

EE 4211 Computer Vision

Lecture 6: Image restoration

Semester A, 2020-2021

Schedules

Week	Date	Topics
1	Sep. 4	Introduction/Imaging
2	Sep. 11	Image enhancement in spatial domain
3	Sep. 18	Image enhancement in frequency domain (HW1 out)
4	Sep. 25	Morphological processing
5	Oct. 2	Image restoration(HW1 due)
6	Oct. 9	Image restoration
7	Oct. 16	Midterm (no tutorials this week)
8	Oct. 23	Edge detection (HW2 out, illustrate the project)
9	Oct. 30	Image segmentation (HW2 due)
10	Nov. 6	Face recognition with PCA, LDA (tutorial on deep learning framework)
11	Nov. 13	Face recognition based on deep learning Image segmentation based on deep learning (tutorial on coding)
12	Nov. 20	Object detection with traditional methods (Quiz) Object detection based on deep learning
13	Nov. 27	Project presentation
14	Dec. 4	Review and Summary

Lecture outline

- Degradation & Restoration Process Models
- Noise Models
- Restoration in the presence of noise only (spatial filtering)
- Estimation of the degeneration function
- Inverse filtering
- Wiener filtering

Image Restoration

- To restore the original image that has been degraded
- Image restoration vs image enhancement
 - Objective process vs. subjective process
 - A prior knowledge vs heuristic process
 - A prior knowledge of the degradation phenomenon is considered
 - Model the degradation and apply the inverse process to recover the original image

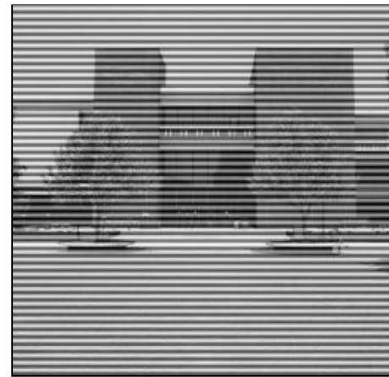
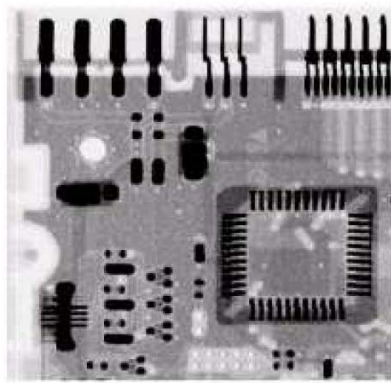
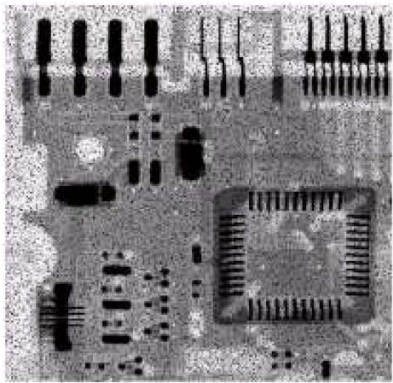
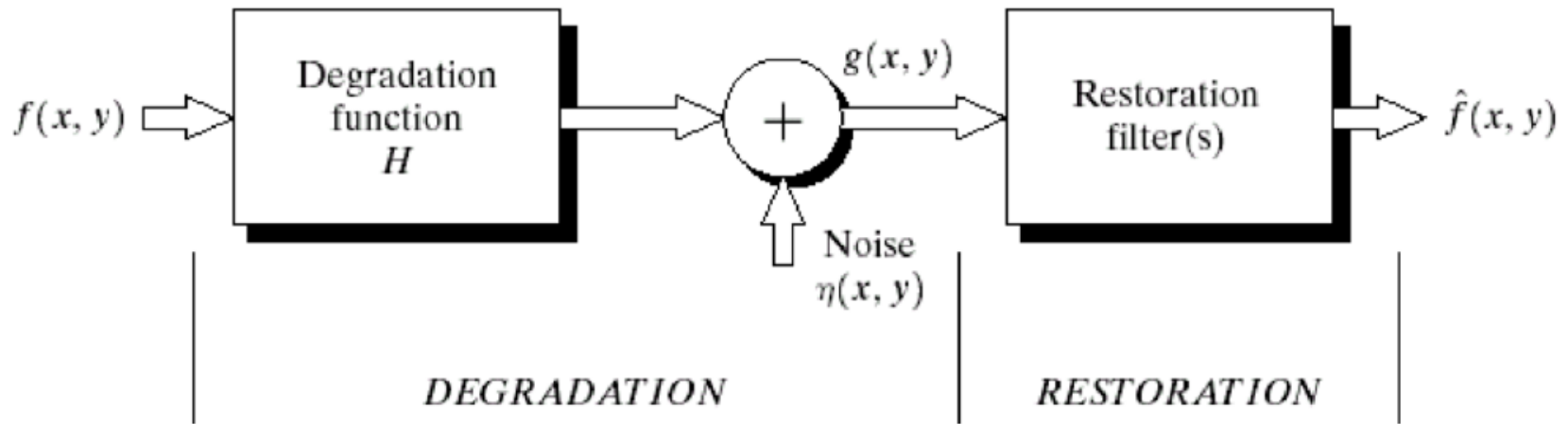


Image Restoration



- Consists of 2 parts – Degradation function & Noise function
- General model in spatial domain:

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

- $g(x, y)$: degraded image, $h(x, y)$: degradation function
- $f(x, y)$: original image, $n(x, y)$: additive noise function

Degradation Model in Freq. Domain

- General model in frequency domain:

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

- $G(x,y)$: Fourier transform of degraded image,
 - $H(x,y)$: Fourier transform of degradation function,
 - $F(x,y)$: Fourier transform of original image,
 - $N(x,y)$: Fourier transform of additive noise function
- What needs to be done??
 - **FIND** Degradation function and Noise model

Lecture outline

- Degradation & Restoration Process Models
- Noise Models
- Restoration in the presence of noise only (spatial filtering)
- Estimation of the degeneration function
- Inverse filtering
- Wiener filtering

Noise Models

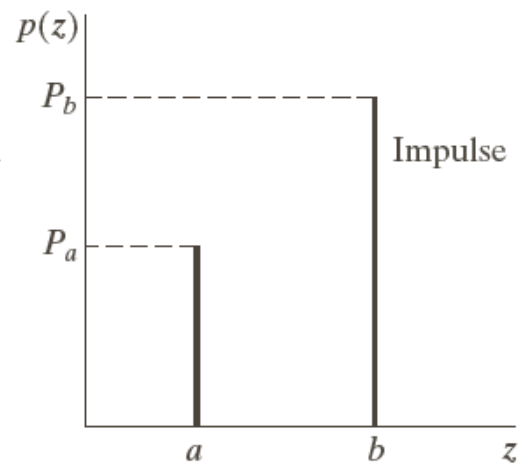
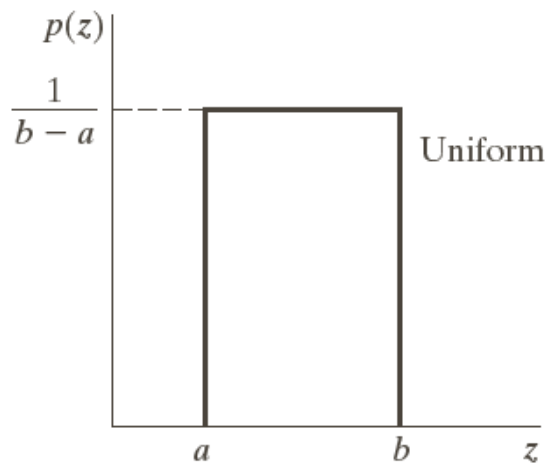
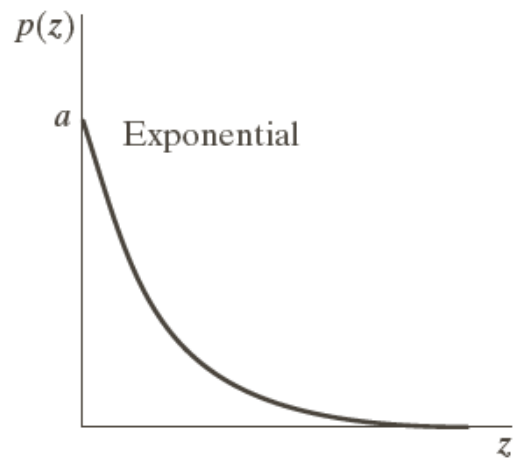
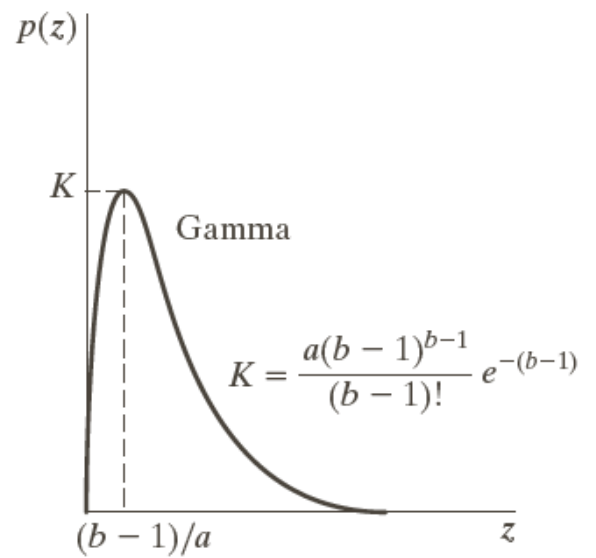
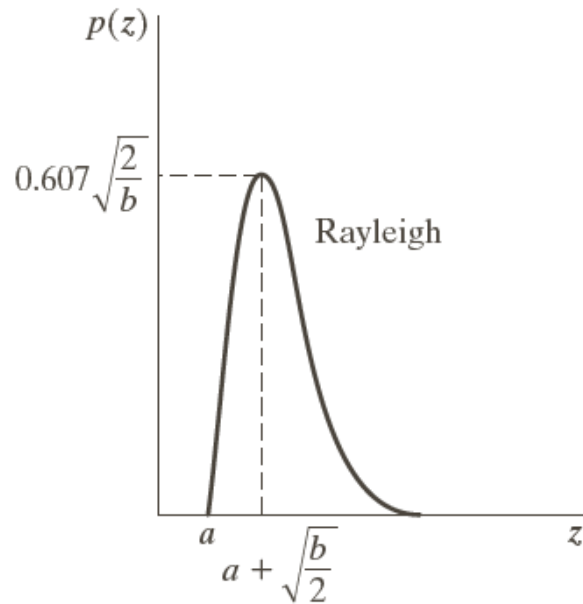
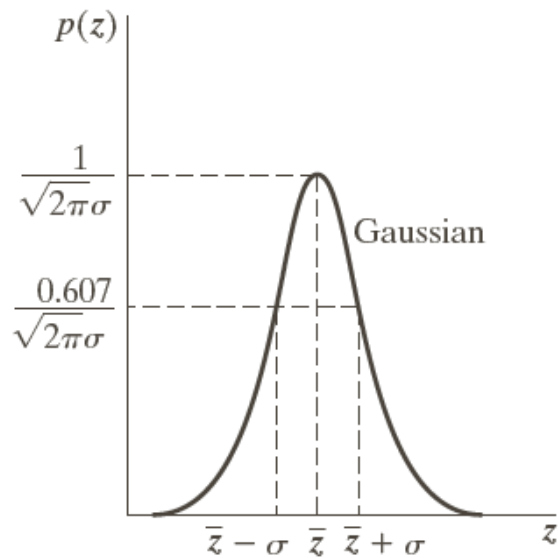
- **Noise** – Any undesired information that contaminates an image
- Variety of sources:
 - **Digital image acquisition process**, e.g. For CCD (charged coupled device) camera, electronics signal fluctuations in detector, caused by thermal energy and light levels
 - During **image transmission**, e.g. Wireless network, may be corrupted by lightning or other atmospheric disturbance

Noise Models

- Assumptions of Noise models in this course:
 - Random variable
 - Noise is independent of spatial coordinates
 - Noise is uncorrelated w.r.t the image (pixel values)
- Spatial Noise Descriptor – concern of the statistical behavior of the intensity values in the noise component of the model
 - Characterized by probability density function (PDF)

Noise Models

- Typical image noise models that can be modeled:
 - Gaussian
 - Rayleigh
 - Erlang (Gamma)
 - Exponential
 - Uniform
 - Impulse (salt-and-pepper)



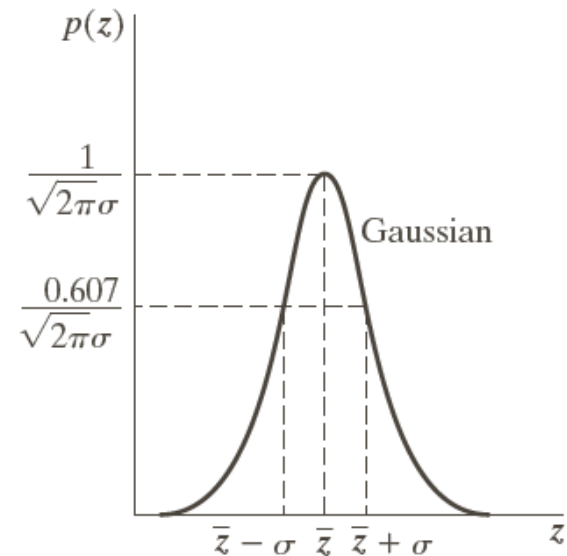
Images and corresponding pdfs

Gaussian noise

- The PDF of a Gaussian random variable, z , is given by

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

- z represents gray level,
- μ is the mean or average value of z ,
- σ is its standard deviation.
- Gaussian white noise: mean value 0



Rayleigh noise

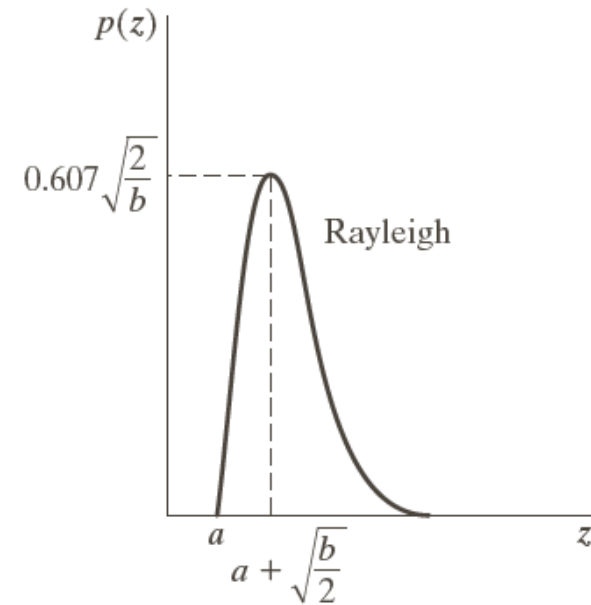
- The PDF of Rayleigh noise is given by

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

- The mean and variance of this density are given by

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

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



Erlang(Gamma) noise

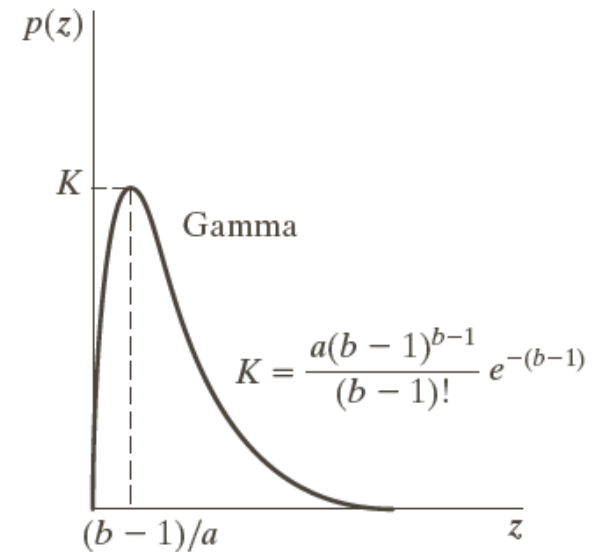
- The PDF of Gamma noise is given by

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

- The mean and variance of this density are given by

$$\mu = \frac{b}{a}$$

$$\sigma^2 = \frac{b}{a^2}$$



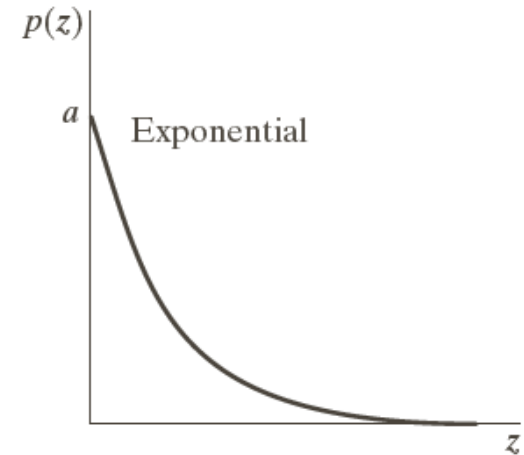
Exponential noise

- The PDF of exponential noise is given by

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

- The mean and variance of this density function are

$$\mu = \frac{1}{a} \quad \sigma^2 = \frac{1}{a^2}$$



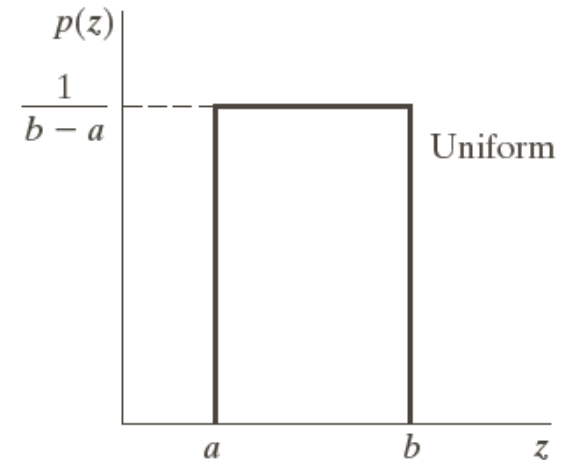
Uniform noise

- The PDF of uniform noise is given by

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

- The mean of this density function is given by

$$\mu = \frac{a+b}{2} \quad \sigma^2 = \frac{(b-a)^2}{12}$$



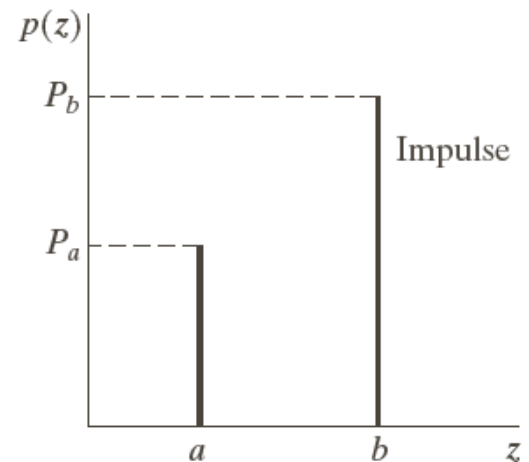
Impulse (salt-and-pepper) noise

- The PDF of (bipolar) impulse noise is given by

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

where

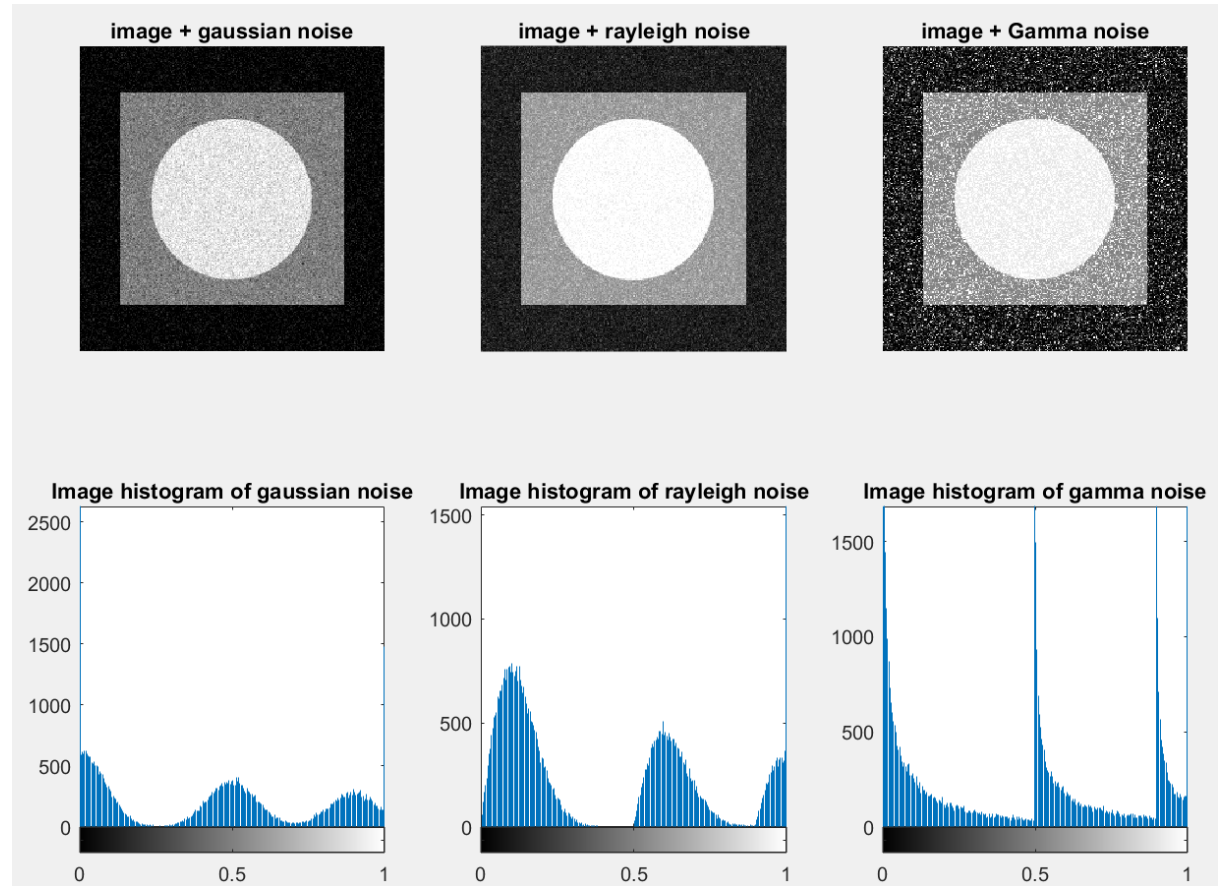
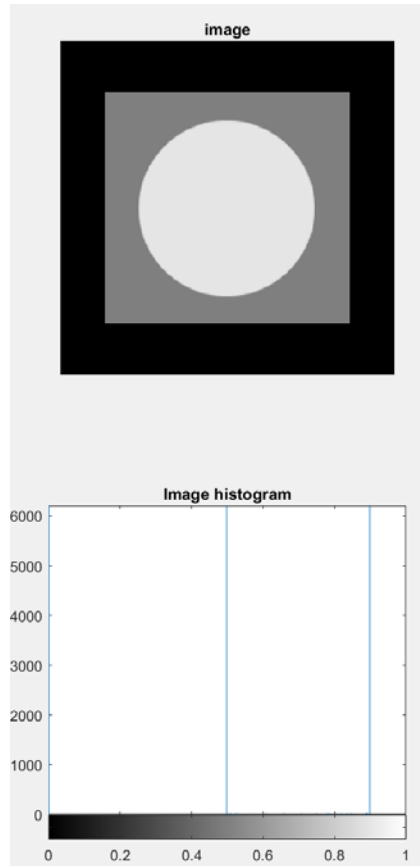
$$b > a$$



Noise Models

- Different PDFs provide useful tools for modeling a broad range of noise corruption situations:
- **Gaussian noise**: due to factors such as electronic circuit noise, sensor noise (due to poor illumination or high temperature)
- **Rayleigh noise**: model noise in range imaging
- **Exponential and Gamma**: laser imaging
- **Impulse noise**: found in quick transients (e.g., faulty switches)

Example: Noise Models



```
I = im2double(imread('Fig0503 (original_pattern).tif'));
[m n]=size(I);
```

%% Gaussian noise

```
% mean and std are the parameters of Gaussian distribution
```

```
mean = 0;
```

```
std = 0.08;
```

```
gaussian_noise = mean + std.*randn(m,n);
```

```
J1 = I + gaussian_noise;
```

```
subplot(2,3,1);|
imshow(J1)
title('image + gaussian noise')
```

```
subplot(2,3,4);
imhist(J1);
title('Image histogram of gaussian noise')
set(gca,'YTickLabel',[]);
```

%% Rayleigh noise

```
% a is the translation parameter of Rayleigh noise
```

```
% b is the shape parameter, it is also the sigma mentioned in WIKIPEDIA
```

```
a=0;
```

```
b=0.1;
```

```
rayleigh_noise = a+raylrnd(b,m,n);
```

```
% a = -0.2;|
% b = 0.03;
% rayleigh_noise = a+(-b .* log(1 - rand(m,n))).^0.5;
J2 = I + rayleigh_noise;
```

```
subplot(2,3,2);
imshow(J2)
title('image + rayleigh noise')
```

```
subplot(2,3,5);
imhist(J2);
title('Image histogram of rayleigh noise')
set(gca,'YTickLabel',[]);
```

%% Gamma noise

```
% a is the shape parameter, which is denoted as k in WIKI
```

```
% b is the scale parameter, which is denoted as theta in WIKI
```

```
a = 0.4;
```

```
b = 0.4;
```

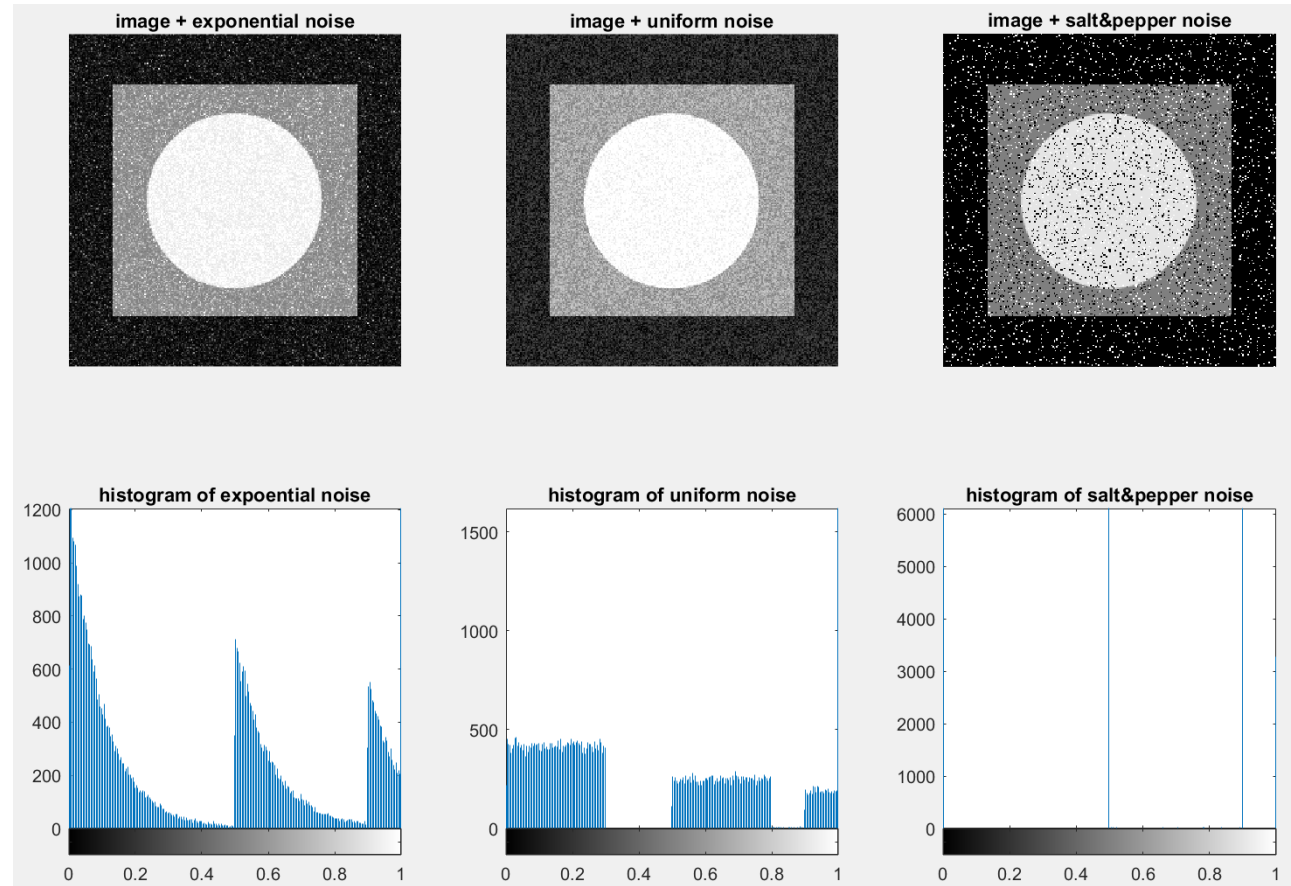
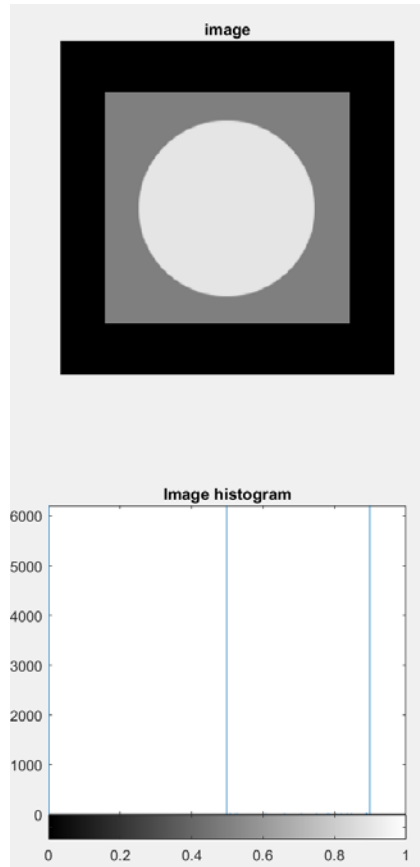
```
gamma_noise = gamrnd(a,b,m,n);
```

```
J3 = I + gamma_noise;
```

```
subplot(2,3,3);|
imshow(J3)
title('image + Gamma noise')
```

```
subplot(2,3,6);
imhist(J3);
set(gca,'YTickLabel',[]);
title('Image histogram of gamma noise')
```

Example: Noise Models



%% Exponential noise

% lambda is the parameter used in exponential distribution

lambda = 0.1;

exponential_noise = exprnd(lambda, m, n);

J4 = I + exponential_noise;

figure

subplot(2,3,1);

imshow(J4)

title('image + exponential noise')

subplot(2,3,4);

imhist(J4);

set(gca, 'YTickLabel', []);

title('histogram of exponential noise')

%% Uniform noise

% a is the lower bound

% b is the upper bound

a = 0;

b = 0.3;

uniform_noise = a + (b-a)*rand(m,n);

J5 = I + uniform_noise;

subplot(2,3,2);

imshow(J5)

title('image + uniform noise')

subplot(2,3,5);

imhist(J5);

set(gca, 'YTickLabel', []);

title('histogram of uniform noise')

p_salt = 0.05;

p_pepper = 0.08;

tmp_mask = rand(m,n);

salt_idx = find(tmp_mask <= p_salt);

pepper_idx = find(tmp_mask > p_salt & tmp_mask <= p_salt+p_pepper);

J6 = I;

J6(salt_idx) = 1;

J6(pepper_idx) = 0;

subplot(2,3,3);

imshow(J6)

title('image + salt&pepper noise')

subplot(2,3,6);

imhist(J6)

set(gca, 'YTickLabel', []);

title('histogram of salt&pepper noise')

Lecture outline

- Degradation & Restoration Process Models
- Noise Models
- Restoration in the presence of noise only (spatial filtering)
- Estimation of the degeneration function
- Inverse filtering
- Wiener filtering

Restoration in Presence of Noise Only

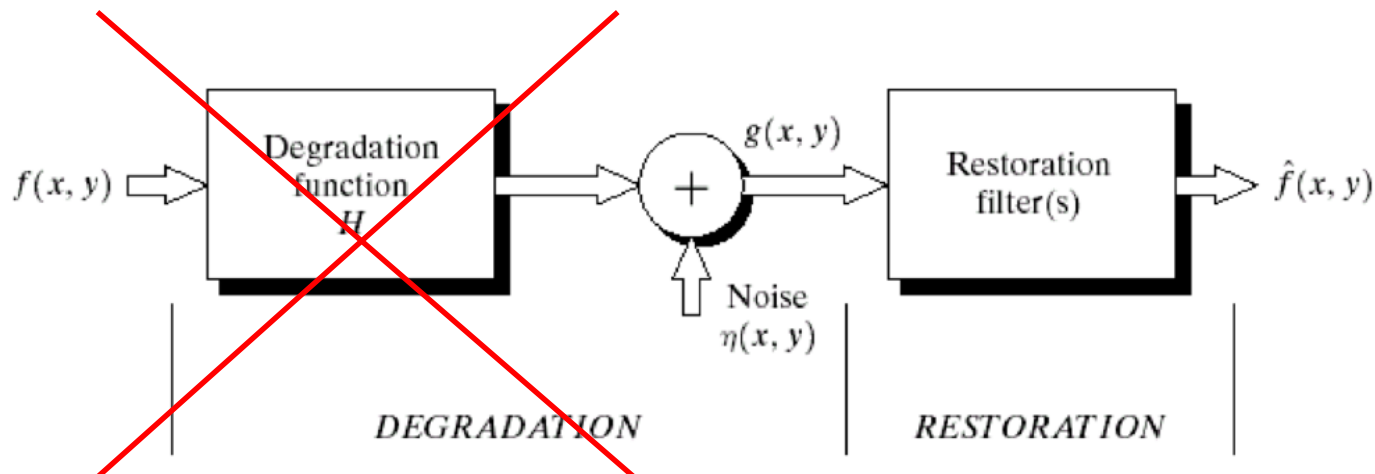
- Consider an image degraded with only additive noise. The degradation model is further simplified as

- Spatial domain

$$g(x, y) = f(x, y) + n(x, y)$$

- Frequency domain

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



Noise Removal with Spatial Filters

- Spatial filters can effectively remove various types of noise in digital images
- Typically operate on small neighborhoods, from 3x3 to 11x11
- Common filters to remove noise
 - Mean filters
 - Order-statistics filters
 - Adaptive mean filters

Noise Removal with Spatial Filters

- Common filters to remove noise
 - Mean filters
 - Order-statistics filters
 - Adaptive mean filters

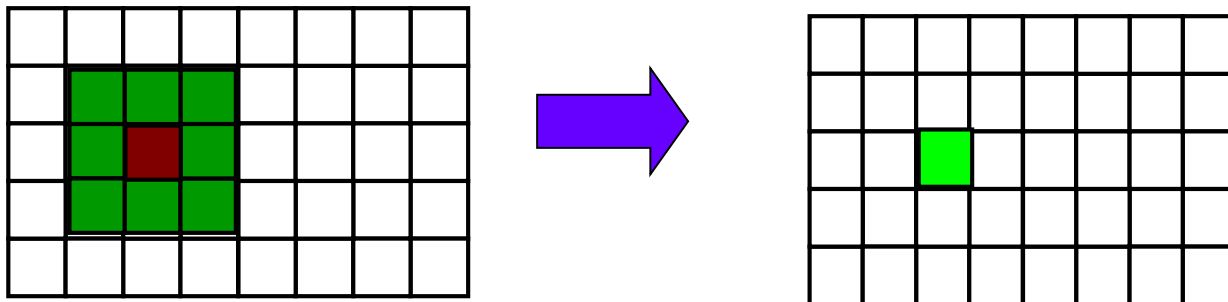
Mean Filters

■ Arithmetic Mean Filter

- Computes the average value of the corrupted image $g(x,y)$ in area defined by S_{xy} . S_{xy} represents the set of coordinates in a rectangular subimage window of size $m \times n$, centred at point (x,y)

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

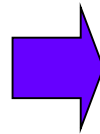
- For random noise



Mean Filters

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

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



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

Mean Filters

■ Geometric Mean Filter

- The image restored using a geometric mean filter:

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

- For random noise: lose less detail than Arithmetic Mean
- **Ineffective in the presence of pepper noise** (if very low values present in the window, the equation will return a very small number)

1	200	200
200	200	200
200	200	200



1	200	200
200	111	200
200	200	200

Mean Filters

■ Harmonic Mean Filter

- The image restored using a harmonic mean filter:

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

- Works well for salt noise (if very **high values** present in the window, the below value will return a very small number).
- Works with Gaussian noise
- Retains detail better than arithmetic mean filter

1	1	1
1	1	1
1	1	200



1	1	1
1	1.124	1
1	1	200

Mean Filters

■ Contraharmonic Mean Filter

- The image restored with a contraharmonic mean filter:

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

where Q is called the order of filter

- For Q=0, the filter reduces to an arithmetic mean filter.
- For Q=-1, the filter reduces to a harmonic mean filter.
- **Positive Q: suitable for eliminating pepper noise.**
- Negative Q: suitable for eliminating salt noise.

1	200	200
200	200	200
200	200	200

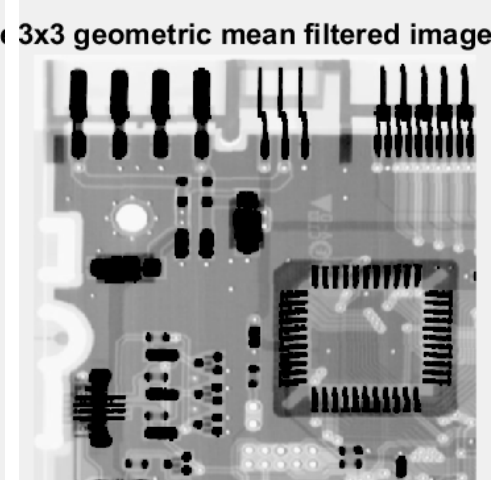
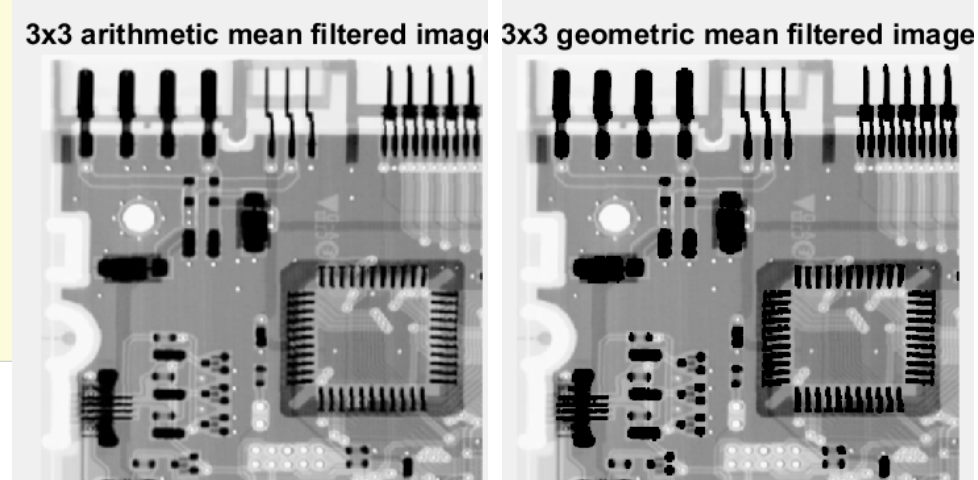
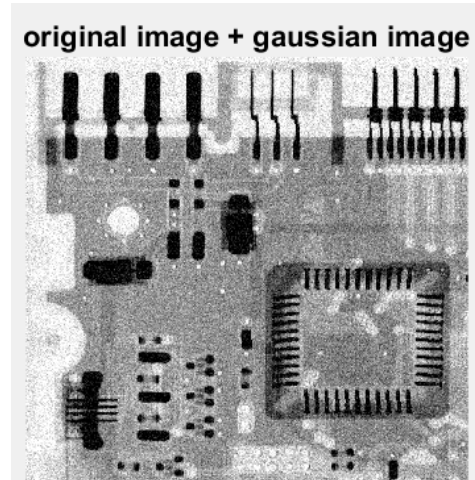
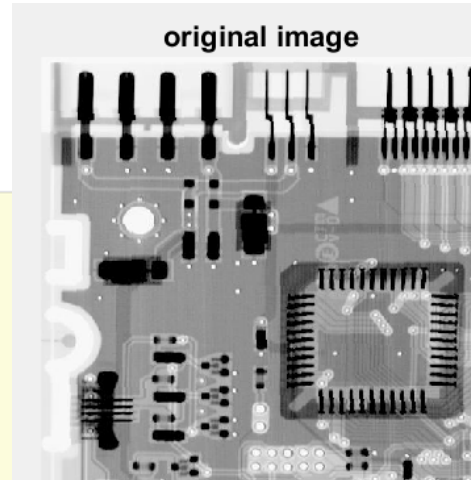
If Q is positive with 1



1	200	200
200	199.87	200
200	200	200

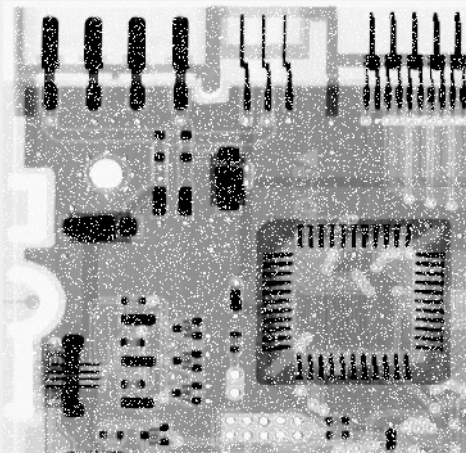
Example: Mean Filters

```
clc;
close;
clear all;
I = im2double(imread('Fig0507(a) (ckt-board-orig).tif'));
[m, n] = size(I);
%% Mean Filters (page 32 in lecture note 5)
% adding Gaussian noise, you can change both parameters as you like.
mean = 0.0;
std = 0.1;
gaussian_noise = mean + std*randn(m,n);
J1 = I + gaussian_noise;
% using 3x3 arithmetic mean filter
h = 1/ (3. ^2)*ones (3);
J2 = imfilter(I, h, 'conv', 'replicate', 'same');
% using 3x3 geometric mean filter
tmp_J3 = imfilter(log(I), h, 'conv', 'replicate', 'same');
J3 = exp(tmp_J3);
% plot
figure;
subplot(221);
imshow(I)
title('original image')
subplot(222);
imshow(J1);
title('original image + gaussian image')
subplot(223);
imshow(J2);
title('3x3 arithmetic mean filtered image')
subplot(224);
imshow(J3);
title('3x3 geometric mean filtered image')
```

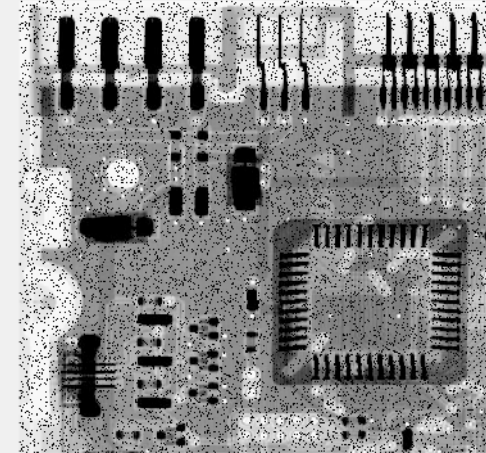


Example: Mean Filters (Cont'd)

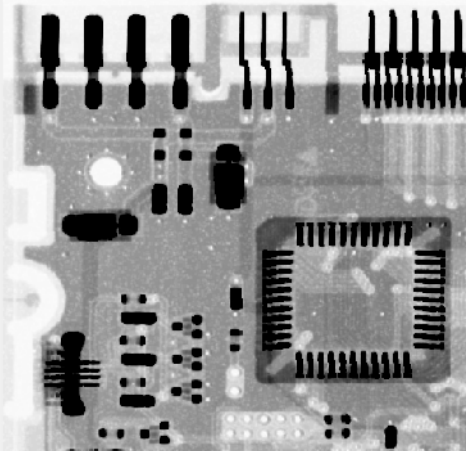
original image + salt noise (J1)



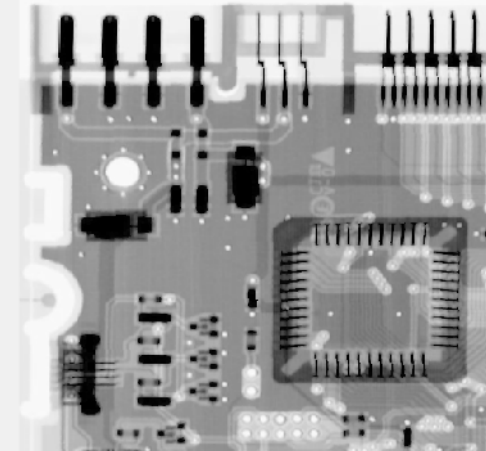
original image + pepper image (J2)



3x3 contraharmonic mean filtered J1 with $Q=-1.5$ (Correct)

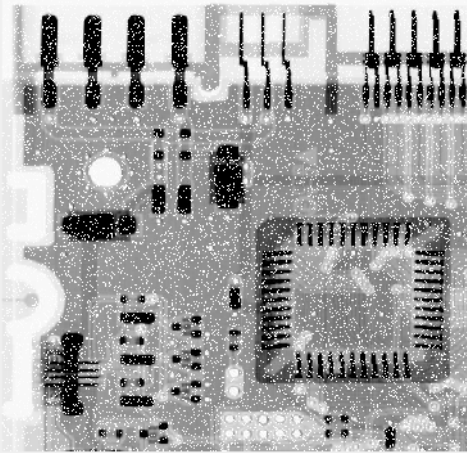


3x3 contraharmonic mean filtered J2 with $Q=1.5$ (Correct)

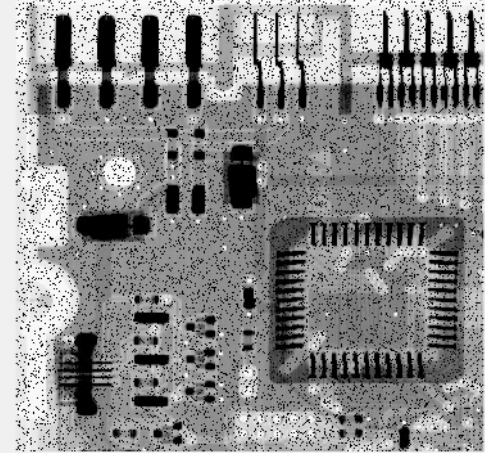


Example: Mean Filters (Cont'd)

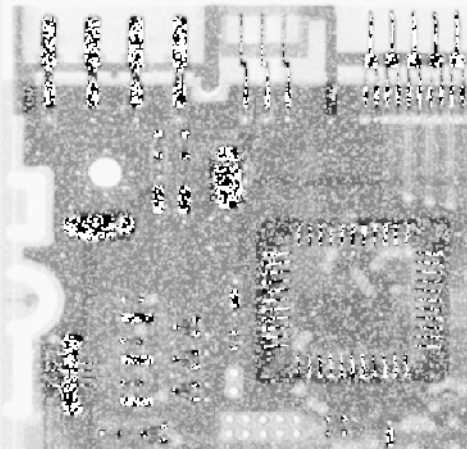
original image + salt noise (J1)



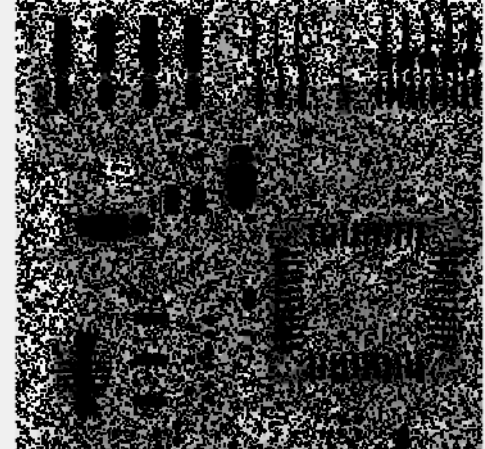
original image + pepper image (J2)



3x3 contraharmonic mean filtered J1 with $Q=1.5$ (Wrong)



3x3 contraharmonic mean filtered J2 with $Q=-1.5$ (Wrong)



```

%% Mean filters Cont'd do (page 33-34| in lecture note 5)
% more details about adding salt & pepper noise are displayed in AddNoise.m
% just add salt noise (light points) with probability 0.1
tmp_mask = rand(m,n);
salt_idx = find(tmp_mask <= 0.1);
J1 = I;
J1(salt_idx) = 1;
% just add pepper noise (dark points) with probability 0.1
tmp_mask = rand(m,n);
pepper_idx = find(tmp_mask <= 0.1);
J2 = I;
J2(pepper_idx) = 0;
% using 3x3 Contraharmonic Filter with parameter Q = 1.5 to process J1
J3 = Contraharmonic_Filter(J1, [3,3], 1.5);
% using 3x3 Contraharmonic Filter with parameter Q = -1.5 to process J2
J4 = Contraharmonic_Filter(J2, [3,3], -1.5);
% using 3x3 Contraharmonic Filter with parameter Q = -1.5 to process J1
J5 = Contraharmonic_Filter(J1, [3,3], -1.5);
% using 3x3 Contraharmonic Filter with parameter Q = 1.5 to process J2
J6 = Contraharmonic_Filter(J2, [3,3], 1.5);
figure;
subplot(321);
imshow(J1);
title('original image + salt noise (J1)')
subplot(322);
imshow(J2);
title('original image + pepper image (J2)')
subplot(323);
imshow(J3);
title('3x3 contraharmonic mean filtered J1 with Q=1.5 (Wrong)')
subplot(324);
imshow(J4);
title('3x3 contraharmonic mean filtered J2 with Q=-1.5 (Wrong)')
subplot(325);
imshow(J5);
title('3x3 contraharmonic mean filtered J1 with Q=-1.5 (Correct)')
subplot(326);
imshow(J6);
title('3x3 contraharmonic mean filtered J2 with Q=1.5 (Correct)')

```

Noise Removal with Spatial Filters

- Common filters to remove noise
 - Mean filters
 - Order-statistics filters
 - Adaptive mean filters

Order-Statistics Filters

- **Order Filter** – based on a specific type of image statistics called order statistics, sometimes known as **Order-Statistics Filter**
- Order Statistics: **Arranges all pixels in sequential order** (smallest to largest), based on gray-level value
- The selection of the value to replace the center pixel, is determined by the statistical function used
 - Min and max
 - Median filter
 - Midpoint filter
 - Alpha-trimmed mean filter

Order-Statistics Filters

- **Minimum Filter** – select the smallest value within an ordered window of pixel values, denoted as

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

1	1	1
1	1	1
1	1	200

- Works best when the noise is primarily of **the salt-type** (high value)
- **Maximum Filter** – select the largest value within an ordered window of pixel values, denoted as

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

1	200	200
200	200	200
200	200	200

- Works best for **pepper-type noise** (low value)

Examples: Min & Max Filters

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

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

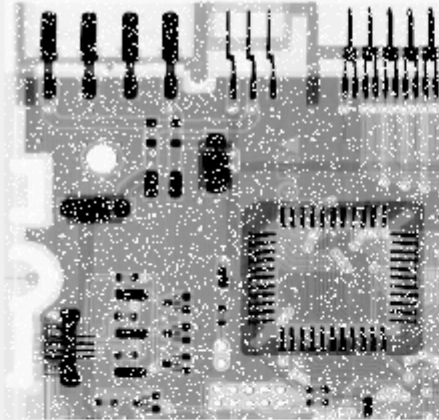
Min filter

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

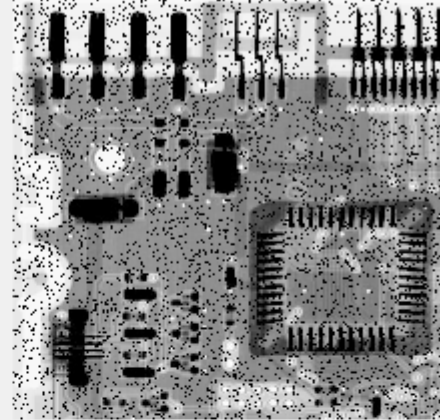
max filter

Example: Min & Max Filters

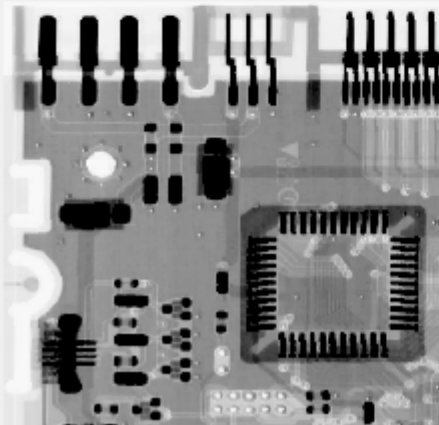
original image + salt noise (J1)



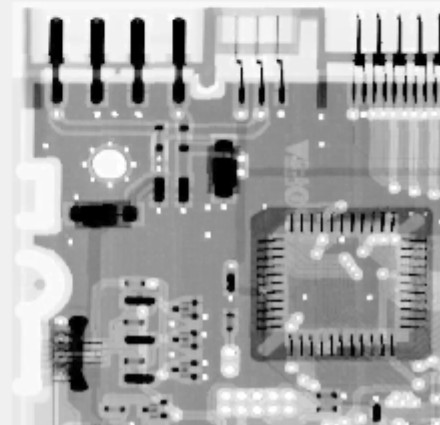
original image + pepper image (J2)



3x3 Min filtered J1



3x3 Max filtered J2



```

%% Min and Max filters (page 40-41 in lecture note 5)
% just add salt noise (light points) with probability 0.1
tmp_mask = rand(m,n);
salt_idx = find(tmp_mask <= 0.1);
J1 = I;
J1(salt_idx) = 1;
% just add pepper noise (dark points) with probability 0.1
tmp_mask = rand(m,n);
pepper_idx = find(tmp_mask <= 0.1);
J2 = I;
J2(pepper_idx) = 0;
% using 3x3 Min filter to process J1
J3 = ordfilt2(J1,1,ones(3,3), 'symmetric');
% using 3x3 Max filter to process J2
J4 = ordfilt2(J2,9,ones(3,3), 'symmetric');

figure;
subplot(221);imshow(J1);title('original image + salt noise (J1)')
subplot(222);imshow(J2);title('original image + pepper image (J2)')
subplot(223);imshow(J3);title('3x3 Min filtered J1')
subplot(224);imshow(J4);title('3x3 Max filtered J2')

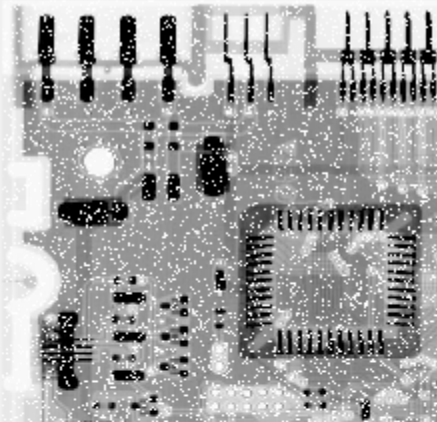
J2 = I;
J2(pepper_idx) = 0;
% using 3x3 Min filter to process J2
J4 = ordfilt2(J2,1,ones(3,3), 'symmetric');
% using 3x3 Max filter to process J1
J3 = ordfilt2(J1,9,ones(3,3), 'symmetric');

figure;
subplot(221);imshow(J1);title('original image + salt noise (J1)')
subplot(222);imshow(J2);title('original image + pepper image (J2)')
subplot(223);imshow(J3);title('3x3 Max filtered J1')
subplot(224);imshow(J4);title('3x3 Min filtered J2')

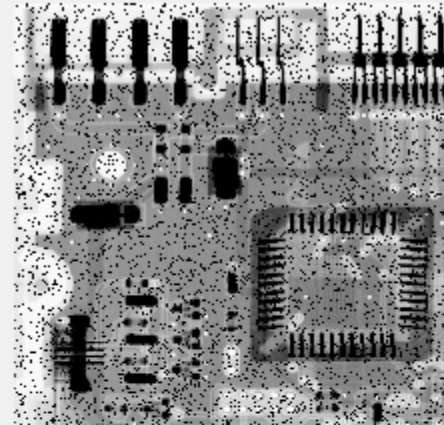
```

Example: Min & Max Filters

original image + salt noise (J1)



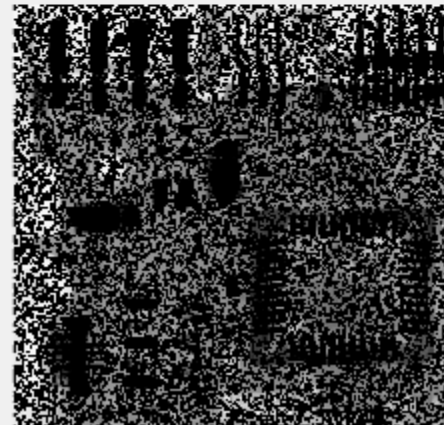
original image + pepper image (J2)



3x3 Max filtered J1



3x3 Min filtered J2



Order-Statistics Filters: Median Filters

- **Median Filter** – select the middle pixel value within an ordered window of pixel values, denoted as

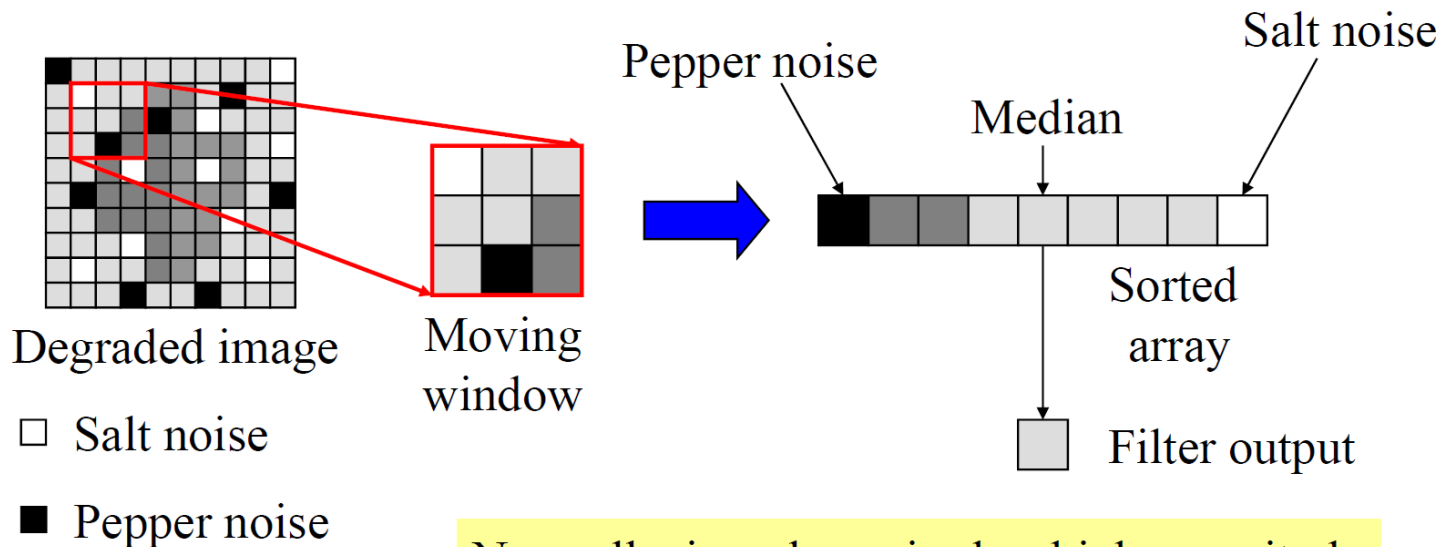
$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\text{median}}\{g(s, t)\}$$

1	100	100
100	100	100
100	100	200

- **Works best with salt-and-pepper noise** (both high and low values)
- **Better noise remover than Averaging Filter**, which causes blurry edges and details in image, thus not effective against impulse (salt-and-pepper) noise
- Preserve line structures

Order-Statistics Filters: Median Filters

- Working strategy:



Normally, impulse noise has high magnitude and is isolated. When we sort pixels in the moving window, noise pixels are usually at the ends of the array.

- Therefore, it's rare that the noise pixel will be a median value.

Examples: Median Filters (1)

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

Original image

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

Median filter

Examples: Median Filters (2)

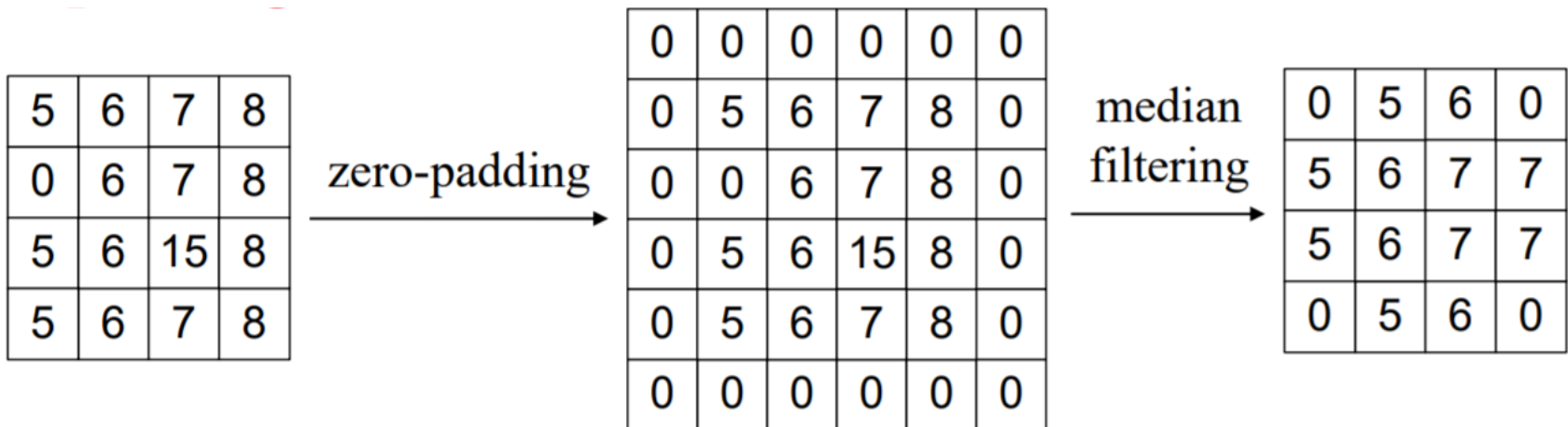
- A 4x4 grayscale image is given by

5	6	7	8
0	6	7	8
5	6	15	8
5	6	7	8

impulse? (pointing to 0)

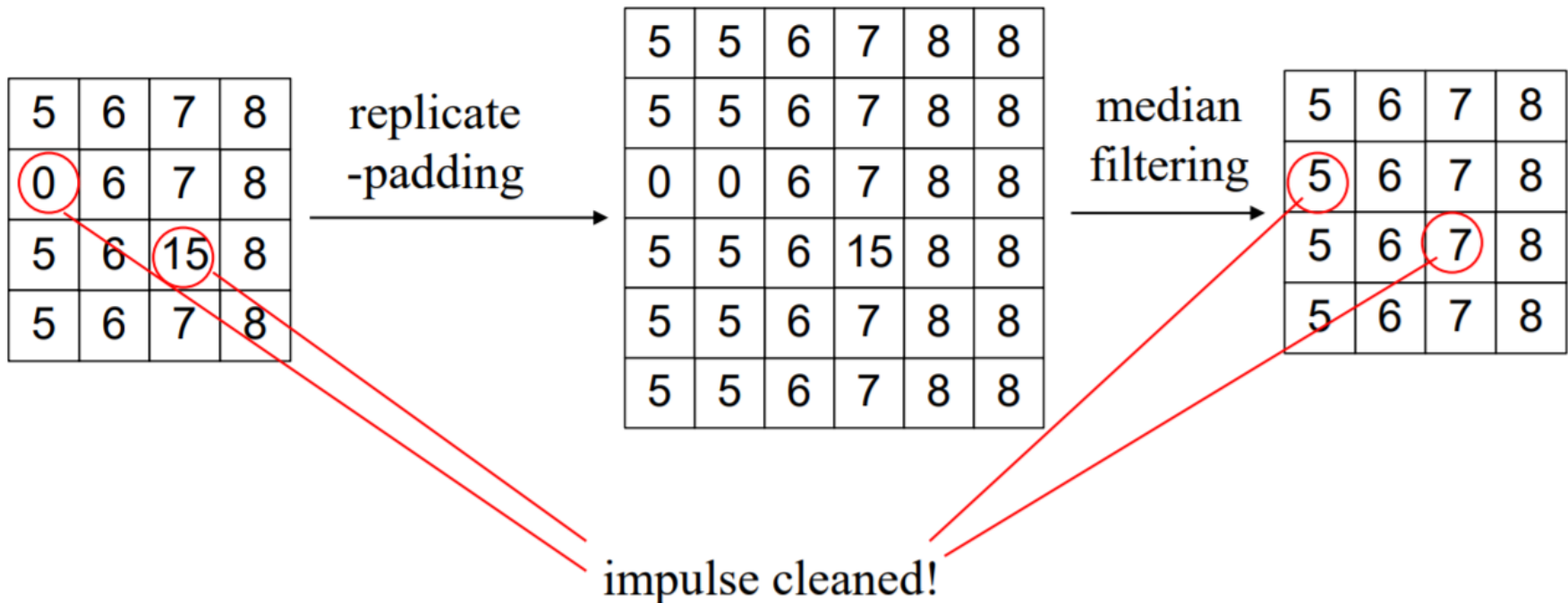
impulse? (pointing to 15)

- Filter the image with a 3x3 median filter, after **zero-padding**



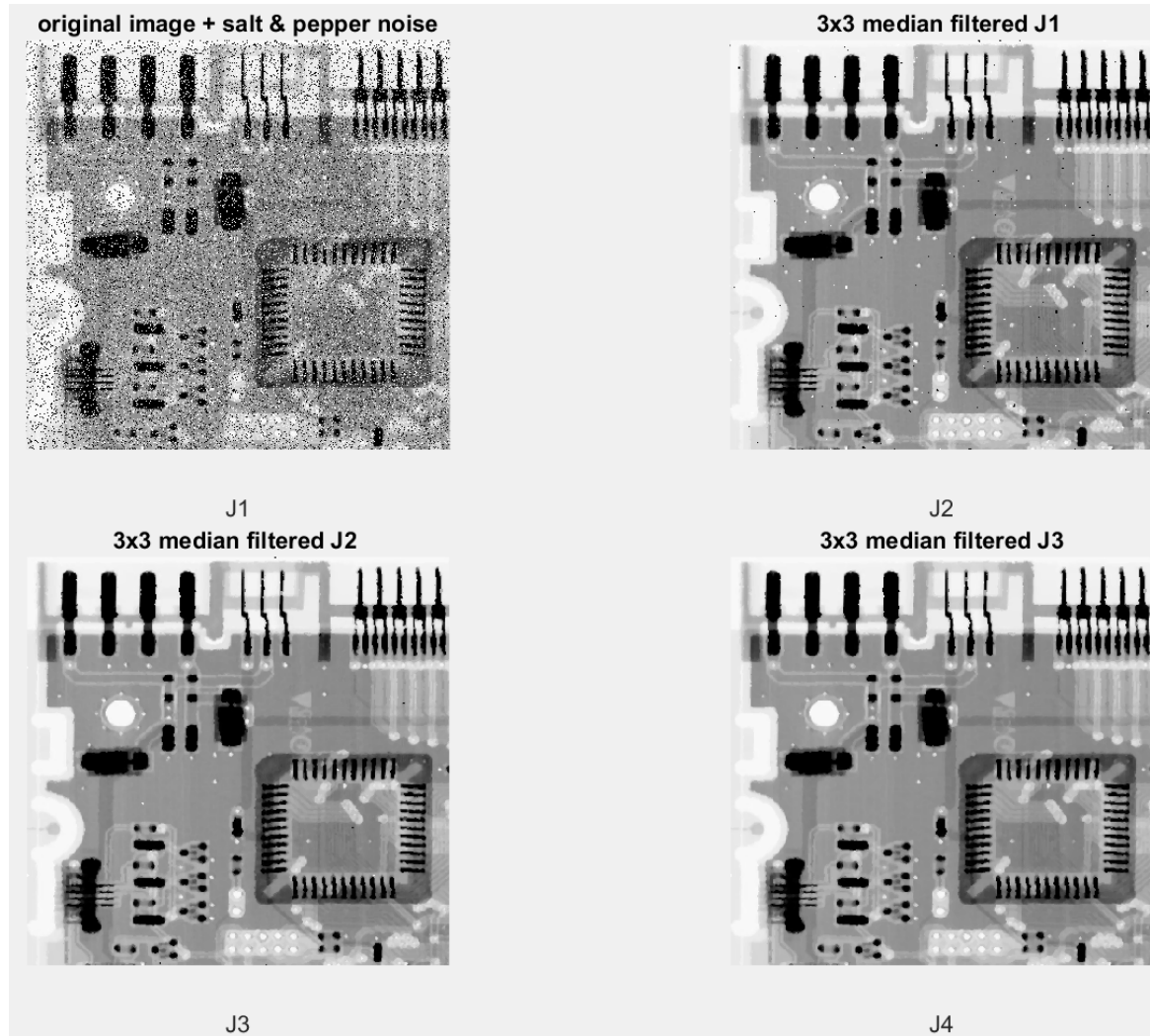
Examples: Median Filters (2)

- 2) Filter the image with a 3x3 median filter, after **replicate-padding** at the image borders



Example: Median Filter

- Can be worked one median filter following by another one



```

%% Median filters (page 48 in lecture note 5)
% adding salt & pepper noise with the same probability 0.1
% Why 0.2? because 0.1+0.1=0.2
J1 = imnoise(I, 'salt & pepper', 0.2);
% using 3x3 median filter to process J1
J2 = ordfilt2(J1,5,ones(3,3), 'symmetric');
% using 3x3 median filter to process J2
J3 = ordfilt2(J2,5,ones(3,3), 'symmetric');
% using 3x3 median filter to process J3
J4 = ordfilt2(J3,5,ones(3,3), 'symmetric');
figure;
subplot(221);
imshow(J1)
title('original image + salt & pepper noise')
xlabel('J1');
subplot(222);
imshow(J2);
xlabel('J2');
title('3x3 median filtered J1 ')
subplot(223);
imshow(J3);
xlabel('J3');
title('3x3 median filtered J2 ')
subplot(224);
imshow(J4);
xlabel('J4');
title('3x3 median filtered J3')

```

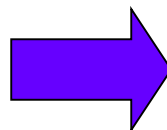
Order-Statistics Filters: Midpoint Filter

- Midpoint Filter – select the average of the maximum and minimum pixel values within the window, denoted as

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

- Useful for Gaussian and uniform noise
- Suitable for salt & pepper noise??

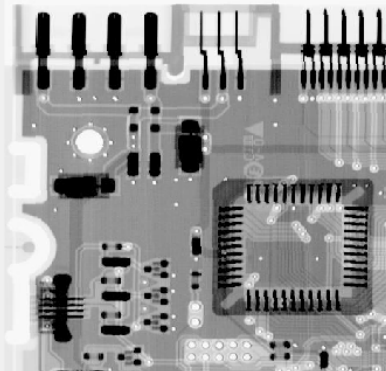
1	2	1	4	3
1	2	2	3	4
5	7	6	8	9
5	7	6	8	8
5	6	7	8	9



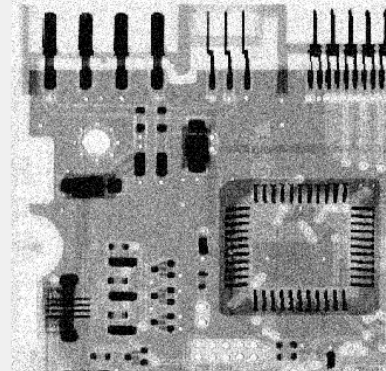
1	2	1	4	3
1	4	5	5	4
5	4	5	6	9
5	6	7	8	8
5	6	7	8	9

Example: Midpoint Filter

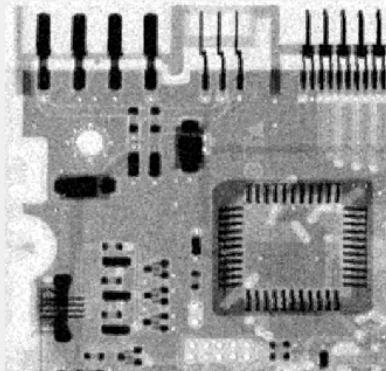
Original image



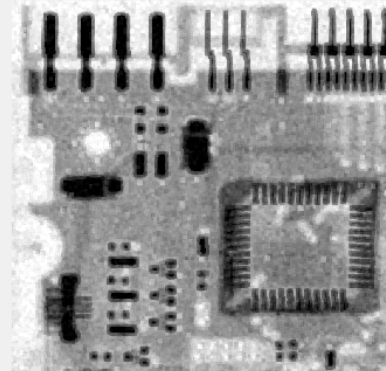
Gaussian Noise image



Gaussian Noise with midpoint filter (3*3)

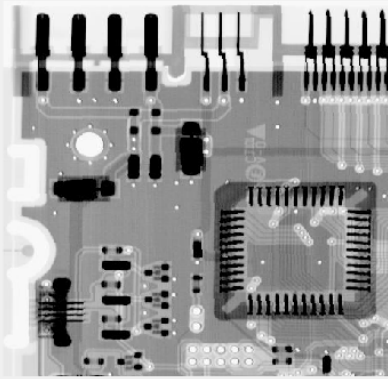


Gaussian Noise with midpoint filter (5*5)

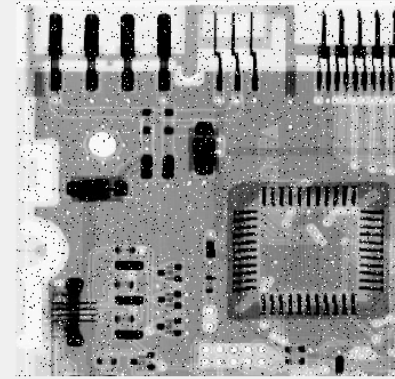


Example: Midpoint Filter

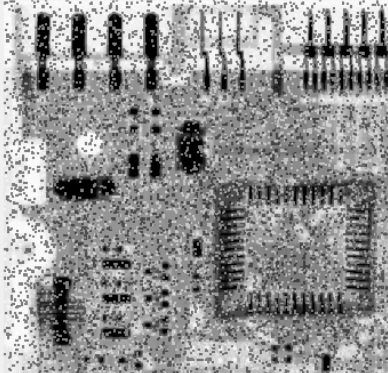
Original image



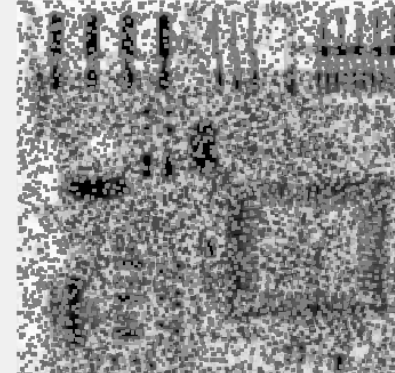
Noise image



salt & pepper Noise with midpoint filter (3*3)



salt & pepper Noise with midpoint filter (5*5)



```
%% midpoint filter (page 51-52 in lecture note 5)
```

```
J1 = imnoise(I, 'salt & pepper', 0.05);
```

```
figure;
```

```
subplot(221);imshow(I);title('Original image');
```

```
subplot(222);imshow(J1); title('salt & pepper Noise image');
```

```
J2 = (ordfilt2(J1,1,ones(3,3), 'symmetric')+ordfilt2(J1,9,ones(3,3), 'symmetric'))/2;
```

```
subplot(223);imshow(J2); title('salt & pepper Noise with midpoint filter (3*3)');
```

```
J3 = (ordfilt2(J1,1,ones(5,5), 'symmetric')+ordfilt2(J1,25,ones(5,5), 'symmetric'))/2;
```

```
subplot(224);imshow(J3); title('salt & pepper Noise with midpoint filter (5*5)');
```

```
mean = 0.0;
```

```
std = 0.1;
```

```
gaussian_noise = mean + std*randn(m,n);
```

```
J1 = I + gaussian_noise;
```

```
figure;
```

```
subplot(221);imshow(I);title('Original image');
```

```
subplot(222);imshow(J1); title('Gaussian Noise image');
```

```
J2 = (ordfilt2(J1,1,ones(3,3), 'symmetric')+ordfilt2(J1,9,ones(3,3), 'symmetric'))/2;
```

```
subplot(223);imshow(J2); title('Gaussian Noise with midpoint filter (3*3)');
```

```
J3 = (ordfilt2(J1,1,ones(5,5), 'symmetric')+ordfilt2(J1,25,ones(5,5), 'symmetric'))/2;
```

```
subplot(224);imshow(J3); title('Gaussian Noise with midpoint filter (5*5)');
```

Order-Statistics Filters: Alpha-trimmed Mean

- **Alpha-trimmed Mean Filter** - select the average of the values within the window, but with some of the endpoint-ranked values excluded
- Suppose we delete $d/2$ lowest and $d/2$ highest gray level values of $g(s,t)$ in the neighborhood S_{xy} . Let $g_r(s,t)$ represent the remaining **$mn-d$** pixels, the remaining pixels are averaged:

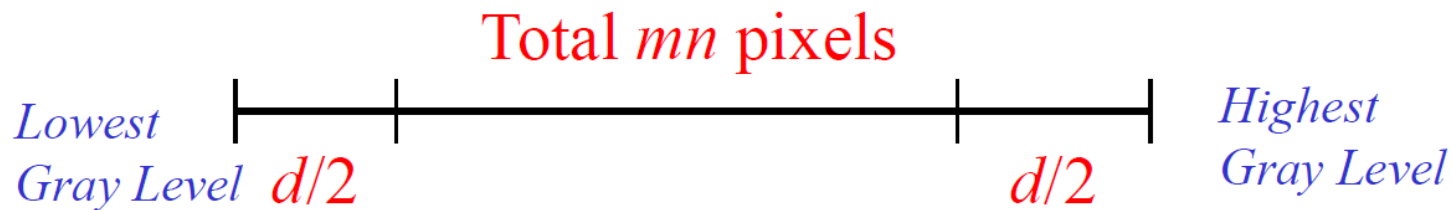
$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s, t) \in S_{XY}} g_r(s, t)$$

- Useful for combination noise such as salt-and-pepper with Gaussian noise

Order-Statistics Filters: Alpha-trimmed Mean

$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s, t) \in S_{XY}} g_r(s, t)$$

where $g_r(s, t)$ represent the remaining $mn-d$ pixels after removing the $d/2$ highest and $d/2$ lowest values of $g(s, t)$.



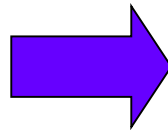
- This filter is useful in situations involving multiple types of noise such as a combination of salt-and-pepper and Gaussian noise.

Example: Alpha-trimmed Mean Filter

$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s, t) \in S_{XY}} g_r(s, t)$$

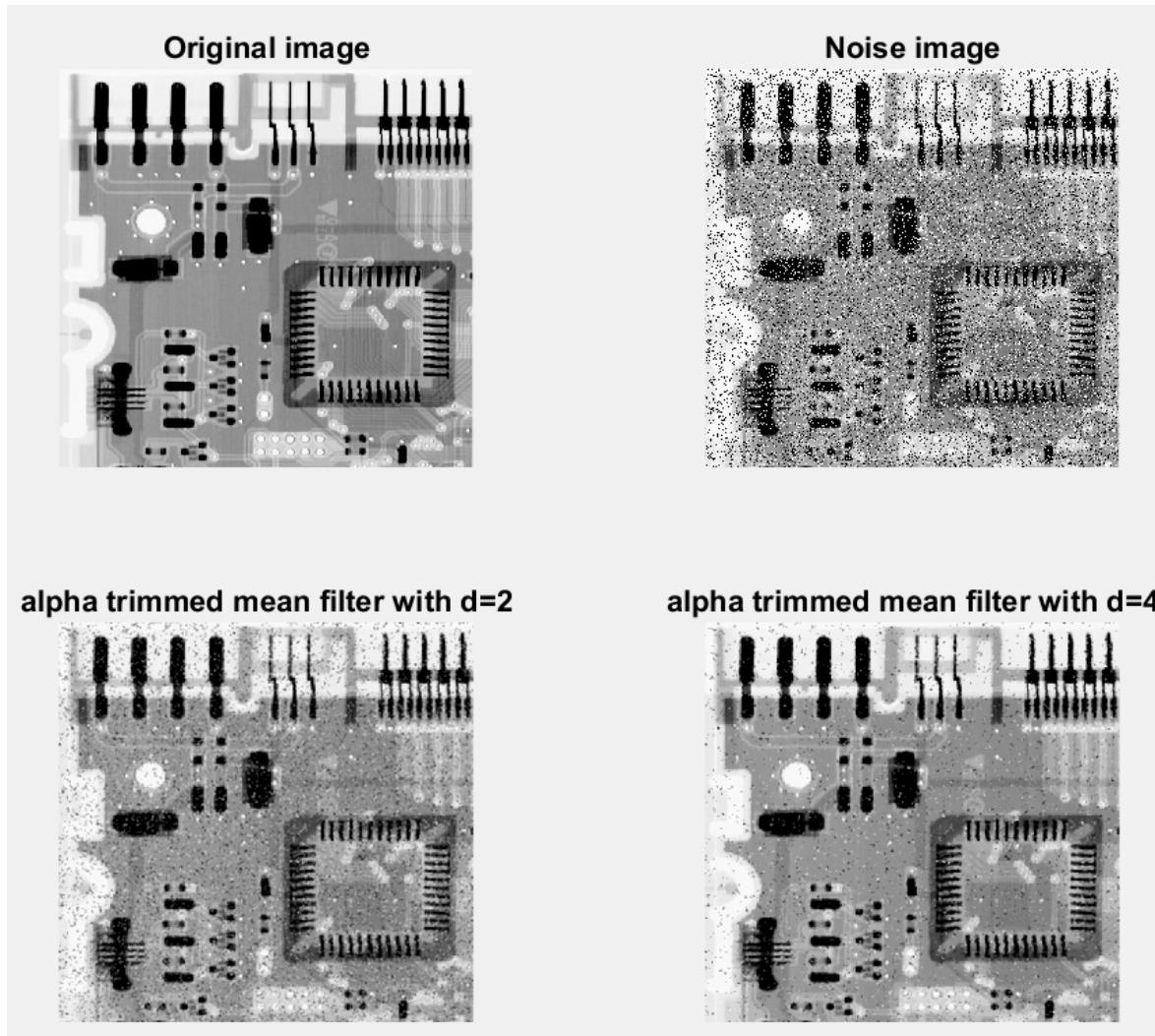
3 × 3 smoothing filters: d=4

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



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

Example: Alpha-trimmed Mean Filter



Noise Removal with Spatial Filters

- Common filters to remove noise
 - Mean filters
 - Order-statistics filters
 - Adaptive mean filters

Adaptive Filters

- The previous filters are applied to an entire image without any regard for how image characteristics vary from one point to another
- The behaviour of adaptive filters changes depending on the **characteristics of the image inside the filter region**

Adaptive Mean Filters

- Filters consider behavior changes based on statistical characteristics of the image inside the filter region
- Capable of superior performance, but causes increase in filter complexity
- Statistical characteristics:

Local mean:

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

Noise variance:

$$\sigma_{\eta}^2$$

Local variance:

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

Adaptive Mean Filters

We want the behavior of the filter to be as follows:

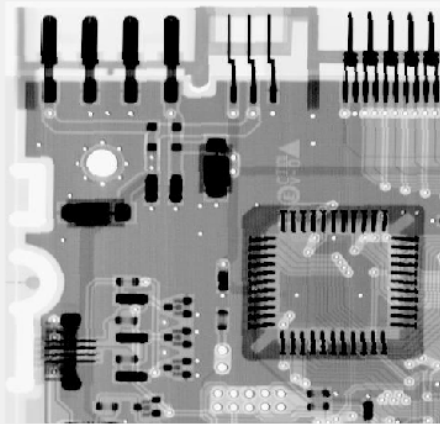
- If σ_{η}^2 is zero, the filter should return simply the value of $g(x, y)$. This is the trivial, zero-noise case in which $g(x, y)$ is equal to $f(x, y)$.
- If the local variance is high relative to σ_{η}^2 , the filter should return a value close to $g(x, y)$. A high local variance typically is associated with edges, and these should be preserved.
- If the two variances are equal, we want the filter to return the arithmetic mean value of the pixels in S_{xy} . This condition occurs when the local area has the same properties as the overall image, and local noise is to be reduced simply by averaging.

Adaptive Mean Filters

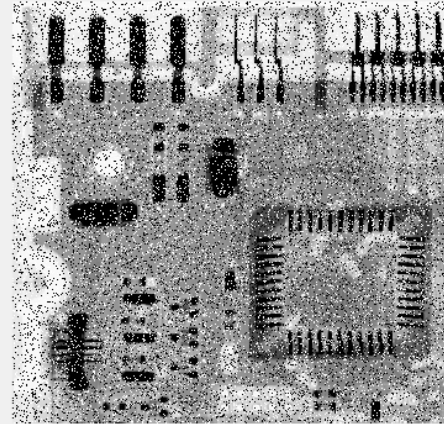
- Purpose: want to preserve edges
- Concept:
 1. If σ_{η}^2 is zero, $\rightarrow$ No noise
the filter should return $g(x,y)$ because $g(x,y) = f(x,y)$
 2. If σ_L^2 is high relative to σ_{η}^2 , $\rightarrow$ Edges (should be preserved),
the filter should return the value close to $g(x,y)$
 3. If $\sigma_L^2 = \sigma_{\eta}^2$, $\rightarrow$ Areas inside objects
the filter should return the arithmetic mean value m_L
- Formula:
$$\hat{f}(x, y) = g(x, y) - \frac{\sigma_{\eta}^2}{\sigma_L^2} [g(x, y) - m_L]$$

Examples: adaptive filters

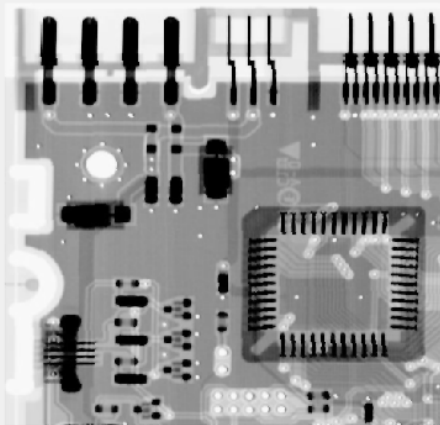
Original image



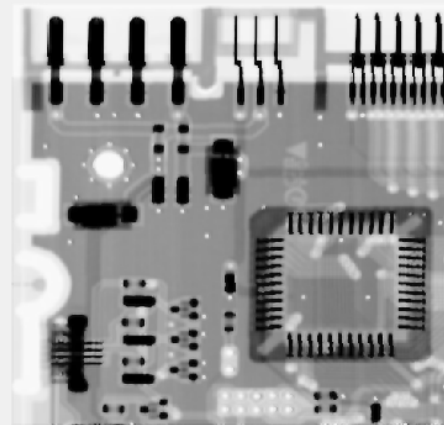
Noise image



Adaptive filter with 3*3



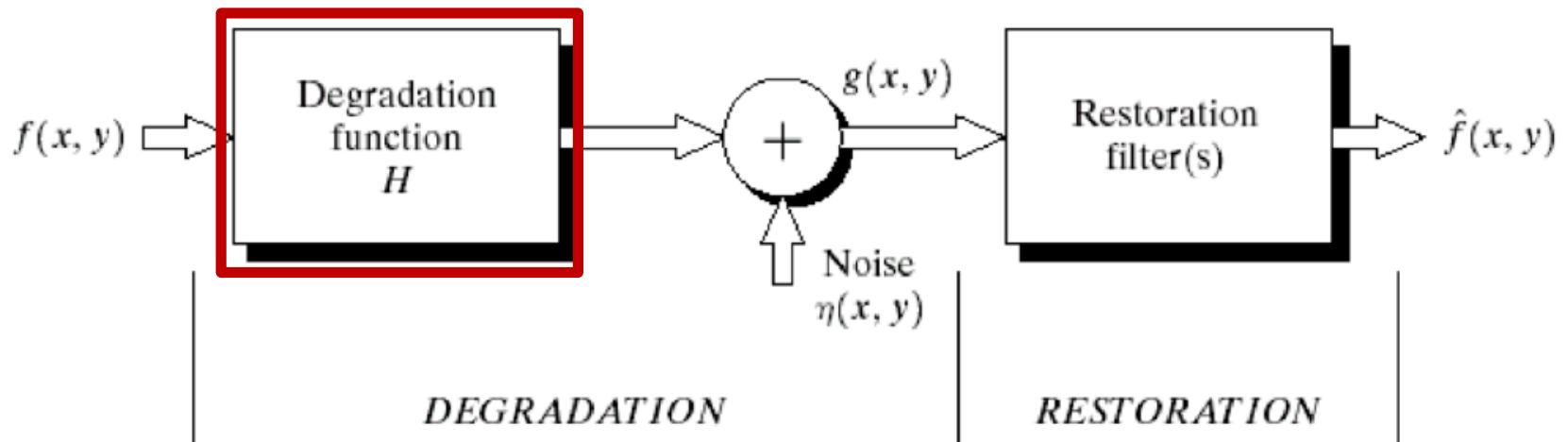
Adaptive filter with 7*7



Lecture outline

- Degradation & Restoration Process Models
- Noise Models
- Restoration in the presence of noise only (spatial filtering)
- Estimation of the degradation function
- Inverse filtering
- Wiener filtering

Estimation the degradation function



■ Mathematical model:

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

where:

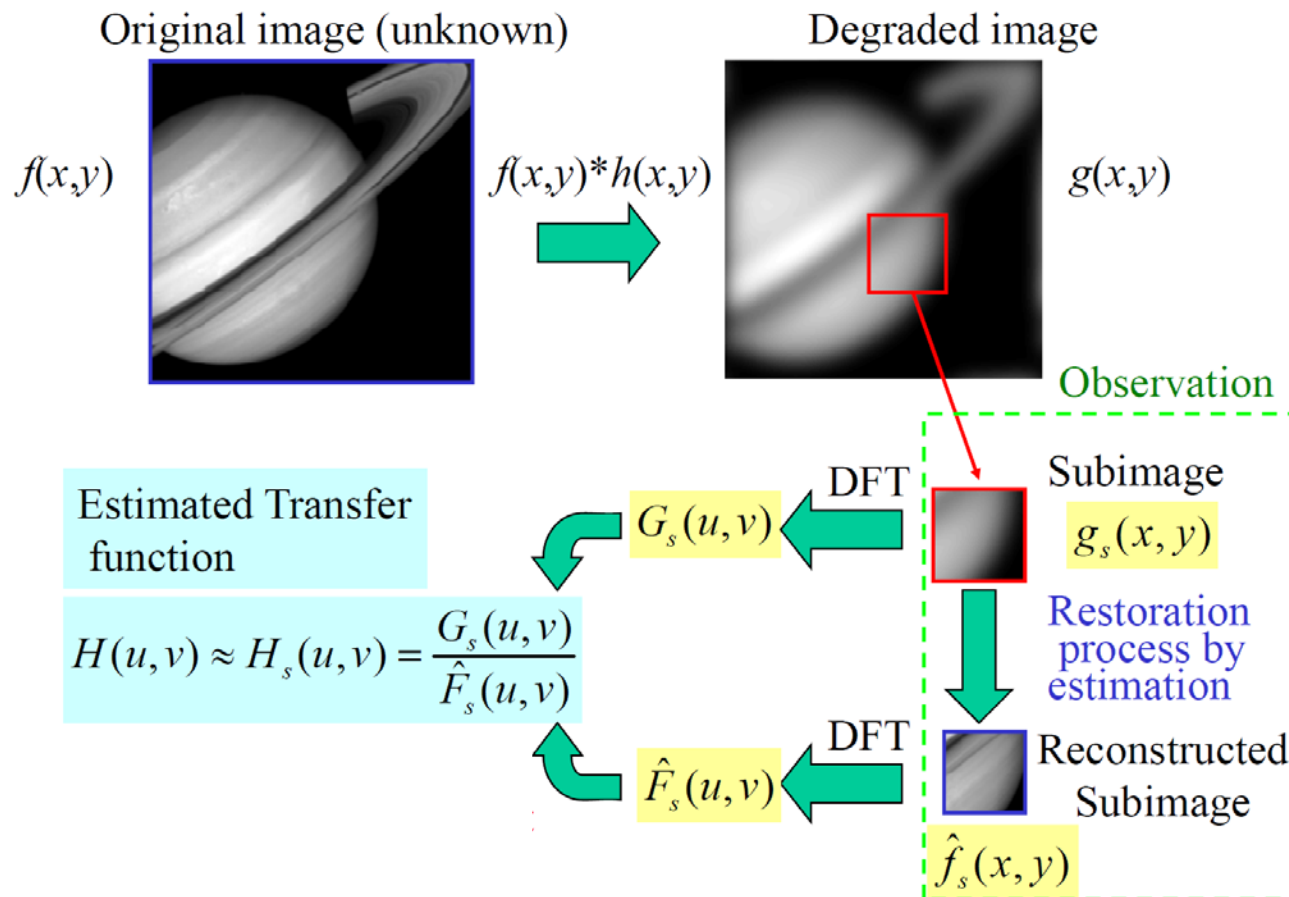
- $G(u, v)$ = Fourier transform of degraded image
- $H(u, v)$ = Fourier transform of degradation function
- $F(u, v)$ = Fourier transform of original image
- $N(u, v)$ = Fourier transform of additive noise function

Estimation the degradation function

- Question: How do we estimate the degradation function?
 - Image Observation
 - Experimentation
 - Mathematical Modeling

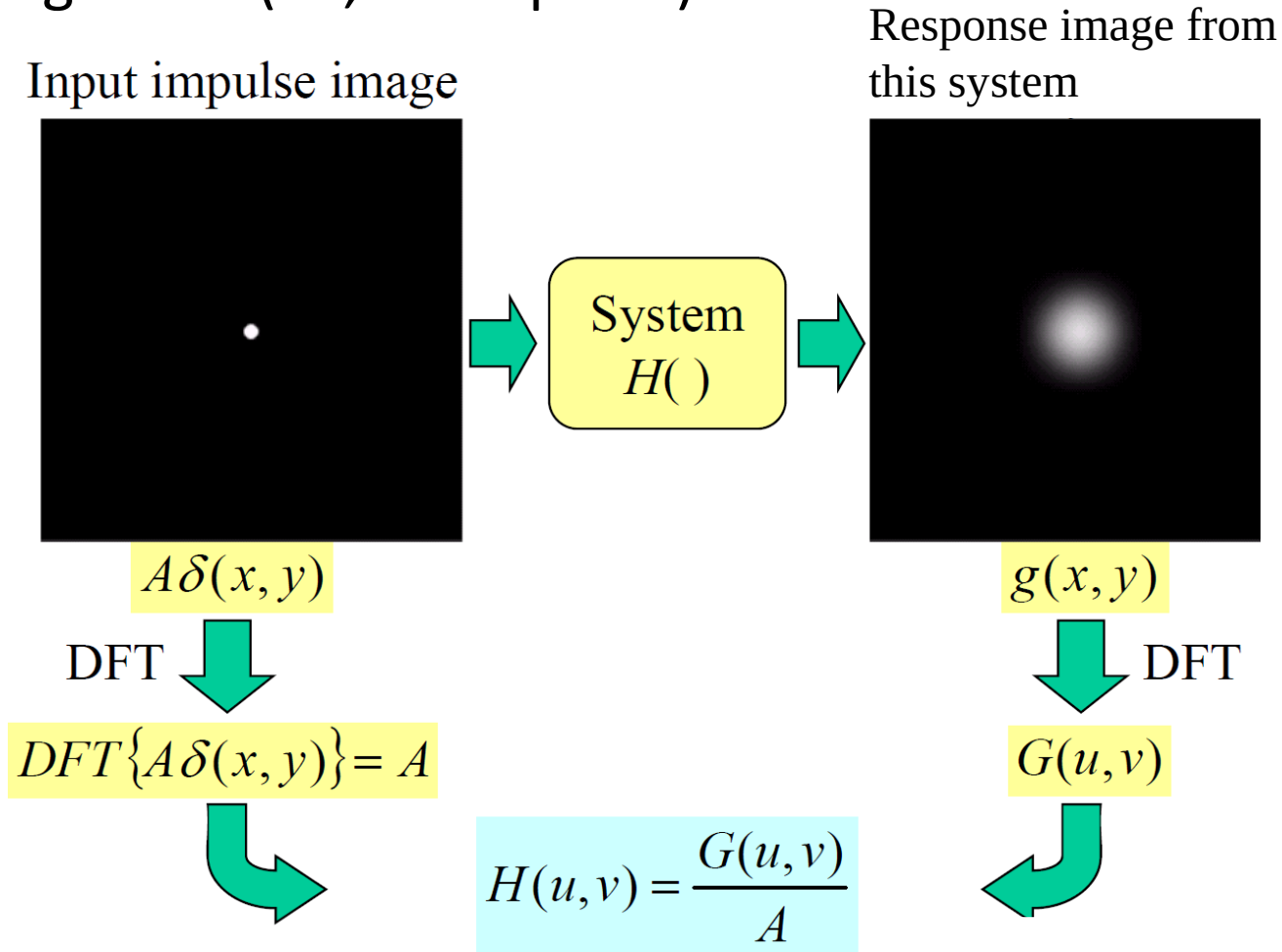
Estimation by Image Observation

- Take a small area of the image, $G_s(u,v)$, If we know what the non-degraded image $F_s(u,v)$.



Estimation by Experimentation

- Using the same imaging system, take another picture $G(u,v)$ of a bright dot (ie., an impulse)



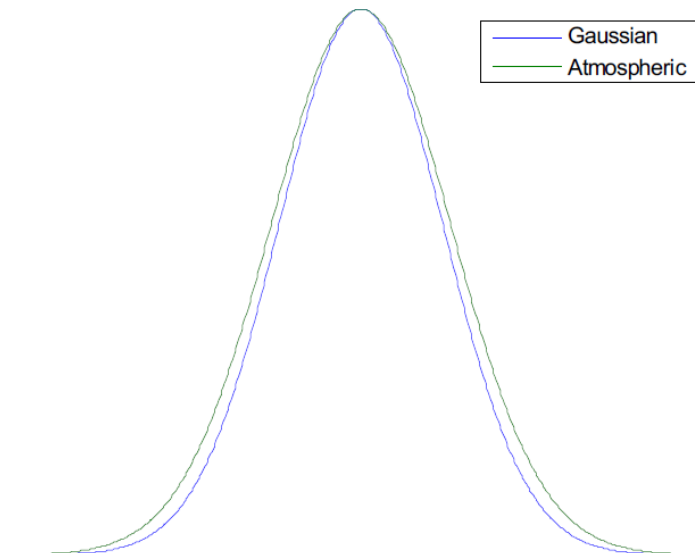
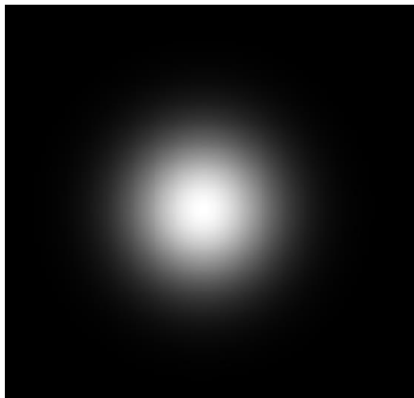
Estimate by Modeling

- Can analytically derive the form of the degradation
- Examples:
 - Atmospheric blur
 - Motion blur

Estimation by Modeling: Atmospheric Blur

- A degradation model proposed by Hufnagel and Stanley is based on the physical characteristics of atmospheric turbulence. This model has a familiar form:

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



Example Images for Atmospheric Blur

original image



Image with atmos blur $k=0.0025$



Image with atmos blur $k=0.001$

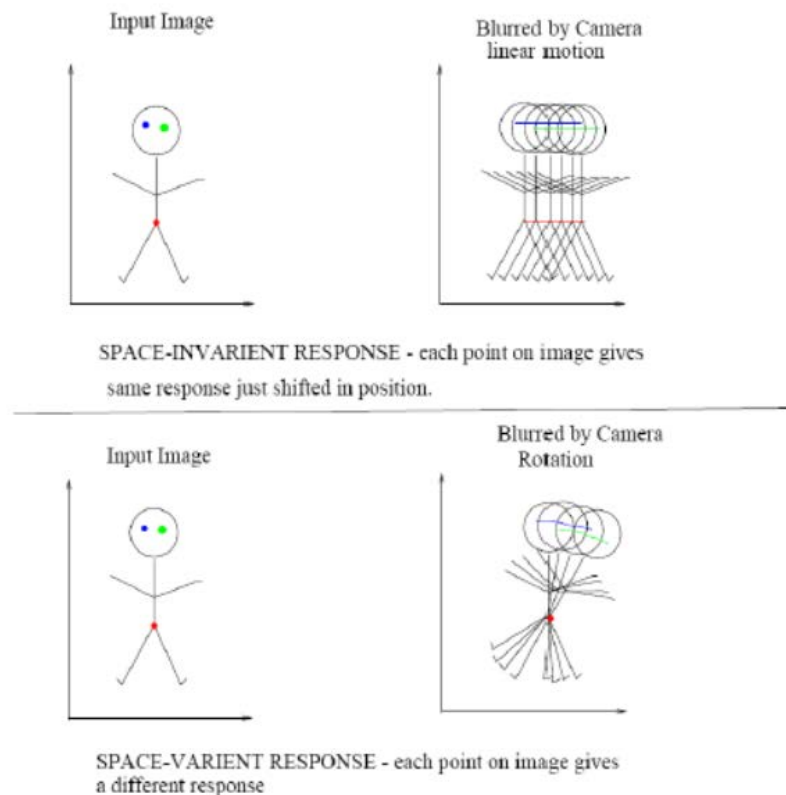


Image with atmos blur $k=0.00025$



Estimation by Modeling: Motion Blur

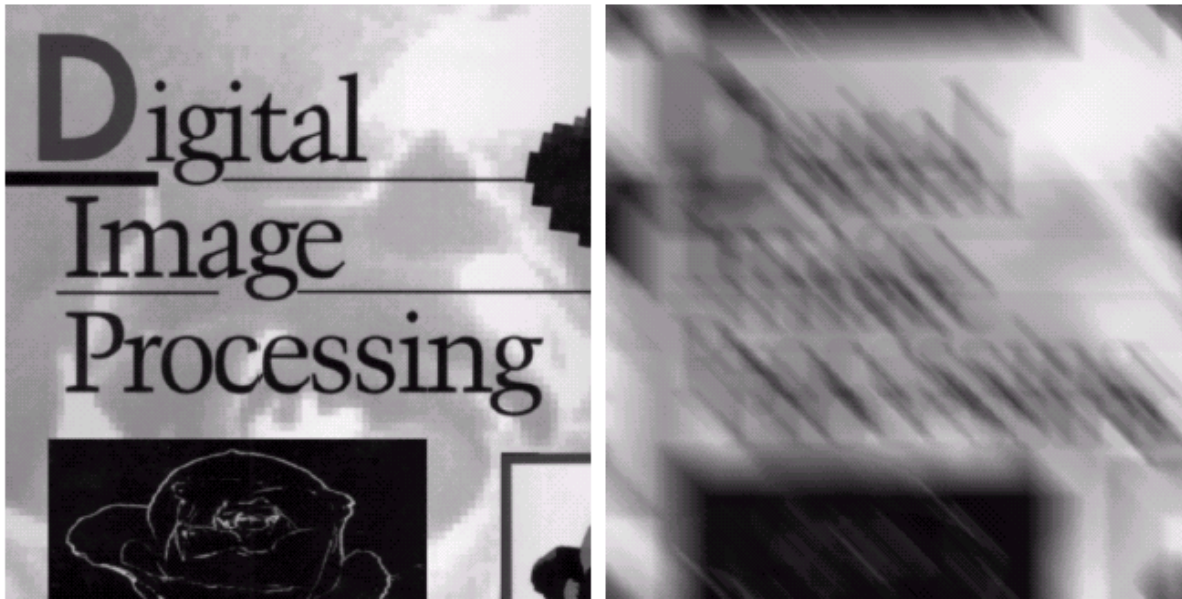
- An image is blurred by uniform linear motion between the image and the sensor during image acquisition.
- Taking images when your hand is not stable



Estimation by Modeling: Motion Blur Model

- Mathematical modeling of motion blur

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



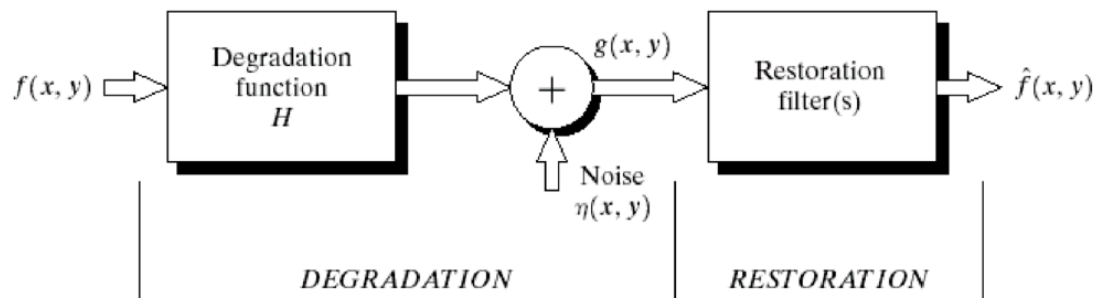
(a) Original image

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

Lecture outline

- Degradation & Restoration Process Models
- Noise Models
- Restoration in the presence of noise only (spatial filtering)
- Estimation of the degeneration function
- Inverse filtering
- Wiener filtering

Restoration by Inverse Filtering



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

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

- With assumption of **no noise**. Fourier transform of degraded image:

$$G(u, v) = H(u, v)F(u, v) + 0$$

- Fourier transform of the original image will be:

$$F(u, v) = \frac{G(u, v)}{H(u, v)} = G(u, v) \frac{1}{H(u, v)}$$

Restoration by Inverse Filtering

- Fourier transform of the original image will be:

$$F(u, v) = \frac{G(u, v)}{H(u, v)} = G(u, v) \frac{1}{H(u, v)}$$

- To find the original image, take the inverse Fourier transform of $F(u, v)$:

$$\begin{aligned} f(x, y) &= F^{-1}[F(u, v)] = F^{-1}\left[\frac{G(u, v)}{H(u, v)}\right] \\ &= F^{-1}\left[G(u, v) \frac{1}{H(u, v)}\right] \end{aligned}$$

Example

original image

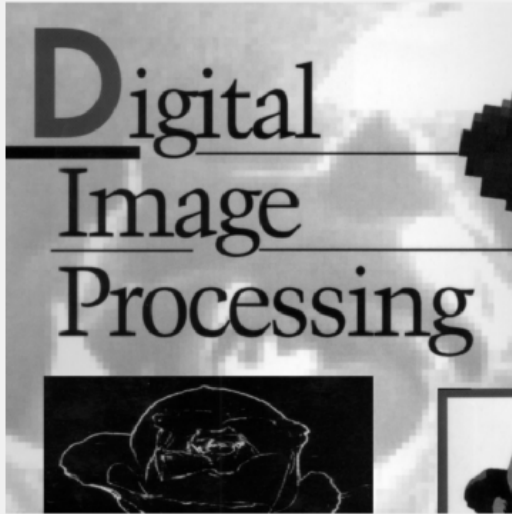
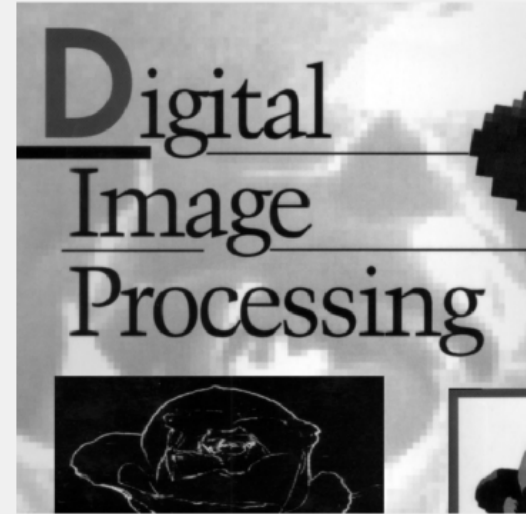


Image with motion blur, no noise



Image restoration with Inverse filter



Inverse Filtering – Effect of Noise

- In presence of noise:

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

- **Question:** If any points in $H(u, v)$ are zero – division by zero
- **Solution:** Do not take zero-points of $H(u, v)$ into account

Inverse Filtering – Effect of Noise

- In presence of noise:

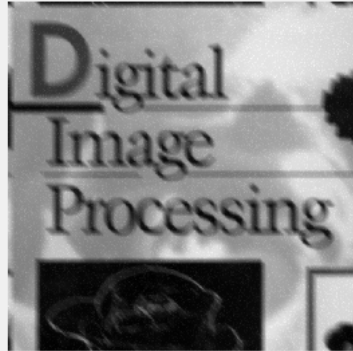
$$\hat{F}(u, v) = \frac{G(u, v)}{H(u, v)} = \frac{H(u, v)F(u, v)}{H(u, v)} + \frac{N(u, v)}{H(u, v)} = F(u, v) + \frac{N(u, v)}{H(u, v)}$$

- **Problem:** If $H(u, v)$ becomes very small, the second term becomes very large, and it overshadows the $F(u, v)$.
- **Solution:**
 - Limit the restoration to a specific radius about the origin in the spectrum – the restoration cutoff frequency
 - Only apply the inverse filter at the low frequencies

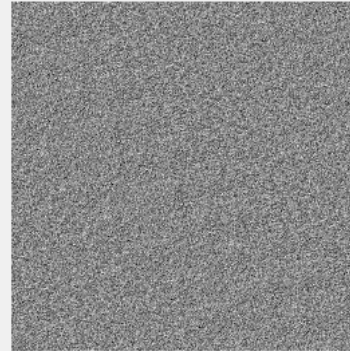
$$\hat{F}(u, v) = \begin{cases} \frac{G(u, v)}{H(u, v)} & \sqrt{u^2 + v^2} \leq r \\ G(u, v) & \sqrt{u^2 + v^2} > r \end{cases}$$

Inverse Filtering – Effect of Noise

Image with noise and blur



Restoration with Full Inverse filter



Restoration with Inverse filter ($r=20$)



Restoration with Inverse filter ($r=30$)



Lecture outline

- Degradation & Restoration Process Models
- Noise Models
- Restoration in the presence of noise only (spatial filtering)
- Estimation of the degeneration function
- Inverse filtering
- Wiener filtering

Wiener Filtering

- **Minimum Mean Square Error Filtering** that invented by N. Wiener(1942)
- Objective: Find an estimate of the uncorrupted image such that the mean square error between them is minimum

$$e^2 = E\left[\left(f - \hat{f}\right)^2\right] = \frac{1}{mn} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} \left(f(x, y) - \hat{f}(x, y)\right)^2$$

Wiener Filtering

- The minimum of the error function is given in the frequency domain by the expression

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

where $H^*(u, v)$ = complex conjugate of $H(u, v)$
 $S_\eta(u, v) = |N(u, v)|^2$ = power spectrum of the noise
 $S_f(u, v) = |F(u, v)|^2$ = power spectrum of the original image

Wiener Filtering

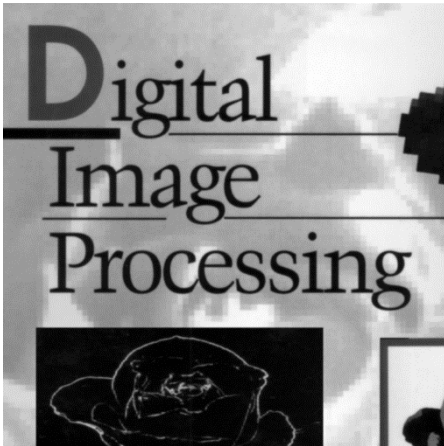
- Note that if the noise is zero, the Wiener filter reduces to the inverse filter.
- The noise power spectrum is constant when dealing with spectrally white noise.
- Approximation frequently used

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

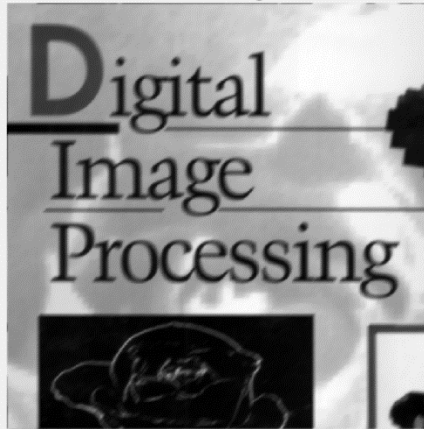
- K is adjusted interactively for best result.

Example: Wiener filtering

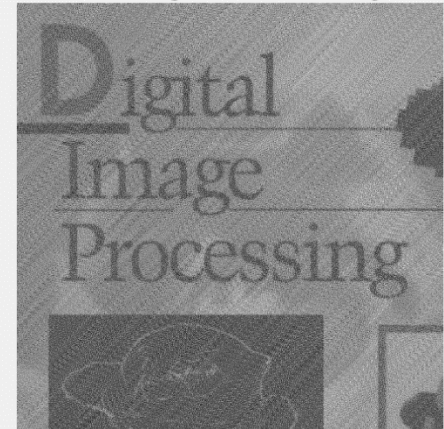
Original images



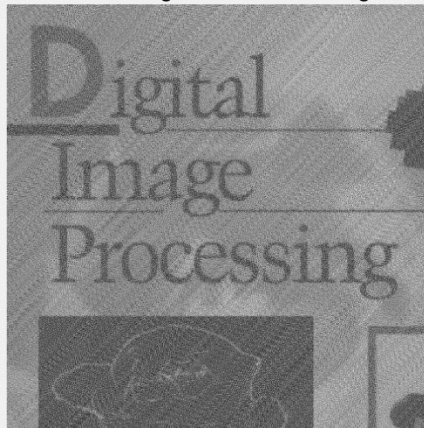
Noise image



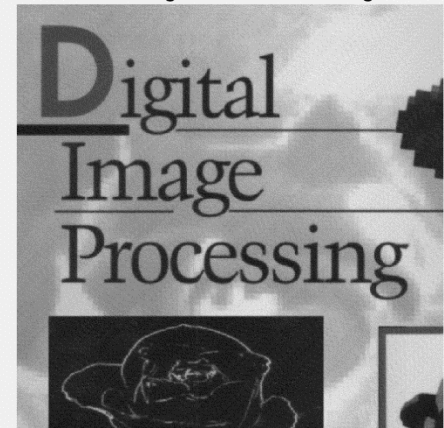
Noise image with inverse filtering



Noise image with Wiener filtering



Noise image with Wiener filtering2



Summary

- Degradation & Restoration Process Models (understanding)
- Noise Models (**6 types**)
- Restoration in the presence of noise only (spatial filtering)
 - **Mean filters:** Arithmetic mean, Geometric Mean, Harmonic Mean Filter, Contraharmonic Mean
 - **Order-statistics filters:** min, max, median, midpoint, alpha-trimmed mean
 - **Adaptive mean filters**
- Estimation of the degeneration function
 - Observation
 - Experimentation
 - Modeling (motion blur, atmospheric blur)
- Inverse filtering (challenges and solutions)
- Wiener Filtering (purpose, relationship with inverse filter, parameters)

Midterm

- Suggest to have two cameras
- It is Ok for you if you have only one camera
- I will publish the midterm in the Canvas
- During the midterm, please open your camera through ZOOM
- Time: 12:00-2:00 PM, Oct. 16
- Will give you extra 15 mins to upload your results through canvas
- No late submission is allowed