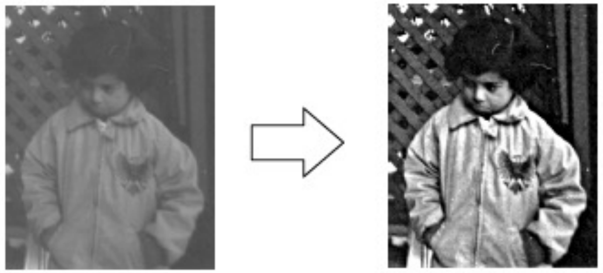
Image Enhancement

Prof. Dr. M. Elif Karslıgil

Yildiz Technical University

Computer Engineering Department

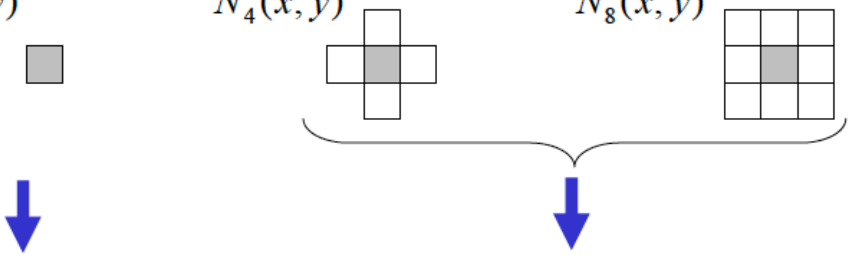
Spatial Domain Processing



$$f \xrightarrow{T_N(.)} g = T_N(f) \qquad f(x,y)\ ,\ 1 \le x \le M, 1 \le y \le N \qquad g(x,y)\ ,\ 1 \le x \le M, 1 \le y \le N$$

$T_N(.)$: Spatial operator defined on a neighborhood *N* of a given pixel

$N_0(x,y)$ $N_4(x,y)$ $N_8(x,y)$



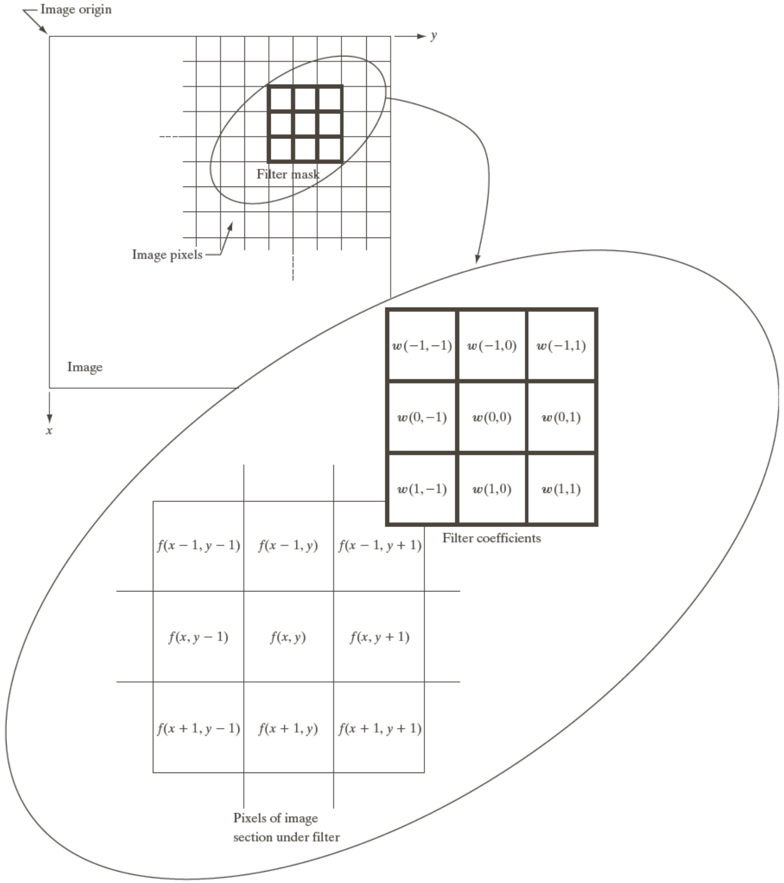
point processing

mask/kernel processing

Hard to tell anything from a single pixel

Example: you see a reddish pixel. Is this the object's color? Illumination? Noise?

Area(Mask) Processing Methods



w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Mask – kernel - window

Area(Mask) Processing Methods

- A pixel's value is computed from its old value and the values of pixels in its vicinity.

- More costly operations than simple point processes, but more powerful.

- **What is a Mask?**

- A mask is a small matrix whose values are called *weights*.

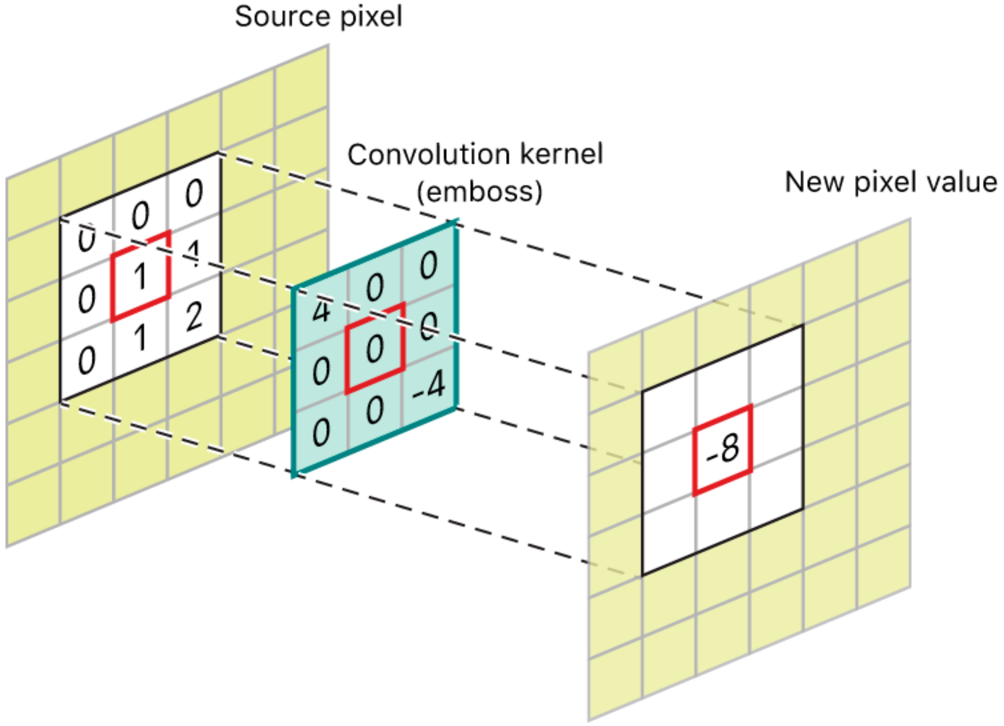
- Each mask has an *origin*, which is usually one of its positions.

- The origins of symmetric masks are usually their center pixel position.

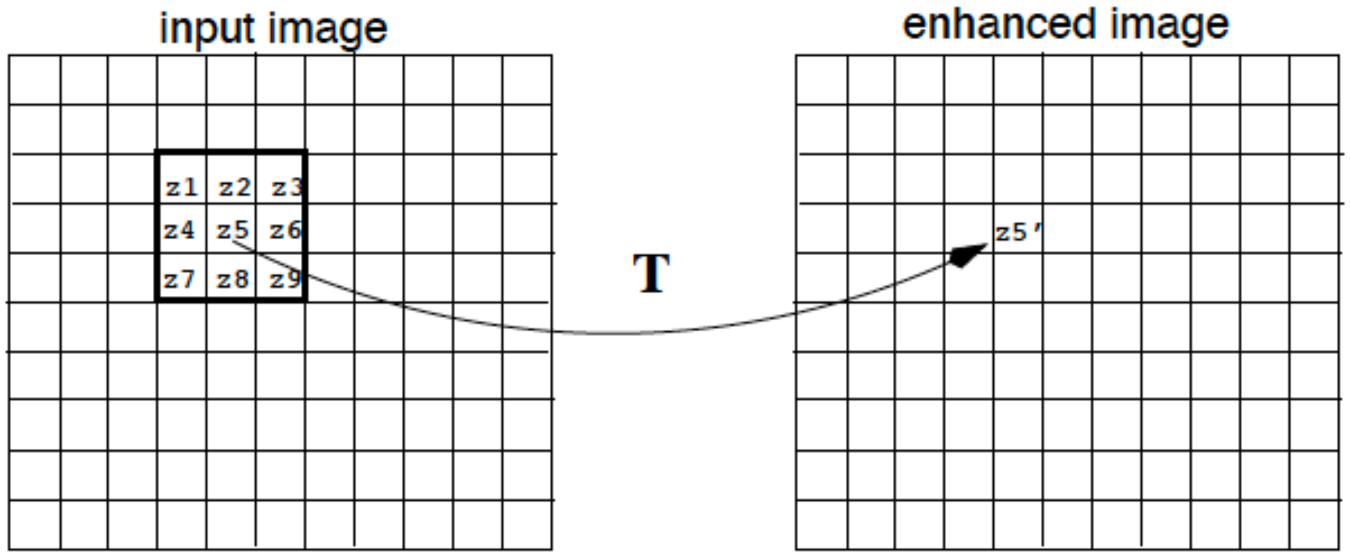
- For nonsymmetric masks, any pixel location may be chosen as the origin (depending on the intended use).

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \qquad \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} \qquad \begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \end{array}$$

What is Convolution ?



Applying mask to image



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

T operates on a neighborhood of pixels

$$z5' = R = w1z1 + w2z2 + \dots + z9w9$$

Applying mask to image

- The application of a mask to an input image produces an output image of the same size as the input.

Convolution

(1) For each pixel in the input image, the mask is conceptually placed on top of the image with its origin lying on that pixel.

(2) The values of each input image pixel under the mask are multiplied by the values of the corresponding mask weights.

(3) The results are summed together to yield a single output value that is placed in the output image at the location of the pixel being processed on the input.

Applying mask to image

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	1	0
0	2	1

Kernel

	16	

Modified image data

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_{mn} z_{mn} = \sum_{i=1}^{mn} w_i z_i$$

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Where the w's are mask coefficients, the z's are the value of the image gray levels corresponding to those coefficients

Smoothing(low pass) Spatial Filters

- Smoothing filters are used for blurring and for noise reduction.
- Blurring is used in preprocessing steps, such as removal of small details from an image prior to object extraction, and bridging of small gaps in lines or curves.
- Noise reduction can be accomplishing by blurring with a linear filter and also by nonlinear filtering.

Types of Smoothing Filters

- There are 2 way of smoothing spatial filters
  - Smoothing Linear Filters
  - Order-Statistics Filters

Averaging filter

- The key requirement is that all coefficients are positive.
- Neighborhood averaging is a special case of LPF where all coefficients are equal.
- It blurs edges and other sharp details in the image.
- Example: $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Averaging filter

$$\frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Standard average

$$\frac{1}{16}\times\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Weighted average

Averaging filter

$$F(x,y) \quad * \quad H(u,v) \quad = \quad G\,(x,y)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$* \ \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"box filter"

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$G = F * H$$

Filter example #1: Moving Average

$f[n,m]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[n,m]$

	0	10							

Filter example #1: Moving Average

$f[n,m]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[n,m]$

	0	10	20						

Filter example #1: Moving Average

$f[n, m]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[n, m]$

	0	10	20	30					

Filter example #1: Moving Average

$f[n,m]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[n,m]$

	0	10	20	30	30				

Filter example #1: Moving Average

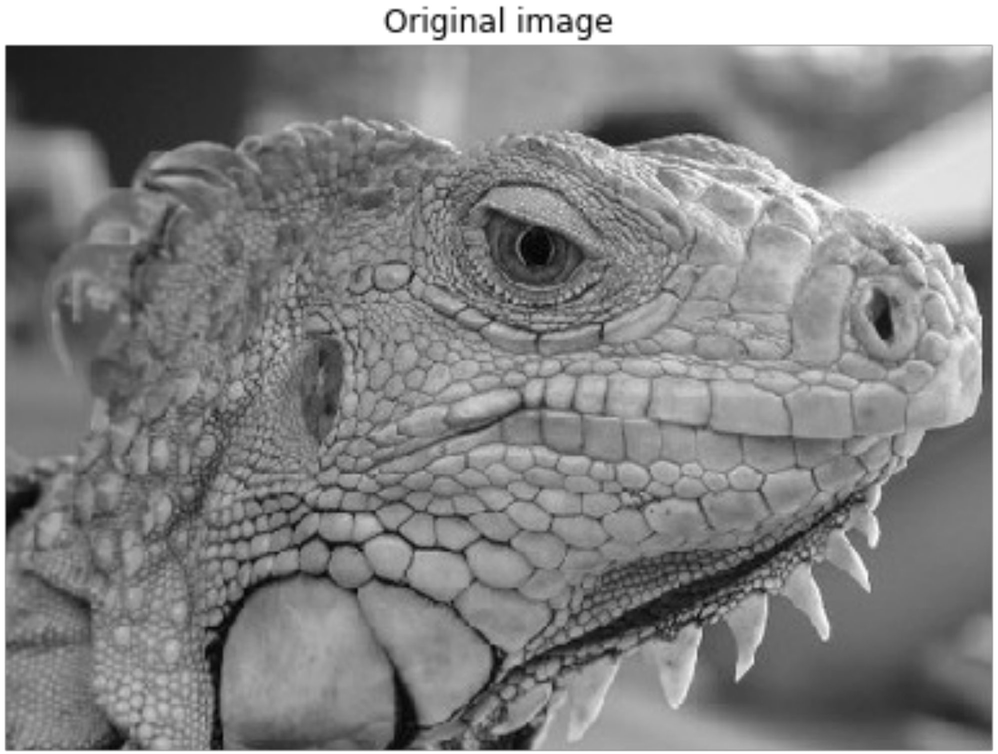
$f[n, m]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

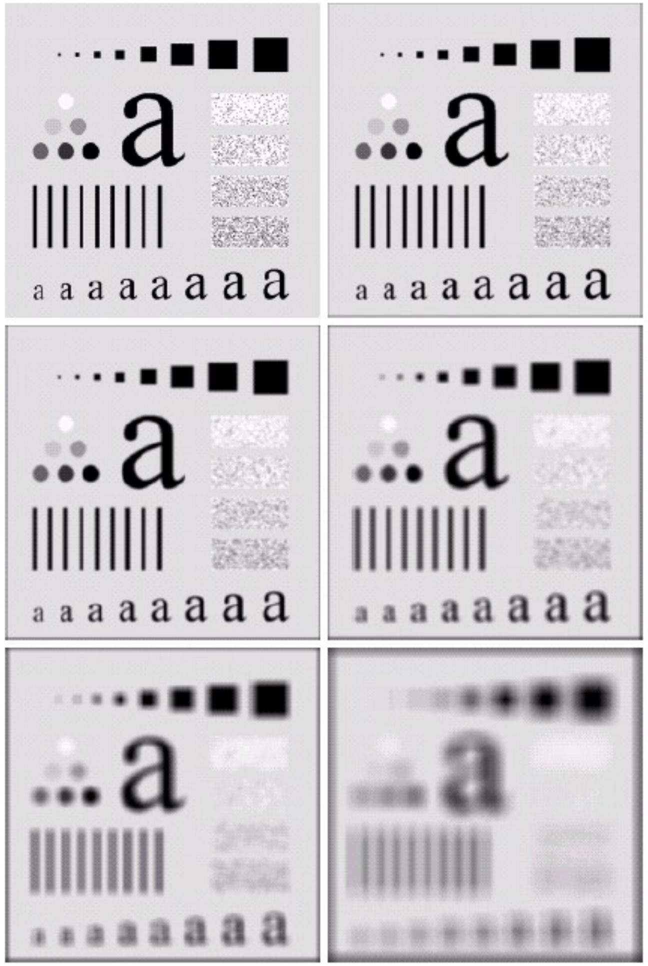
$g[n, m]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Filter example #1: Moving Average



Averaging Filter Resuluts for Different Sizes



original

3x3

5x5

9x9

15x15

35x35

Averaging Filter Resuluts for Different Sizes

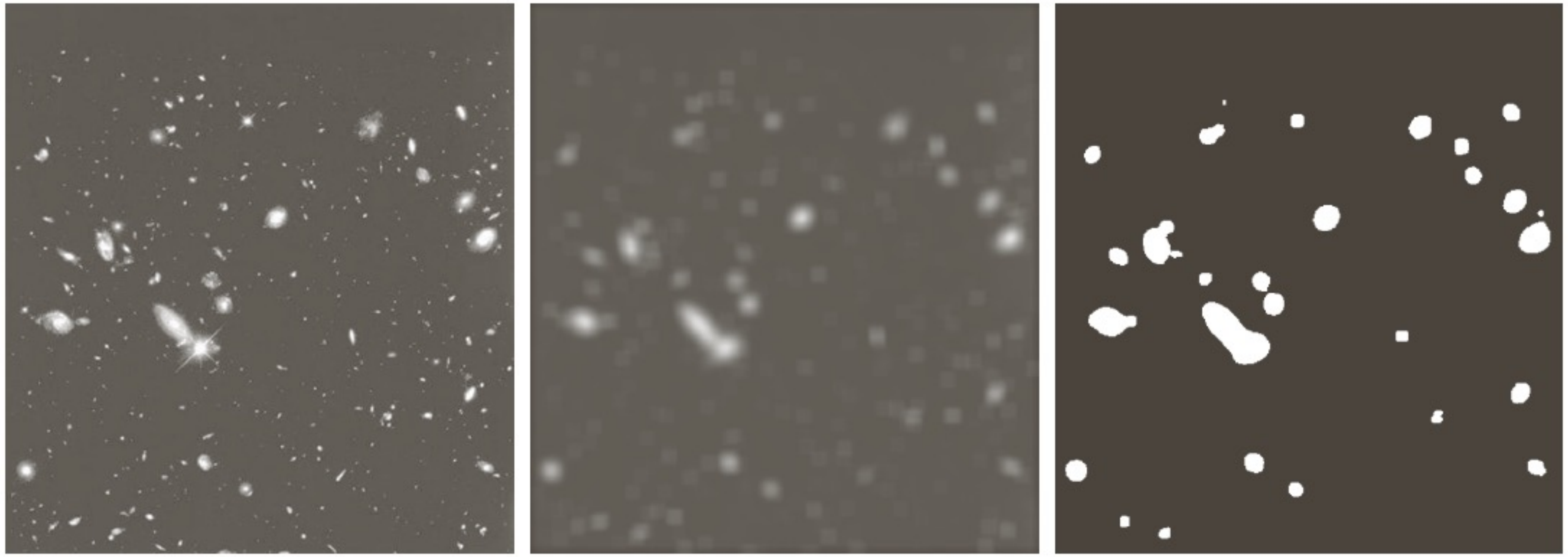
Original Image

[3x3]

[5x5]

[7x7]

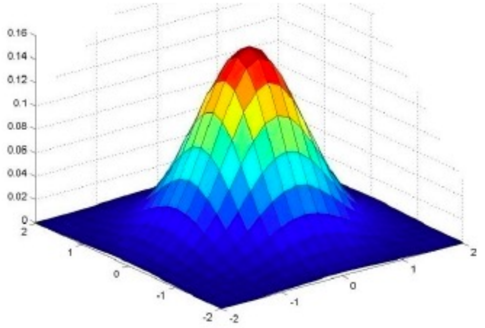
Example

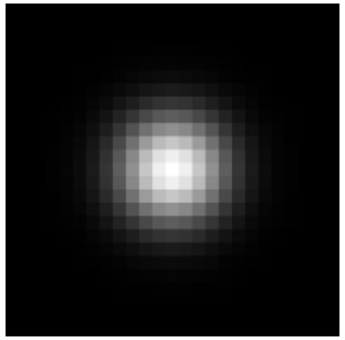


a b c

FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Gaussian Smoothing Filter





$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

1-D Gaussian Function

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

2-D Gaussian Function

$$G(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Calculating Gaussian Convolution Kernels

- For a ***5×5*** kernel and σ = 1

$$G(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Rows ↓ \ Columns →	-2	-1	0	1	2
-2	0.003	0.013	0.022	0.013	0.003
-1	0.013	0.059	0.097	0.059	0.013
0	0.022	0.097	0.159	0.097	0.022
1	0.013	0.059	0.097	0.059	0.013
2	0.003	0.013	0.022	0.013	0.003

Dividing by 0.003 →

1	4	7	4	1
4	20	33	20	4
7	33	55	33	7
4	20	33	20	4
1	4	7	4	1

Calculating Gaussian Convolution Kernels

$$\begin{pmatrix} 0.00509023590810233 84 & 0.005175071225695336 4 & 0.00509023590810233 84 \\ 0.005175071225695336 4 & 0.005261320432789928 8 & 0.005175071225695336 4 \\ 0.00509023590810233 84 & 0.005175071225695336 4 & 0.00509023590810233 84 \end{pmatrix}$$

- Each item should be multiplied by ***1.0 / 0.005090***
- For normalize divide by total of all values in kernel

Gaussian Filters

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3x3

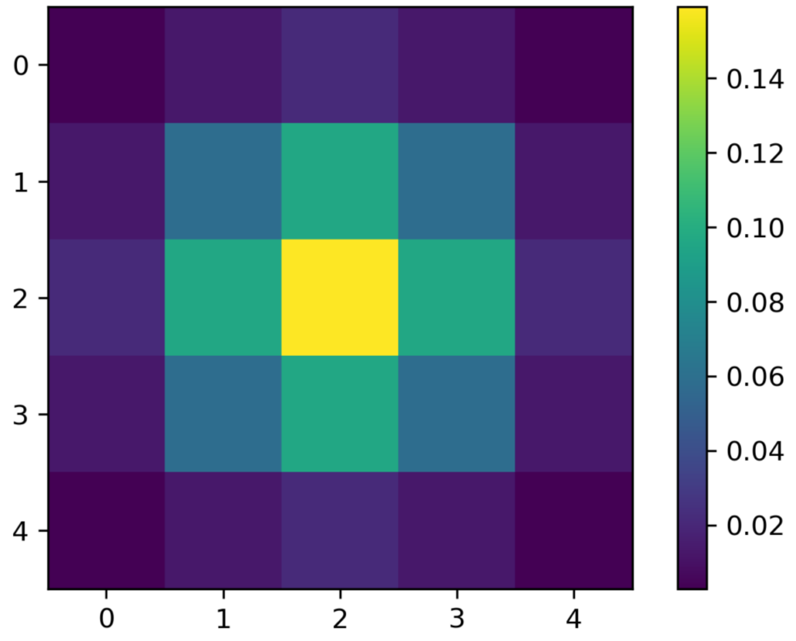
$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

5x5 Sigma=0.1

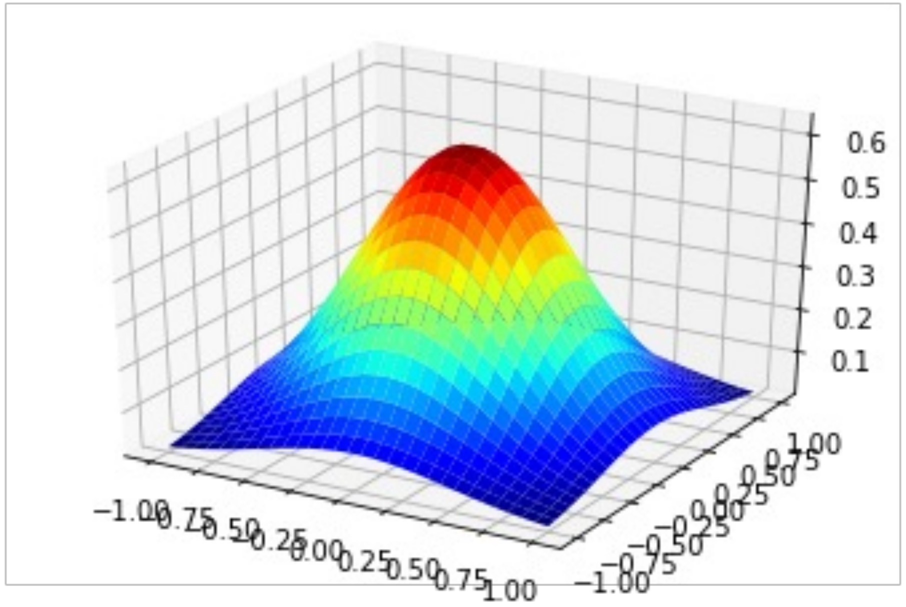
Smoothing with a Gaussian

7 x 7 gaussian filter

Gaussian Function

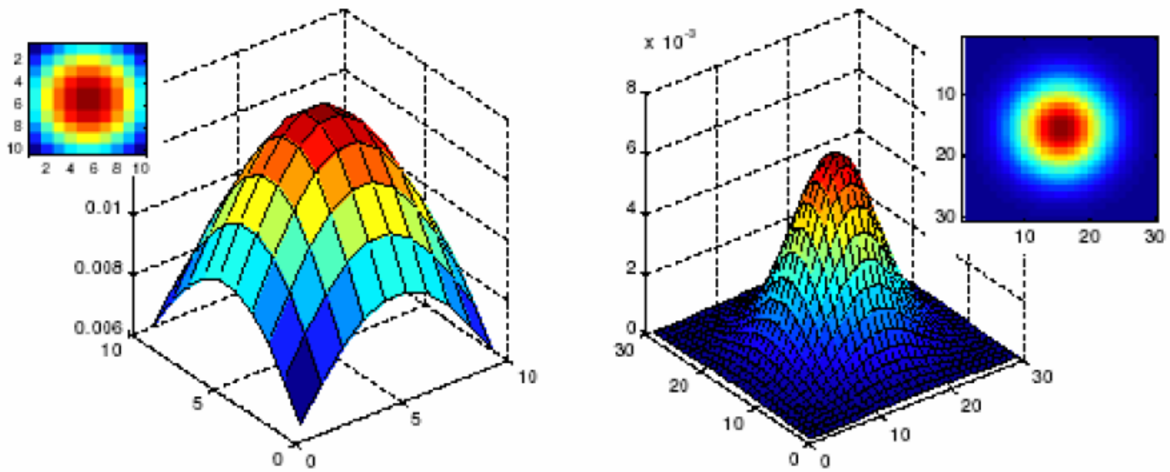


2-d visualization of a Gaussian function



3-d visualization of a Gaussian function

Choosing Kernel Width



σ = 5 with 10x10 kernel

σ = 5 with 30x30 kernel

Gaussian Filters

Ladybird: Gaussian Kernel 5×5 Weight 5.5

Ladybird: Gaussian Kernel 13×13 Weight 9.5

Gaussian Filters

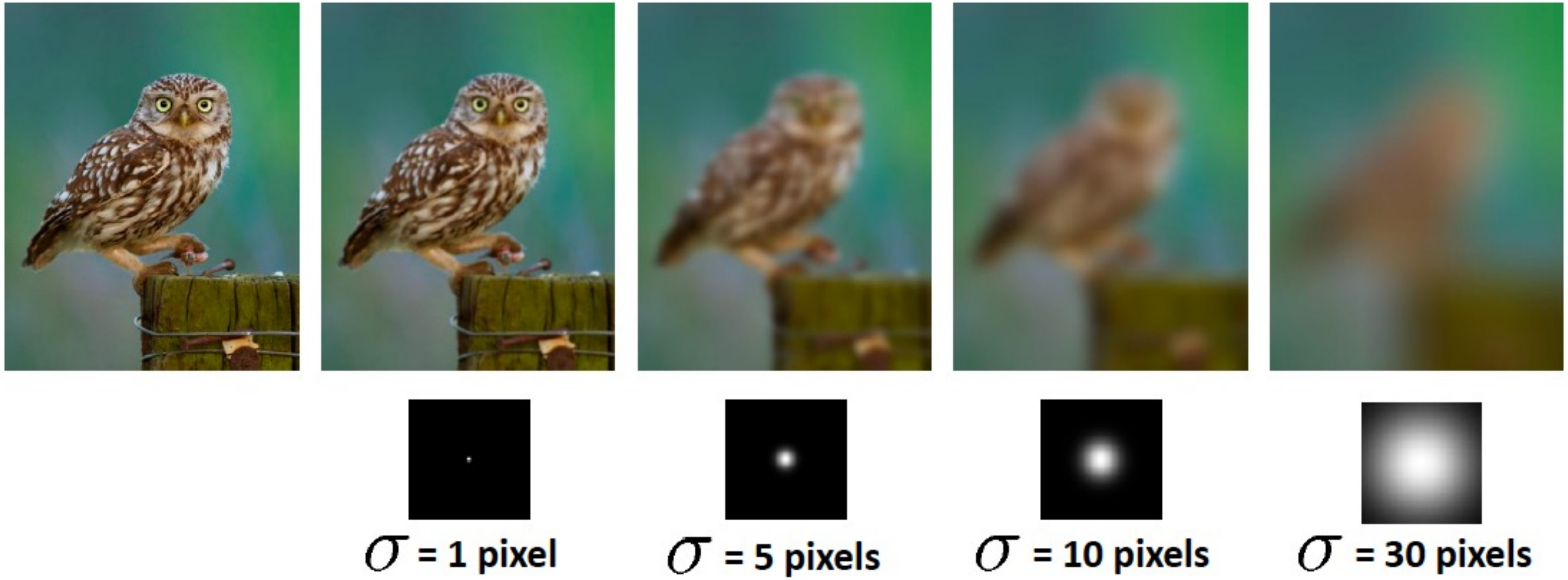
Original image

$\sigma = 2$ pixels

$\sigma = 4$ pixels

$\sigma = 8$ pixels

Gaussian Filters



Smoothing with a Gaussian

7x7 filter

Smoothing with a non Gaussian

7x7 filter

Order-statistic (Nonlinear) Filters

— Nonlinear

— Based on ordering (ranking) the pixels contained in the filter mask

— Replacing the value of the center pixel with the value determined by the ranking result

E.g., median filter, max filter, min filter

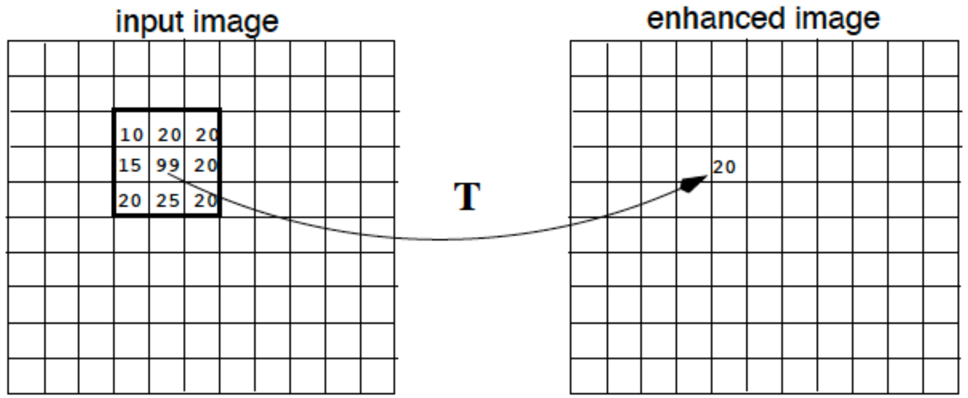
Median Filter

- If the objective is to achieve noise reduction instead of blurring, this method should be used.
- This method is particularly effective when the noise pattern consists of strong, spike-like components and the characteristic to be preserved is edge sharpness.
- It is a nonlinear operation.
- For each input pixel $f(x,y)$, we sort the values of the pixel and its neighbors to determine their median and assign its value to output pixel $g(x,y)$.

Median Filter

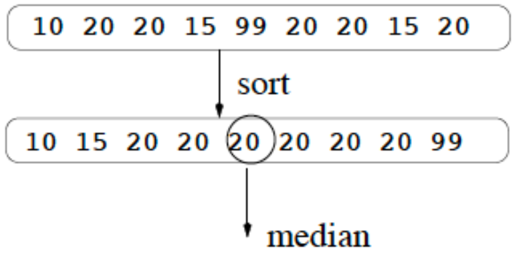
- Replace each pixel value by the median of the gray-levels in the neighborhood of the pixels

Area or Mask Processing Methods

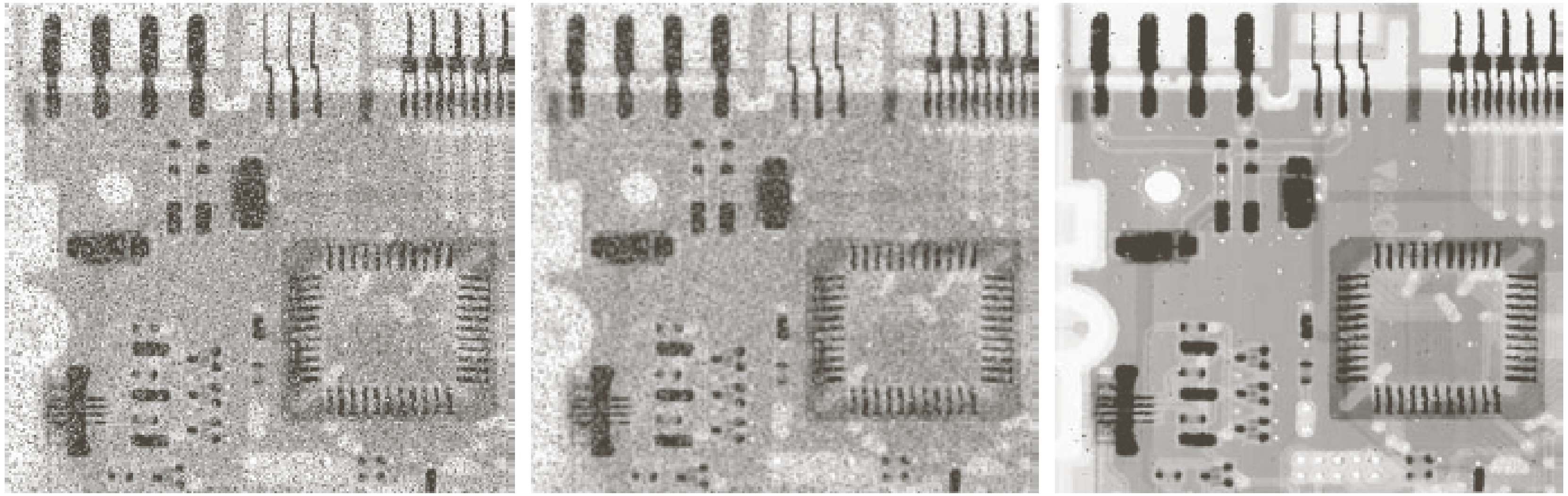


$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels



Example: Use of Median Filtering for Noise Reduction



a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Median vs. Gaussian Filtering