

Image Restoration

Dr. Mongkol Ekpanyapong

A decorative image in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

Model of Image Degradation

- Degradation and additive noise

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

- If H function is linear and spatial invariant

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

- Frequency domain

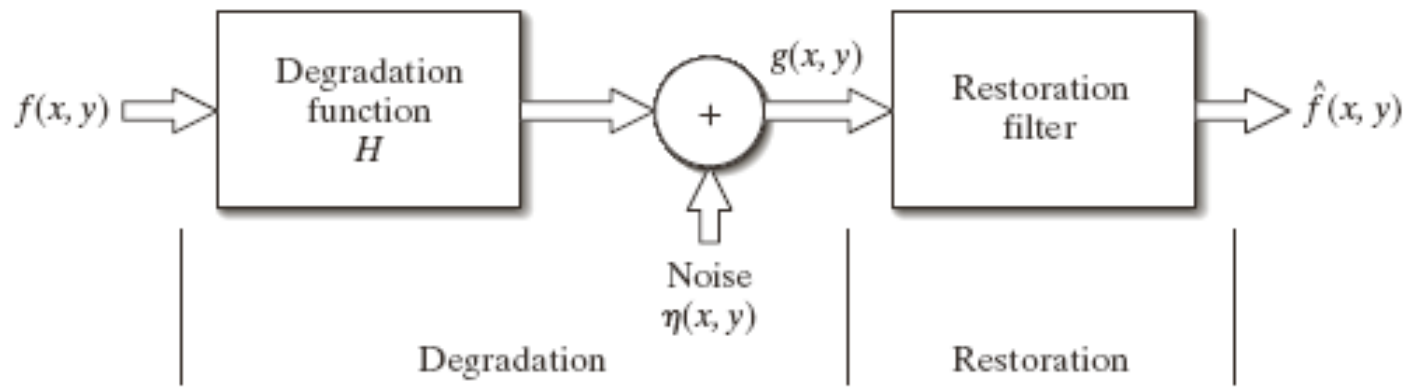
$$G(u, v) = H(u, v)F(u, v) + N(u, v)$$



The Model

FIGURE 5.1

A model of the
image degradation/
restoration process.



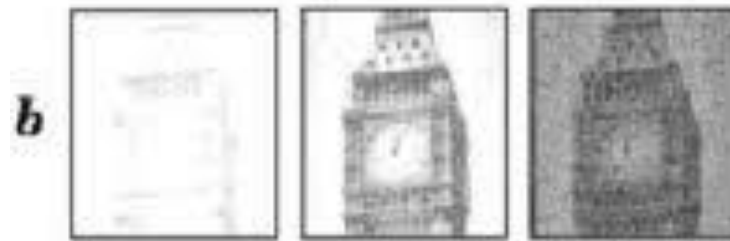
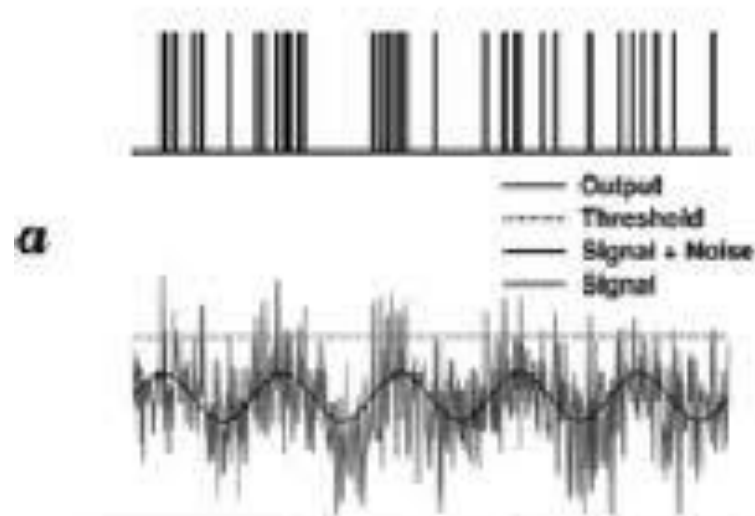
Limitation of Imaging Technology

- Two plagues in image acquisition
 - **Noise** interference
 - **Blur** (motion, out-of-focus, hazy weather)
- Difficult to obtain high-quality images as imaging goes
 - Beyond visible spectrum
 - Micro-scale (microscopic imaging)
 - Macro-scale (astronomical imaging)

What is Noise?

- Wiki definition: **noise** means any **unwanted** signal
- One person's signal is another one's noise
- Noise is not always random and **randomness** is an artificial term
- Noise is not always bad (see stochastic resonance example in the next slide)

Stochastic Resonance



no noise

light noise

heavy noise

A decorative image in the top-left corner showing a blue square above a circuit board.

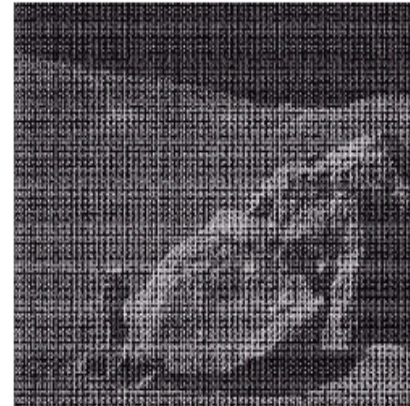
Image Denoising

- Where does noise come from?
 - Sensor (e.g., thermal or electrical interference)
 - Environmental conditions (rain, snow etc.)
- Why do we want to denoise?
 - Visually unpleasant
 - Bad for compression
 - Bad for analysis

Noisy Image Examples



thermal imaging



electrical interference



ultrasound imaging



physical interference

Noise Model

- MATLAB noise function
 $g = \text{imnoise}(f, \text{type}, \text{parameters})$
- Noise type can be:
 - Gaussian -Salt & pepper
 - Speckle (from radar or ultra sound)
 - Poisson (Exponential)

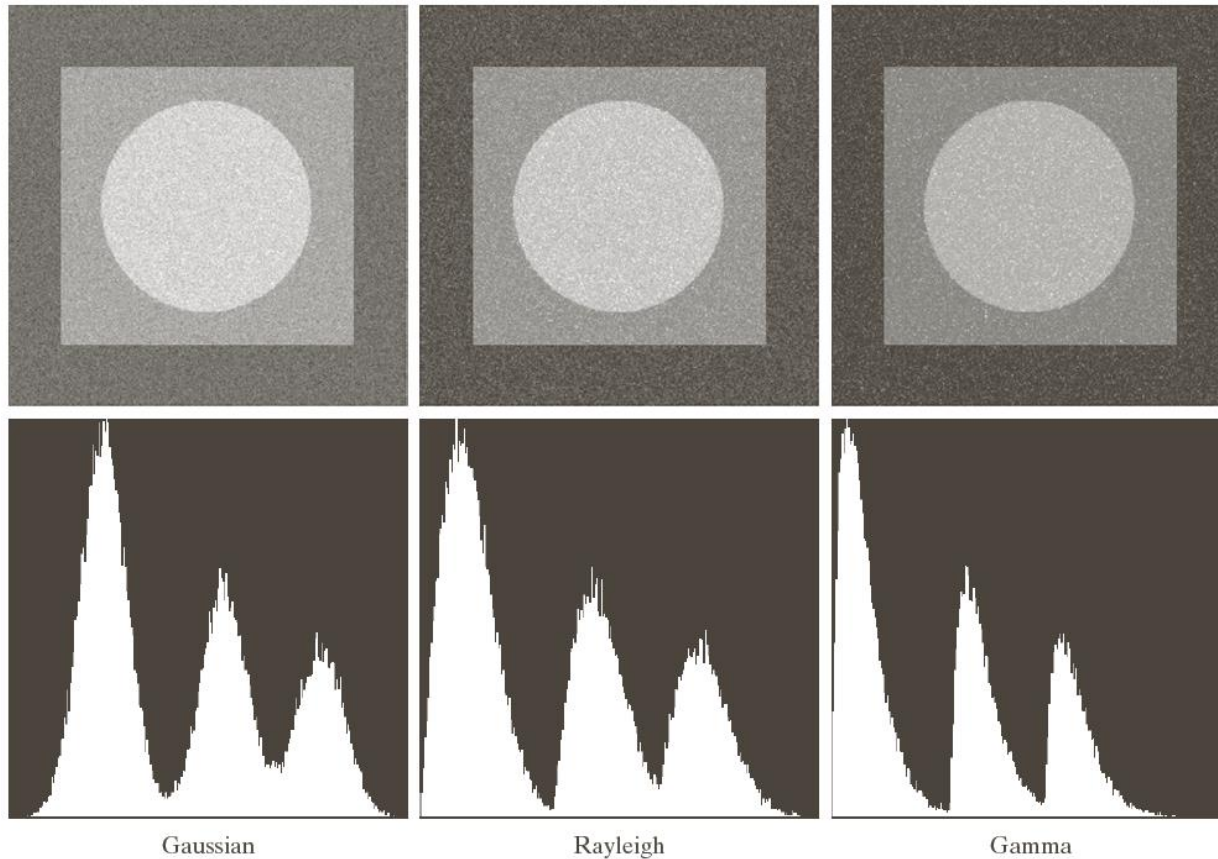
Noise Type

- Original Image



Noise Type

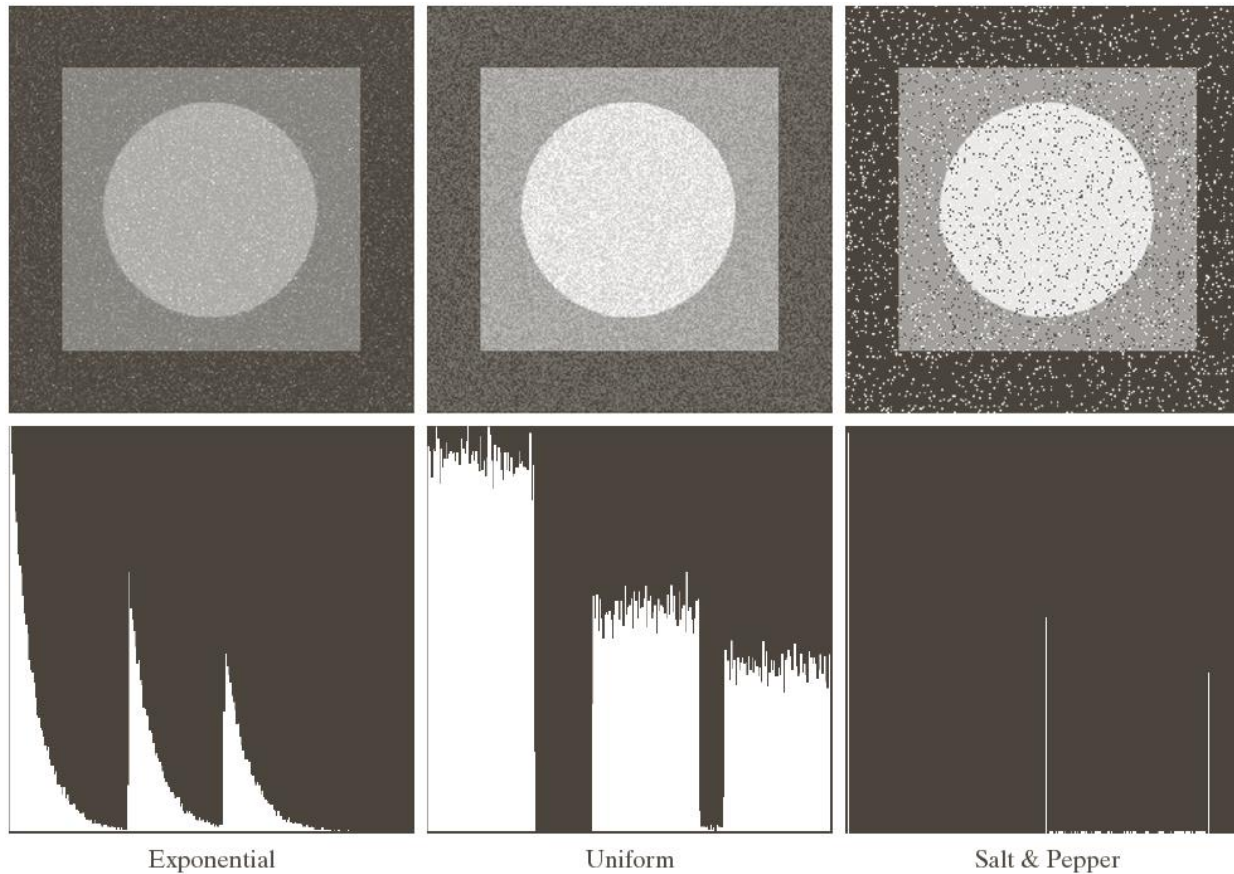
- Image and histogram



a	b	c
d	e	f

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

Noise Type



g	h	i
j	k	l

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

Matlab:

```
i=imread('checkerboard.png');  
I1=imnoise(i, 'gaussian',0.01);  
I2=imnoise(i, 'poisson');  
I3=imnoise(i, 'salt & pepper',0.01);  
I4=imnoise(i, 'speckle',0.01);  
% uniform distribution I4 = i + n * i;
```



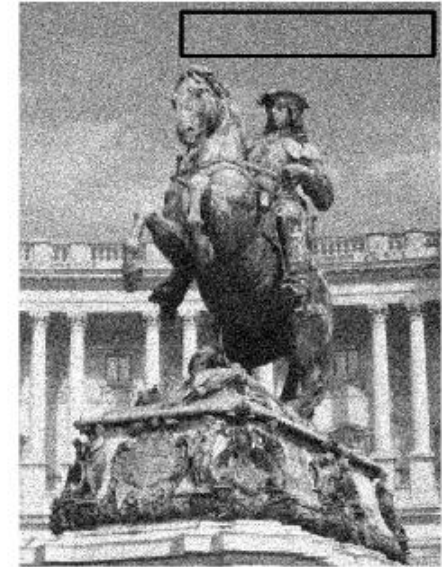
Question?

- Plot histogram of all noisy images



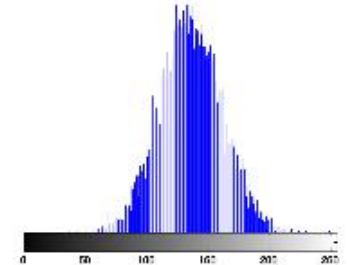
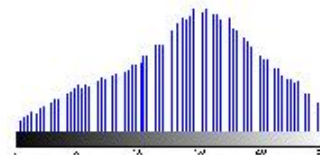
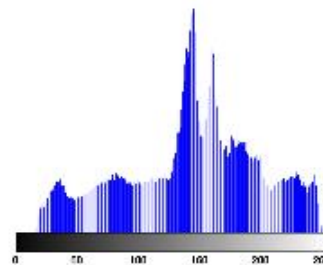
Gaussian Noise

- a) original image
- b) Image with noise
- c) Histogram of a
- d) Histogram of b
- e) Histogram of rectangle in image b



(a)

(b)

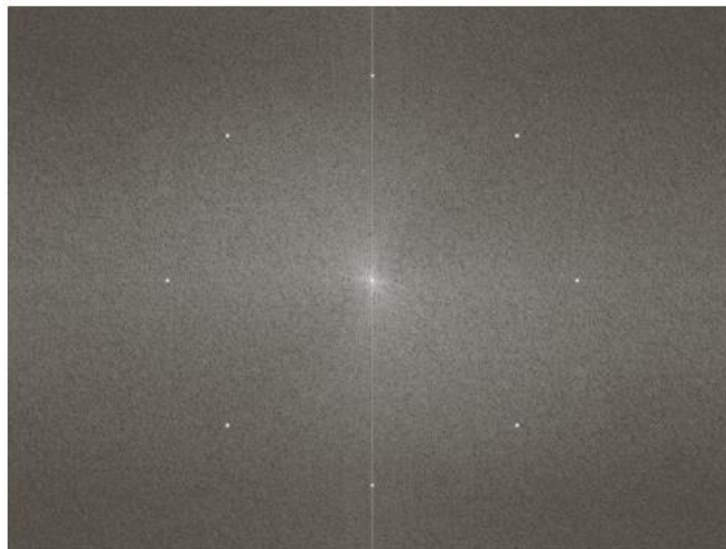
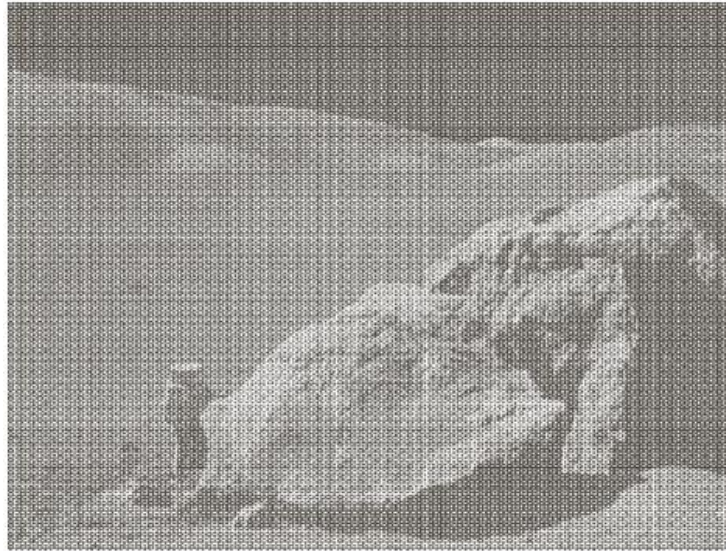


(c)

(d)

(e)

Image corrupted by sinusoidal noise



Restoration in the Presence of Noise only

- When only degradation is only noise

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

- Frequency domain

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

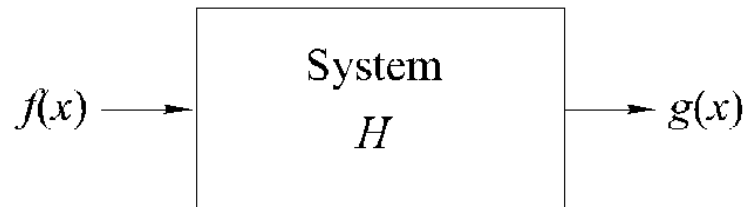
(Ad-hoc) Noise Modeling

- Simplified assumptions
 - Noise is independent of signal
- Noise types
 - Independent of spatial location
 - Impulse noise
 - Additive white Gaussian noise
 - Spatially dependent
 - Periodic noise

Noise Removal Techniques

- Linear filtering
- Nonlinear filtering

Recall



Linear system

$$\begin{aligned} H[a_i f_i(x) + a_j f_j(x)] &= a_i H[f_i(x)] + a_j H[f_j(x)] \\ &= a_i g_i(x) + a_j g_j(x) \end{aligned}$$

Image Denoising

- Introduction
- Impulse noise removal
 - Median filtering
- Additive white Gaussian noise removal
 - 2D convolution and DFT
- Periodic noise removal
 - Band-rejection and Notch filter

Impulse Noise (salt-pepper Noise)

Definition

Each pixel in an image has the probability of $p/2$ ($0 < p < 1$) being contaminated by either a white dot (salt) or a black dot (pepper)

$$Y(i, j) = \begin{cases} 255 & \text{with probability of } p/2 \\ 0 & \text{with probability of } p/2 \\ X(i, j) & \text{with probability of } 1-p \end{cases} \begin{matrix} \diagup \\ \diagdown \\ \text{---} \end{matrix} \begin{matrix} \text{noisy pixels} \\ \text{noisy pixels} \\ \text{clean pixels} \end{matrix}$$

$1 \leq i \leq H, 1 \leq j \leq W$ X: noise-free image, Y: noisy image

Note: in some applications, noisy pixels are not simply black or white, which makes the impulse noise removal problem more difficult

Numerical Example

$$P=0.1$$

128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128

X

128	128	255	0	128	128	128	128	128	128
128	128	128	128	0	128	128	128	128	0
128	128	128	128	128	128	128	128	128	128
128	128	0	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128
0	128	128	128	128	255	128	128	128	128
128	128	128	128	128	128	128	128	128	255
128	128	128	128	128	128	128	255	128	128

Y

Noise level $p=0.1$ means that approximately 10% of pixels are contaminated by salt or pepper noise (highlighted by red color)

MATLAB Command

```
>Y = IMNOISE(X,'salt & pepper',p)
```

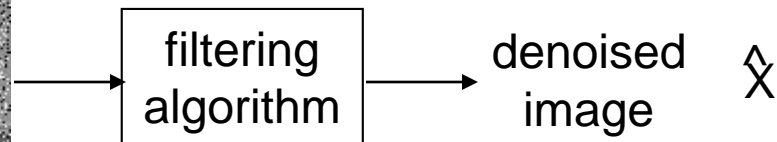
Notes:

- The intensity of input images is assumed to be normalized to [0,1].
If X is double, you need to do normalization first, i.e., $X=X/255$;
If X is uint8, MATLAB would do the normalization automatically
- The default value of p is 0.05 (i.e., 5 percent of pixels are contaminated)
- imnoise function can produce other types of noise as well (you need to change the noise type 'salt & pepper')

Impulse Noise Removal Problem



Noisy image Y



Can we make the denoised image \hat{X} as close to the noise-free image X as possible?

Median Operator

- Given a sequence of numbers $\{y_1, \dots, y_N\}$
 - Mean: average of N numbers
 - Min: minimum of N numbers
 - Max: maximum of N numbers
 - Median: half-way of N numbers

Example $\vec{y} = [50, 0, 52, 255, 54, 55, 56]$

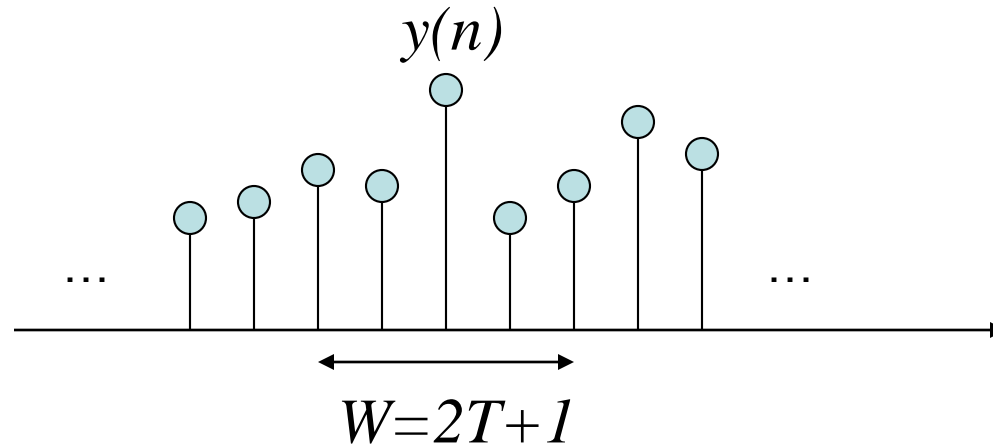
↓

sorted $\vec{y} = [0, 50, 52, 54, 55, 56, 255]$

↓

$median(\vec{y}) = 54$

1D Median Filtering



$$\hat{x}(n) = \text{median}[y(n-T), \dots, y(n), \dots, y(n+T)]$$

MATLAB command: `x=median(y(n-T:n+T));`

Note: median operator is **nonlinear**

Numerical Example

T=1:

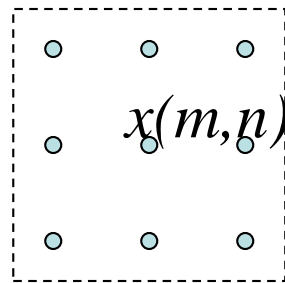
$$\vec{y} = [50, 0, 52, 255, 54, 55, 56]$$

Boundary
Padding

$$\vec{y} = [\mathbf{50}, 50, 0, 52, 255, 54, 55, 56, \mathbf{56}]$$

$$\hat{\vec{x}} = [50, 50, 52, 54, 55, 55, 56]$$

2D Median Filtering



W: $(2T+1)$ -by- $(2T+1)$ window

$$\hat{x}(m, n) = \text{median}[y(m-T, n-T), \dots, y(m-T, n+T), \dots, \\ y(m, n), \dots, y(m+T, n-T), \dots, y(m+T, n+T)]$$

MATLAB command: `x=medfilt2(y,[2*T+1,2*T+1]);`

Numerical Example

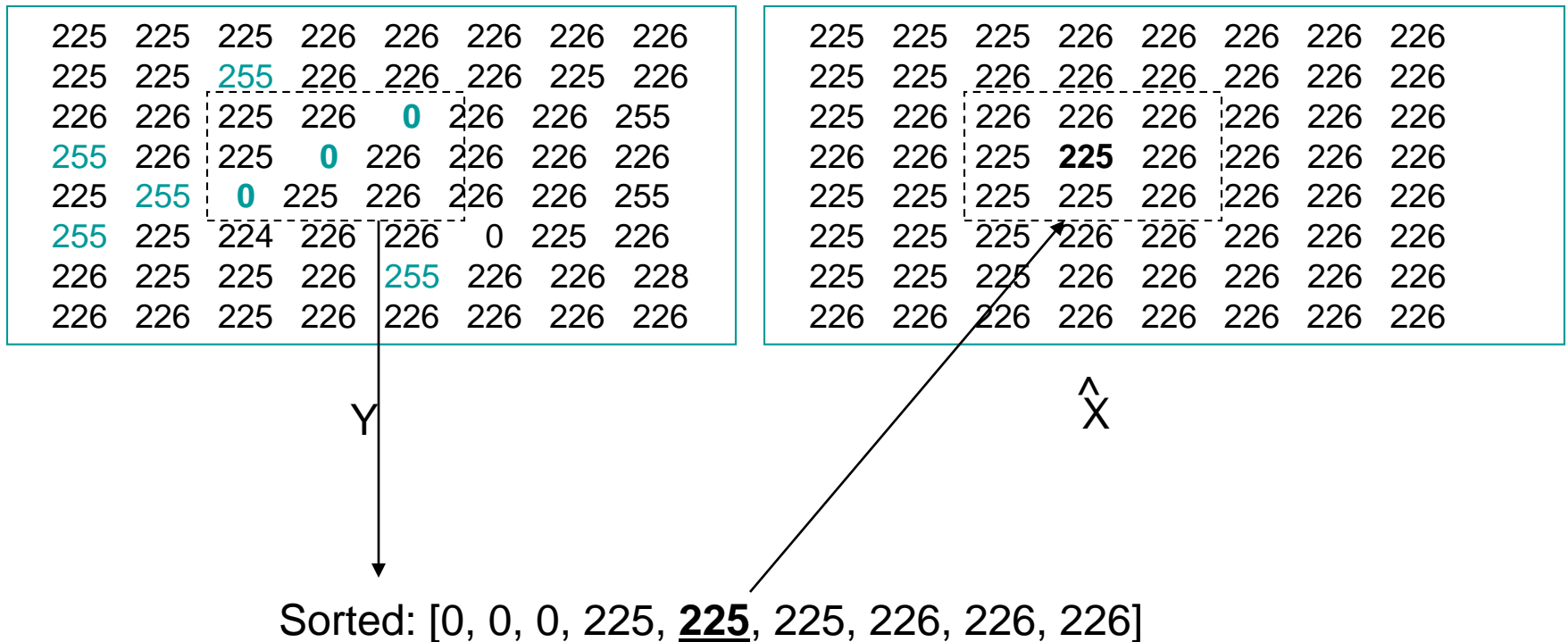
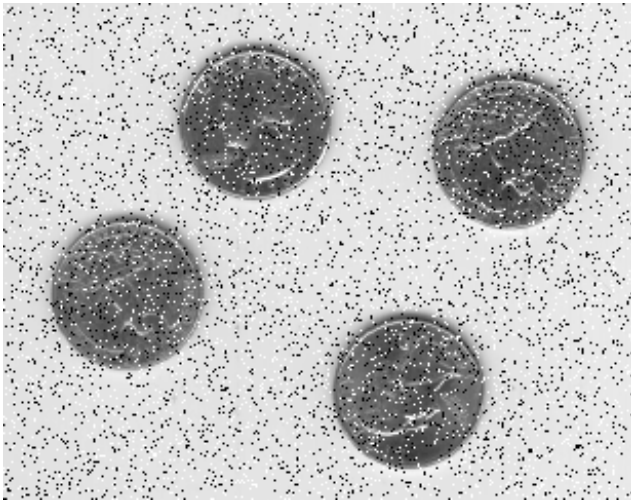


Image Example

$P=0.1$



Noisy image Y



denoised image \hat{X}
3-by-3 window

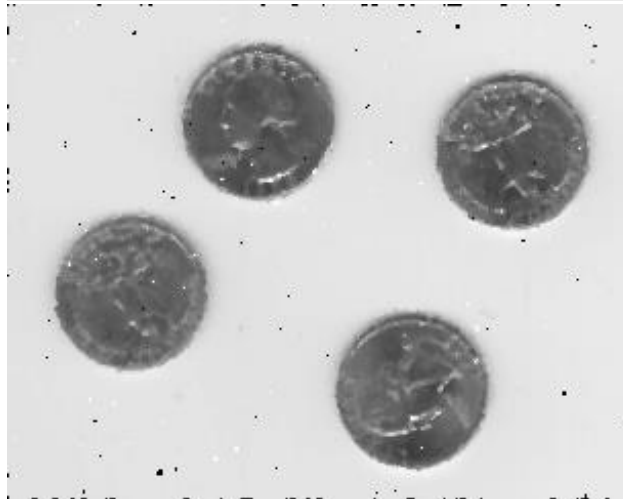
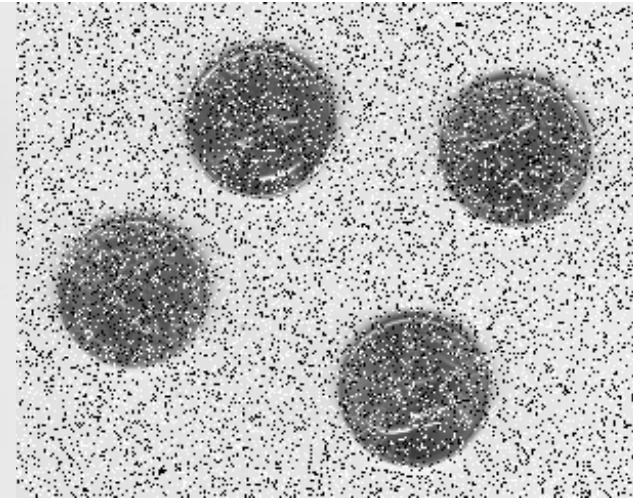


Image Example (Con't)

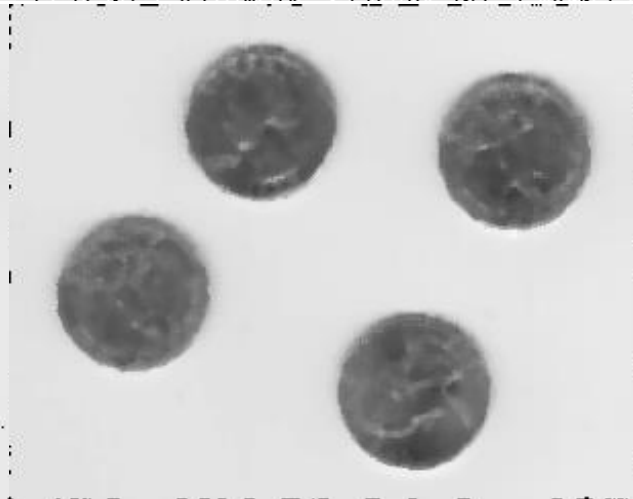
clean



noisy
($p=0.2$)




3-by-3 window




5-by-5 window



A decorative image in the top-left corner consisting of a blue square above a colorful, abstract pattern.

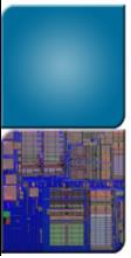
```
int main(int argc, char** argv){  
    Mat img = imread("d:/image/eight_pepper.tif",  
IMREAD_GRAYSCALE);  
    Mat dst3,dst5;  
    imshow("original", img);  
    medianBlur(img, dst3, 3);  
    medianBlur(img, dst5, 5);  
    imshow("result 3", dst3);  
    imshow("result 5", dst5);  
    waitKey();  
}
```



A decorative image in the top-left corner consisting of a blue square above a colorful, abstract pattern.

```
int main(int argc, char** argv){
    Mat img = imread("d:/image/checker_gaussian.png",
        IMREAD_GRAYSCALE);
    Mat dst3, dst5;
    imshow("original", img);
    int i = 3;
    bilateralFilter(img, dst3, i, i*2, i/2);
    i = 5;
    bilateralFilter(img, dst5, i, i*2, i/2);
    imshow("result 3", dst3);
    imshow("result 5", dst5);
    waitKey();
}
```





Try with Median filter



Reflections

- What is good about median operation?
 - Since we know impulse noise appears as black (minimum) or white (maximum) dots, taking median effectively suppresses the noise
- What is bad about median operation?
 - It affects clean pixels as well
 - Noticeable edge blurring after median filtering

Idea of Improving Median Filtering

- Can we get rid of impulse noise without affecting clean pixels?
 - Yes, if we know **where** the clean pixels are or equivalently where the noisy pixels are
- How to detect noisy pixels?
 - They are black or white dots

Median Filtering with Noise Detection

Noisy image Y



Median filtering

```
x=medfilt2(y,[2*T+1,2*T+1]);
```



Noise detection

```
C=(y==0)|(y==255);
```



Obtain filtering results

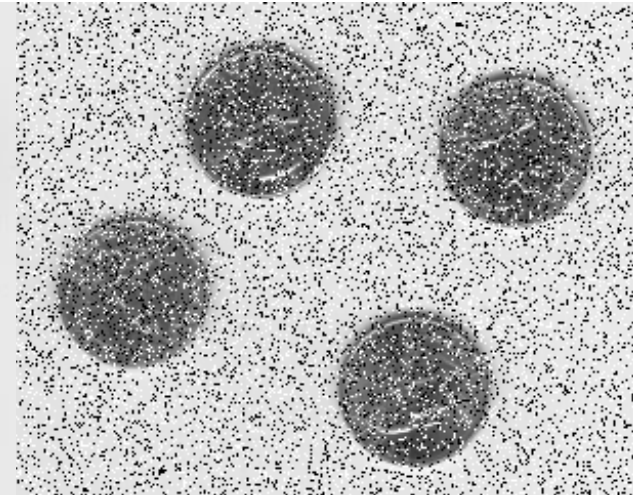
```
xx=c.*x+(1-c).*y;
```

Image Example

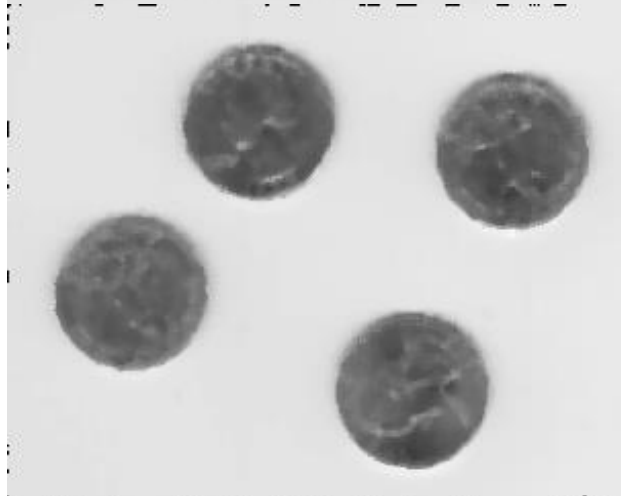
clean



noisy
($p=0.2$)



w/o
noise
detection



with
noise
detection



Image Denoising

- Introduction
- Impulse noise removal
 - Median filtering
- Additive white Gaussian noise removal
 - 2D convolution and DFT
- Periodic noise removal
 - Band-rejection and Notch filter

Additive White Gaussian Noise

Definition

Each pixel in an image is disturbed by a Gaussian random variable
With zero mean and variance σ^2

$$Y(i, j) = X(i, j) + N(i, j),$$

$$N(i, j) \sim N(0, \sigma^2), 1 \leq i \leq H, 1 \leq j \leq W$$

X: noise-free image, Y: noisy image

Note: unlike impulse noise situation, every pixel in the image contaminated by AWGN is noisy

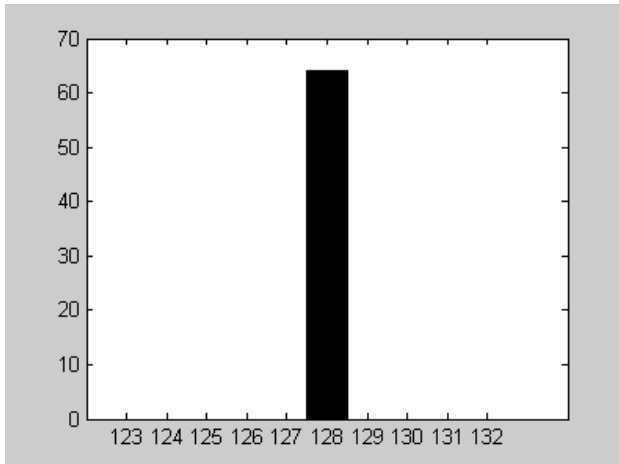
Numerical Example

128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128

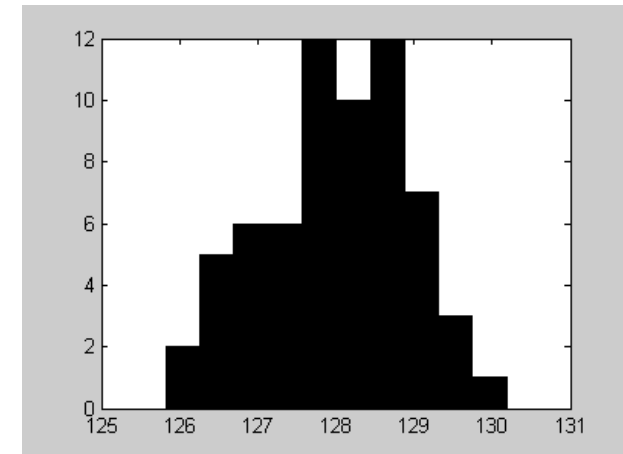
$$\sigma^2 = 1$$

128	128	129	127	129	126	126	128
126	128	128	129	129	128	128	127
128	128	128	129	129	127	127	128
128	129	127	126	129	129	129	128
127	127	128	127	129	127	129	128
129	130	127	129	127	129	130	128
129	128	129	128	128	128	129	129
128	128	130	129	128	127	127	126

X



Y



MATLAB Command

`>Y = IMNOISE(X, 'gaussian',m,v)`

or

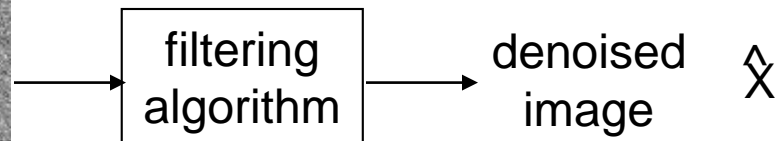
`>Y = X+m+randn(size(X))*v;`

Note: `rand()` generates random numbers uniformly distributed over [0,1]
`randn()` generates random numbers observing Gaussian distribution $N(0,1)$

Image Denoising

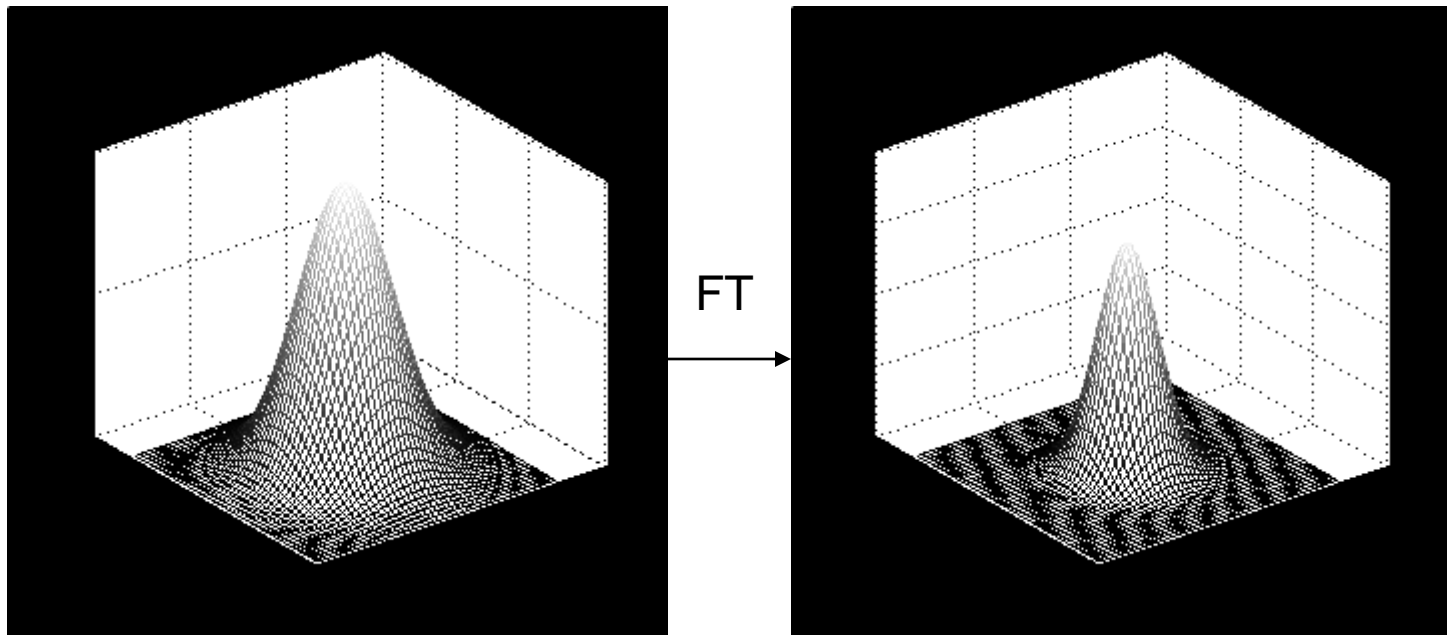


Noisy image Y



Question: Why not use median filtering?
Hint: the noise type has changed.

Gaussian Filter



$$h(m, n) = \exp\left(-\frac{m^2 + n^2}{2\sigma^2}\right)$$

$$H(w_1, w_2) = \exp\left(-\frac{w_1^2 + w_2^2}{2\sigma^2}\right)$$

MATLAB code: `>h=fspecial('gaussian', HSIZE, SIGMA);`

Image Example

noisy



$PSNR=20.2dB$
($\sigma=25$)

denoised



$PSNR=24.4dB$
($\sigma=1$)

denoised



$PSNR=22.8dB$
($\sigma=1.5$)

Matlab functions: imfilter, filter2

Mean Filter

- Arithmetic mean filter

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

- Geometric mean filter

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

- Harmonic mean filter

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

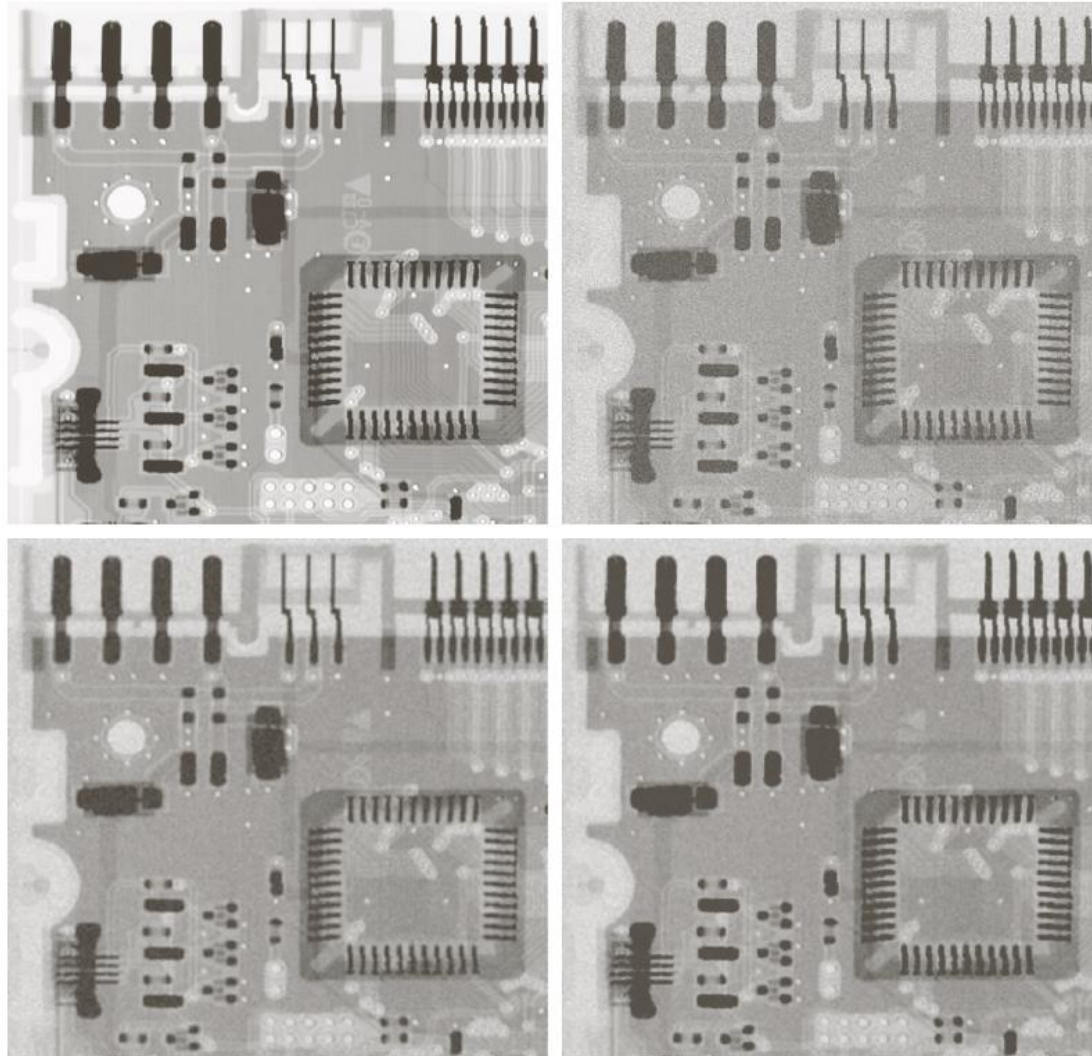


Example

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 3×3 .
 (d) Result of filtering with a geometric mean filter of the same size.
 (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)



Geometric means filter does not blur the image as much as arithmetic mean

Order-Statistic Filter

- Median filter (work with impulse noise)

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

- Max and Min filter (work with peper & salt)

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

- Midpoint filter (work with gaussian)

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

Gaussian Noise Reduction

- a) original image
- b) Image with Gaussian noise
- c) 3x3 arithmetic mean filter
- d) 5x5 arithmetic mean filter
- e) 3x3 geometric mean filter
- f) 3x3 harmonic mean filter



(a)



(b)



(c)



(d)



(e)




(f)



Remarks


- Four filters contains less noise that image b)
- 5x5 window contains less noise, but it is more blurry
- Geometric and harmonic mean filters generate more dark pixels



A small checkerboard pattern image in the top-left corner.

```
int main(int argc, char** argv){  
    Mat img = imread("d:/image/checker_gaussian.png",  
IMREAD_GRAYSCALE);  
    Mat dst3, dst5;  
    imshow("original", img);  
    GaussianBlur(img, dst3, Size(3, 3), 0, 0);  
    GaussianBlur(img, dst5, Size(5, 5), 0, 0);  
    imshow("result 3", dst3);  
    imshow("result 5", dst5);  
    waitKey();  
}
```



A small checkerboard pattern image in the top-left corner.

```
int main(int argc, char** argv){  
    Mat img = imread("d:/image/checker_gaussian.png",  
        IMREAD_GRAYSCALE);  
    Mat dst3, dst5;  
    imshow("original", img);  
    int i = 3;  
    bilateralFilter(img, dst3, i, i*2, i/2);  
    i = 5;  
    bilateralFilter(img, dst5, i, i*2, i/2);  
    imshow("result 3", dst3);  
    imshow("result 5", dst5);  
    waitKey();  
}
```





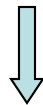
Try Median Filter



Gaussian Filter=Heat Diffusion

Linear Heat Flow Equation:

$$\frac{\partial I(x, y, t)}{\partial t} = \Delta I(x, y, t) = \frac{\partial^2 I(x, y, t)}{\partial x^2} + \frac{\partial^2 I(x, y, t)}{\partial y^2}$$



Isotropic diffusion: $I(x, y, t) = I(x, y, 0) \otimes G(t)$

↓
scale

↓
A Gaussian filter
with zero mean
and variance of t

Basic Idea of Nonlinear Diffusion*

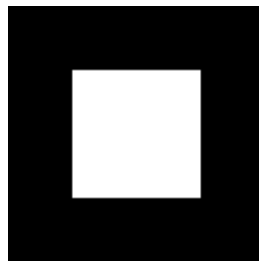


image I

Diffusion should be anisotropic instead of isotropic

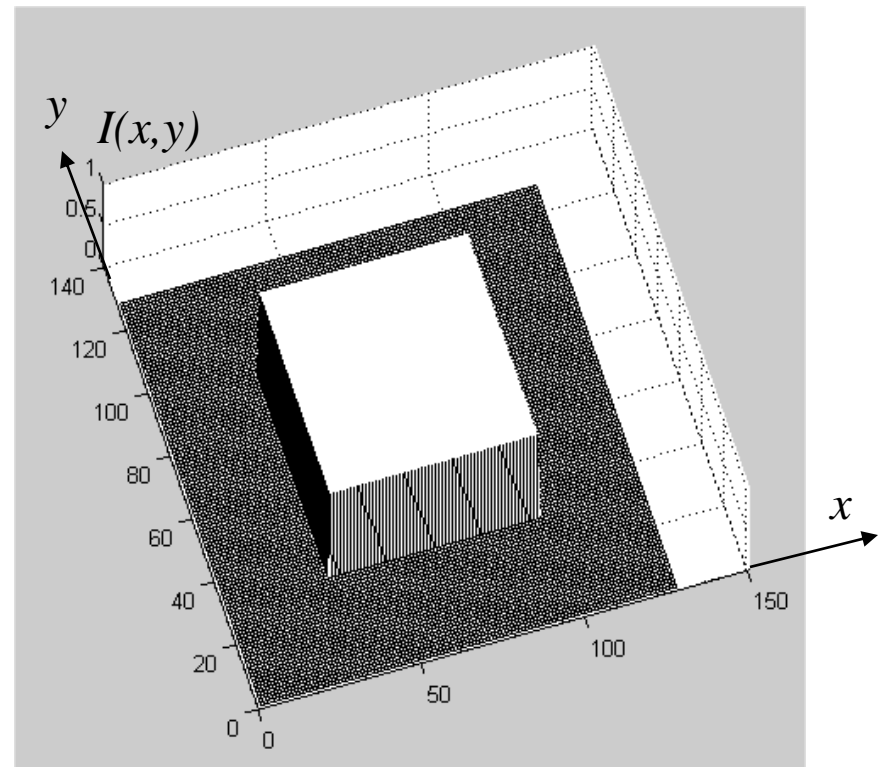


image I viewed as a 3D surface $(x,y,I(x,y))$

Experimental Results

noisy



$PSNR=20.2dB$
($\sigma=25$)

linear diffusion



$PSNR=24.4dB$
(Gaussian filtering)

nonlinear diffusion



$PSNR=27.5dB$
(TV filtering)

Salt & Pepper Noise Reduction

- a) image with noise
- b) 3x3 arithmetic mean filter
- c) 3x3 median filter
- d) 3x3 midpoint filter



(a)



(b)



(c)



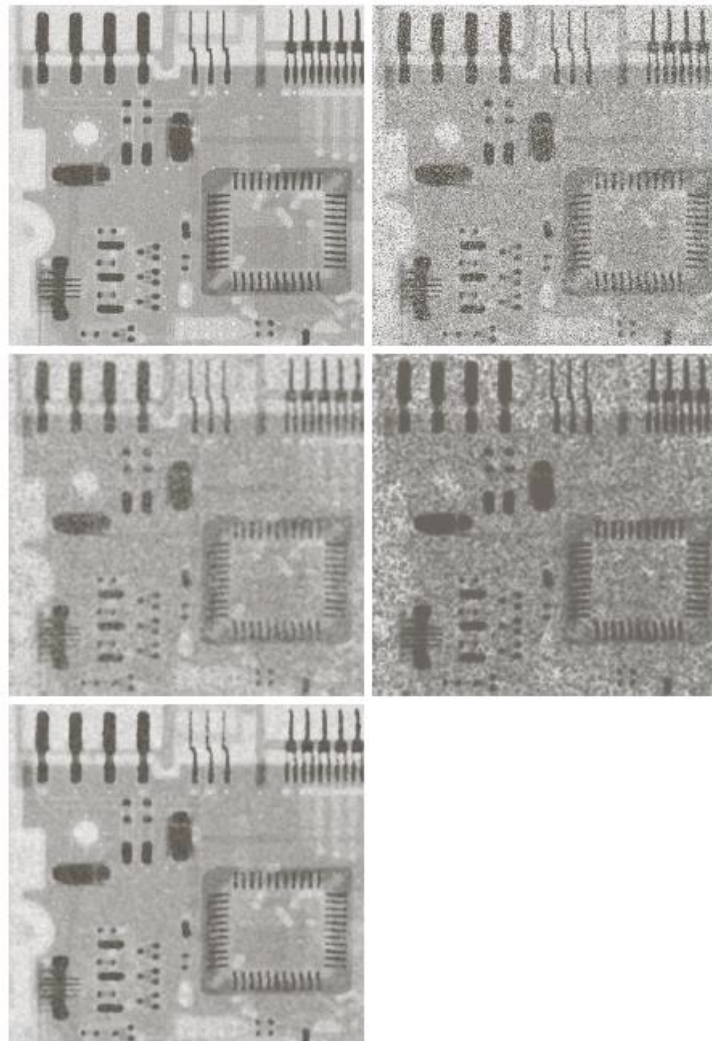
(d)



Remarks

- Mean filter is better than arithmetic filter
- Midpoint filter generates more noise

Example



a	b
c	d
e	f

FIGURE 5.12

(a) Image corrupted by additive uniform noise.
 (b) Image additionally corrupted by additive salt-and-pepper noise.
 Image (b) filtered with a 5×5 ;
 (c) arithmetic mean filter;
 (d) geometric mean filter;
 (e) median filter;
 and (f) alpha-trimmed mean filter with $d = 5$.

Hammer-Nail Analogy

Gaussian filter



median filter



???

salt-pepper/
impulse noise



Gaussian noise



Copyright © - RAMM Fencing

periodic noise



Image Denoising

- Introduction
- Impulse noise removal
 - Median filtering
- Additive white Gaussian noise removal
 - 2D convolution and DFT
- Periodic noise removal
 - Band-rejection and Notch filter

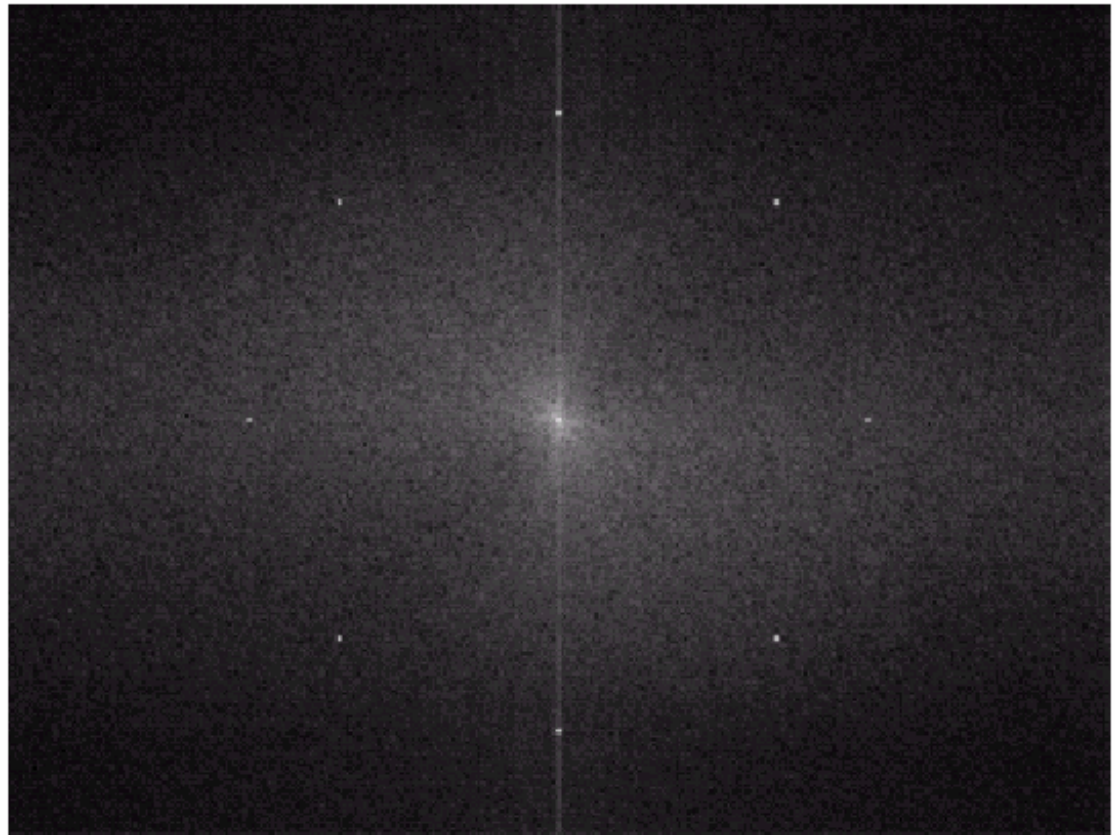
Periodic Noise

- Source: electrical or electromechanical interference during image acquisition
- Characteristics
 - Spatially dependent
 - Periodic – easy to observe in frequency domain
- Processing method
 - Suppressing noise component in frequency domain

Image Example



spatial



Frequency (note the four pairs of bright dots)

MATLAB:

```
i=imread('f617a.tif');  
imshow(i);  
b=imread('periodic_noise.png');  
b1=b(:,:,1);  
BF=fft2(b1);  
imshow(log(1+abs(fftshift(BF))),[]);  
j=i+b1;  
imshow(j);  
JF=fft2(j);  
imshow(log(1+abs(fftshift(JF))),[]);
```





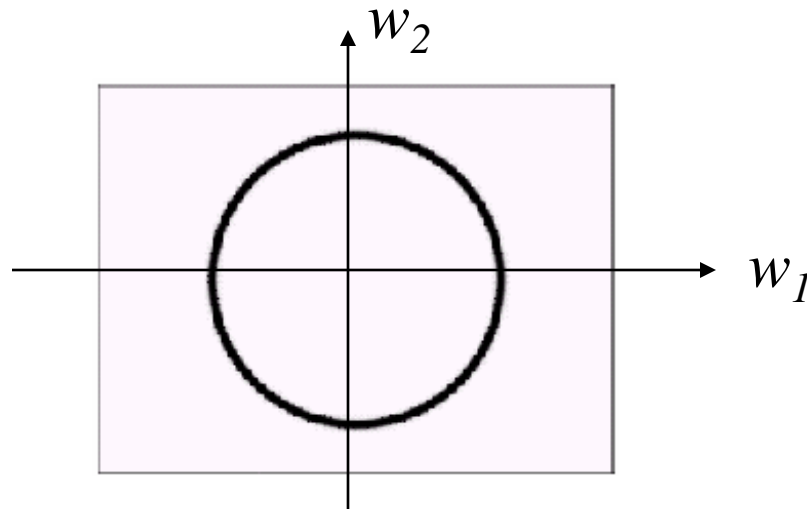
Frequency Domain Filtering

- Band reject filter
- Band pass filter
- Notch filter



Band Rejection Filter

$$H(w_1, w_2) = \begin{cases} 0 & D - \frac{W}{2} \leq \sqrt{w_1^2 + w_2^2} \leq D + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$$

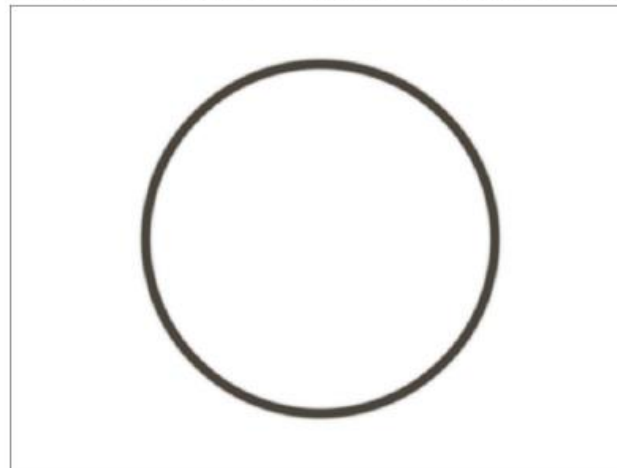
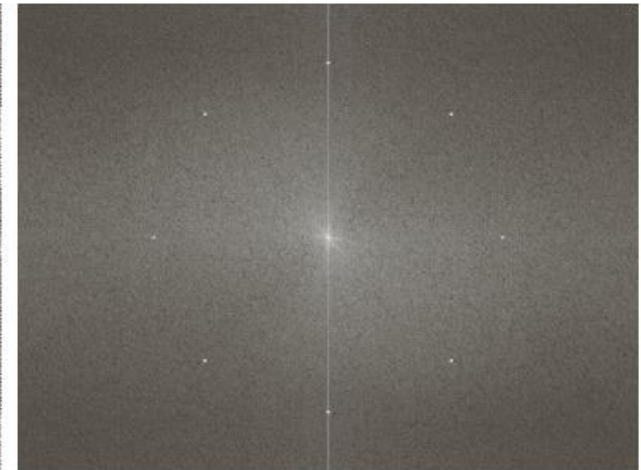
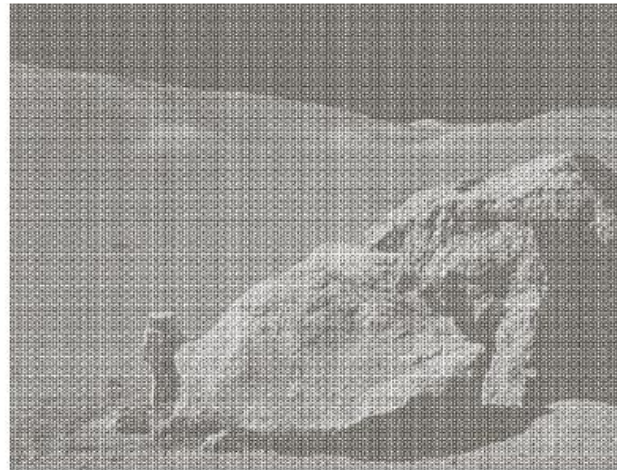


Example

a	b
c	d

FIGURE 5.16

(a) Image corrupted by sinusoidal noise.
 (b) Spectrum of (a).
 (c) Butterworth bandreject filter (white represents 1).
 (d) Result of filtering.
 (Original image courtesy of NASA.)



Band Reject Filter

- Band reject filter is for noise removal in applications where the location in the frequency domain is known.



FIGURE 5.15 From left to right, perspective plots of ideal, Butterworth (of order 1), and Gaussian bandreject filters.

Bandpass Filter

- A bandpass filter is the opposite operation of band reject filter

$$H_{BP}(u, v) = 1 - H_{BR}(u, v)$$

Example

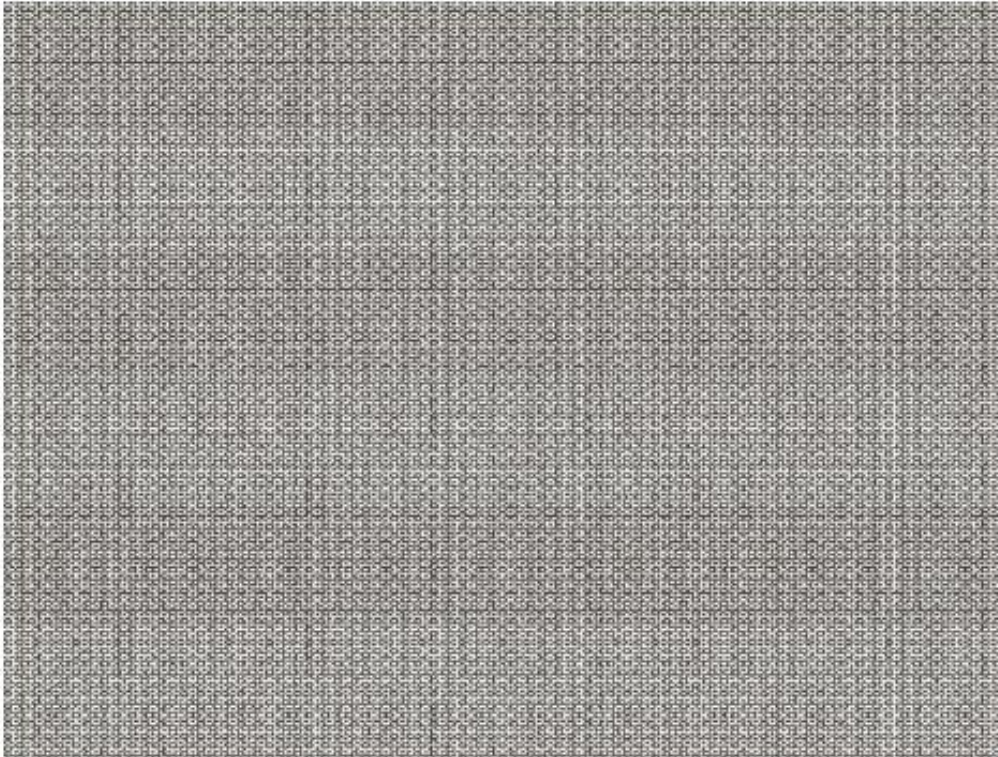


FIGURE 5.17
Noise pattern of
the image in
Fig. 5.16(a)
obtained by
bandpass filtering.



Notch Filter

- Band reject filter with a very narrow band
- Notch filter is a special type of band reject (band pass) filter

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

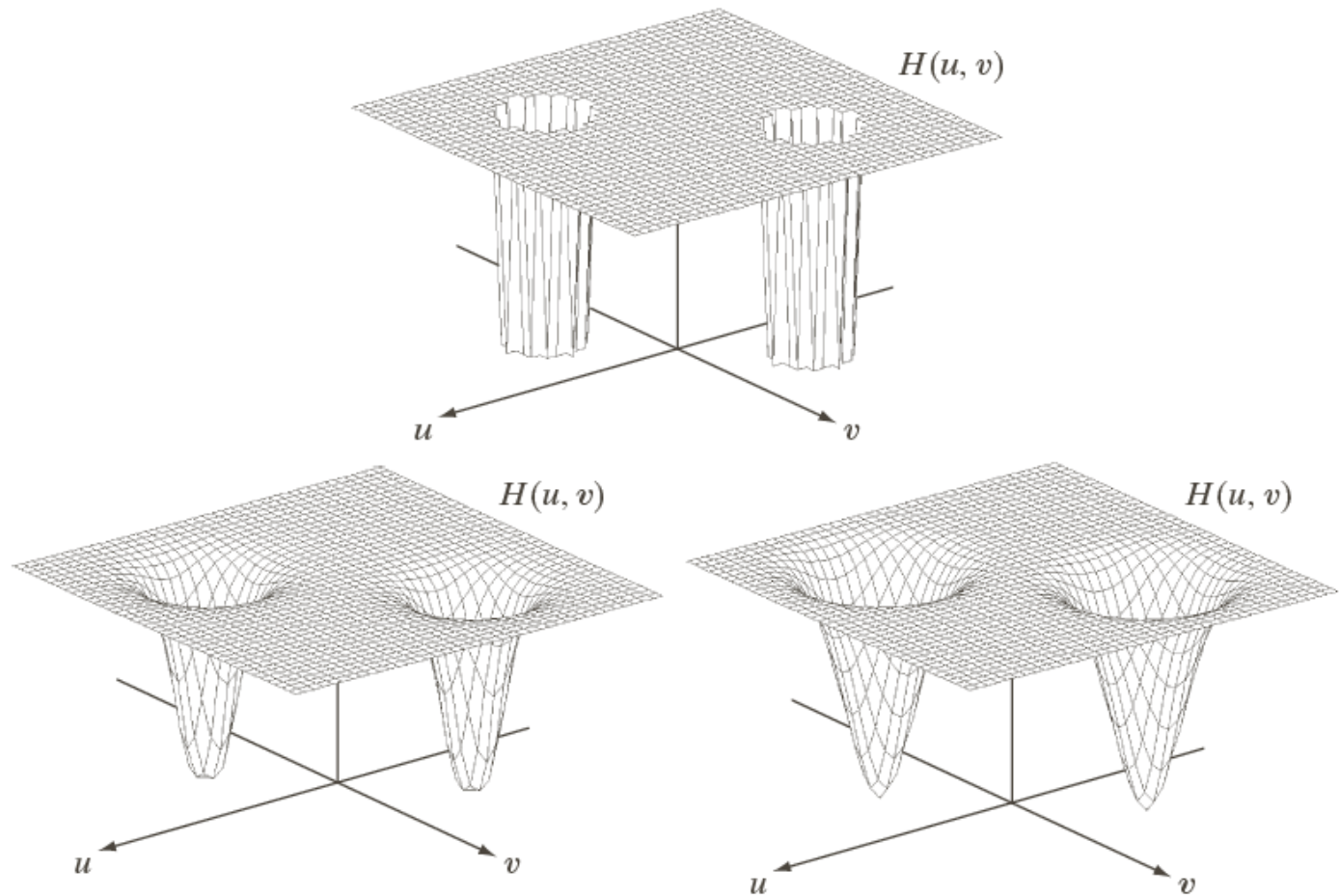
Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u, v) = \frac{1}{1 + \left[\frac{DW}{D^2 - D_0^2} \right]^{2n}}$	$H(u, v) = 1 - e^{-\left[\frac{D^2 - D_0^2}{DW} \right]^2}$

Notch Filter

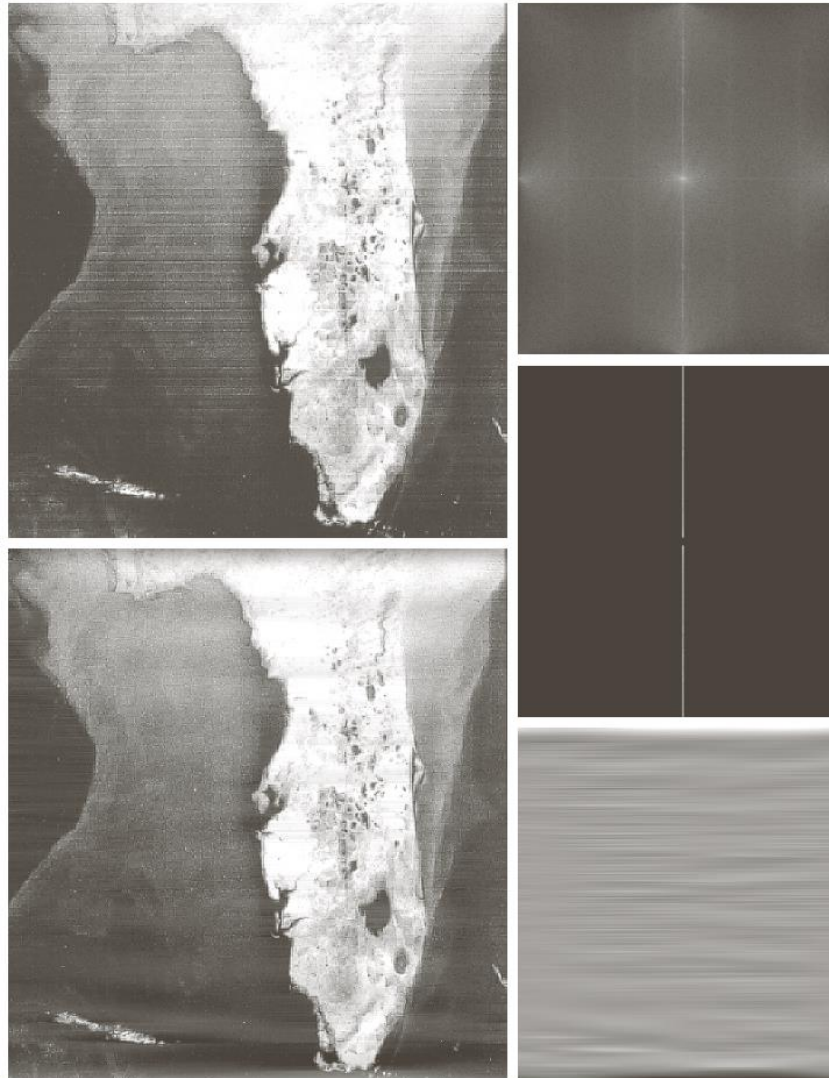
a
b c

FIGURE 5.18

Perspective plots of (a) ideal, (b) Butterworth (of order 2), and (c) Gaussian notch (reject) filters.



Example



a	b
e	d

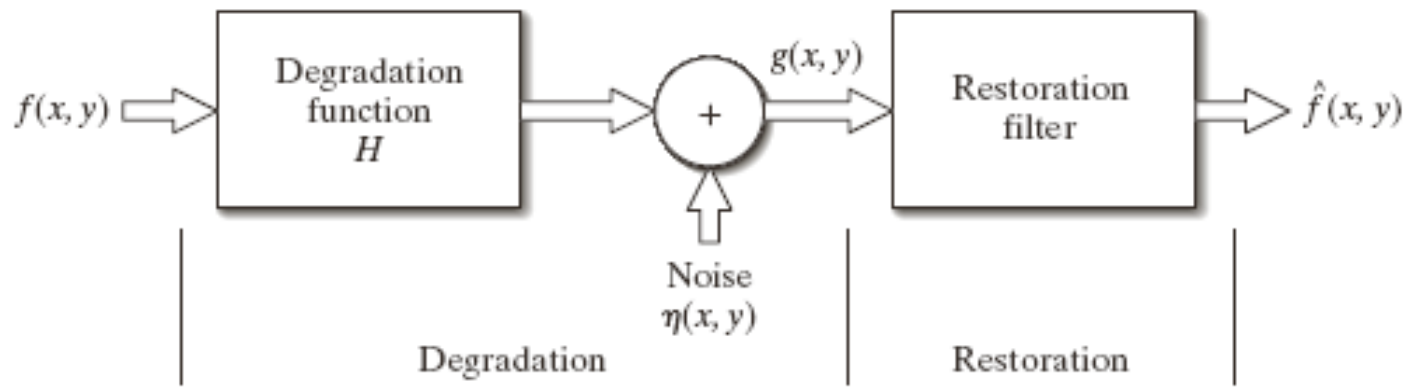
FIGURE 5.19

(a) Satellite image of Florida and the Gulf of Mexico showing horizontal scan lines. (b) Spectrum. (c) Notch pass filter superimposed on (b). (d) Spatial noise pattern. (e) Result of notch reject filtering. (Original image courtesy of NOAA.)

The Model

FIGURE 5.1

A model of the
image degradation/
restoration process.



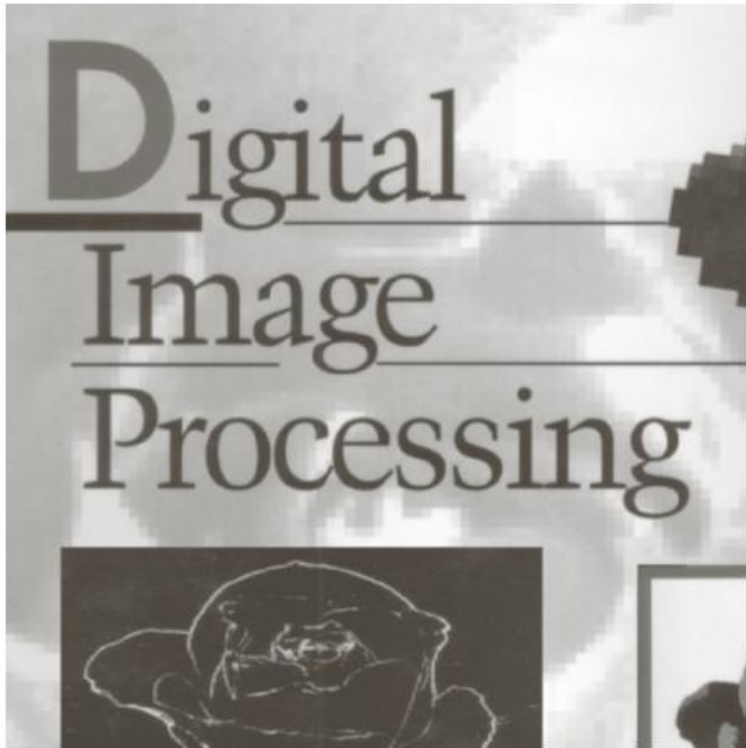
The Degradation Function

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

- There are three ways to estimate the degradation function
 - Observation
 - Experimentation
 - Mathematical modeling

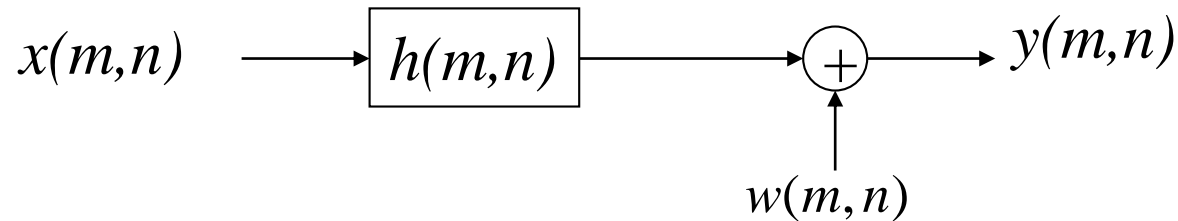
Example of Degradation Function

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



Modeling Blurring Process

- Linear degradation model

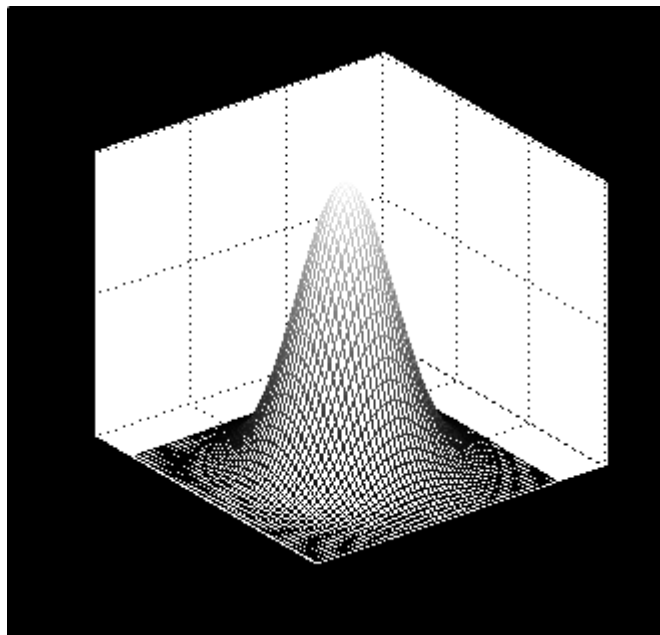


$h(m, n)$

blurring filter

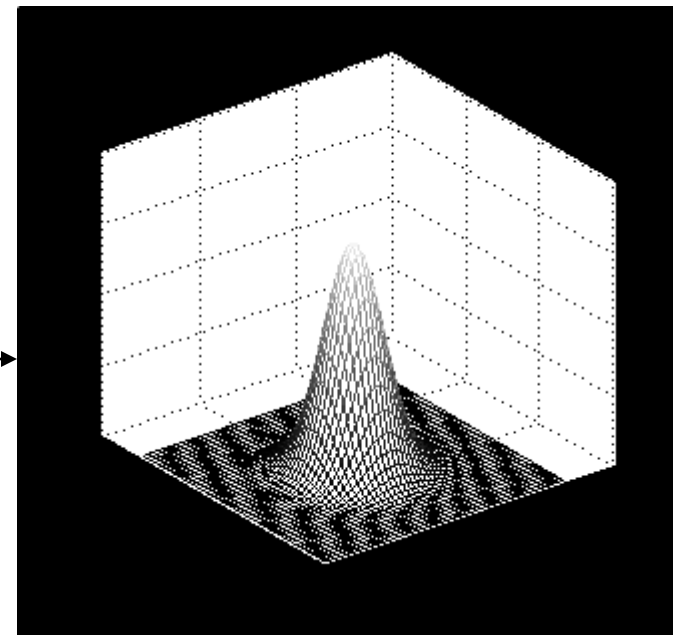
$w(m, n) \sim N(0, \sigma_w^2)$ additive white Gaussian noise

Blurring Filter Example



$$h(m, n) = \exp\left(-\frac{m^2 + n^2}{2\sigma^2}\right)$$

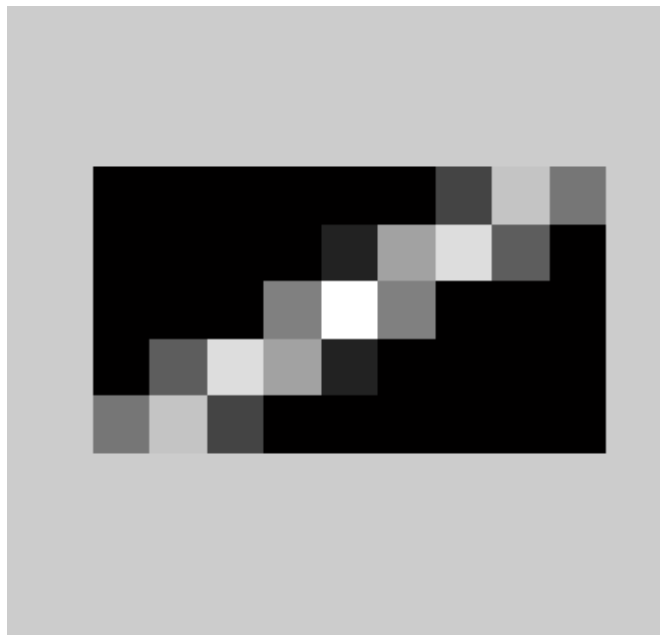
FT



$$H(w_1, w_2) = \exp\left(-\frac{w_1^2 + w_2^2}{2\sigma^2}\right)$$

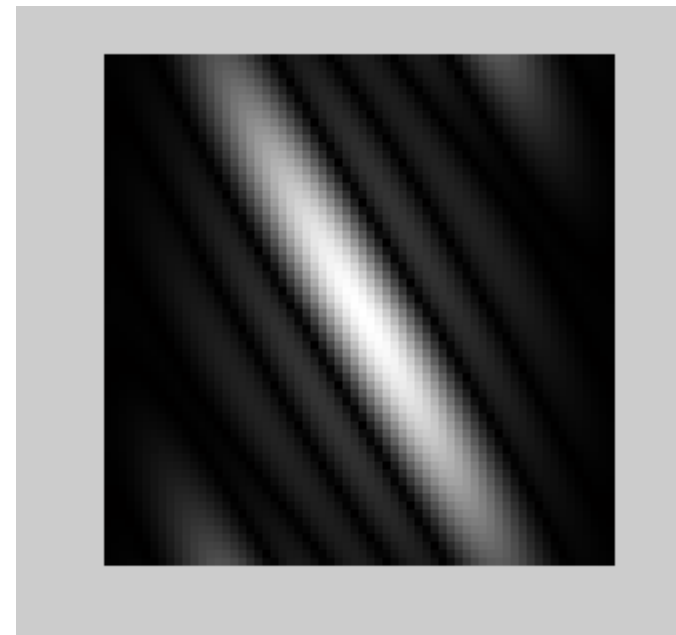
Gaussian filter can be used to approximate out-of-focus blur

Blurring Filter Example (Con't)



$h(m,n)$

FT
→

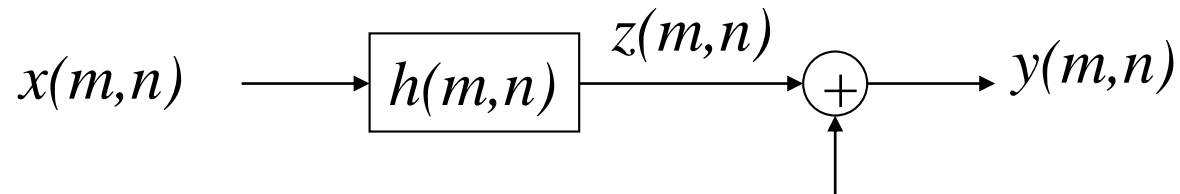


$H(w_1, w_2)$

MATLAB code: `h=FSPECIAL('motion',9,30);`

Motion blurring can be approximated by 1D low-pass filter along the moving direction

The Curse of Noise



$$w(m,n) \sim N(0, \sigma_w^2)$$

Blurring SNR

$$BSNR = 10 \log_{10} \frac{\sigma_z^2}{\sigma_w^2}$$

Image Example



$x(m,n)$




BSNR=40dB



BSNR=10dB

$h(m,n)$: 1D horizontal motion blurring $[1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]/7$

Matlab

A decorative image in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

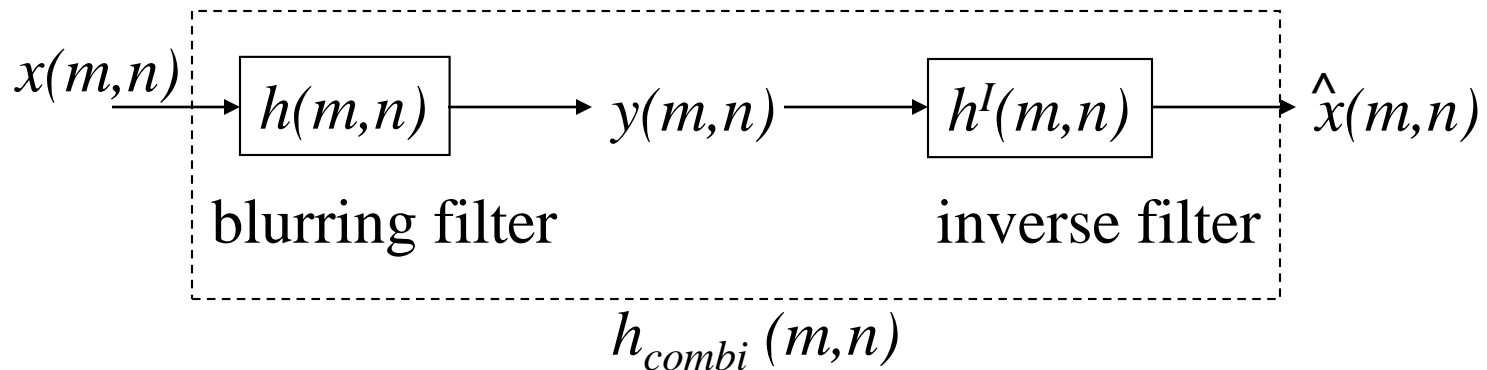
```
f=imread('lenna.tif');  
h = fspecial('motion',30,90);  
% magnitude and direction (0°= horizontal)  
m = imfilter(f,h,'replicate');  
Imshow(m);  
  
% try with different magnitude and direction
```



Blind vs. Nonblind Deblurring

- Blind deblurring (deconvolution):
blurring kernel $h(m,n)$ is unknown
- Nonblind deconvolution:
blurring kernel $h(m,n)$ is known
- In this course, we only cover the
nonblind case (the easier case)

Inverse Filter



To compensate the blurring, we require

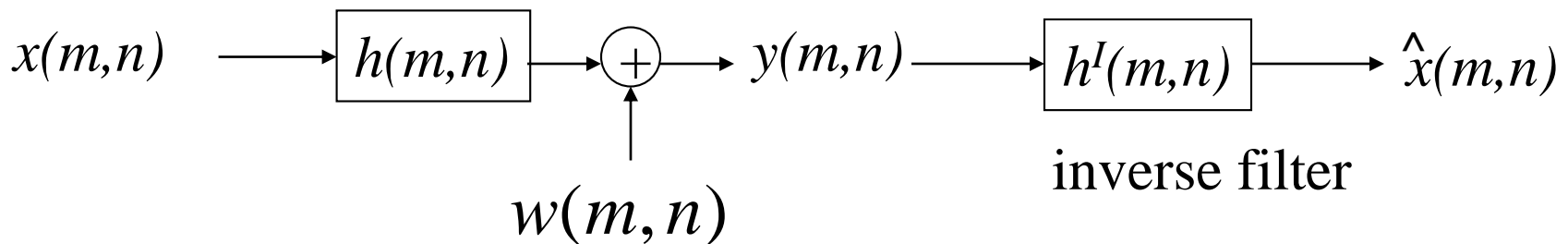
$$h_{combi}(m,n) = h(m,n) \otimes h^I(m,n) =$$

$$\sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h^I(m-k, n-l) h(k, l) = \delta(m, n), \forall (m, n)$$



$$H^I(w_1, w_2) = \frac{1}{H(w_1, w_2)}$$

Inverse Filtering (Con't)



Spatial:

$$\hat{x}(m, n) = y(m, n) \otimes h^I(m, n) = (x(m, n) \otimes h(m, n) + w(m, n)) \otimes h^I(m, n)$$

Frequency:

$$\hat{X}(w_1, w_2) = Y(w_1, w_2)H^I(w_1, w_2) = \frac{X(w_1, w_2)H(w_1, w_2) + W(w_1, w_2)}{H(w_1, w_2)}$$

$$= X(w_1, w_2) + \frac{W(w_1, w_2)}{H(w_1, w_2)} \rightarrow \text{amplified noise}$$

Inverse Filtering

- Assume that the image is degraded by function H , the inverse filter can estimate the original image $\hat{F}(u,v)$ as follows:

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

- With the noise model:

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



A decorative image in the top-left corner consisting of a blue square above a circuit board pattern.

Problem with Inverse Filtering

- $H(u,v)$ function is usually not known
- $N(u,v)$ function is not known
- There will be a problem, if $H(u,v)$ function is small or close to zero



Image Example



motion blurred image
at BSNR of 40dB



deblurred image after
inverse filtering

Pseudo-inverse Filter

Basic idea:

To handle zeros in $H(w_1, w_2)$, we treat them separately when performing the inverse filtering

$$H^{-}(w_1, w_2) = \begin{cases} \frac{1}{H(w_1, w_2)} & |H(w_1, w_2)| > \delta \\ 0 & |H(w_1, w_2)| \leq \delta \end{cases}$$

Image Example



motion blurred image
at BSNR of 40dB



deblurred image after
Pseudo-inverse filtering
($\delta=0.1$)

MATLAB Example

```
N=512;                n=.2;
k=0; % try k = 0 to 10
f=imread('lenna.tif');
figure(1); imagesc(f); colormap(gray);
b=ones(9,9)/9^2;
F=fft2(f);
B=fft2(b,N,N);
G=F.*B;
g=ifft2(G)+k*randn(N,N);
G=fft2(g);
figure(2); imagesc(abs(ifft2(G))); colormap(gray);
BF=find(abs(B)<n); %B(BF)=max(max(B))/1.5;
B(BF)=n;
H=ones(N,N)./B;
figure(4); mesh(abs(fftshift(H)));
I=G.*H;
im=abs(ifft2(I));
figure(3);imagesc(im); colormap(gray)
```

Norbert Wiener (1894-1964)



The renowned MIT professor Norbert Wiener was famed for his absent-mindedness. While crossing the MIT campus one day, he was stopped by a student with a mathematical problem. The perplexing question answered, Norbert followed with one of his own: "In which direction was I walking when you stopped me?" he asked, prompting an answer from the curious student. "Ah," Wiener declared, "then I've had my lunch"

Anecdote of Norbert Wiener

Weiner Filtering

- Wiener filter is based on minimum mean square error

$$e^2 = E\{(f - \hat{f})^2\}$$

Minimum Mean Square Error

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

$H(u, v)$ = degradation function

$H^*(u, v)$ = complex conjugate of $H(u, v)$

$|H(u, v)|^2 = H^*(u, v)H(u, v)$

$S_\eta(u, v) = |N(u, v)|^2$ = power spectrum of the noise [see Eq. (4.6-18)][†]

$S_f(u, v) = |F(u, v)|^2$ = power spectrum of the undegraded image



Wiener Filtering

Also called Minimum Mean Square Error (MMSE) or Least-Square (LS) filtering

$$H_{mmse}(w_1, w_2) = \frac{H^*(w_1, w_2)}{|H(w_1, w_2)|^2 + K}$$

constant

Example choice of K:

$$K = \frac{\sigma_w^2}{\sigma_z^2}$$

noise energy

signal energy

$K=0 \rightarrow$ inverse filtering



MATLAB

```
I = im2double(imread('f617a.tif'));  
imshow(I);  
%simulate motion blur  
  
LEN = 21;  
THETA = 11;  
PSF = fspecial('motion', LEN, THETA);  
blurred = imfilter(I, PSF, 'conv', 'circular');  
figure,imshow(blurred);  
title('Blurred Image');  
  
% wiener filter  
  
wnr1 = deconvwnr(blurred, PSF, 0.01);  
figure,imshow(wnr1);  
title('Restored Image');
```



Image Example



motion blurred image
at BSNR of 40dB



deblurred image after
wiener filtering
($K=0.01$)



Image Example (Con't)



$K=0.1$



$K=0.01$



$K=0.001$



Constrained Least Square Filtering

Similar to Wiener but a different way of balancing the tradeoff between

$$H_{mmse}(w_1, w_2) = \frac{H^*(w_1, w_2)}{|H(w_1, w_2)|^2 + \gamma |C(w_1, w_2)|^2}$$

Example choice of C:

$$C(m, n) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$\gamma=0 \rightarrow$ inverse filtering

Laplacian operator

Image Example



$\gamma=0.1$



$\gamma=0.01$



$\gamma=0.001$

Example



FIGURE 5.29 (a) 8-bit image corrupted by motion blur and additive noise. (b) Result of inverse filtering. (c) Result of Wiener filtering. (d)–(f) Same sequence, but with noise variance one order of magnitude less. (g)–(i) Same sequence, but noise variance reduced by five orders of magnitude from (a). Note in (h) how the deburred image is quite visible through a “curtain” of noise.

Application: Image Deblurring

Can be solved by:

- Inverse filtering
 - Suffer from noise amplification
- Wiener filtering
 - Tradeoff between image recovery and noise suppression
- Iterative deblurring*
 - Landweber algorithm

Blurring

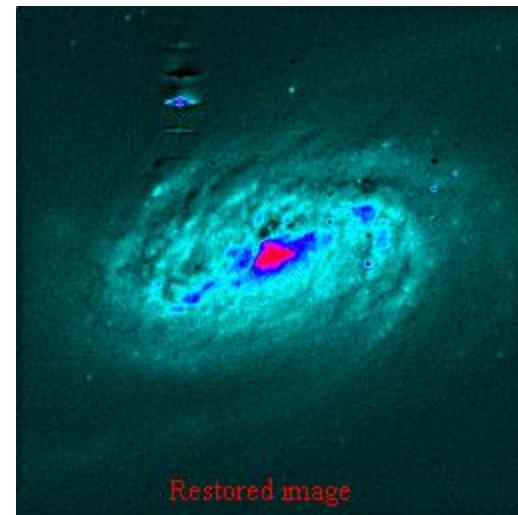
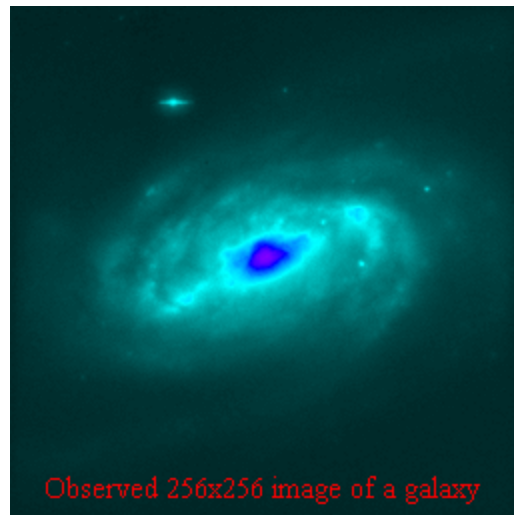
- Where does blur come from?
 - Optical blur: camera is out-of-focus
 - Motion blur: camera or object is moving
- Why do we need deblurring?
 - Visually annoying
 - Wrong target for compression
 - Bad for analysis
 - Numerous applications

Application (I): Astronomical Imaging

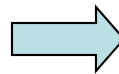
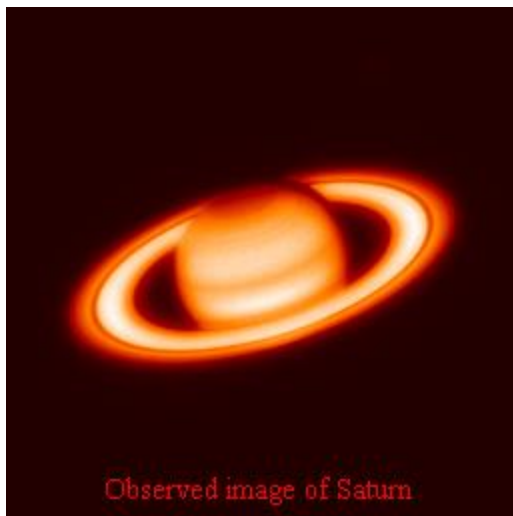
- The Story of Hubble Space Telescope (HST)
 - HST Cost at Launch (1990): \$1.5 billion
 - Main mirror imperfections due to human errors
 - Got repaired in 1993



Restoration of HST Images



Another Example





The Real (Optical) Solution



Before the repair



After the repair





Application (II): Law Enforcement



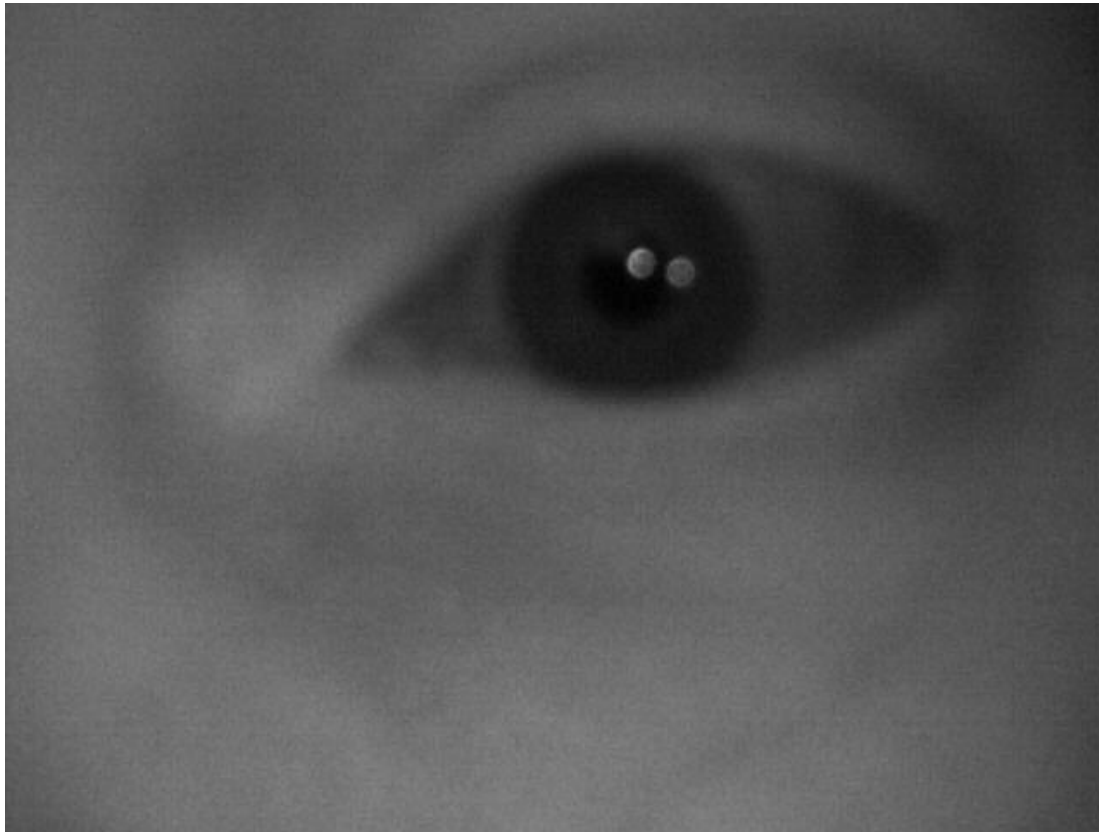
Motion-blurred license plate image



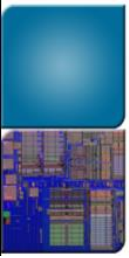
Restoration Example



Application into Biometrics



out-of-focus iris image



Questions?



Homework

- Try to remove this periodic noise (using Matlab) ppt. 57