2110433 Computer Vision

Image Filtering

Thanarat H. Chalidabhongse
Chula PIC Lab,
Dept. of Computer Engineering
Chulalongkorn University

Today's Outline

- Filtering
- Convolution
  - Lab 2

Image as a function

- "ภาพ" (Image) ⇨ ฟังก์ชันสองมิติ $f(x,y)$ โดยที่
  - x และ y เป็นพิกัดเชิงพื้นที่ (spatial coordinates) และ
  - ขนาดของฟังกชัน f คือ "ความเข้มของจุดภาพ" (intensity) หรือ gray level ของจุด x,y นี้
- "ภาพดิจิทัล" (Digital Image) ⇨ x,y,f are finite and discrete

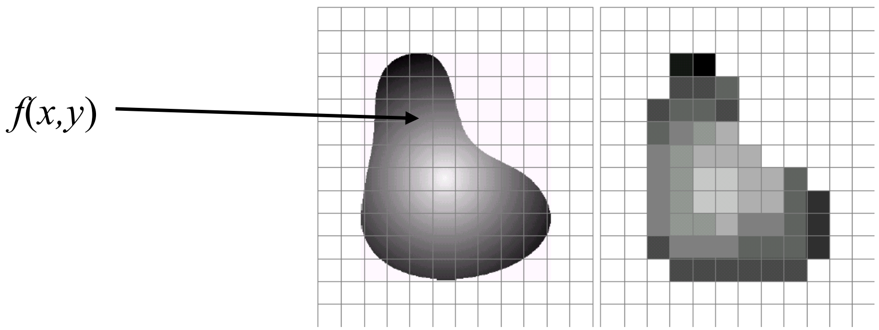
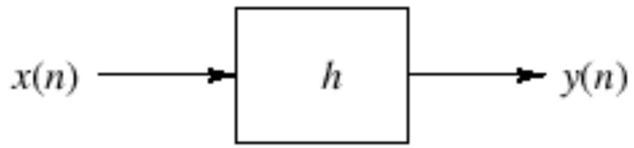


Image filtering

- Linear transforms that change the frequency contents of signals.



- Example: two point moving average
  - $y(n) = [x(n)+x(n\text{-}1)] / 2$
- A useful way to view filtering is by convolution

Convolution

$$G = H * F$$

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

- โดยที่ $H(u,v)$ คือ ตัวพราง (mask/kernel) ขนาด $(2k+1 \text{ x } 2k+1)$

  $F(i,j)$ คือ ภาพนำเข้า

Kernel/Mask/Probe

- A kernel is a set of pixel positions and corresponding values called weights.
- Each kernel has an origin.

1	1	1
1	**1**	1
1	1	1

1	2	1
2	**4**	2
1	2	1

1
1
1

- Applying a kernel on a R x C image (F) yields a R x C output image (G).

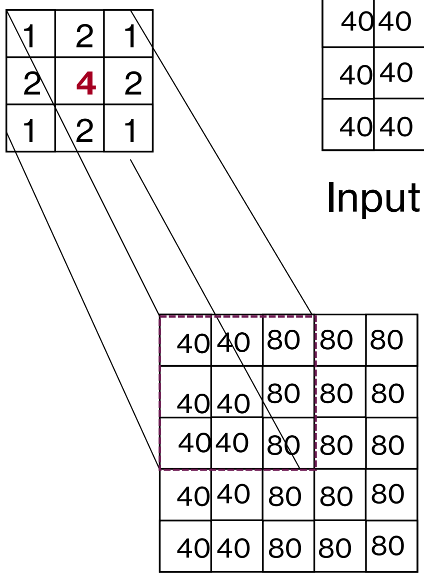
Applying kernels to Images

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

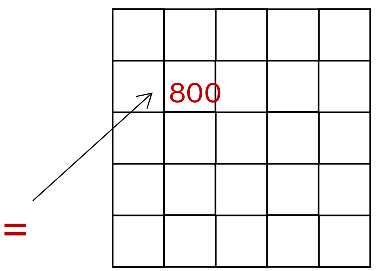
Input image ***F***

1	2	1
2	**4**	2
1	2	1

Kernel



Apply kernel to every pixel,
e.g., at *F*(0,0)
G(0,0) = 40*1 + 40*2 + 80*1 +
40*2 + 40*4 + 80*2 +
40*1 + 40*2 + 80*1 =
800



Temporary image *G*

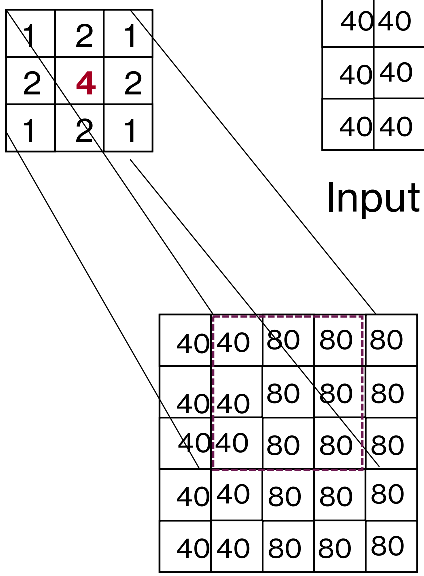
Applying kernels to Images

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

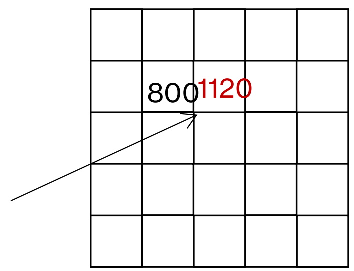
Input image ***F***

1	2	1
2	**4**	2
1	2	1

Kernel

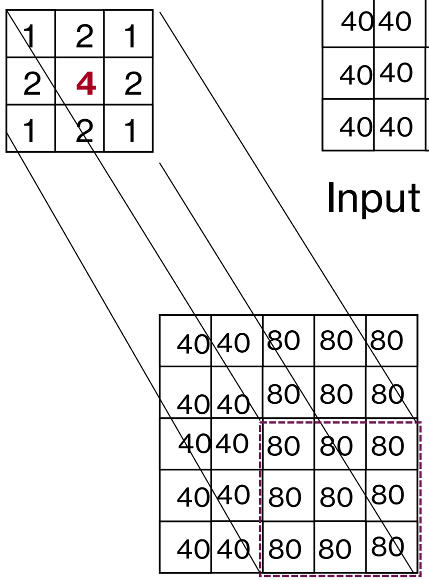


Apply kernel to every pixel,
e.g., at $F(0,1)$
$G(0,1) = 40*1 + 80*2 + 40*1 + 40*2 + 80*4 + 40*2 + 40*1 + 80*2 + 40*1 =$
1120



Temporary image *G*

Applying kernels to Images



40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

Input image ***F***

1	2	1
2	**4**	2
1	2	1

Kernel

Apply kernel to every pixel, you get

	800	1120	1280	
	800	1120	1280	
	800	1120	1280	

Temporary image *G*

Applying kernels to Images with padding

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

Input image ***F***

1	2	1
2	**4**	2
1	2	1

Kernel

40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80

Virtual image

1. Create a virtual image by expand top&bottom rows and left&right columns

Applying kernels to Images with **padding**

1	2	1
2	**4**	2
1	2	1

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

Input image ***F***

1	2	1
2	**4**	2
1	2	1

Kernel

40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80

Virtual image

2. Apply kernel to every pixel, e.g., at $F(0,0)$

$G(0,0)$ =

40*1 + 40*2 + 40*1 + 40*2 + 40*4 + 40*2 + 40*1 + 40*2 + 40*1 = **640**

640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280

Temporary image *G*

Applying kernels to Images with padding

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

Input image ***F***

1	2	1
2	**4**	2
1	2	1

Kernel

640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280

Temporary image ***G***

3. Normalize the image *G* by dividing by sum of the weights (16) to obtain *G'*

40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80

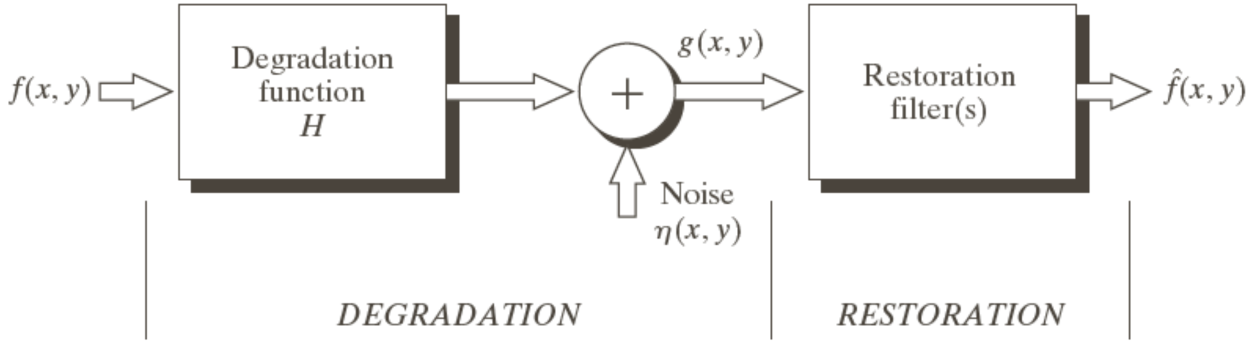
Normalized Output image ***G'***

การกรองภาพ (Image filtering)

- ภาพที่ได้จากอุปกรณ์ถ่ายภาพ มักมีสัญญาณรบกวน (image noise) ปนเปื้อนอยู่
  - จากกระบวนการถ่ายภาพ
  - จากกระบวนการส่งผ่านสัญญาณภาพ
  - ฯลฯ
- ดังนั้นในหลายกรณีเราจำเป็นต้องขจัดสัญญาณรบกวนเหล่านี้ออกเพื่อปรับปรุงคุณภาพของภาพ ก่อนที่จะนำภาพไปประมวลในขั้นตอนอื่นต่อไป

Image Degradation/Restoration Process

FIGURE 5.1
A model of the image degradation/ restoration process.



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

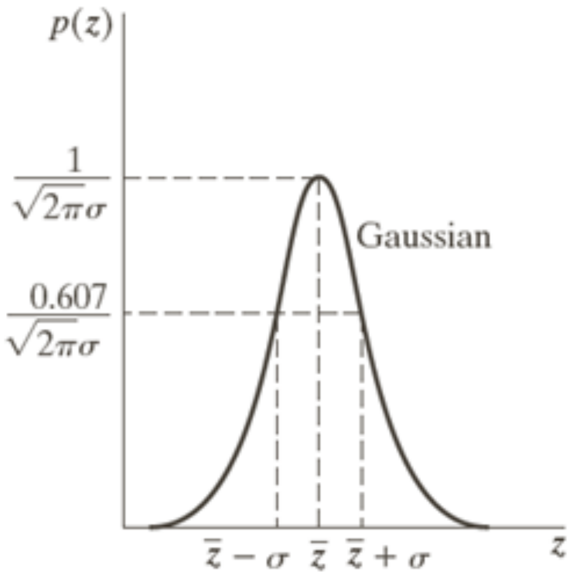
$$g(x,y) = h(x,y) \star f(x,y) + \eta(x,y)$$

Noise Models (1)

- Sources of noise in digital images arise during image acquisition and/or transmission.
- In CCD camera, light level and sensor temperature are major factors affecting the amount of noise in the resulting image.
- Image might be corrupted during transmission due to interference.

Gaussian Noise

- Gaussian (normal) noise models are used frequently in practice.
- The PDF of Gaussian ranc
given by



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

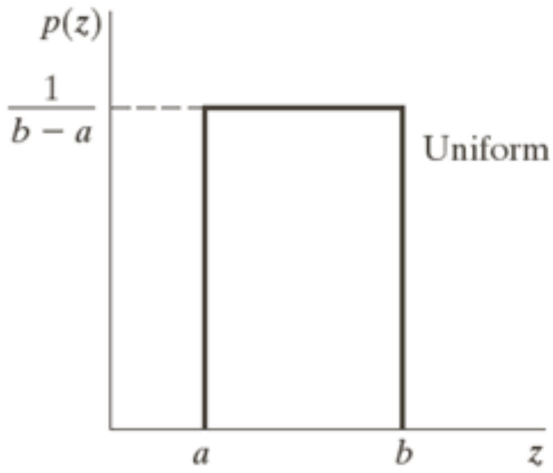
Uniform Noise

- The PDF of uniform noise is given by

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

$$\bar{z} = \frac{a+b}{2}$$

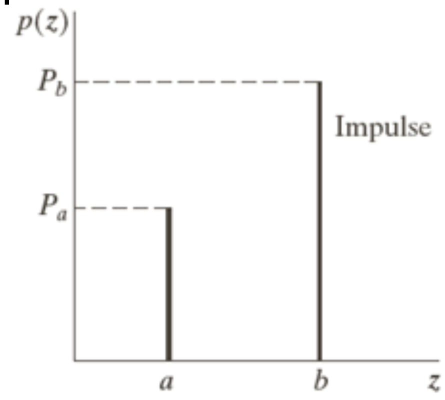
$$\sigma^2 = \frac{(b-a)^2}{12}$$



Impulse (Salt-and-Pepper) Noise

- The PDF of impulse (*bipolar*) 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 $b>a$, intensity b will appear as a white dot in the image.
- If either P_a or P_b is zero, the impulse noise is called *unipolar*

Test Pattern

- Composed of simple, constant areas span the grey scale from black to near white in 3 increments.

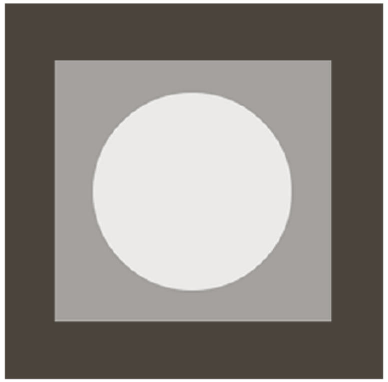
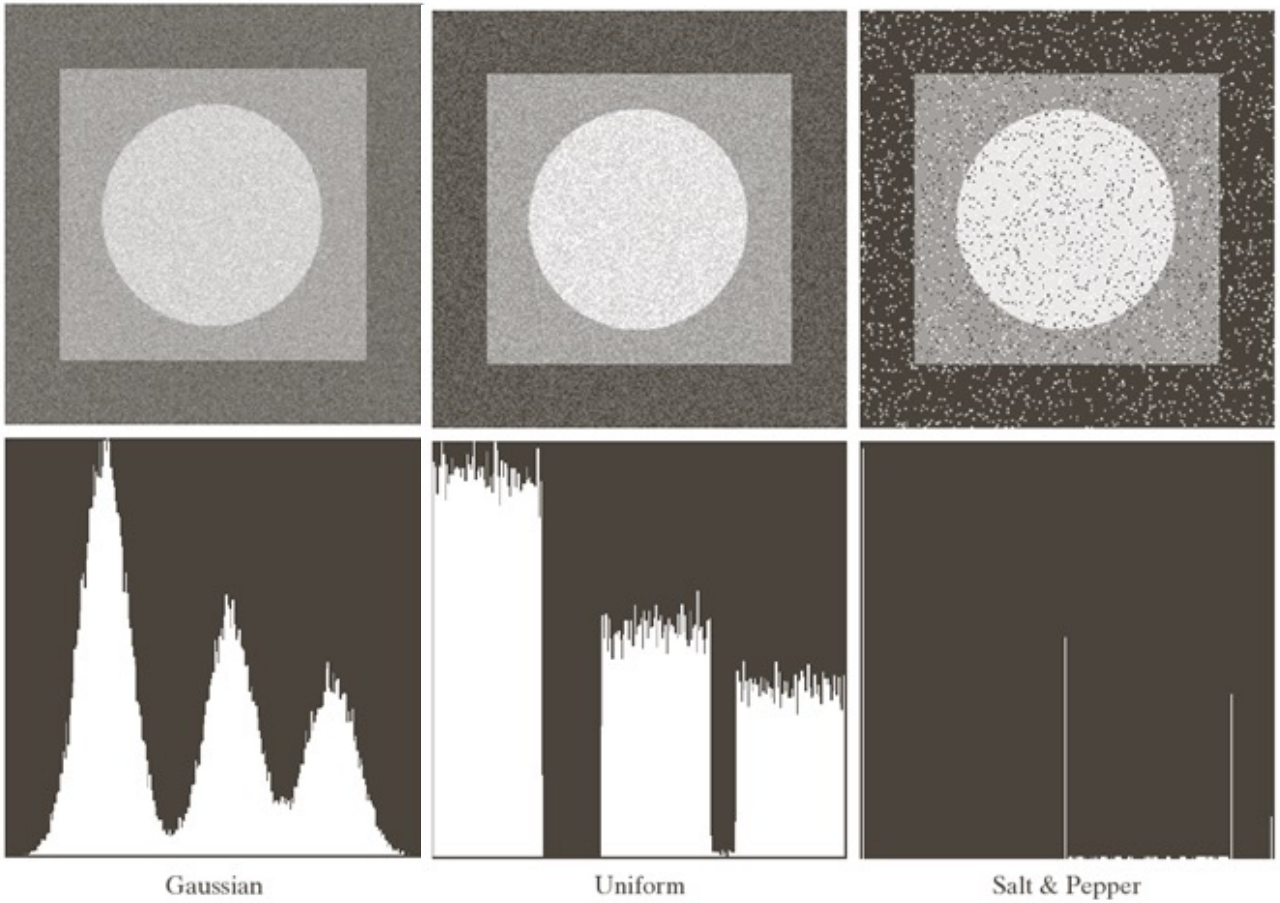


FIGURE 5.3 Test pattern used to illustrate the characteristics of the noise PDFs shown in Fig. 5.2.

Test Pattern with Noises



Restoration – Spatial Filtering

- When only degradation present in an image is noise.

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

- The noise terms are unknown, so subtracting them from g(x,y) is not a realistic option.
- Spatial filter is a method of choice in situation when only additive random noise is present.

Mean Filters

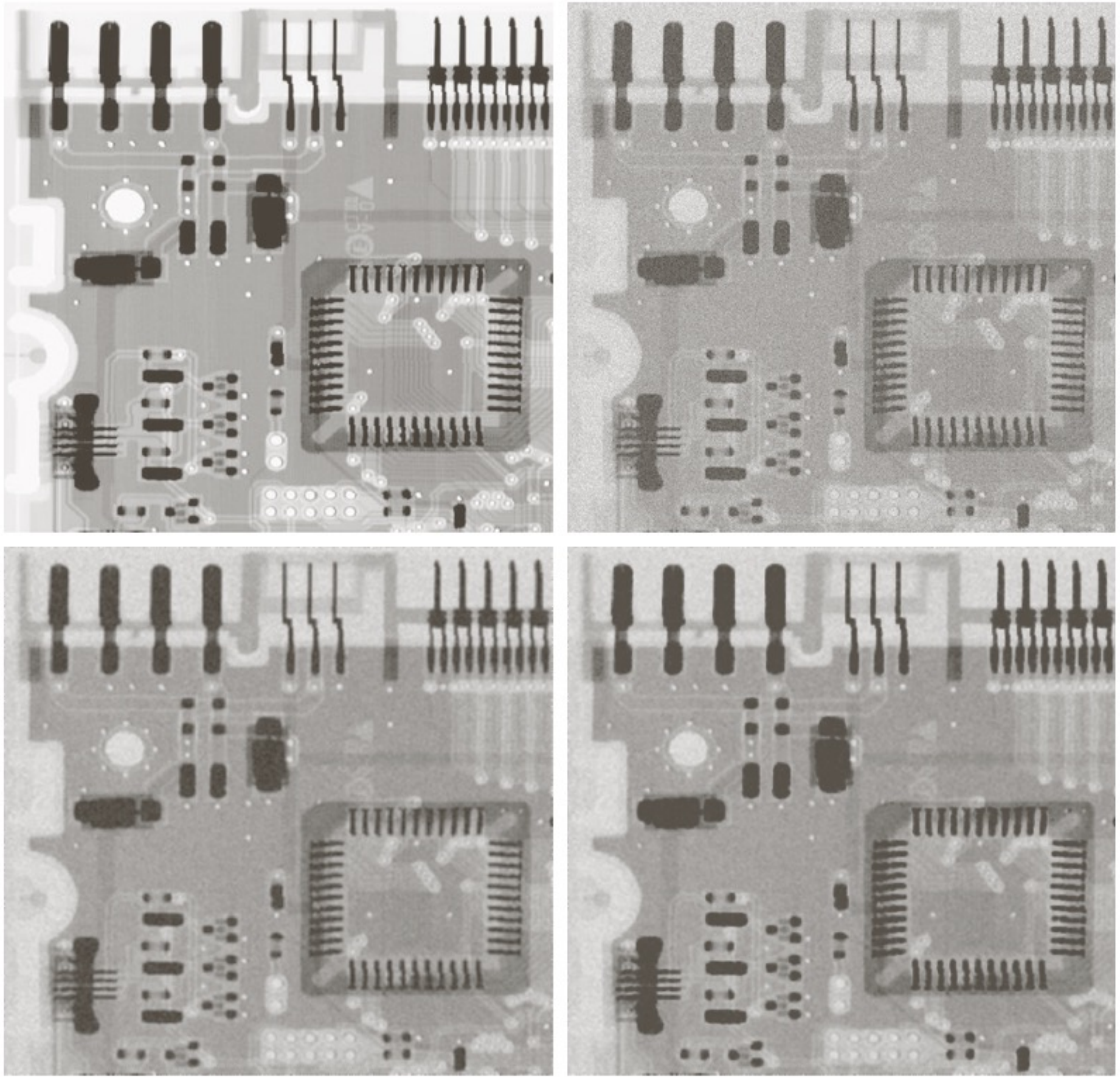
- Arithmetic mean filter: $$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}} g(s,t)$$
- Geometric mean filter: $$\hat{f}(x,y)=\left[\prod_{(s,t)\in S_{xy}} g(s,t)\right]^{\frac{1}{mn}}$$
- Harmonic mean filter: $$\hat{f}(x,y)=\frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

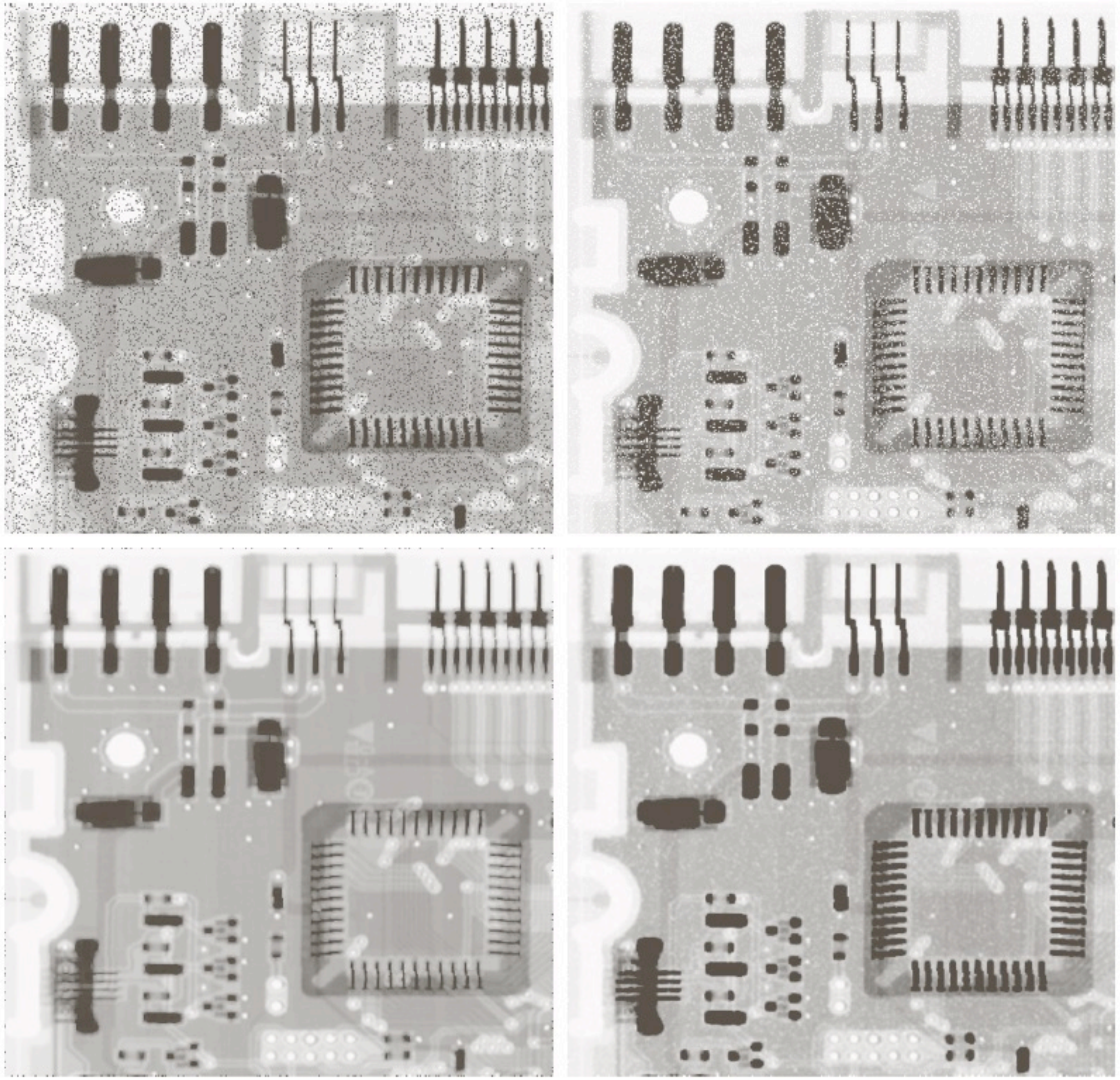
  Good for salt noise, not for pepper noise
- Contraharmonic mean filter: $$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

  Q+ good for pepper noise,
  Q- good for salt noise

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.)





a b
c d

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

Order-Statistic Filters

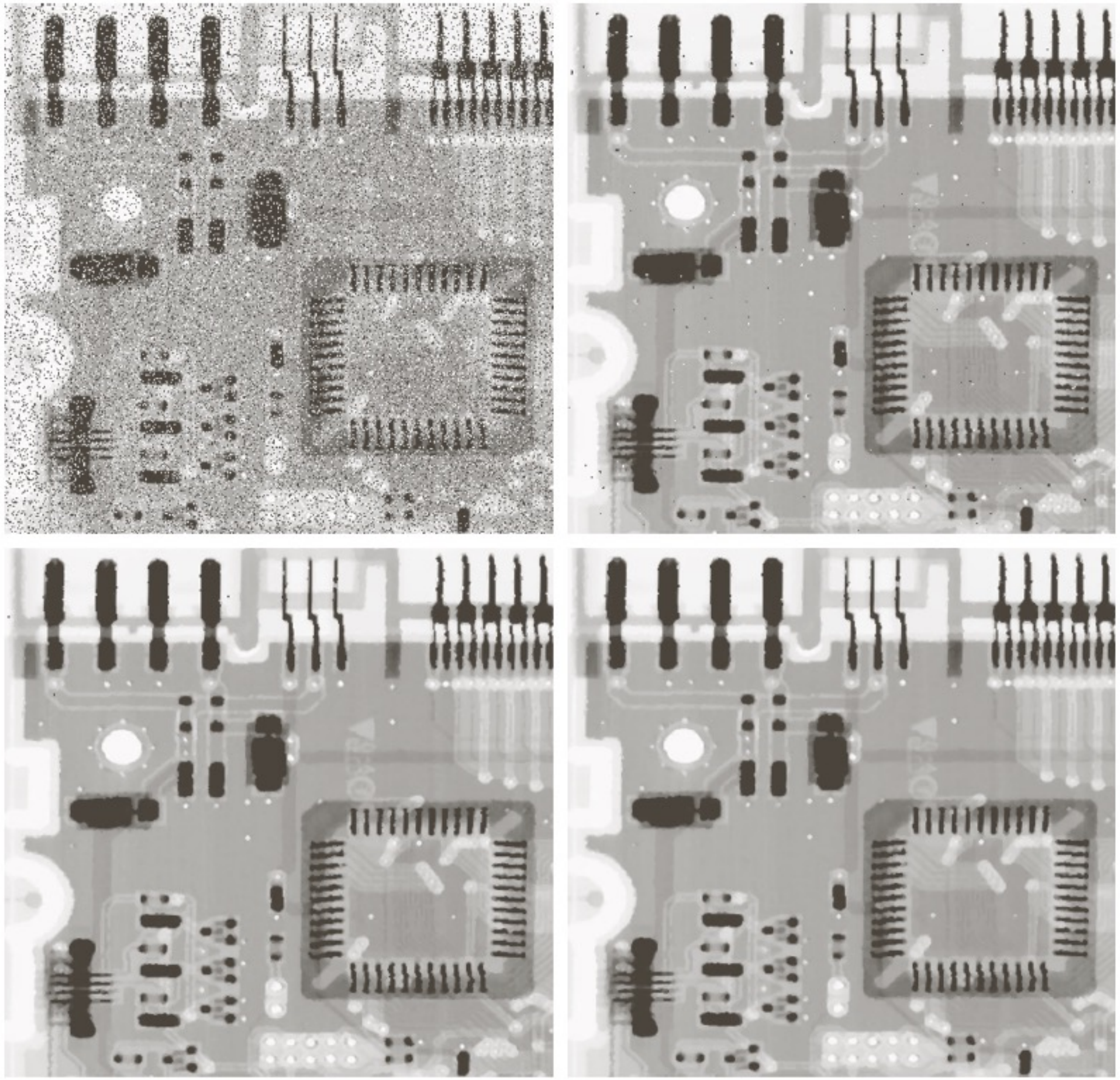
- Median filter: $$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\}$$

- Max and Min filter: $$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\}, \quad \min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

  Good for salt/pepper noise

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

- Alpha-trimmed mean filter: $$\hat{f}(x,y) = \frac{1}{mn-d}\sum_{(s,t)\in S_{xy}} g_r(s,t)$$



a b
c d

FIGURE 5.10
(a) Image corrupted by salt-and-pepper noise with probabilities $P_a = P_b = 0.1$.
(b) Result of one pass with a median filter of size 3×3.
(c) Result of processing (b) with this filter.
(d) Result of processing (c) with the same filter.

a b

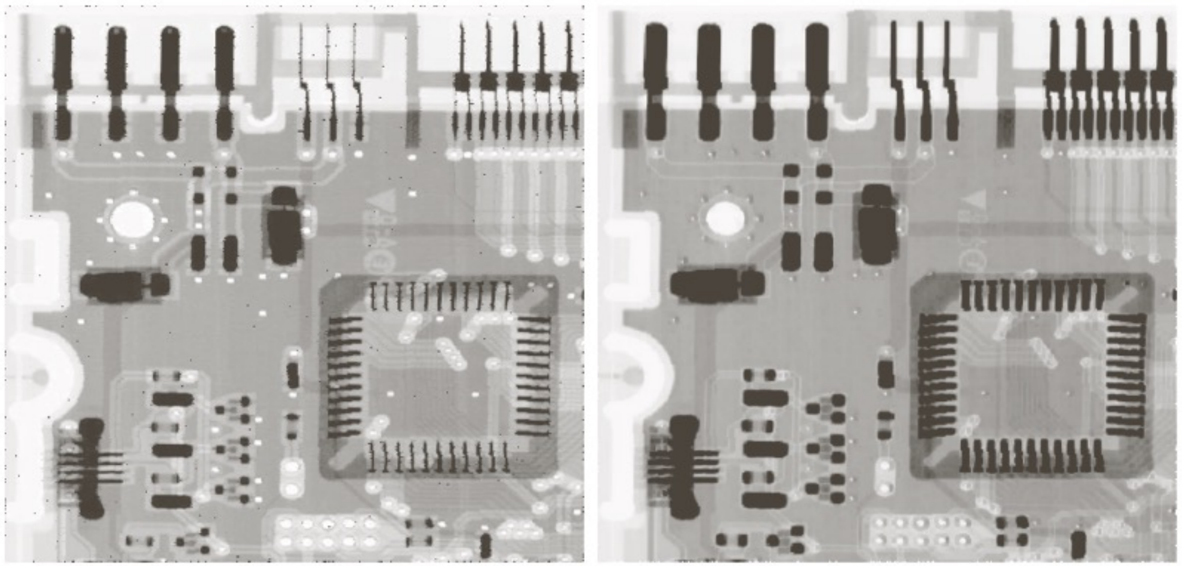
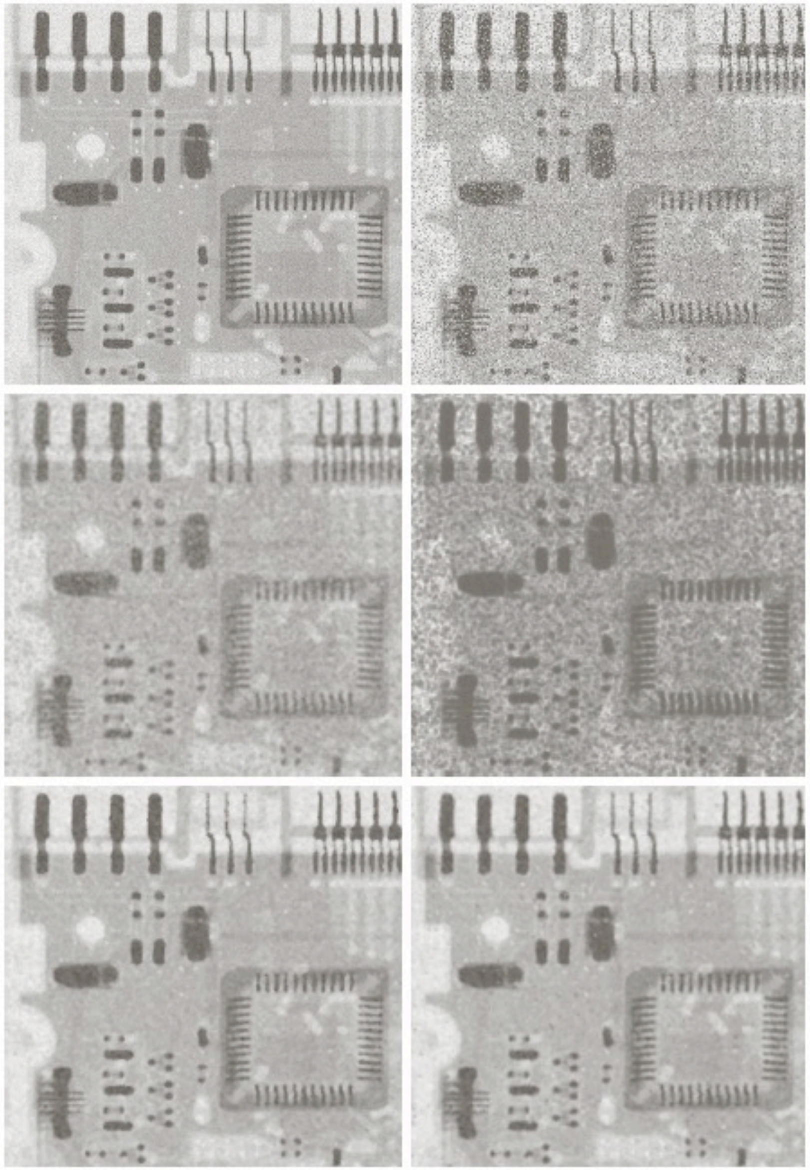


FIGURE 5.11 (a) Result of filtering Fig. 5.8(a) with a max filter of size 3×3. (b) Result of filtering 5.8(b) with a min filter of the same size.

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 $d = 5$.



Gaussian Smoothing

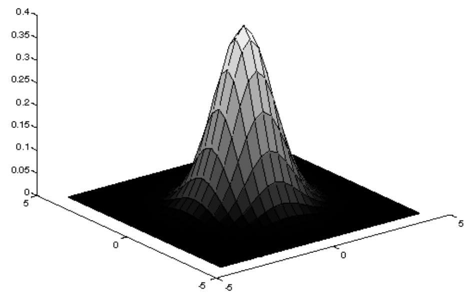
- Gaussian filters are a class of linear smoothing filters with the weights chosen according to the shape of a Gaussian function
- Very effective for removing Gaussian noise

Gaussian Smoothing

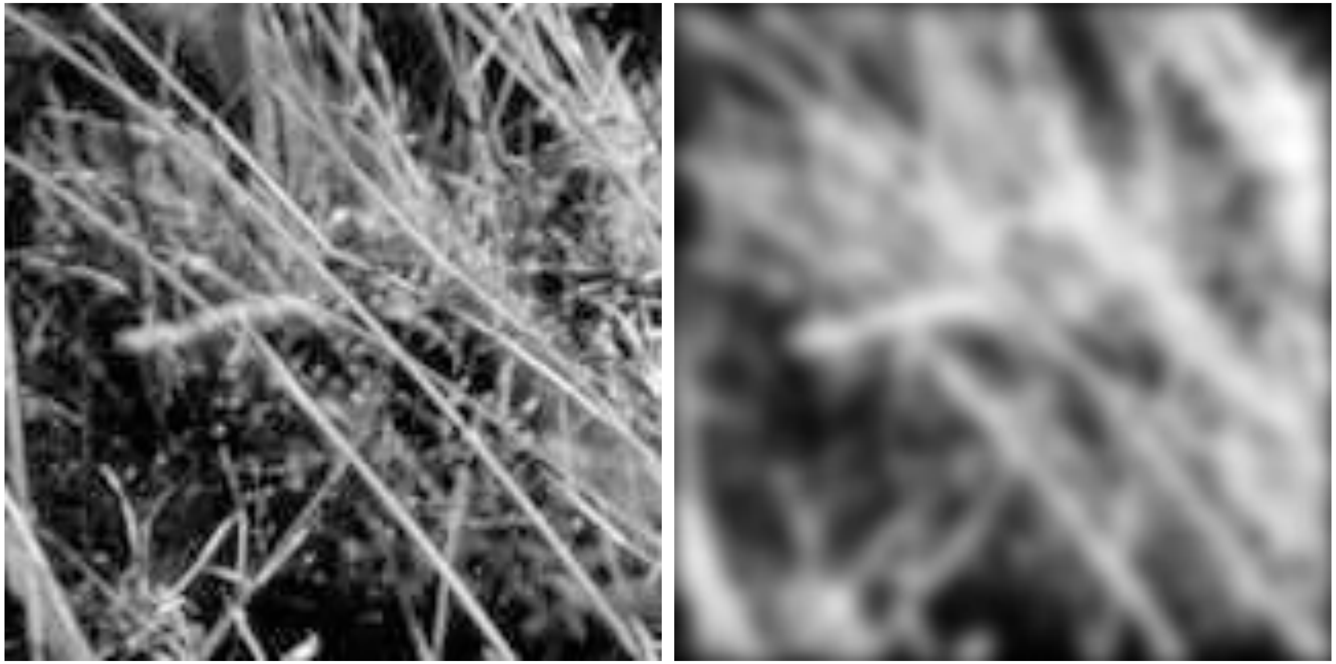
- Zero-mean Gaussian function in one dimension:
  - $g(x) = e^{\wedge}(-x^2/2\sigma^2)$

  where the Gaussian spread parameter σ determines the width of the Gaussian

- Two-dimensional zero-mean discrete Gaussian function:
  - $g(i,j) = e^{\wedge}(-(i^2+j^2)/2\sigma^2)$



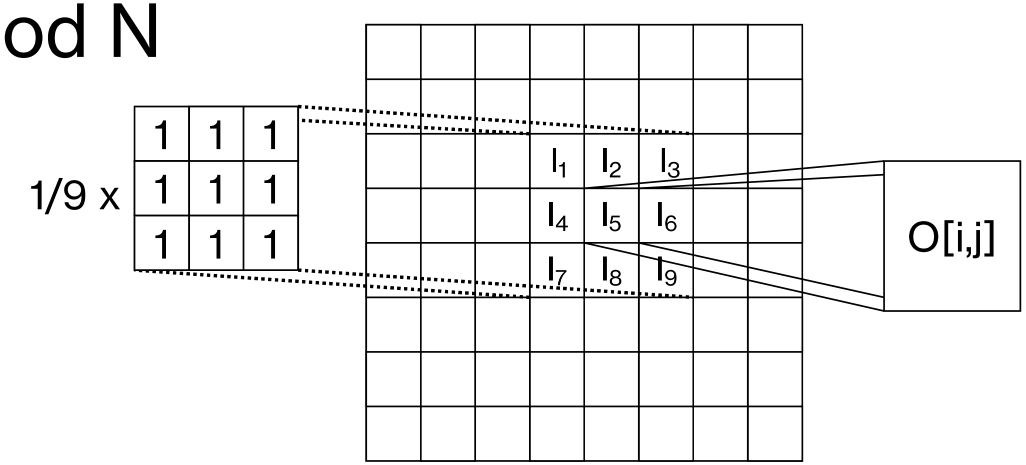
Gaussian Smoothing Example



Using Mask for filtering

- Local averaging operation:
  - $\mathbf{O}[i,j] = 1/M \sum_{(k,l) \text{ in } N} \mathbf{I}[k,l]$

where M is the number of pixels in the neighborhood N



Mean Filter Example

Effects of smoothing as a function of filter size

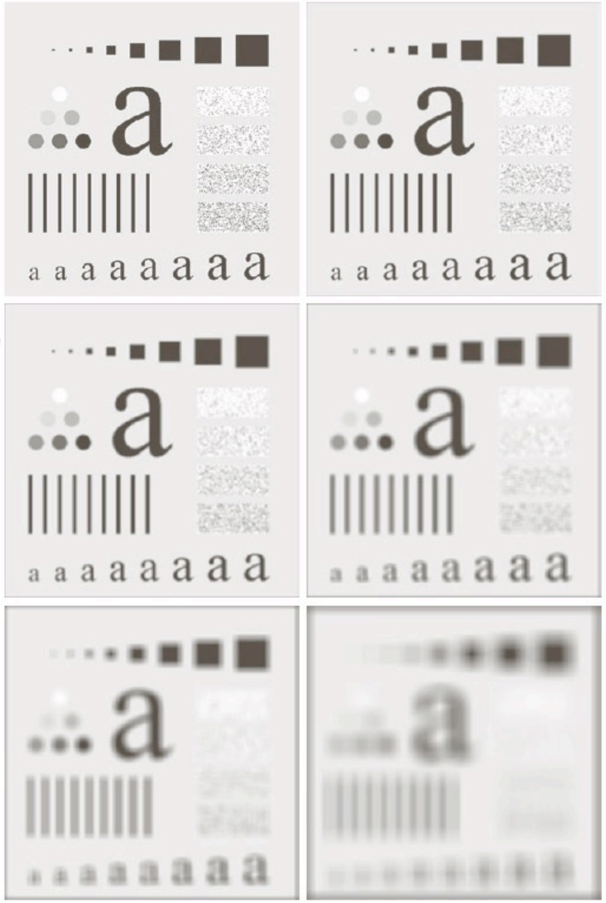


FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

a b
c d
e f

Discrete Gaussian Filter

- Compute the mask directly from the discrete Gaussian distribution

1	4	7	10	7	4	1
4	12	26	33	26	12	4
7	26	55	71	55	26	7
10	33	71	91	71	33	10
7	26	55	71	55	26	7
4	12	26	33	26	12	4
1	4	7	10	7	4	1

7x7 Gaussian mask

$\sigma^2 = 2$, n=7

Discrete Gaussian Filter

- Approximation is provided by coefficients of the binomial expansion
  - $(1+x)^n = C(n,0) + C(n,1)x + C(n,2)x^2 + \ldots + C(n,n)x^n$
- Work well for filter size up to around n=10
- Large Gaussian filters can be implemented by repeatedly applying a smaller Gaussian filter


Q
A

Lab-02

TODO:
Assignments : HW2
Term Project : Team forming & pre-proposal