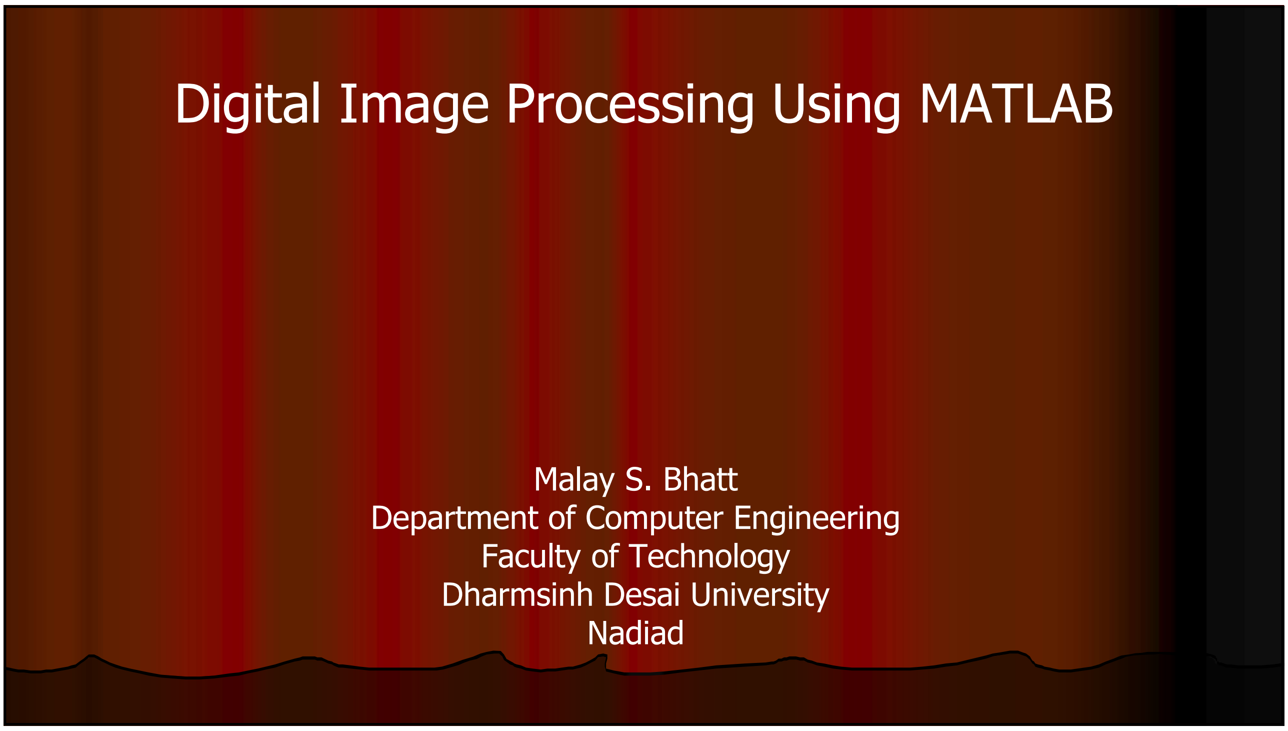


Digital Image Processing Using MATLAB

Malay S. Bhatt
Department of Computer Engineering
Faculty of Technology
Dharmsinh Desai University
Nadiad

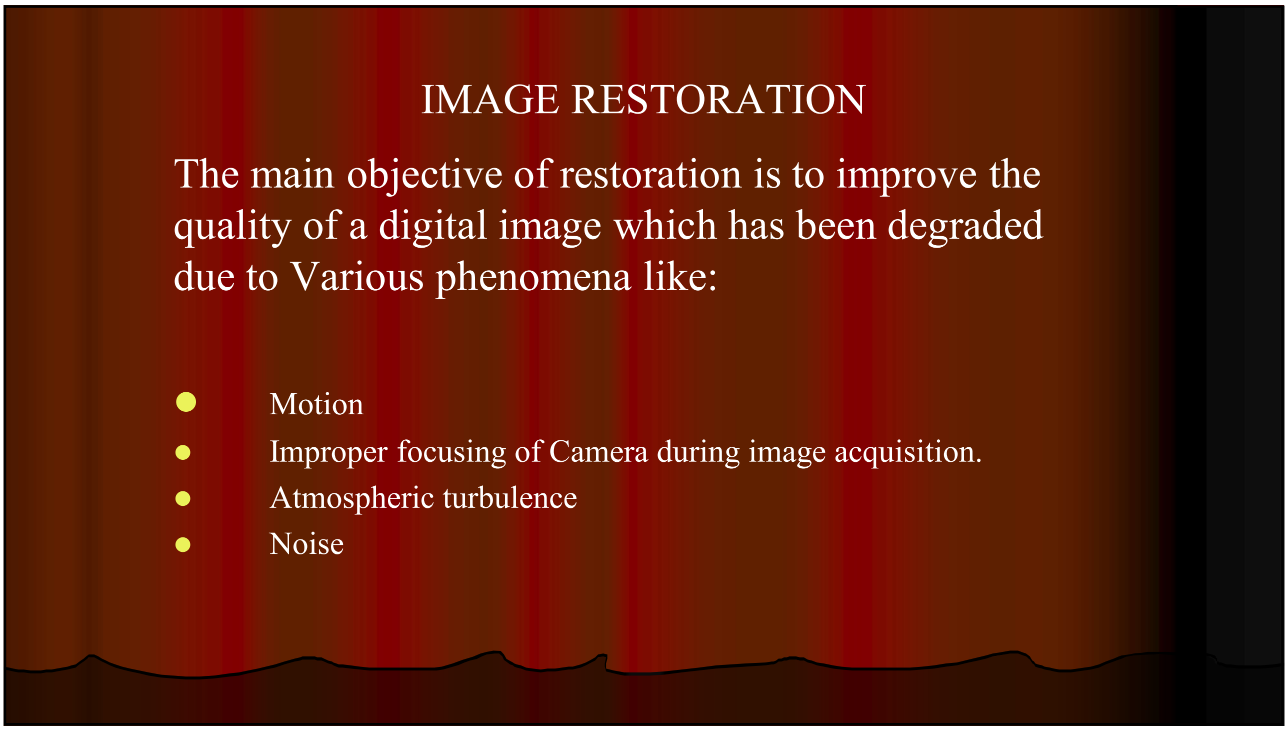


IMAGE RESTORATION

The main objective of restoration is to improve the quality of a digital image which has been degraded due to Various phenomena like:

- Motion
- Improper focusing of Camera during image acquisition.
- Atmospheric turbulence
- Noise

Enhancement versus Restoration

- Both processes try to improve an image in some predefined sense
- Image enhancement is largely a subjective process, while image restoration is for the most part an objective process

Enhancement:

(1)Manipulating an image in order to take advantage of the psychophysics of the human visual system.

(2)Techniques are usually "heuristic."

(3)Example: Contrast stretching, histogram equalization.

- **Restoration**:

(1)A process that attempts to reconstruct or recover an image that has been degraded by using some prior knowledge of the degradation phenomenon.

(2)Involves modeling the degradation process and applying the inverse process to recover the original image.

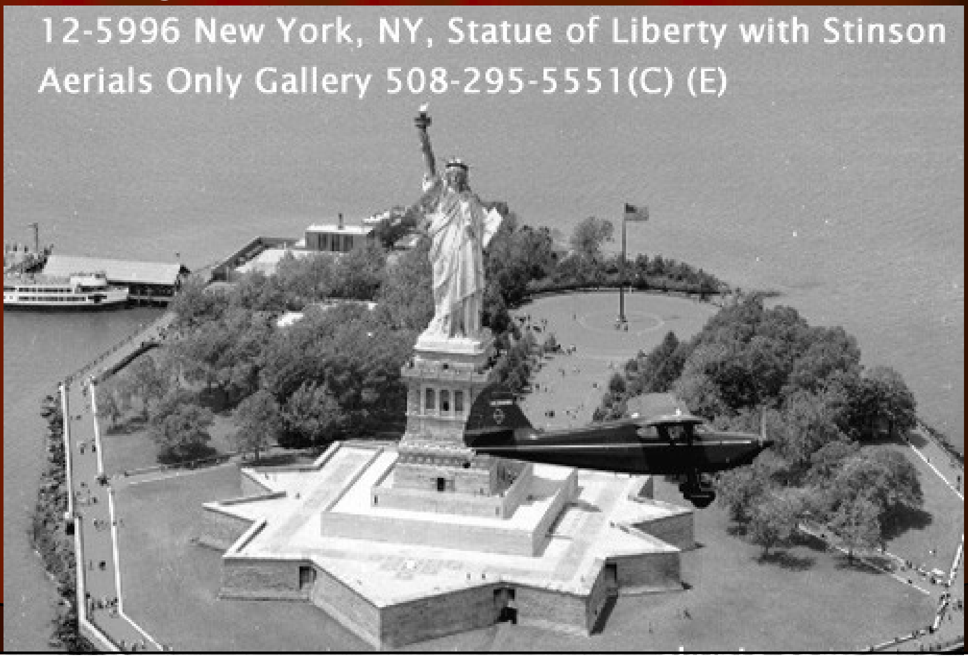
(3)A criterion for "goodness" is required that will recover the image in an optimal fashion with respect to that criterion.

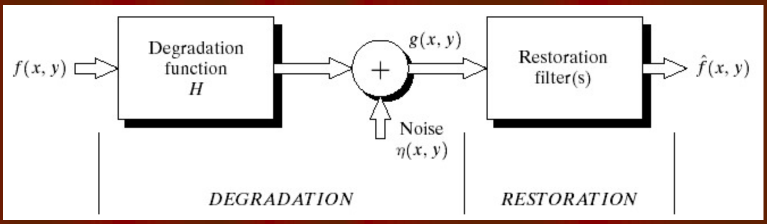
(4) Example: removal of blur by applying a deblurring function.

Problem:

- You want to know some image X.
- But you only have a corrupted version Y .
- How do you determine X from Y ?

Blurring due to uniform motion

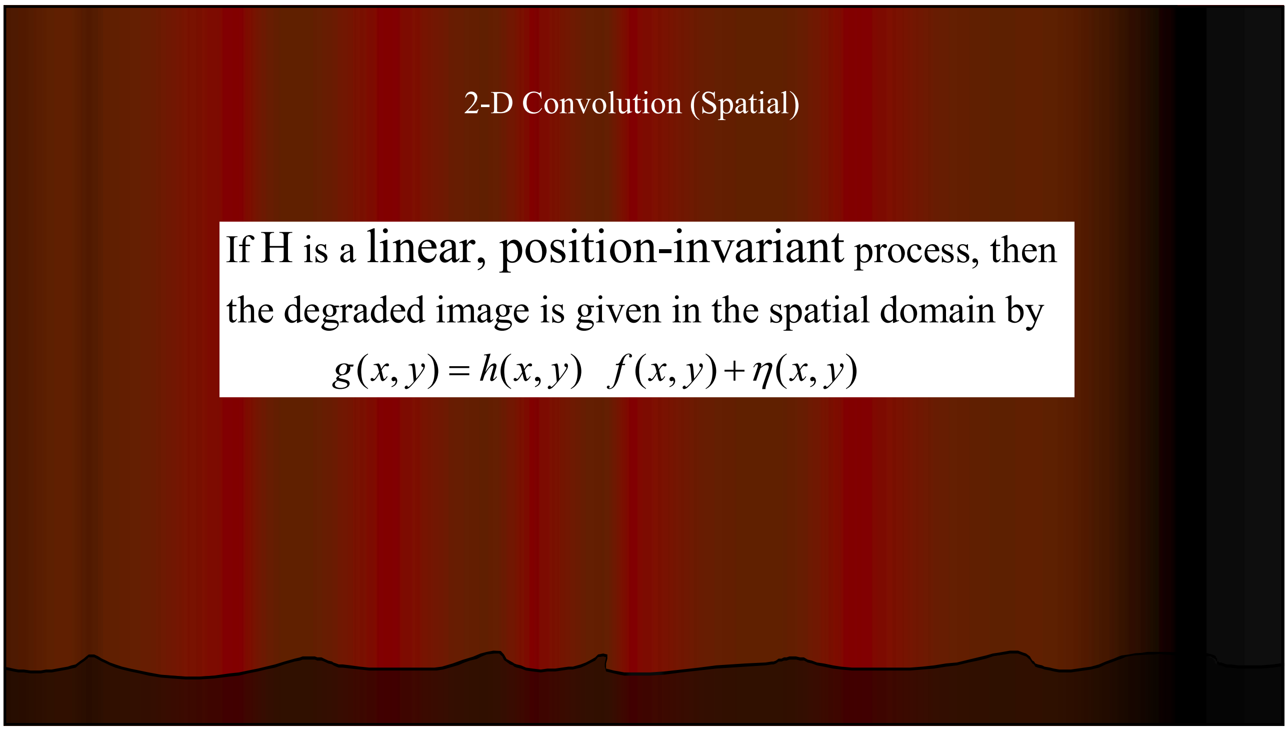

12-5996 New York, NY, Statue of Liberty with Stinson
Aerials Only Gallery 508-295-5551(C) (E)



Degradation model:

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

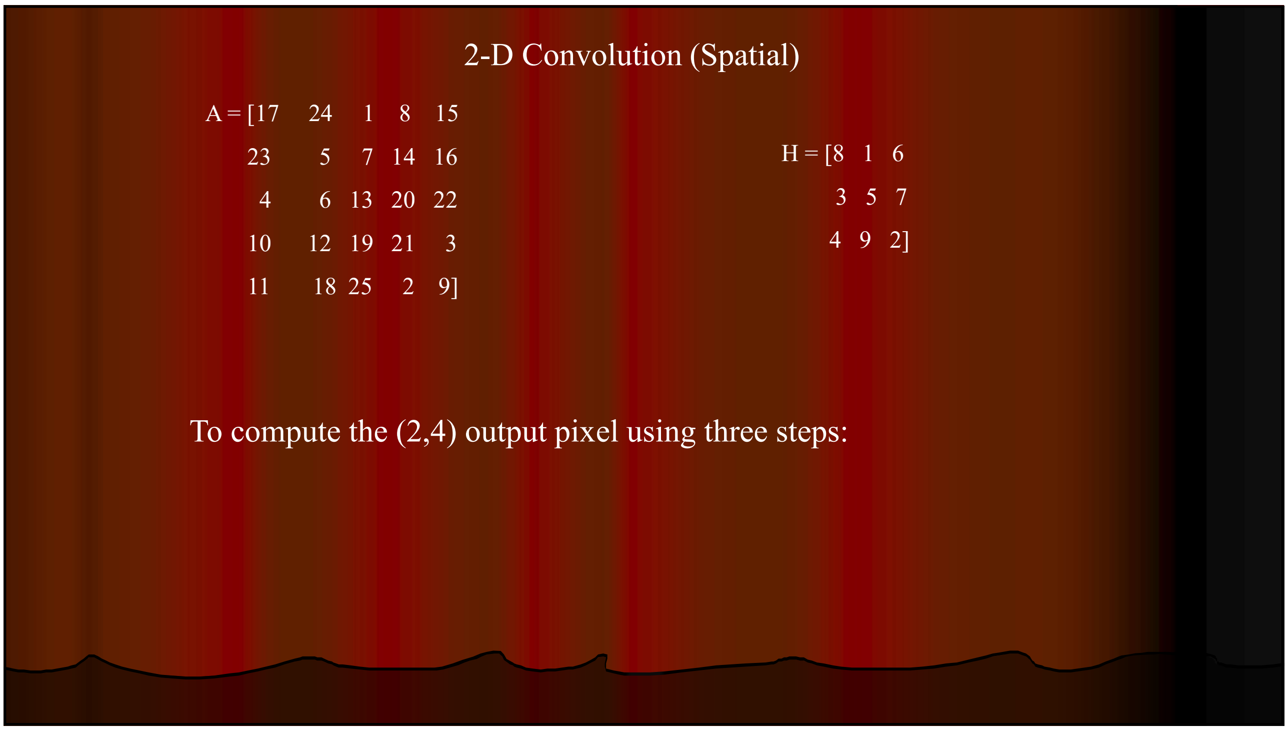
where $h(x,y)$ is a system that causes image distortion and $\eta(x,y)$ is noise.



2-D Convolution (Spatial)

If H is a linear, position-invariant process, then the degraded image is given in the spatial domain by

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



2-D Convolution (Spatial)

```
A = [17   24    1    8   15
     23    5    7   14   16
      4    6   13   20   22
     10   12   19   21    3
     11   18   25    2    9]
```

```
H = [8  1  6
     3  5  7
     4  9  2]
```

To compute the (2,4) output pixel using three steps:

2-D Convolution (Spatial)

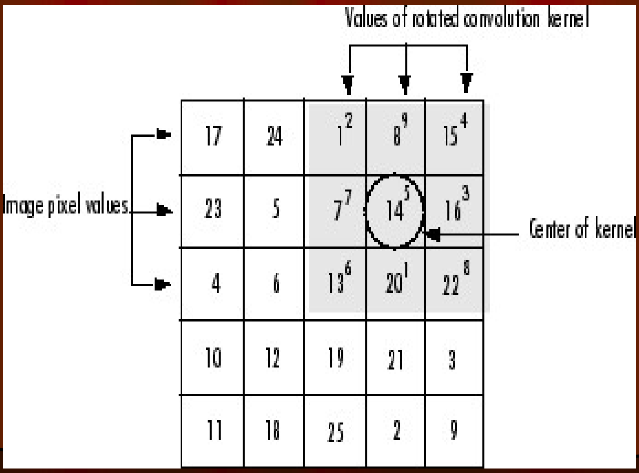
Rotate the convolution kernel 180 degrees about its center element.

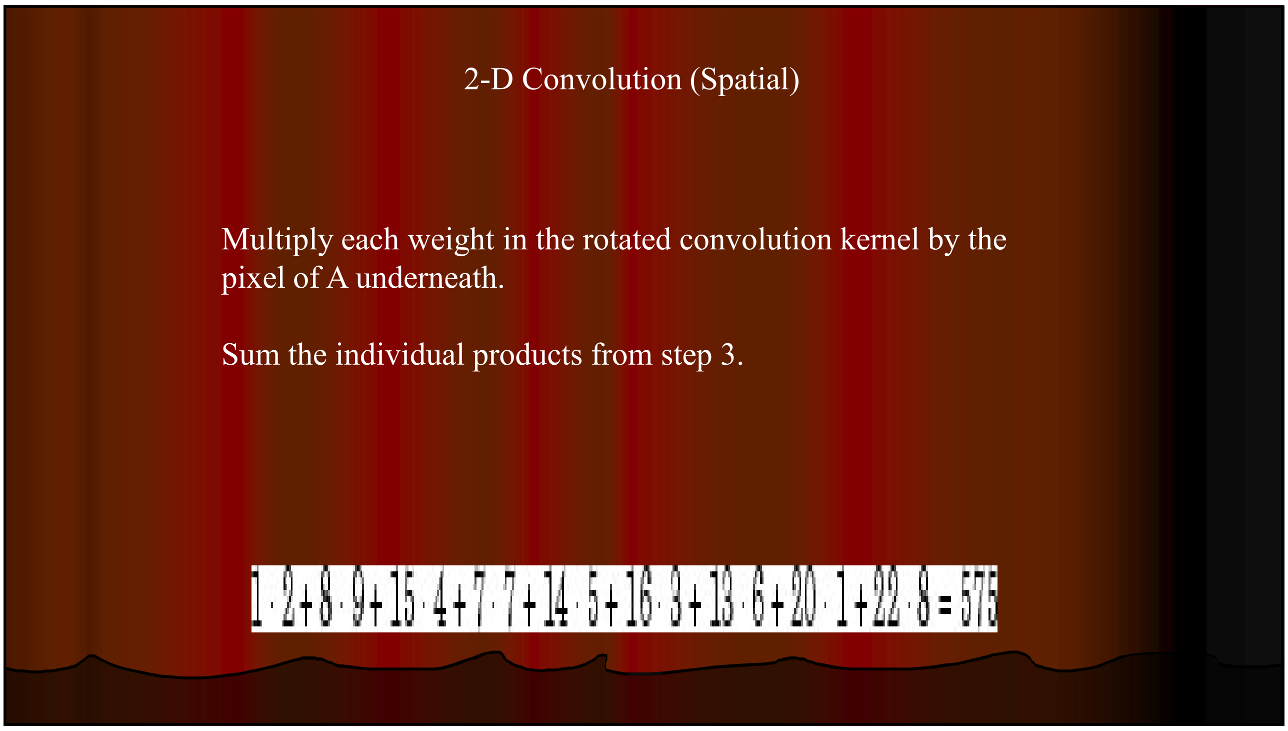
```
H = [8  1  6
     3  5  7
     4  9  2]
```

```
>> p = rot90(H)

    6    7    2
    1    5    9
    8    3    4

>> q = rot90(p)

    2    9    4
    7    5    3
    6    1    8
```

2-D Convolution (Spatial)

Slide the center element of the convolution kernel so that it lies on top of the (2,4) element of A.



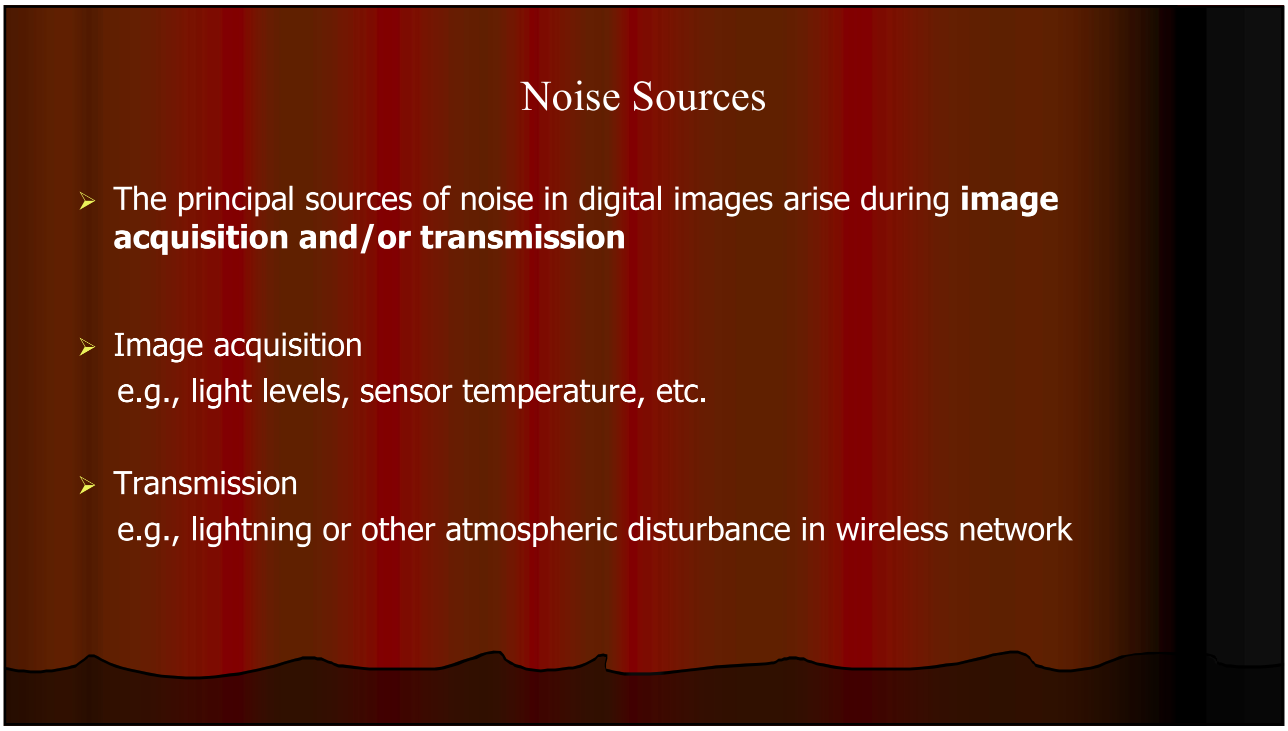


2-D Convolution (Spatial)

Multiply each weight in the rotated convolution kernel by the pixel of A underneath.

Sum the individual products from step 3.

$$1\cdot 2+8\cdot 9+15\cdot 4+7\cdot 7+14\cdot 5+16\cdot 3+13\cdot 6+20\cdot 1+22\cdot 8=575$$



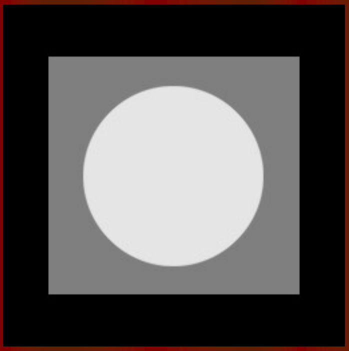
Noise Sources

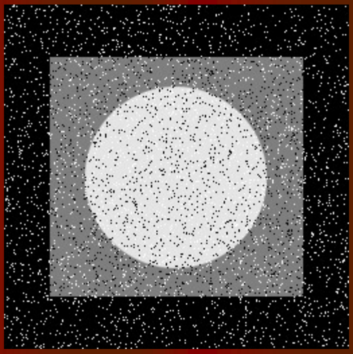
- The principal sources of noise in digital images arise during **image acquisition and/or transmission**
- Image acquisition
  e.g., light levels, sensor temperature, etc.
- Transmission
  e.g., lightning or other atmospheric disturbance in wireless network

Noise probability density functions

- Noises are taken as random variables
- Random variables
  - Probability density function (PDF)

Salt & Pepper Noise





Noise Probability Distribution (Salt & Pepper Noise)

The PDF of (bipolar) impulse noise is given by

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

if $b > a$, gray-level b will appear as a light dot, while level a will appear like a dark dot.

If either P_a or P_b is zero, the impulse noise is called *unipolar*

Salt & Pepper Noise

- a = imread('C:\lake2.bmp');
- a = double(a);
- a = mat2gray(a);
- imhist(a);
- b = imnoise(a,'Salt & Pepper');
- figure,imshow (b);
- figure, imhist(b)

Implementation: Salt & Pepper

```
function i= saltpepper(img1,a,b)
 [m,n]=size(img1);
 img1=mat2gray(double(img1));
 r= rand(m,n);
 x=find(r <=a);
 img1(x)=0;
 x=find(r >a & r <=(a+b));
 img1(x)=255;
 figure,imhist(img1);
 figure,imshow(img1);
imwrite(img1,'C:\board_salt.tif');
return i;
```

Noise Patterns

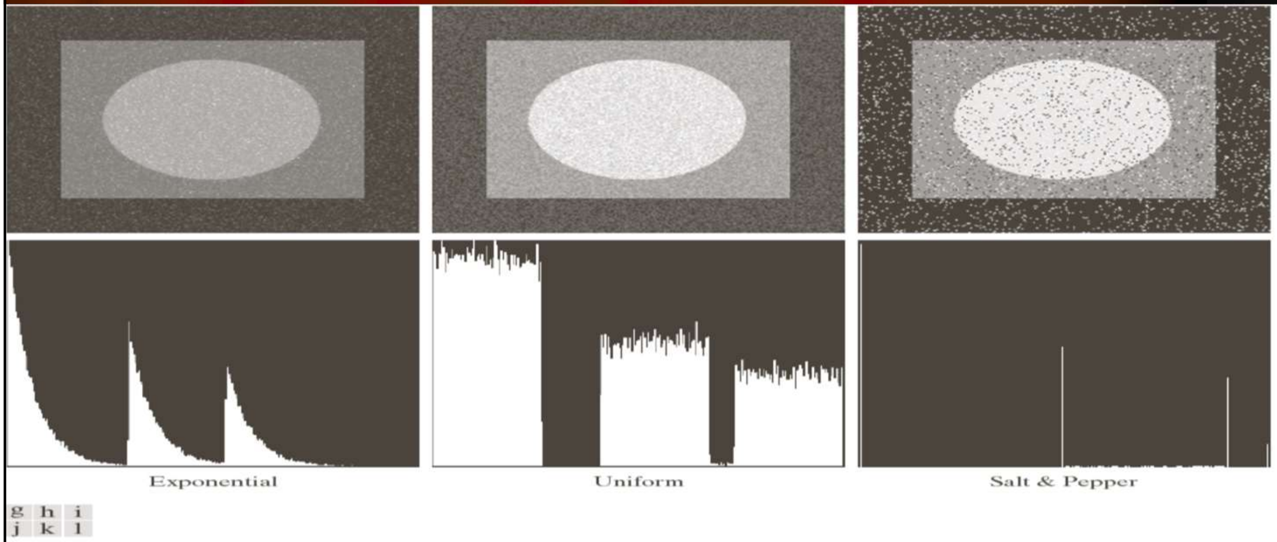


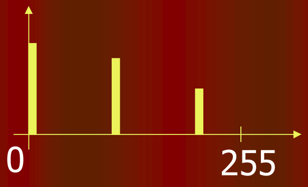
FIGURE 5.4 *(Continued)* Images and histograms resulting from adding exponential, uniform, and salt and pepper noise to the image in Fig. 5.3.

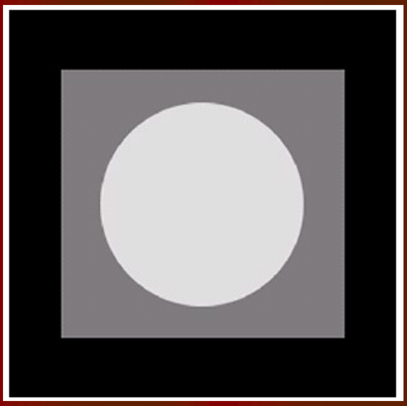
Test for noise behavior

- Uniform noise

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

Its histogram:





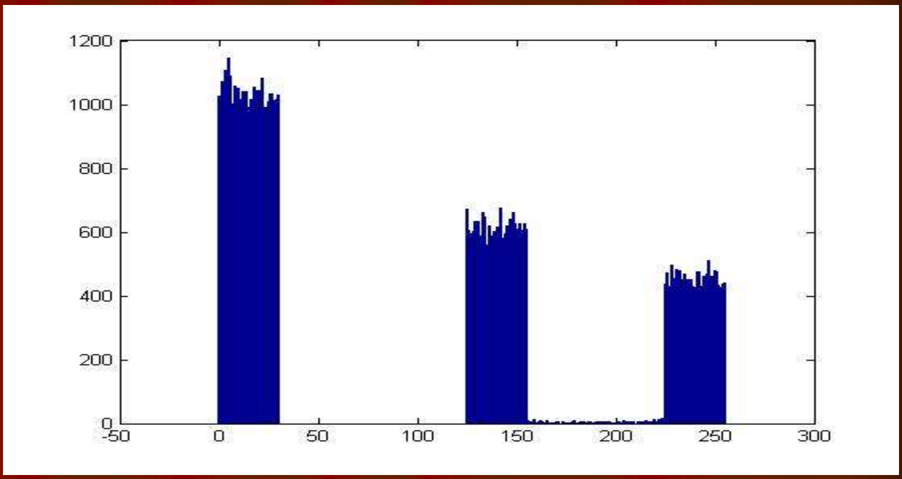
Test for noise behavior

```
a1=(imread('Fig0503 (original_pattern).tif'));
[m,n]=size(a1);
z=uint8(randi([10,40],m,n));

noizy_a=double(a1)+ double(z) ;
noisy=imhist(mat2gray(noizy_a));
original=imhist((a1));
```

Test for noise behavior

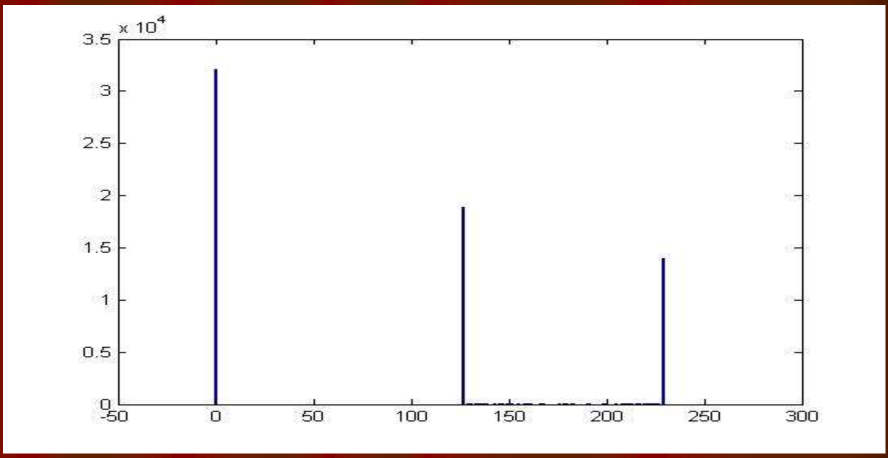
```
figure(1), bar(0:255,noisy)
figure(2) , bar(0:255,original)
figure (3)
subplot(211)
imshow(a1)
title('original image')
subplot(212)
imshow(mat2gray(noizy_a))
title('noisy image-uniform noise')
```

Test for noise behavior



Noisy Image

Test for noise behavior



Original Image

Noise Probability Distribution (Uniform Noise)

The PDF of uniform noise is given by

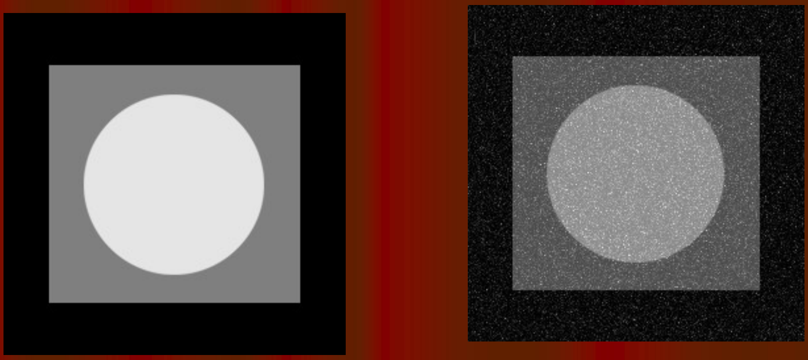
$$p(z) = \begin{cases} \dfrac{1}{b-a} & \text{for a} \le z \le b \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance of this density are given by

$$\bar{z} = (a+b)/2$$

$$\sigma^2 = (b-a)^2/12$$

Exponential Noise



Noise Patterns

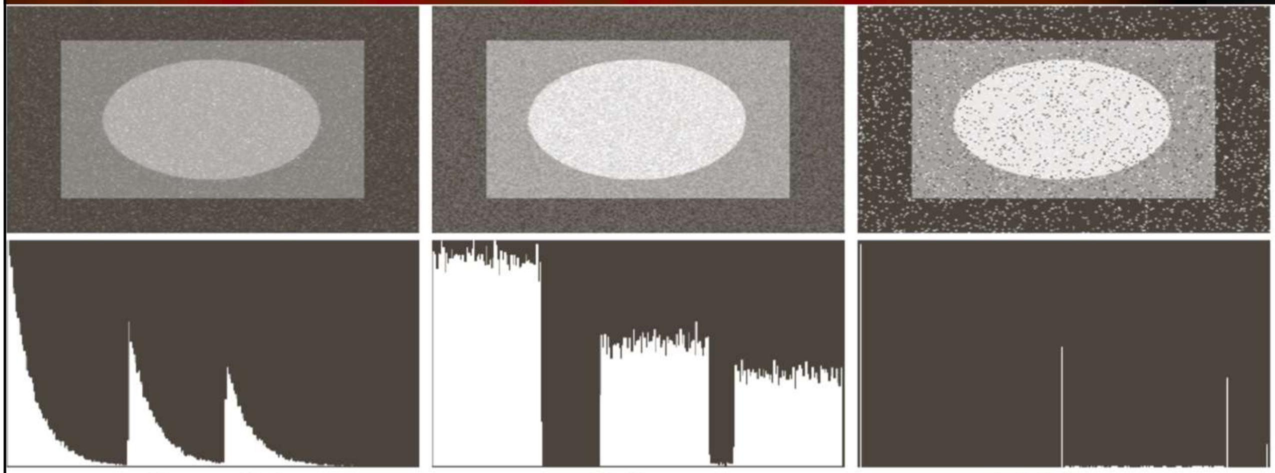


FIGURE 5.4 *(Continued)* Images and histograms resulting from adding exponential, uniform, and salt and pepper noise to the image in Fig. 5.3.

Noise Probability Distribution (Exponential Noise)

The PDF of exponential noise is given by

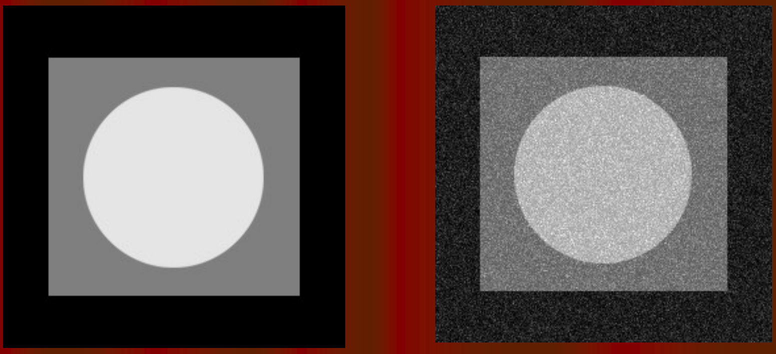
$$p(z) = \begin{cases} ae^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < a \end{cases}$$

The mean and variance of this density are given by

$$\bar{z} = 1/a$$

$$\sigma^2 = 1/a^2$$

Rayleigh Noise



Noise Probability Distribution (Rayleigh Noise)

The PDF of Rayleigh noise is given by

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

The mean and variance of this density are given by

$$\bar{z} = a + \sqrt{\pi b / 4}$$

$$\sigma^2 = \frac{b(4-\pi)}{4}$$

Gaussian Noise (Normal Noise)

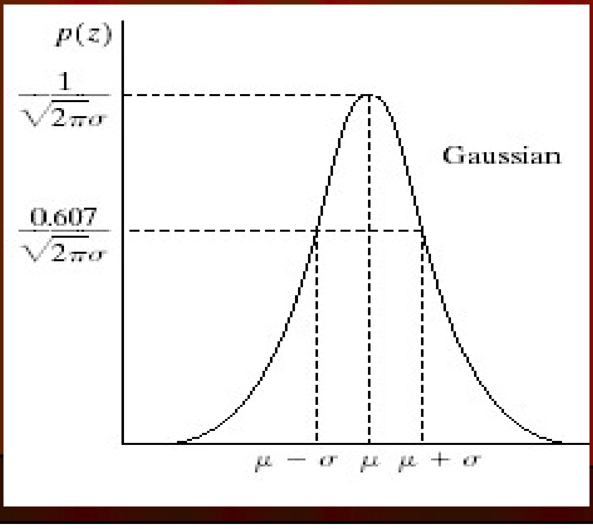
- The pdf of a Gaussian random variable *z* is given by:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$$

where *z* represents (noise) gray value, μ is the mean, and σ is its standard deviation. The squared standard deviation σ^2 is usually referred to as variance.

Gaussian Noise

- For a Gaussian pdf, approximately 70% of its values will be in the range $[(\mu-\sigma),(\mu+\sigma)]$, and 95% of its values will be in the range $[(\mu-2\sigma),(\mu+2\sigma)]$



Gaussian Noise

```
clc;
clear;
img2=imread('D:\Image Processing\IP_sttp\lena_gray_256.tif');
imshow(img2);
img2=double(img2);
img2 = mat2gray(img2);
r = imnoise(img2, ‘Gaussian’, 20/256, 0.0001);
imshow(r);
imhist(r);
```

Gamma Noise

- The pdf of Erlang noise is given by:

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

where, $a > 0$, b is an integer and "!" represents factorial.

- The mean and variance are given by:

$$\mu = b/a$$

$$\sigma^2 = b/a^2$$

Noise Patterns

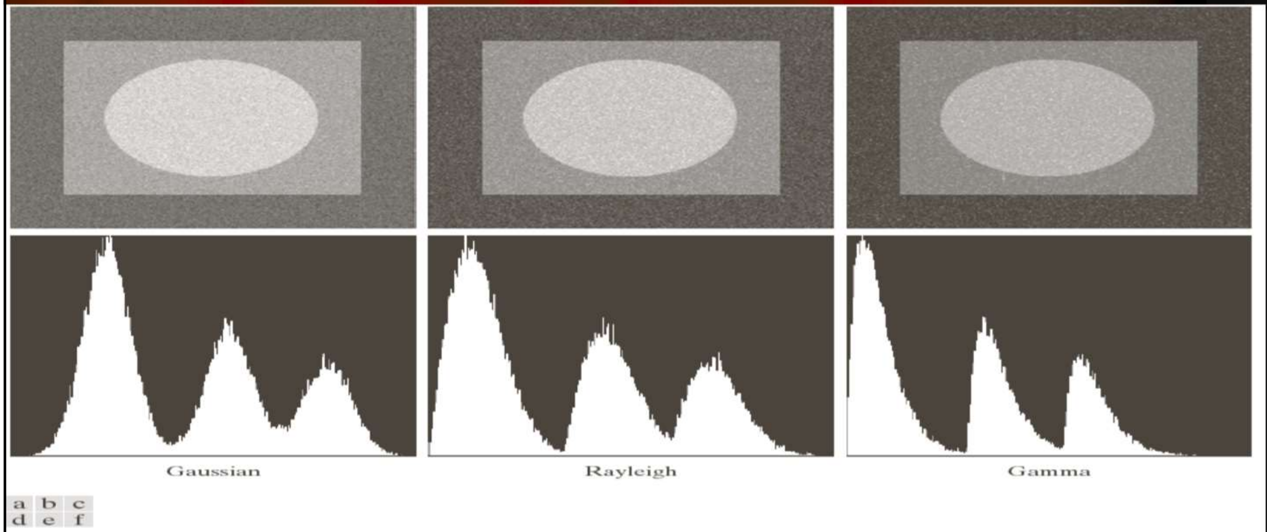
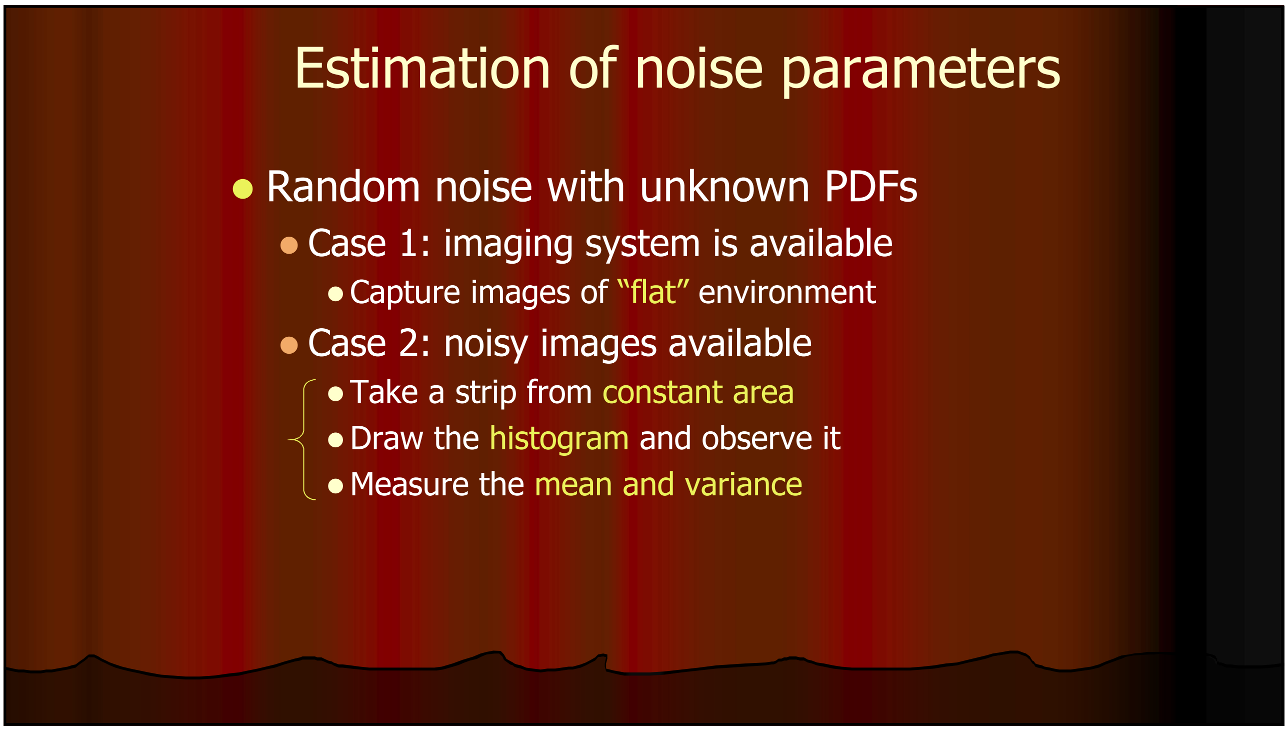


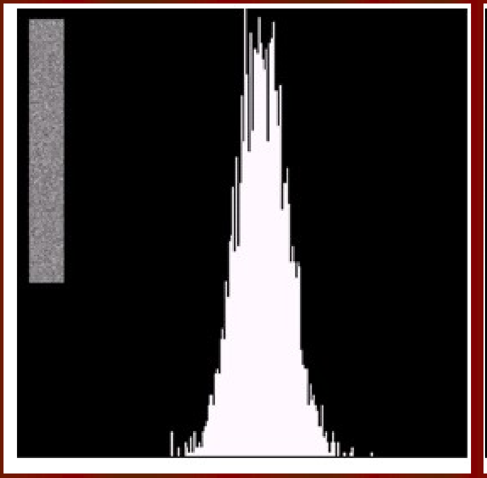
FIGURE 5.4 Images and histograms resulting from adding Gaussian, Rayleigh, and gamma noise to the image in Fig. 5.3.



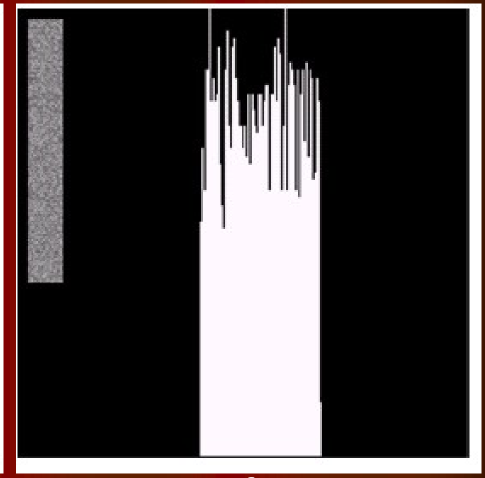
Estimation of noise parameters

- Random noise with unknown PDFs
  - Case 1: imaging system is available
    - Capture images of "flat" environment
  - Case 2: noisy images available
    - Take a strip from constant area
    - Draw the histogram and observe it
    - Measure the mean and variance

Observe the histogram



Gaussian



uniform

Estimation of Noise Parameters

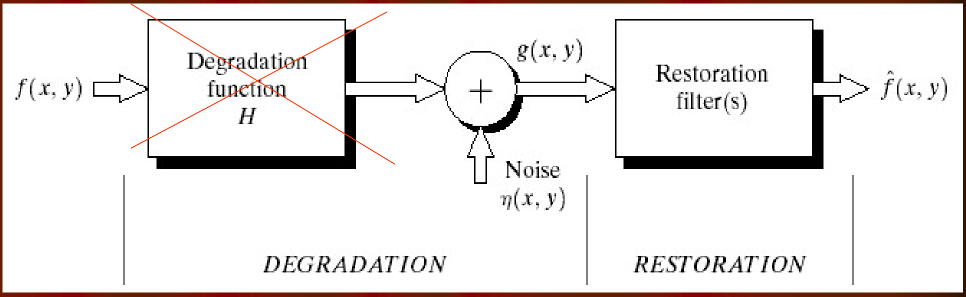
Consider a subimage denoted by S, and let $p_s(z_i)$, $i = 0, 1, \ldots, L-1$, denote the probability estimates of the intensities of the pixels in S. The mean and variance of the pixels in S:

$$\bar{z} = \sum_{i=0}^{L-1} z_i p_s(z_i)$$

and

$$\sigma^2 = \sum_{i=0}^{L-1} (z_i - \bar{z})^2 p_s(z_i)$$

Additive noise only



$$\begin{cases} g(x,y)=f(x,y)+\eta(x,y) \\ G(u,v)=F(u,v)+N(u,v) \end{cases}$$

Spatial filters for de-noising additive noise

- Skills similar to image enhancement
- Mean filters
- Order-statistics filters
- Adaptive filters

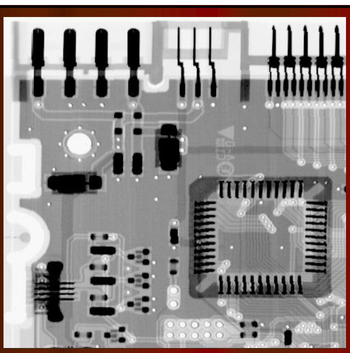
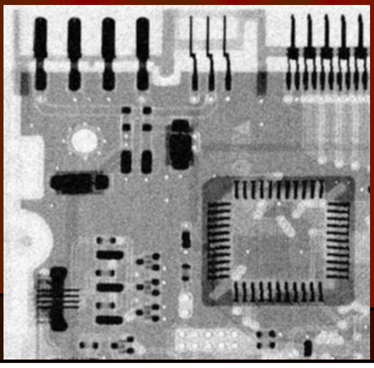
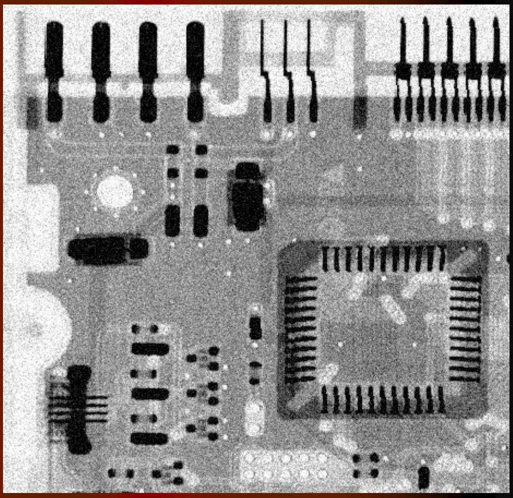
Spatial Filtering: Mean Filter

Let S_{xy} represent the set of coordinates in a rectangle subimage window of size $m \times n$, centered at (x, y).

Arithmetic mean filter

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

Spatial Filtering: Mean Filter

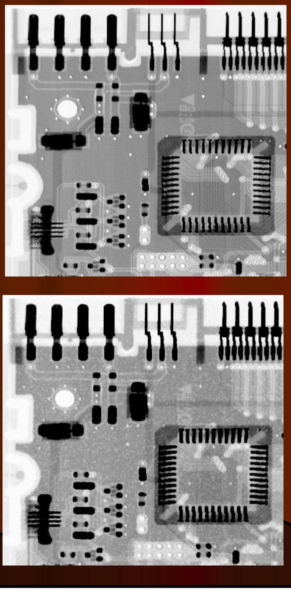
Spatial Filtering: Harmonic Mean Filter

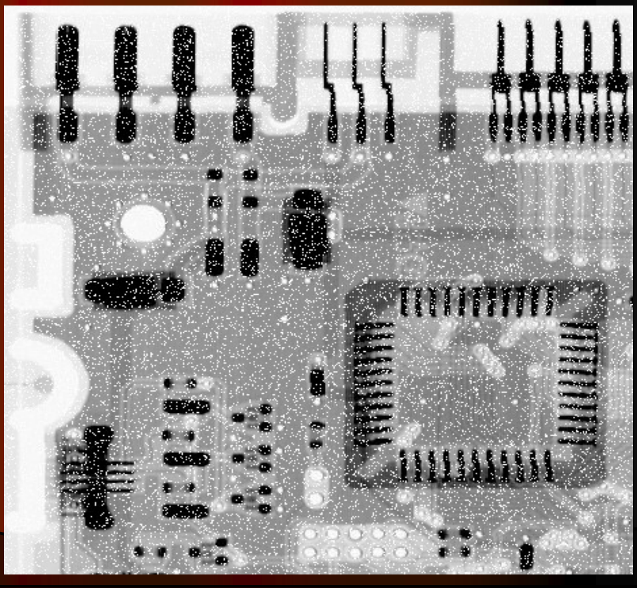
Harmonic mean filter

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

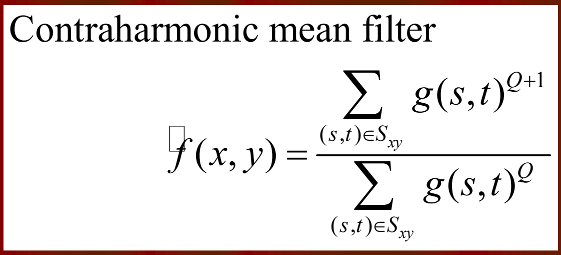
It works well for salt noise, but fails for pepper noise. It does well also with other types of noise like Gaussian noise.

Spatial Filtering: Harmonic Mean Filter





Spatial Filtering: Contra Harmonic Mean Filter

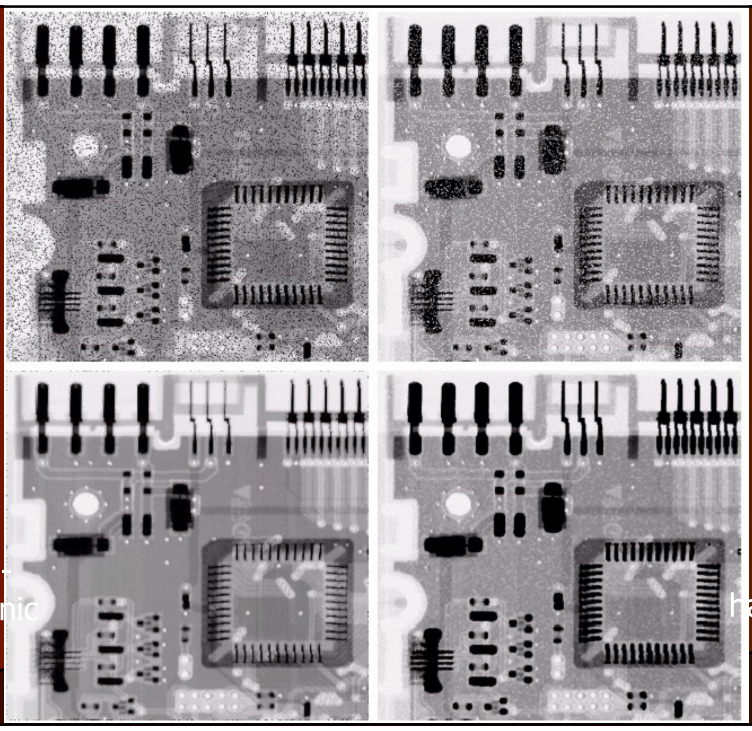


Contraharmonic mean filter

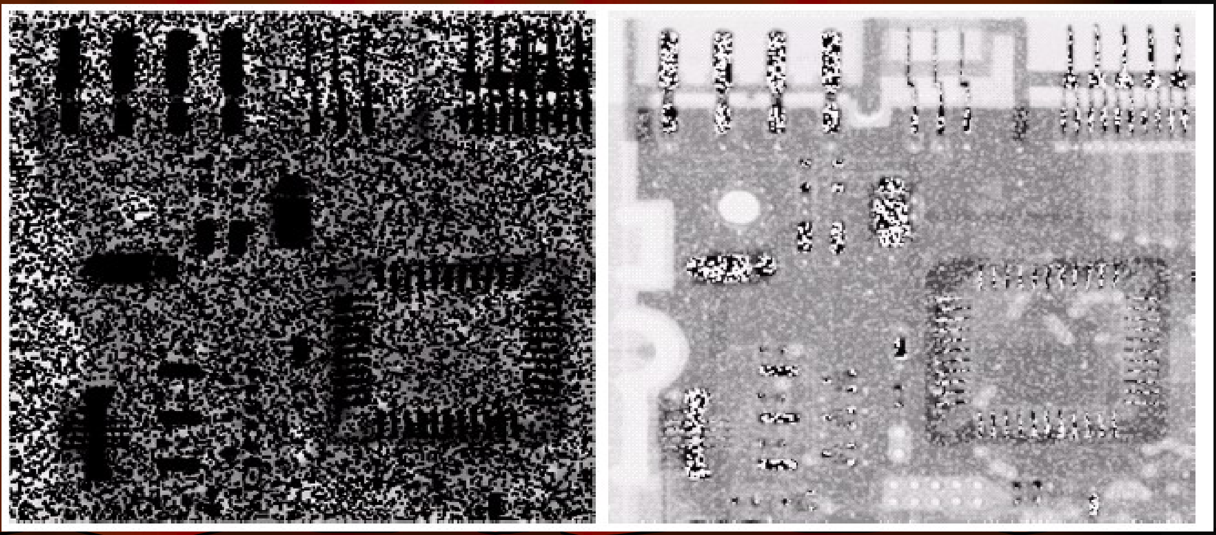
$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

Q is the order of the filter.

It is well suited for reducing the effects of salt-and-pepper noise. Q>0 for pepper noise and Q<0 for salt noise.



Wrong sign in contra-harmonic filtering



Spatial Filtering: Geometric Mean Filter

Geometric mean filter

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

Generally, a geometric mean filter achieves smoothing comparable to the arithmetic mean filter, but it tends to lose less image detail in the process

Spatial Filtering: Geometric Mean Filter