

2022春



数字图像处理



# 图像复原与重建

曹劲舟

助理教授

深圳技术大学

大数据与互联网学院

2022年4月6日

# 课程概述

周次 Week	周学时 Week Hour	主要教学内容 Course Content	实验实践教学内容 Exercise/Experiment
1	2	图像处理概述及图像处理软件基础	
2	2	图像处理概述及图像处理软件基础	
3	2	灰度变换与空间滤波	
4	2	灰度变换与空间滤波	
5	2+4	频率域滤波	图像增强实验
6	2+4	频率域滤波	图像傅里叶变换实验

# 课程概述

<https://www.aojz.cn/courses/dip/>

7	2+4	图像复原与重建	图像的几何变换实验
8	2+4	图像复原与重建	图像复原实验
9	2+4	图像分割	图像分割实验
10	2+4	图像压缩	数字图像压缩与解压
11	2+4	图像压缩	数字图像压缩与解压
12	2+4	彩色图像处理	基于嵌入式设备的图像处理
13	2+4	彩色图像处理	基于嵌入式设备的图像处理
14	2	表示与描述	
15	2	表示与描述	
16	2	基于嵌入式硬件系统的图像处理	
17	2	基于嵌入式硬件系统的图像处理	
18	2	复习与答疑	

# 概述：图像退化、复原与重建

- **图像复原 (image restoration) 定义：**利用“图像”退化过程的先验知识，去**恢复已被退化图像**的本来面目。
  - 例如：对图像资料进行大气影响的校正以及对由于设备原因造成的模糊等的改正，将退化图像重建成接近于或完全无退化的原始理想图像的过程。
- **图像重建 (image reconstruction) 定义：**根据场景的投影数据获取场景中物质分布的信息或重构二、三维场景。
  - 例如：卫星影像立体重建；医学影像重建等。

# 图像复原和图像增强的区别

---

图像增强**不考虑图像是如何退化的**，而是试图采用各种技术来增强图像的视觉效果。因此，图像增强可以不顾增强后的图像是否失真，只要看得舒服就行。

而图像复原就完全不同，需知道**图像退化的机制和过程等先验知识**，据此找出一种相应的逆处理方法，从而得到复原的图像。

如果图像已退化，应先作复原处理，再作增强处理。二者的目的都是为了改善图像的质量。

- **图像退化定义：**图像在**形成、传输和记录**过程中，由于成像系统、传输介质和设备的不完善，使图像的质量变坏，这一过程称为**图像退化**。
- **引起图像退化的原因：**
  - ◆ 成像系统的散焦
  - ◆ 成像设备与物体的相对运动
  - ◆ 成像器材的固有缺陷
  - ◆ 外部干扰等

# 图像退化的数学模型

- 成像系统：输入—— $f(x, y)$ ，输出—— $g(x, y)$
- 成像系统作用—— $H[\cdot]$
- 退化图像为：

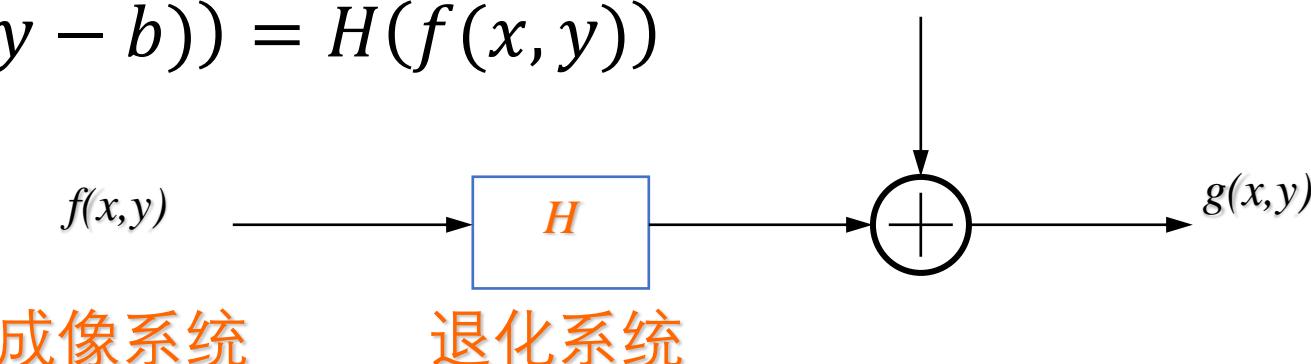
$$g(x, y) = H[f(x, y)]$$

- 线性位移不变成像系统**

退化系统  $H$  满足:  $\forall$  常数  $k, a, b$  有

- (1) 线性:  $H(kf + g) = kH(f) + H(g)$
- (2) 位移不变性:  $H(f(x - a, y - b)) = H(f(x, y))$

计算结果仅取决于输入值，与位置无关

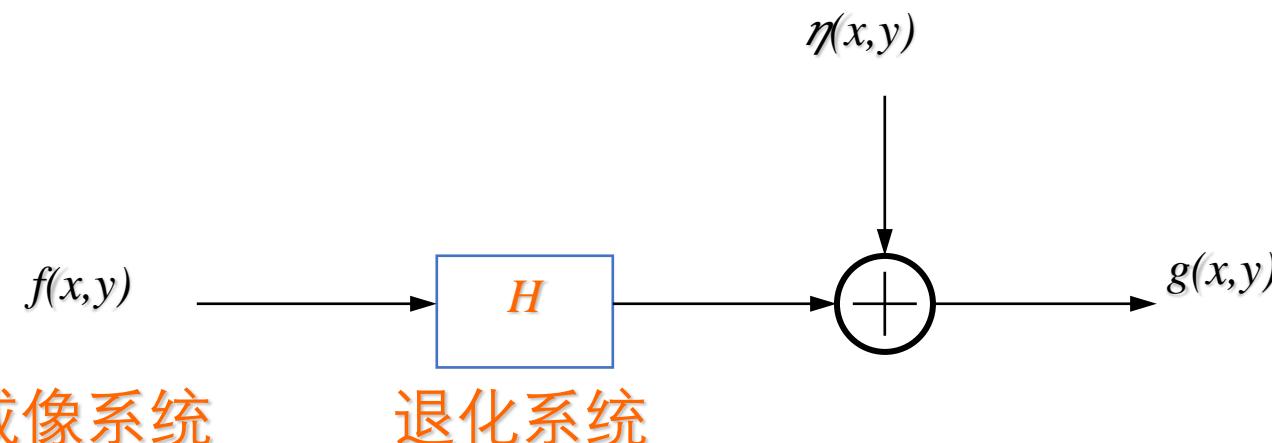


# 图像退化的数学模型

- 线性位移不变成像系统 **图像退化模型**

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

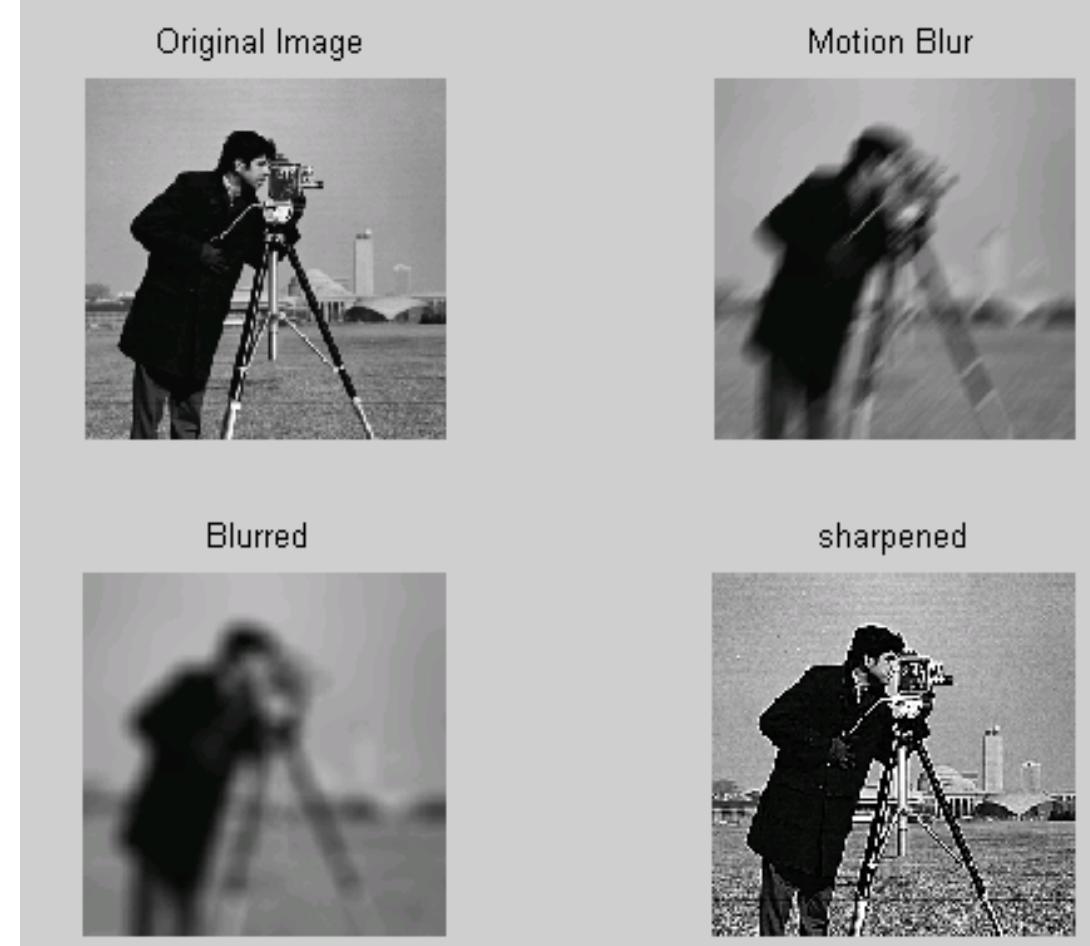
- $g(x, y)$  —— 退化图像
- $f(x, y)$  —— 理想图像 (输入图像)
- $h(x, y)$  —— 点扩散函数 (退化函数)
- $\eta(x, y)$  —— 加性噪声
- 空间域  $g(x, y) = h(x, y) \odot f(x, y) + \eta(x, y)$
- 频率域  $G(x, y) = H(x, y)F(x, y) + N(x, y)$



# MATLAB应用实例

## 模糊图像

```
I=imread('cameraman.tif');
subplot(2,2,1);
imshow(I);title('Original Image');
H=fspecial('motion',20,45);
MotionBlur=imfilter(I,H,'replicate');
subplot(2,2,2);
imshow(MotionBlur);title('Motion Blur');
H=fspecial('disk',10);
blurred=imfilter(I,H,'replicate');
subplot(2,2,3);
imshow(blurred);title('Blurred ');
H=fspecial('unsharp');
sharpened=imfilter(I,H,'replicate');
subplot(2,2,4);
imshow(sharpened);title('sharpened');
```



# 噪声模型

---

- **图像噪声**是指造成图像失真、质量下降的图像信号，在图像上表现为引起较强视觉效果的孤立像元点或像元块。
- 噪声来源：
  - **图像获取过程**：成像传感器自身性能和环境条件（热噪声、光照水平等）。
  - **图像传输过程**：信号可能被传输信道中的干扰污染，如通过无线网络传输的图像会受到光或其它大气因素的干扰。

# 噪声模型

---

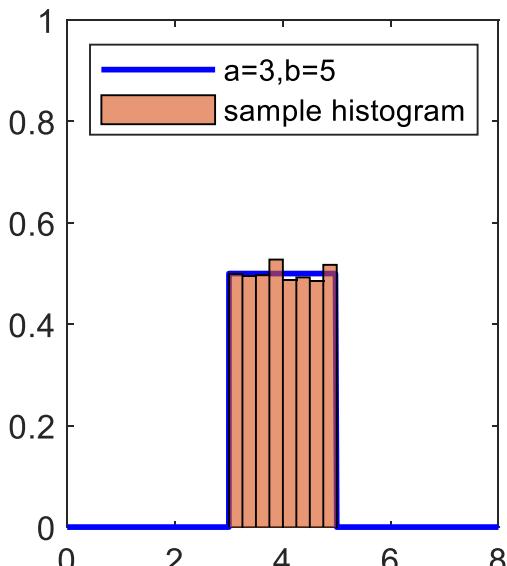
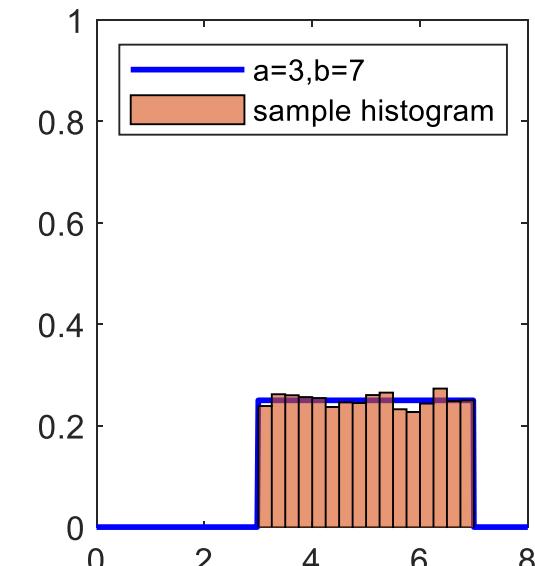
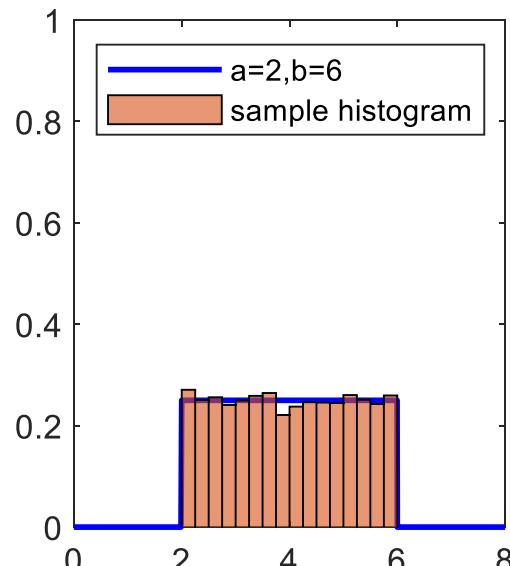
- 空间噪声**不能被预测**，但是可以是由**概率密度函数（PDF）来统计描述**，因此可采用相当简单的概率函数来生成噪声随机数。
- 一些重要的空间噪声类型
  - 高斯噪声
  - 瑞利噪声
  - 伽马（爱尔兰）噪声
  - 指数分布噪声
  - 均匀分布噪声
  - 脉冲噪声（椒盐噪声）

# 空间域噪声

## 1. 均匀噪声

$$p(z) = \begin{cases} \frac{1}{b-a}, & a \leq z \leq b \\ 0, & \text{其他} \end{cases}$$

其中， $z$ 表示灰度， $z$ 的平均值=  $(a + b)/2$ ， $z$ 的方差=  $\frac{(b-a)^2}{12}$ 。



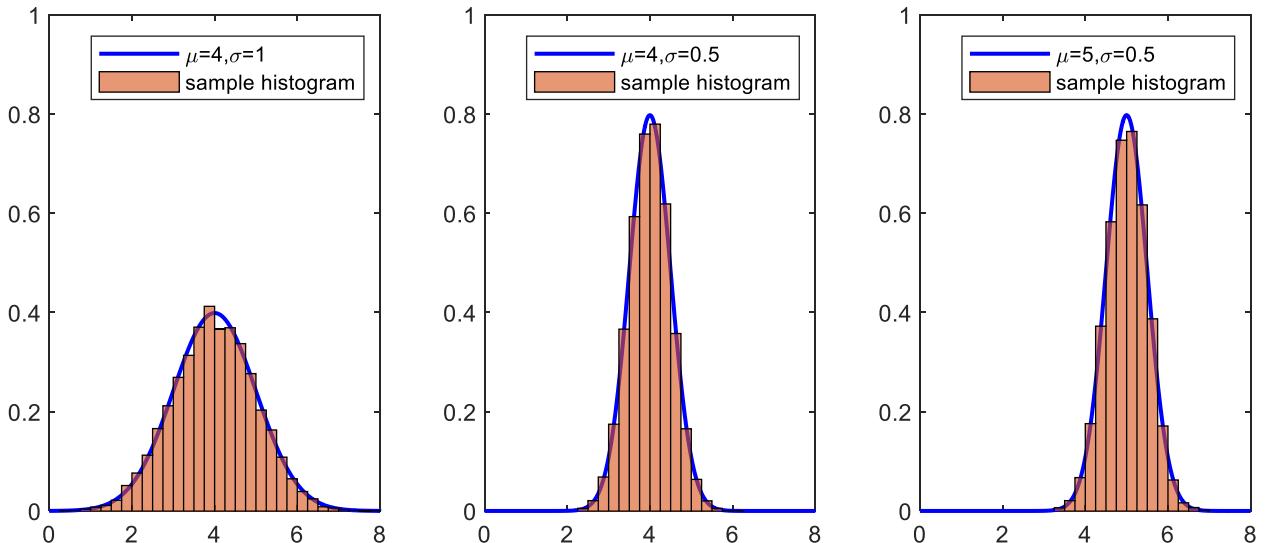
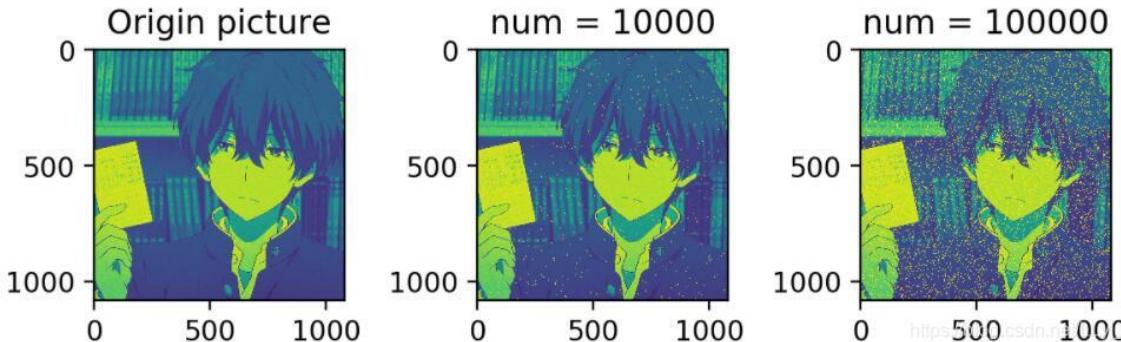
# 空间域噪声

## 2. 高斯噪声

高斯噪声的概率密度函数(PDF):

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}}, \quad -\infty < z < +\infty$$

其中 $z$ 表示灰度， $\bar{z}$ 是灰度 $z$ 的平均值， $\sigma$ 是灰度 $z$ 的标准差。



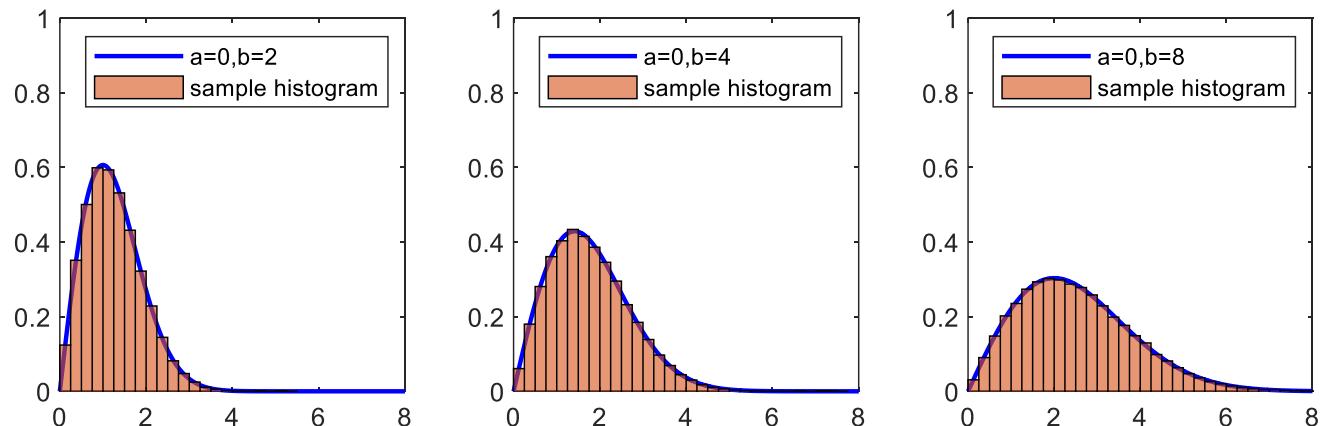
# 空间域噪声

## 3. 瑞利噪声

$$p(z) = \begin{cases} \frac{2}{b}(z - a)e^{-\frac{(z-a)^2}{b}}, & z \geq a \\ 0, & z < a \end{cases}$$

其中， $z$ 表示灰度， $z$ 的平均值=  $a + \sqrt{\pi b / 4}$ ， $z$ 的方差=  $b(4 - \pi)/4$ ， $a$ 表示距离原点的位移。

当一个随机二维向量的两个分量呈独立的、有着相同的方差的正态分布时，这个向量的模呈瑞利分布。



# 空间域噪声

## 4.脉冲噪声 (椒盐噪声)

一般是由图像传感器、传输信道及解码处理等产生的黑白相间的亮暗点噪声。

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

- 如果  $P_a$  或  $P_b$  为零，则脉冲噪声称为单极脉冲。
- 如果  $P_a$  或  $P_b$  均不为零，则脉冲噪声称为双极脉冲噪声或椒盐噪声。

# 空间域噪声

## 4. 椒盐噪声

- 椒盐噪声是指两种噪声： 椒噪声 + 盐噪声。 盐噪声一般是白色噪声， 椒噪声一般是黑色噪声， 前者高灰度噪声， 后者属于低灰度噪声， 一般两种噪声同时出现， 呈现在图像上就是黑白杂点。
- 以8比特图像为例， 出现0的位置分配0值， 出现255的位置分配255， 其他值保持不变。

$$p(z) = \begin{cases} P_a & z = 0(\text{胡椒}) \\ P_b & z = 2n-1(\text{盐粒}) \\ 1-(P_a+P_b) & z=k \end{cases}$$

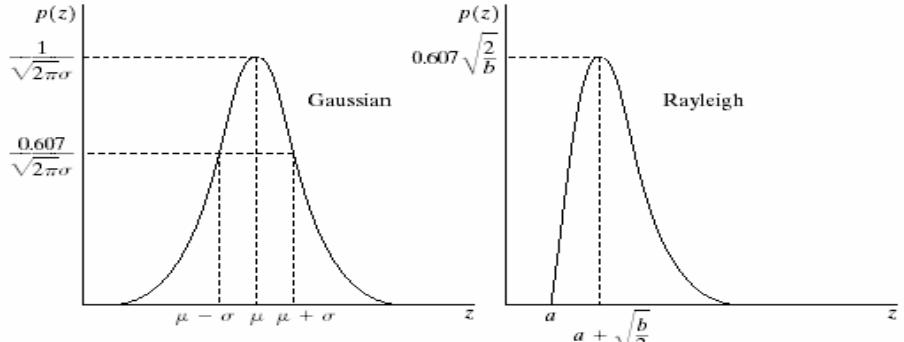
- 脉冲噪声可以为正， 也可为负； 负脉冲以黑点（胡椒点）出现， 正脉冲以白点（盐点）出现。
- 一个像素被椒盐噪声污染的概率为：  $P = P_a + P_b$



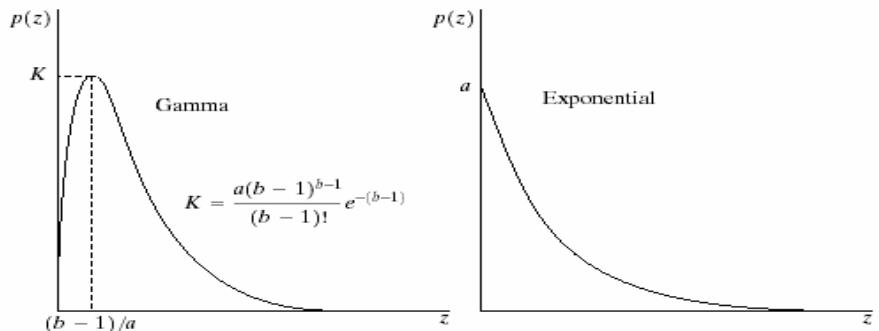
# 空间域噪声

►一些重要噪声的概率密度函数(PDF)

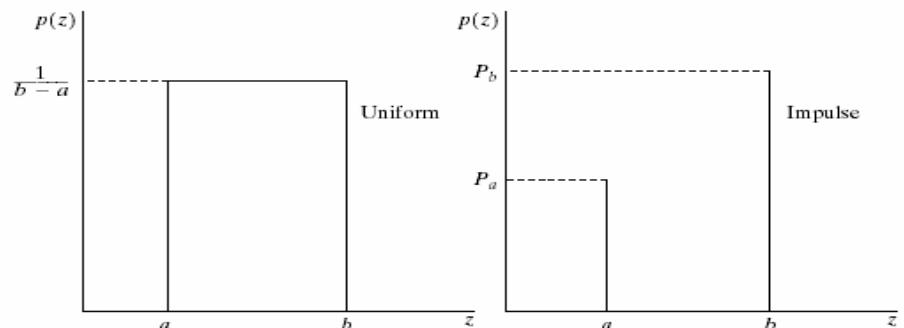
高斯



伽马



均匀



瑞利

指数

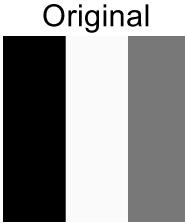
椒盐

Name	PDF
Uniform	$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } 0 \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$
Gaussian	$p(z) = \frac{1}{\sqrt{2\pi}b} e^{-(z-a)^2/2b^2} \quad -\infty < z < \infty$
Lognormal	$p(z) = \frac{1}{\sqrt{2\pi}bz} e^{-[\ln(z)-\bar{\ln}^2]/2b^2} \quad z > 0$
Rayleigh	$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & z \geq a \\ 0 & z < a \end{cases}$
Exponential	$p(z) = \begin{cases} ae^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases}$
Erlang	$p(z) = \frac{a^b z^{b-1}}{(b-1)!} e^{-az} \quad z \geq 0$
Salt & Pepper <sup>‡</sup>	$p(z) = \begin{cases} P_p & \text{for } z = 0 \text{ (pepper)} \\ P_s & \text{for } z = 2^n - 1 \text{ (salt)} \\ 1 - (P_p + P_s) & \text{for } z = k \quad (0 < k < 2^n - 1) \end{cases}$

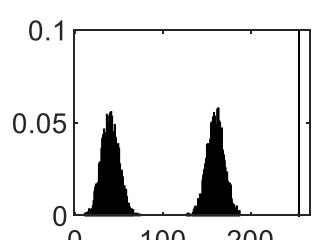
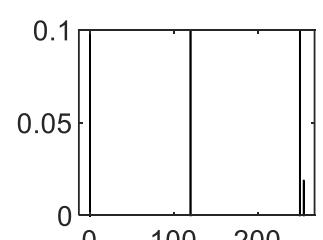
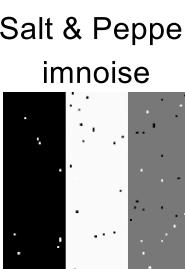
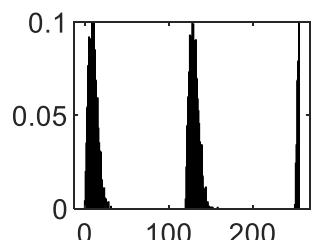
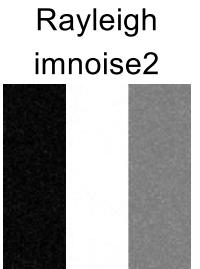
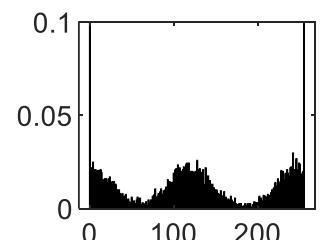
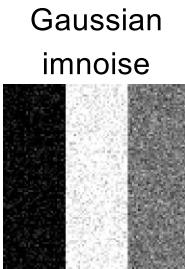
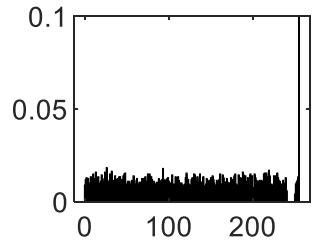
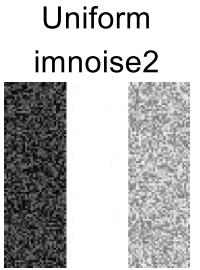
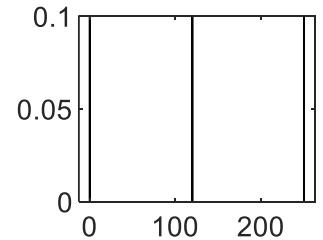
# 空间域噪声

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

- ✓ 用于噪声模型的测试图
- ✓ 由简单、恒定的区域组成
- ✓ 仅有3个灰度级的变化



样本噪声图像及其直方图



## 结论

- 上述噪声图像的直方图和它们的概率密度函数曲线对应相似
- 前面5种噪声的图像并没有显著不同，但它们的直方图具有明显的区别

# 用imnoise函数为图像添加噪声

图像处理工具箱采用 **imnoise** 函数，使噪声污染一幅图像，基本语法

**g = imnoise(f, type, parameters)**

- 注意：函数 `imnoise` 在为图像添加噪声之前，将图像转换为范围在[0,1]的double类。
- 将均值为m、方差为var的高斯噪声加到图像上

```
>> f = imread('jimei2.jpg' );
>> d = im2double(f); //转换为范围在[0,1]的double类
>> g = imnoise(d, 'gaussian', 0.1, 0.05); //这里均值为0.1， 方差为0.05
>> subplot(1,2,1),imshow(d),title('Original Image' );
>> subplot(1,2,2),imshow(g,[]),title('Polluted By Gaussian');
```

# MATLAB应用实例

## 加噪声

```
w = imread('x.jpg');
f = im2double(w); %转换为范围在[0,1]
g = imnoise(f, 'gaussian', 0.1,0.01); %方差为0.01
g = imnoise(f, 'gaussian', 0.1,0.05); %方差为0.05
g = imnoise(f, 'gaussian', 0.1,0.2); %方差为0.2
subplot(2, 2, 1), imshow(w), title('原图像');
subplot(2, 2, 2), imshow(g,[]), title('均值为0.1方差为0.01的高斯噪声加到图像');
subplot(2, 2, 3), imshow(g,[]), title('均值为0.1方差为0.05的高斯噪声加到图像');
subplot(2, 2, 4), imshow(g,[]), title('均值为0.1方差为0.2的高斯噪声加到图像');
```



# MATLAB应用实例

## 加噪声

%图像加噪

```
w = imread('D:\数字图像处理\第四章学习\tree.jpg');
f = im2double(rgb2gray(w)); %把图像变为灰度图像
g = imnoise (f, 'salt & pepper',0.12); %d噪声密度
subplot(1, 3, 1), imshow(w), title('原图像');
subplot(1, 3, 2), imshow(f), title('原图像的灰度图像');
subplot(1, 3, 3), imshow(g), title('椒盐噪声加到图像');
```

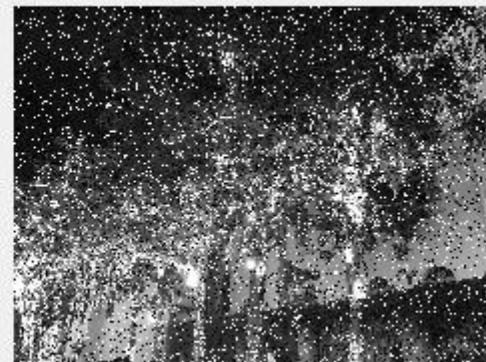
原图像



原图像的灰度图像



椒盐噪声加到图像



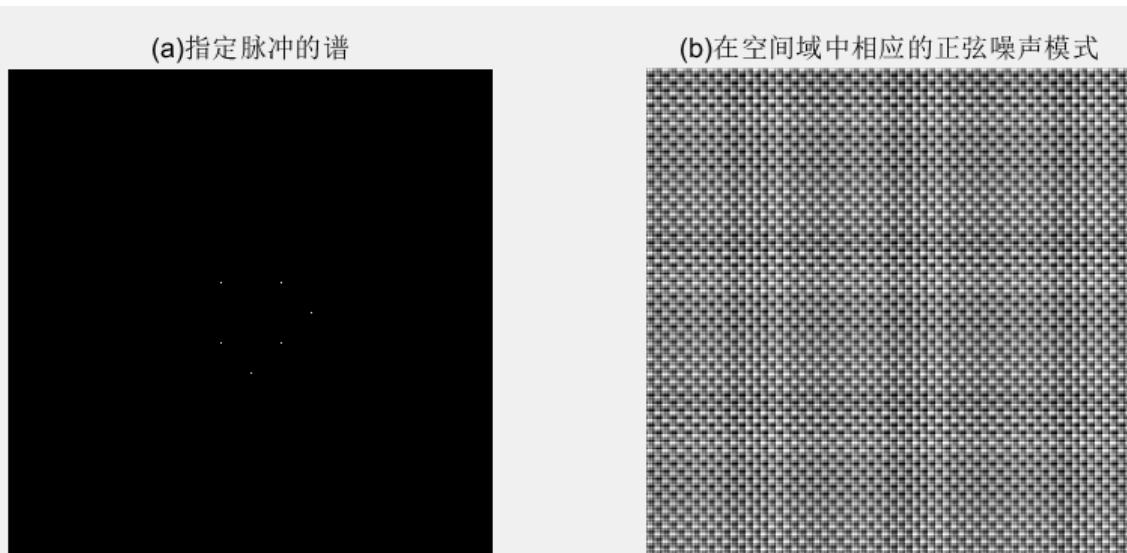
# 周期噪声

**周期噪声：**空间上有周期性变化的噪声，是在图像获取中从电力或机电干扰中产生，可以通过频率域滤波显著减少

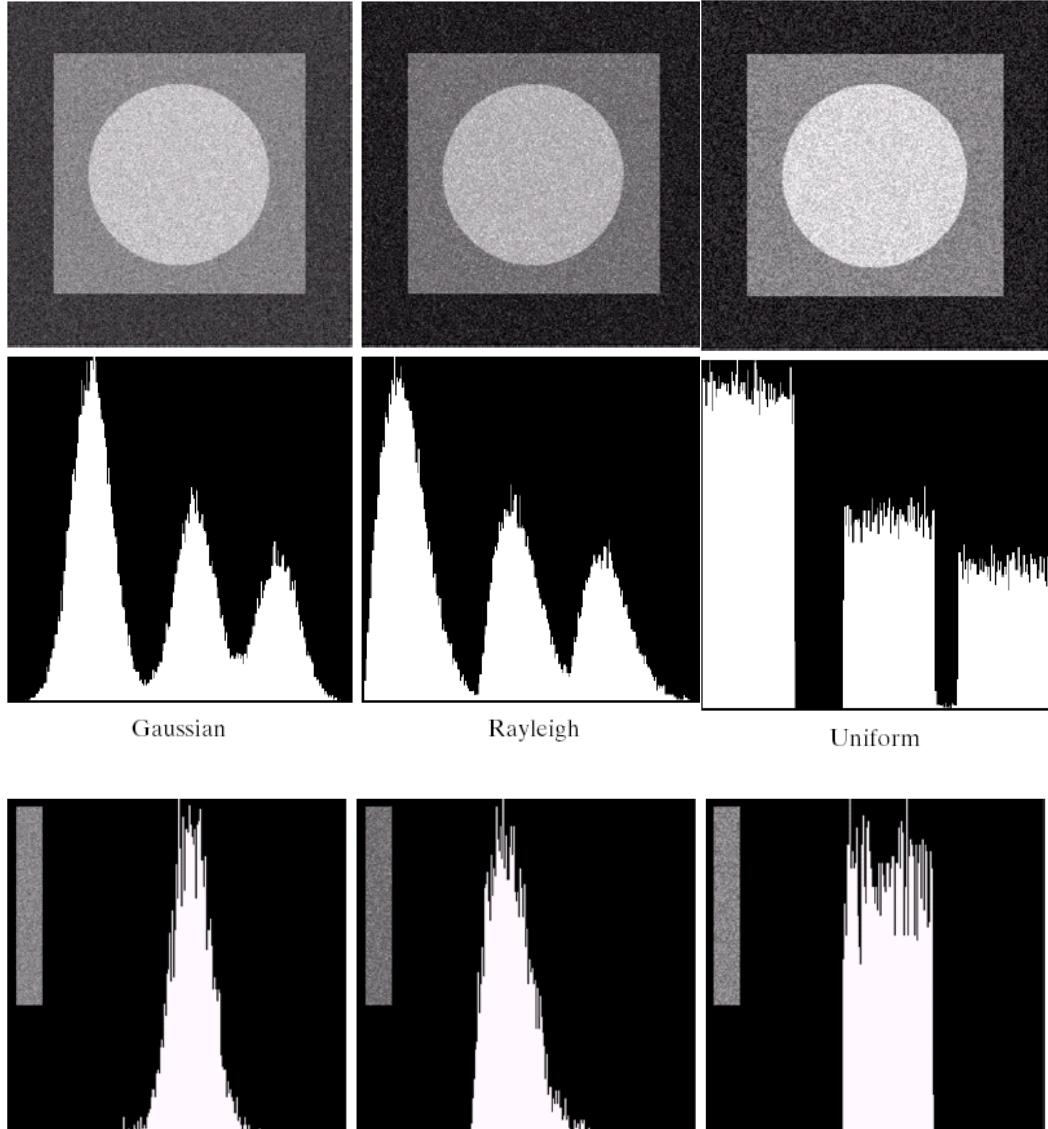
模型：离散的二维正弦波

$$r(x, y) = A \sin\left[\frac{2\pi u_0(x + B_x)}{M} + \frac{2\pi v_0(y + B_y)}{N}\right]$$

```
>> C = [0 64; 0 128; 32 32; 64 0; 128 0; -32 32];
% K对频域坐标 (u, v) , 表示频域中脉冲的位置
>> [r, R, S] = imnoise3(512,512,C);
% 空间正弦噪声模式r, 其傅里叶变换R和频谱S
>> subplot(1,2,1),imshow(S,[],),title('Specified
Pulse');
>> subplot(1,2,2),imshow(r,[],),title('Spatial
sinusoidal noise');
```



# 估计噪声参数



无法使用噪声直方图  
推测噪声PDF

最好选取图像中的一  
个无特色的背景区域，  
以便该区域灰度值的  
变化主要由噪声引起  
**(ROI)**，来进行推  
测

# 图像复原

## • 空间滤波：仅有噪声的复原

- 均值滤波器（算数平均、几何均值、谐波平均、反谐波平均...）
- 统计排序滤波器（中值滤波、最大最小值、中点、修正阿尔法值...）
- 自适应滤波器（自适应局部降噪、自适应中值...）

## • 频率域滤波：削减周期噪声

- 陷波滤波器

# 空间域滤波复原

- 当唯一的退化是噪声时，

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



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

$g(x, y)$  —— 退化图像

$f(x, y)$  —— 理想图像（输入图像）

$h(x, y)$  —— 点扩散函数

$\eta(x, y)$  —— 加性噪声

- 但是，噪声项未知，不能从 $g(x, y)$ 减去噪声
- 可以选择空间滤波方法进行图像复原

# 空间噪声滤波器

## ➤ 图像复原的空间滤波器

### ➤ 均值滤波器

算术均值滤波器、几何均值滤波器、调和均值滤波器、逆调和均值滤波器

### ➤ 顺序统计滤波器

中值滤波器、最大值滤波器、最小值滤波器、中点滤波器、修正后的阿尔法均值滤波器

### ➤ 自适应滤波器

自适应局部噪声消除滤波器、自适应中值滤波器

# 均值滤波器

- 算数均值滤波器

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

其中 $S_{xy}$ 表示中心为 $(x, y)$ 、大小为 $m \times n$ 的矩形子图窗口（邻域）的一组坐标。 $r, c$ 是邻域 $S_{xy}$ 中包含的像素的行坐标和列坐标。

- **特点：**

- 平滑了一幅图像的局部变化，在模糊了结果的同时减少了噪声

# 均值滤波器

---

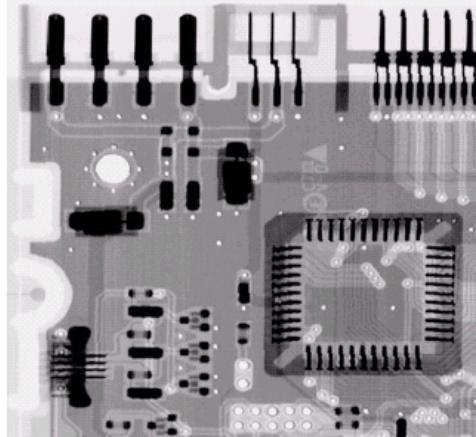
- **几何均值滤波器**

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

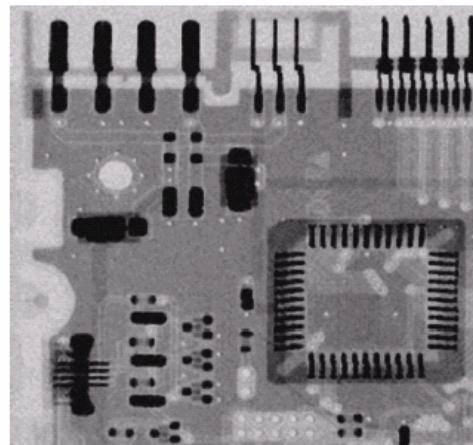
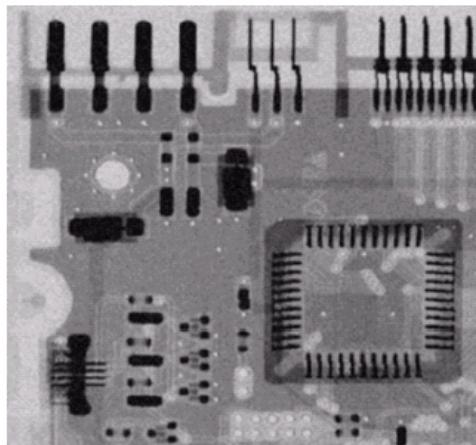
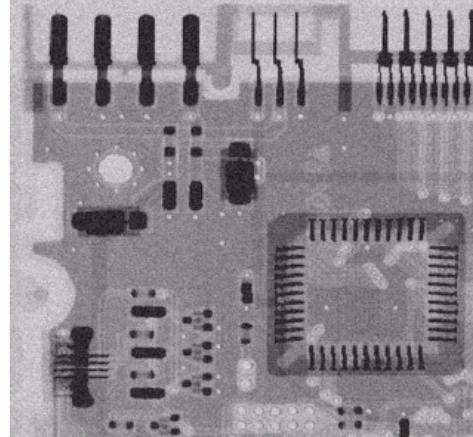
- **特点：**
  - 几何均值滤波器所达到的平滑度可以与算术均值滤波器相比
  - 几何均值滤波器在滤波过程中，与算术均值滤波器相比，会丢失更少的图像细节——相对锐化

# 均值滤波器

原图



被均值为0，方差为400的高斯噪声污染



3×3算术均值滤波器

3×3几何均值滤波器，图像更清晰

a  
b  
c  
d

**FIGURE 5.7** (a) X-ray image.  
(b) Image corrupted by additive Gaussian noise. (c) Result of filtering with an arithmetic mean filter of size  $3 \times 3$ . (d) Result of filtering with a geometric mean filter of the same size. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

# 均值滤波器

- 调和均值滤波器

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

- 特点:

- 调和均值滤波器对于“盐”噪声效果好，但不适用于“胡椒”噪声。
- 善于处理高斯噪声。

# 均值滤波器

---

## • 逆调和平均滤波器

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

- Q称为滤波器的阶数。
- 适用于消除椒盐噪声的影响：当Q为正数时，消除“胡椒”噪声；当Q为负数时，消除“盐”噪声，但不能同时消除“椒盐”噪声。
- 当Q=0，转变为算术均值滤波器。
- 当Q=-1，转变为调和均值滤波器。

# 均值滤波器

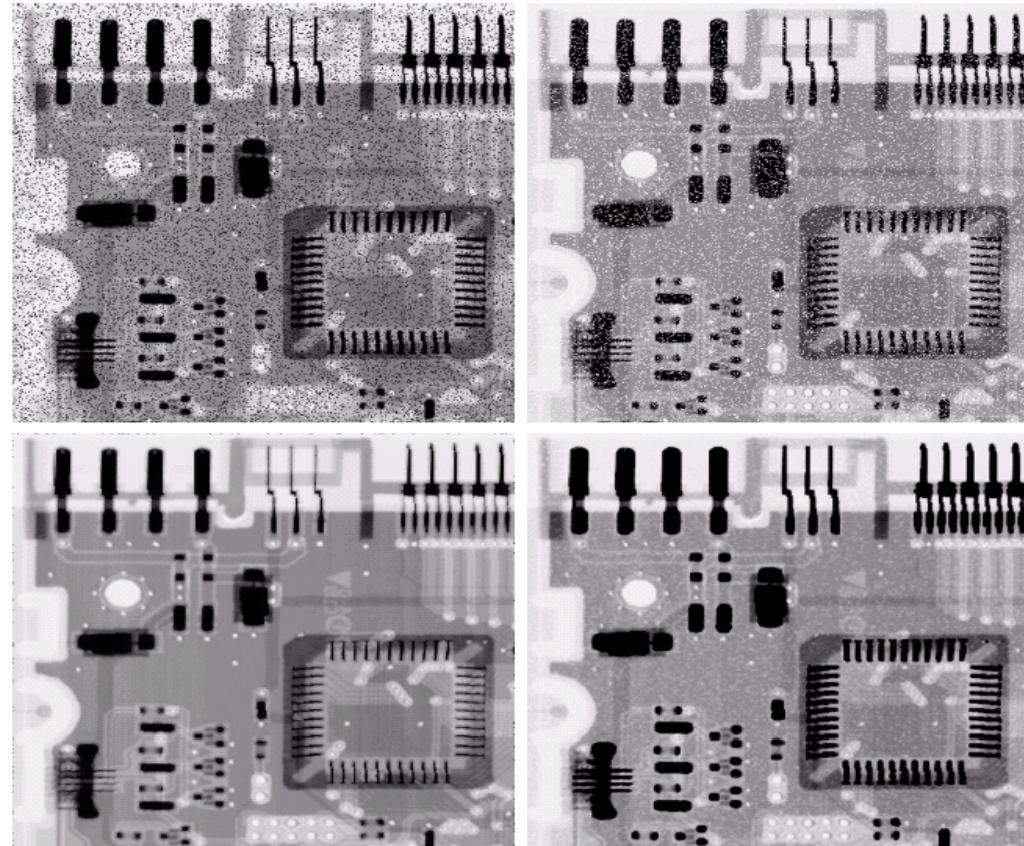
prob. = 0.1

“胡椒”噪声干扰图像

“盐”噪声干扰图像

a b  
c d

**FIGURE 5.8**  
 (a) Image corrupted by pepper noise with a probability of 0.1. (b) Image corrupted by salt noise with the same probability. (c) Result of filtering (a) with a  $3 \times 3$  contraharmonic filter of order 1.5. (d) Result of filtering (b) with  $Q = -1.5$ .



$3 \times 3$ 大小,  $Q=1.5$ 的  
逆调和均值滤波器

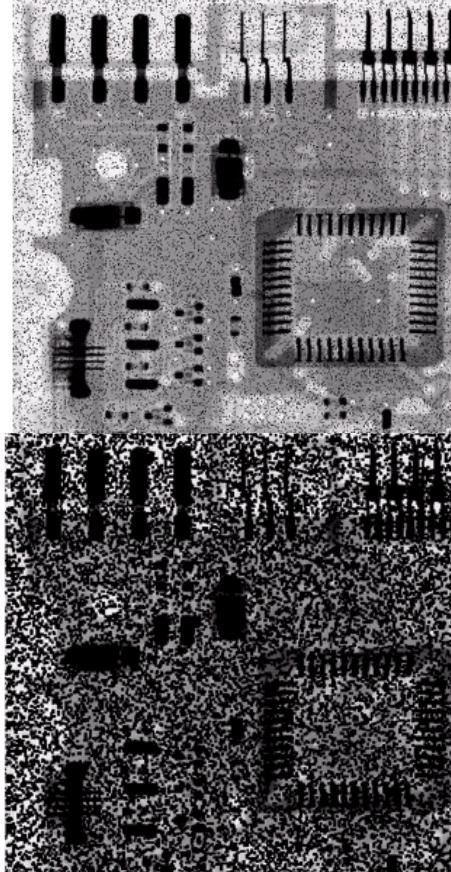
$3 \times 3$ 大小,  $Q=-1.5$ 的  
逆调和均值滤波器

# 均值滤波器

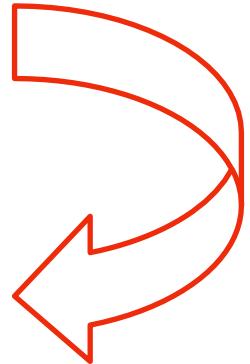
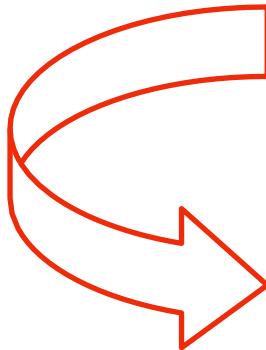
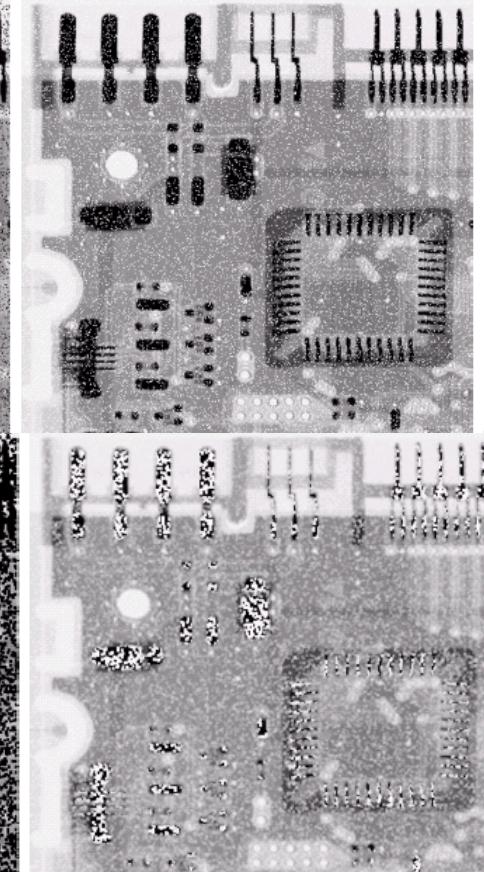
错误案例

prob. = 0.1

“胡椒”噪声干扰图像



“盐”噪声干扰图像



3×3大小，Q=-1.5的  
逆调和均值滤波器

3×3大小，Q=1.5的  
逆调和均值滤波器

DIP4-3-2.m

# 均值滤波器

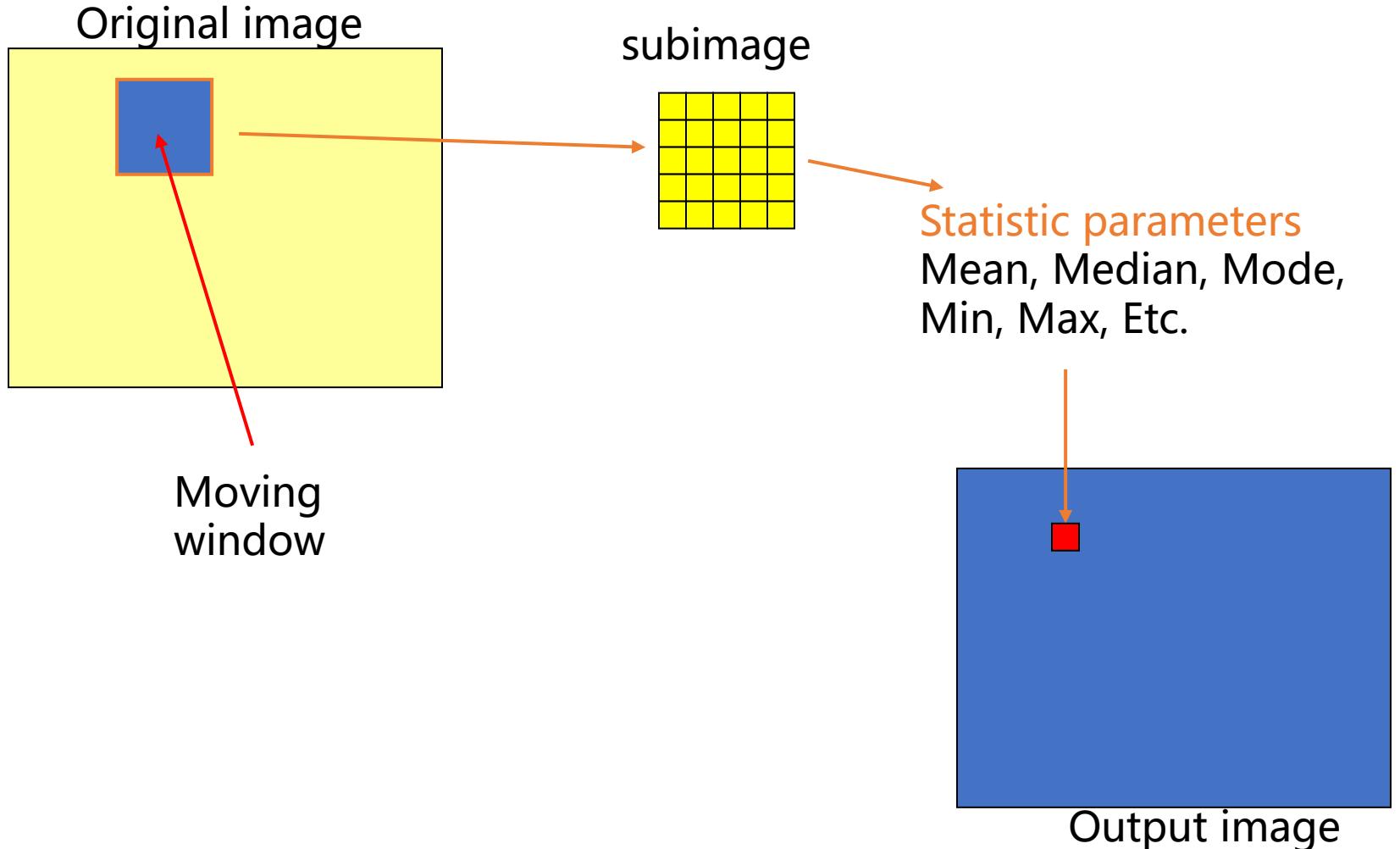
---

## • 总结

- 算术均值滤波器和几何均值滤波器适合处理高斯或均匀等随机噪声。
- 调和均值滤波器适合处理脉冲噪声（椒盐噪声），但必须事先知道噪声是“盐”还是“胡椒”，以便于选择合适的Q符号。

# 顺序统计滤波器

- 顺序统计滤波器的响应是基于由滤波器包围的图像区域中像素点的排序，任一点的响应由**排序结果**决定。



# 顺序统计滤波器

- **中值滤波器**

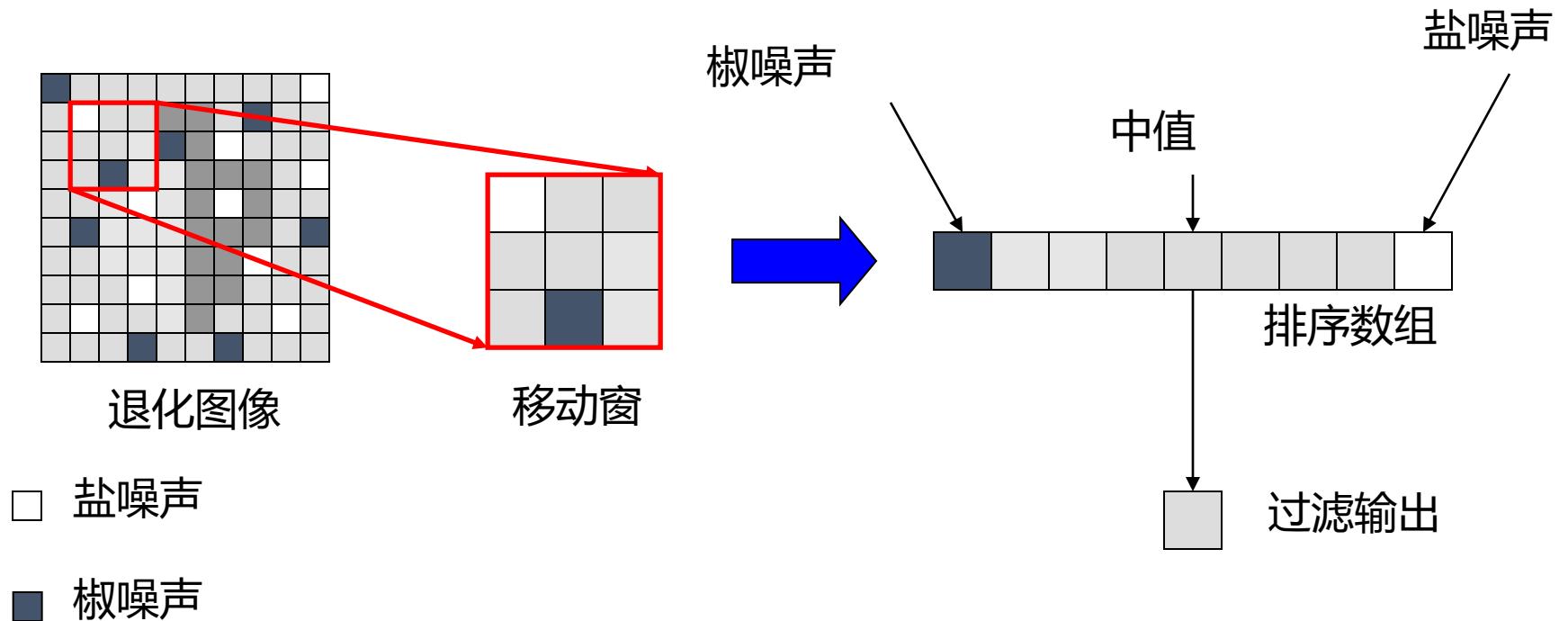
使用一个像素邻域中的灰度级的中值来替代该像素的值

$$\hat{f}(x, y) = \text{median}_{(r,c) \in S_{xy}} \{g(r, c)\}$$

- **特点：**

- 在相同尺寸下，比起均值滤波器引起的模糊少。
- 对椒盐噪声非常有效。

# 中值滤波器示例



通常, 脉冲噪声是极值。当在移动窗中进行排序时, 噪声像元通常位于数组的末端。

**噪声像元通常不可能是中值, 可有效去除**

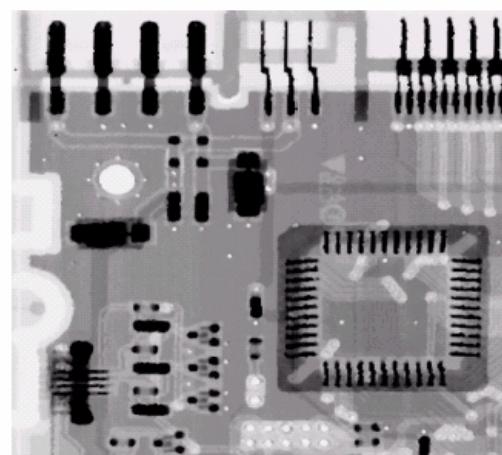
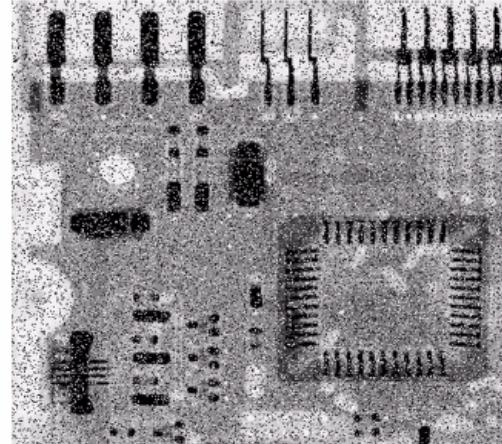
# 空间噪声滤波器

$P_a=P_b=0.1$  的脉冲噪声

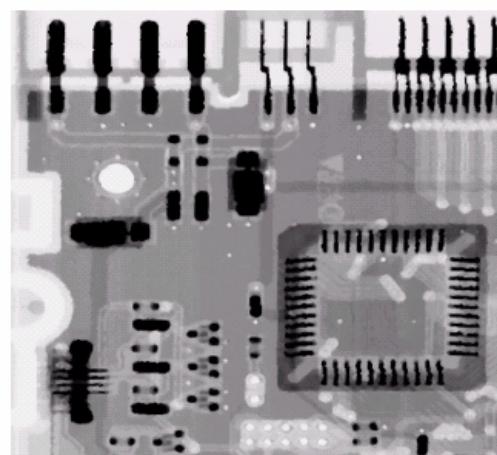
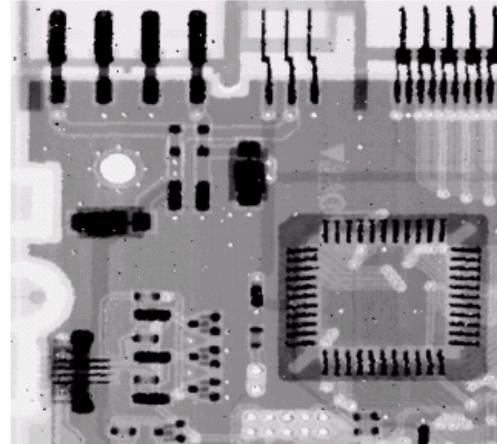
a b  
c d

FIGURE 5.10

(a) Image corrupted by salt-and-pepper noise with probabilities  $P_a = P_b = 0.1$ .  
 (b) Result of one pass with a median filter of size  $3 \times 3$ .  
 (c) Result of processing (b) with this filter.  
 (d) Result of processing (c) with the same filter.



$3 \times 3$  的中值滤波器



第二次中值滤波器处理

第三次中值滤波器处理，全部噪声消除

# 顺序统计滤波器

- **最大/最小值滤波器**

$$\hat{f}(x, y) = \max_{(r,c) \in S_{xy}} \{g(r, c)\}$$

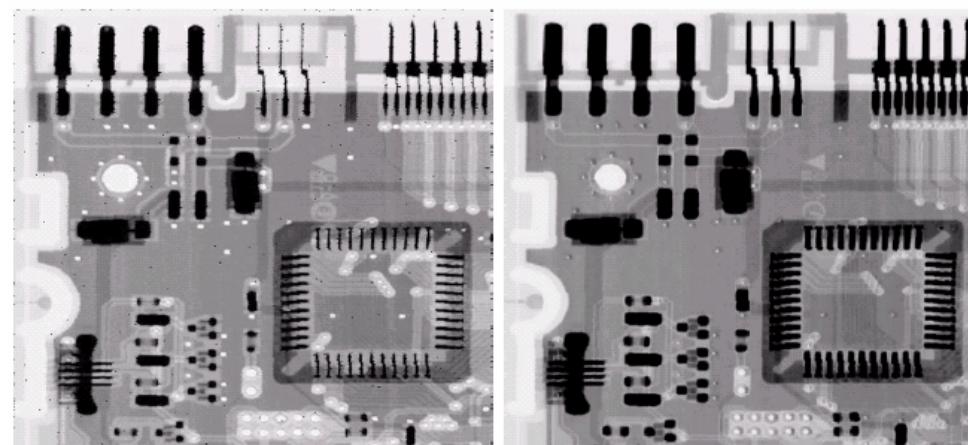
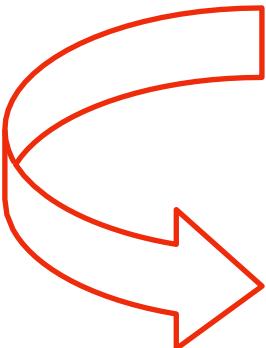
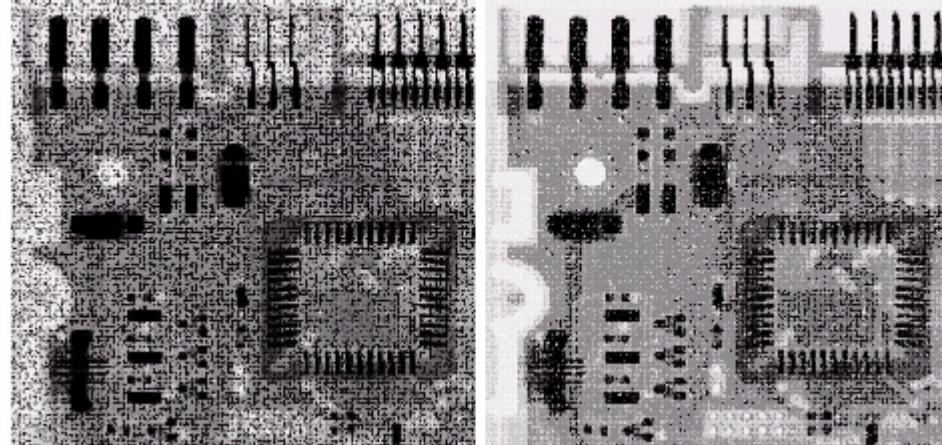
$$\hat{f}(x, y) = \min_{(r,c) \in S_{xy}} \{g(r, c)\}$$

- **特点：**

- 最大值滤波器：用于发现图像中的最亮点；可以有效过滤“胡椒”噪声  
(因为“胡椒”噪声是非常低的值)
- 最小值滤波器：用于发现图像中的最暗点；可以有效过滤“盐”噪声  
(因为“盐”噪声是非常高的值)

# 空间噪声滤波器

“胡椒”噪声干扰图像 “盐”噪声干扰图像



最大值滤波器处理

最小值滤波器处理

a b  
**FIGURE 5.11**  
(a) Result of  
filtering  
Fig. 5.8(a) with a  
max filter of size  
 $3 \times 3$ . (b) Result  
of filtering 5.8(b)  
with a min filter  
of the same size.

# 顺序统计滤波器

---

## • 中点滤波器

$$\hat{f}(x, y) = \frac{1}{2} [min_{(r,c) \in S_{xy}}\{g(r, c)\} + max_{(r,c) \in S_{xy}}\{g(r, c)\}]$$

## • 特点：

- 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。

# 顺序统计滤波器

## • 修正阿尔法滤波器

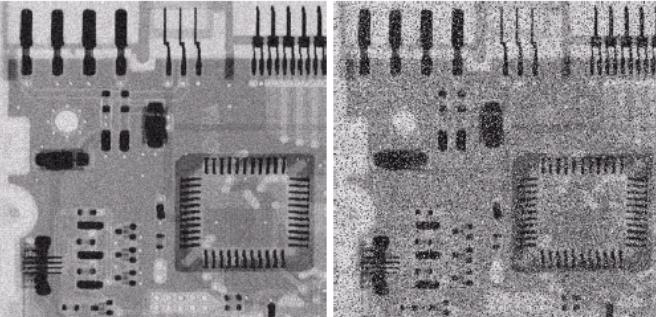
$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(r,c) \in S_{xy}} g_R(r, c)$$

- 在 $S_{xy}$ 邻域内删除 $g_R(r, c)$   $d/2$ 个最低灰度值和 $d/2$ 个最高灰度值。
- $g_R(r, c)$ 代表剩余的 $mn - d$ 个像素
- 当 $d=0$ , 退变为算术均值滤波器
- 当 $d=(mn-1)/2$ , 退变为中值滤波器
- 当 $d$ 取其它值时, 适用于包括**多种噪声**的情况下, 例如高斯噪声和椒盐噪声混合的情况

# 空间噪声滤波器

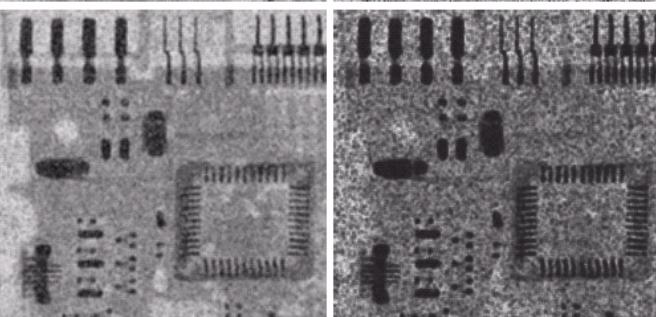
由于脉冲噪声的存在，算术均值和几何均值滤波器没有起到好的作用

均值为0，方差为 800  
的噪声干扰的图像



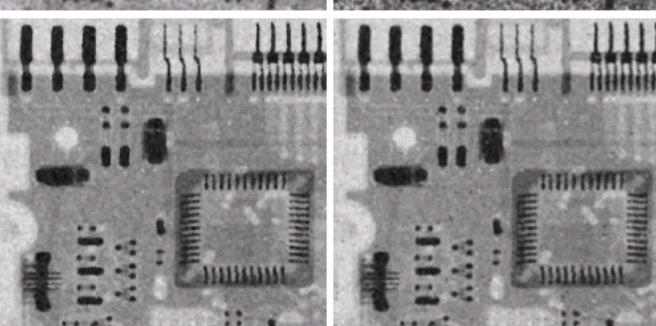
被 $P_a=P_b=0.1$ 的椒盐噪  
声叠加，进一步恶化

算术均值滤波器 ×



几何均值滤波器 ×

中值滤波器



d=5, 规格为 $5 \times 5$ 的修正  
后的阿尔法均值滤波器

✓

# 空间噪声滤波器

## 例 4.5 使用函数 **spfilt**。

图 4.5(a)所示的图像是一幅被概率只有 0.1 的胡椒噪声污染的 uint8 类图像。这幅图像是使用下面的命令生成的 [ f 是来自图 2.19(a) 的图像 ] :

```
>> [M, N] = size(f);
>> R = imnoise2('salt & pepper', M, N, 0.1, 0);
>> gp = f;
>> gp(R == 0) = 0;
```

图 4.5(b) 中的图像仅被盐粒噪声污染，它是使用如下语句生成的：

```
>> R = imnoise2('salt & pepper', M, N, 0, 0.1);
>> gs = f;
>> gs(R == 1) = 255;
```

过滤胡椒噪声的一种较好办法是，使用  $Q$  为正值的逆调和滤波器。图 4.5(c)是使用如下语句生成的：

```
>> fp = spfilt(gp, 'chmean', 3, 3, 1.5);
```

同样，盐粒噪声可以使用  $Q$  为负值的逆调和滤波器过滤：

```
>> fs = spfilt(gs, 'chmean', 3, 3, -1.5);
```

图 4.5(d)显示了结果。使用最大和最小滤波器可以得到类似的结果。例如，图 4.5(e) 和图 4.5(f) 所示的图像是通过使用如下命令分别由图 4.5(a) 和图 4.5(b) 生成的：

```
>> fpmax = spfilt(gp, 'max', 3, 3);
>> fsmin = spfilt(gs, 'min', 3, 3);
```

使用 **spfilt** 的其他解决方法可以类似的方式实现。

# 自适应空间滤波器

## • 自适应滤波器：考虑图像中不同点的特征变化

- 基于由 $m \times n$ 矩形窗口 $S_{xy}$ 定义的区域内图像的**统计特性**
- 与前述滤波器相比，**性能更优**，但也增加了**算法复杂度**
- **包括：**

- **自适应局部降噪滤波器**

(什么时候应该取均值？什么时候不应该？)

- **自适应中值滤波器**

(什么时候应该取中值？什么时候不应该？)

# 自适应局部降噪滤波器

- 滤波器响应基于以下4个统计特性：

- 全局的噪声统计特征：**方差** $\sigma^2$

- 邻域 $S_{xy}$ 内的统计特征：**局部均值** $\bar{z}_{S_{xy}}$

$$\bar{z}_{S_{xy}} = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s, t)$$

- 邻域 $S_{xy}$ 内的统计特征：**局部方差** $\sigma_{S_{xy}}^2$

$$\sigma_{S_{xy}}^2 = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} (g(s, t) - m_L)^2$$

# 自适应局部降噪滤波器

- 目的：保留边缘
- 运行规则：

1. 如果  $\sigma^2 = 0$ , -> 没有噪声

滤波器不需要操作，直接返回  $g(x,y)$ 。  $g(x,y)$  下零噪声的情况等同于  $f(x,y)$ 。

2. 若  $\sigma_{S_{xy}}^2 \gg \sigma^2$ , -> 边缘区域 (方差大通常是边缘，应保留)

滤波器返回接近  $g(x,y)$  的值

3. 若  $\sigma_{S_{xy}}^2$  接近  $\sigma^2$ , -> 图像物体内部

进行算数平均滤波，返回  $\bar{z}_{S_{xy}}$

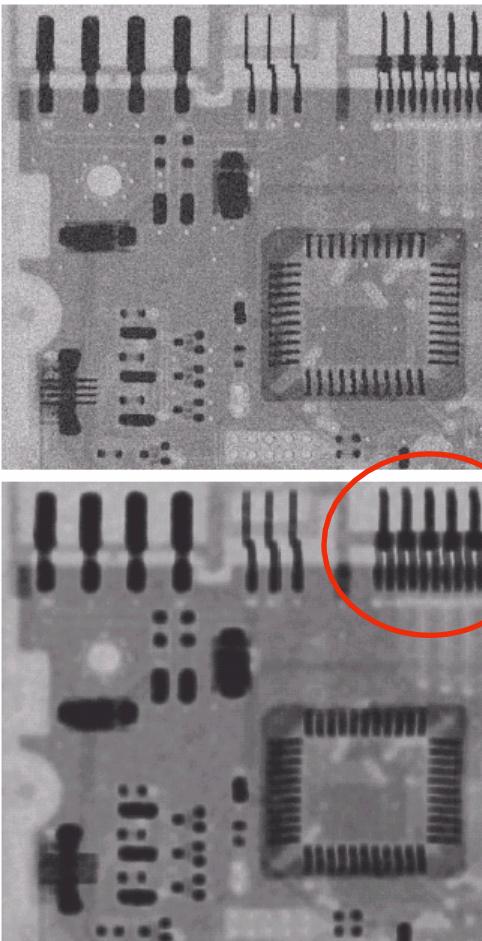
$$\hat{f}(x,y) = g(x,y) - \frac{\sigma^2}{\sigma_{S_{xy}}^2} [g(x,y) - \bar{z}_{S_{xy}}]$$

# 自适应局部降噪滤波器

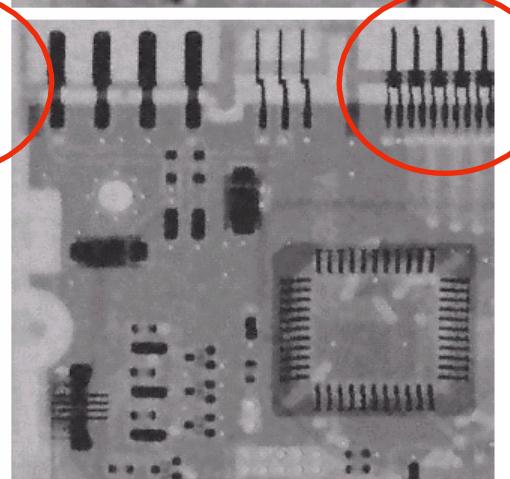
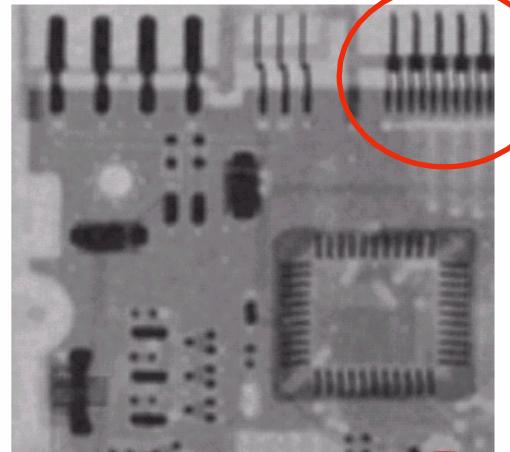
均值为0，方差为1000的高斯噪声

a  
b  
c  
d

**FIGURE 5.13**  
(a) Image corrupted by additive Gaussian noise of zero mean and variance 1000.  
(b) Result of arithmetic mean filtering.  
(c) Result of geometric mean filtering.  
(d) Result of adaptive noise reduction filtering. All filters were of size  $7 \times 7$ .



$7 \times 7$ 的算术均值滤波器



$7 \times 7$ 的几何均值滤波器

更加尖锐

$7 \times 7$ 的自适应滤波器

# 自适应中值滤波器

## 主要优势

- 传统中值滤波器只能处理空间密度不大的椒盐噪声 ( $p_a, p_b < 0.2$ )，而自适应中值滤波器可以处理具有**更大概率的椒盐噪声**
- 可以在平滑非椒盐噪声时保存细节，而传统中值滤波器无法做到

滤波器响应基于以下5个量：

- $z_{min}$ ：模板  $S_{xy}$  中灰度级的最小值
- $z_{max}$ ：模板  $S_{xy}$  中灰度级的最大值
- $z_{med}$ ：模板  $S_{xy}$  中灰度级的中值
- $z_{xy}$ ：坐标  $(x,y)$  上的灰度级
- $S_{max}$ ：模板  $S_{xy}$  允许的最大尺寸

# 自适应中值滤波器

- 操作步骤

**步骤A:** 若  $z_{min} < z_{med} < z_{max}$ , 则转到步骤B

说明  $z_{med}$  不是脉冲，直接输出

否则，增大  $S_{xy}$  的尺寸，

若  $S_{xy} \leq S_{max}$ , 则重复步骤A

否则，输出  $z_{xy}$

} 决定  $z_{med}$  是否是脉冲噪声

**步骤B:** 若  $z_{min} < z_{xy} < z_{max}$ , 则输出  $z_{xy}$

说明  $z_{xy}$  不是脉冲，直接输出

否则，输出  $z_{med}$

说明  $z_{xy} = z_{min}$  或  $z_{xy} = z_{max}$ , 去除噪声，  
输出  $z_{med}$

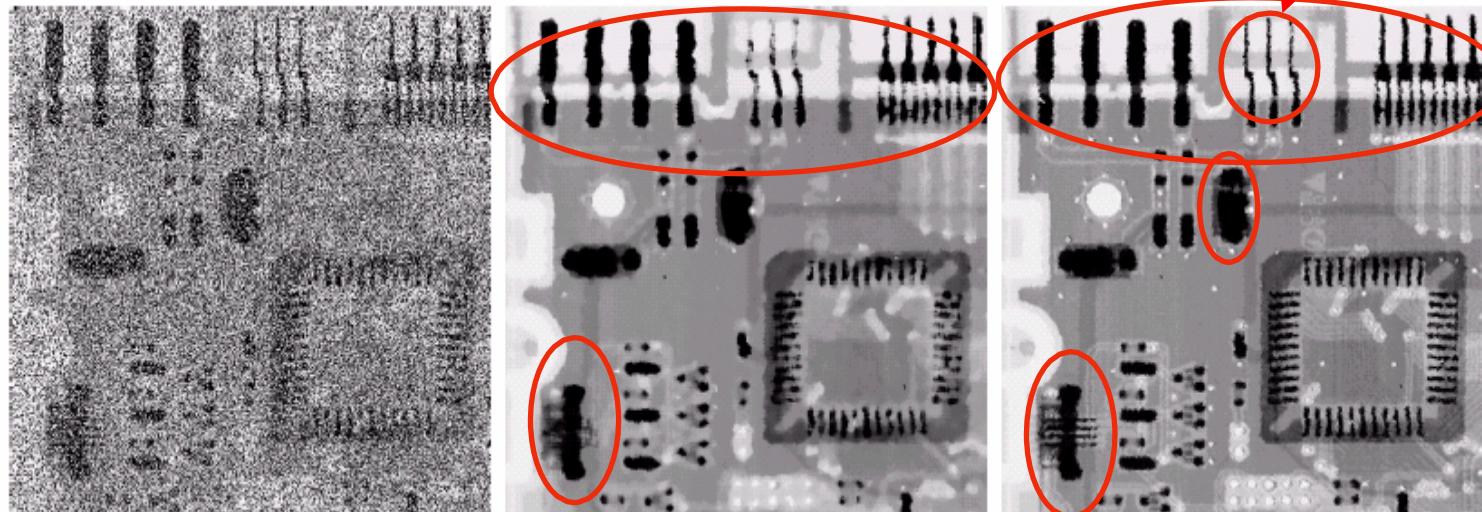
} 决定  $z_{xy}$  是否是脉冲噪声

# 自适应中值滤波器

- 自适应中值滤波器算法

更多小细节得到保留

$P_a = P_b = 0.25$ 的椒盐噪声     $7 \times 7$ 的中值滤波器     $S_{\max} = 7$ 的自适应中值滤波器



a b c

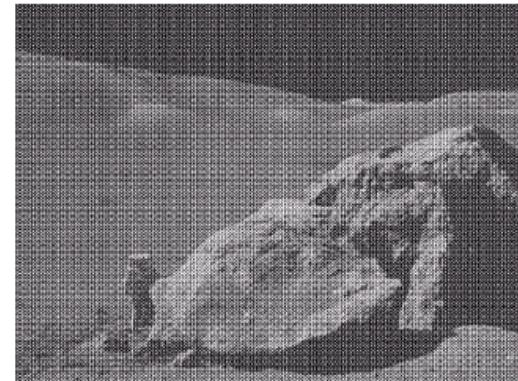
**FIGURE 5.14** (a) Image corrupted by salt-and-pepper noise with probabilities  $P_a = P_b = 0.25$ . (b) Result of filtering with a  $7 \times 7$  median filter. (c) Result of adaptive median filtering with  $S_{\max} = 7$ .

更锐化 ✓

# 频率域滤波

- **频率域滤波**: 从 $G(u, v)$ 频谱中将周期噪声减弱，再还原到时域
- 频率域滤波滤除周期噪声可行的原因：周期噪声在对应于周期干扰的频率处，以**集中的能量脉冲**形式出现。
- **陷波滤波器**
  - 阻止事先定义的中心频率邻域内的频率
  - 可以通过将与噪声对应的**频率分量设置为零**来减少周期性噪声。

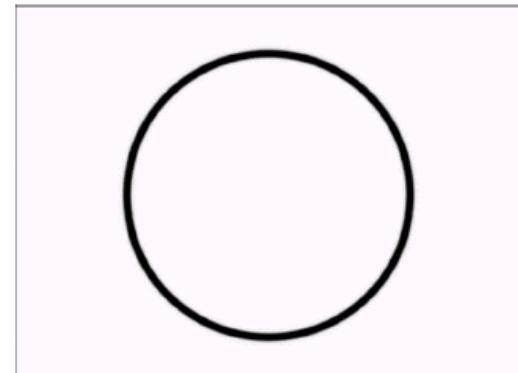
退化图像



傅里叶变换



陷波滤波器



复原图像



# 图像复原

- **图像复原定义：**利用“图像”退化过程的先验知识，建立一个退化模型，以此模型为基础，采用各种逆退化处理方法进行恢复，使图像质量得到改善，恢复已被退化图像的本来面目。
- 两种方法：
  - 在图像退化可知的情况下，**图像退化的逆操作**：建立模型，使用某种概率准则对参数进行估计，检测问题。
  - **退化过程不可知或无法精确获得**：对退化过程（模糊与噪声）建立模型，寻找一种去除或削弱其影响的过程，估计问题。
- 可见，图像复原主要取决于对图像退化过程的先验知识所掌握的精确程度。过程如下：



# 图像退化/复原的数学模型

## • 图像退化模型

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

or

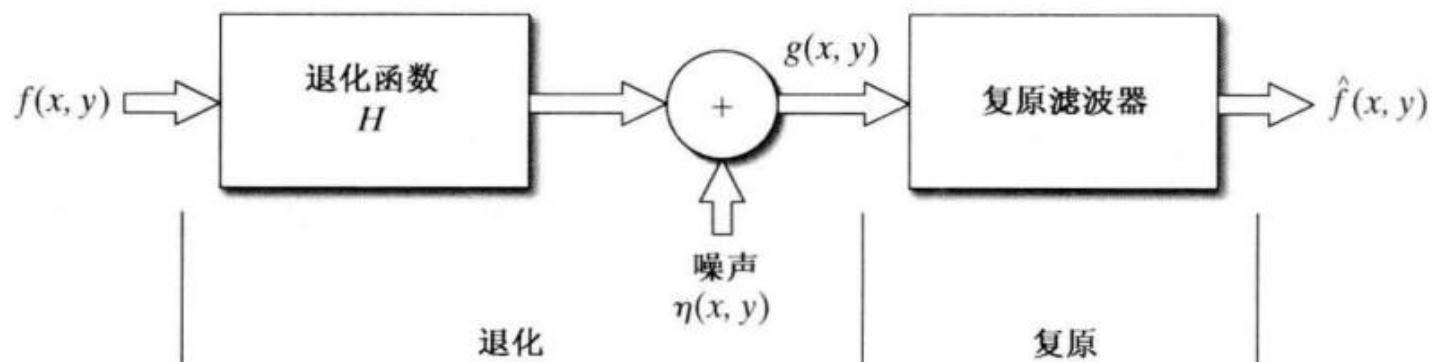
$$G(u, v) = F(u, v)H(u, v) + N(u, v)$$

↑卷积操作

$g(x, y)$  —— 退化图像;  $f(x, y)$  —— 理想图像 (输入图像)

$h(x, y)$  —— 点扩散函数;  $\eta(x, y)$  —— 加性噪声

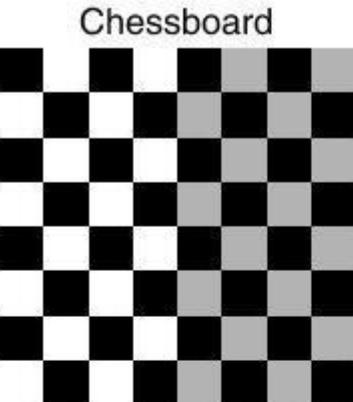
- 目标: **估计**  $h(x, y)$  或者  $H(u, v)$
- 为什么? 如果知道了  $h(x, y)$  或者  $H(u, v)$ , 在忽略噪声的前提下, 可以从  $g(x, y)$  最后得到去除退化的估计  $\hat{f}(x, y)$ 。



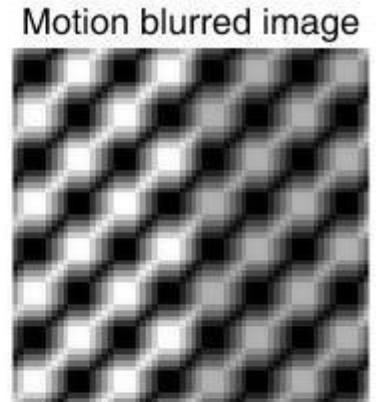
# MATLAB应用实例

退化函数建模

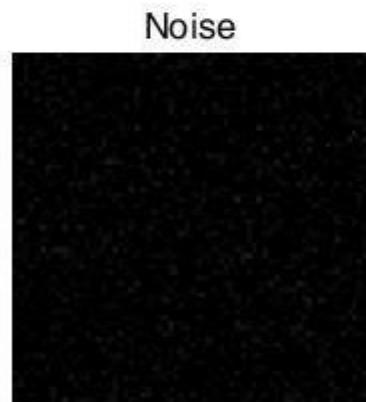
```
>> f = checkerboard(8);  
>> PSF = fspecial('motion', 7, 45);  
>> gb = imfilter(f, PSF, 'circular');  
>> noise = imnoise(zeros(size(f)), 'gaussian', 0, 0.001);  
>> g = gb + noise;  
  
>> subplot(2, 2, 1),  
imshow(pixeldup(f,8),[ ]);title('Chessboard');  
>> subplot(2, 2, 2), imshow(gb);title('Motion blurred image');  
>> subplot(2, 2, 3), imshow(noise);title('Noise');  
>> subplot(2, 2, 4), imshow(g);title('Noise And Motion');
```



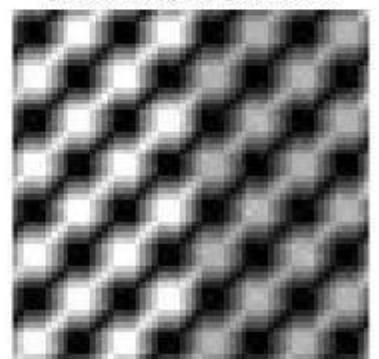
Chessboard



Motion blurred image

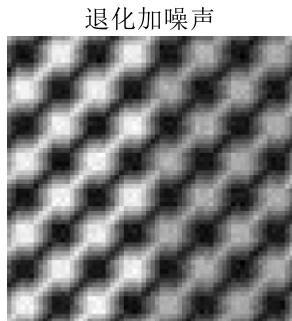
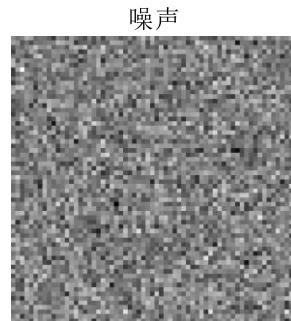
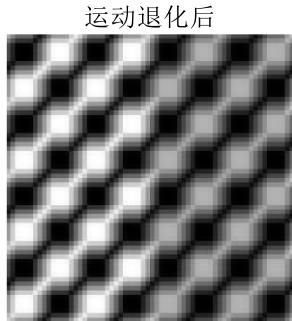
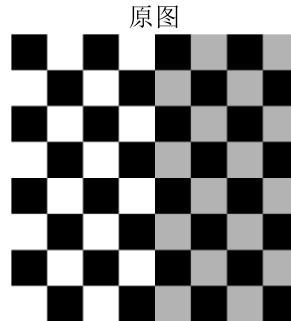


Noise



Noise And Motion

# 退化函数建模



## • 退化函数的建模：

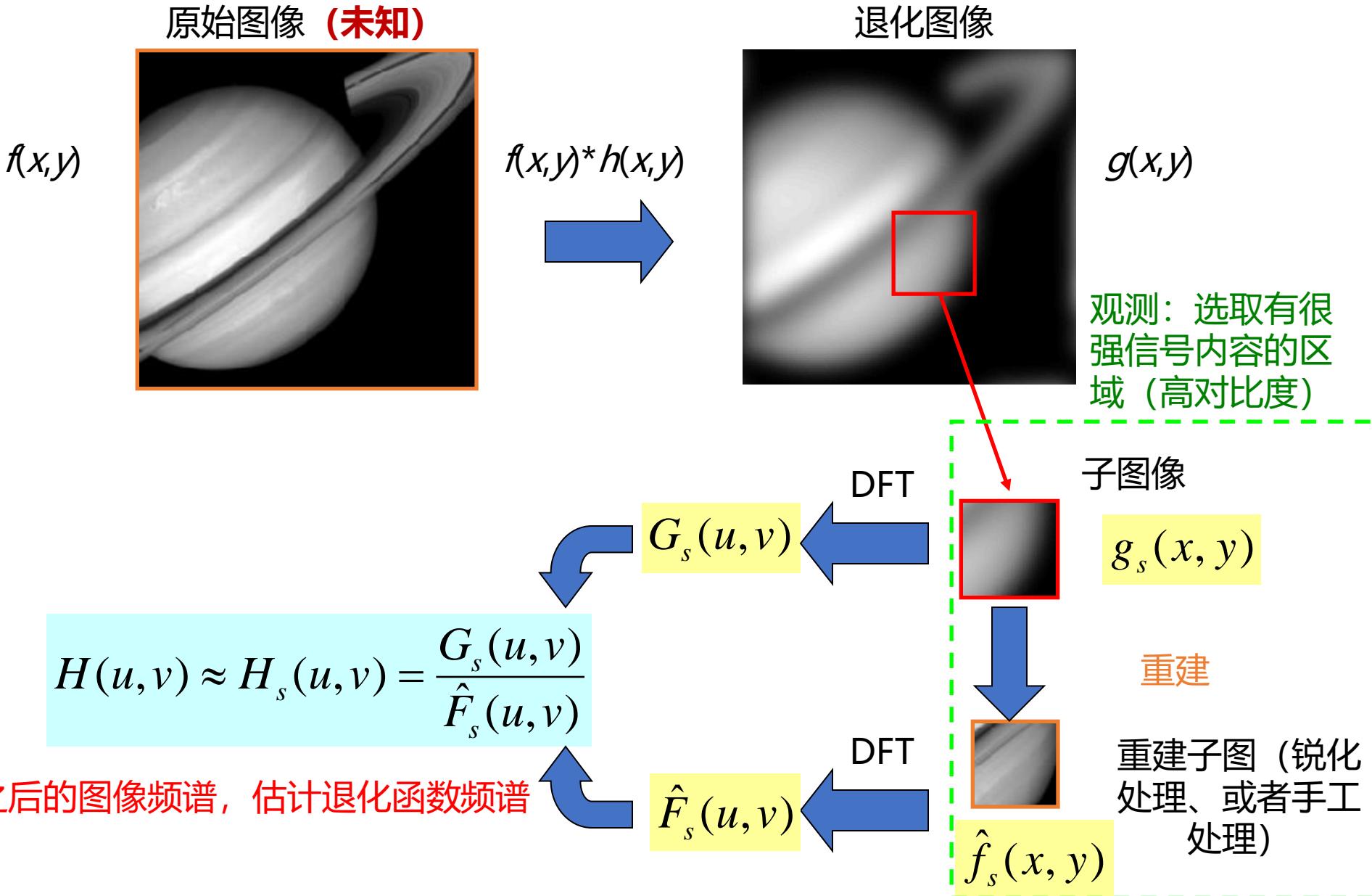
- 主要退化场景：传感器低通特性/相对运动导致的图像模糊。
- 一般会进行线性位置不变的假设，退化过程也可以用卷积来描述。
- 需要对退化函数进行建模或估计。

# 估计退化函数 H

---

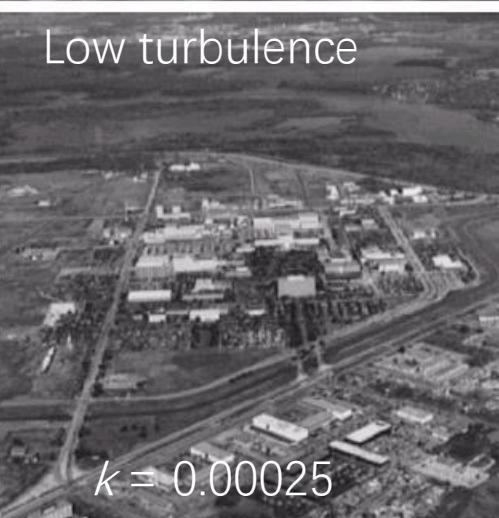
- **Estimating the Degradation Function**
  - 通过观测图像估计(Estimation by Image Observation)
  - 通过实验估计(Estimation by Experimentation)
  - 通过数学建模估计(Estimation by Modeling)

# 通过图像本身估计退化函数H



# 通过数学建模估计

在已知物理退化过程机理和数学表达式的前提下，对退化过程背后的物理特性进行建模。



案例:

Atmospheric  
Turbulence model  
大气湍流模型

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

# 通过数学建模估计：运动模糊

Assume that camera velocity is

$$(x_0(t), y_0(t))$$

The blurred image is obtained by

$$g(x, y) = \int_0^T f(x + x_0(t), y + y_0(t)) dt$$

where  $T$  = exposure time.

$$\begin{aligned} G(u, v) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-j2\pi(ux+vy)} dx dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[ \int_0^T f(x + x_0(t), y + y_0(t)) dt \right] e^{-j2\pi(ux+vy)} dx dy \\ &= \int_0^T \left[ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x + x_0(t), y + y_0(t)) e^{-j2\pi(ux+vy)} dx dy \right] dt \end{aligned}$$

# 通过数学建模估计：运动模糊

$$\begin{aligned}
 G(u, v) &= \int_0^T \left[ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x + x_0(t), y + y_0(t)) e^{-j2\pi(ux+vy)} dx dy \right] dt \\
 &= \int_0^T [F(u, v) e^{-j2\pi(ux_0(t)+vy_0(t))}] dt \\
 &= F(u, v) \int_0^T e^{-j2\pi(ux_0(t)+vy_0(t))} dt
 \end{aligned}$$

Then we get, the motion blurring transfer function:

$$H(u, v) = \int_0^T e^{-j2\pi(ux_0(t)+vy_0(t))} dt$$

For constant motion

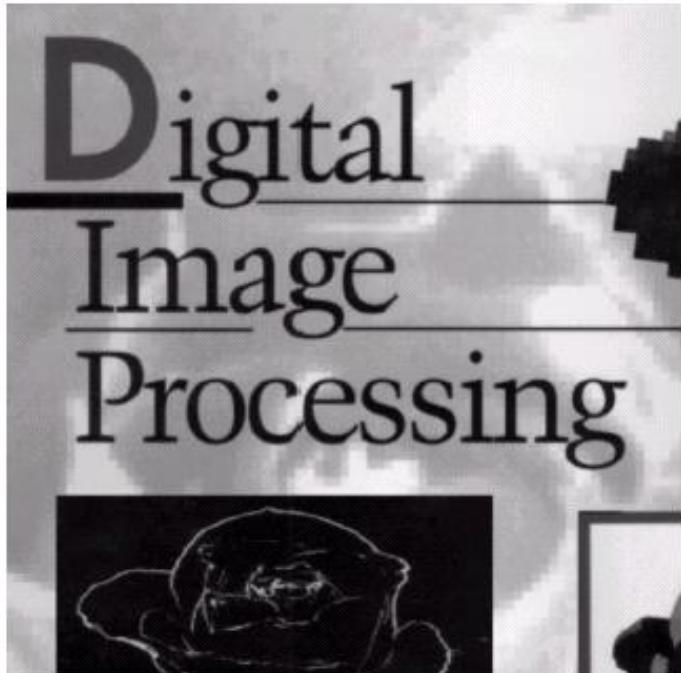
$$(x_0(t), y_0(t)) = (at, bt)$$

$$H(u, v) = \int_0^T e^{-j2\pi(ua+vb)} dt = \frac{T}{\pi(ua + vb)} \sin(\pi(ua + vb)) e^{-j\pi(ua + vb)}$$

# 通过数学建模估计：运动模糊

For constant motion

$$H(u, v) = \frac{T}{\pi(ua + vb)} \sin(\pi(ua + vb)) e^{-j\pi(ua + vb)}$$



Original image



Motion blurred image  
 $a = b = 0.1, T = 1$

# 逆滤波退化复原

## ➤ 最简单且粗糙的恢复方法

退化函数模型的频率域版本  $G(u, v) = H(u, v)F(u, v) + N(u, v)$

**直接逆滤波：**在无噪声的理想情况下，估计出退化函数，在频率域滤波，上式可简化为：

$$F(u, v) = G(u, v) / H(u, v)$$

式中， $1/H(u, v)$ 称为逆滤波器。对上式再进行傅立叶反变换可得到 $f(x, y)$ 。

## ➤ 复原步骤：

- ① 对退化图像 $g(x, y)$ 作二维离散傅立叶变换，得到 $G(u, v)$
- ② 计算退化函数 $h(x, y)$ 的二维傅立叶变换，得到 $H(u, v)$
- ③ 按  $F(u, v) = G(u, v) / H(u, v)$ ，计算  $\hat{F}(u, v)$
- ④ 计算  $\hat{F}(u, v)$  的逆傅立叶变换，求得  $\hat{f}(x, y)$

# 逆滤波退化复原

实际上，很多实际问题都是有噪声，且由于 $N(u, v)$ 未知，即使得到退化函数 $H(u, v)$ ，也难以精确重建，因而只能求 $F(u, v)$ 的估计值  $\hat{F}(u, v)$ ：

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

$H(u, v)$ 很小或为零时，则噪声被放大，会对逆滤波恢复的图像产生很大影响，使恢复图像和 $f(x, y)$ 相差很大，甚至面目全非

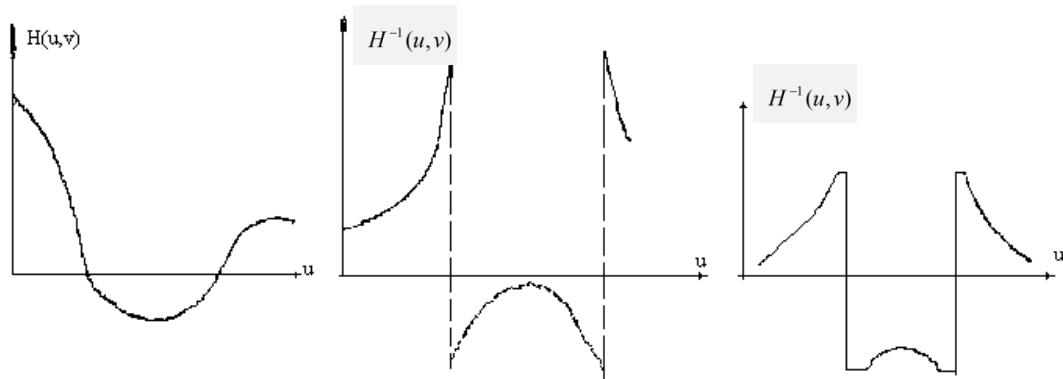
由于 $H(u, v)$ 处于分母的位置上，可能会发生下列情况，即在 $u, v$ 平面上有些点或区域会产生 $H(u, v) = 0$ 或 $H(u, v)$ 非常小的情况，在这种情况下，即使没有噪声也无法精确恢复 $f(x, y)$ 。

另外，在有噪声存在时，在 $H(u, v)$ 的邻域内， $H(u, v)$ 的值可能比 $N(u, v)$ 的值小得多，因此，上面的式子得到的噪声项可能会非常大，使 $f(x, y)$ 不能正确恢复。

# 逆滤波退化复原

相应回答：

(a) 在  $H(u, v) = 0$  及其附近，人为地仔细设置  $H^{-1}(u, v)$  的值，使  $N(u, v) * H^{-1}(u, v)$  不会对  $\hat{F}(u, v)$  产生太大影响。



图像退化响应      逆滤波器响应      改进的逆滤波器响应

(b) 使  $H^{-1}(u, v)$  具有低通滤波性质，如下式所示：

$$H^{-1}(u, v) = \begin{cases} \frac{1}{H(u, v)} & D \leq D_0 \\ 0 & D > D_0 \end{cases}$$

- a)用全滤波的复原结果
- b)半径为40时的结果
- c)半径为70时的结果
- d)半径为85时的结果

半径为**70**时效果最好，  
大于**70**时产生退化，  
小于**70**时变得模糊。

**这个例子的结果说明**  
**直接进行逆滤波的结**  
**果是较差的。**



剧烈湍流图像的复原结果

# 维纳滤波

---

- 逆滤波只能解决只有退化函数，没有加噪声的问题。
- 维纳滤波**又称最小均方误差滤波，综合考虑了退化函数和噪声，是一种建立在**最小化统计准则**的基础上的复原方法。
- 按照使恢复的图像与原图像 $f(x,y)$  的均方差最小原则来恢复图像。即

$$E[(\hat{f}(x,y) - f(x,y))^2] = \min$$

- 目标：找到未污染图像的一个估计，使它们之间的均方差最小，可以去除噪声，同时清晰化模糊图像。

解： $\hat{F}(u,v) = \left[ \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)} \right] G(u,v)$

其中， $S_\eta(u,v) = |N(u,v)|^2$ 是噪声功率谱， $S_f(u,v) = |F(u,v)|^2$ 为未退化图像功率谱。

# 维纳滤波复原步骤

简化形式：

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

估算困难，替换成  $K$ ， $K$  的值手动选择

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

- (a) 计算图像  $g(x, y)$  的二维离散傅立叶变换得到  $G(u, v)$ ；
- (b) 计算退化函数  $h(x, y)$  的二维离散傅立叶变换  $H(u, v)$ ；
- (c) 估算图像的功率谱密度  $P_f$  和噪声的谱密度  $P_n$ ； -> 手动设置  $K$
- (d) 由公式计算图像的估计值  $\hat{F}(u, v)$ ；
- (e) 计算  $\hat{F}(u, v)$  的逆傅立叶变换，得到恢复后的图像  $\hat{f}(x, y)$

# MATLAB应用实例

```
color_pic=imread('lena512color.bmp'); %读取彩色图像
gray_pic=rgb2gray(color_pic); %将彩色图转换成灰度图
double_gray_pic=im2double(gray_pic); %将uint8转成im2double型
[width,height]=size(double_gray_pic);
H=fspecial('average',9); %9x9均值滤波器
degrade_img1=imfilter(double_gray_pic, H, 'conv', 'circular'); %卷积滤波
```

%-----在均值滤波模糊图像基础上添加椒盐噪声-----

```
degrade_img2=imnoise(degrade_img1,'salt & pepper',0.05);
%给退化图像1添加噪声密度为0.05的椒盐噪声(退化图像2)
```

```
figure('name','退化图像');
subplot(2,2,1);imshow(color_pic,[]);title('原彩色图');
subplot(2,2,2);imshow(double_gray_pic,[]);title('原灰度图');
subplot(2,2,3);imshow(degrade_img1,[]);title('退化图像1');
subplot(2,2,4);imshow(degrade_img2,[]);title('退化图像2');
```



原彩色图



原灰度图



退化图像1



退化图像2

# MATLAB应用实例

%-----对退化图像1逆滤波复原-----

```
fourier_H=fft2(H,width,height);
fourier_degrade_img1=fft2(degrade_img1);
%相当于 G(u,v)=H(u,v)F(u,v), 已知G(u,v),H(u,v), 求F(u,v)
restore_one=ifft2(fourier_degrade_img1./fourier_H);
%矩阵相除用./

figure('name','退化图像1逆滤波复原');
subplot(1,2,1);imshow(im2uint8(degrade_img1),[]);title('退化图像1');
subplot(1,2,2);imshow(im2uint8(restore_one),[]);title('复原图像1');
```



# MATLAB应用实例

%-----对退化图像2直接逆滤波复原-----

```

fourier_degrade_img2=fft2(degrade_img2);

%相当于 G(u,v)=H(u,v)F(u,v)+N(u,v)
restore_two=ifft2(fourier_degrade_img2./fourier_H);

%-----去掉噪声分量逆滤波复原-----
noise=degrade_img2-degrade_img1;    %提取噪声分量

fourier_noise=fft2(noise);    %对噪声进行傅里叶变换

restore_three=ifft2((fourier_degrade_img2-fourier_noise)./fourier_H);

%G(u,v)=H(u,v)F(u,v)+N(u,v),解得F(u,v)=[G(u,v)-N(u,v)]/H(u,v)

figure('name','退化图像2逆滤波复原');

subplot(2,2,1);imshow(double_gray_pic,[]);title('原灰度图');

subplot(2,2,2);imshow(im2uint8(degrade_img2),[]);title('退化图像2');

subplot(2,2,3);imshow(im2uint8(restore_two),[]);title('直接逆滤波复原');

subplot(2,2,4);imshow(im2uint8(restore_three),[]);title('去掉噪声分量逆滤
波复原');

```

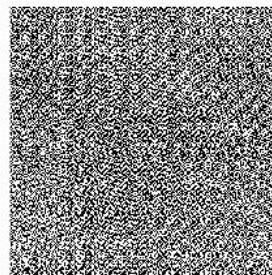
原灰度图



退化图像2



直接逆滤波复原

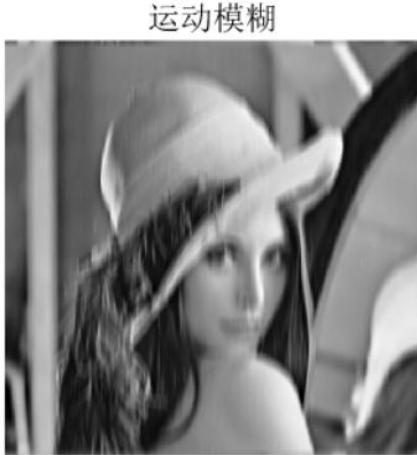


去掉噪声分量逆滤波复原



# MATLAB应用实例

```
%-----添加运动模糊-----  
H_motion = fspecial('motion', 18, 90);  
%运动长度为18, 逆时针运动角度为90°  
  
motion.blur = imfilter(double_gray_pic, H_motion, 'conv',  
'circular'); %卷积滤波  
  
noise_mean=0; %添加均值为0  
noise_var=0.001; %方差为0.001的高斯噪声  
  
motion.blur_noise=imnoise(motion.blur, 'gaussian', noise_mean, noise_var);  
%添加均值为0, 方差为0.001的高斯噪声
```



```
figure('name','运动模糊加噪');  
subplot(1,2,1);imshow(motion.blur,[ ]);title('运动模糊');  
subplot(1,2,2);imshow(motion.blur_noise,[ ]);title('运动模糊添加噪声');
```

# MATLAB应用实例

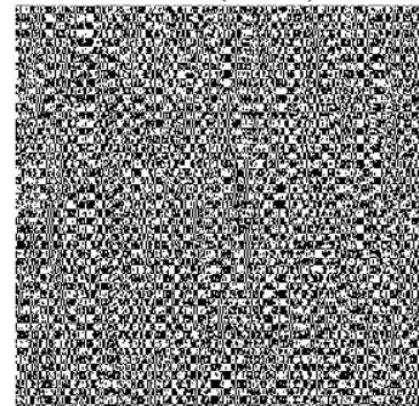
```
%-----维纳滤波matlab自带函数deconvwnr-----
restore_ignore_noise = deconvwnr(motion.blur_noise, H_motion,
0);
%nsr=0,忽视噪声

signal_var=var(double_gray_pic(:));
estimate_nsr=noise_var/signal_var;    %噪信比估值

restore_with_noise=deconvwnr(motion.blur_noise,H_motion,estima
te_nsr);
%信号的功率谱使用图像的方差近似估计

figure('name','函数法维纳滤波');
subplot(1,2,1);imshow(im2uint8	restore_ignore_noise),[];title(
'忽视噪声直接维纳滤波(nsr=0), 相当于逆滤波');
subplot(1,2,2);imshow(im2uint8	restore_with_noise),[];title('
考虑噪声维纳滤波');
```

忽视噪声直接维纳滤波(nsr=0), 相当于逆滤波

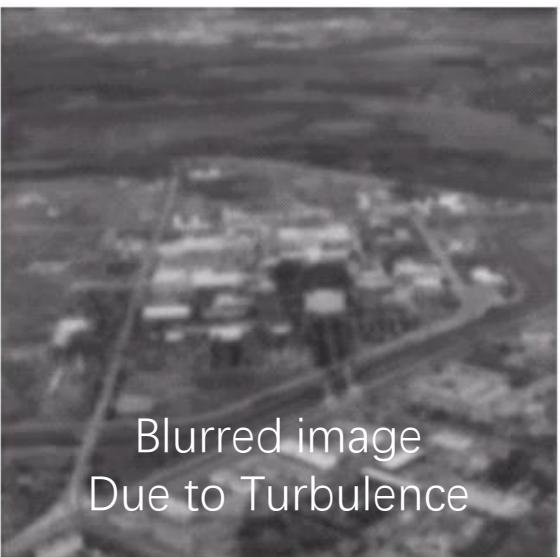
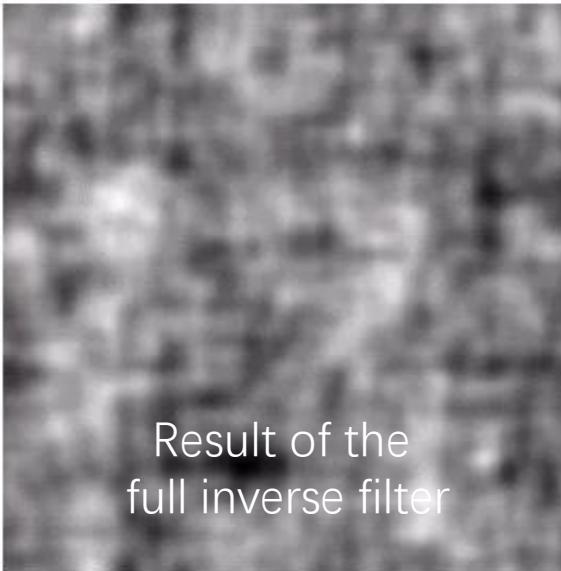


考虑噪声维纳滤波



# 逆滤波 VS 维纳滤波

维纳滤波效果最好，接近于原图像



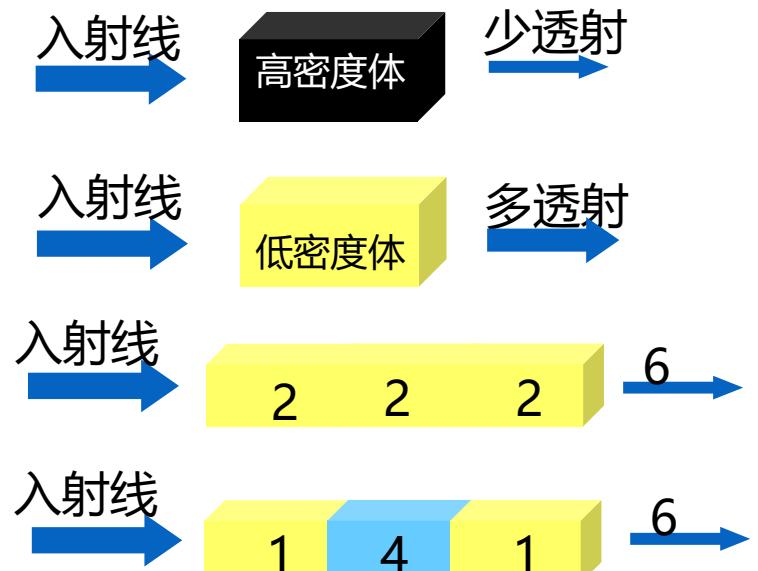
# 图像投影重建

---

- **投影重建 Image Reconstruction:** 由一个物体的多个（轴向）投影图重建目标图像的技术。
- 投影重建图像是一类特殊的图像恢复技术，把投影看出是一种**退化过程**，而重建则是一种**复原过程**。
- 通过投影重建可以直接看到原来被投影物体**某种特性的空间分布**，比直接观察投影图直观，因此在医疗放射学、核医学、光显微镜和全息成像学及理论视觉等领域都有广泛应用。

# 射线投影重建

- 射线穿过物体，在检测器上得到的**遭受衰减的值=射线的投影**
- 根据投影可以了解物体对射线的**吸收程度**
- 每块上的数字表示每块的密度或衰减，**总的衰减是叠加的**
  - 一条射线束通过均匀密度物质的厚块
  - 另一射线通过不等密度的厚块组合，但检测器的记录相同

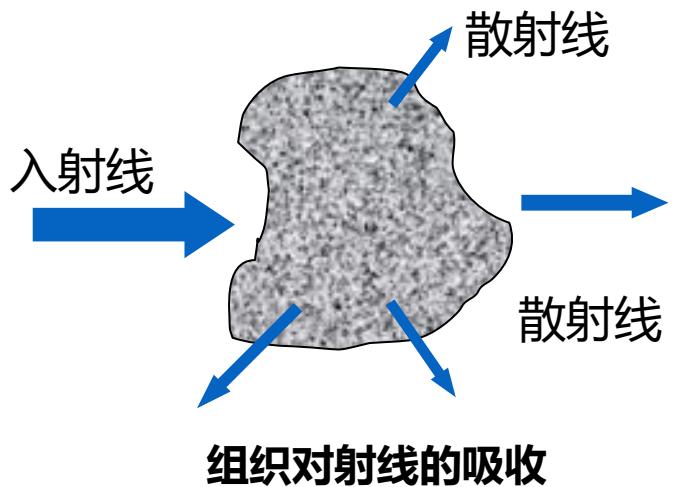


等强度射线穿透不同组织的情况

# 射线投影重建

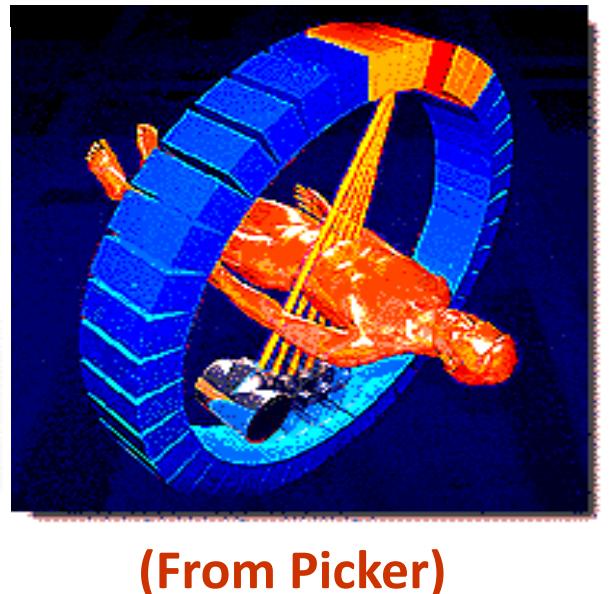
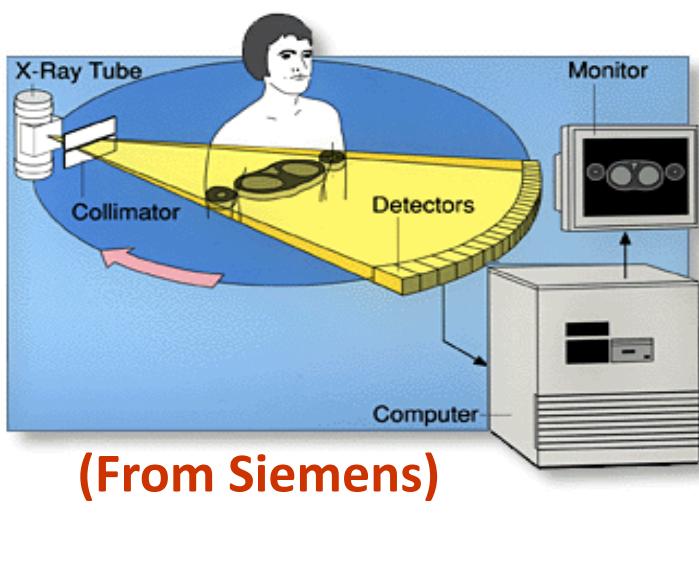
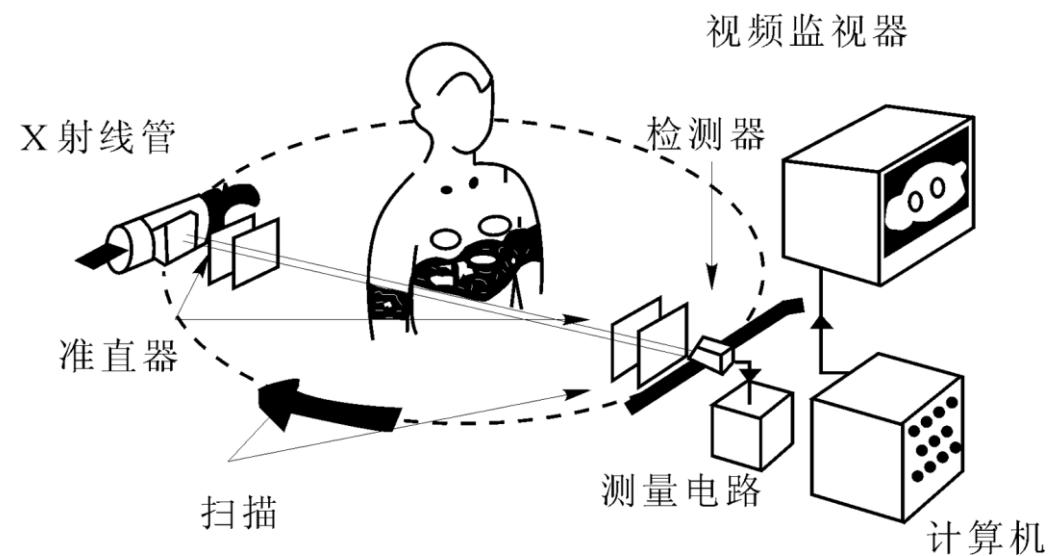
## ➤ 射线投影成像的基本原理：

- 人体组织对X射线吸收和散射，造成衰减
- 人体内的不同结构，比如脂肪、胰、骨骼对X射线吸收能力有所不同



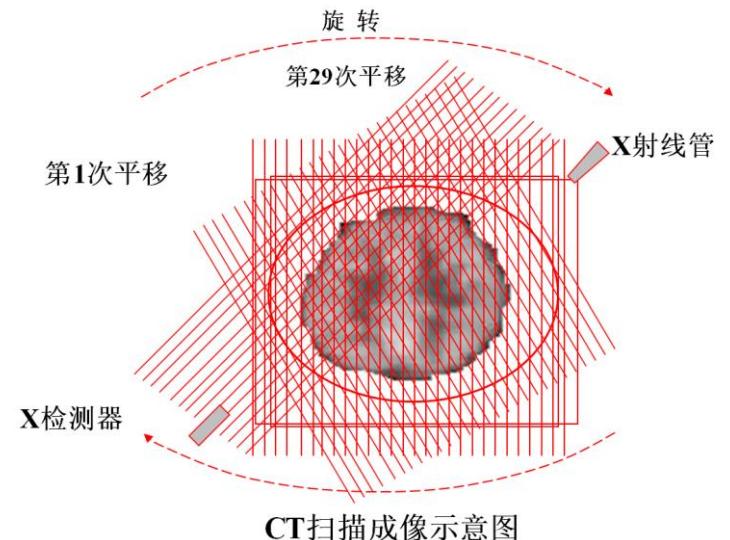
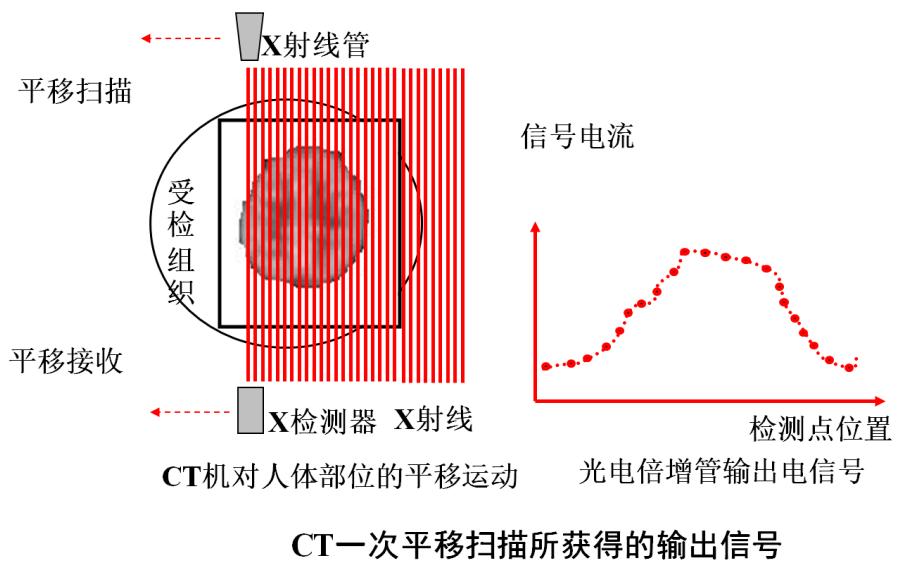
# CT扫描的二维重建

- **计算机断层扫描技术 (Computerized Tomography, CT)** : 用于人体头部、腹部等内部器官的无损伤诊断，其基本方法就是根据人体截面投影，经过计算机处理来重建截面图象。
- 计算机断层扫描的基本原理：从线性并排着的X射线源发射一定强度的**X射线**，把通过组织的X射线用与X射线源平行排列的**X射线检测器**接收。然后把X射线源和检测器组以体轴为中心一点一点地**旋转**，反复进行同样地操作。利用这样求得的在**各个角度上的投影数据**，就可得到了垂直于体轴的**断面图像**。



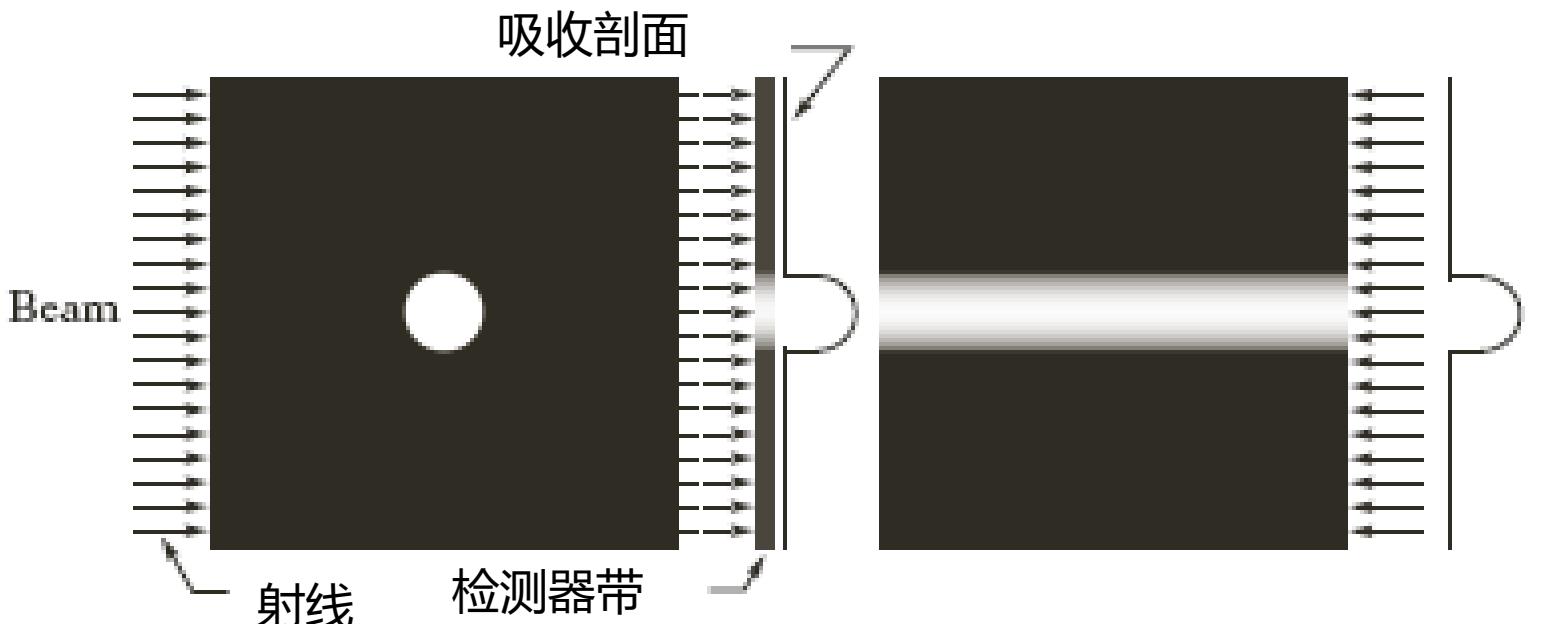
# X射线计算机断层成像 (CT)

- 扫描系统的X射线源和检测器，始终保持严格的相对静止；
- 射线管发出的是**直线形波束**，扫描机围绕人体作**旋转加平移**运动；
- 以检测器的位置为自变量，就构成电流—位置函数曲线；
- 第一次直线平移扫描完毕后，扫描系统旋转一个小角度，再作第二次直线式平移扫描，获得另一组投影数据；
- 重复以上过程，便得到很多组投影数据；
- 对这些数据进行处理形成**三维图像**。



# 反投影法

- 假设一个人体截面切片显示均匀组织（黑色背景）和肿瘤（白色明亮区域）
- 用一束细的X射线垂直穿过该切片，利用放在另一端的X射线吸收检测器来记录测量值。肿瘤吸收的射线束能量比背景吸收的射线束能量多。
- 右侧信号（吸收剖面）最大的吸收出现在该区域中心，形成函数图像。**
- 基于上述函数的信息来重建一幅图像：**沿着射线来的方向把一维信号反投影回去，形成由一个吸收剖面波形生成的二维数字图像。**



# 反投影法

- 反复投影过程：利用上面描述的反投影法，继续进行X射线带和检测带的旋转，得到 $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ ,  $135^\circ$ 的四个角度。
- 当反投影的数量增加时，相对于原始区域的平坦区，有较大吸收区域（肿瘤）的强大变大。
- 以此为理论依据，把旋转角度变小，设置为间隔 $5.625^\circ$ ：**基本可以还原出物体轮廓。**
- 间隔不能设置得太小，间隔越小导致图像越模糊。

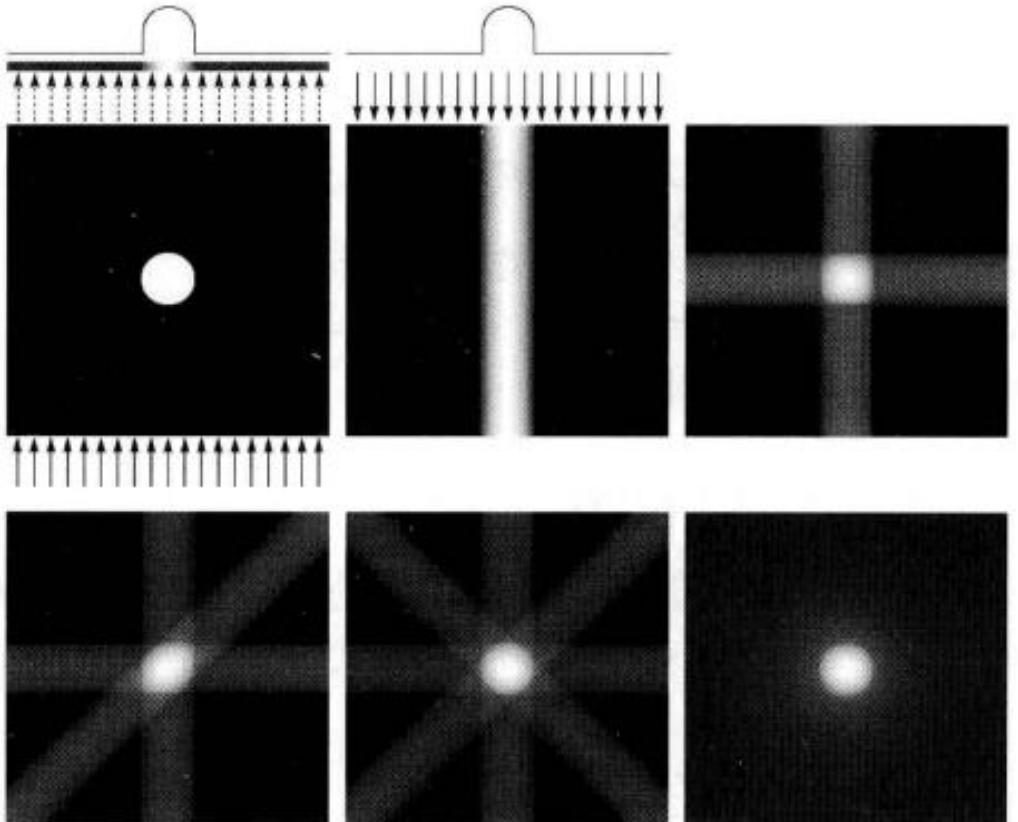


图 4.9 (a) 显示一个简单物体、一束平行射线、一个检测器条带和吸收剖面的平坦区域；(b) 吸收剖面的反投影；(c) 旋转  $90^\circ$  的射线束和检测器条带；(d) 吸收剖面的反投影；(e) 图(b) 和图(d) 的和；(f) 加上另一个反投影( $45^\circ$ )后的结果；(g) 再加上另一个反投影( $135^\circ$ )后的结果；(h) 加上 32 个相隔  $5.625^\circ$  的反投影的结果

# 投影定理

---

- 投影切片定理给出了图像在**空间域**上对X轴的投影与在**频率域**u轴的切片之间的关系。
- **一个二维图像，它的一维投影的傅立叶变换与二维物体的傅立叶变换的主体部分相等**

设 $G(r, \theta)$ 是 $g(s, \theta)$ 对变量s的一维傅立叶变换，

$$G(r, \theta) = \int_{(s, \theta)} g(s, \theta) \exp[-j2\pi rs] ds$$

设 $F(u, v)$ 是 $f(x, y)$ 的二维傅立叶变换

$$F(u, v) = \iint_Q f(x, y) \exp[-j2\pi(ux + vy)] dx dy$$

# 投影定理证明

---

► 一个 $N$ 维函数  $f(x_1, x_2, x_3, \dots, x_N)$  在第 $N-1$ 维上的映射称为函数  $f$  在第 $N-1$ 维的投影。

● 设  $f(x,y)$  为二维图像，在 $x$ 轴上（沿 $y$ 方向）的投影

$$g_y(x) = \int_l f(x, y) dy = \int_{-\infty}^{\infty} f(x, y) dy \quad (1)$$

在 $y$ 轴上（沿 $x$ 方向）的投影

$$g_x(x) = \int_l f(x, y) dx = \int_{-\infty}^{\infty} f(x, y) dx \quad (2)$$

● 设  $f(x,y)$  的傅立叶变换为  $F(u, v)$ ，可得：

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp[j2\pi(ux + vy)] du dv \quad (3)$$

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp[-j2\pi(ux + vy)] dx dy$$

# 投影定理证明

- 把式 (3) 代入到式 (1) 可得:

$$\begin{aligned}
 g_y(x) &= \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp[j2\pi(ux + vy)] dudv \right] dy \\
 &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp(j2\pi ux) dudv \int_{-\infty}^{\infty} \exp(j2\pi vy) dy \\
 &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp(j2\pi ux) \delta(v) dudv \\
 &= \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} F(u, v) \delta(v) dv \right] \exp(j2\pi ux) du \\
 &\quad \int_{-\infty}^{\infty} F(u, 0) \exp(j2\pi ux) du
 \end{aligned} \tag{4}$$

- $g_y(x)$ 是 $F(u, 0)$  的傅立叶反变换， 或 $g_y(x)$  的傅立叶变换 $G(u)$ 与 $F(u, 0)$ 相同。

函数 $f(x, y)$ 在 $x$ 轴上投影 $g_y(x)$ 的傅立叶变换

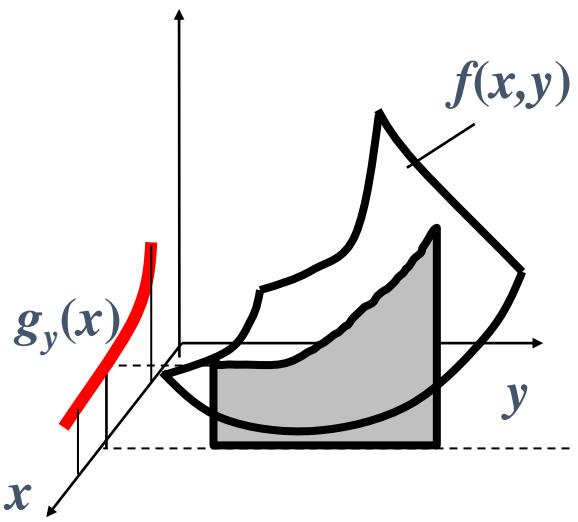
等于  $G_Y(u) = F(u, 0)$

$f(x, y)$ 的傅立叶变换 $F(u, v)$ 在 $(u, v)$ 平面上沿 $u$ 轴平面上的切片 $F(u, 0)$

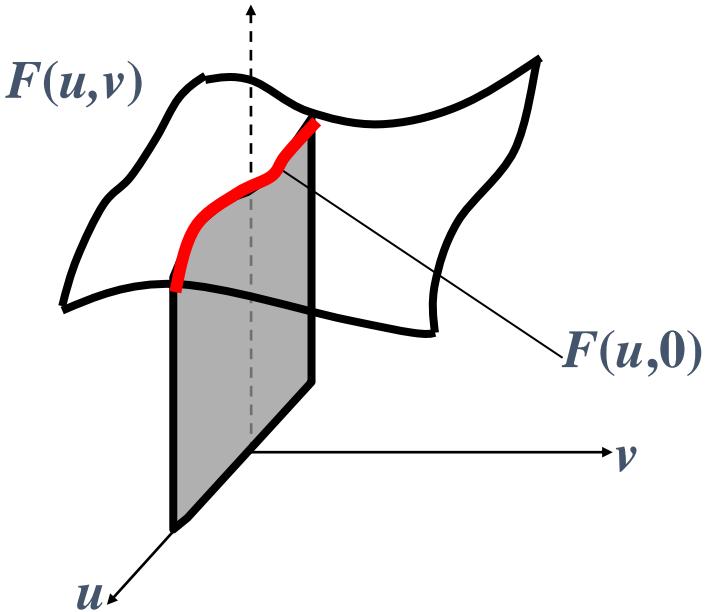
同理:  $G_X(v) = F(0, v)$

函数  $f(x,y)$  在  $x$  轴上投影  $g_y(x)$  的傅立叶变换  
等于

$f(x,y)$  的傅立叶变换  $F(u,v)$  在  $(u,v)$  平面上沿  $v$  轴平面上的切片  $F(u,0)$



(a) 二维函数  $f(x,y)$  在  $x$  轴上投影



(b)  $f(x,y)$  傅立叶变换  $F(u,v)$  在  $u$  轴上切片

# 投影定理

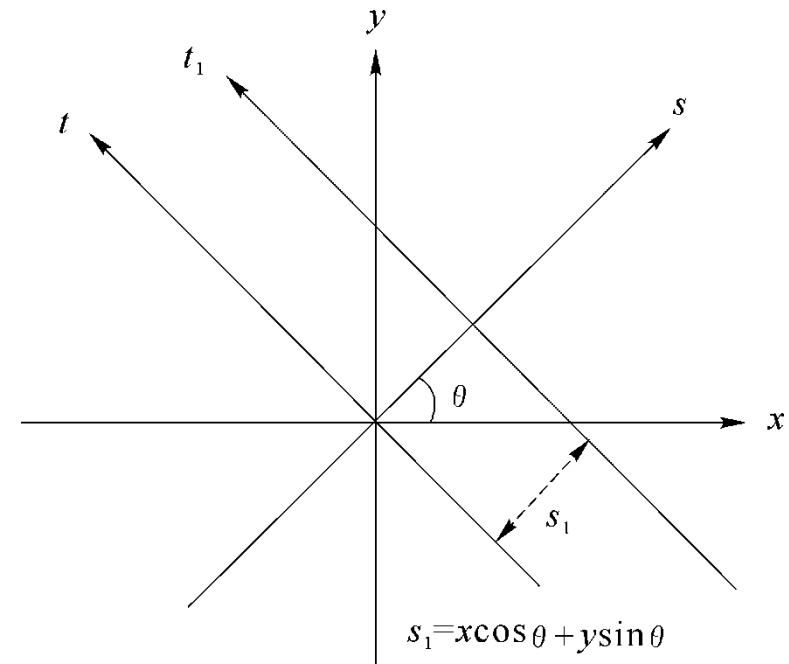
不失一般性，如果投影并非是对 $X$ 轴或 $Y$ 轴进行，而是对与空间域的 $X$ 轴成任意的角度 $\theta$ 的方向，即 $S$ 轴，进行投影。

- 假设函数 $f(x, y)$ 投影到一条经过旋转的直线上 $t_1$ ， $t$ 是一条与 $t_1$ 平行经过原点的直线，直线 $t_1$ 离开原点的距离为 $s_1$ 。
- 与 $t$ 垂直经过原点的直线为 $s$ ，该直线 $s$ 与 $x$ 轴的夹角为 $\theta$ ，以 $s$ 和 $t$ 可用 $\theta$ 为极坐标：

$$\begin{cases} s = x \cos \theta + y \sin \theta \\ t = -x \sin \theta + y \cos \theta \end{cases}$$

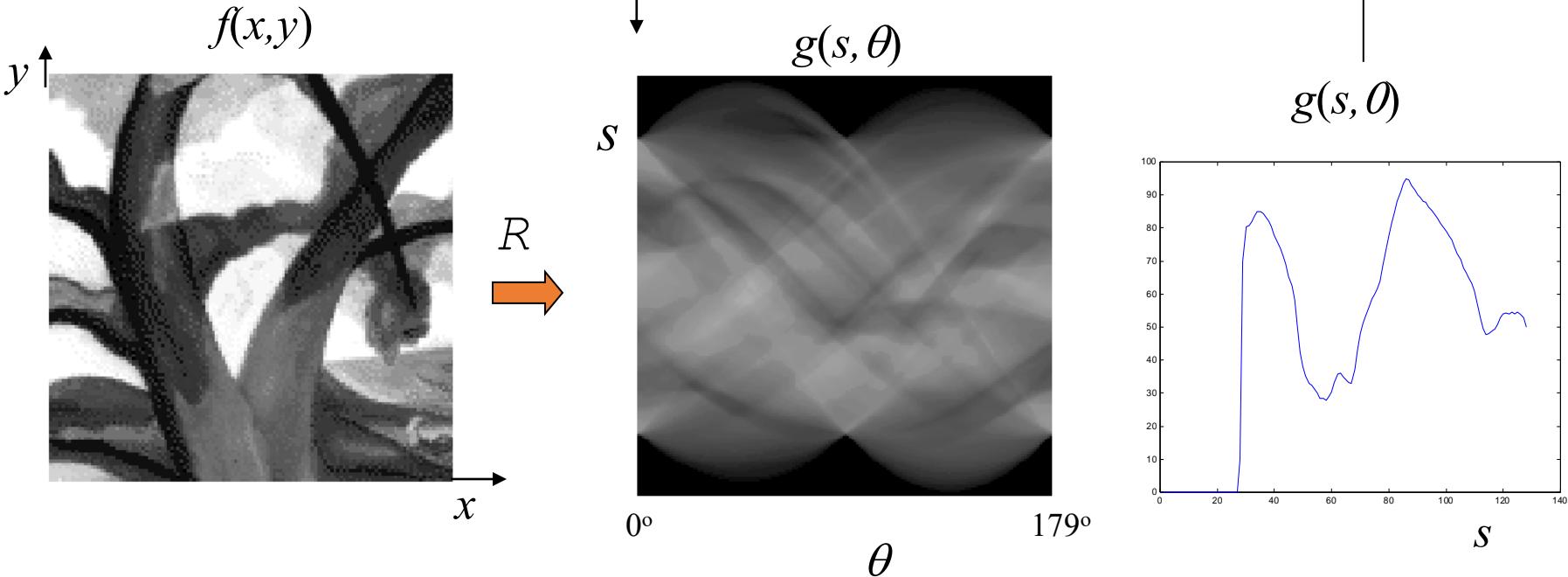
- 函数 $f(x, y)$ 沿着 $t_1$ 方向对 $s_1$ 投影为：

雷顿变换      
$$g_t(s_1, \theta) = \int_t f(x, y) dt \quad (7.6)$$



坐标旋转关系

# MATLAB 雷顿变换



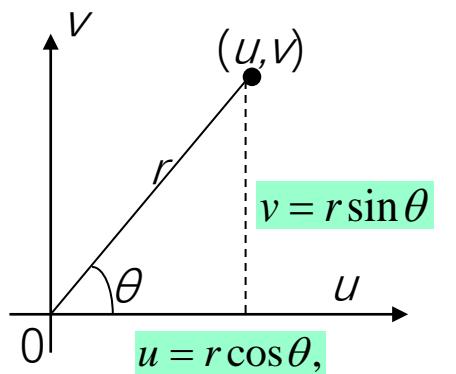
```
B=radon(A,0:179);
```

# 投影定理

- 将投影式(7.6)只对 $s_1$ 作一维傅立叶变换，将指数项作变换，得

$$\begin{aligned}
 G(r, \theta) &= \int_{-\infty}^{\infty} g(s_1, \theta) \exp(-j2\pi r s_1) ds_1 \\
 &= \int_{-\infty}^{+\infty} \left[ \int_{-\infty}^{+\infty} f(x, y) dt \right] \exp(-j2\pi r s_1) ds_1 \\
 &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \begin{vmatrix} \frac{\partial s_1}{\partial x} & \frac{\partial s_1}{\partial y} \\ \frac{\partial s_1}{\partial t} & \frac{\partial s_1}{\partial t} \end{vmatrix} f(x, y) \exp[-j2\pi r(x \cos \theta + y \sin \theta)] dx dy \\
 &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (\cos^2 \theta + \sin^2 \theta) f(x, y) \exp[-j2\pi(xr \cos \theta + yr \sin \theta)] dx dy \\
 &= \color{red} F(r \cos \theta, r \sin \theta) \\
 &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} 1 \cdot f(x, y) \exp[-j2\pi r(xu + yv)] dx dy \\
 &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) \exp[-j2\pi r(xu + yv)] dx dy = F(u, v)
 \end{aligned}$$

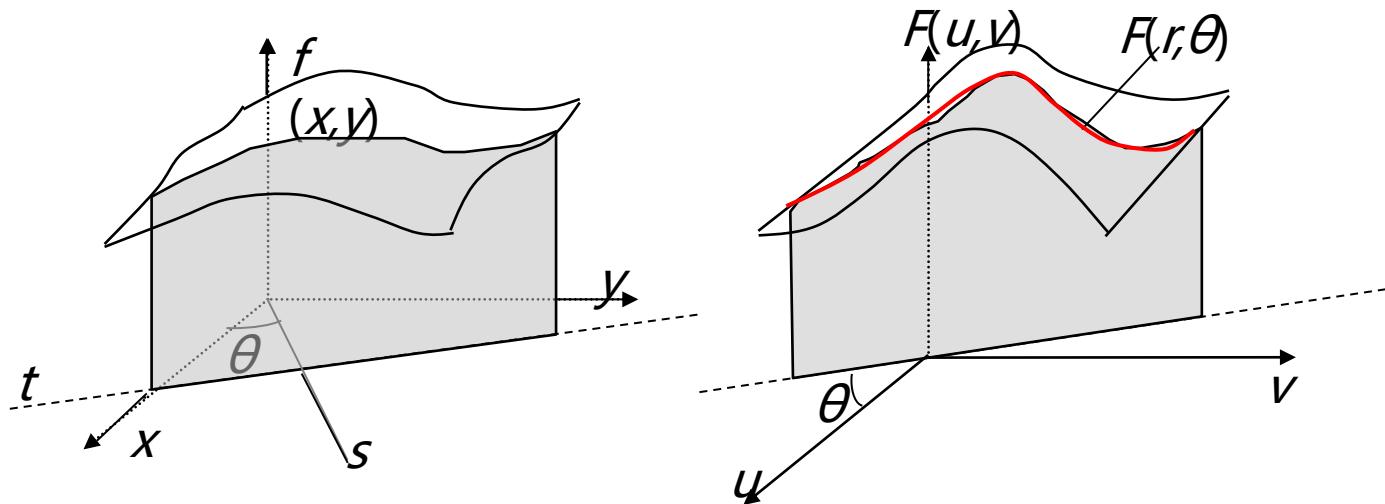
令 $u = r \cos \theta, v = r \sin \theta$



# 投影定理

- $f(x,y)$ 在一条与  $x$  轴夹角为  $\theta$ , 离开原点距离为  $s$  的直线上的投影  $g(s, \theta)$  的傅立叶变换  $G(r, \theta)$  等于  $f(x,y)$  的傅立叶变换  $F(u, v)$  在傅立叶空间里  $(r, \theta)$  处的值, 也等于其二维傅立叶变换  $F(u, v)$  在与  $u$  轴成  $\theta$  方向上的切片。

$$G(r, \theta) = F(r \cos \theta, r \sin \theta) = F(u, v)$$



投影定理示意图

# 傅里叶变换重建

- 若投影变换  $G(r, \theta)$  中的所有  $r$  及  $\theta$  值都是已知的，则图像的二维傅里叶变换也是确定的。为得到图像函数，须进行反变换运算，即：

$$f(x, y) = \int \int_{-\infty}^{\infty} F(u, v) \exp[j2\pi(ux + vy)] du dv.$$

极坐标表示为：

$$f(x, y) = \int_0^{2\pi} \int_0^{+\infty} G(r, \theta) \exp[j2\pi r(x \cos \theta + y \sin \theta)] dr d\theta$$

采用极坐标表示投影函数时，其傅立叶变换仅是对坐标轴的一维变换， $\theta$ 取决于投影角度的变换。要准确的重建图像，必须有足够的 $\theta$ 值进行投影。

# 傅里叶变换重建

通过将投影进行旋转和部分傅里叶变换可以首先构造整个的傅里叶变换的平面，然后只须再通过傅里叶反变换就可以得到重建后的物体。

## ➤ 步骤：

- 根据投影定理，可以得到  $F(u,v)$  分别在相应角度位置上的切片；
  - 当切片趋向无穷多，就可获得在  $(u,v)$  平面上的所有  $F(u,v)$  值；
  - 由  $F(u,v)$  进行傅立叶反变换就可以重建图像  $f(x,y)$ 。
- 结论：如果知道所有  $r$  和  $\theta$  的投影变换值  $G(r,\theta)$ ，则变换域的二维函数将全部确定，取傅立叶反变换就可以得到图像函数。

$$f(x,y) = \int_0^{2\pi} \int_0^{\infty} G(r,\theta) \exp[j2\pi r(x \cos \theta + y \sin \theta)] r dr d\theta \quad (7.15)$$

# 本章小结

---

- 图像复原和重建概念
- 噪声模型：空间随机噪声、周期噪声、估计噪声参数
- 图像复原：仅有噪声的图像复原
- 图像退化：退化模型、退化函数建模
- 直接逆滤波、维纳滤波
- 图像重建：由投影重建图像