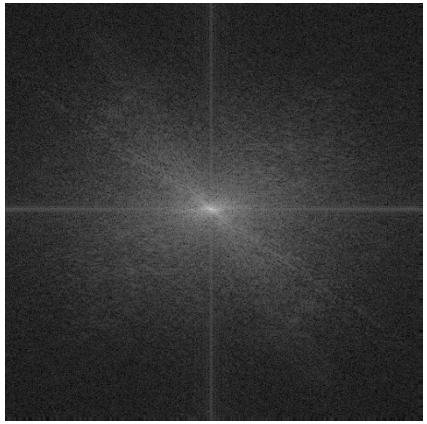
$$R(u,v) = \begin{cases} 1/H(u,v) & \text{if } |H(u,v)| > n \\ 1/n & \text{otherwise} \end{cases}$$

$$g(x,y) = h(x,y) \star f(x,y)$$
$$G(u,v) = H(u,v)F(u,v)$$

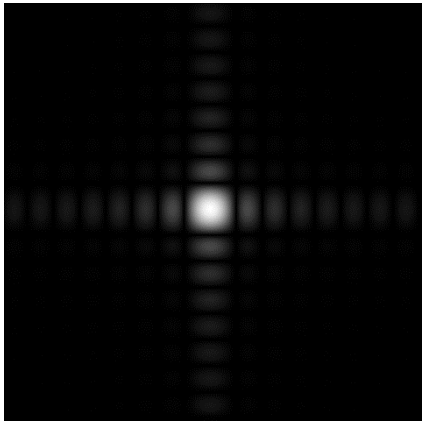
$$n = 0.00002$$

f

$|F|$

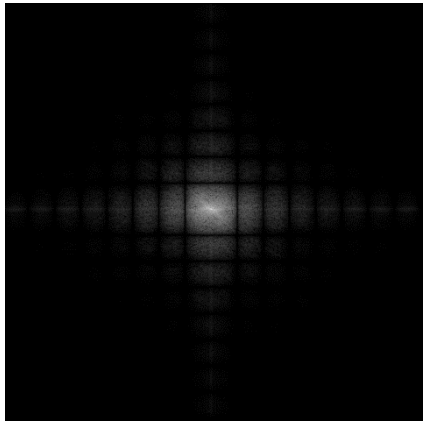


$|H|$

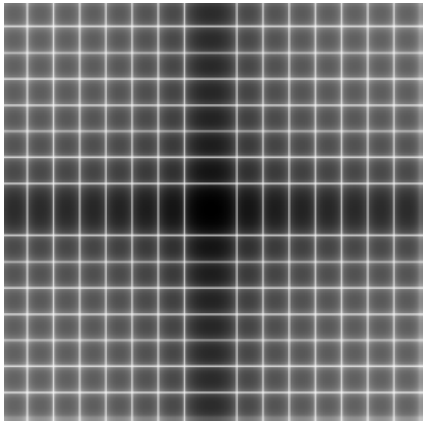


g

$|G|$

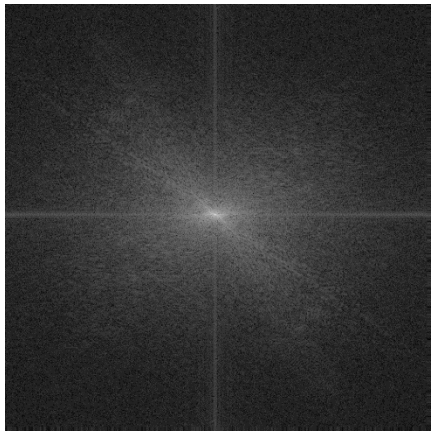


$|R|$

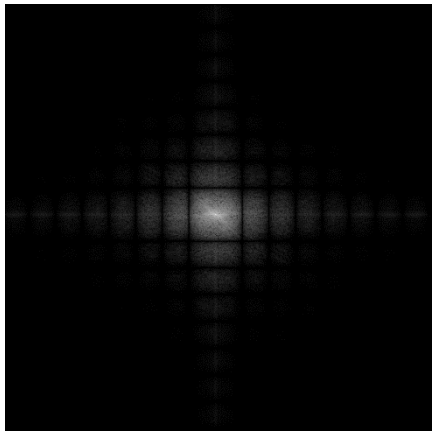


除了退化函数为0的地方，
几乎完全恢复出$|F|$

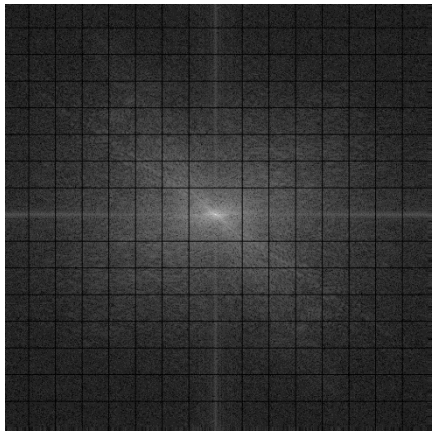
$|F|$



$|G|$



$|\hat{F}|$



f

g

$\hat{f}$

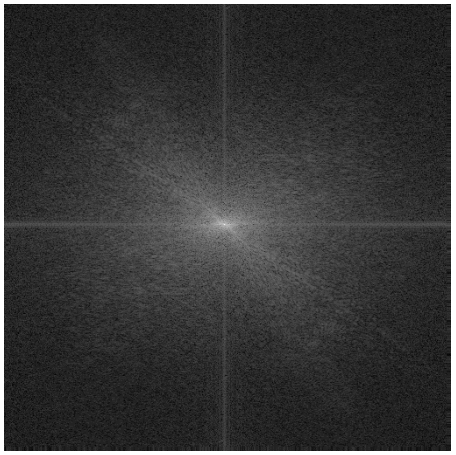
- 退化函数为16×16均值滤波
- 高斯噪声

$$g(x,y) = h(x,y) \star f(x,y) + \eta(x,y)$$
$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

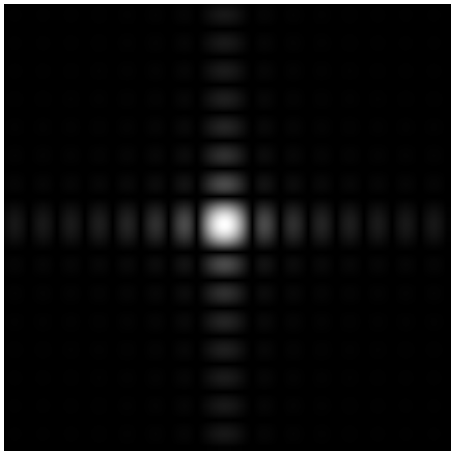
$$n = 0.00002$$

f

$|F|$

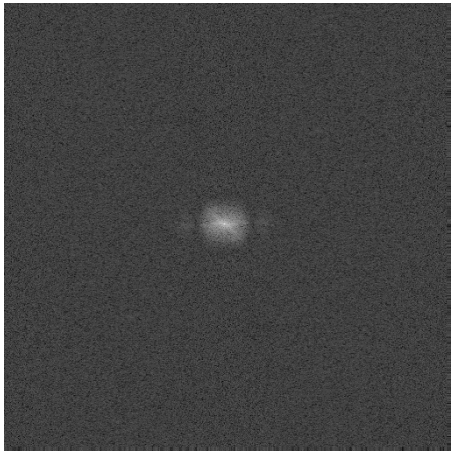


$|H|$

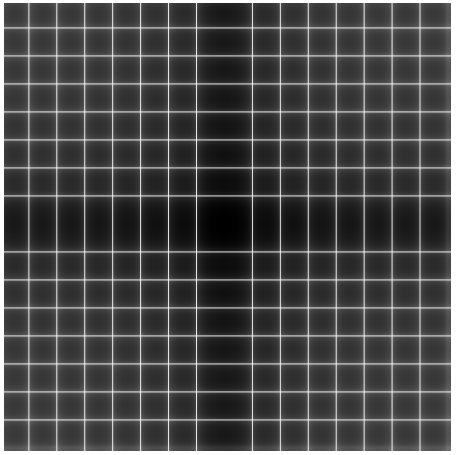


g

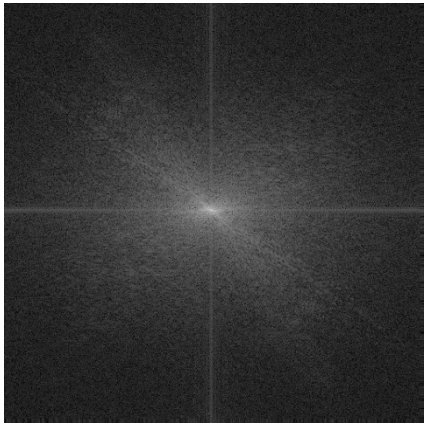
$|G|$



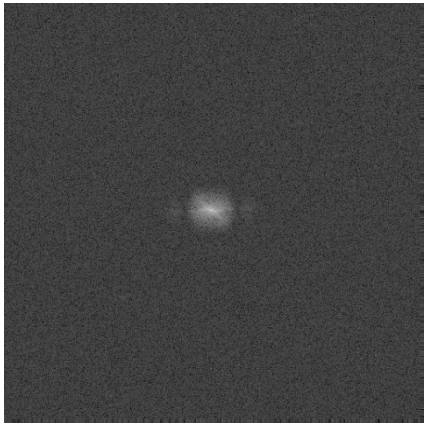
$|R|$



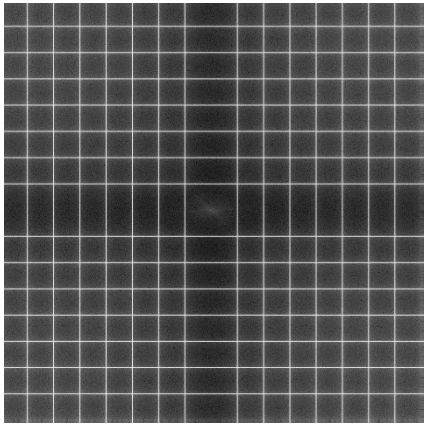
$|F|$



$|G|$



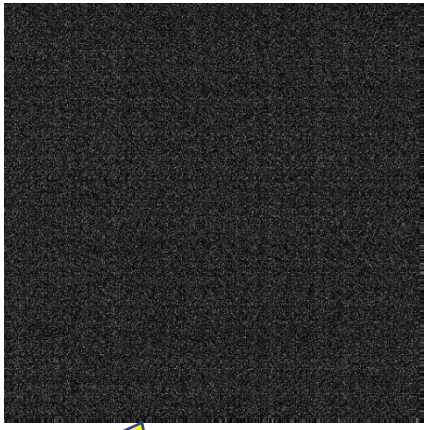
$|\hat{F}|$



f

g

$\hat{f}$



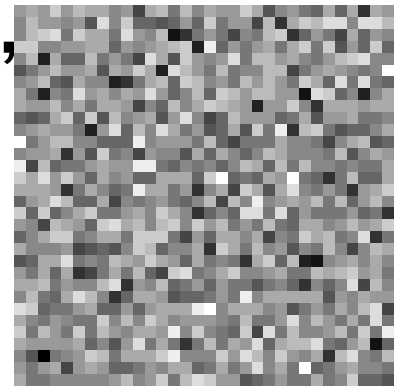
完全没有恢复

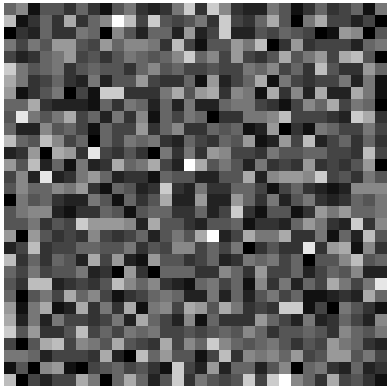
$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

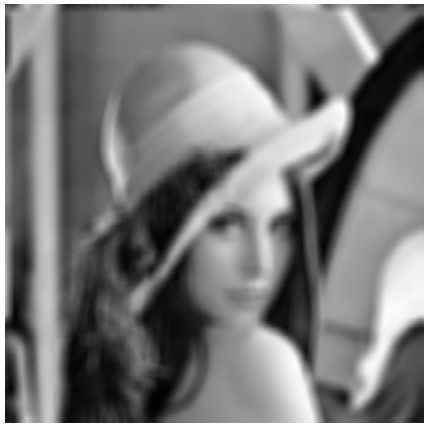
模糊图像$H(u,v)F(u,v)$的高频成分弱

而噪声有很强高频成分（如高斯噪声及频谱）

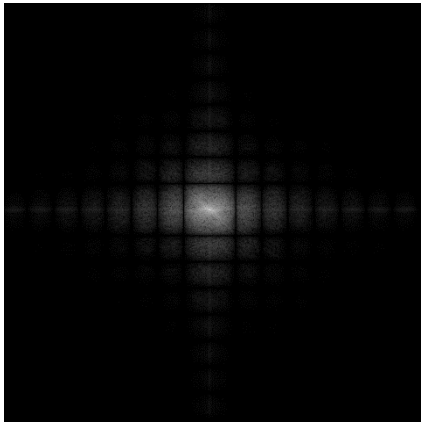
如果某些位置$H(u,v)$值很小，$\frac{N(u,v)}{H(u,v)}$会远超$F(u,v)$





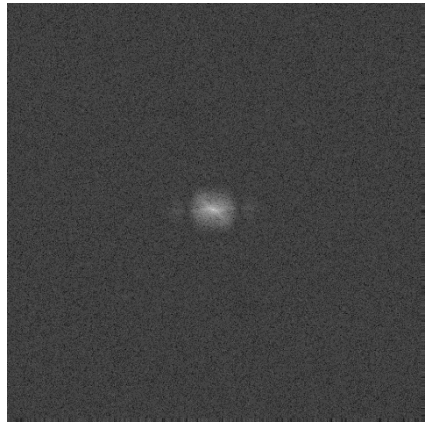


模糊



幅度谱

模糊+噪声



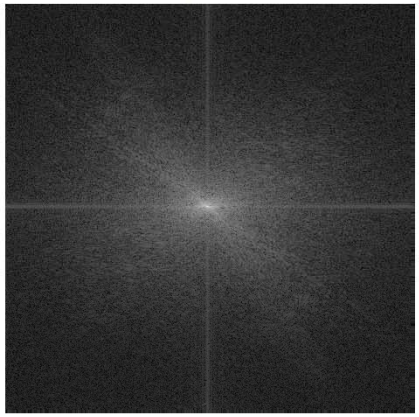
幅度谱

$$R(u,v) = \begin{cases} 1/H(u,v) & \text{if } |H(u,v)| > n \\ 1/n & \text{otherwise} \end{cases}$$

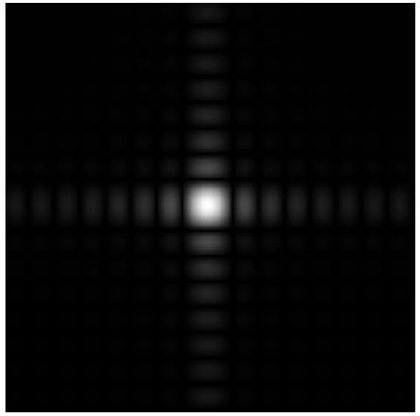
提高$n = 0.3$

f

|F|

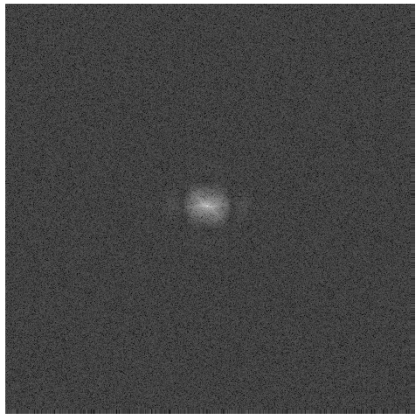


|H|



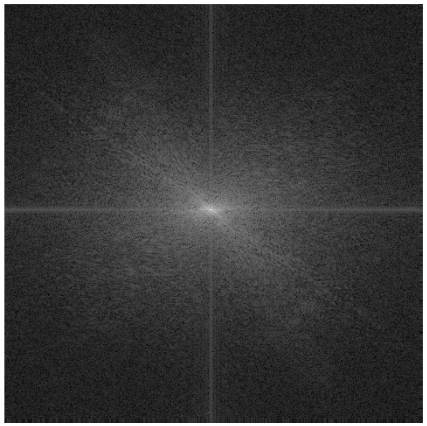
g

|G|

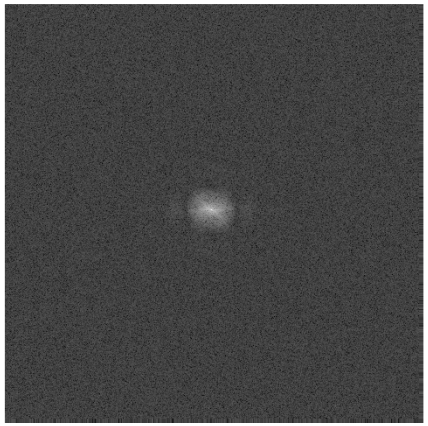


|R|

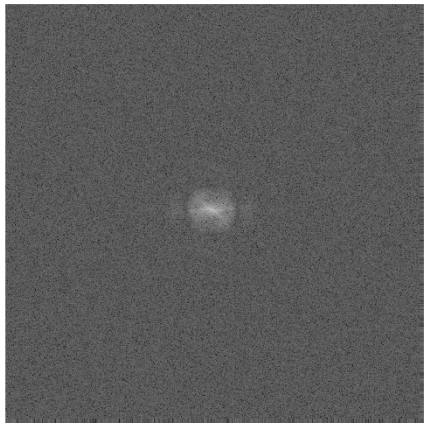
|F|



|G|



|F2|



f

g

f2

```
% invH
close all
f = im2double(imread('..\data\lena512.bmp'));
N = 512;
F = fft2(f);

% blur
b = 16;
h = ones(b,b)/b^2;
H = fft2(h,N,N);
G = F.*H;
noise_level = 0.05;
g = ifft2(G) + noise_level*randn(N,N);
G = fft2(g);

figure(1),
subplot(2,3,1), imshow(f), title('f')
subplot(2,3,2), imshow(log(abs(fftshift(F))+1),[]), title('|F|')
subplot(2,3,3), imshow(log(abs(fftshift(H))+1),[]), title('|H|')
subplot(2,3,4), imshow(g,[]), title('g')
subplot(2,3,5), imshow(log(abs(fftshift(G))+1),[]), title('|G|')

n=.3;
H2 = H;
H2(abs(H2)<n) = n;
R = ones(N,N)./H2;% restoration filter
F2 = G.*R;
f2 = abs(ifft2(F2));
subplot(2,3,6), imshow(log(abs(fftshift(R))+1),[]), title('|R|')

figure(2),
subplot(2,3,1), imshow(log(abs(fftshift(F))+1),[]), title('|F|')
subplot(2,3,2), imshow(log(abs(fftshift(G))+1),[]), title('|G|')
subplot(2,3,3), imshow(log(abs(fftshift(F2))+1),[]), title('|F2|')
subplot(2,3,4), imshow(f,[]), title('f')
subplot(2,3,5), imshow(g,[]), title('g')
subplot(2,3,6), imshow(f2,[]), title('f2')
```

维纳滤波（Wiener Filtering）

- 最小化误差平方均值：$E\left\{\left(f-\hat{f}\right)^2\right\}$
- 解为$\hat{F}(u,v)=\left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2+S_\eta(u,v)/S_f(u,v)}\right]G(u,v)$
- $S_\eta(u,v)$为噪声的功率谱，$S_f(u,v)$为原图像的功率谱
- 当噪声为0时，退化为逆滤波
- 通常$S_f(u,v)$是未知的

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

K的大小需多次尝试

```
% Wiener filtering
close all
I = im2double(imread('cameraman.tif'));
imshow(I);
title('Original Image (courtesy of MIT)');

LEN = 21;
THETA = 11;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');
figure, imshow(blurred)

noise_mean = 0;
noise_var = 0.0001;
blurred_noisy = imnoise(blurred, 'gaussian', noise_mean, noise_var);
figure, imshow(blurred_noisy)
title('Simulate Blur and Noise')

estimated_nsr = 0;
wnr2 = deconvwnr(blurred_noisy, PSF, estimated_nsr);
figure, imshow(wnr2)
title('Restoration of Blurred, Noisy Image Using NSR = 0')

estimated_nsr = noise_var / var(I(:));
wnr3 = deconvwnr(blurred_noisy, PSF, estimated_nsr);
figure, imshow(wnr3)
title('Restoration of Blurred, Noisy Image Using Estimated NSR');
```