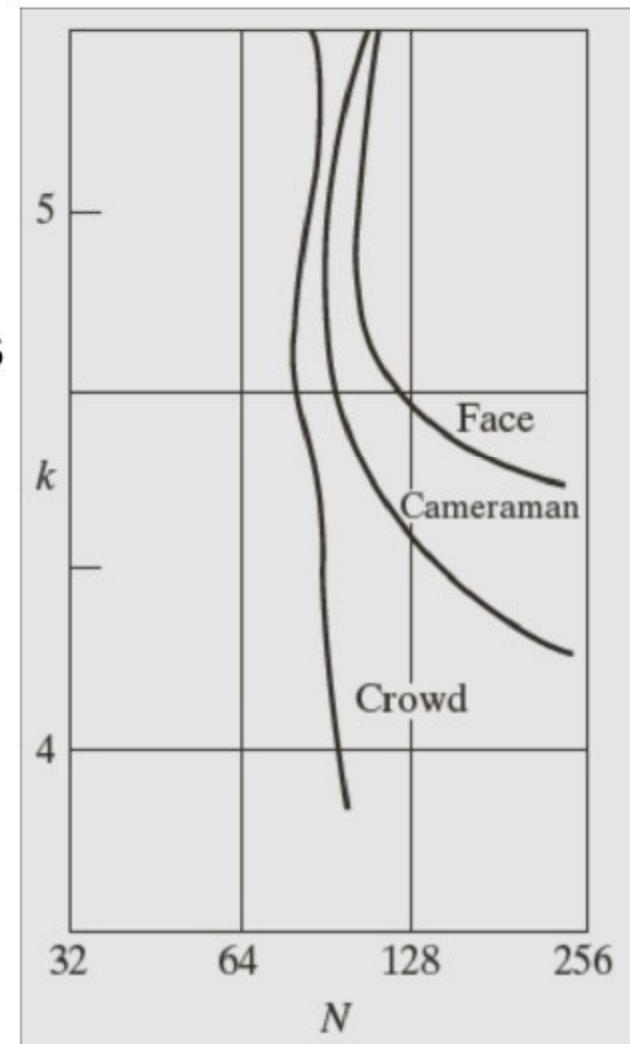


# Iso-preference curves

What happens when we vary both Sampling & Quantization?

A study conducted by Haung [1965] on this effect of gray level and contouring, and the results were shown in the graph in the form of curves, known as **Iso-preference curves**

- Image having total pixels as  $N$  with intensity of  $k$  bits are plotted in **Iso-preference curves**
- Points lying on an iso-preference curve correspond to images of **equal subjective quality**.



# Sampling-Quantization Tradeoff

## Types of images



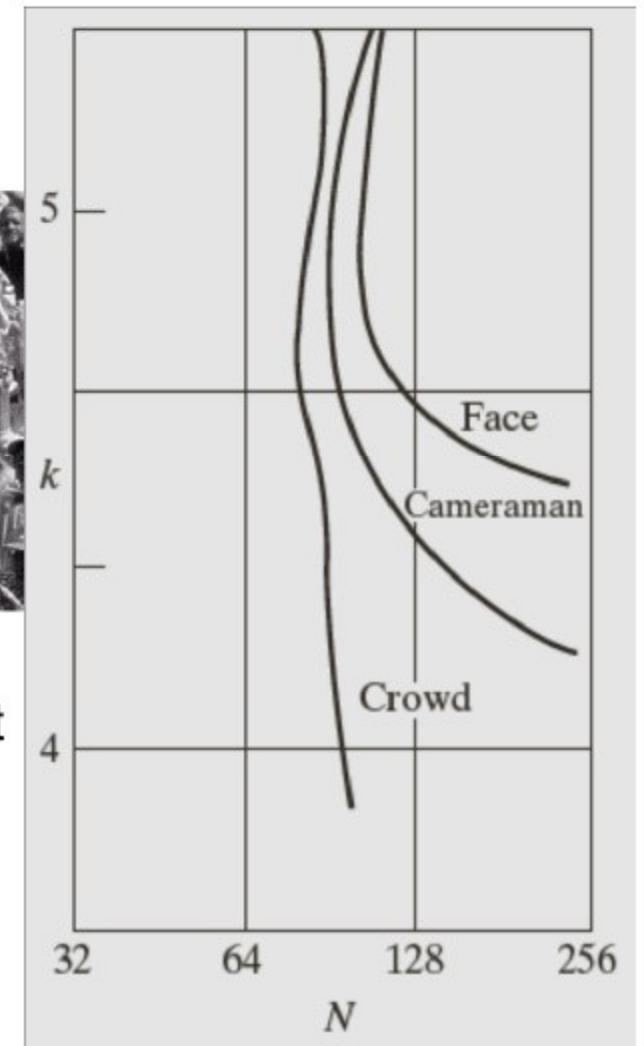
low detail



Intermediate detail



Large amount of detail



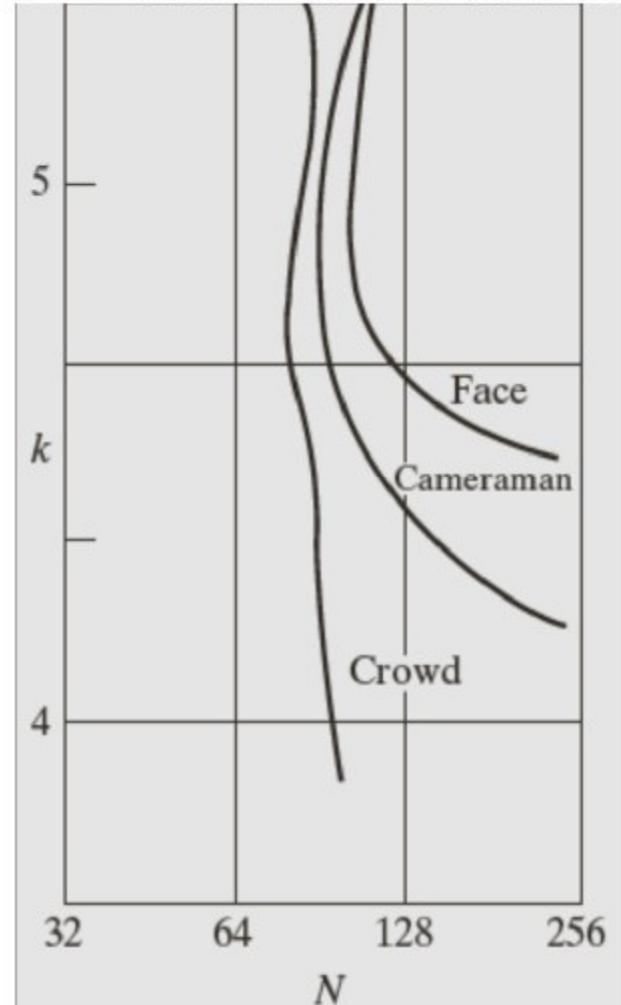
# Remarks



- Face and Cameraman: The perceived quality of two images, remained the same in some intervals in which the spatial resolution was increased, but the no. of gray levels actually decreased.

Reason: decrease in  $k$  tends to increase the apparent contrast.

- Crowd: For fixed value of  $N$ , the perceived quality for this type of image is nearly independent of the number of gray levels.

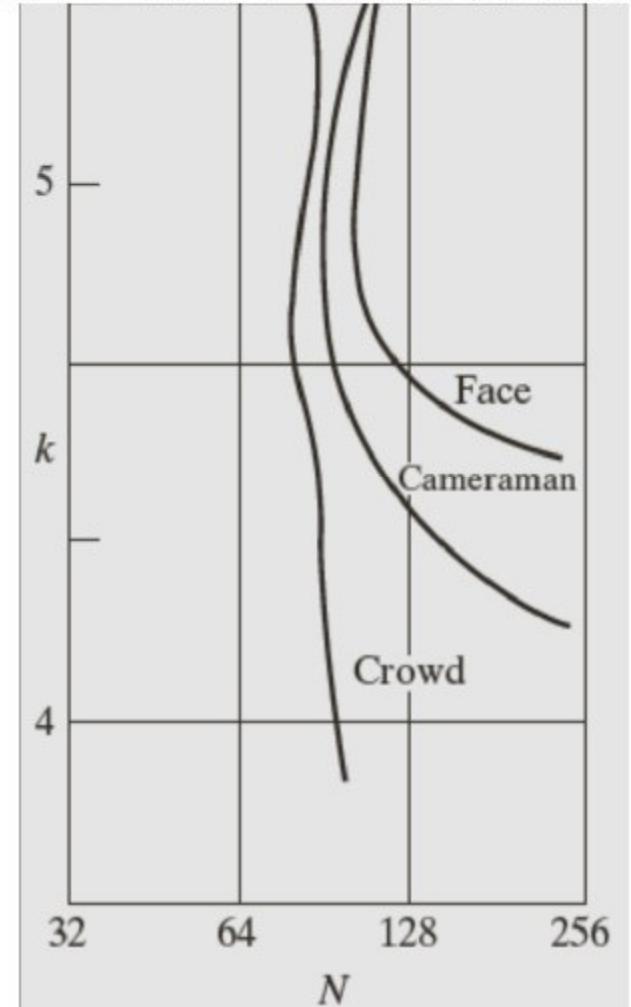


Iso-preference curve

# Remarks



1. Iso-preference curves tend to shift right and upward (i.e. better image quality)
2. In images with a large amount of details, only a few gray levels are needed
3. In the other two image categories, the perceived quality remained the same in some intervals in which  $N$  was increased but  $k$  actually decreased.



Iso-preference curve

# Zooming and Shrinking

1024x1024



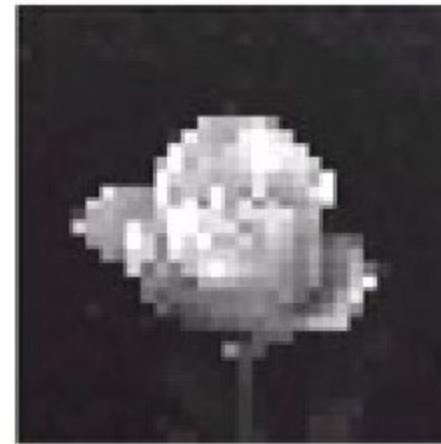
128x128



256x256



32x32

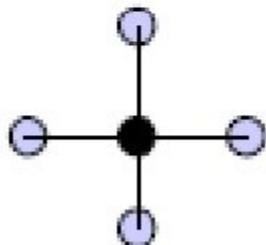


# Interpolation

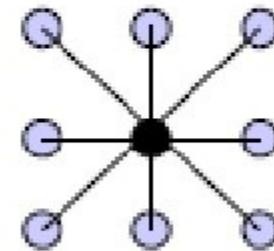
- ❖ To overcome sampling and quantization problems, we need image interpolation.
- ❖ Interpolation is the process of using known data to estimate values at unknown locations data.

# Commonly used Terminology

- ❖ Neighbors of a pixel  $p(i, j)$



$$N_4(p) = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$$

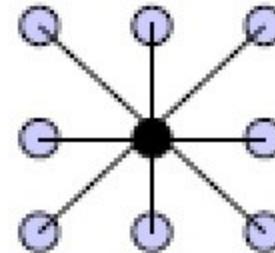
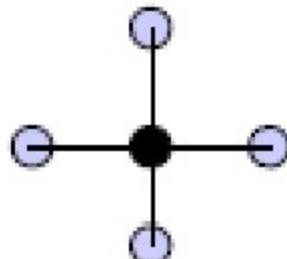


$$N_8(p) = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1), (i-1, j-1), (i-1, j+1), (i+1, j-1), (i+1, j+1)\}$$

- ❖ Adjacency

4-adjacency:  $p, q$  are 4-adjacent if  $p$  is in the set  $N_4(q)$

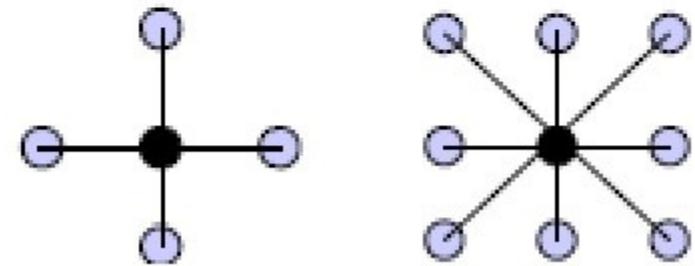
8-adjacency:  $p, q$  are 8-adjacent if  $p$  is in the set  $N_8(q)$



# Distance measures D

D is distance between pixels as function or metric:

- a)  $D(p,q) \geq 0$  ( $D(p,q) = 0$  iff  $p=q$ )
- b)  $D(p,q) = D(q,p)$
- c)  $D(p,z) \leq D(p,q) + D(q,z)$



Examples:

- Euclidean:  $D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$
- $D_4$  (City Block or Manhattan):  $D_4(p,q) = |x-s| + |y-t|$
- $D_8$  (Chessboard):  $D_8(p,q) = \text{Max}\{|x-s|, |y-t|\}$

# Commonly used Terminology

- Euclidean:  $D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$
- $D_4$  (City Block or Manhattan):  $D_4(p, q) = |x-s| + |y-t|$
- $D_8$  (Chessboard):  $D_8(p, q) = \text{Max}\{|x-s|, |y-t|\}$

Euclidean distance  
*(2-norm)*

		2		
		1		
2	1	0	1	2
		1		
		2		

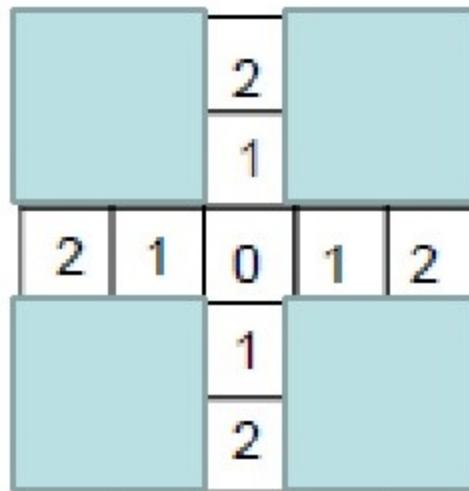
$D_4$  distance  
*(city-block distance)*

$D_8$  distance  
*(checkboard distance)*

# Commonly used Terminology

- Euclidean:  $D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$
- $D_4$  (City Block or Manhattan):  $D_4(p, q) = |x-s| + |y-t|$
- $D_8$  (Chessboard):  $D_8(p, q) = \text{Max}\{|x-s|, |y-t|\}$

Euclidean distance  
*(2-norm)*



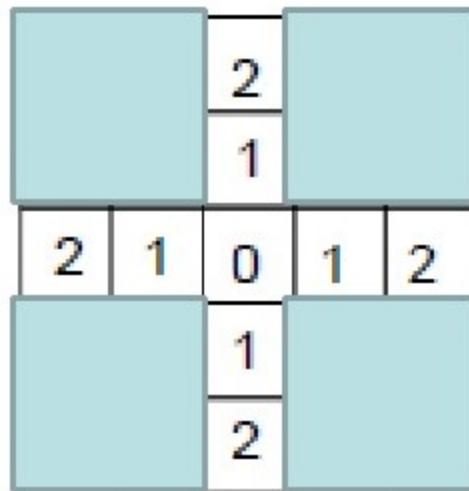
$D_4$  distance  
*(city-block distance)*

$D_8$  distance  
*(checkboard distance)*

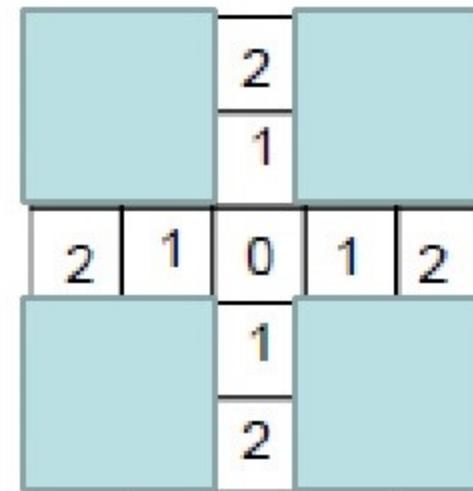
# Commonly used Terminology

- Euclidean:  $D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$
- $D_4$  (City Block or Manhattan):  $D_4(p, q) = |x-s| + |y-t|$
- $D_8$  (Chessboard):  $D_8(p, q) = \text{Max}\{|x-s|, |y-t|\}$

Euclidean distance  
(*2-norm*)



$D_4$  distance  
(city-block distance)

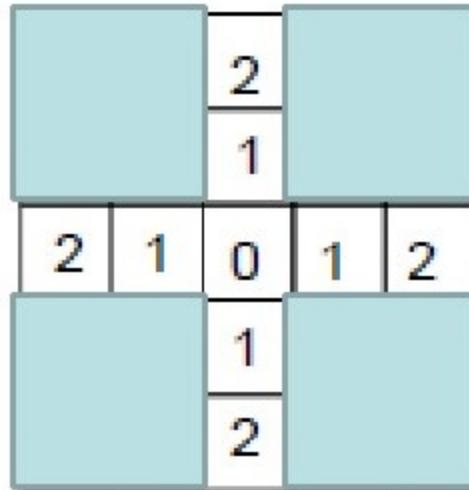


$D_8$  distance  
(checkboard distance)

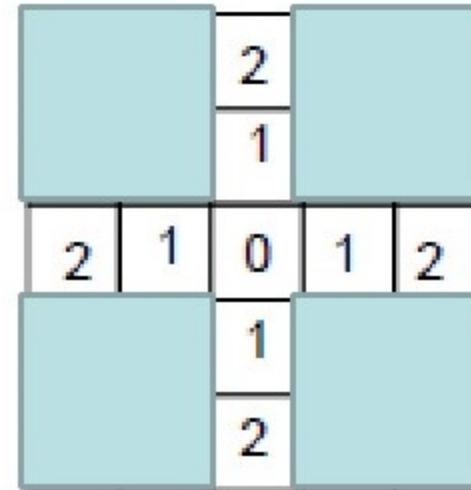
# Commonly used Terminology

- Euclidean:  $D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$
- $D_4$  (City Block or Manhattan):  $D_4(p, q) = |x-s| + |y-t|$
- $D_8$  (Chessboard):  $D_8(p, q) = \text{Max}\{|x-s|, |y-t|\}$

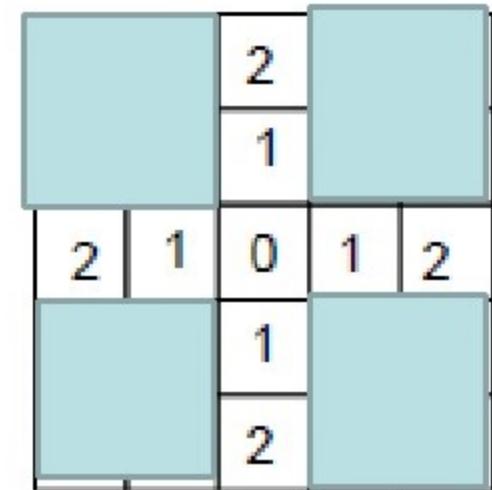
Euclidean distance  
*(2-norm)*



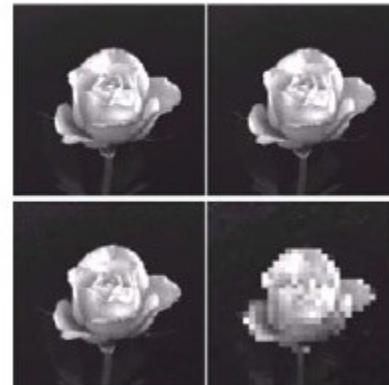
$D_4$  distance  
*(city-block distance)*



$D_8$  distance  
*(checkboard distance)*



# Zooming and Shrinking



- ❖ To overcome these problems, we need image interpolation.
- ❖ Interpolation is the process of using known data to estimate values at unknown locations.
- ❖ Interpolation is a basic tool used extensively in tasks such as  
**zooming, shrinking, rotating, and geometric corrections**

# How does Interpolation Work?

34	30	23	25
20	26	10	28
25	32	30	35
12	34	35	25



34	0	30	0	23	0	25	0
0	0	0	0	0	0	0	0
20	0	26	0	10	0	28	0
0	0	0	0	0	0	0	0
25	0	32	0	30	0	35	0
0	0	0	0	0	0	0	0
12	0	34	0	35	0	25	0
0	0	0	0	0	0	0	0

1/4 = 25 % of original.

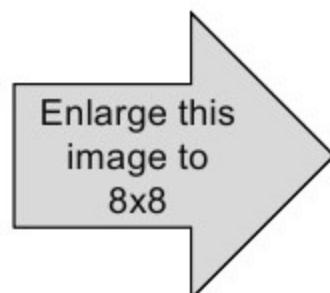
The pixels with no values assigned to them are particularly problematic because they appear as ‘holes’ in the destination image that are aesthetically unpleasing

# How does Interpolation Work?

Simplest method is Linear interpolation

Suppose that we want to find the value of the unknown function  $f$  at the point  $P = (x, y)$ . It is assumed that we know the value of  $f(x_1, y_1) = Q_{11}$  then the intensity at  $F(x_1, y_2) = aQ_{11}$  and  $f(x_2, y_1)$  is  $Q_{21}$  then  $F(x_2, y_2) = bQ_{21}$

34	30	23	25
20	26	10	28
25	32	30	35
12	34	35	25



34		30		23		25	
20		26		10		28	
25		32		30		35	
12		34		35		25	

1/4 = 25 % of original.

# How does Interpolation Work?

34	30	23	25
20	26	10	28
25	32	30	35
12	34	35	25

Enlarge this  
image to 8x8

34	0	30	0	23	0	25	0
0	0	0	0	0	0	0	0
20	0	26	0	10	0	28	0
0	0	0	0	0	0	0	0
25	0	32	0	30	0	35	0
0	0	0	0	0	0	0	0
12	0	34	0	35	0	25	0
0	0	0	0	0	0	0	0

1/4 = 25 % of original.

The more suitable approach is , we use the four nearest neighbours to estimate the intensity at a given location.

- Nearest neighbor interpolation and
- Bilinear interpolation.

# Interpolation

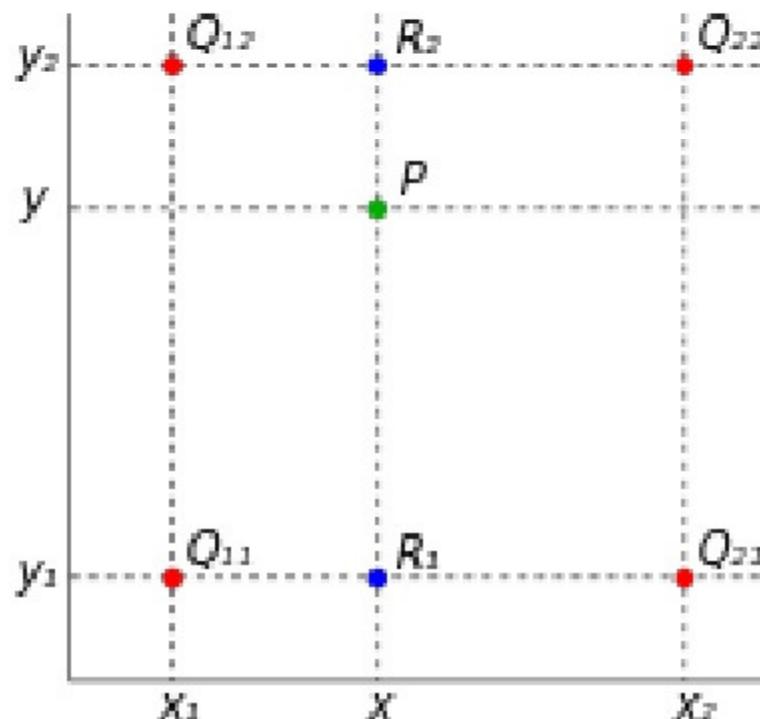
Two simplest method:

- *Nearest neighbor interpolation and*
- *Bilinear interpolation.*

# Bilinear Interpolation

In bilinear interpolation, linear interpolation is carried out in each of the two orthogonal coordinates of the image.

$$P = (1-c).(1-r).Q_{12} + (1-c).r. Q_{22} + c.(1-r).Q_{11} + c.r. Q_{21}$$



[http://en.wikipedia.org/wiki/  
Bilinear\\_interpolation](http://en.wikipedia.org/wiki/Bilinear_interpolation)

# Bilinear Interpolation

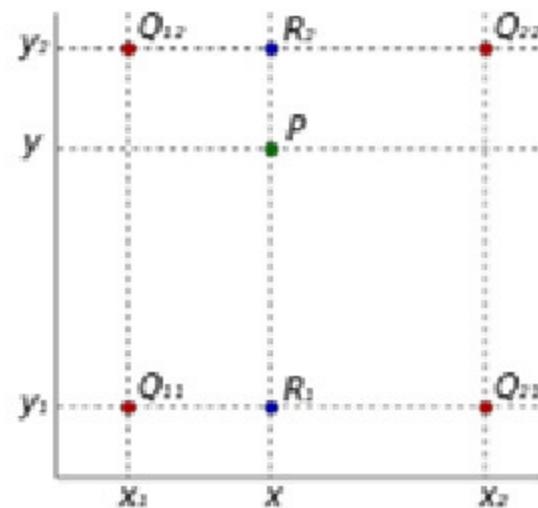
Linear interpolation in x between red value gives us the blue value

$$R_1 = (1-r) Q_{11} + r Q_{21}$$

$$R_2 = (1-r) Q_{12} + r Q_{22}$$

Linear interpolation in y between the blue values gives us the green value

$$P = (1-c) R_2 + c R_1$$



# Example

Apply Bilinear interpolation for given patch 4x4 image and find out interpolated 8x8 image.

Assume  $c = r = 0.5$

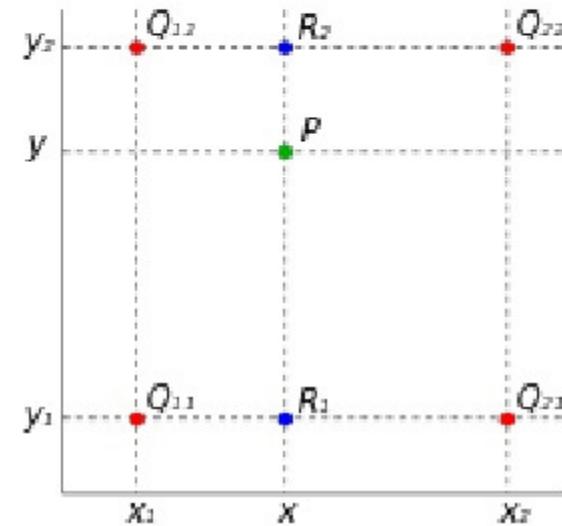
2	1	4	2
3	1	2	1
1	2	1	4
4	1	2	3

# Example

Apply Bilinear interpolation for given patch 4x4 image and find out interpolated 8x8 image.

Assume  $c = r = 0.5$

2	1	4	2
3	1	2	1
1	2	1	4
4	1	2	3



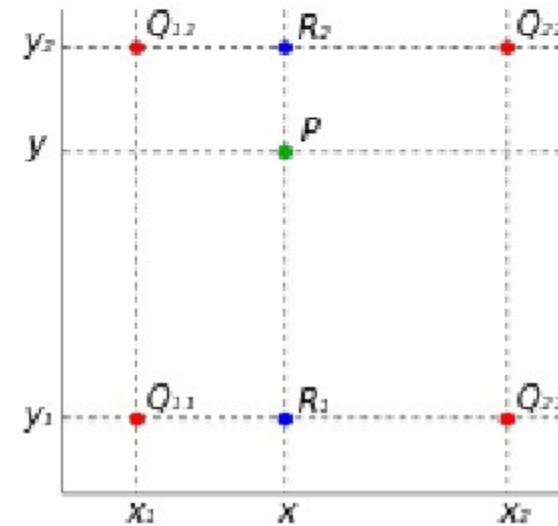
$$P = (1-c).(1-r).Q_{12} + (1-c).r.Q_{22} + c.(1-r).Q_{11} + c.r.Q_{21}$$

# Example

Apply Bilinear interpolation for given patch 4x4 image and find out interpolated 8x8 image.

Assume  $c = r = 0.5$

2	1	4	2
3	1	2	1
1	2	1	4
4	1	2	3



$$R_2 = (1-r) Q_{12} + r Q_{22}; \quad R_1 = (1-r) Q_{11} + r Q_{21}$$

$$P = (1-c) R_2 + c R_1$$

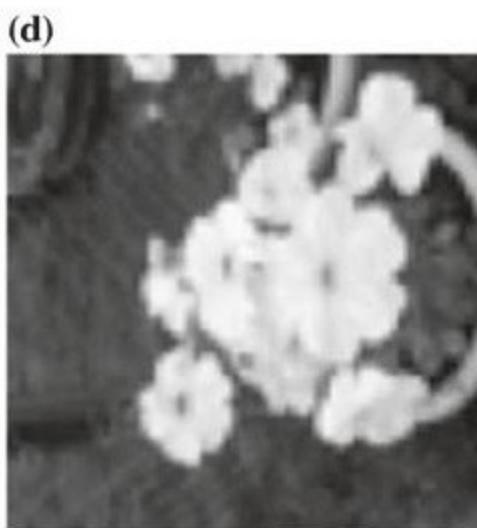
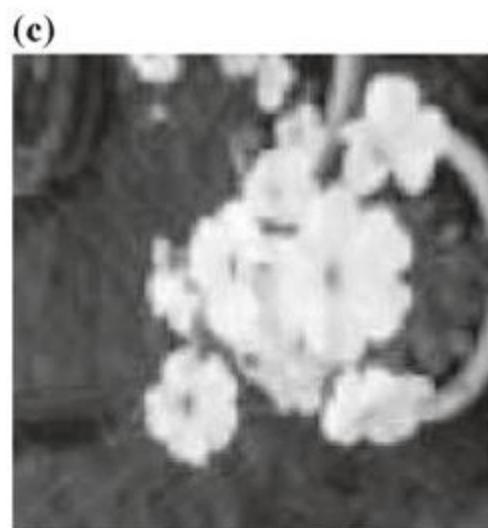
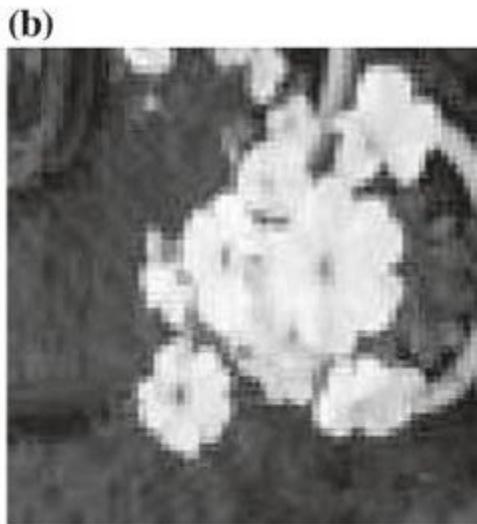
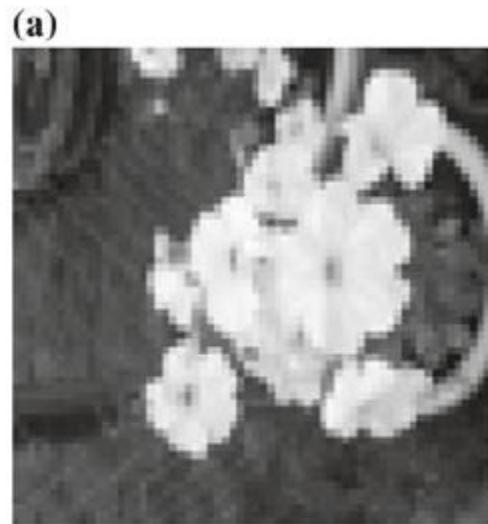
# Example

2	1	4	2
3	1	2	1
1	2	1	4
4	1	2	3

2.00	1.50	1.00	2.50	4.00	3.00	2.00
3.00	2.00	1.00	1.50	2.00	1.50	1.00
1.00	1.50	2.00	1.50	1.00	2.50	4.00
4.00	2.50	1.00	1.50	2.00	2.50	3.00

2.00	1.50	1.00	2.50	4.00	3.00	2.00
2.50	1.75	1.00	2.00	3.00	2.25	1.50
3.00	2.00	1.00	1.50	2.00	1.50	1.00
2.00	1.75	1.50	1.50	1.50	2.00	2.50
1.00	1.50	2.00	1.50	1.00	2.50	4.00
2.50	2.00	1.50	1.50	1.50	2.50	3.50
4.00	2.50	1.00	1.50	2.00	2.50	3.00

# Example



a) A  $64 \times 64$ ,  
8-bit gray

b) nearest-neighbor  
interpolated  $127 \times$   
 $127$  image

c) bilinear interpolated  
 $127 \times 127$  image

d) bilinear interpolated  
 $253 \times 253$  image

# Interpolation

- ❖ The high-end interpolation methods and advanced image interpolation algorithms are more complex & computationally more expensive.
  - ❖ e.g., bicubic and *B-spline*, edge orientation-based interpolation, transform domain etc.

# Zoom in – zoom out

There are two cases: the scaling factor(s) can be either greater than 1 (*upsampling*) or less than 1 (*downscaling*).

For *binary* images, we simply replicate pixels when up scaling.

When downscaling binary images:

- Downscale by simple subsampling

# Interpolation Artifacts

- ❖ There are two cases: the scaling factor(s) can be either greater than 1 (*upsampling*) or less than 1 (*downscaling*).



- Downscale by simple subsampling, and take chances with aliasing.
- Apply a lowpass filter before subsampling

<https://www.cambridgeincolour.com/tutorials/image-interpolation.htm>

# Interpolation

Method	Description
'nearest'	Nearest-neighbor interpolation; the output pixel is assigned the value of the pixel that the point falls within. No other pixels are considered.
'bilinear'	Bilinear interpolation; the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood
'bicubic'	Bicubic interpolation (the default); the output pixel value is a weighted average of pixels in the nearest 4-by-4 neighborhood
Interpolation Kernel	Description
'box'	Box-shaped kernel
'triangle'	Triangular kernel (equivalent to 'bilinear')
'cubic'	Cubic kernel (equivalent to 'bicubic')
'lanczos2'	Lanczos-2 kernel
'lanczos3'	Lanczos-3 kernel

## Where Are Good Interpolation Techniques Coming From?

From Computational methods, Computers are nothing but Maths machines, using nothing but Maths to crunch our image improvement.

Maths is not always easy, so all interpolation methods are designed to excel at one single task: producing the highest interpolation quality for the broadest ranges of data. The rest, as always, depends on your definitions and priorities.

## Super-Resolution Image Reconstruction

It produces high-resolution images, using the existing low-cost imaging devices from a single image or few snapshots of low-resolution images.

# Image Histogram

Histogram is a **global descriptor**; reflect no **structural or spatial information of pixel**.

**It is good method of image enhancement but it does not consider human visual response to light which is non-linear.**

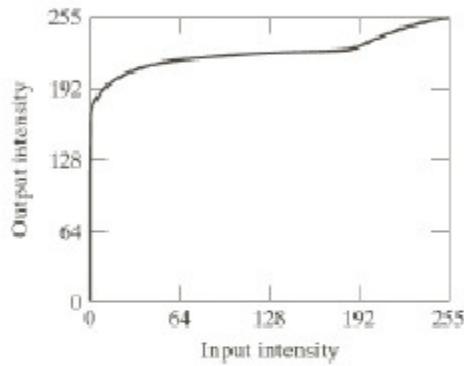
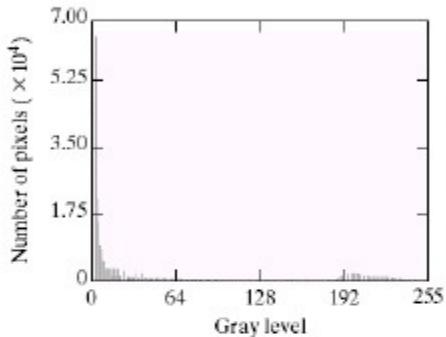
- ❖ If we required local enhancement then we have to modify Histogram on the bases of local requirements or specifications.

Statistics from local image histograms can be used for local image enhancement; it can enhance dark areas while leaving the light area as unchanged as possible

# Histogram Matching/Specification



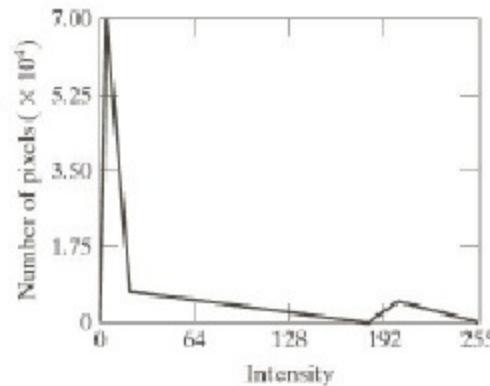
a



b

FIGURE 3.20 (a) Image of the Mars moon Photos taken by NASA's *Mars Global Surveyor*. (b) Histogram. (Original image courtesy of NASA.)

Modify Histogram on  
the bases of local  
enhancement  
requirements or  
specifications.



# Histogram Matching/Specification

Modify Histogram on the bases of local enhancement requirements.

- Histogram matching/specification enables us to “match” the grayscale distribution in one image to the grayscale distribution in another image.
- Given images - input A and desired B, using point processing we would like to generate an image C from A such that  $p_C(i) \sim p_B(i)$ ; ( $i = 0, \dots, 255$ ).
- More generally, given an image A and a histogram  $h_B(i)$ , we would like to generate an image C such that  $p_C(i) \sim p_B(i)$ ; ( $i = 0, \dots, 255$ ).

# Procedure

- Input transform –

$$s = T(r) = \int_o^r p_r(w)dw; p_r(r) \text{ for given image}$$

- Output desired –

$$G(z) = \int_o^z p_z(t)dt; p_z(z) \text{ for desired image}$$

$$s = G(z)$$

- Inverse transform to get output for specified histogram

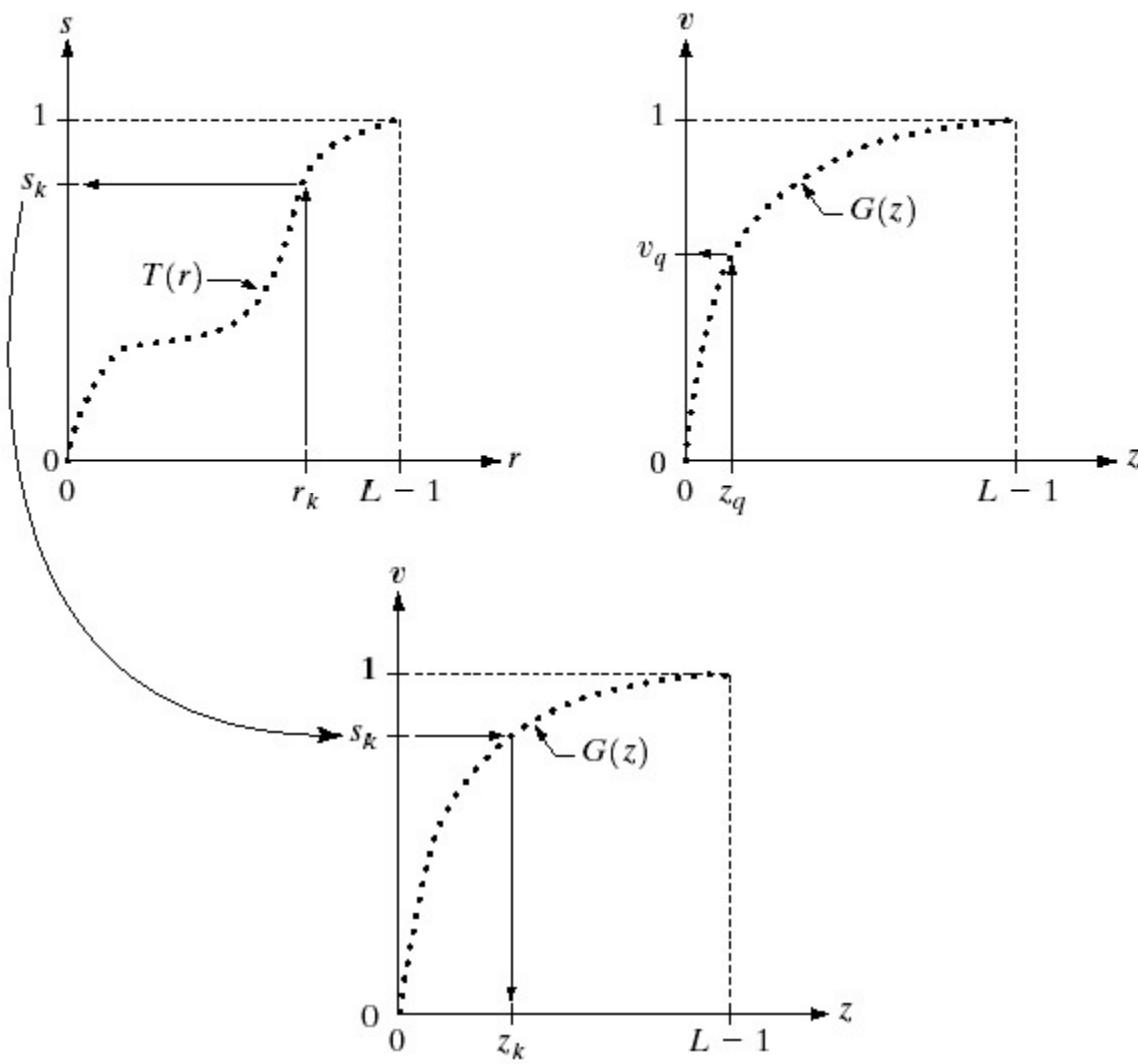
$$G(z) = T(r) = s$$

$$\Rightarrow z = G^{-1}(s) = G^{-1}(T(r))$$

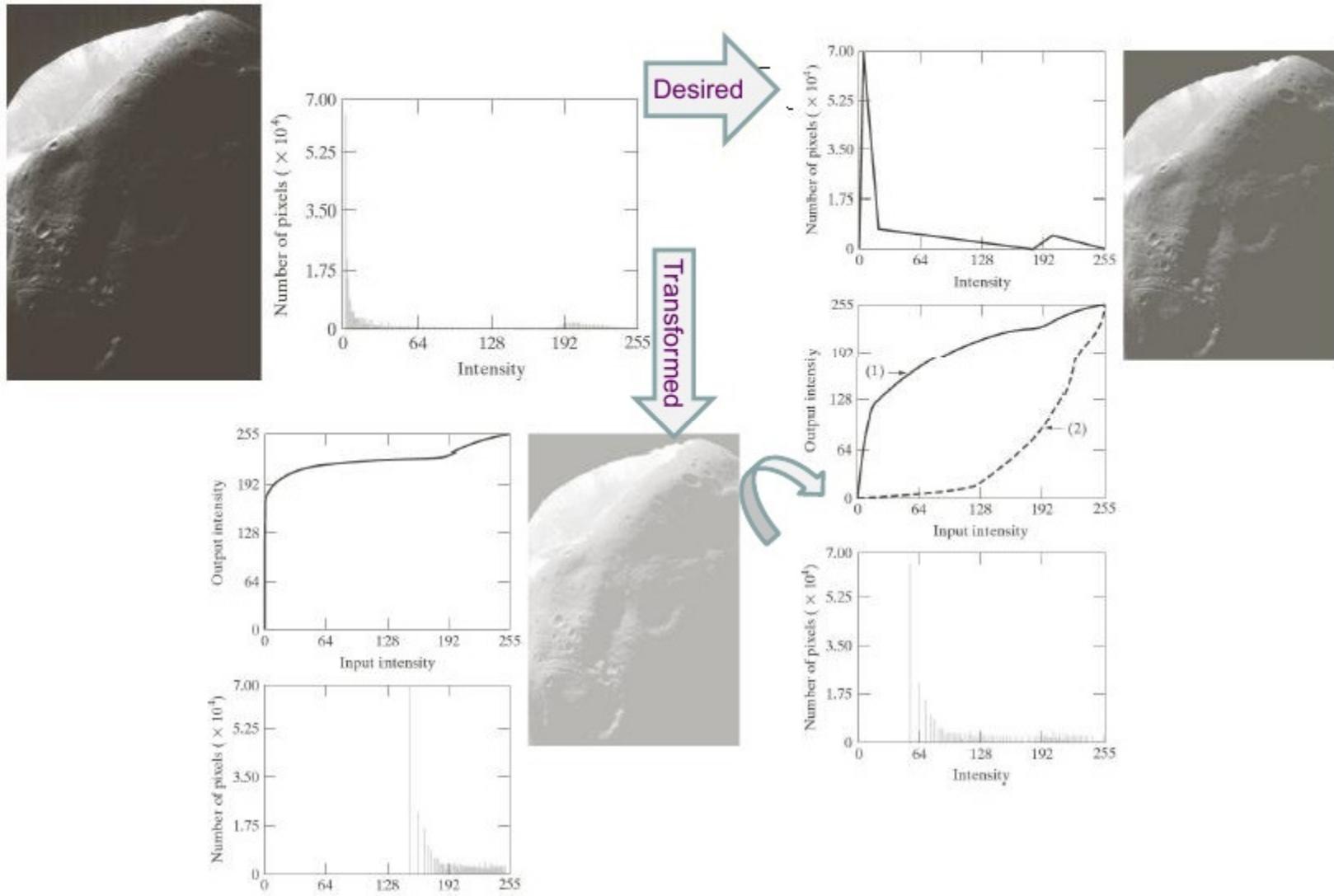
# Procedure

a  
b  
c

**FIGURE 3.19**  
(a) Graphical interpretation of mapping from  $r_k$  to  $s_k$  via  $T(r)$ .  
(b) Mapping of  $z_q$  to its corresponding value  $v_q$  via  $G(z)$ .  
(c) Inverse mapping from  $s_k$  to its corresponding value of  $z_k$ .



# Procedure



## Example 3.7

Assume continuous intensity values, suppose that an image has the intensity PDF  $p(r) = 2r/(L-1)^2$  and  $p(r)=0$  for other values.

Find the transformation function that will produce an image whose intensity PDF is  $p(z) = 3z^2/(L-1)^3$  and  $p(z)=0$  for other value of  $z$ .

$$s = T(r) = (L-1) \int_0^r p(w) dw = \frac{r^2}{(L-1)}$$

$$G(z) = (L-1) \int_0^z p(w) dw = \frac{z^3}{(L-1)^2}$$

$$s = G(z) \Rightarrow z = [(L-1)^2 s]^{1/3}$$

$$z = [(L-1)r^2]^{1/3}$$

# Procedure

An image with the specified probability density function from an input function using the following steps:

1. Obtain  $s_k = T(r)$  using histogram equalization.
2. Obtain  $G(z)$  by histogram equalizing to specified histogram.
3. For every value of  $s_k$ ,  $k=0, 1, 2 \dots L-1$ , use the stored values of  $G$  from step 2 to find the corresponding value of  $z$  so that  $G(z)$  is closest to  $s_k$  and store these mapping from  $s$  to  $z$ .
4. Obtain the output image by applying  $G(z)$  to all pixels in input image.

It is difficult to obtain analytical expressions for  $T(r)$  and  $G(z)$

## Example 3.8

Suppose that a 3-bit image ( $L=8$ ) of size  $64 \times 64$  pixels ( $MN = 4096$ ) has the intensity distribution shown in the following table (on the left). Get the histogram transformation function and make the output image with the specified histogram, listed in the table on the right.

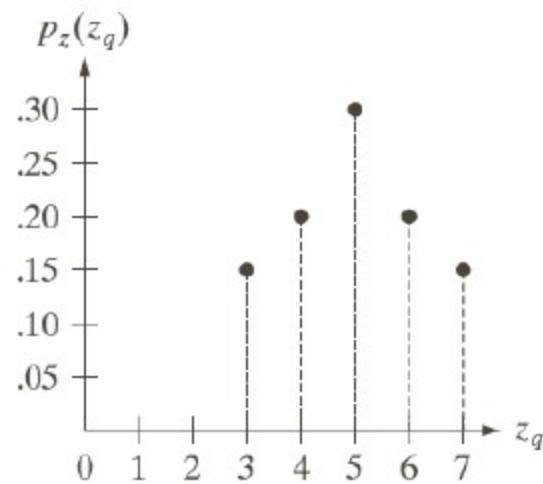
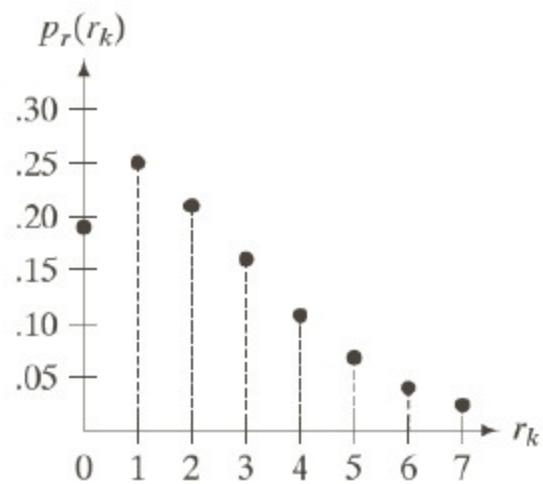
$r_k$	$n_k$	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

$z_q$	Specified $p_z(z_q)$
$z_0 = 0$	0.00
$z_1 = 1$	0.00
$z_2 = 2$	0.00
$z_3 = 3$	0.15
$z_4 = 4$	0.20
$z_5 = 5$	0.30
$z_6 = 6$	0.20
$z_7 = 7$	0.15

## Example 3.8

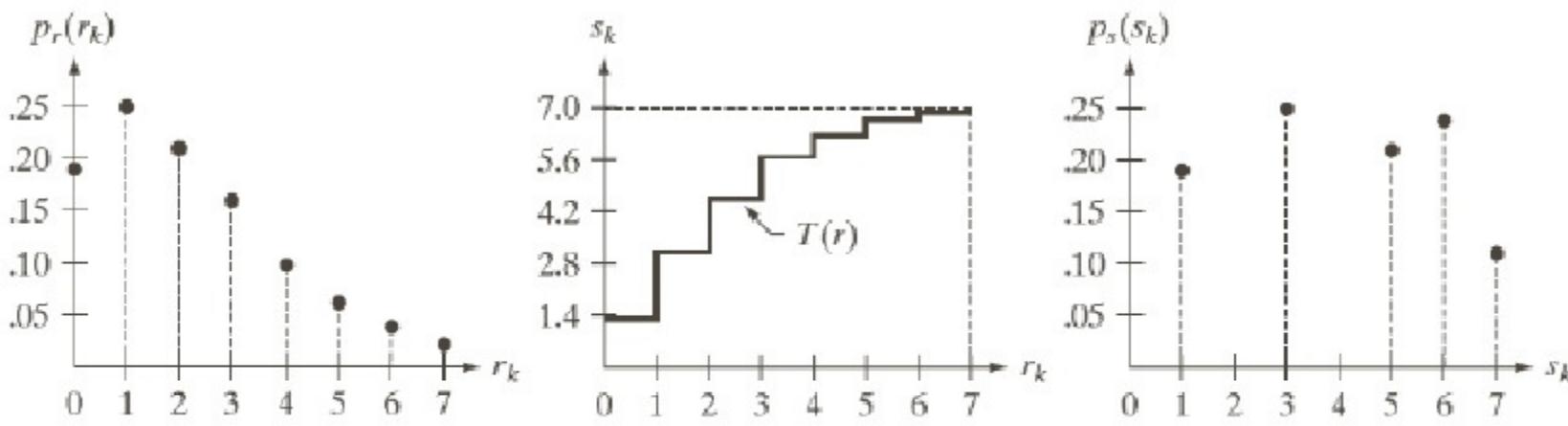
$r_k$	$n_k$	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

$z_q$	Specified $p_z(z_q)$
$z_0 = 0$	0.00
$z_1 = 1$	0.00
$z_2 = 2$	0.00
$z_3 = 3$	0.15
$z_4 = 4$	0.20
$z_5 = 5$	0.30
$z_6 = 6$	0.20
$z_7 = 7$	0.15



## Example 3.8

Pixels modified intensities and its distribution.



a b c

**FIGURE 3.19** Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

$$S_0 = 1.33=1; \quad S_1 = 3.08=3; \quad S_2 = 4.55=5;$$

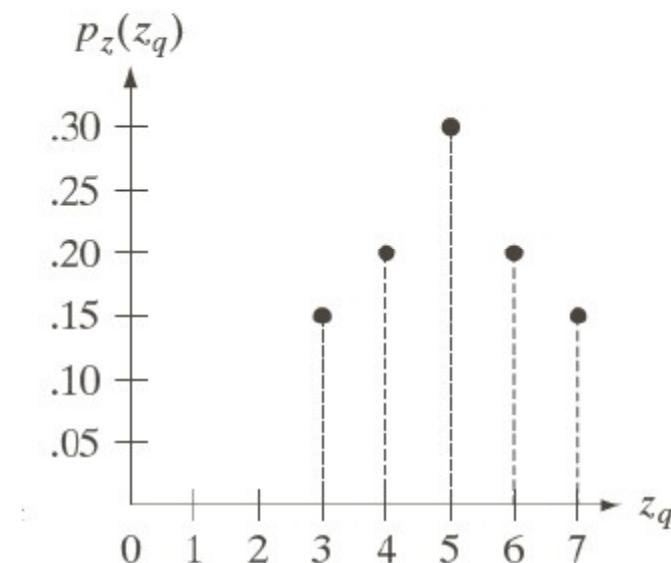
$$S_3 = 5.67=6; \quad S_4 = 6.23=6; \quad S_5 = 6.65=7;$$

$$S_6 = 6.86=7; \quad S_7 = 7.00=7$$

## Example 3.8

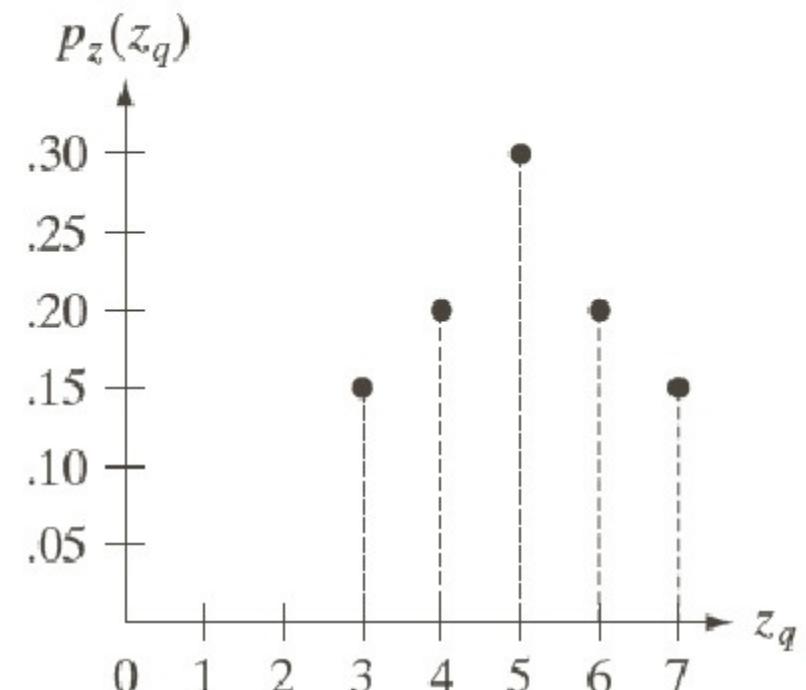
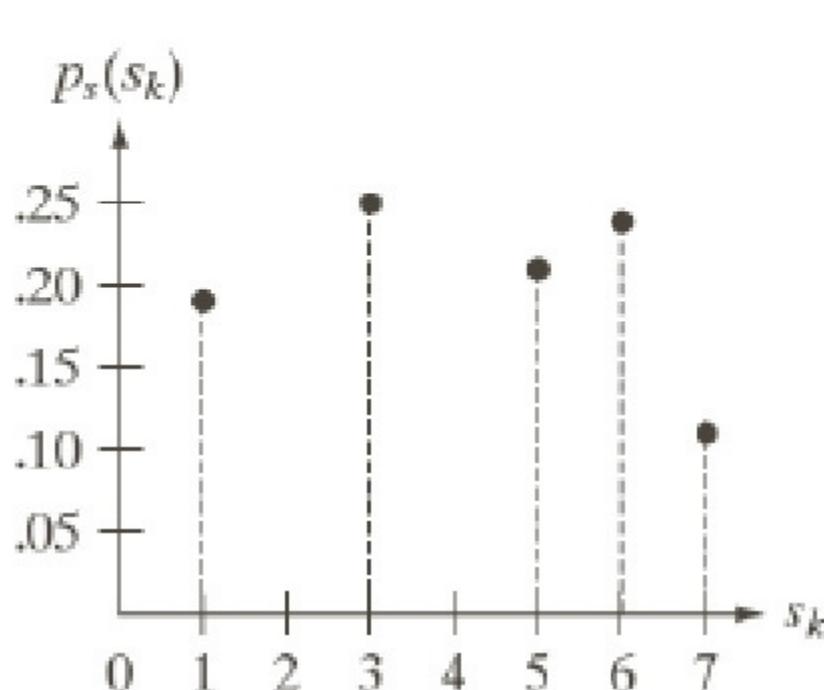
Desired pixels modified intensities and its distribution.

$z_q$	Specified $p_z(z_q)$
$z_0 = 0$	0.00
$z_1 = 1$	0.00
$z_2 = 2$	0.00
$z_3 = 3$	0.15
$z_4 = 4$	0.20
$z_5 = 5$	0.30
$z_6 = 6$	0.20
$z_7 = 7$	0.15



## Example 3.8

Histogram equalized and desired pixels distribution.

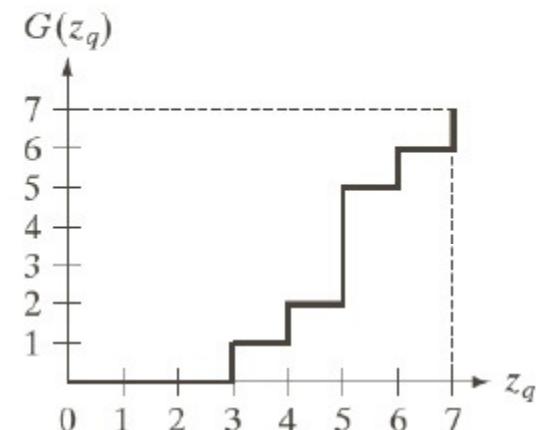


## Example 3.8

Compute all the values of the transformation function  $G(z)$ ,

$z_q$	Specified $p_z(z_q)$
$z_0 = 0$	0.00
$z_1 = 1$	0.00
$z_2 = 2$	0.00
$z_3 = 3$	0.15
$z_4 = 4$	0.20
$z_5 = 5$	0.30
$z_6 = 6$	0.20
$z_7 = 7$	0.15

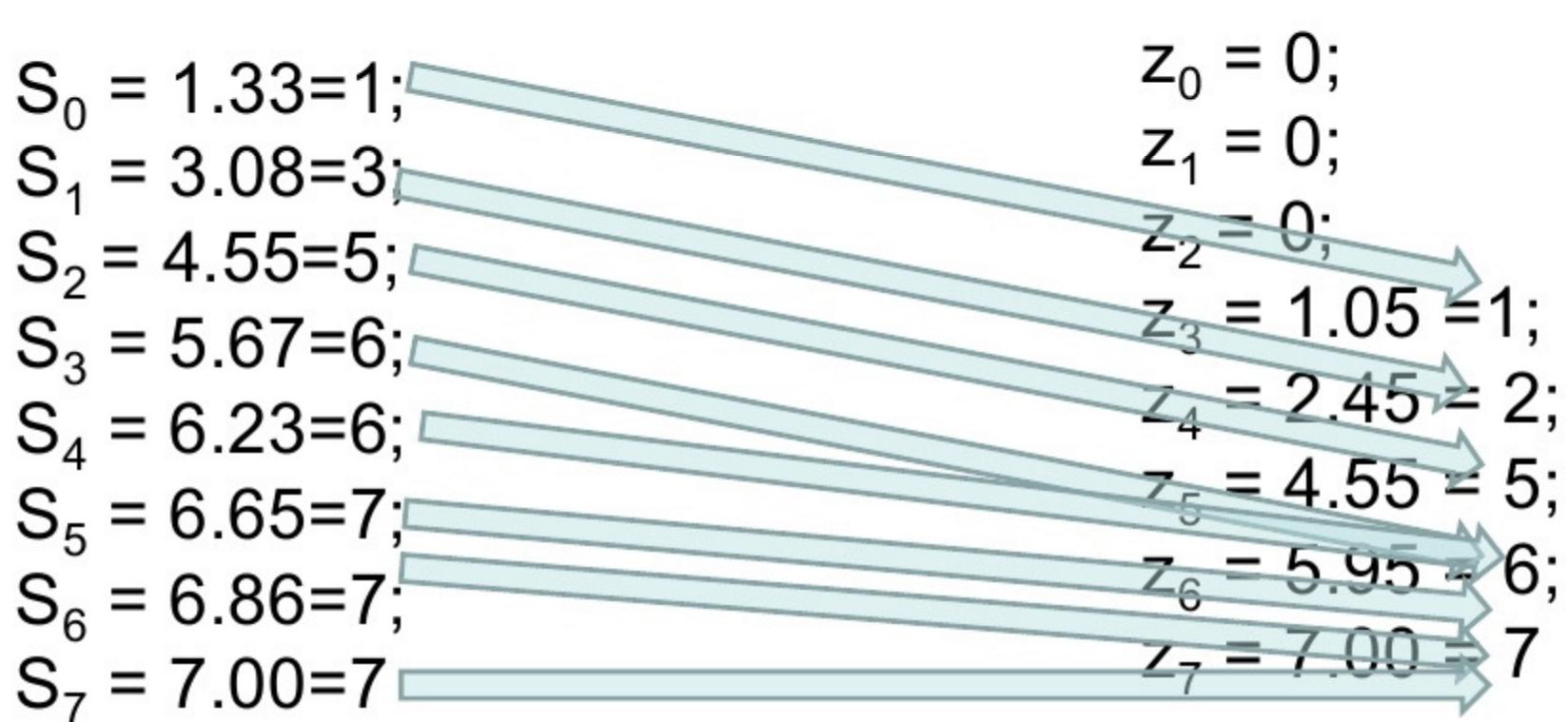
$$G(z_k) = 7 \sum_{j=0}^0 p_z(z_j)$$



$$\begin{aligned}z_0 &= 0; z_1 = 0; z_2 = 0; z_3 = 1.05 = 1; \\z_4 &= 2.45 = 2; z_5 = 4.55 = 5; \\z_6 &= 5.95 = 6; z_7 = 7.00 = 7\end{aligned}$$

## Example 3.8

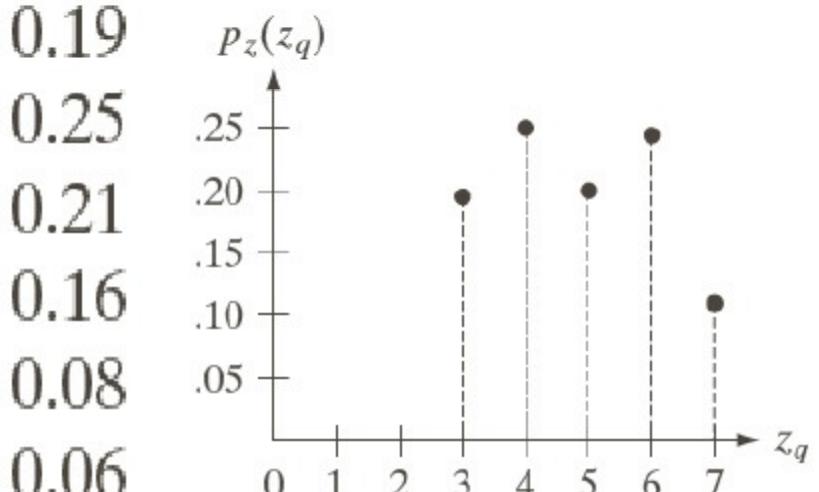
Mapped the values from 's' to desired histogram of output image 'z'



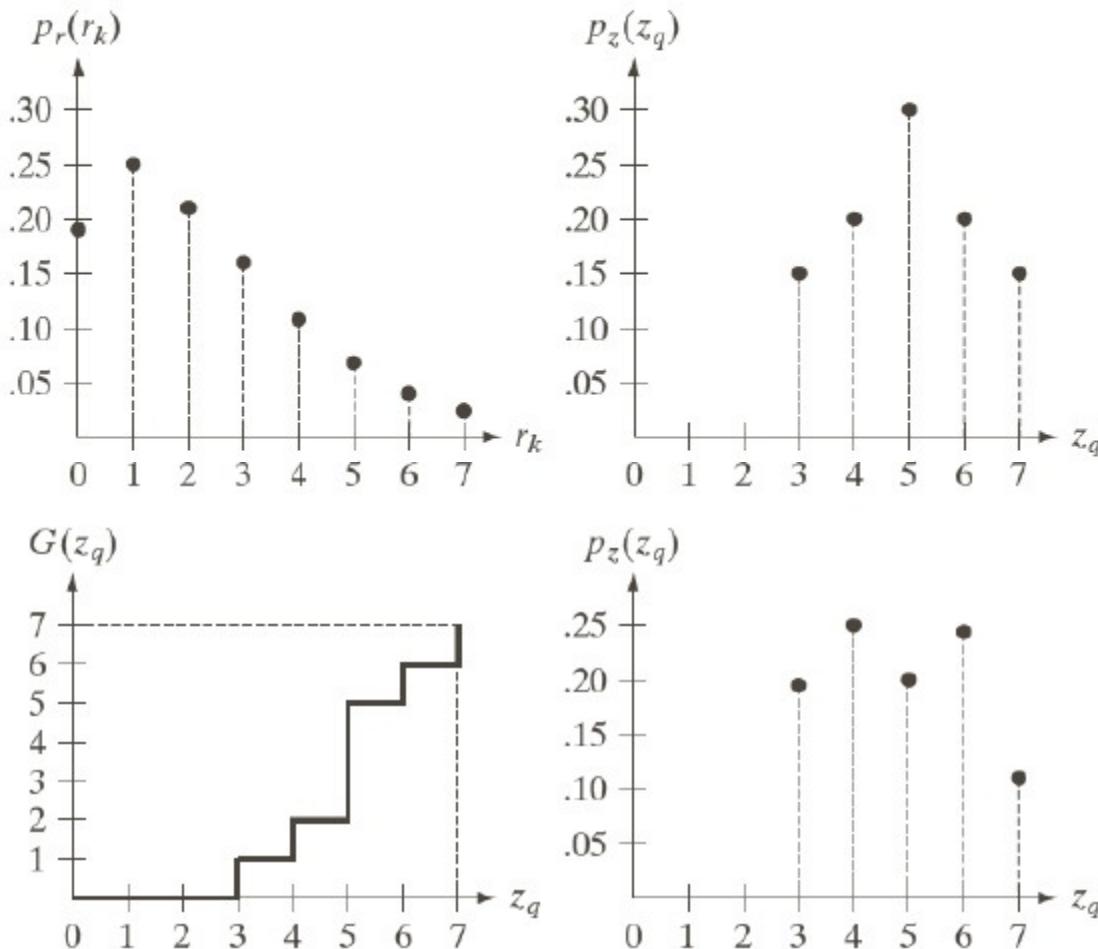
## Example 3.8

Mapped the values from 's' to desired histogram of output image 'z'

$S_0 = 1.33 = 1;$	$z_0 = 0;$	0.19
$S_1 = 3.08 = 3;$	$z_1 = 0;$	0.25
$S_2 = 4.55 = 5;$	$z_2 = 0;$	0.21
$S_3 = 5.67 = 6;$	$z_3 = 1.05 = 1;$	0.16
$S_4 = 6.23 = 6;$	$z_4 = 2.45 = 2;$	0.08
$S_5 = 6.65 = 7;$	$z_5 = 4.55 = 5;$	0.06
$S_6 = 6.86 = 7;$	$z_6 = 5.95 = 6;$	0.03
$S_7 = 7.00 = 7$	$z_7 = 7.00 = 7$	0.02



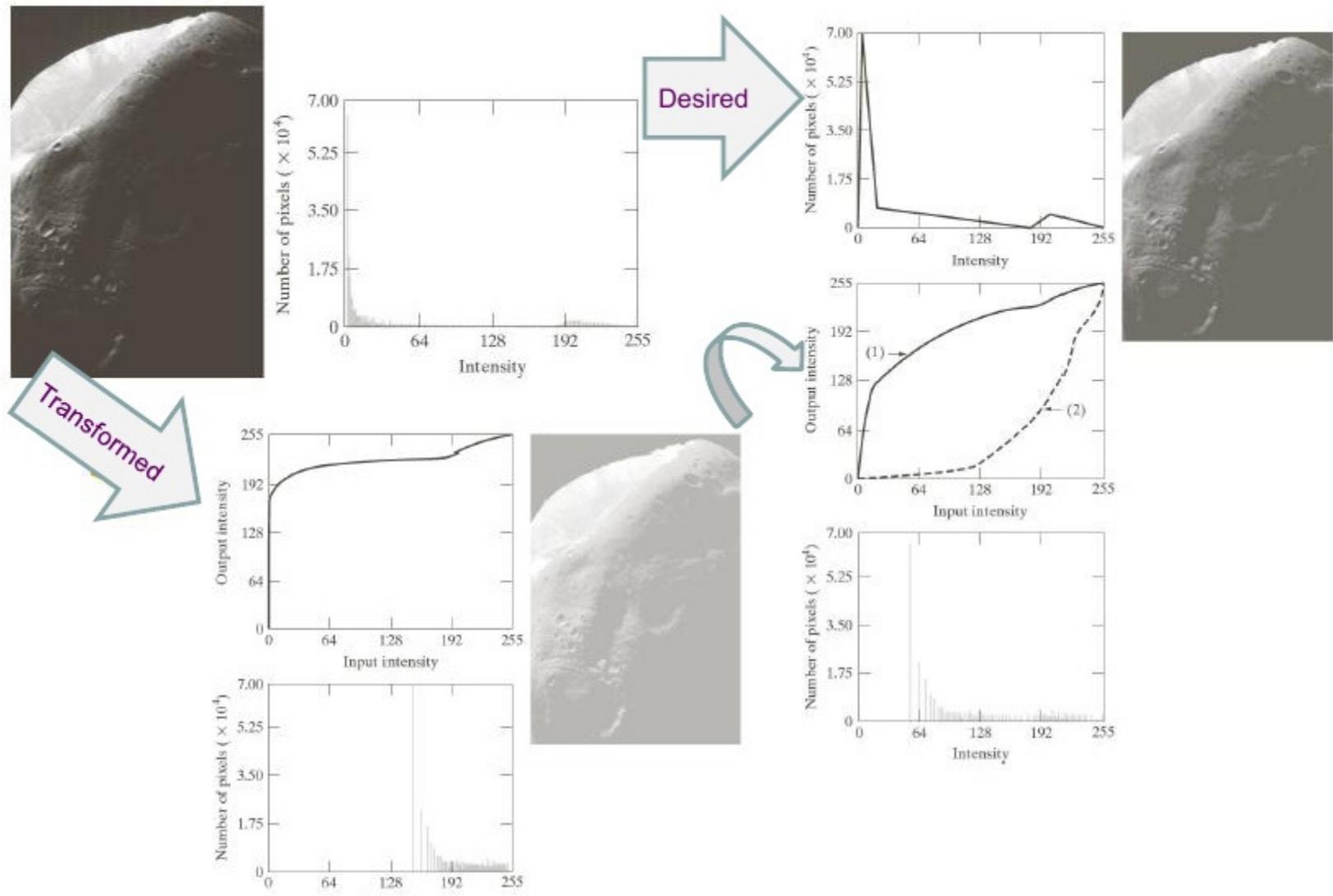
# Example 3.8



**FIGURE 3.22**

- (a) Histogram of a 3-bit image. (b) Specified histogram.  
(c) Transformation function obtained from the specified histogram.  
(d) Result of performing histogram specification. Compare (b) and (d).

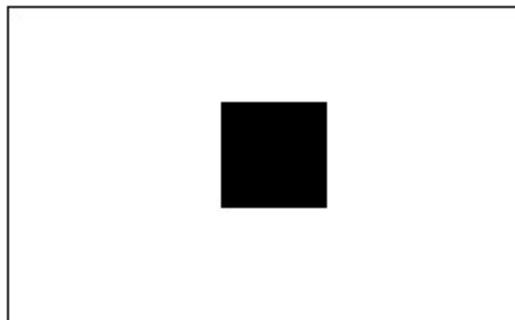
# Example 3.8



# Smoothing Spatial Filters

A human eye cannot distinguish a change in intensity of upto 5 gray levels (i.e., a pixel with gray level 250 appears identical to a pixel of gray level 255),

What is the size of the smallest square averaging mask which when applied on image will make the black square indistinguishable from the background for a human observer?



An image consists of a black square of size 5x5 pixels at the center of a white background

# Image Sharpening

Image sharpening is to enhance detail that has been blurred (perhaps due to noise or other defects, such as motion during image acquisition) or by enhancing the high-frequency components.



# Image Sharpening

surface normal discontinuity  
depth discontinuity  
Gray color discontinuity  
illumination discontinuity

## Geometric events

Surface orientation (boundary) discontinuities depth  
discontinuities color and texture discontinuities

## Non-geometric events

illumination changes secularities Shadows inter-reflections

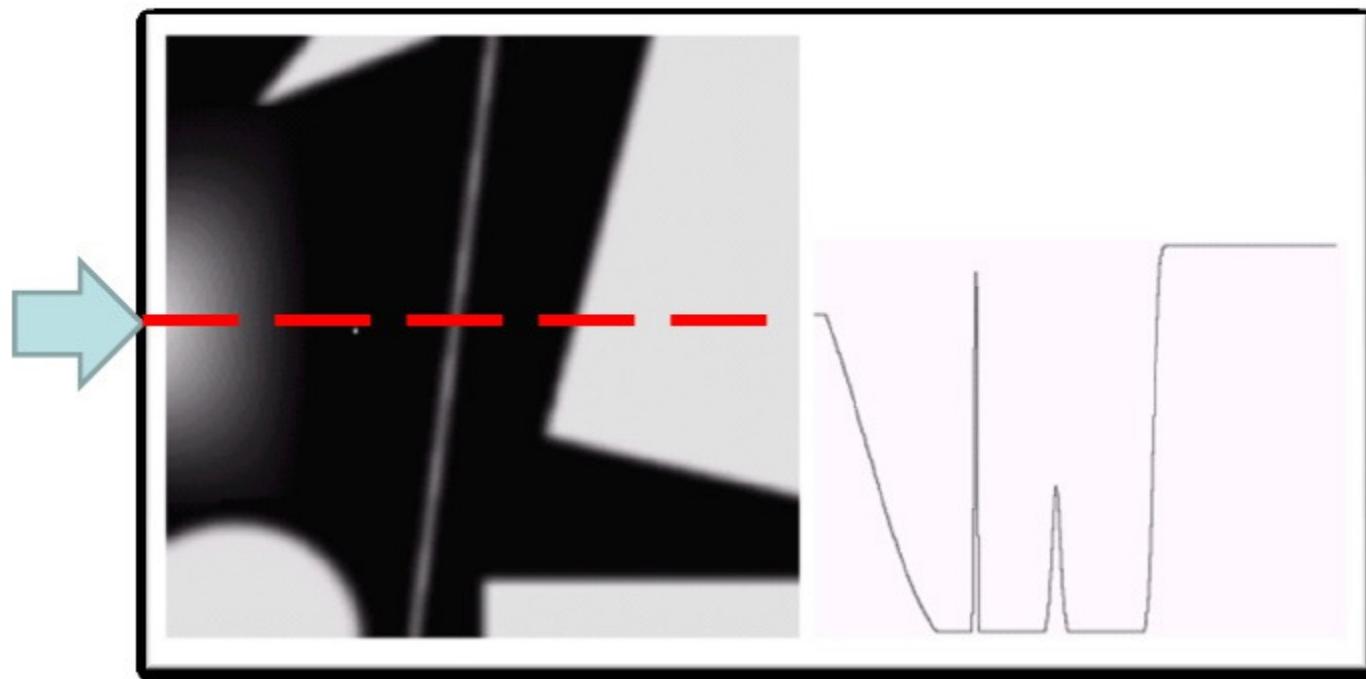
# Image Sharpening

The objective of sharpening is to highlight the details in the image that has been, either in error or a natural effect of image acquisition method.

It can be done by:

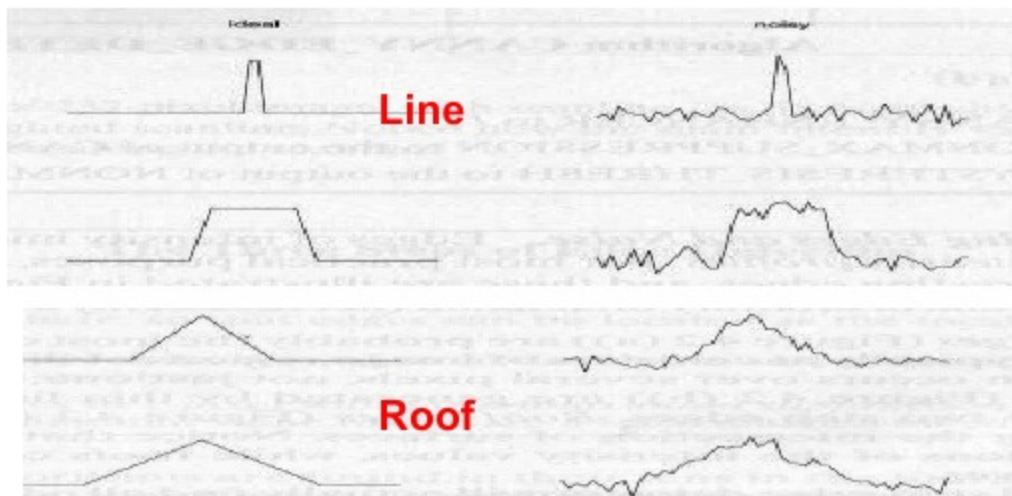
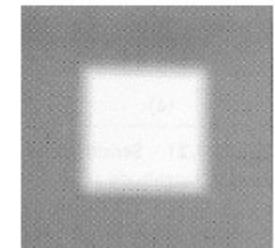
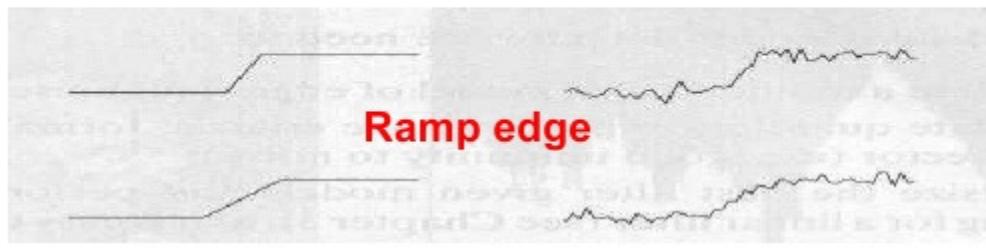
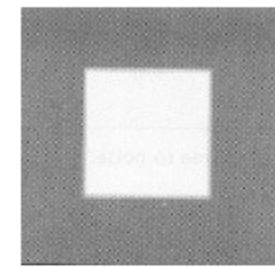
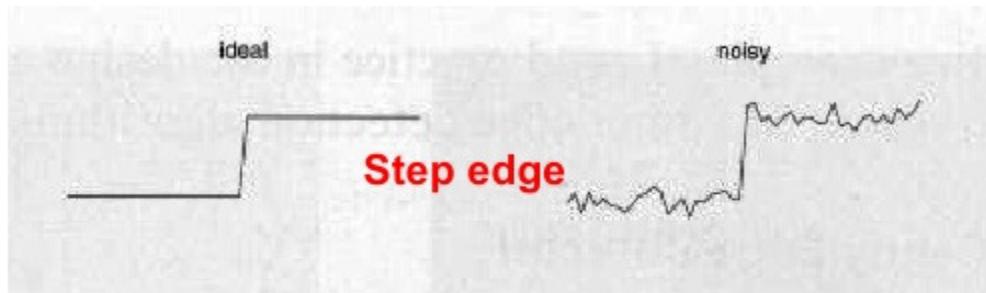
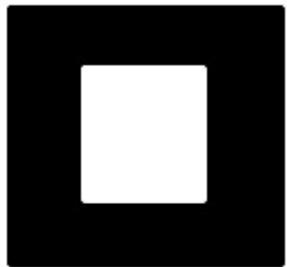
- Enhancing discontinuities in intensity
- Edges represent borders between regions on an object or in a scene and has a significant intensity change in image.

# Type of discontinuities in Intensity



**FIGURE 3.36**  
1-D digital function representing a section of a horizontal intensity profile from an image. In

# Type of discontinuities in Intensity



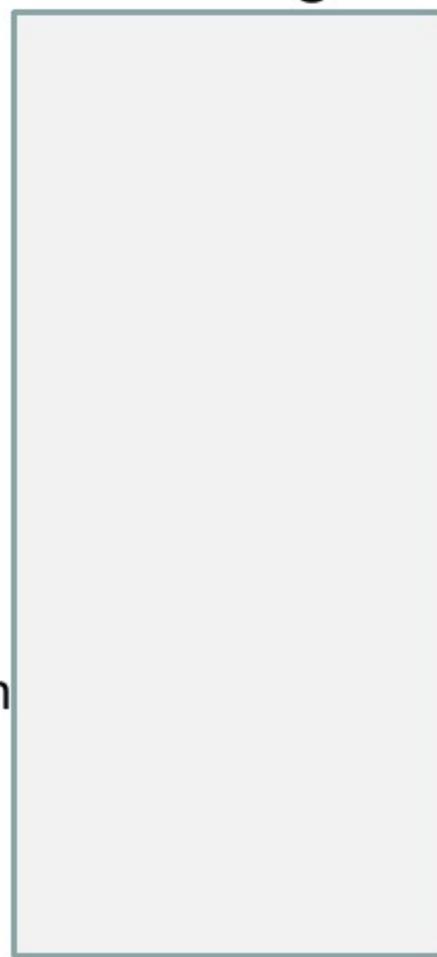
# Derivatives of intensity

Intensity discontinuity is detected by differentiating with neighboring pixels

Intensity variation

first-order derivation

Second-order derivation



The derivatives of the intensity values across the image and find the maximums.

## Derivatives of Image Intensity: Remarks

- ❖ First derivative **is non-zero along the entire ramp**, while the second derivative is **non-zero only at the onset and end of the ramp**
- ❖ For a point, response is **much stronger for the second than the first derivative.**
- ❖ For a step, the second derivative gives rise to a **double response**. This is used for **edge detection**.
- ❖ Second derivative is better suited than the first derivative for **enhancing fine detail**.

# First and second Derivative

$$f(x+h) = f(x) + h \frac{\partial f}{\partial x} + \frac{h^2}{2!} \frac{\partial^2 f}{\partial x^2} + \dots$$

How can we approximate it for a discrete function?

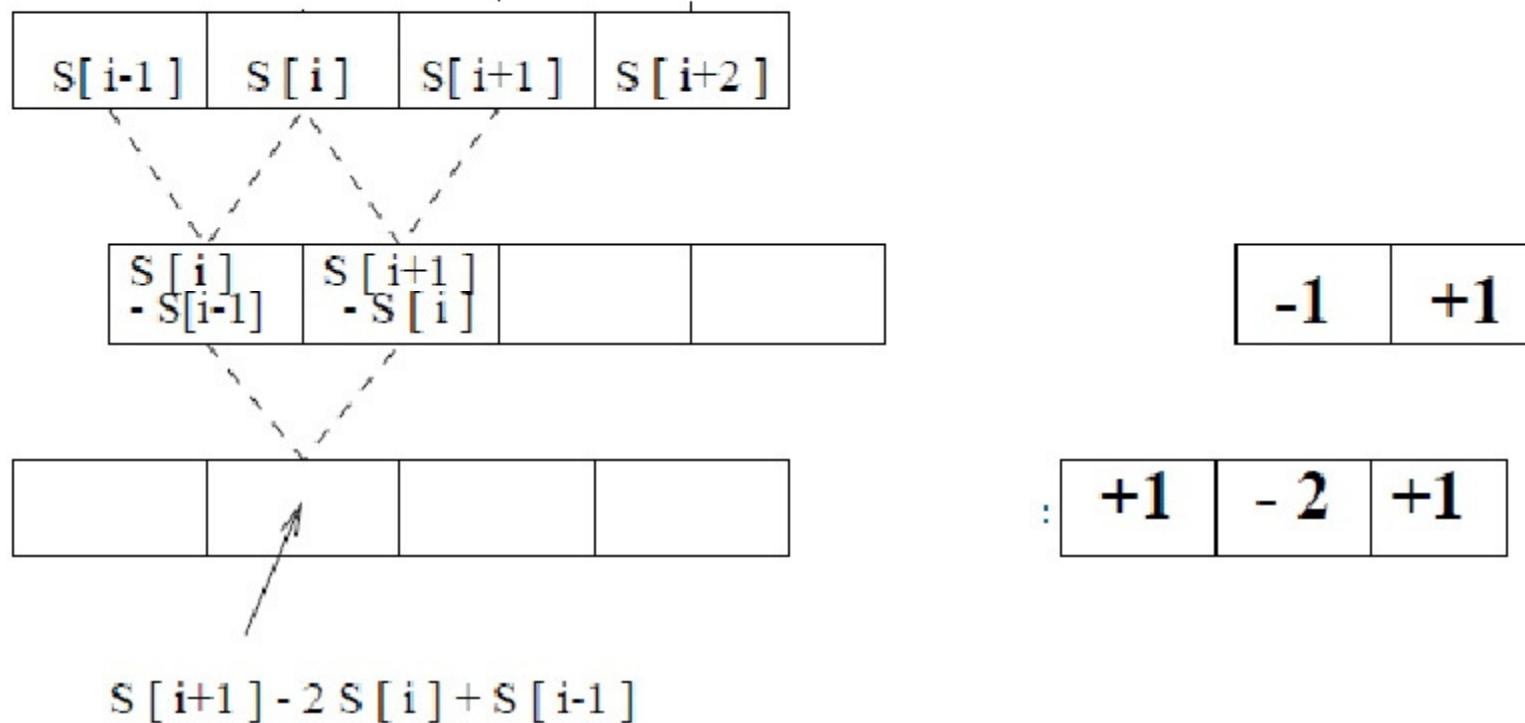
$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \left( \frac{f(x+h, y) - f(x, y)}{h} \right)$$

$$\frac{\partial f}{\partial x} = f(x+1) - f(x); \text{ for } h = 1$$

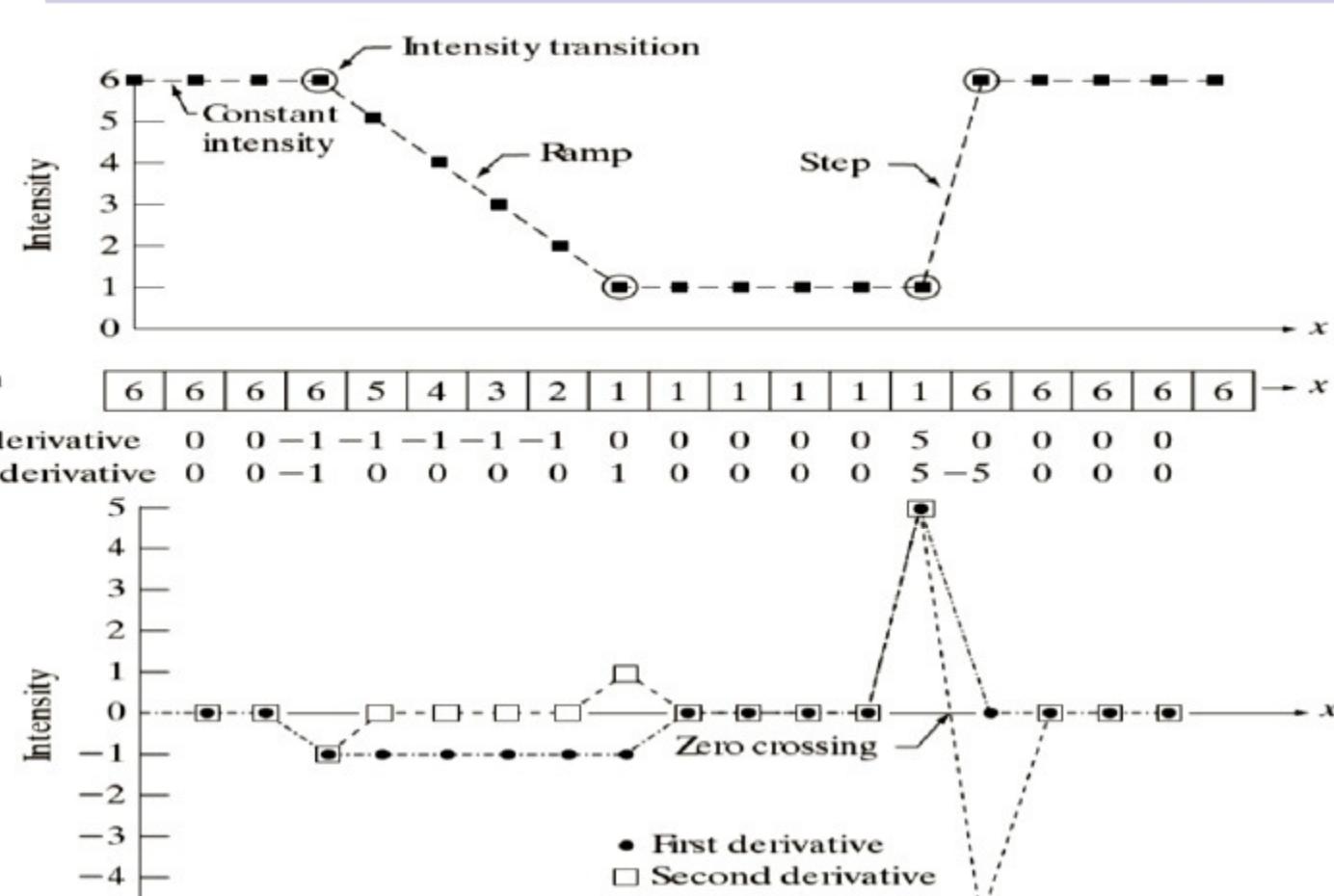
$$\frac{\partial^2 f}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$

# First and second Derivative

How can we approximate it for a discrete function?



# Derivatives of Image Intensity



$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

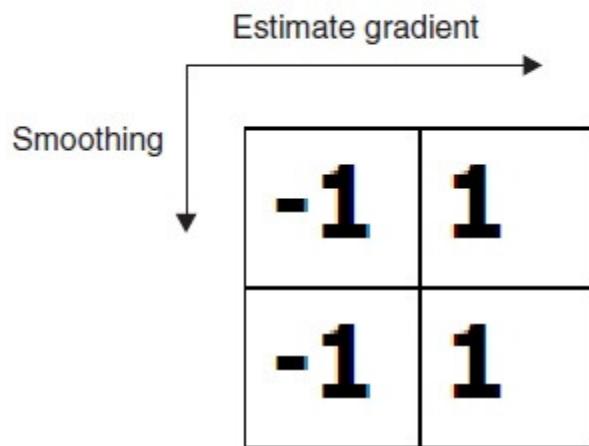
a  
b  
c

**FIGURE 3.36**  
Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

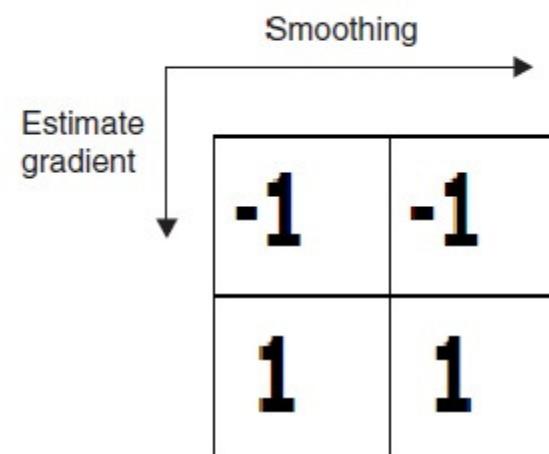
# First Derivative mask

## Approximate discrete function mask

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$



$$\frac{\partial f}{\partial y} = f(y+1) - f(y)$$



# Second Derivative mask

Approximate discrete function mask

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) - 2f(x, y) + f(x-1, y)$$

0	0	0
1	-2	1
0	0	0

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1)$$

0	1	0
0	-2	0
0	1	0

# Derivative spatial mask

- Combine horizontal and vertical edge estimates are also used.
- A number of edge operators are described by a collection of masks.
- The simplest is Roberts edge detector and **inaccurate localization**.

<b>1</b>	<b>0</b>
<b>0</b>	<b>-1</b>

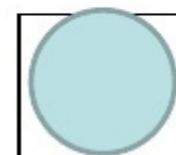
<b>0</b>	<b>1</b>
<b>-1</b>	<b>0</b>

Robert operator

# First Derivative mask

Example:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	3	3	3	3
0	0	3	3	3	3
0	0	3	3	3	3
0	0	3	3	3	3

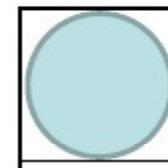
	1
-1	1

0	0	0	0	0	0
0	0	0	0	0	0
0	3	0	0	0	0
0	6	0	0	0	0
0	6	0	0	0	0
0	6	0	0	0	0

# First Derivative mask

Example:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	3	3	3	3
0	0	3	3	3	3
0	0	3	3	3	3
0	0	3	3	3	3

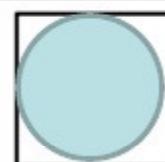
	-1
1	1

0	0	0	0	0	0
0	0	0	0	0	0
0	3	6	6	6	6
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

# Roberts edge detector

Example:

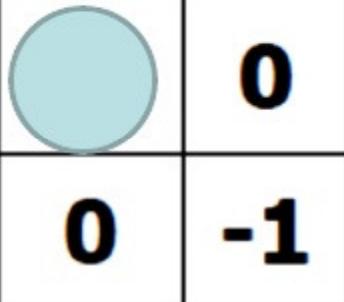
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	3	3	3	3	3
0	0	3	3	3	3	3
0	0	3	3	3	3	3
0	0	3	3	3	3	3

	1
-1	0

# Roberts edge detector

Example:

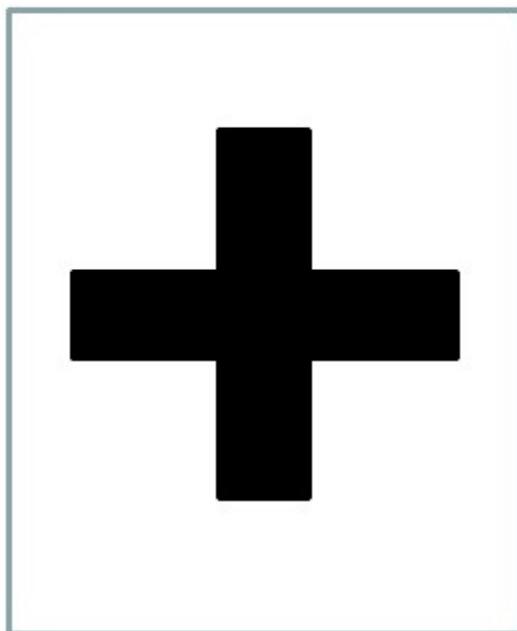
0	0	0	0	0
0	0	0	0	0
0	0	3	3	3
0	0	3	3	3
0	0	3	3	3



0	0
0	-1

# Second Derivative mask

Approximate discrete function mask

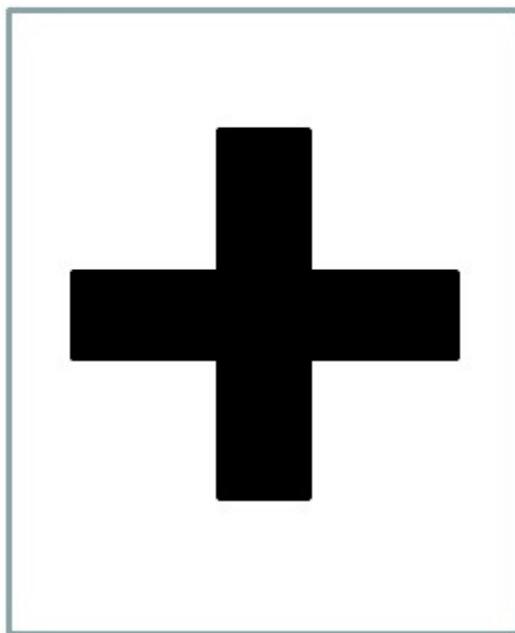


1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	1
1	1	1	0	0	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1

0	0	0
1		1
0	0	0

# Second Derivative mask

Approximate discrete function mask



1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	1
1	1	1	0	0	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1

0	1	0
0	0	0
0	1	0

# Laplacian Derivative operator

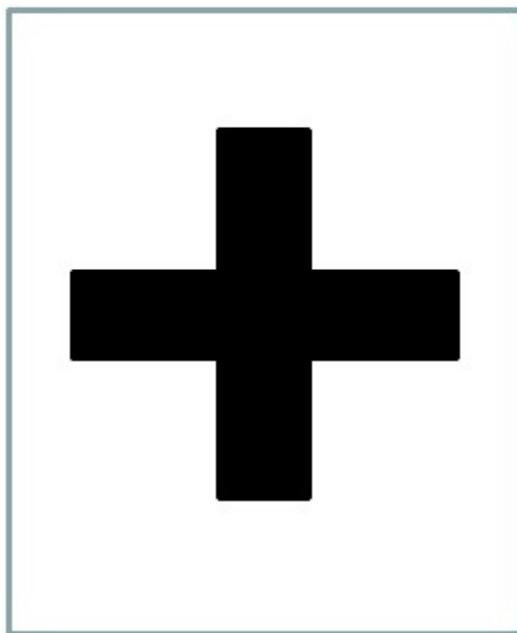
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

0	1	0
1		1
0	1	0

# Second Derivative mask

Approximate discrete function mask



1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	1
1	1	1	0	0	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1

0	1	0
1		1
0	1	0

# Sharpening Spatial Filtering

The mask has a high positive component at the center

0	1	0
1	-4	1
0	1	0

Since the sum of all the weights is zero, the resulting signal will have a zero DC value.

# Laplacian Derivative operator

- The Laplacian is derivative operator, is used to highlights **gray-level discontinuities** in an image and **de-emphasizes regions with slowly varying gray levels.**
- Background features can be recovered** while still preserving the sharpening effect of the Laplacian operation by simply adding **the original and Laplacian images.**

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the} \\ & \text{Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of} \\ & \text{the Laplacian mask is positive} \end{cases}$$

# Sharpening by Laplacian

$$g(x, y) = f(x, y) + c \nabla^2 f(x, y)$$

$$c = -1$$

<table border="1"><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>-4</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr></table>	0	1	0	1	-4	1	0	1	0	<table border="1"><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>-8</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	-8	1	1	1	1
0	1	0																	
1	-4	1																	
0	1	0																	
1	1	1																	
1	-8	1																	
1	1	1																	
<table border="1"><tr><td>0</td><td>-1</td><td>0</td></tr><tr><td>-1</td><td>4</td><td>-1</td></tr><tr><td>0</td><td>-1</td><td>0</td></tr></table>	0	-1	0	-1	4	-1	0	-1	0	<table border="1"><tr><td>-1</td><td>-1</td><td>-1</td></tr><tr><td>-1</td><td>8</td><td>-1</td></tr><tr><td>-1</td><td>-1</td><td>-1</td></tr></table>	-1	-1	-1	-1	8	-1	-1	-1	-1
0	-1	0																	
-1	4	-1																	
0	-1	0																	
-1	-1	-1																	
-1	8	-1																	
-1	-1	-1																	

$$c = 1$$

a	b
c	d

**FIGURE 3.37**

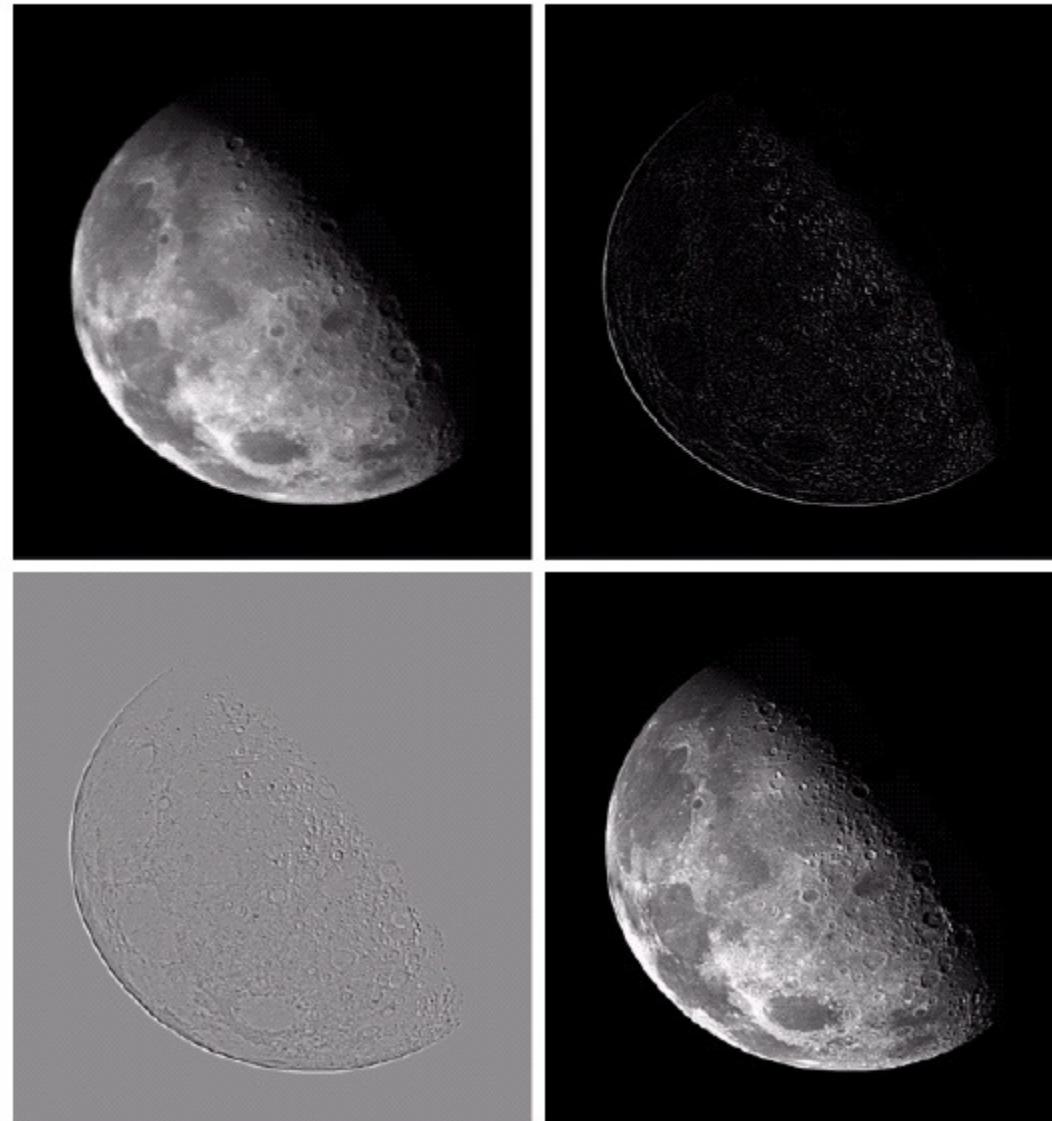
- (a) Filter mask used to implement Eq. (3.6-6).
- (b) Mask used to implement an extension of this equation that includes the diagonal terms.
- (c) and (d) Two other implementations of the Laplacian found frequently in practice.

# Application Laplacian operator

a	b
c	d

**FIGURE 3.40**

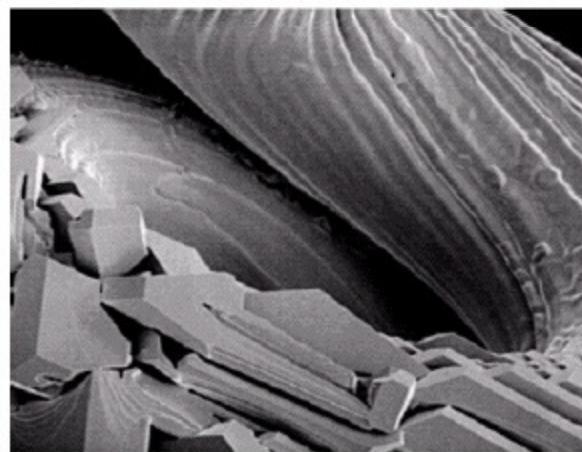
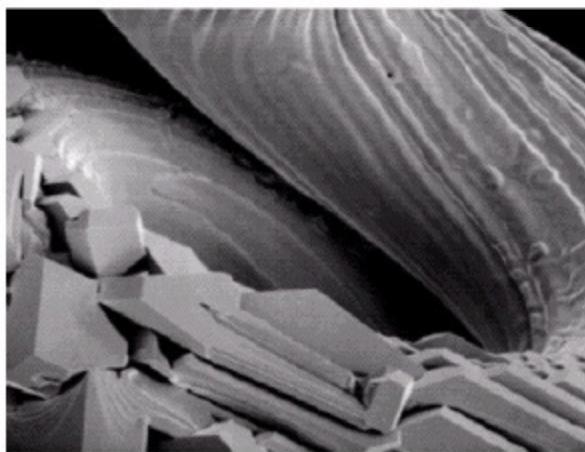
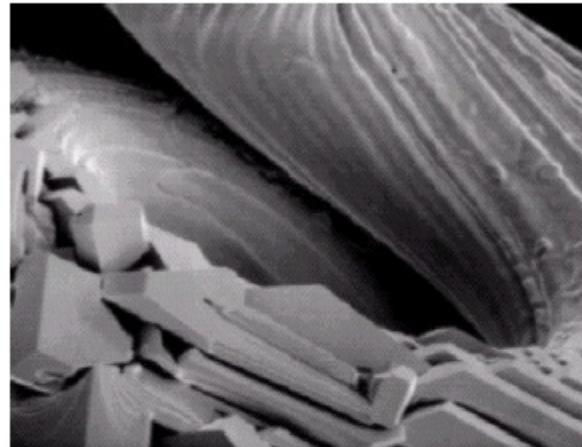
- (a) Image of the North Pole of the moon.
- (b) Laplacian-filtered image.
- (c) Laplacian image scaled for display purposes.
- (d) Image enhanced by using Eq. (3.7-5). (Original image courtesy of NASA.)



# Application Composite Laplacian

0	-1	0
-1	5	-1
0	-1	0

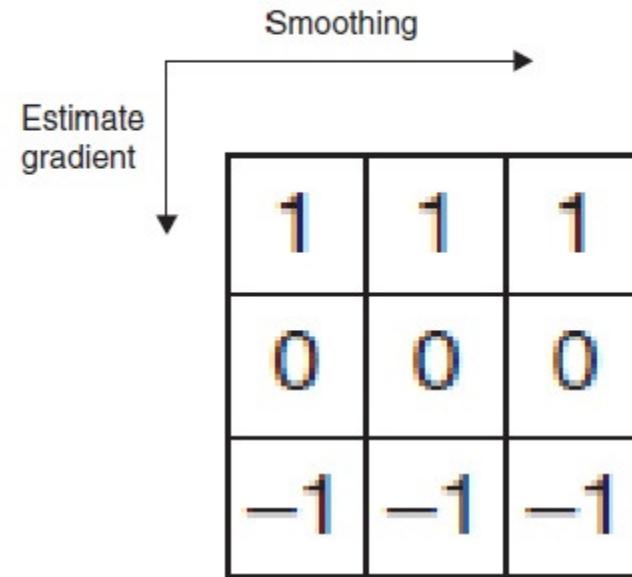
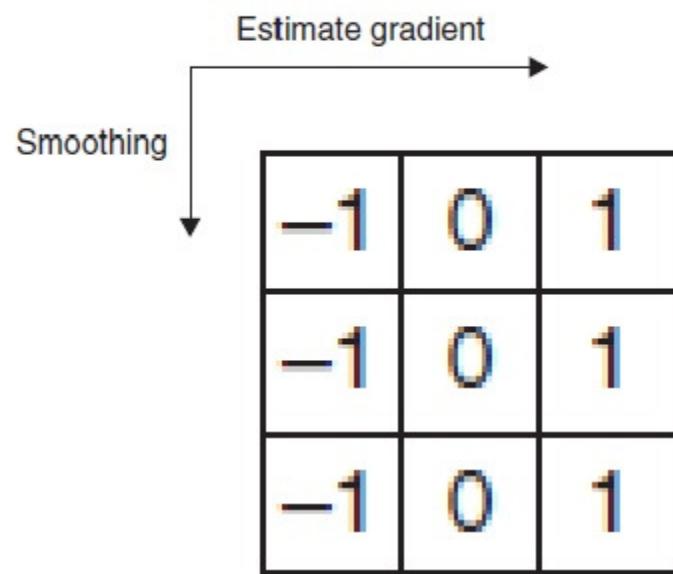
-1	-1	-1
-1	9	-1
-1	-1	-1



a b c  
d e

**FIGURE 3.41** (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

# Prewitt operator



# Robert and Sobel Operator

a	
b	c
d	e

**FIGURE 3.41**

A  $3 \times 3$  region of an image (the  $z$ s are intensity values).

(b)–(c) Roberts cross gradient operators.

(d)–(e) Sobel operators. All the mask coefficients sum to zero, as expected of a derivative operator.

$z_1$	$z_2$	$z_3$
$z_4$	$z_5$	$z_6$
$z_7$	$z_8$	$z_9$

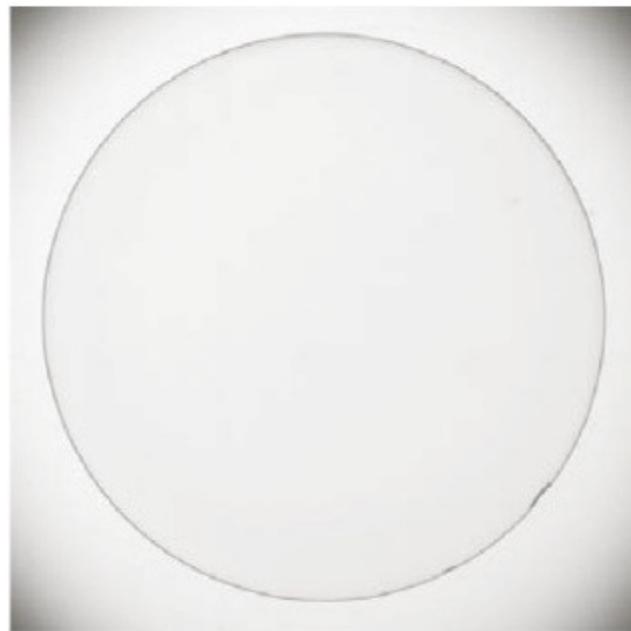
-1	0
0	1
1	0

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

# Sobel Operator

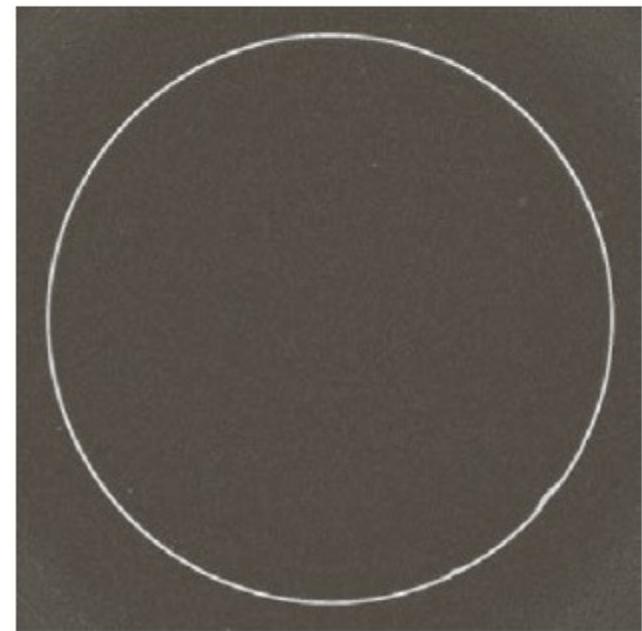
-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1



a b

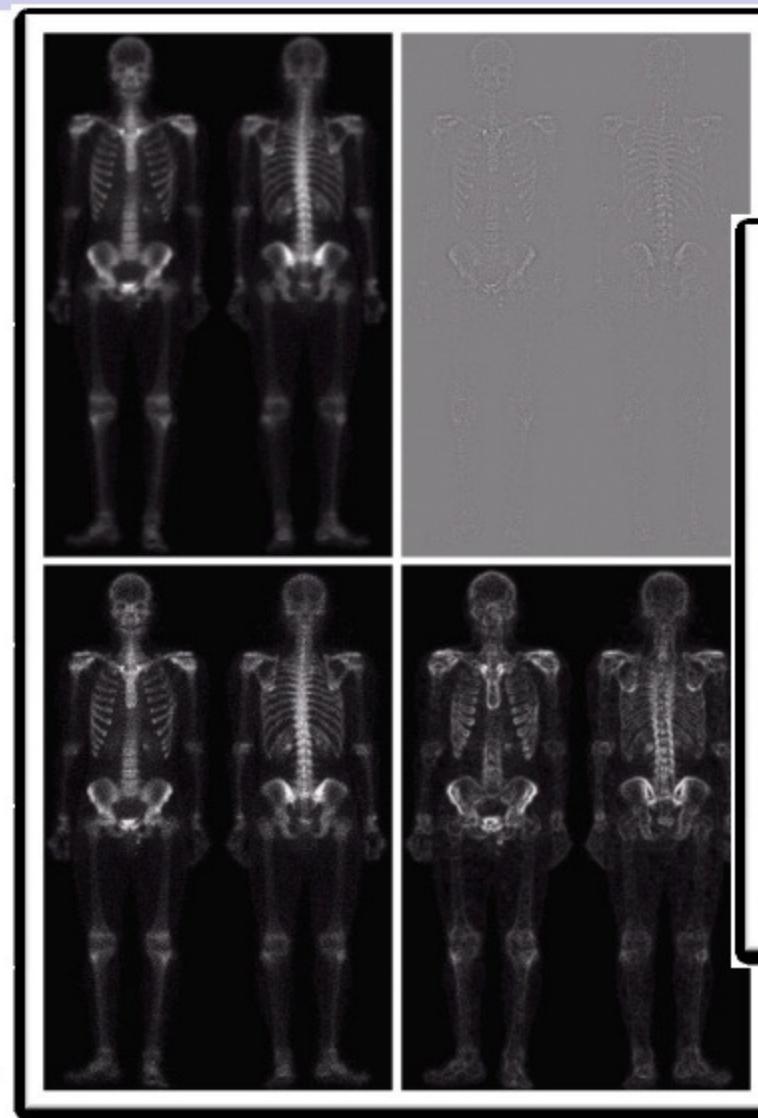
**FIGURE 3.42**

(a) Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock).  
(b) Sobel gradient.  
(Original image courtesy of Pete Sites, Perceptics Corporation.)



# Combining Spatial Enhancement

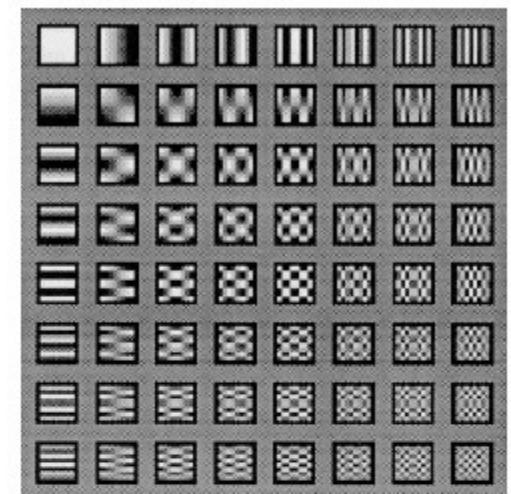
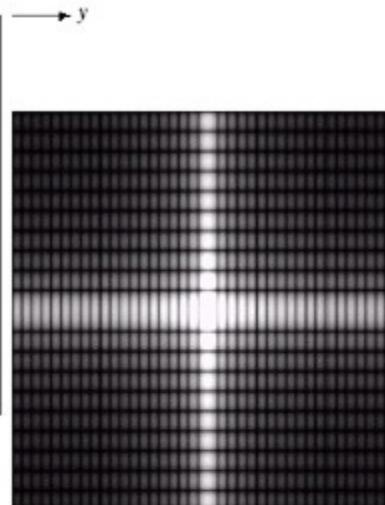
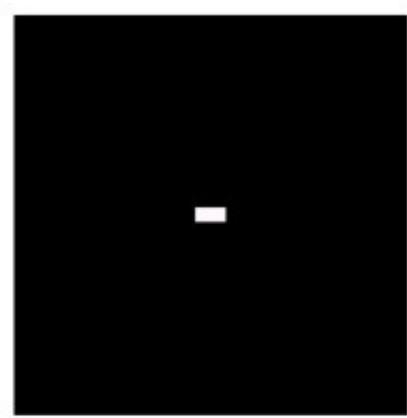
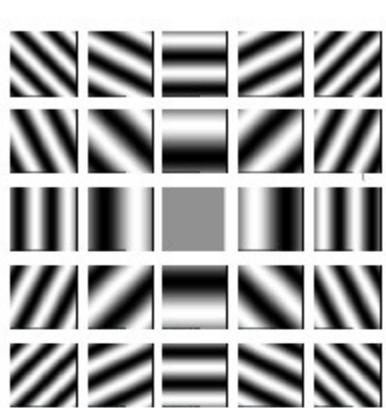
- Laplacian to highlight fine detail
- Gradient to enhance prominent edges
- Smoothed version of the gradient image used to mask the Laplacian image
- Increase the dynamic range of the gray levels by using a gray-level transformation



a  
b  
c  
d

**FIGURE 3.46**  
(a) Image of whole body bone scan.  
(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).

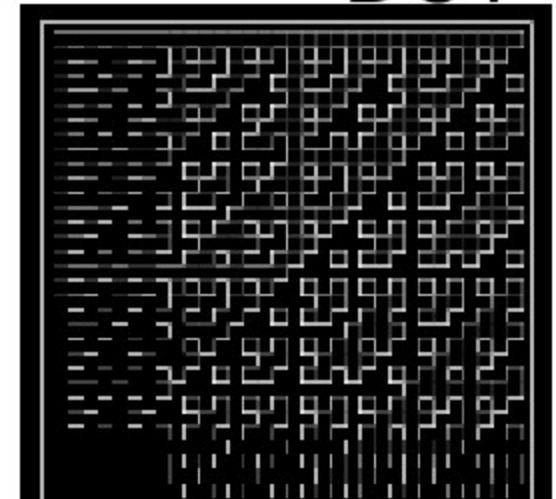
# Image Transforms



DCT

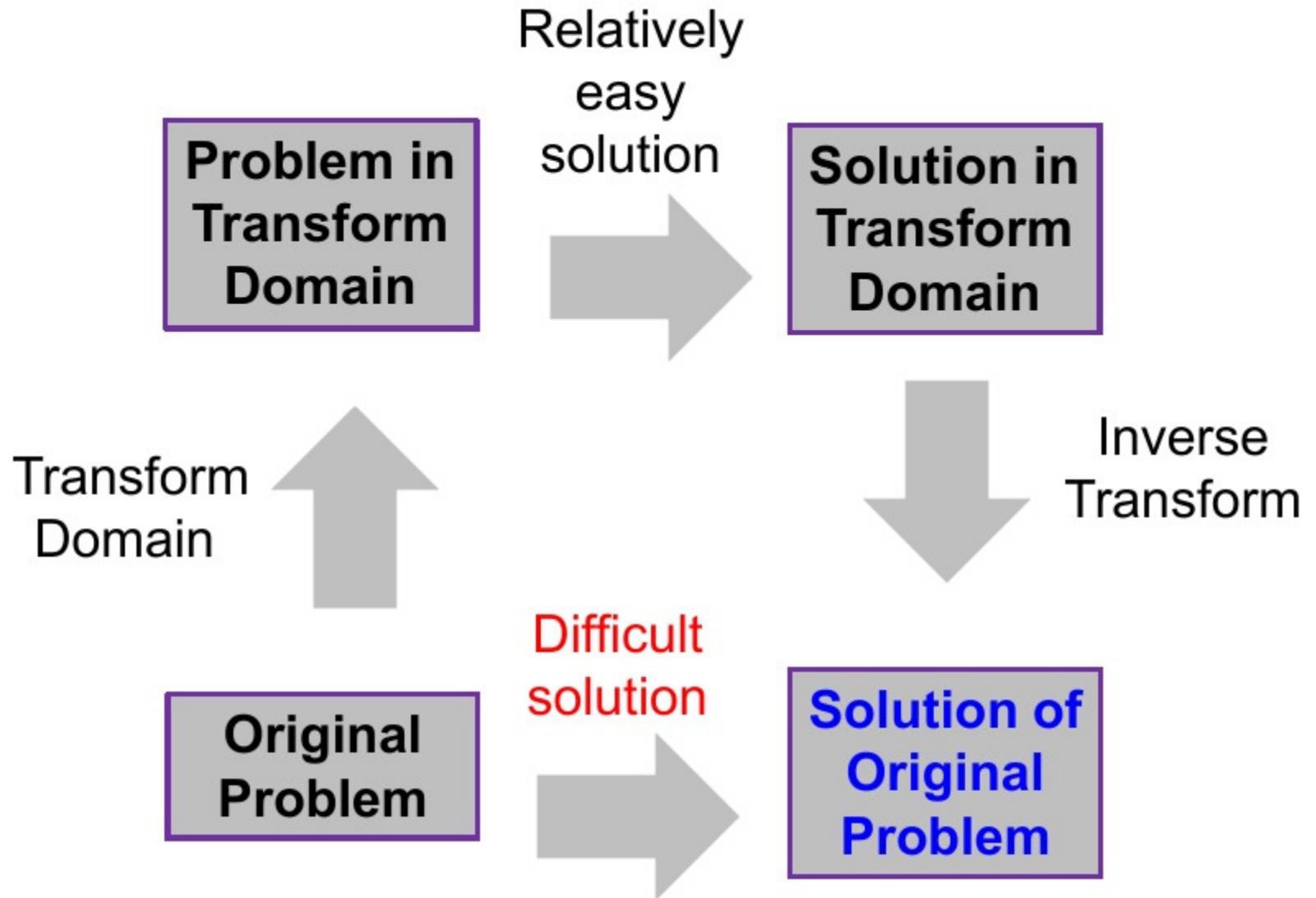


DWT



WHT

# Why do we need Image transform?



# Why do we need Image transform?

- Better image processing-
  - Take into account long-range correlations in space
  - Conceptual insights in spatial-frequency information.
- Fast computation
- Alternative representation-
  - Obtain transformed data as compressed, measurement in radiology images, Biometric signature etc.
- Efficient storage and transmission-
  - Energy compaction
  - Pick a few “representatives”
  - Just store / send the “contribution” from representatives

# Desired Properties of Transform

- It should be reversible.
- Localization of the essential part of the signal energy in a small number of transform coefficients.
- Transform coefficients should be uncorrelated.
  - Transform should be orthonormal.
  - Low computational complexity of computing coefficients etc.

# Image Transforms: DFT

Advantage of working in the frequency domain-

- Frequency Coefficients are segregated into low and high image frequencies.
- Our visual system is less sensitive to distortion around edges
- Transition associated with the edge masks our ability to perceive the noise.

# Image Transforms: DFT

The two-dimensional discrete Fourier transform and its inverse

- Fourier transform (case)

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$$

for  $u = 0, 1, 2, \dots, M-1, v = 0, 1, 2, \dots, N-1$

- Inverse Fourier transform:

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}$$

for  $x = 0, 1, 2, \dots, M-1, y = 0, 1, 2, \dots, N-1$

$u, v$  : the transform or frequency variables

$x, y$  : the spatial or image variables

# Image Transforms: DFT

**Fourier spectrum:**

$$|F(u, v)| = \left[ R^2(u, v) + I^2(u, v) \right]^{1/2}$$

**Phase:**

$$\phi(u, v) = \tan^{-1} \left[ \frac{I(u, v)}{R(u, v)} \right]$$

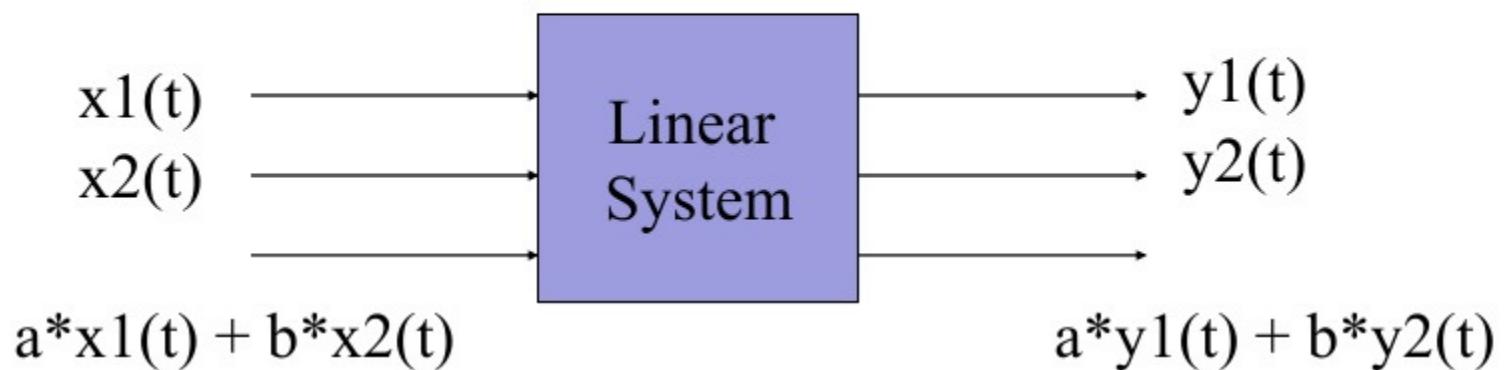
**Power spectrum:**

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$$

# Properties of DFT

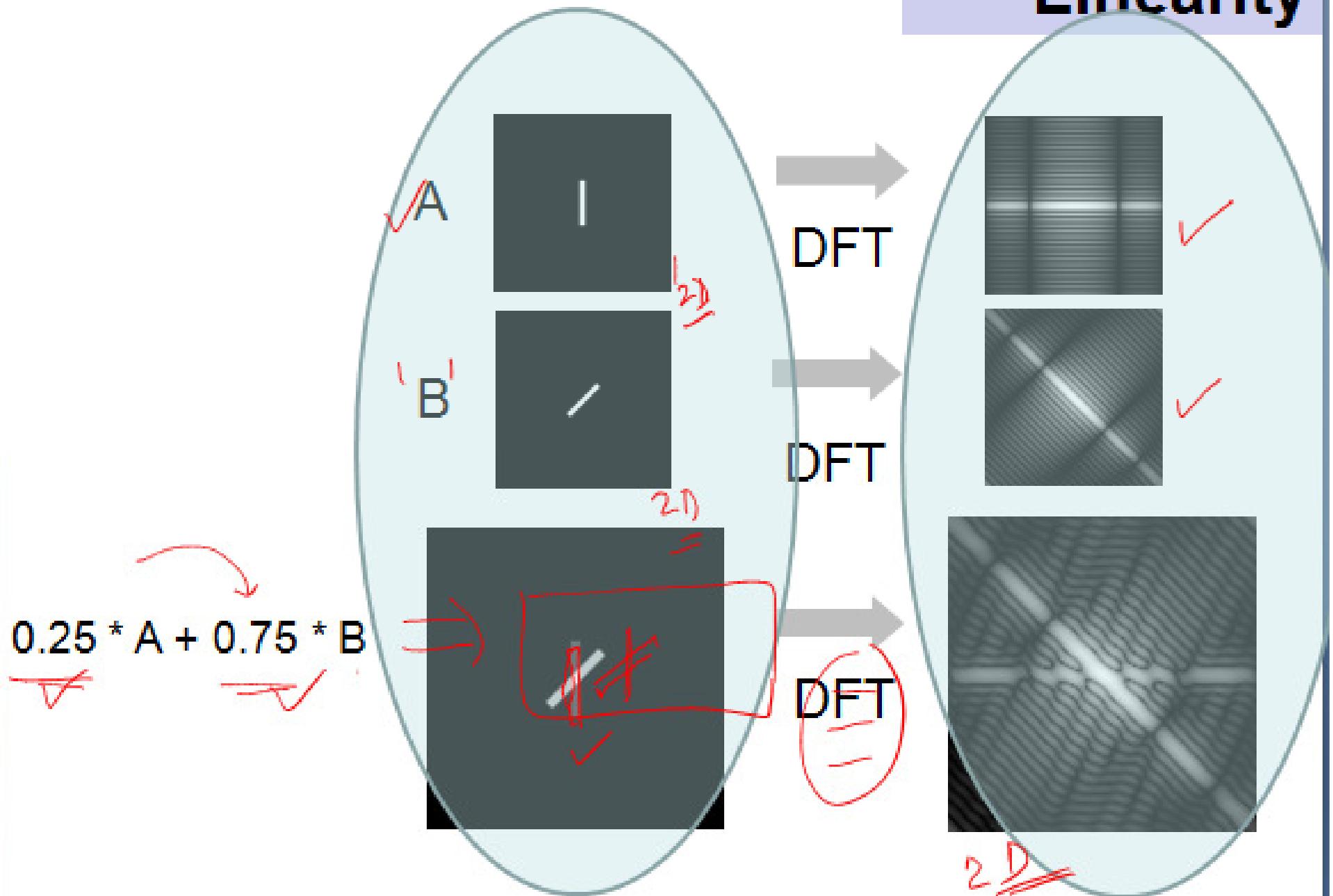
- Linearity (distributive && scaling)
- Separability
- Translation
- Periodicity
- Conjugate symmetry
- Rotation
- Convolution
- Correlation
- Sampling

# Linearity



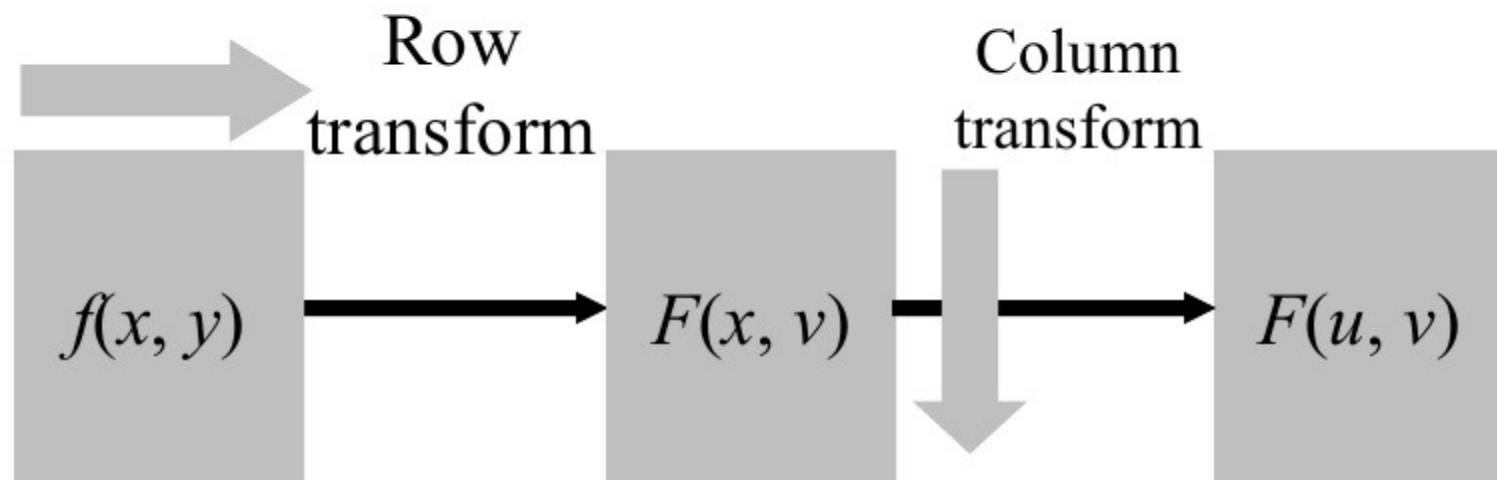
FT is a linear image processing method

# Linearity

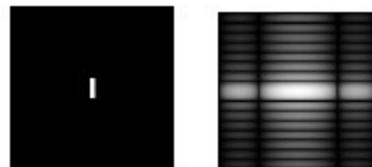


# Separability

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi(ux/M + vy/N)]$$

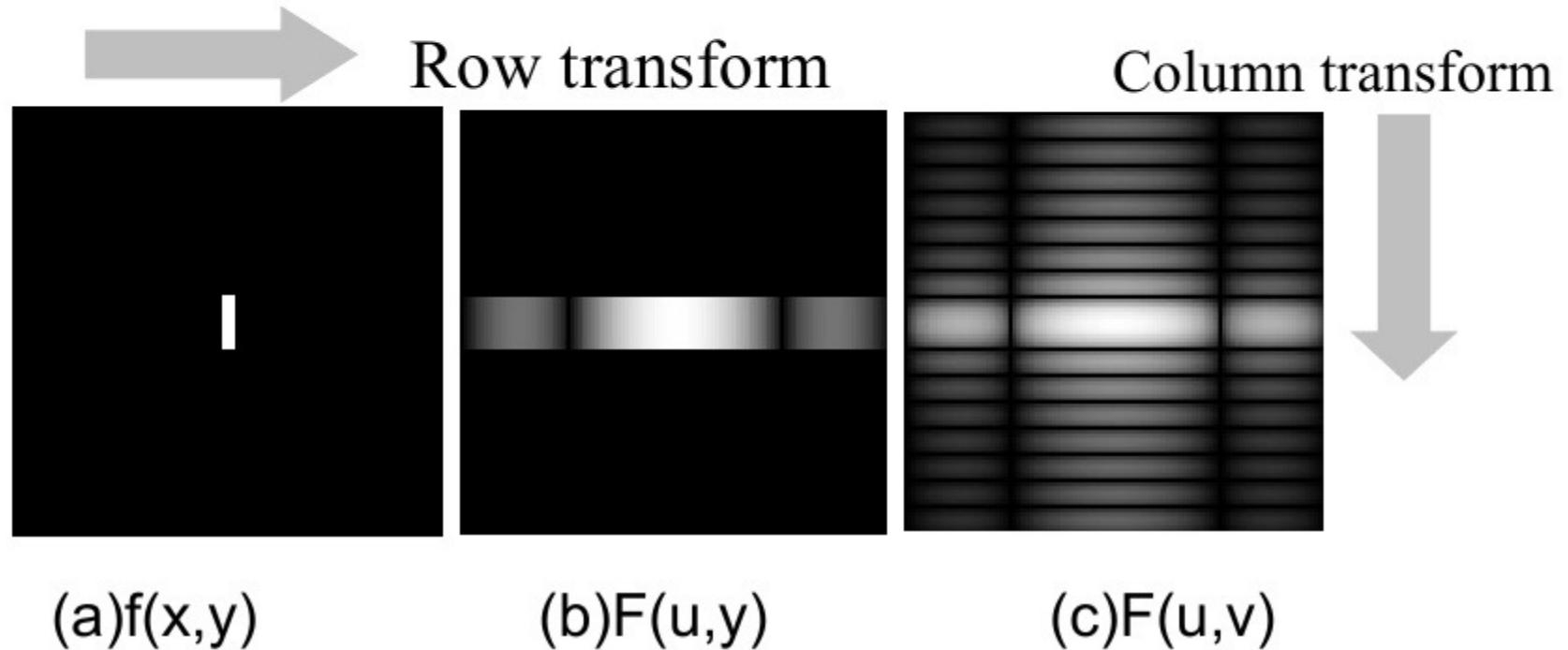


# Separability

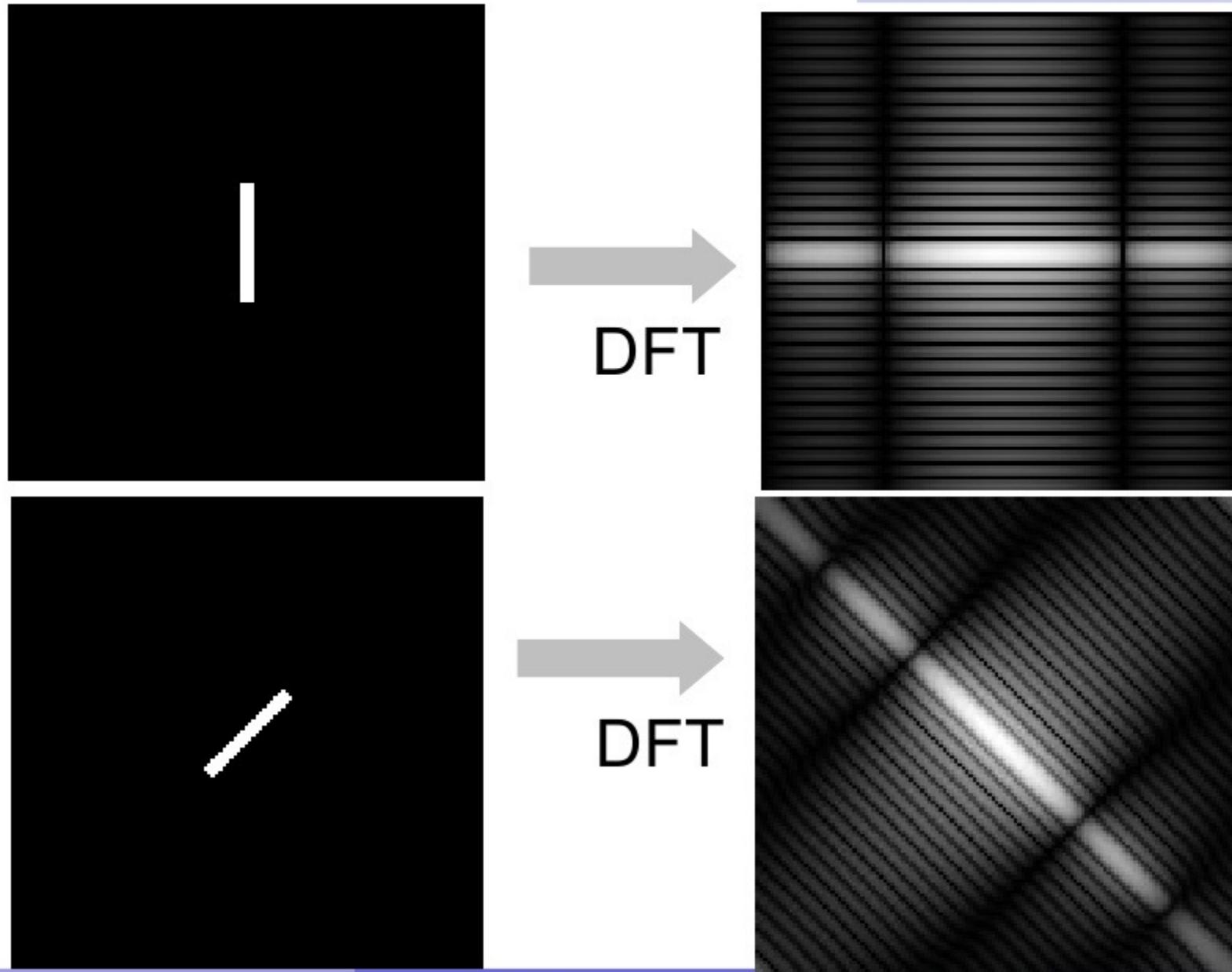


The 2D DFT  $\mathbf{F}(u,v)$  can be obtained by

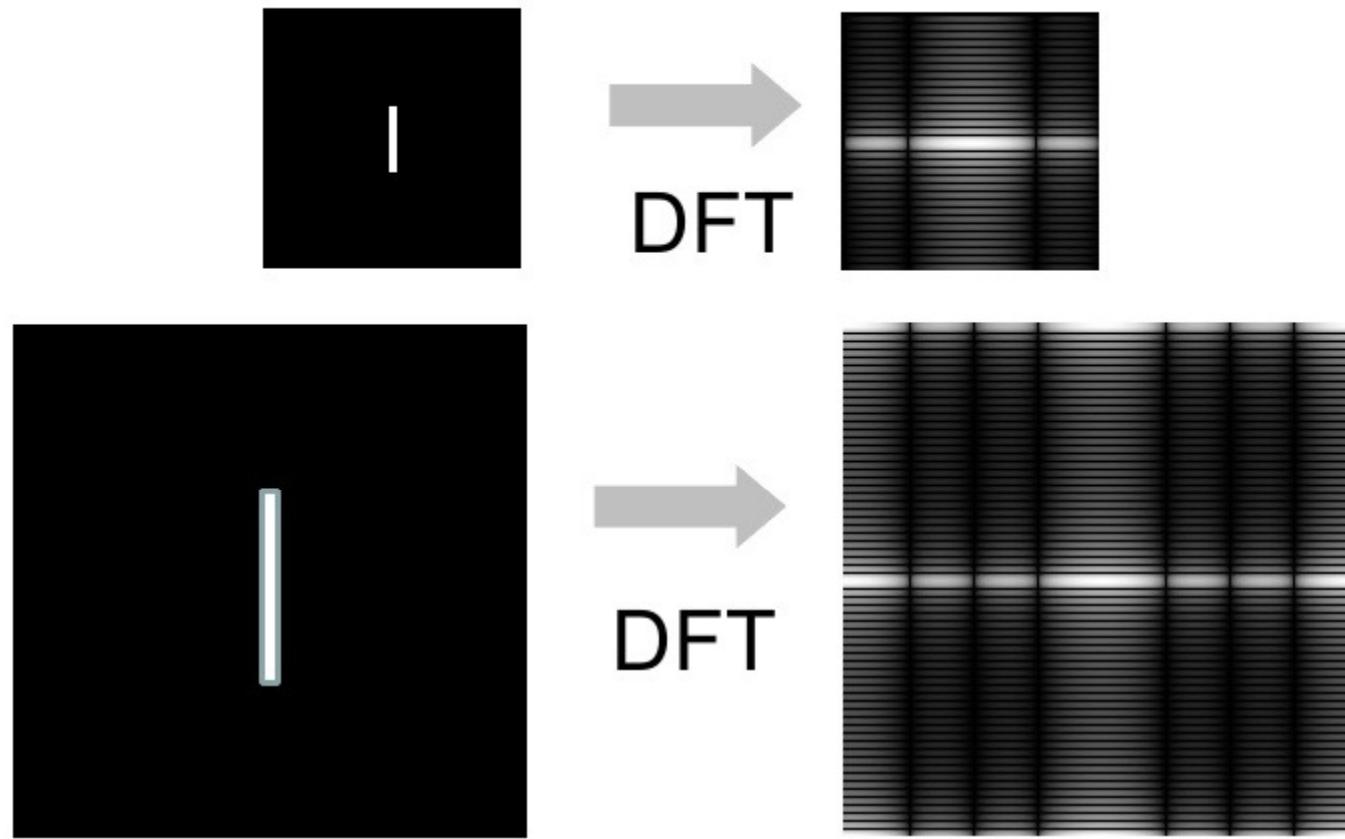
1. taking the 1D DFT of every row of image  $f(x,y) \rightarrow F(u,y)$ ,
2. taking the 1D DFT of every column of  $F(u,y) \rightarrow F(u,v)$



# Rotation



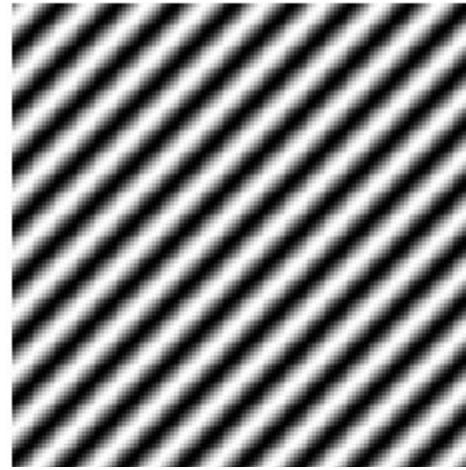
# Expansion



Expanding the original image by a factor of  $n$  ( $n=2$ ), filling the empty new values with zeros, results in the same DFT.

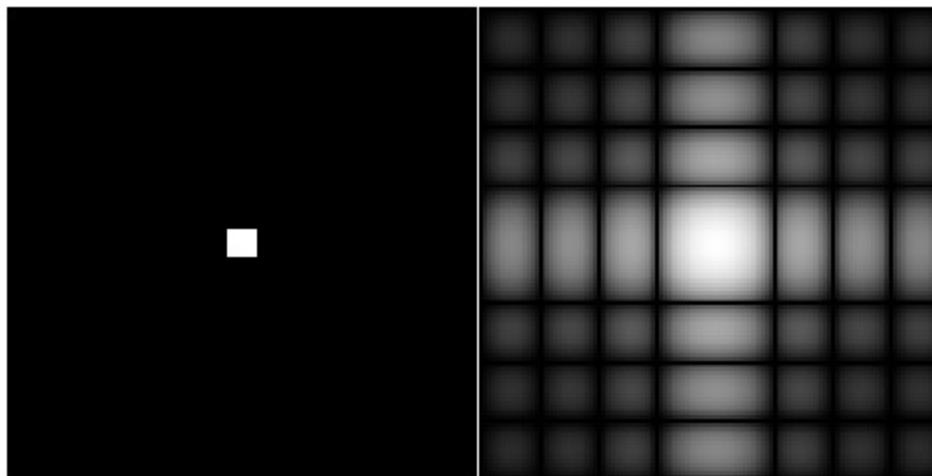
# Functions

Sine wave



Its DFT

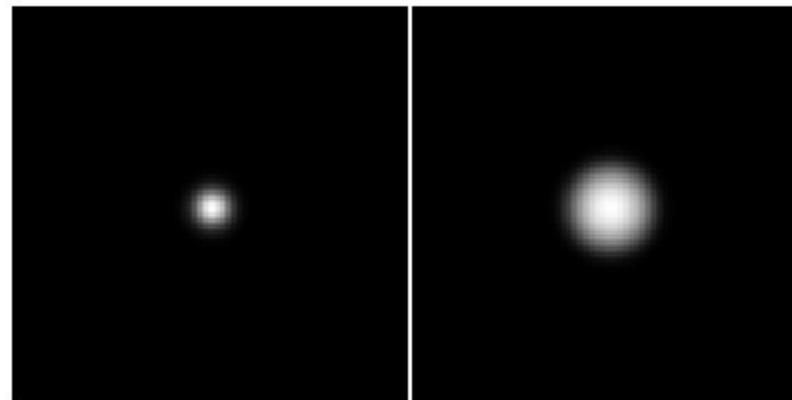
Rectangle



Its DFT

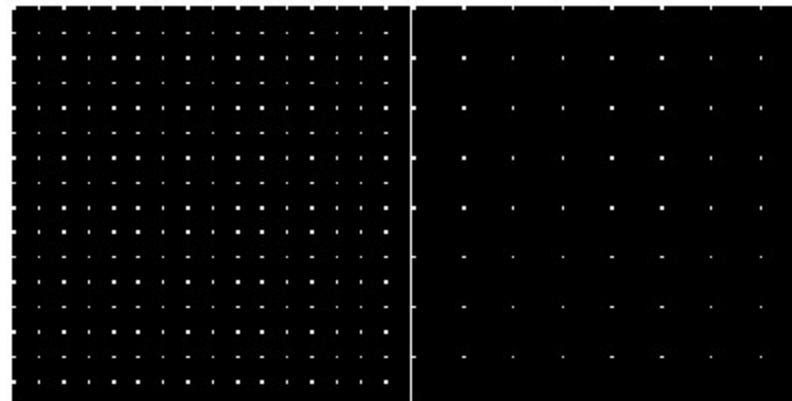
# Functions

2D Gaussian



Its DFT

Impulses



Its DFT

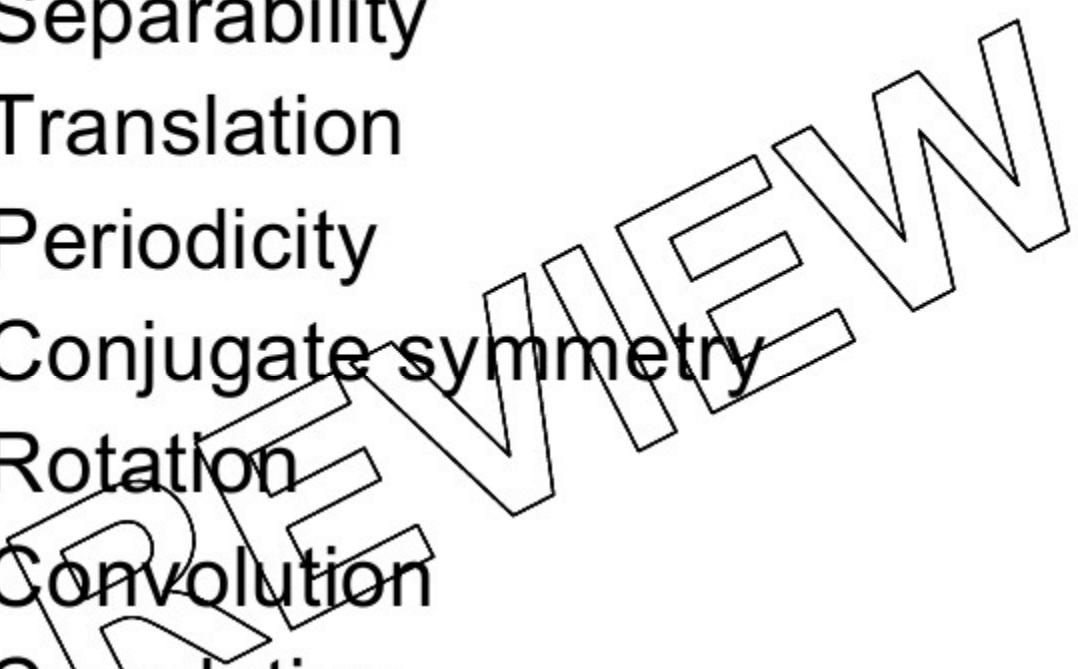
# Averaging

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi(ux/M + vy/N)]$$

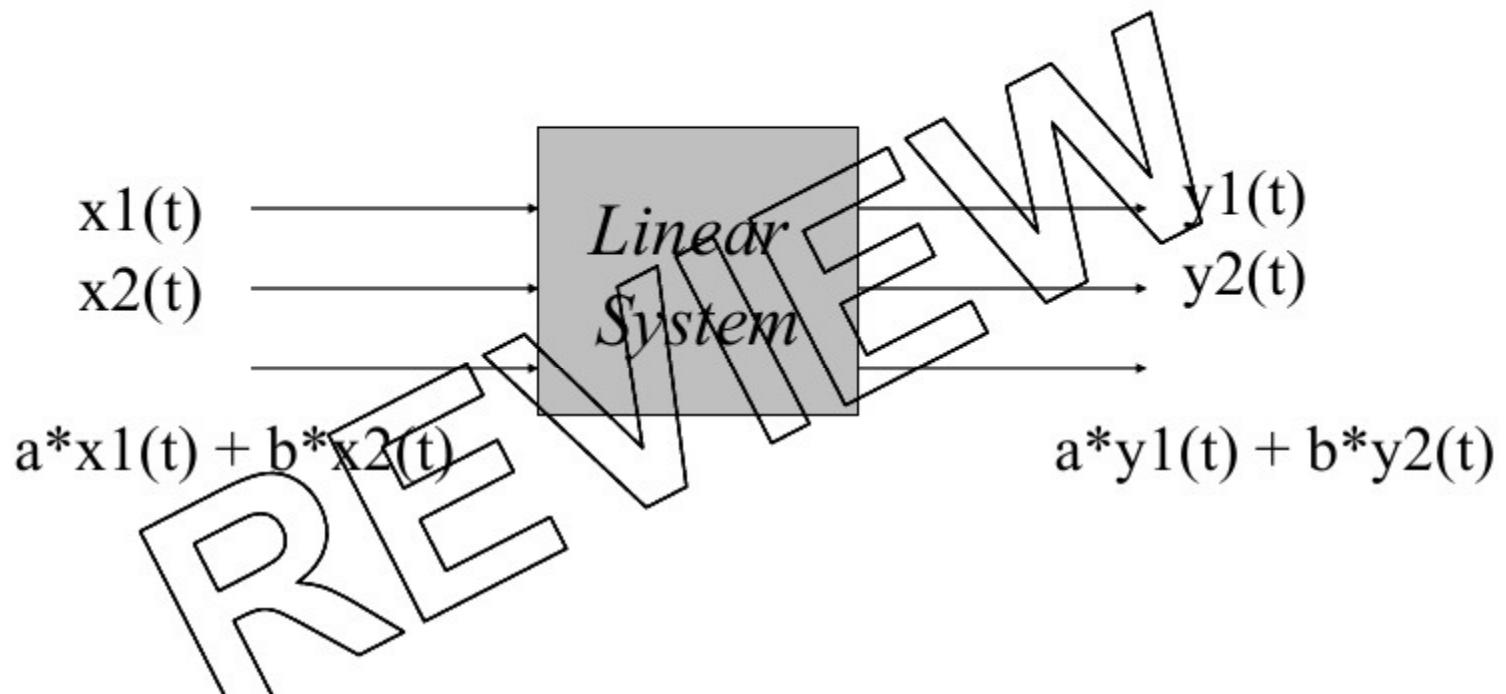
$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi(ux/M + vy/N)]$$

$$\bar{f}(x, y) = F(0, 0)$$

# Properties of DFT

- Linearity (distributive && scaling)
  - Separability
  - Translation
  - Periodicity
  - Conjugate symmetry
  - Rotation
  - Convolution
  - Correlation
  - Sampling
- 

# Linearity



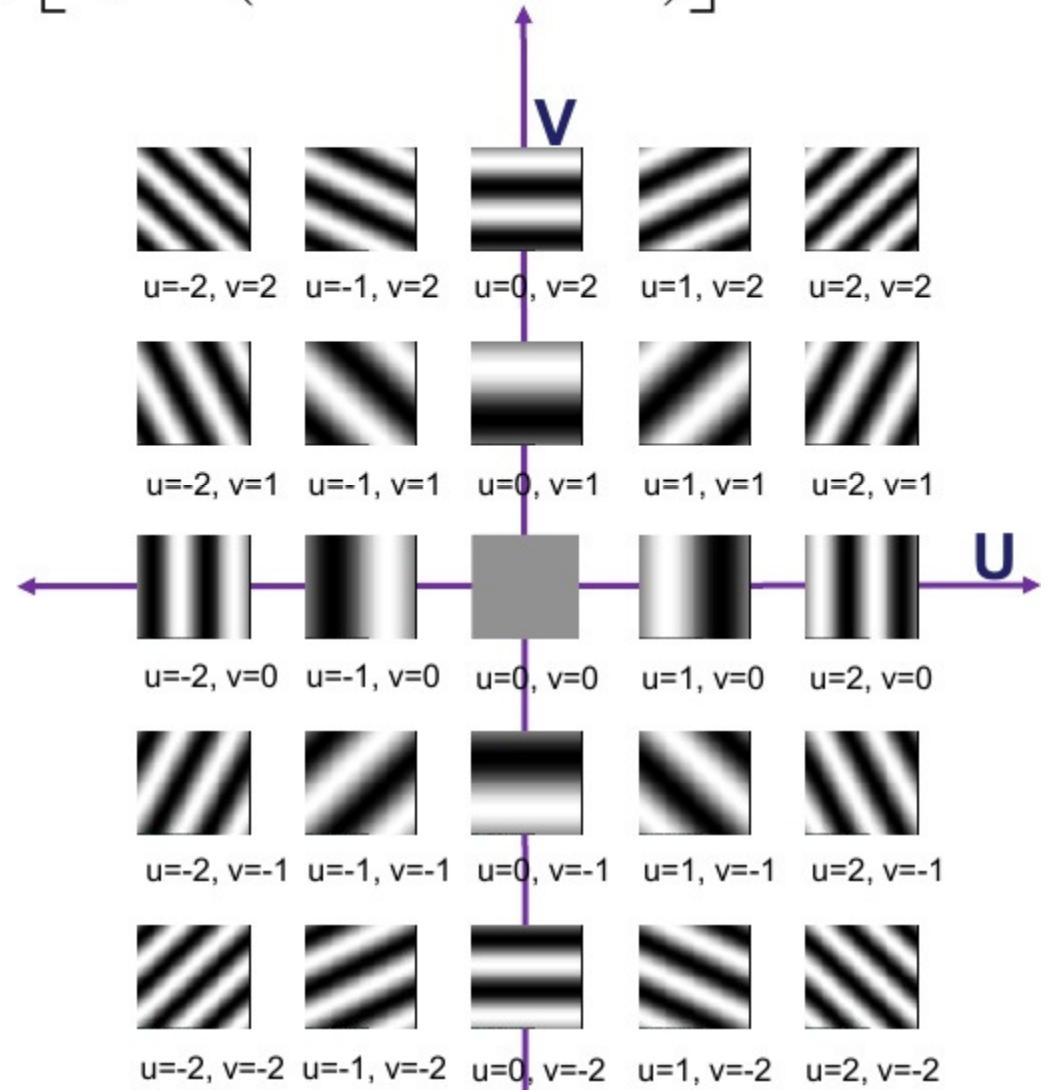
FT is a linear image processing method

# Implementation: DFT

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp\left[-j2\pi(ux/M + vy/N)\right]$$

$$F_{u,v} = \frac{1}{MN} \sum_{x=-M}^M \sum_{y=-N}^N e^{-j2\pi(ux/M + vy/N)}$$

**Remark:**  
 $F_{u,v}$  and  $F_{-u,-v}$  have similar frequencies but inverted shifts



# Visualization of Fourier Image

- $F(u,v)$  is a Fourier transform of  $f(x,y)$  and it has complex entries.

Matlab:  $F = \text{fft2}(f);$

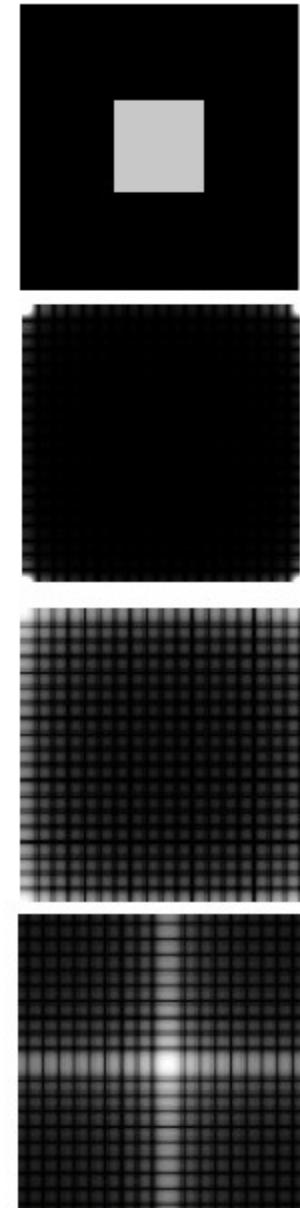
- In order to display the Fourier Spectrum  $|F(u,v)|$

- Reduce dynamic range of  $|F(u,v)|$  by displaying the log:

$$D = \log(1 + \text{abs}(F));$$

- Cyclically rotate the image so that  $F(0,0)$  is in the center:

$\text{fftshift}(D);$



# Visualization of Fourier Image

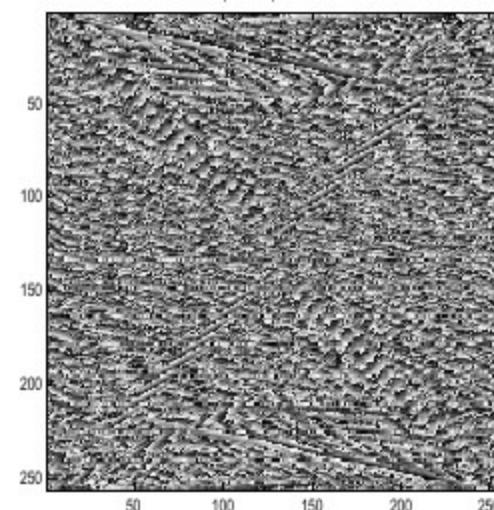
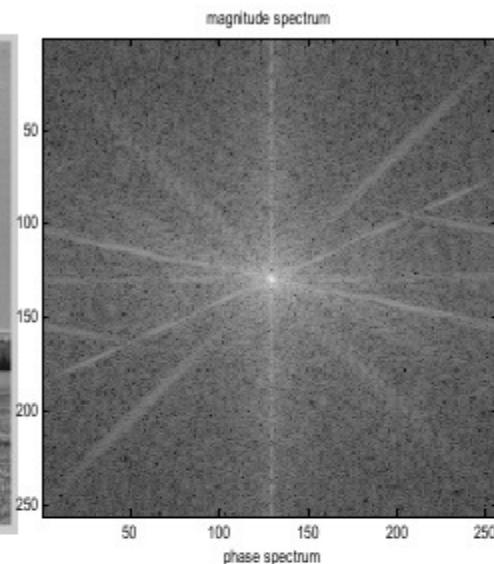
```
close all; clear all; clc
```

```
img = imread('cameraman.tif','tif')
imshow(img)
img = fftshift(img(:,:,__));
F = fft2(img);
```

```
figure;
```

```
imagesc(100*log(1+abs(fftshift(F))));
colormap(gray);
title('magnitude spectrum');
```

```
figure;
imagesc(angle(F)); colormap(gray);
title('phase spectrum');
```



# Visualization of Fourier Image

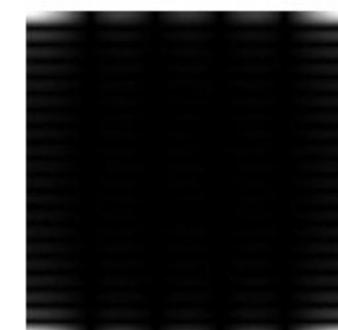
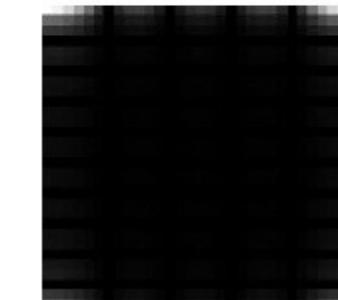
```
%Create a black 30x30 image  
f=zeros(30,30);
```

```
f(5:24,13:17)=1;  
imshow(f);
```



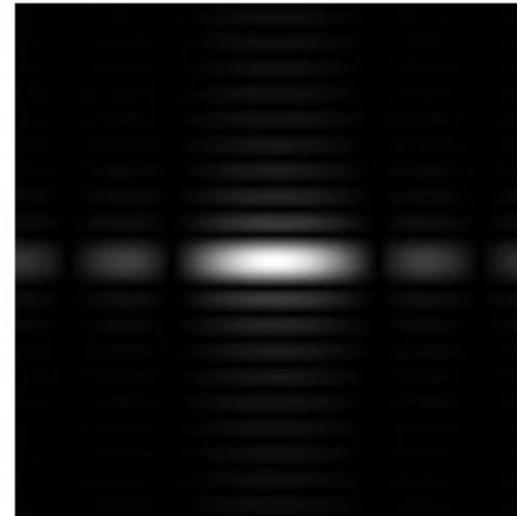
```
%Calculate the DFT.  
F=fft2(f);  
F2=abs(F);  
figure, imshow(F2,[])
```

```
F=fft2(f, 256, 256);  
F2=abs(F);  
figure, imshow(F2, [])
```

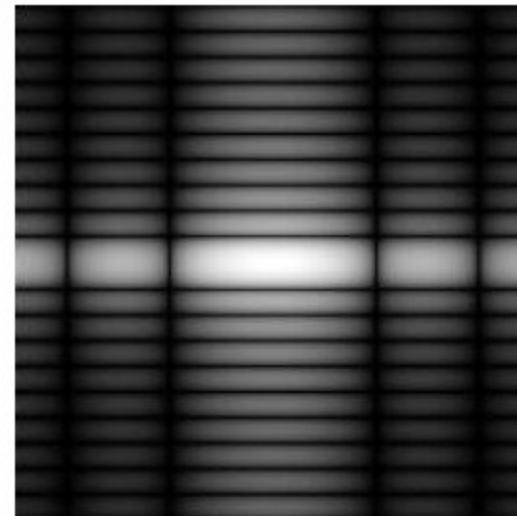


# Visualization of Fourier Image

```
F2=fftshift(F);  
F2=abs(F2);  
figure,imshow(F2,[ ])
```

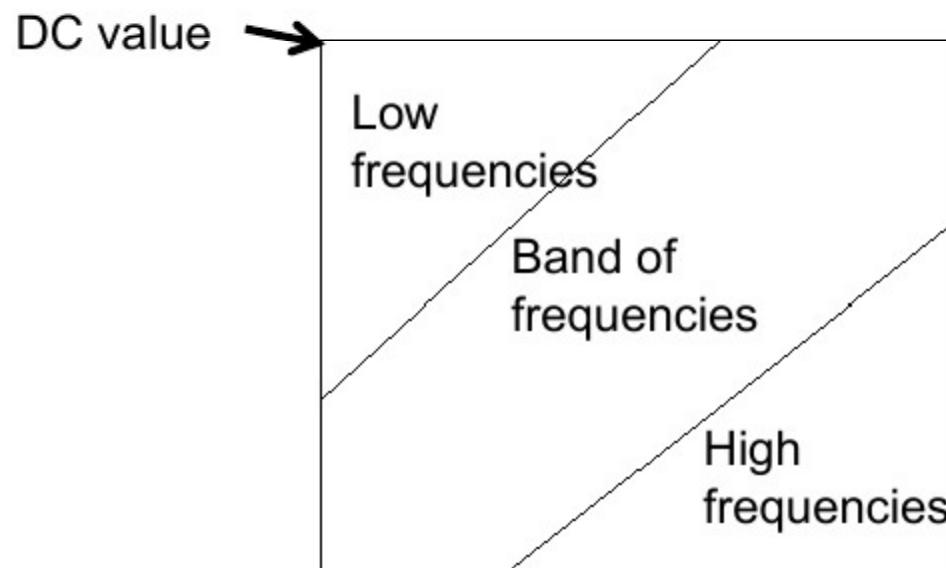


```
F2=log(1+F2);  
figure,imshow(F2,[])
```



## The Discrete Cosine Transform (DCT)

The discrete cosine transform (DCT) helps separate the image into parts or spectral sub-band.



## Implementation

- The formula for DCT applied on a image:

$$r(x, y, u, v) = s(x, y, u, v) \\ = \alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2M}\right] \quad (8.2-18)$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, \dots, n-1 \end{cases} \quad (8.2-19)$$

$$F(u, v) = \frac{\alpha(u)\alpha(v)}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2M}\right]$$

## DCT: Implementation

- The input image is N by M;
- $f(x, y)$  is the intensity of the pixel in row  $x$  and column  $y$ ;
- $F(u,v)$  is the DCT coefficient in row  $u$  and column  $v$  of the DCT matrix.
- For an  $N \times M$  image there are  $N \times M$  coefficients
  - Each image sample contributes to each coefficient
  - Each  $(u,v)$  pair corresponds to a ‘pattern’ or ‘basis function’

# DCT basis functions (patterns)

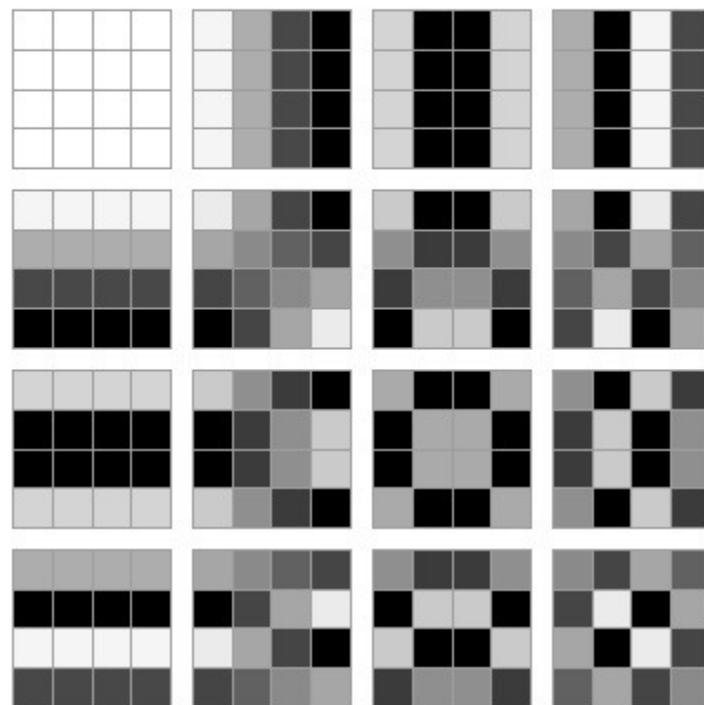


Figure 8.23 Basic Patterns or  
Image for N = 4

*Normalized Values ---*

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2M}\right]$$

## DCT Properties

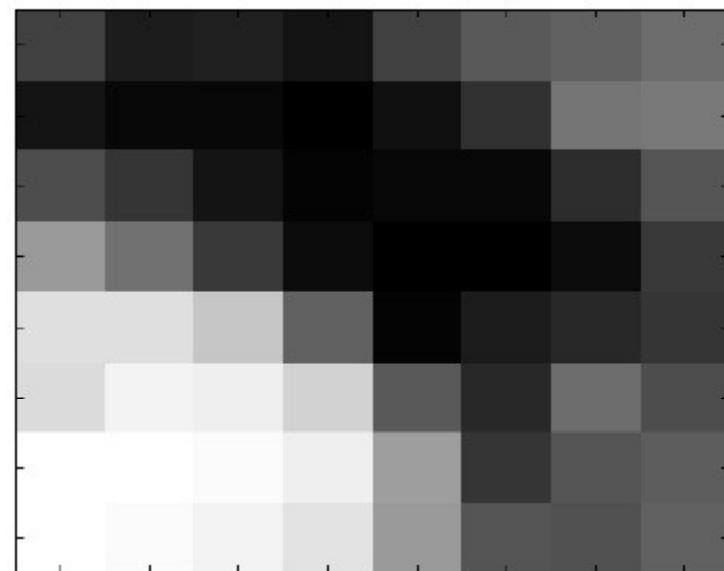
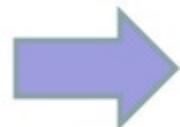
- For most images, much of the signal energy lies at **low frequencies**; these appear in the upper left corner of the DCT.
- **Compression is achieved** since the lower right values represent higher frequencies, and are often small - small enough to be neglected with little visible distortion.

- **Decorelation** – the principal advantage of transformed images is the low redundancy between neighbours pixels. From this fact results uncorrelated coefficients which can be coded independently
- **Energy compactness** – the capacity of the transformation to pack the input datas in as few coefficients as possible
- **Separability** – the 2D DCT can be calculated in two steps by applying the 1D formula successively on the lines and the columns of an image

## Example: DCT

### Image Patch –

63	33	36	28	63	81	86	98
27	18	17	11	22	48	104	108
72	52	28	15	17	16	47	77
132	100	56	19	10	9	21	55
187	186	166	88	13	34	43	51
184	203	199	177	82	44	97	73
211	214	208	198	134	52	78	83
211	210	203	191	133	79	74	86



Numerical Values of Pixels

Pixels gray shades as image

\* Value Range 0 (black) --- 255 (white)

## Example: DCT

### Image Patch –

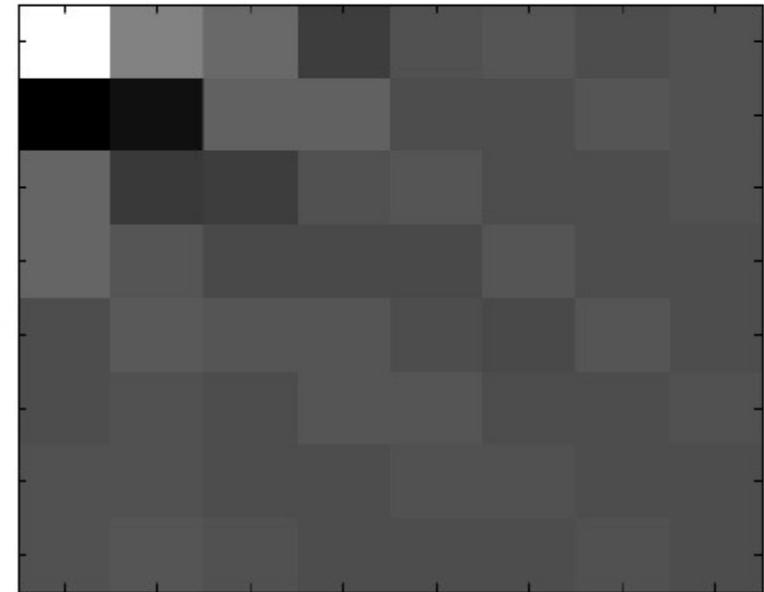
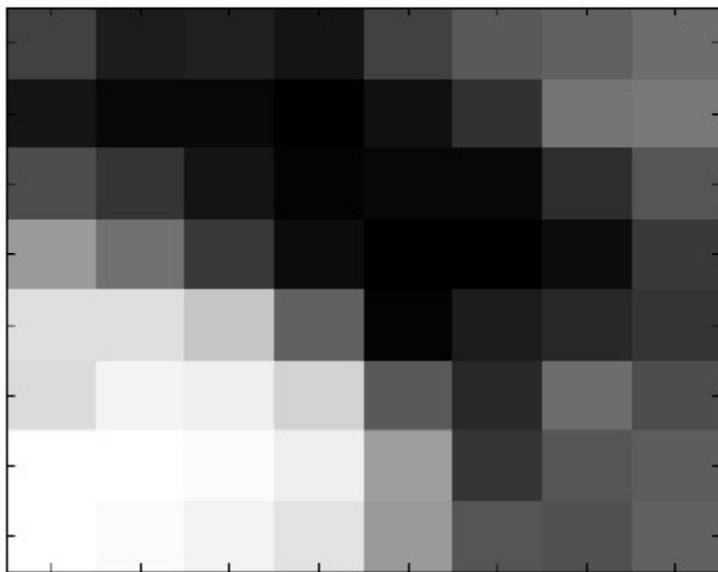
63	33	36	28	63	81	86	98
27	18	17	11	22	48	104	108
72	52	28	15	17	16	47	77
132	100	56	19	10	9	21	55
187	186	166	88	13	34	43	51
184	203	199	177	82	44	97	73
211	214	208	198	134	52	78	83
211	210	203	191	133	79	74	86



304	210	104	-69	10	20	-12	7
-327	-260	67	70	-10	-15	21	8
93	-84	-66	16	24	-2	-5	9
89	33	-19	-20	-26	21	-3	0
-9	42	18	27	-7	-17	29	-7
-5	15	-10	17	32	-15	-4	7
10	3	-12	-1	2	3	-2	-3
12	30	0	-3	-3	-6	12	-1

## Example: DCT

- 2D-DCT matrix as Image



Pixels gray shades as image

## Example: DCT

- Cut the least significant components (modified)

304	210	104	-69	10	20	-12	7	-304	210	104	-69	10	20	-12	0
-327	-260	67	70	-10	-15	21	8	-327	-260	67	70	-10	-15	0	0
93	-84	-66	16	24	-2	-5	9	93	-84	-66	16	24	0	0	0
89	33	-19	-20	-26	21	-3	0	89	33	-19	-20	0	0	0	0
-9	42	18	27	-7	-17	29	-7	-9	42	18	0	0	0	0	0
-5	15	-10	17	32	-15	-4	7	-5	15	0	0	0	0	0	0
10	3	-12	-1	2	3	-2	-3	10	0	0	0	0	0	0	0
12	30	0	-3	-3	-6	12	-1	0	0	0	0	0	0	0	0

As you can see, we save a little over half the original memory.

## Example: DCT

- Cut the least significant components

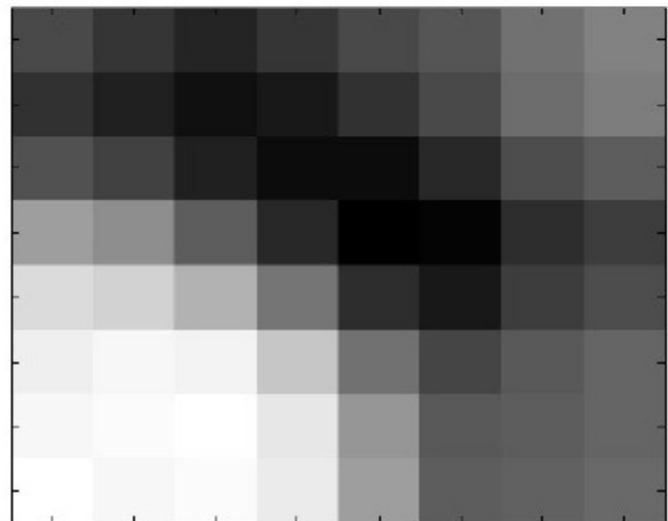
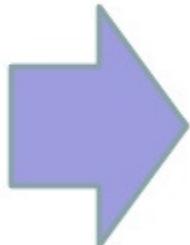
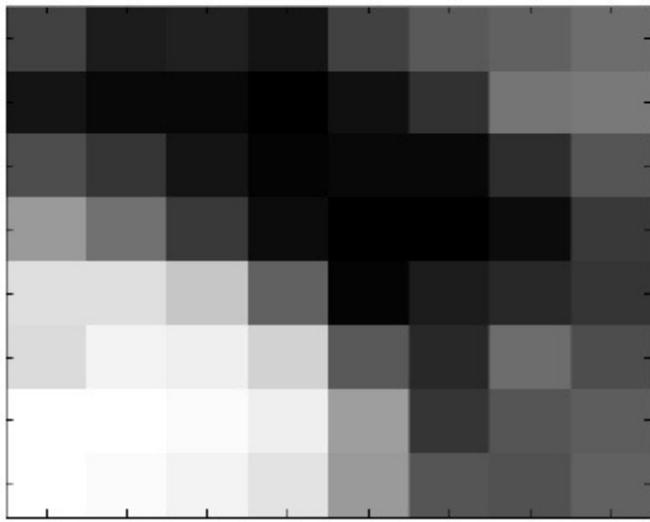
```
-304 210 104 -69 10 20 -12 0  
-327 -260 67 70 -10 -15 0 0  
 93 -84 -66 16 24 0 0 0  
 89 33 -19 -20 0 0 0 0  
 -9 42 18 0 0 0 0 0  
 -5 15 0 0 0 0 0 0  
 10 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0
```



As you can see, we save a little over half the original memory.

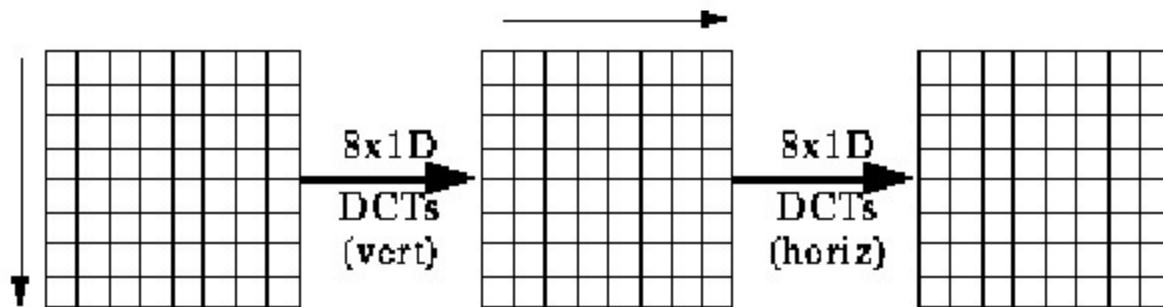
## Example: DCT

### Reconstruction



# Implementation

1. 2D DCT Implementation
2. 1D DCT implementation - Factoring reduces problem to a series of 1D DCTs :



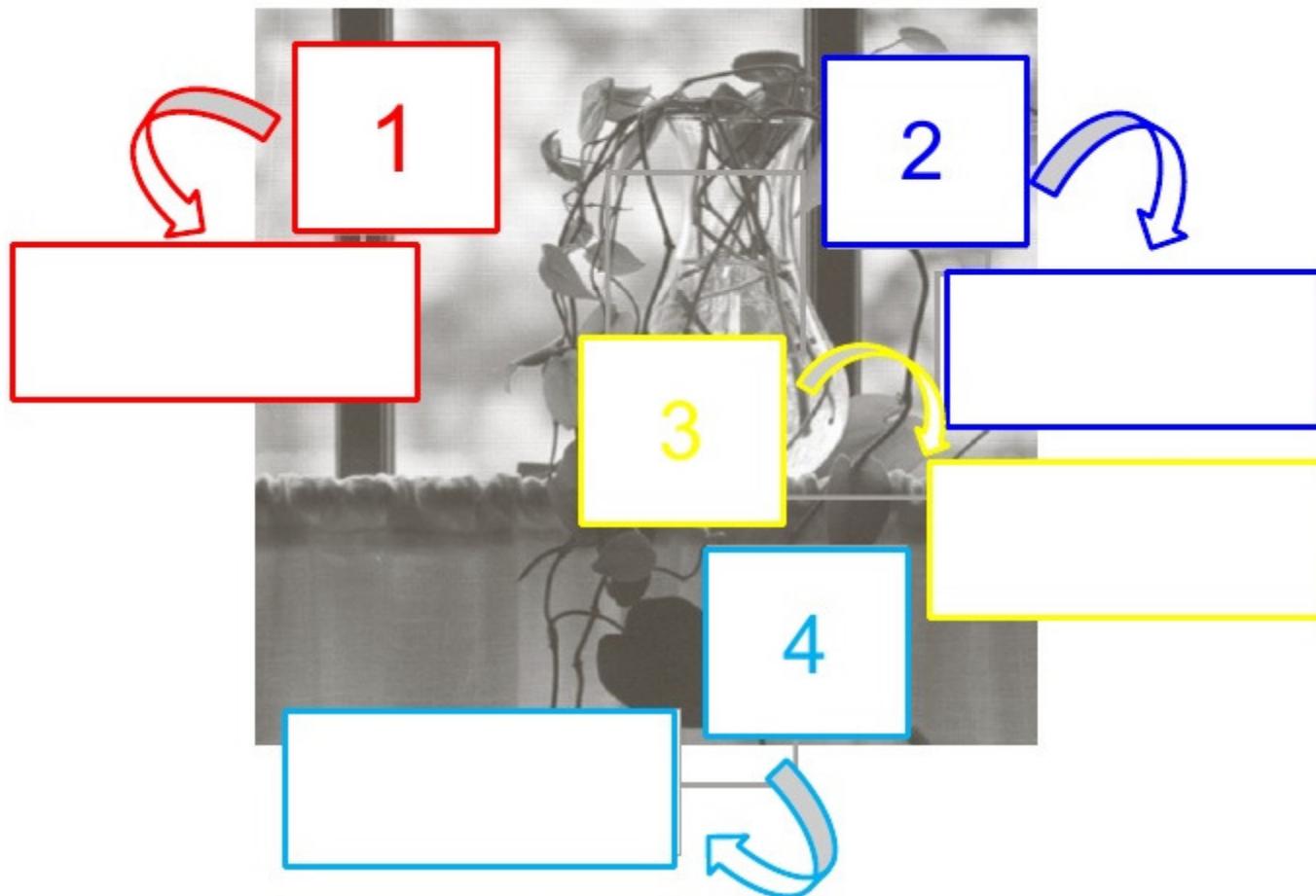
## Relationship between DCT and FFT

DCT is actually a cut-down version of the FFT:

- Only the real part of FFT
- Computationally simpler than FFT
- DCT -- Effective for Multimedia Compression

# Wavelets and Multi-resolution: Introduction

Local histograms can vary from one part of an image to another making statistical modeling over the span of an entire image is a difficult, or impossible task.



**FIGURE 7.1**

An image and its local histogram variations.

# Wavelets and Multi-resolution: Introduction

The Fourier transform is an useful tool to analyze the frequency components of the signal. However, if we take the Fourier transform over the whole Image, we cannot tell at where a particular feature rises.

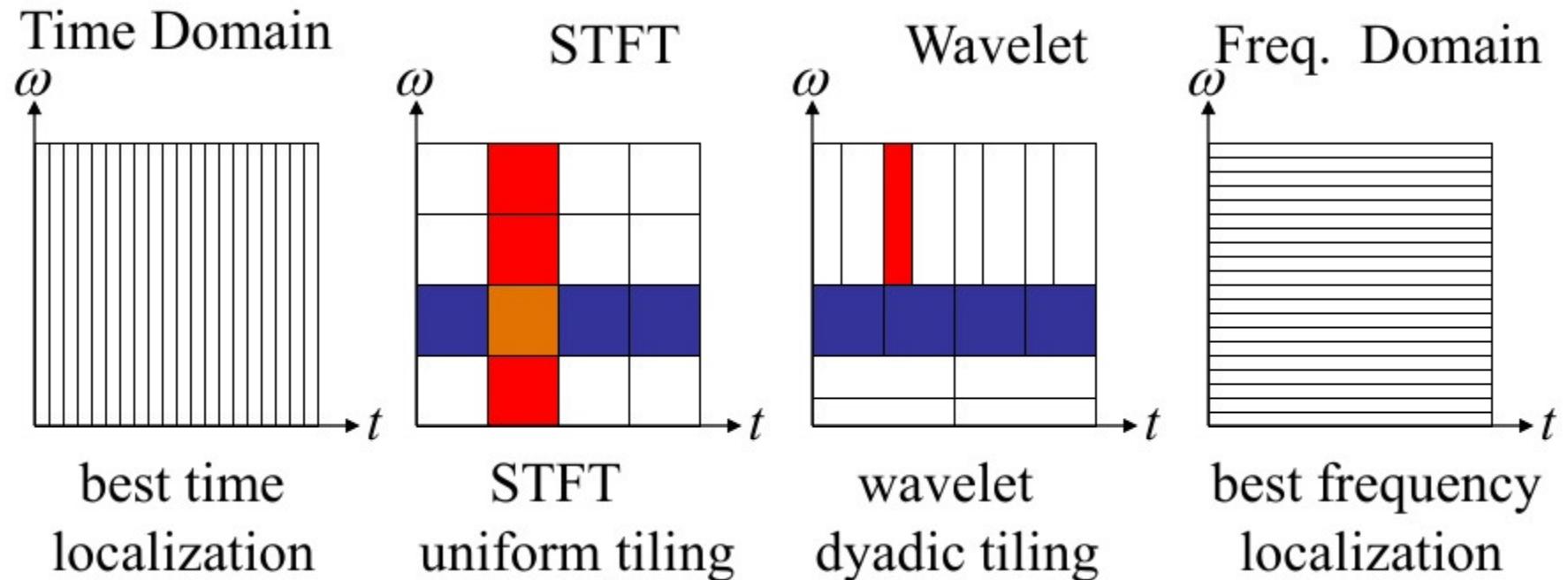
- Short-time Fourier transform (STFT) uses a sliding window to find spectrogram, which gives the information of both spatial and frequency. But still another problem exists: The length of window limits the resolution in frequency.

# Wavelets and Multi-resolution: Introduction

Wavelet transform seems to be a solution to the problem. It represent image in Multiresolutions.

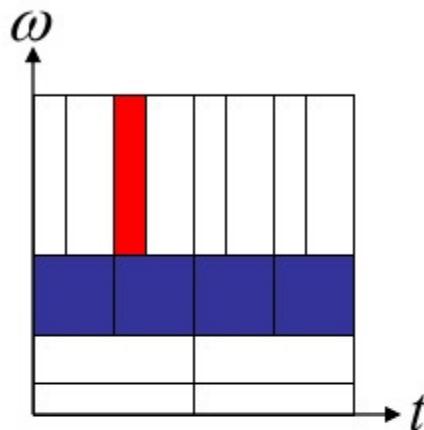
This allows them to provide the equivalent of a musical score for an image, revealing not only what notes (or frequencies) to play but also when to play them.

# Time-Frequency Localization



- ◆ Heisenberg's Uncertainty Principle: bound on T-F product
- ◆ Wavelets provide flexibility and good time-frequency trade-off

# Wavelet Transform (WT)



- Fourier Transform (FT)
- Window Fourier Transform (WFT)
- Heisenberg uncertainty principle.
  - $S_o W_o \geq 1/4\pi$
- low frequencies the "wide" window is more appropriate because the signal changes slowly, and for high frequencies a "thin" is more adequate.
- This flexibility can be achieved in wavelet domain by Multi Resolution Analysis

# History Wavelet Transform (WT)

- The first literature that relates to the wavelet transform is Haar wavelet. It was proposed by the mathematician Alfrd Haar in 1909.
- Then the concept was proposed by the geophysicist Jean Morlet in 1981. Afterward, Morlet and the physicist Alex Grossman invented the term wavelet in 1984.

# History Wavelet Transform (WT)

Before 1985, Haar wavelet was the only orthogonal wavelet people know. A lot of researchers even thought that there was no orthogonal wavelet except Haar wavelet.

Fortunately, the mathematician Yves Meyer constructed the second orthogonal wavelet called Meyer wavelet in 1985.

As more and more scholars joined in this field, the 1st international conference was held in France in 1987.

# History Wavelet Transform (WT)

- In 1988, Stephane Mallat and Meyer proposed the concept of multiresolution.
  - “[A Wavelet Tour of Signal Processing](#)”
- In the same year, Ingrid Daubechies found a systematical method to construct the compact support orthogonal wavelet.
- In 1989, Mallat proposed the fast wavelet transform. With the appearance of this fast algorithm, the wavelet transform had numerous applications in the signal processing field.

# Why wavelets?

- The Wavelet transform performs a correlation analysis, therefore the output is expected to be maximal when the input signal most resembles the mother wavelet.

**Fourier** ----- **the basis function are sines and cosines.**

**Wavelet** ----- **set of wavelet functions.**

- Decompose a signal into component parts
  - Fourier analysis: a signal can be represented as a (possibly infinite) sum of sine and cosine functions.
  - Signal becomes a set of wavelet coefficients. Coefficients represent features of signal.
- Signal can be completely reconstructed from all the coefficients.
- Signal can be partially reconstructed from some coefficients.

# Wavelets and Multi-resolution: Introduction

- When we look at images, generally we see connected regions of similar texture and intensity levels that combine to form objects.
- If the objects are small in size or low in contrast, we normally examine them at high resolutions;  
If they are large in size or high in contrast, a low resolution or coarse view is required.
- If both small and large objects—or low- and high-contrast objects—are present simultaneously, it can be advantageous to study them at several resolutions.

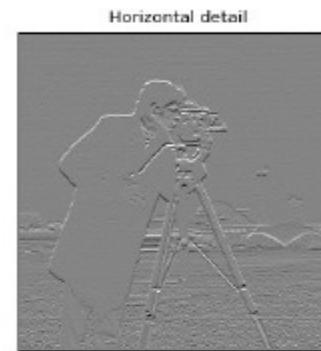
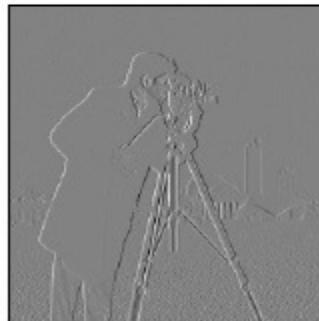
# Wavelets and Multi-resolution: Introduction

First level Decomposition



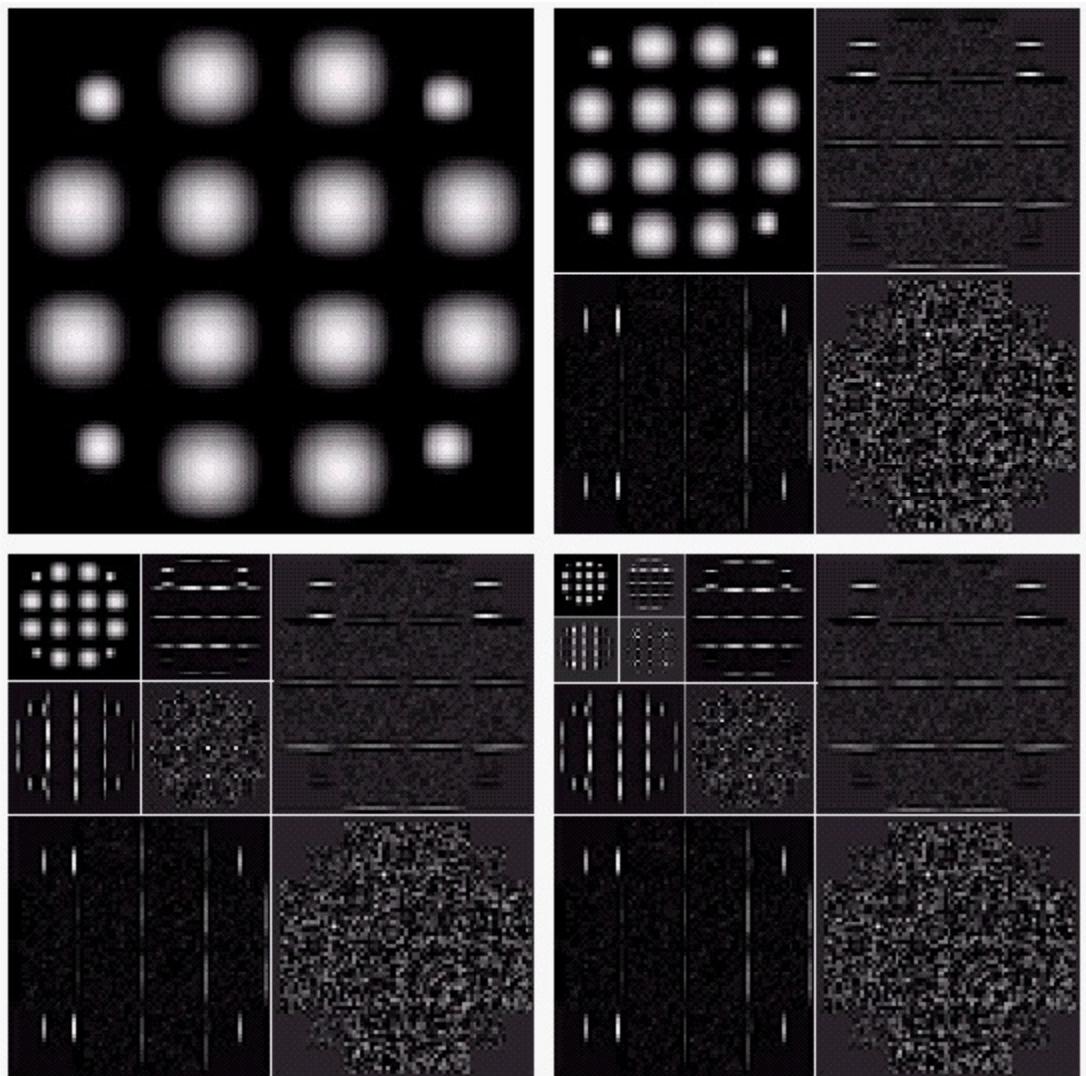
# Wavelets and Multi-resolution: Introduction

## First level Decomposition



# Wavelets and Multi-resolution: Introduction

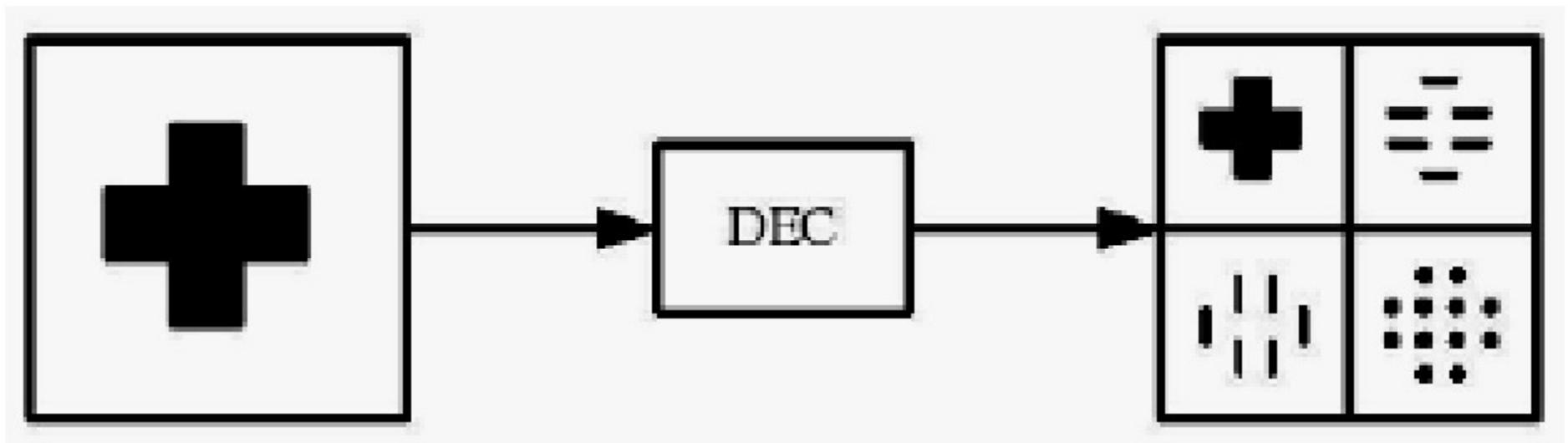
## Decomposition



**FIGURE 7.23** A  
three-scale FWT.

# Wavelets and Multi-resolution: Introduction

## Decomposition



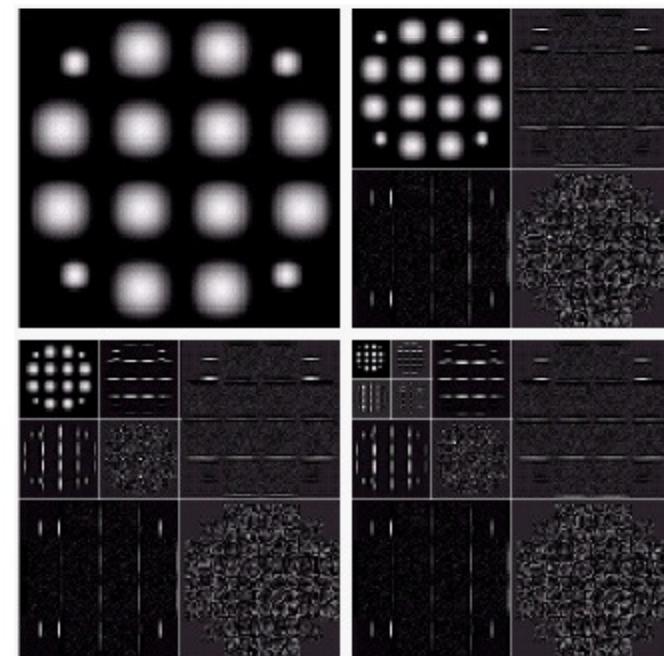
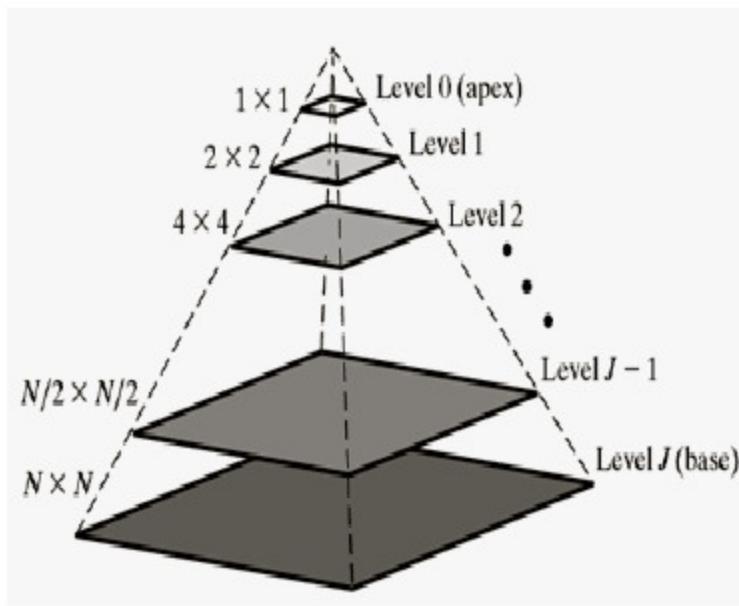
# Wavelets

- The wavelet transform (WT) has gained widespread acceptance in signal processing and image compression. Because of their **inherent multi-resolution nature**.
  - The JPEG committee has recommend its image coding standard, JPEG-2000, which has been based upon DWT.

# Image Pyramid

*A powerful, simple structure for representing images at more than one resolution.*

*An image pyramid is a collection of decreasing resolution images arranged in the shape of a pyramid.*

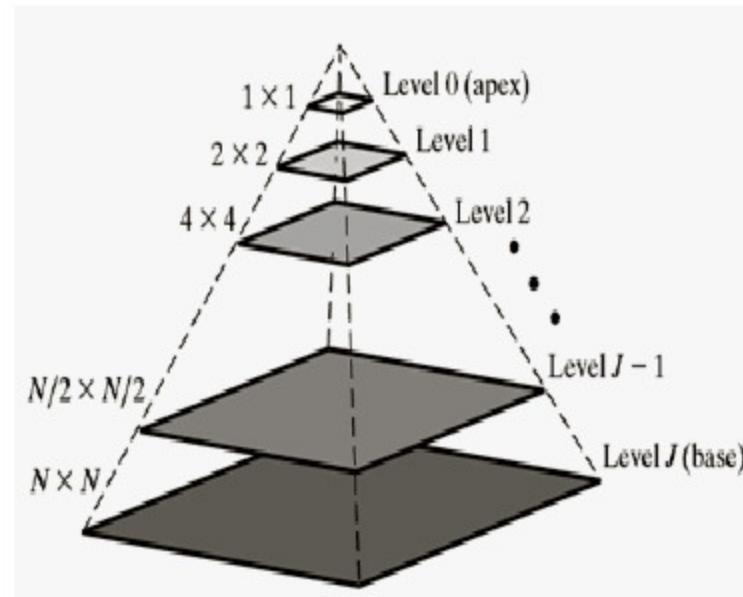


# Image Pyramid

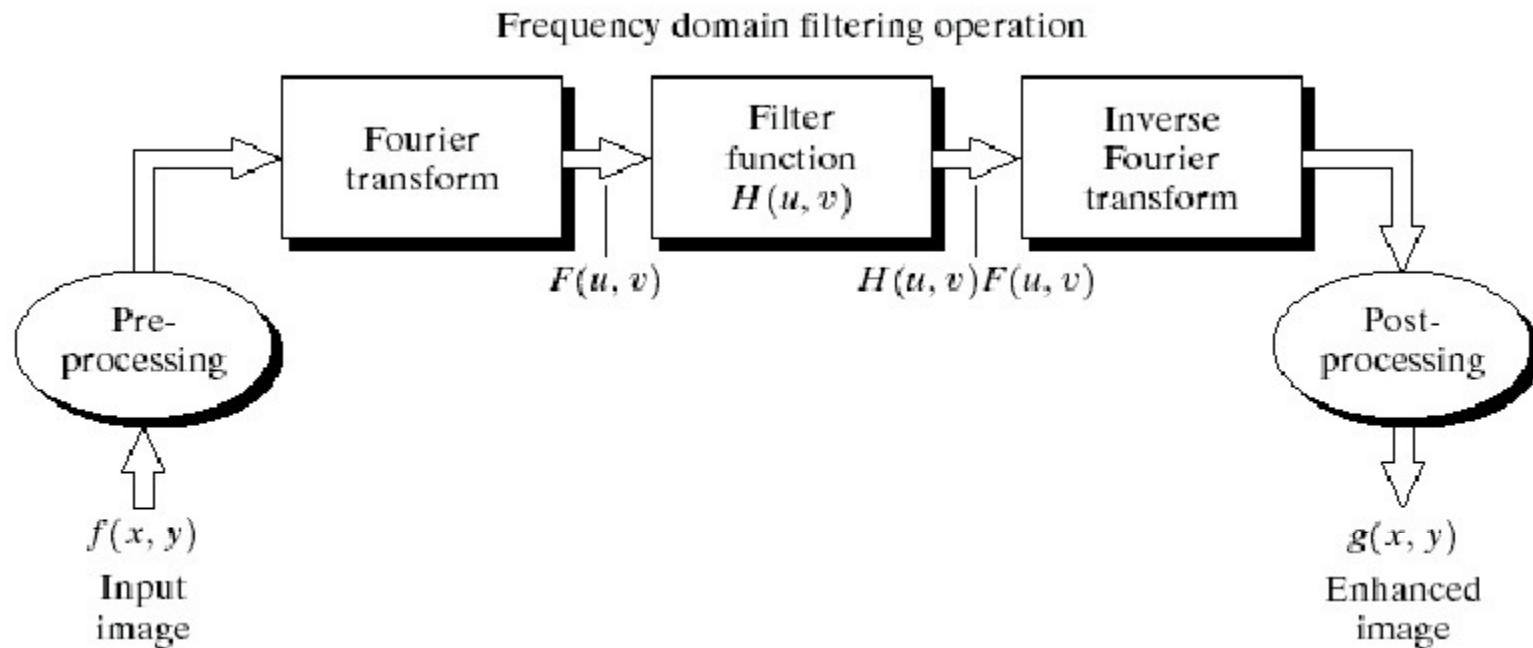
The base of the pyramid contains a high-resolution representation of the image being processed; the apex contains a low-resolution approximation. As you move up the pyramid, both size and resolution decrease.

a  
b

**FIGURE 7.2**  
(a) An image pyramid. (b) A simple system for creating approximation and prediction residual pyramids.



# Frequency Domain Filtering



To filter an image in the frequency domain:

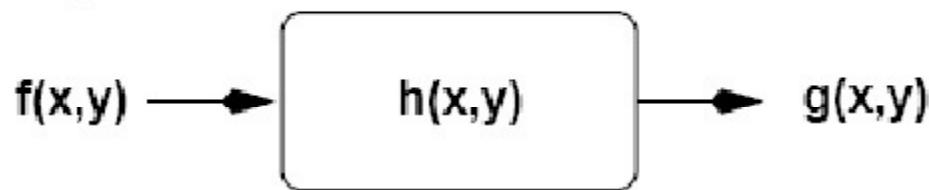
1. Transformation of the input image and the system response from the spatial domain to the frequency domain.
2. Processing the image in the frequency domain.
3. Transformation of the processed image back to the spatial domain.

# Properties of 2-D DFT

Convolution Property:

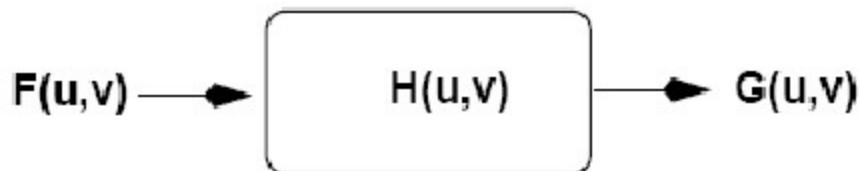
$$f(x, y) * h(x, y) \Leftrightarrow F(u, v)H(u, v)$$
$$f(x, y)h(x, y) \Leftrightarrow F(u, v) * H(u, v)$$

Spatial Domain



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

Frequency Domain



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

# Frequency Domain Analysis

Filters are classified by their properties in the frequency domain:

- (1) Low-pass
- (2) High-pass
- (3) Band-pass
- (4) Band-stop

# Low-pass Filters

*Low pass filters* – only pass the low frequencies, drop the high ones

The basic model for filtering is:

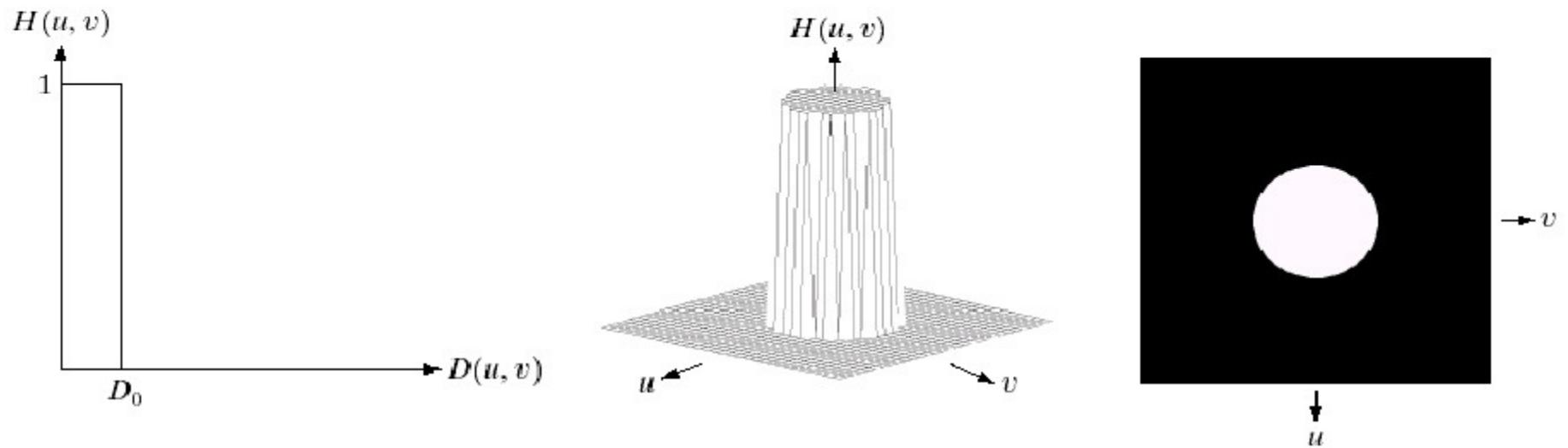
$$G(u,v) = H(u,v)F(u,v)$$

where  $F(u,v)$  is the Fourier transform of the image being filtered and  $H(u, v)$  is the filter transform function.

Smoothing is achieved in the frequency domain by dropping out the high frequency components

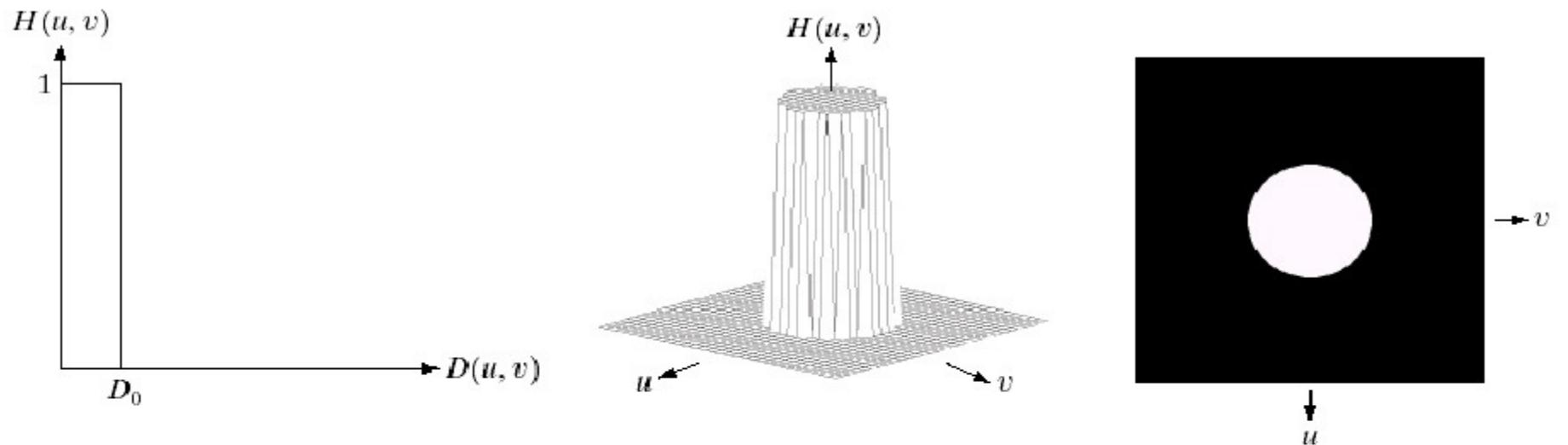
# Ideal Low-pass Filters

Simply cut off all high frequency components and image power to be removed that are a specified distance  $D_0$  from the origin of the transform



Changing the distance changes the behaviour of the filter

# Ideal Low-pass Filters



The transfer function for the ideal low pass filter can be given as:

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$$

where  $D(u, v)$  is given as:

$$D(u, v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$$

# Example

Design a ILPF of size [4x4] with centre at (2,2) and cutoff radius of 1.9.

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

## Example

Design a ILPF of size [4x4] with centre at (2,2) and cutoff radius of 1.9.

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

A [4×4] Distance matrix with the center at coordinates (2, 2) is-

$$D(u,v) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \text{blue circle} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

## Example

Design a ILPF of size [4x4] with centre at (2,2) and cutoff radius of 1.9.

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

A [4×4] Distance matrix with the center at coordinates (2, 2) is-

$$D(u,v) = \begin{bmatrix} 2.8284 & 2.2361 & 2.0000 & 2.2361 \\ 2.2361 & 1.4142 & 1.0000 & 1.4142 \\ 2.0000 & 1.0000 & 0 & 1.0000 \\ 2.2361 & 1.4142 & 1.0000 & 1.4142 \end{bmatrix}$$

$$H(k,l) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

## Example

Design a ILPF of size [4x4] with centre at (2,2) and cutoff radius of 0.9.

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

A [4×4] Distance matrix with the center at coordinates (2, 2) is-

$$D(u,v) = \begin{bmatrix} 2.8284 & 2.2361 & 2.0000 & 2.2361 \\ 2.2361 & 1.4142 & 1.0000 & 1.4142 \\ 2.0000 & 1.0000 & 0 & 1.0000 \\ 2.2361 & 1.4142 & 1.0000 & 1.4142 \end{bmatrix}$$

$$H(k,l) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# Example

Find the 2-D DFT of the given average spatial filter:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$F(u, v) = \frac{1}{M \cdot N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$$

for  $u = 0, 1, 2, \dots, (M-1)$  and  $v = 0, 1, 2, \dots, (N-1)$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# Example

Find the 2-D DFT of the given average spatial filter:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# Example

Find the 2-D DFT of the given average spatial filter:

$$\begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

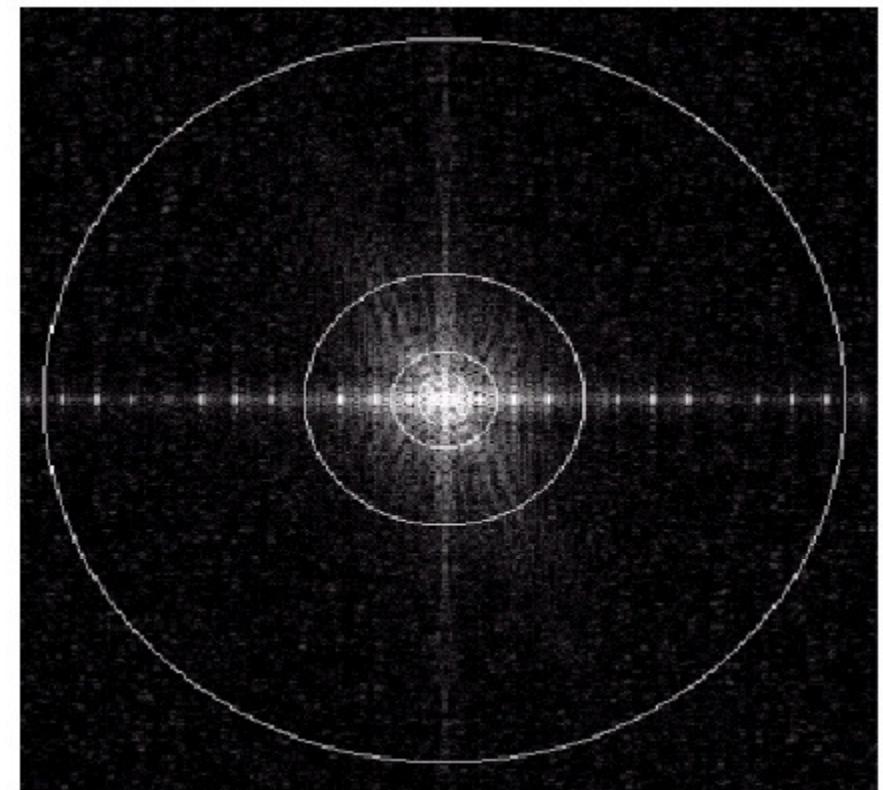
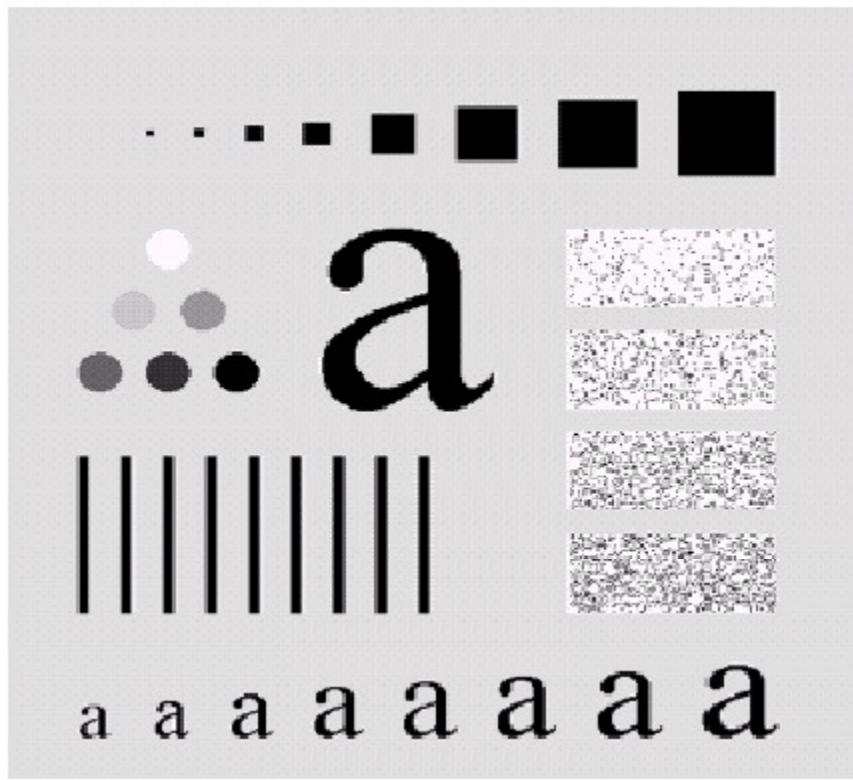
# Ideal Low Pass Filter

The other way to establish a set of standard cutoff frequency loci is to compute circles that enclose specified amounts of total image power

$$P_T = \sum_{u=0}^{P-1} \sum_{v=0}^{Q-1} P(u, v)$$

$$\alpha = 100 \left[ \sum_u \sum_v P(u, v) / P_T \right]$$

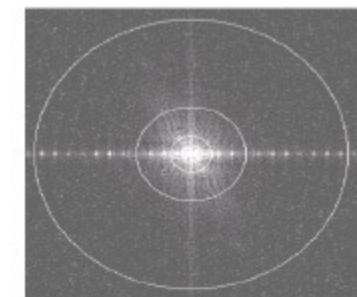
# Example: DFT of Test Image



a b

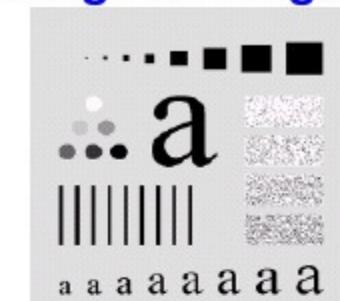
**FIGURE 4.41** (a) Test pattern of size  $688 \times 688$  pixels, and (b) its Fourier spectrum. The spectrum is double the image size due to padding but is shown in half size so that it fits in the page. The superimposed circles have radii equal to 10, 30, 60, 160, and 460 with respect to the full-size spectrum image. These radii enclose 87.0, 93.1, 95.7, 97.8, and 99.2% of the padded image power, respectively.

# Ideal Low Pass Filter

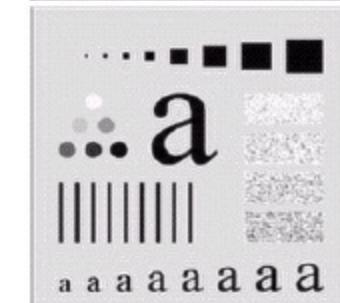
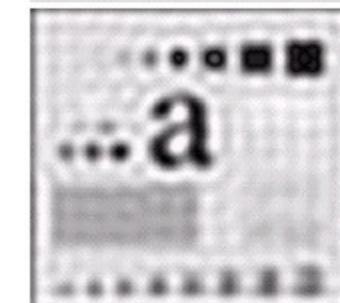


Result of filtering with ideal low pass filter of radius 15

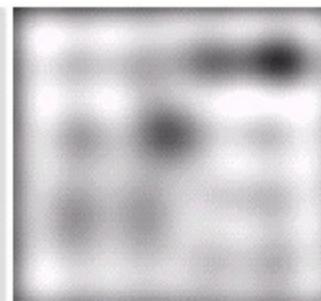
Original image



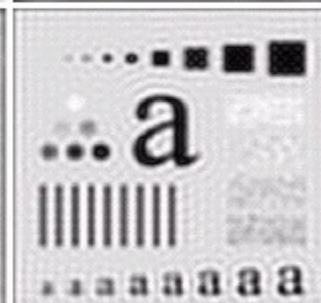
Result of filtering with ideal low pass filter of radius 80



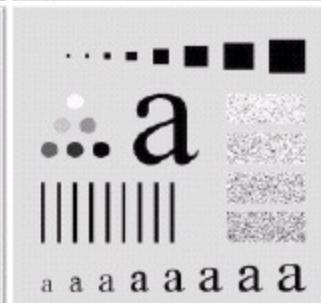
Result of filtering with ideal low pass filter of radius 5



Result of filtering with ideal low pass filter of radius 30



Result of filtering with ideal low pass filter of radius 230



**FIGURE 4.42** (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values : 5, 15, 30, 80, and 230, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

## Drawback: ILPF

*In the case of ideal LPF filter, the cutoff frequency is sharp. The sharp cutoff frequencies of an ILPF cannot be realized with electronic components, although they certainly can be simulated in a computer.*

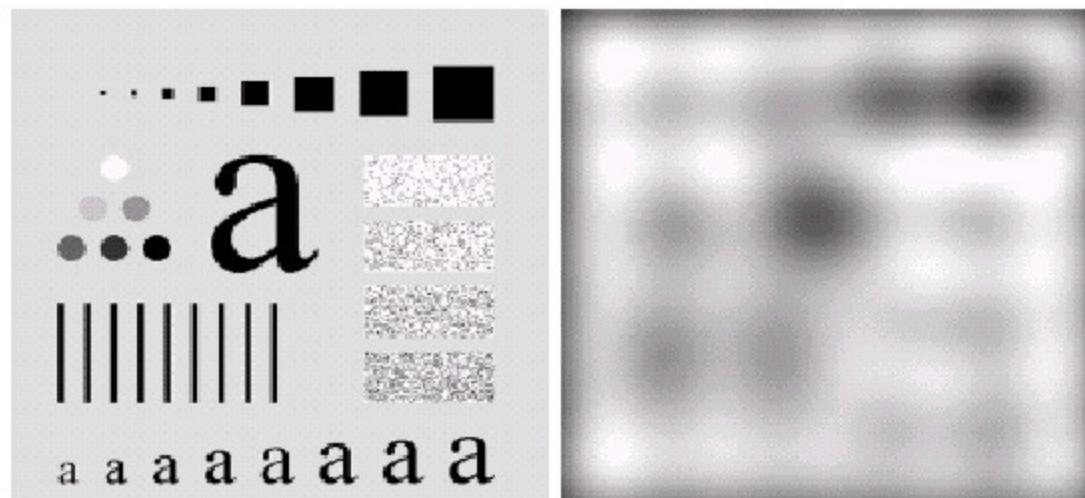
$$H(k,l) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

The effects of using these “nonphysical” filters on a digital image will generate ringing effect.

# Ideal Low Pass Filter

Radius 10 filtered image is useless for all practical purposes, unless the objective of blurring is to eliminate all detail in the image, except the “blobs” representing the largest objects.

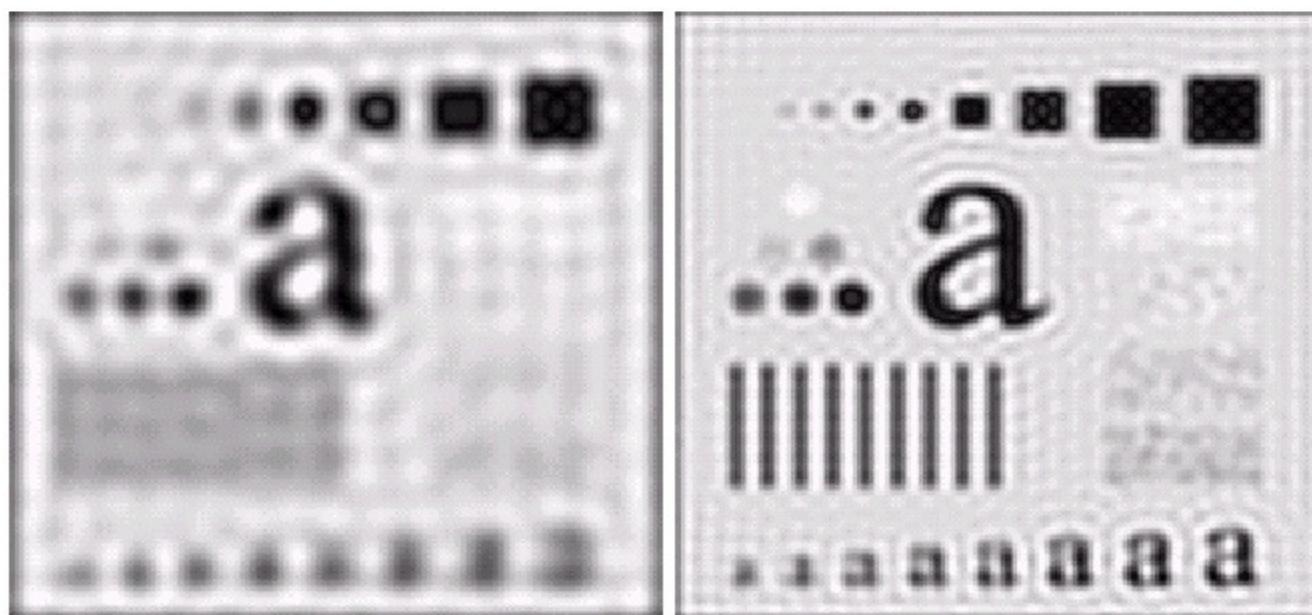
- The severe blurring in this image is a clear indication that most of the sharp detail information in the picture is contained in the 13% power removed by the filter.



# Ideal Low Pass Filter

As the filter radius increases, less and less power is removed, resulting in less blurring.

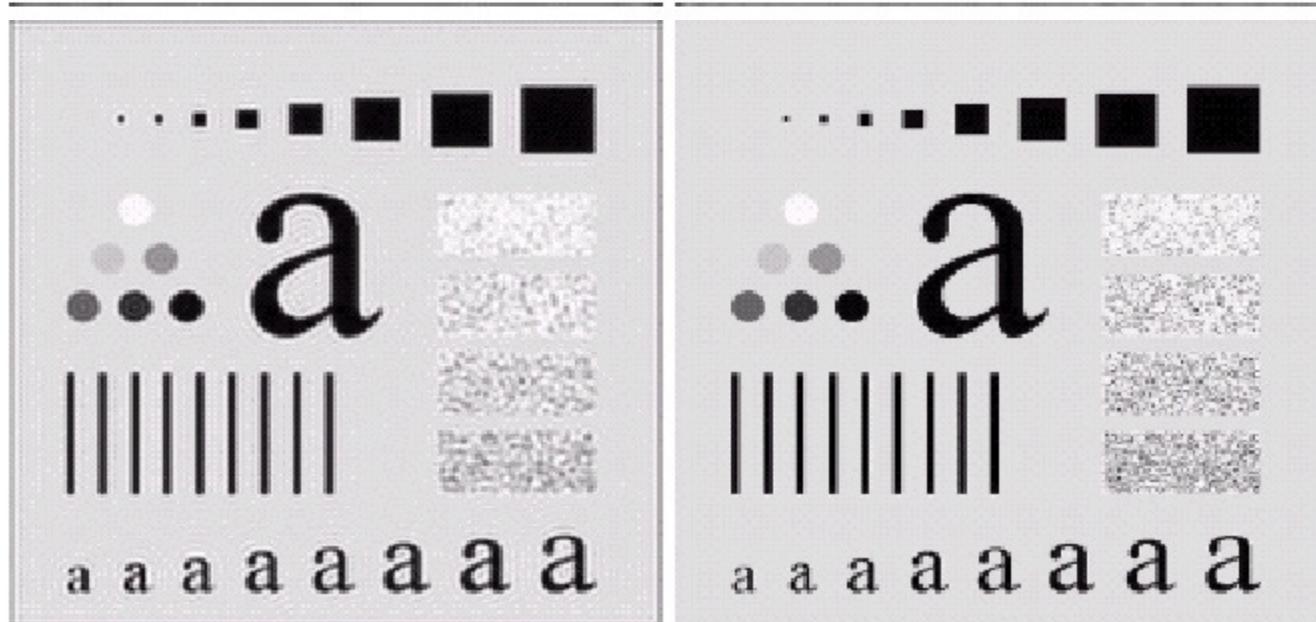
- Images are characterized by “ringing,” which becomes finer in texture as the amount of high frequency content removed decreases.



# Ideal Low Pass Filter

This ringing behavior is a characteristic of ideal filters, as you will see shortly.

The result for shows very slight blurring in the noisy squares but, for the most part, this image is quite close to the original. This indicates that little edge information is contained in the upper 0.8% of the spectrum power in this particular case.



# Ideal Low Pass Filter

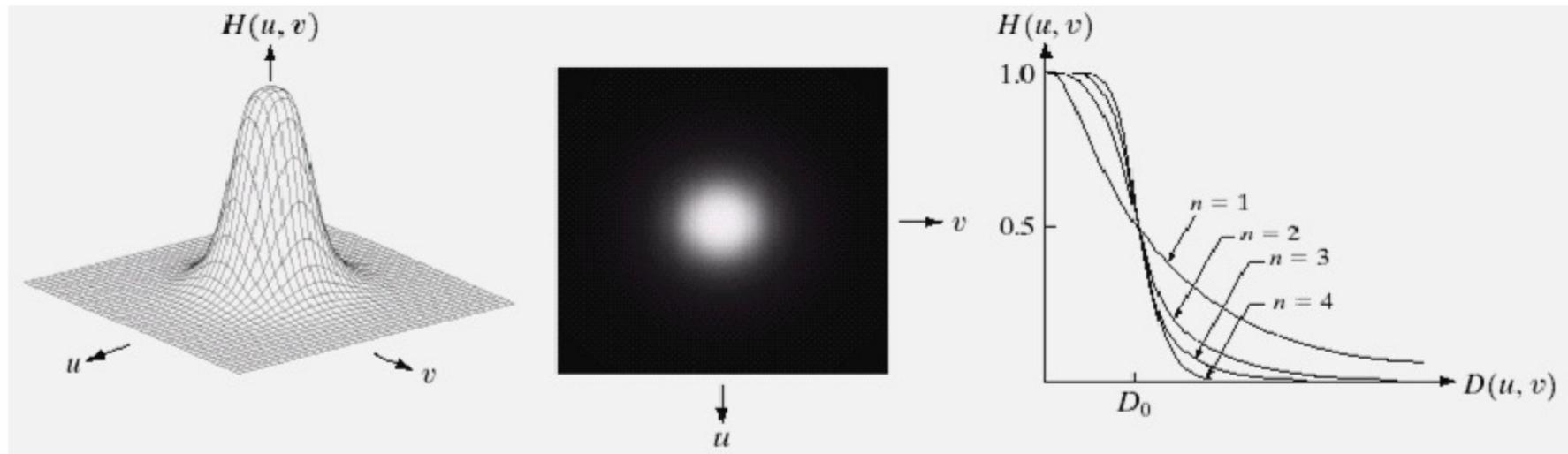
It is clear from this example that ideal low-pass filtering is not very practical.

However, it is useful to study their behavior as part of our development of filtering concepts. Also, as shown in the discussion that follows, some interesting insight is gained by attempting to explain the ringing property of ILPFs in the spatial domain.

# Butterworth Lowpass Filters

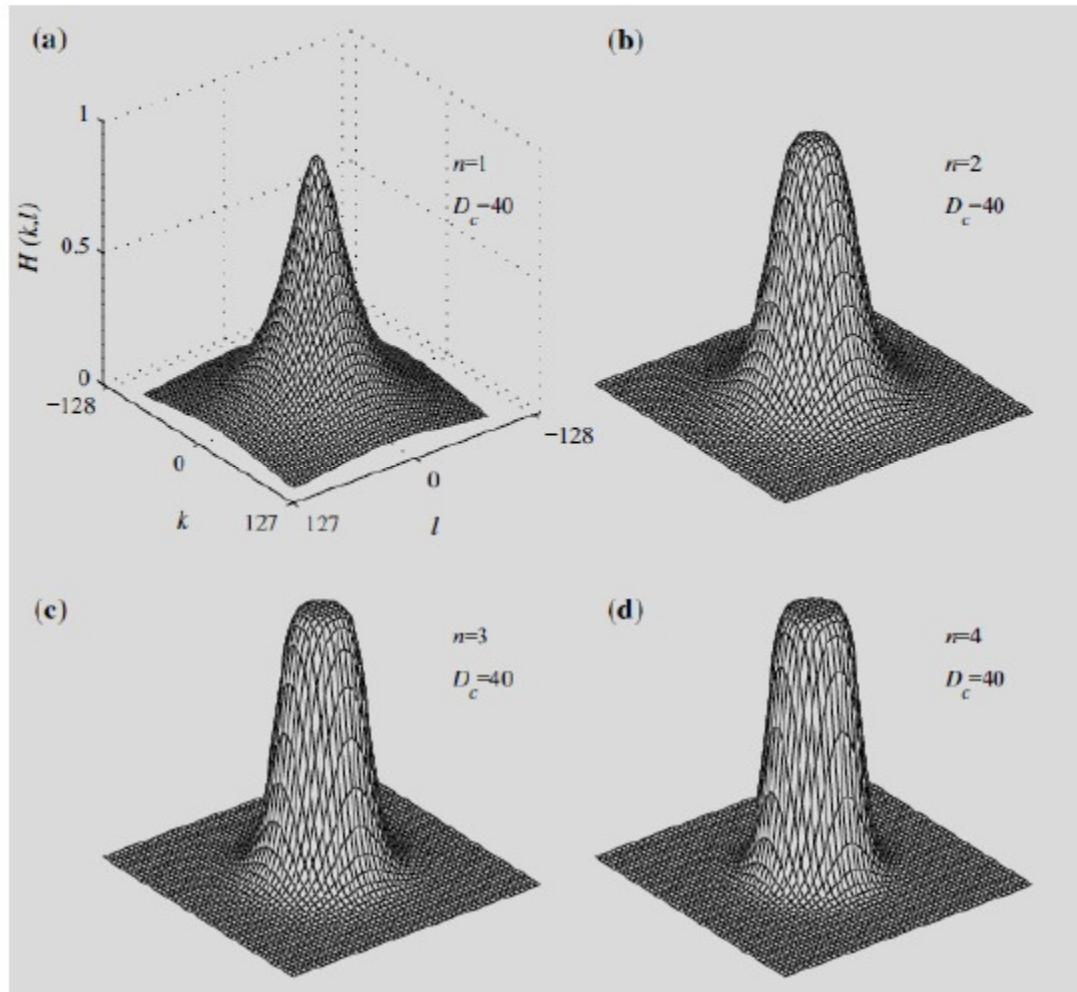
The transfer function of a Butterworth lowpass filter of order  $n$  with cutoff frequency at distance  $D_0$  from the origin is defined as:

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}}$$



Unlike the ILPF, the BLPF transfer function does not have a sharp discontinuity that gives a clear cutoff between passed and filtered frequencies.

# Butterworth Lowpass Filters

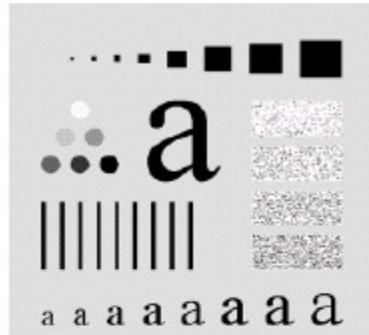


$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}}$$

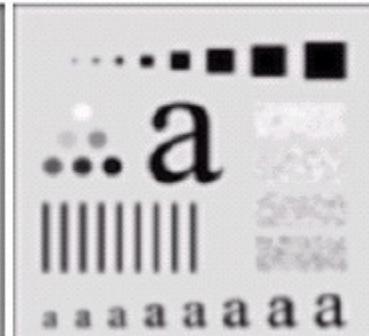
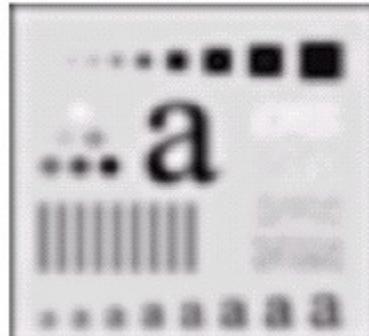
For  $n=1$  there is no ringing. The ringing will increase as  $n$  increases.

# Butterworth Lowpass Filters

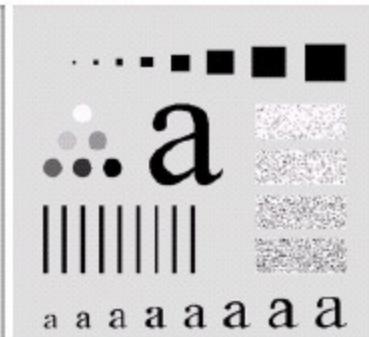
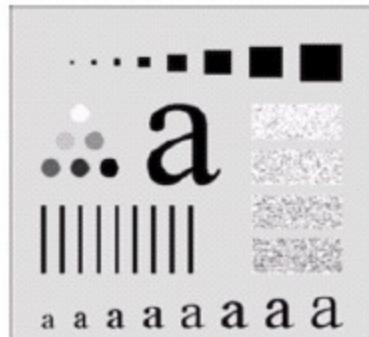
Original image



Result of filtering with Butterworth filter of order 2 and cutoff radius 15



Result of filtering with Butterworth filter of order 2 and cutoff radius 80

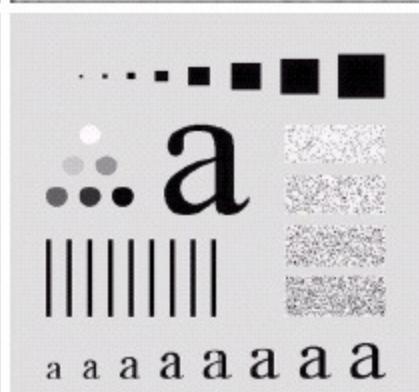
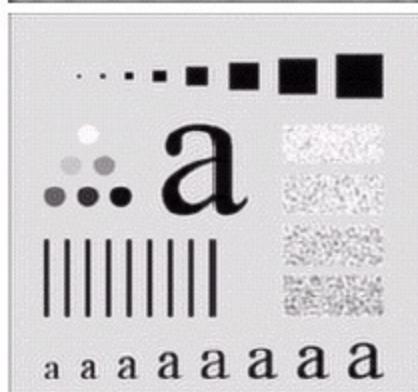
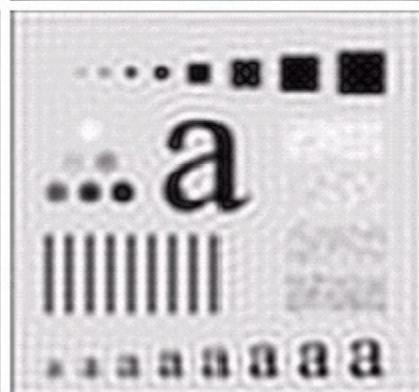
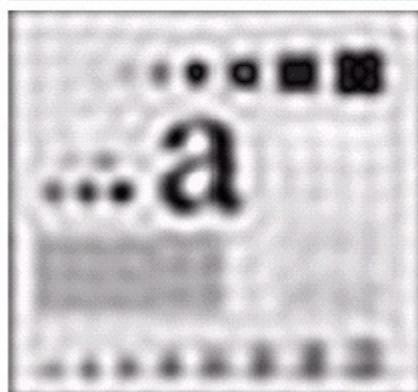
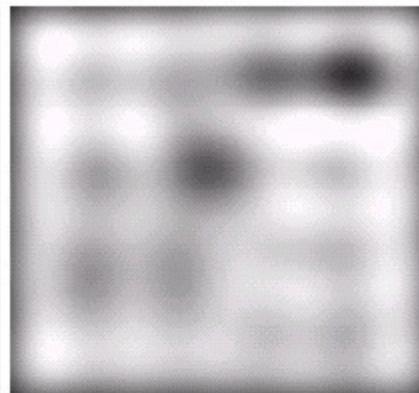
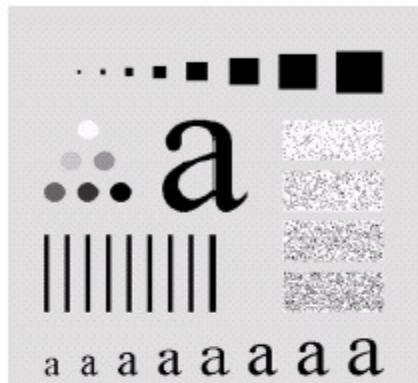


Result of filtering with Butterworth filter of order 2 and cutoff radius 5

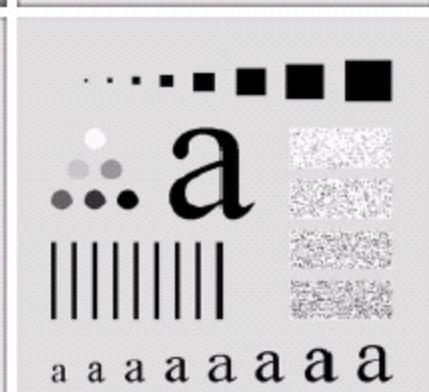
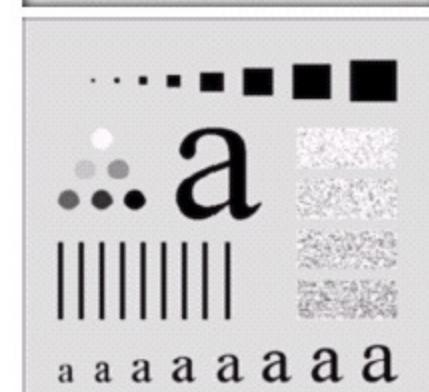
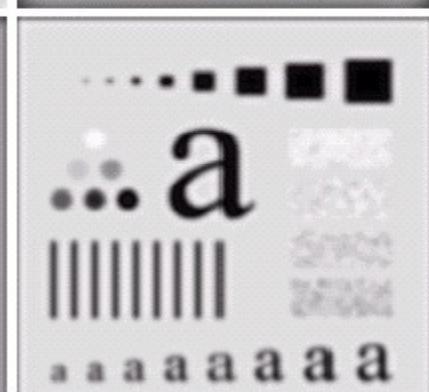
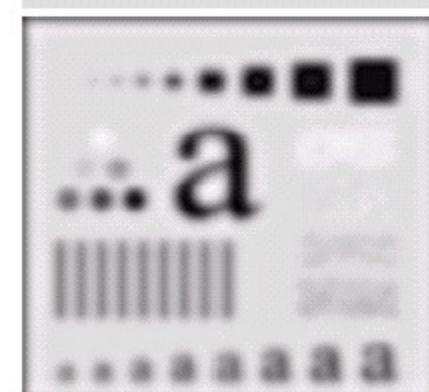
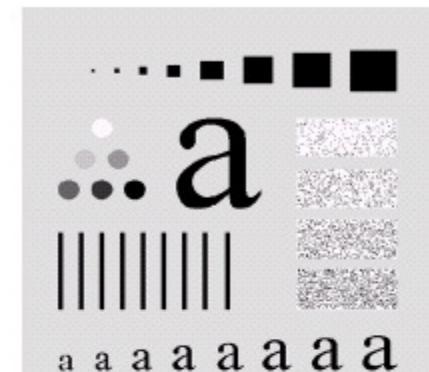
Result of filtering with Butterworth filter of order 2 and cutoff radius 30

Result of filtering with Butterworth filter of order 2 and cutoff radius 230

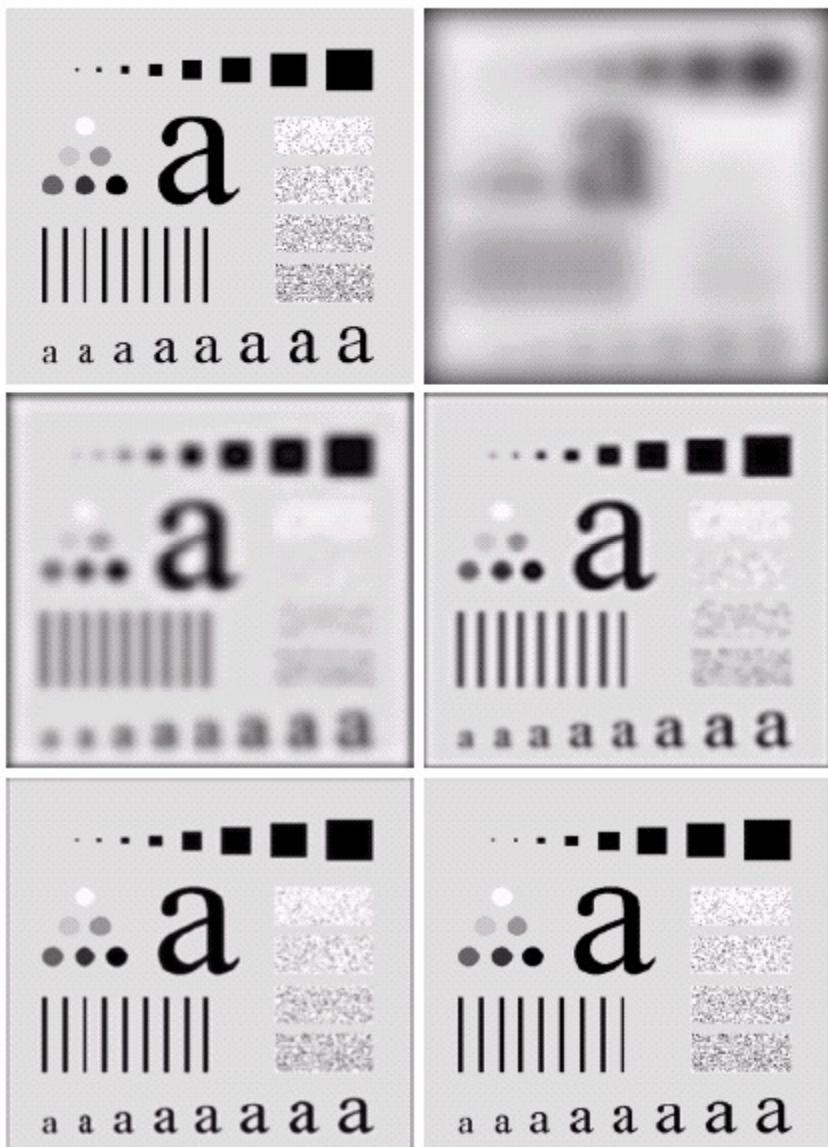
## Ideal LPF



## Butterworth LPF



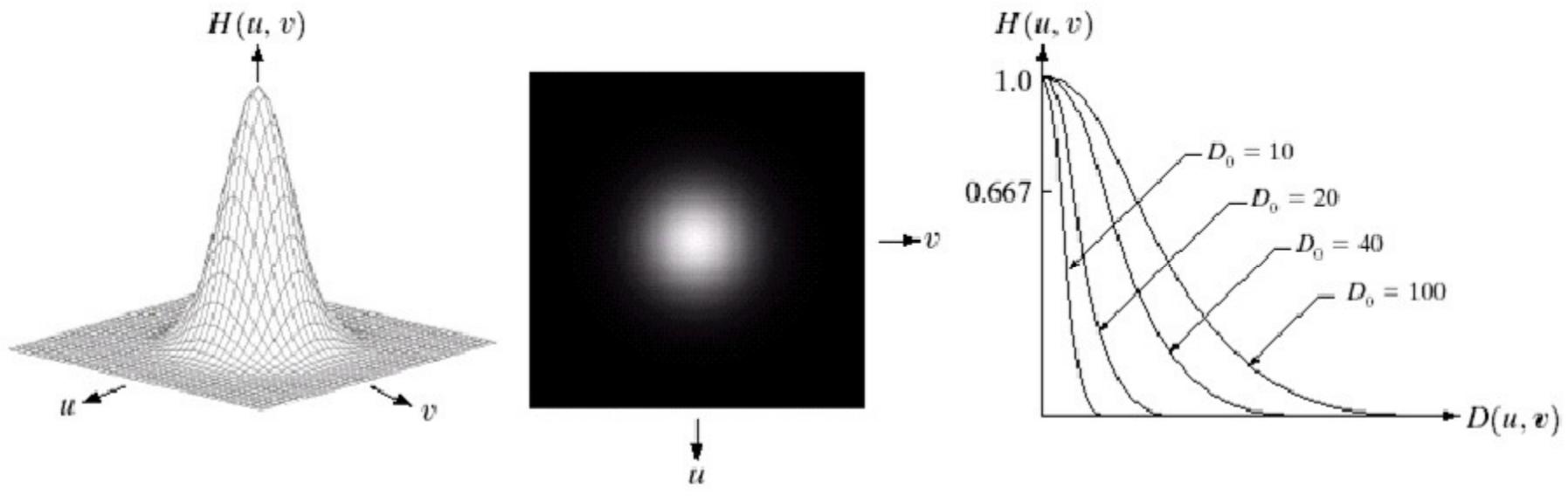
# Butterworth Lowpass Filters



Moreover, no ringing is visible in any of the images processed with this particular BLPF, a fact attributed to the filter's smooth transition between low and high frequencies.

# Gaussian Low-pass Filters

$$H(u, v) = e^{-D^2(u, v)/2D_0^2}$$



a b c

**FIGURE 4.17** (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of  $D_0$ .

The Gaussian filter is the only filter that has no ripple and hence no ringing effect.

# Gaussian Low-pass Filters

Gaussian Low-pass

*Radii= 15*

*Radii= 80*

*Radii= 5*

*Radii= 30*

*Radii= 230*

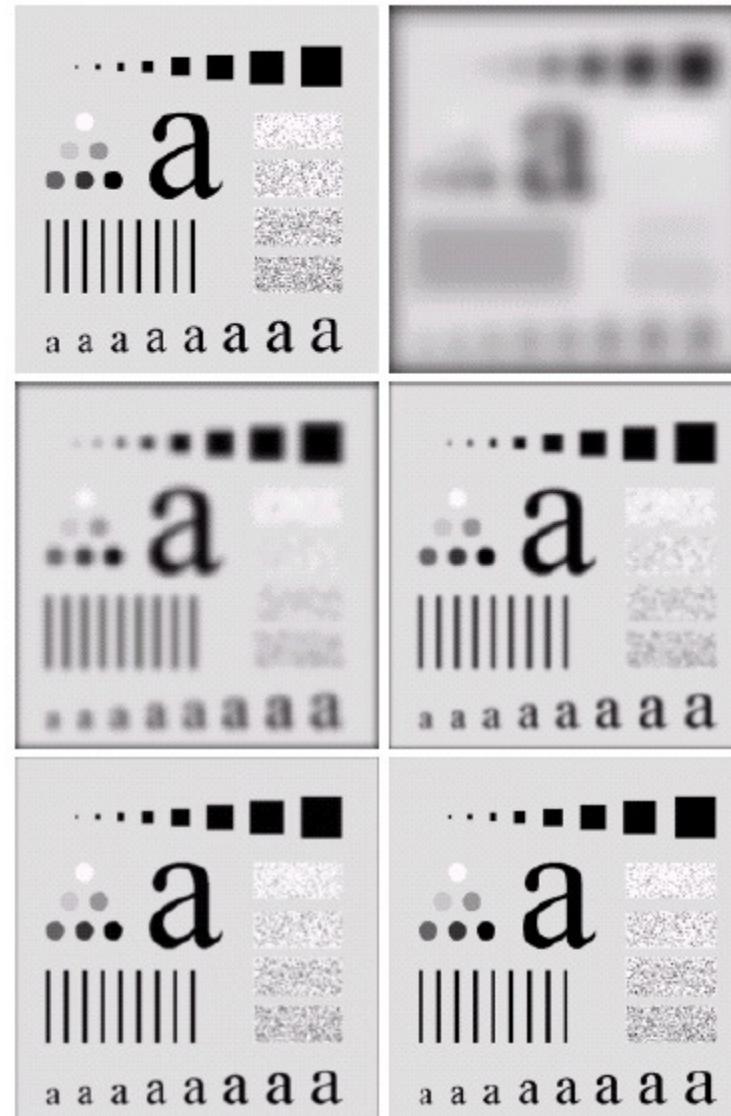


FIGURE 4.18 (a) Original image. (b)–(f) Results of filtering with Gaussian lowpass filters with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Figs. 4.12 and 4.15.

a b  
c d  
e f

# Gaussian Low-pass Filters



TABLE 4.4

Lowpass filters.  $D_0$  is the cutoff frequency and  $n$  is the order of the Butterworth filter.

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

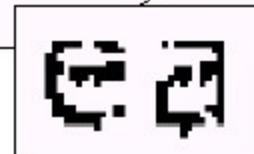
# Filtering for OCR: Gaussian Low-pass Filters

a b

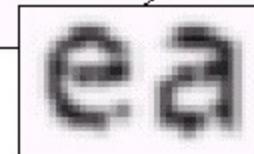
**FIGURE 4.19**

(a) Sample text of poor resolution (note broken characters in magnified view).  
(b) Result of filtering with a GLPF (broken character segments were joined).

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Filtered with a GLPF with  $D_0=80$ .  
Characters are fuller and filled in.

# Gaussian Low-pass Filters

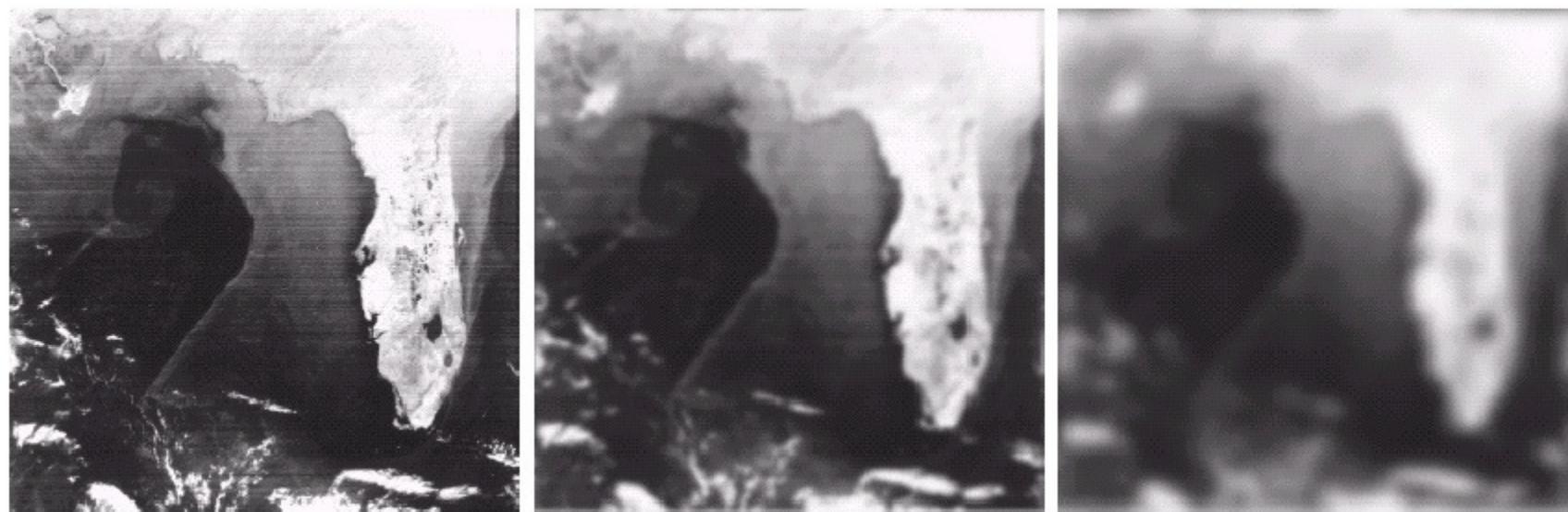


Removes wrinkles around eyes

a b c

**FIGURE 4.20** (a) Original image ( $1028 \times 732$  pixels). (b) Result of filtering with a GLPF with  $D_0 = 100$ . (c) Result of filtering with a GLPF with  $D_0 = 80$ . Note reduction in skin fine lines in the magnified sections of (b) and (c).

# Gaussian Low-pass Filters

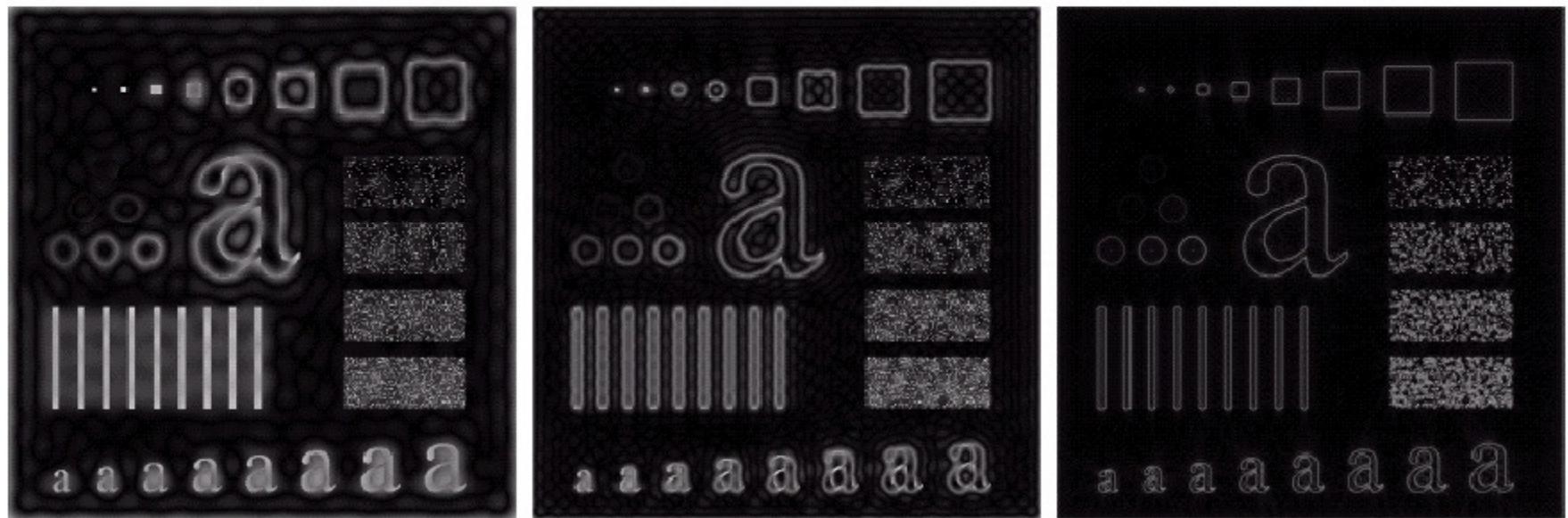


a b c

**FIGURE 4.21** (a) Image showing prominent scan lines. (b) Result of using a GLPF with  $D_0 = 30$ . (c) Result of using a GLPF with  $D_0 = 10$ . (Original image courtesy of NOAA.)

# Ideal High-pass Filters

$$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$$



a b c

**FIGURE 4.24** Results of ideal highpass filtering the image in Fig. 4.11(a) with  $D_0 = 15, 30$ , and  $80$ , respectively. Problems with ringing are quite evident in (a) and (b).

# Example

Design a IHPF of size [4x4] with centre at (2,2) and cutoff radius of 1.9.

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

# Example

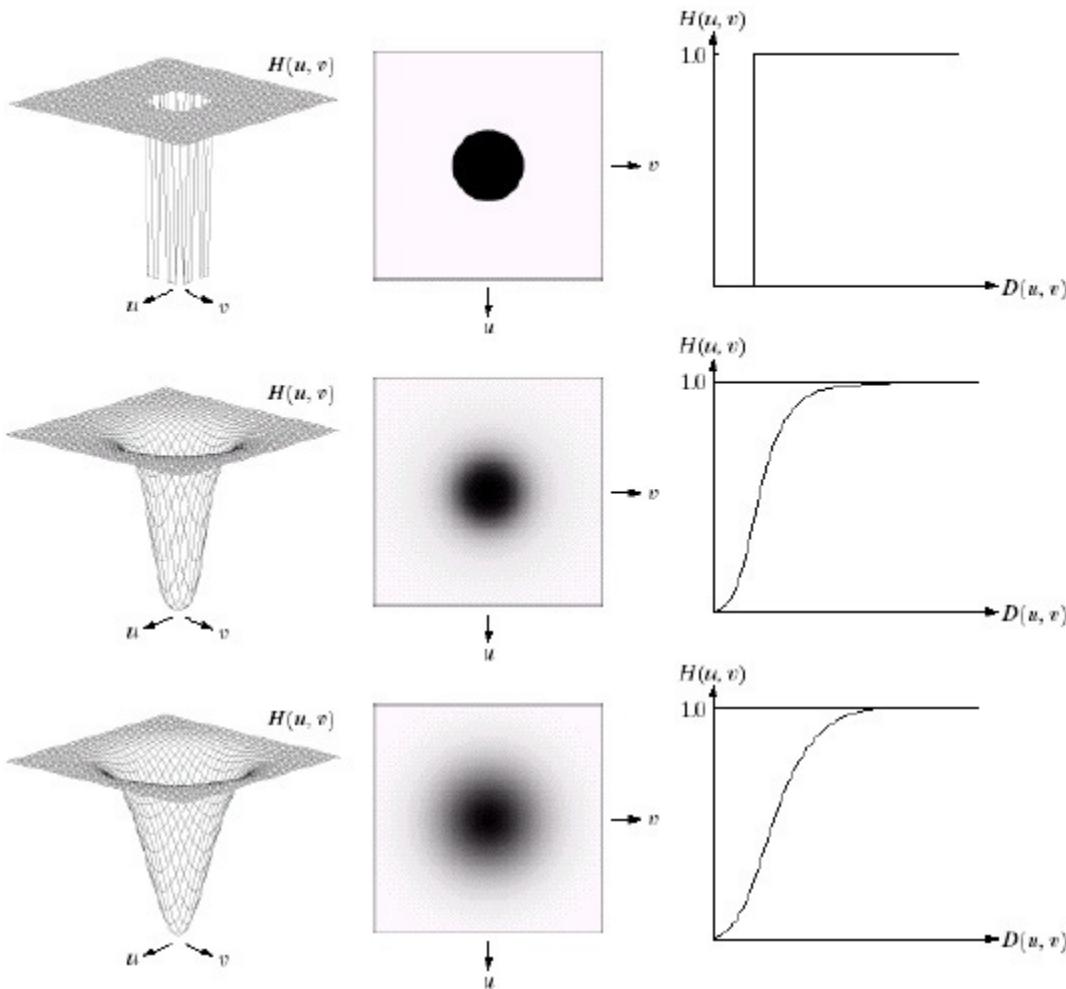
Design a ILPF of size [4x4] with centre at (2,2) and cutoff radius of 1.9.

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

$$D(k,l) = \begin{bmatrix} 2.8284 & 2.2361 & 2.0000 & 2.2361 \\ 2.2361 & 1.4142 & 1.0000 & 1.4142 \\ 2.0000 & 1.0000 & 0 & 1.0000 \\ 2.2361 & 1.4142 & 1.0000 & 1.4142 \end{bmatrix} \quad H(k,l) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$H_h(k,l) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

# Sharpening: High pass filter



Ideal highpass filter

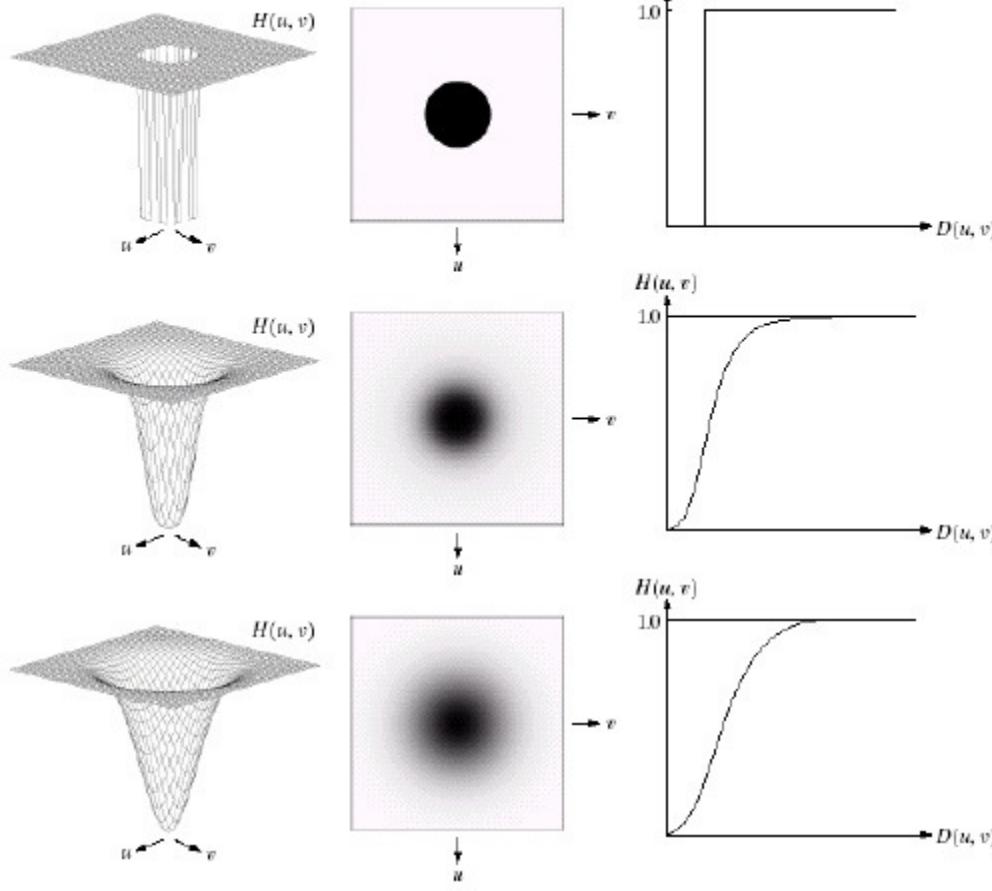
Butterworth highpass filter

Gaussian highpass filter

a	b	c
d	e	f
g	h	i

**FIGURE 4.22** Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

# Sharpening: High pass filter



a b c  
d e f  
g h i

**FIGURE 4.22** Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Ideal highpass filter

$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

$$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$$

Butterworth highpass filter

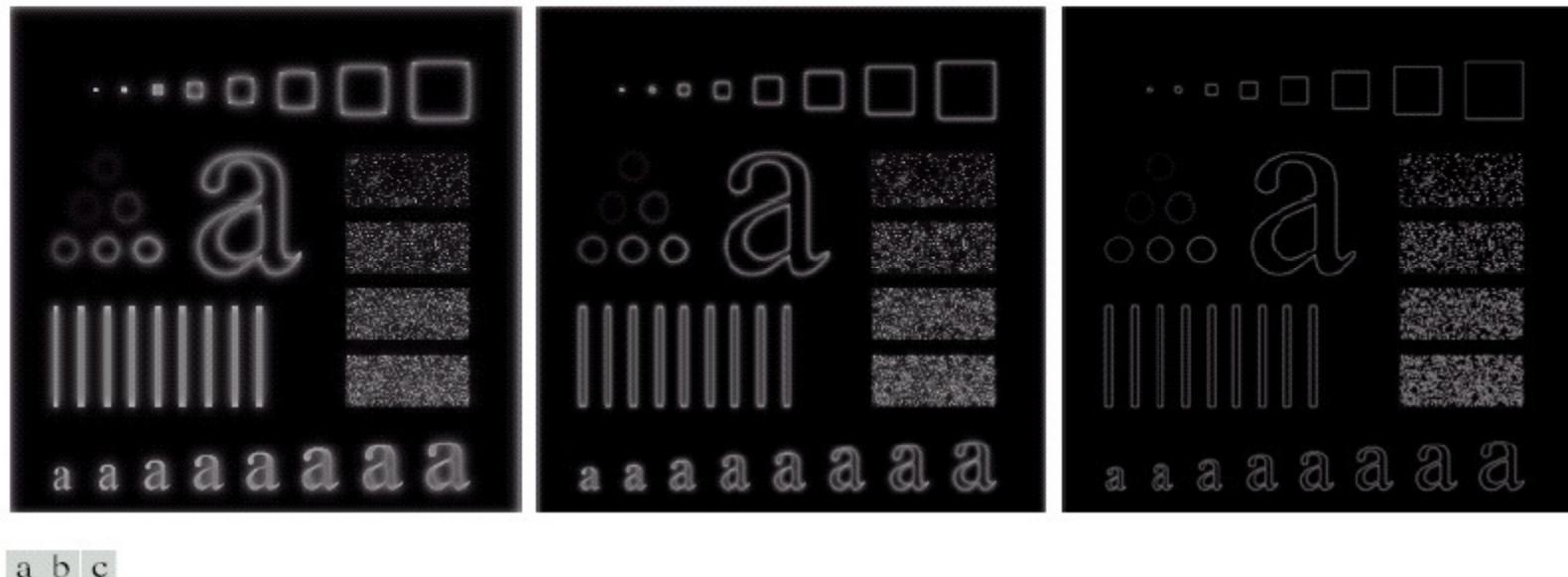
$$H(u, v) = \frac{1}{1 + [D_0 / D(u, v)]^{2n}}$$

Gaussian highpass filter

$$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2}$$

# Gaussian High-pass Filters

$$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2}$$



a b c

**FIGURE 4.26** Results of highpass filtering the image of Fig. 4.11(a) using a GHPF of order 2 with  $D_0 = 15$ , 30, and 80, respectively. Compare with Figs. 4.24 and 4.25.

# Comparisons

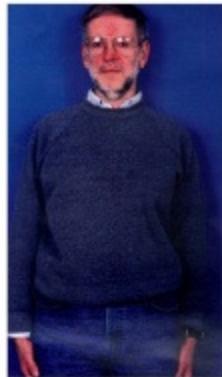
<u>Point processing</u>	<u>Spatial Processing</u>	<u>Frequency Processing</u>
<ul style="list-style-type: none"><li>• Simple gray level transformations<ul style="list-style-type: none"><li>• Image negatives</li><li>• Log transformations</li><li>• Power-law transformations</li><li>• Contrast stretching</li><li>• Gray-level slicing</li><li>• Bit-plane slicing</li></ul></li><li>• Histogram processing<ul style="list-style-type: none"><li>• Histogram equal.</li><li>• Histogram matching/specifi.</li></ul></li><li>• Arithmetic/logic<ul style="list-style-type: none"><li>• Image averaging</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Smoothing filters (blur details)<ul style="list-style-type: none"><li>• Average, weighted average</li><li>• Order statistics (e.g. median)</li></ul></li><li>• Sharpening filters (highlight details)<ul style="list-style-type: none"><li>• Unsharp masking</li><li>• High-boost filters</li><li>• Derivative filters<ul style="list-style-type: none"><li>• The Laplacian</li><li>• The Gradient</li></ul></li></ul></li></ul>	<ul style="list-style-type: none"><li>• Smoothing filters (blur details)<ul style="list-style-type: none"><li>• Ideal lowpass filter</li><li>• Butterworth lowpass</li><li>• Gaussian lowpass</li></ul></li><li>• Sharpening filters (highlight details)<ul style="list-style-type: none"><li>– Unsharp masking</li><li>– High-boost filters</li><li>– Derivative filters - The Laplacian</li><li>– Ideal highpass filter</li><li>– Butterworth highpass filter</li><li>– Gaussian highpass filter</li></ul></li></ul>

# Image Restoration

Things which we see are not by themselves what we see...

It remains completely unknown to us what the objects may be by themselves and apart from the receptivity of our senses. We know nothing but our manner of perceiving them.

**Immanuel Kant**



# Restoration Process

Image restoration attempts to restore images that have been degraded during acquisition, digitalization and transmission

- Identify the degradation process and attempt to reverse it

# Image Restoration vs. Image Enhancement

## Enhancement:

- largely a subjective process
- Priori knowledge about the degradation is not a must.
- Procedures are heuristic and take advantage of the psychophysical aspects of human visual system
- Example: contrast stretching

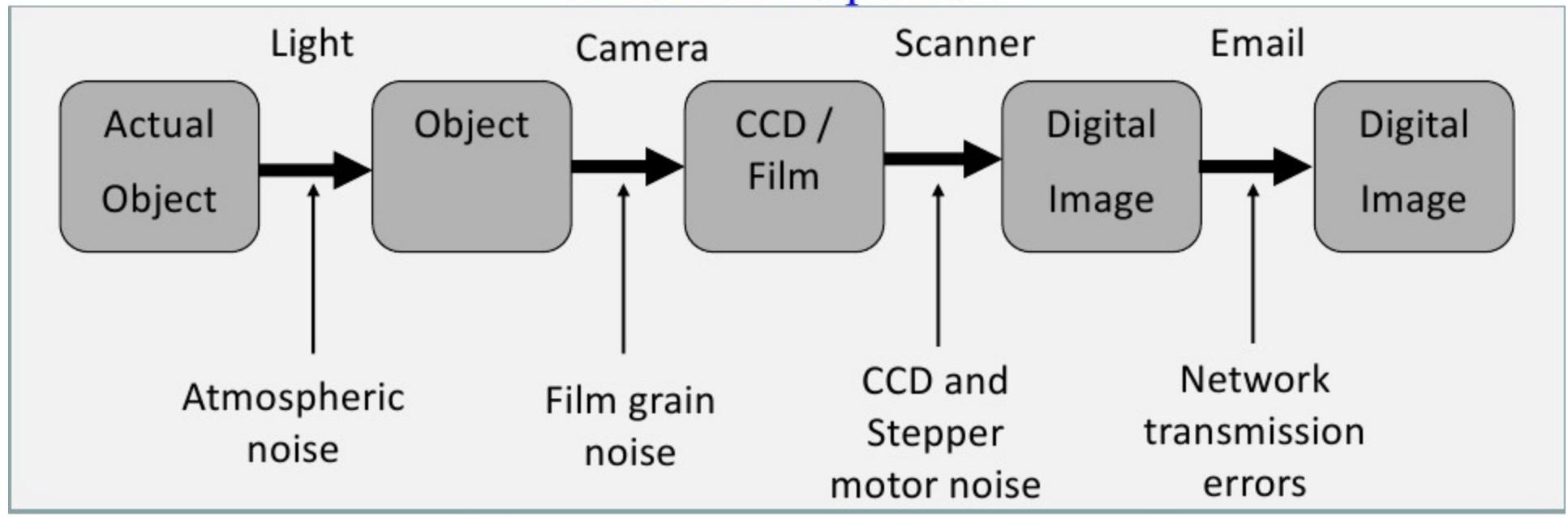
## Restoration:

- more an objective process
- Images are degraded
- Tries to recover the images by using the knowledge about the degradation process
- Example: Deblurring

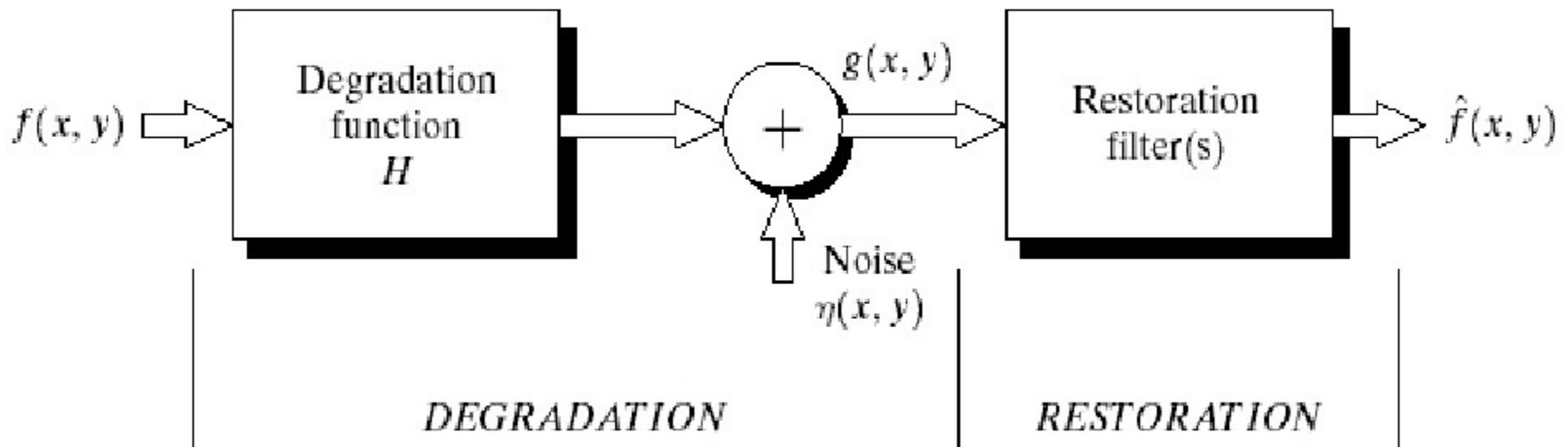
# Source of Degradation

- Imaging sensors or image acquisition can be affected by ambient conditions.
- Interference can be added to an image during transmission

Degradation may be introduced at each step in the acquisition to transmission process



# Image Degradation Model



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

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

If we know exactly  $h(x, y)$ , we can do deconvolution to get  $f(x, y)$  back from  $g(x, y)$ .

# Image Restoration

Image restoration can be done either spatial or frequency domain.

1. Spatial – if noise is additive

2. Frequency domain – Image blur

Noise will be assumed to be:

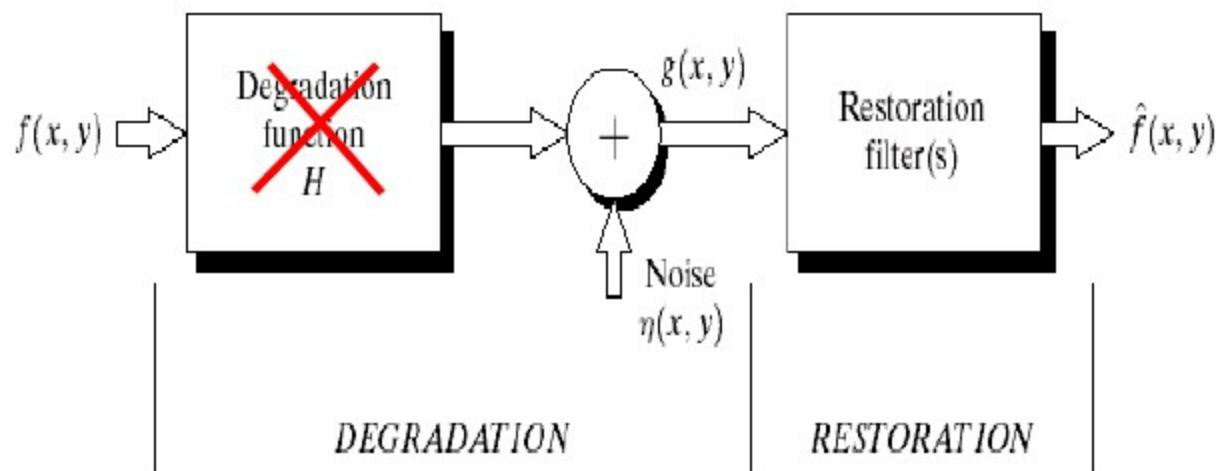
- Independent of spatial coordinates
- Uncorrelated w.r.t. the image (i.e. no correlation between pixel values and the values of noise components)

# Additive noise

Noisy image can be modelled as follows:

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

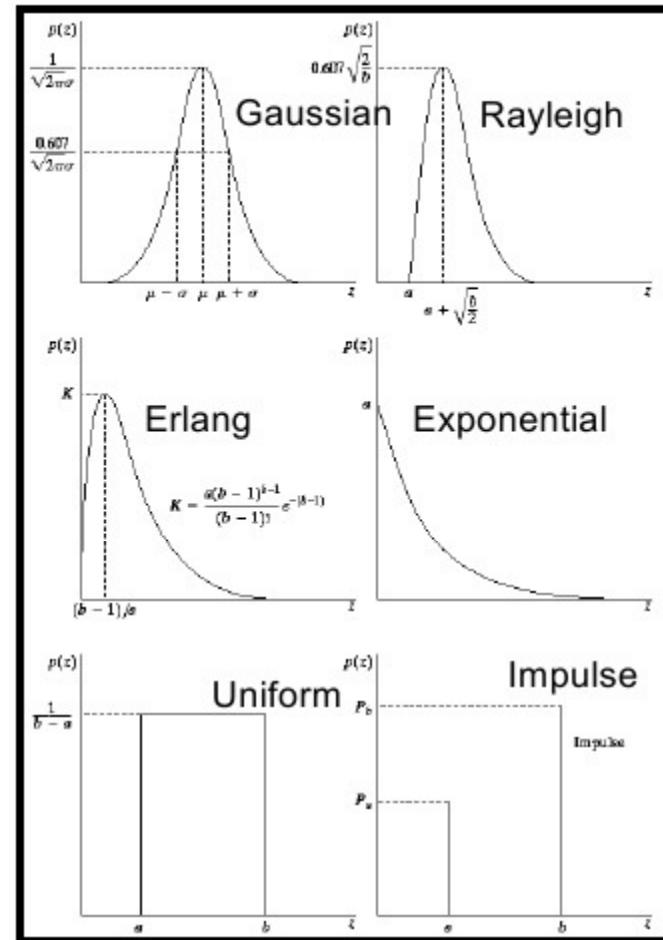
Where  $\eta(x, y)$  is the noise term and  $g(x, y)$  is the resulting noisy pixel.



# Noise Model

Noise models  $\eta(x, y)$ :

- Gaussian
  - more common noise
- Rayleigh
- Gamma
- Exponential
- Uniform
- Impulse
  - *Salt and pepper* noise

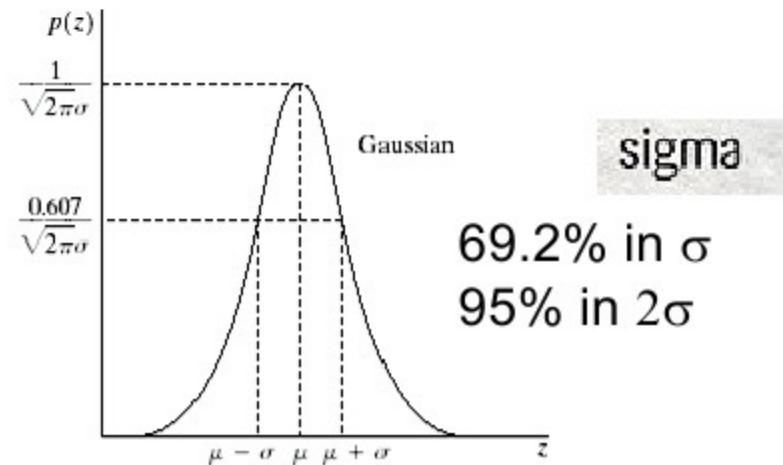


Noises are defined as random variables and defined by Probability density function (PDF) as spatial descriptor

# Gaussian noise

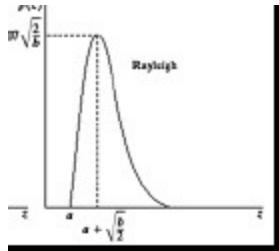
- Electronic circuit noise and sensor noise due to poor illumination and /or high temperature.
- Because of mathematical traceability in both the spatial and frequency domain, it is used frequently in practice -

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



Remark ----

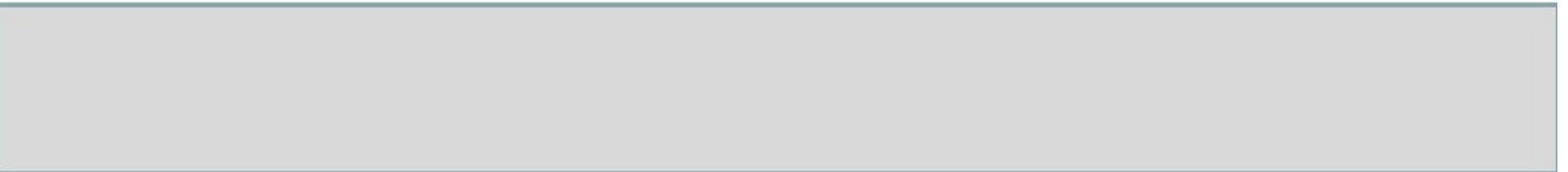
$$\int_{-\infty}^{\infty} p(z) dz = 1$$



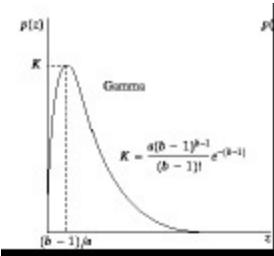
# Rayleigh noise

- The mathematical model -

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

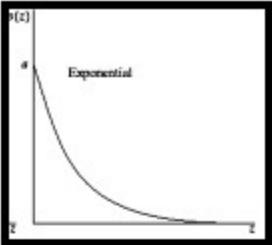


# Gamma noise



- The mathematical model -

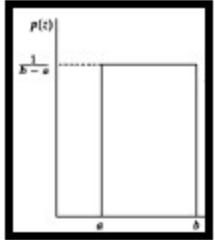
$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$



# Exponential noise

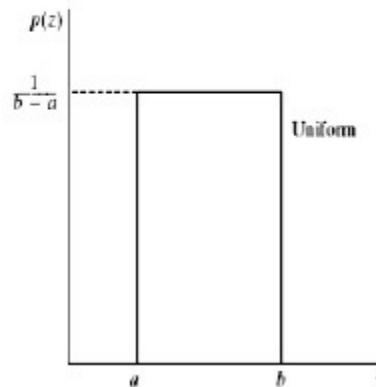
- The mathematical model -

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



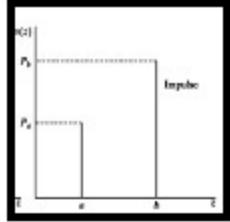
## Uniform noise

- Least descriptive, less practical, used for random number generator



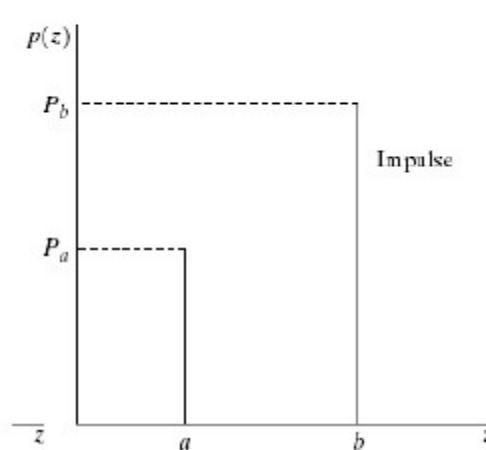
$$p(z) = \begin{cases} \frac{1}{b-a} & ; \text{ if } a \leq z \leq b \\ 0 & ; \text{ otherwise} \end{cases}$$





# Impulse (salt-and-pepper) noise

- Quick transients, such as faulty switching during imaging



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

**Remark-** In practical, impulses are usually stronger than image signals.

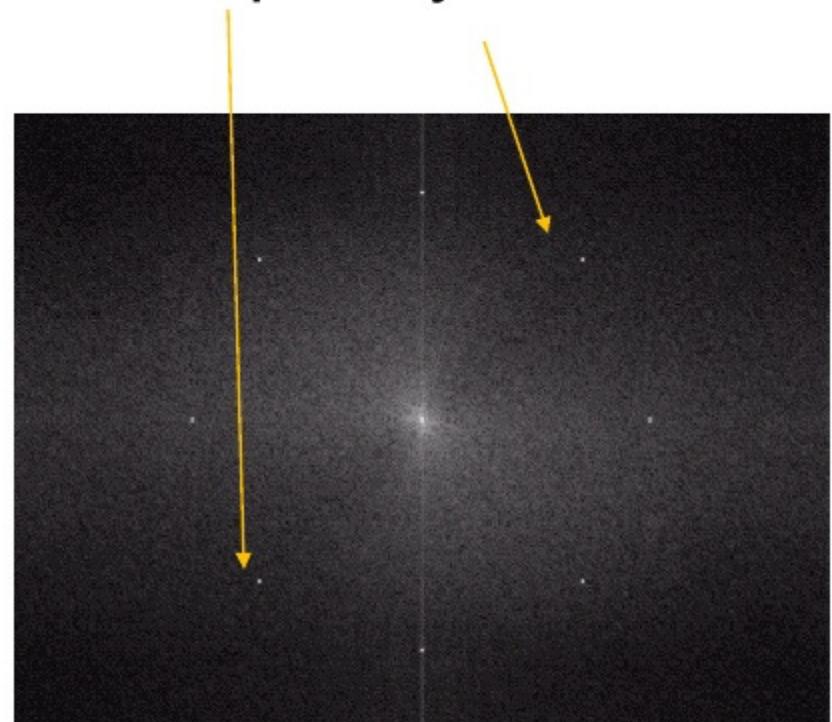
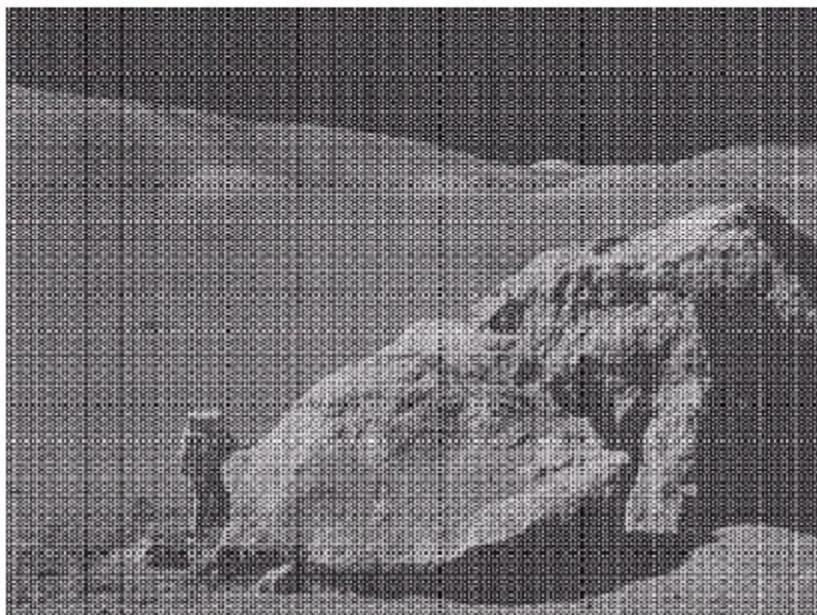
In gray 8-bit image – a=0(black) and b=255(white)

# Periodic noise

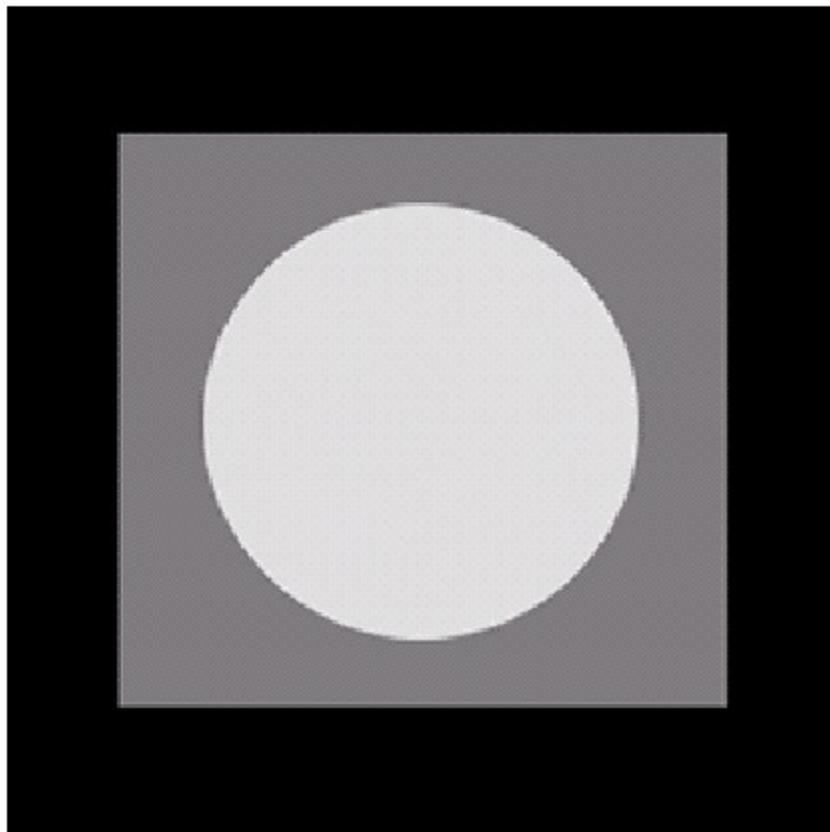
- Arise from electrical or electromechanical interference during image acquisition or transmission
- Spatial dependence
- Observed in the frequency domain

# Periodic noise

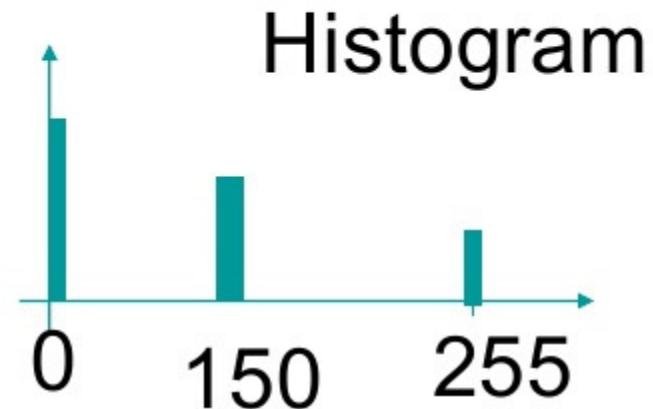
Complex conjugate pair  
in frequency domain



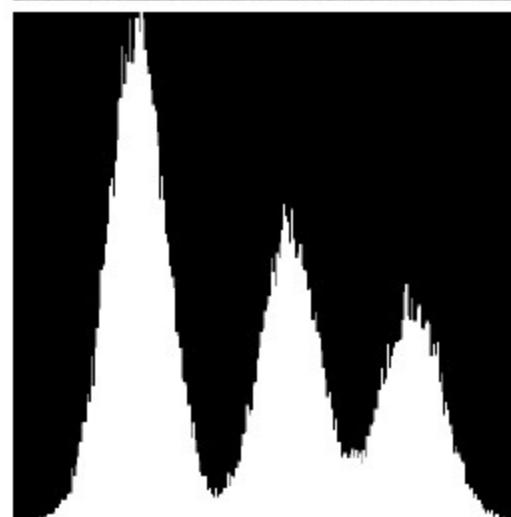
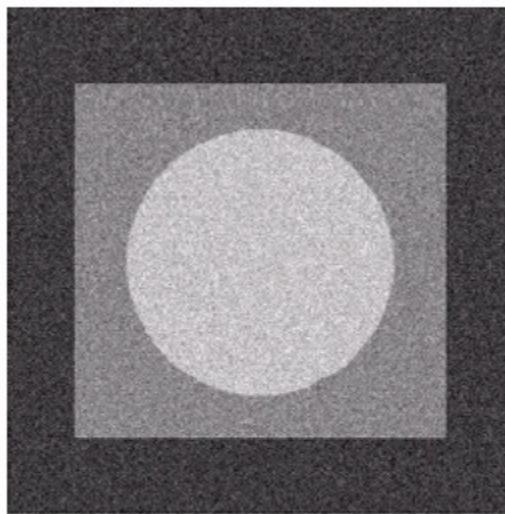
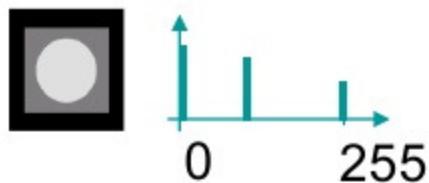
# Test Image



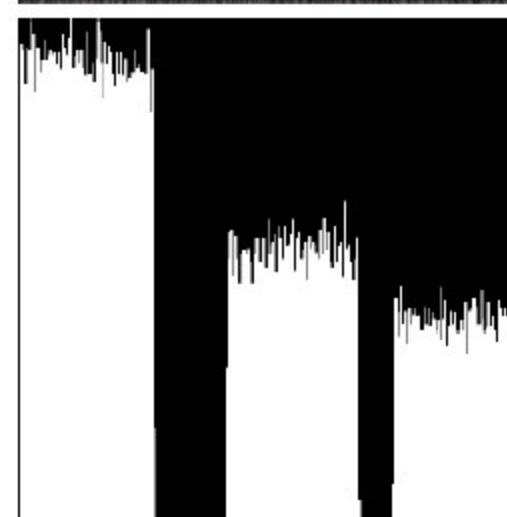
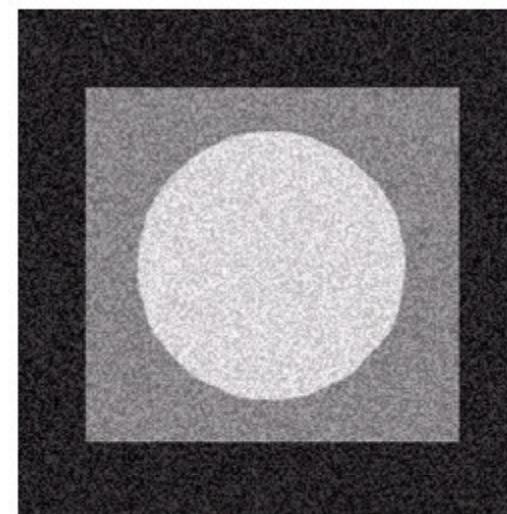
This test pattern is well-suited for illustrating the noise models



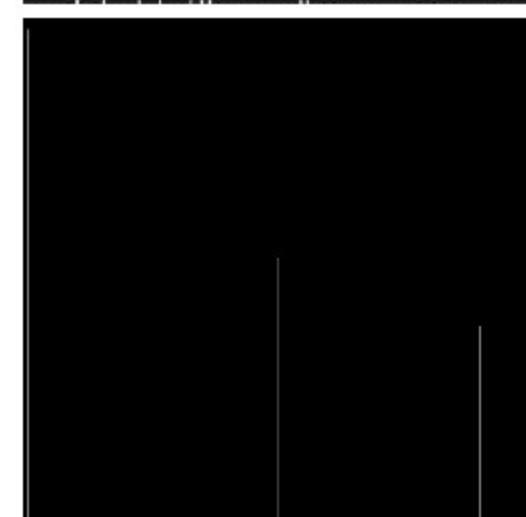
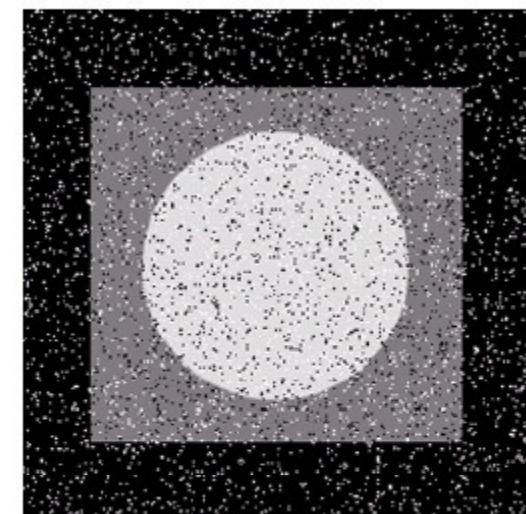
Test pattern is composed of simple, constant areas that span the grey scale from black to white in only three increments. This facilitates visual analysis of the characteristics of the various noise components added to the image.



**Gaussian**

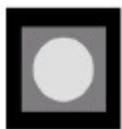


**Uniform**

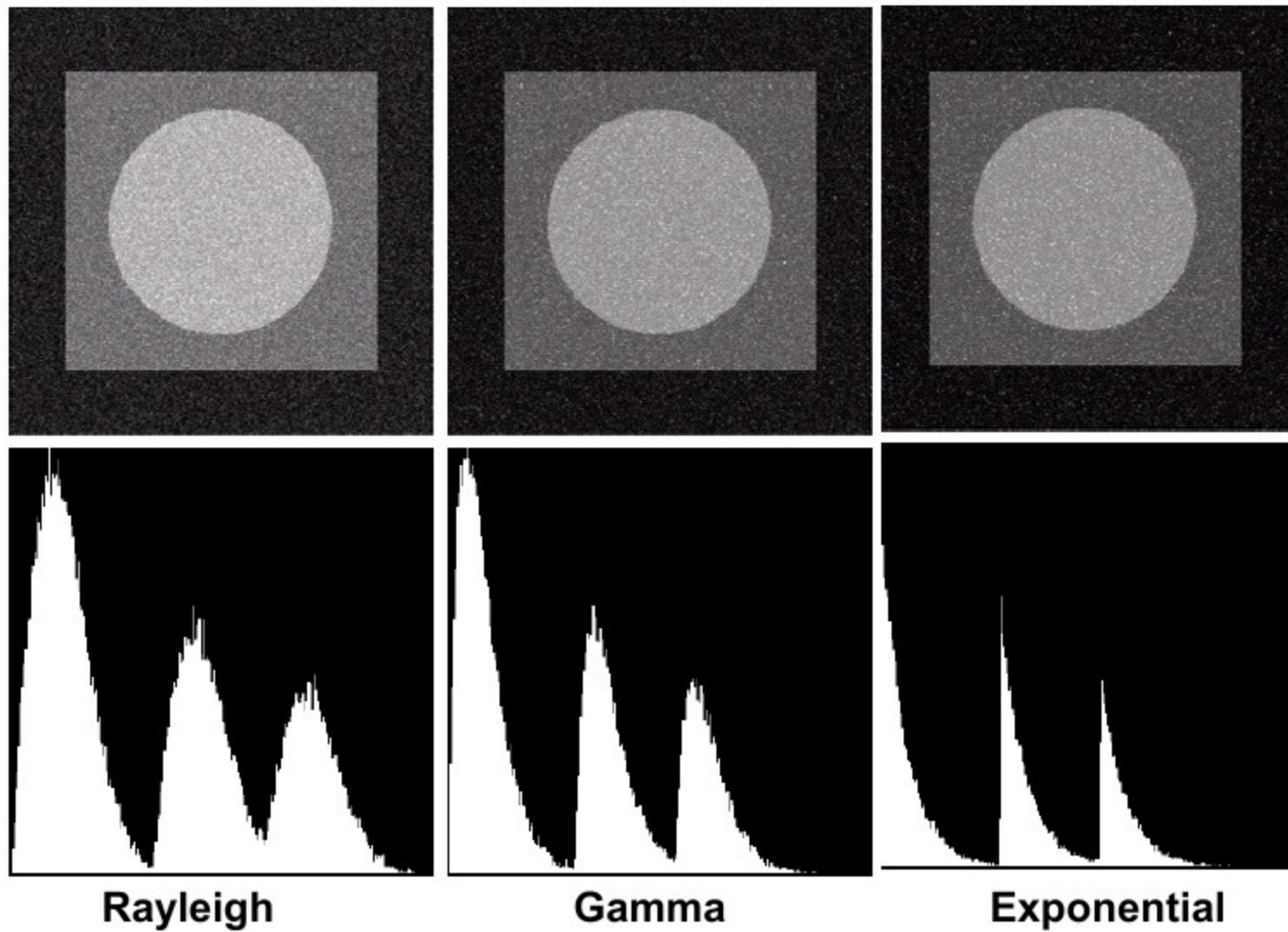


**Salt & pepper**

**Test Image**



# Test Image



# Estimation of Noise Parameters

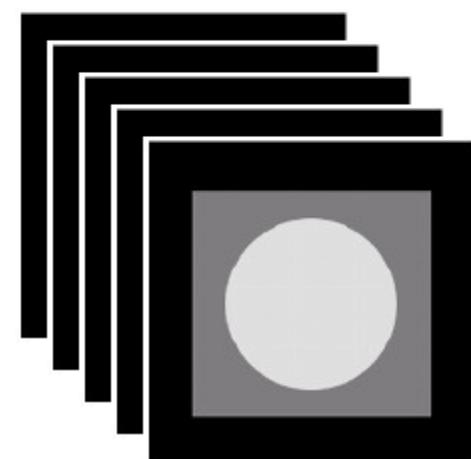
## Case 1: Imaging system is available

- Capture images of “flat” environment



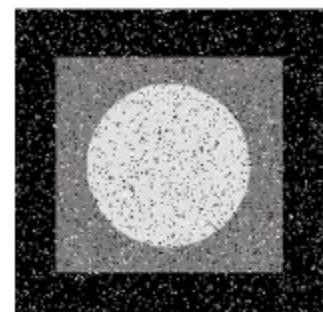
## Case 2: Imaging system is available

- Capture multiple images.



## Case 3: Noisy images available

- Take a strip from constant area
- Draw the histogram and observe it
- Measure the mean and variance



## Spatial filters for additive noise

- Mean filters
  - Arithmetic, Geometric, Harmonic, Contra Harmonic
- Order-statistics filters
  - Median, max & min, Mid-point
- Adaptive filters
  - Wiener Filter

## Arithmetic Mean Filter

It is a very simple one and restored pixel is calculated as follows:

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

Where  $g(s,t)$  is noisy image  
 $m \times n$  is size of filter

This is implemented as the simple smoothing filter  
It blurs the image.

GM is calculated as follows:

$$\tilde{g} = \left[ \prod_{(s,t) \in S_{xy}} g(s,t) \right]^{1/mn}$$

Achieves similar smoothing to the arithmetic mean, but tends to lose less image detail.

### Harmonic Mean

$$\tilde{g} \sim \frac{mn}{\sum_{(s,t) \in S_{xy}} g(s,t)}$$

Works well for salt noise, but fails for pepper noise.  
Also does well for other kinds of noise such as Gaussian noise.

## Contra harmonic Mean Filter

### Contra harmonic Mean--

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

*Q* is the *order* of the filter.

Positive values of  $Q > 0$  eliminate pepper noise.

Negative values of  $Q < 0$  eliminate salt noise.

It cannot eliminate both simultaneously.

## Order Statistics Filters

The response of the order statistics filters is based on ordering or ranking of pixels values that make up the neighbourhood defined by the filter support.

- Median filter
- Max and min filter
- Midpoint filter
- Alpha trimmed mean filter

The median filter removes a pixel value using the median value of its neighboring pixel. The output of this filter is given as follows:

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

- Excellent at noise removal, without the smoothing effects that can occur with other smoothing filters.
- Particularly good when salt and pepper noise is present.

## Max and Min Filter

The output of the max filter is given as follows:

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

The output of the min filter is given as follows:

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

Max filter is good for pepper noise and Min filter is good for salt noise.

## Mid-Point Filter

The midpoint between the minimum and maximum values of intensity pixels is calculated. This filter is a combination of an order statistics filter and average filter. It removes Gaussian noise and uniform noise from the noisy image.

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

Max filter is good for pepper noise and Min filter is good for salt noise.

## Alpha-Trimmed Mean Filter

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

$g_r(s, t)$  represents the remaining  $mn - d$  pixels.

# Estimating the Degradation Function

## Degradation Model

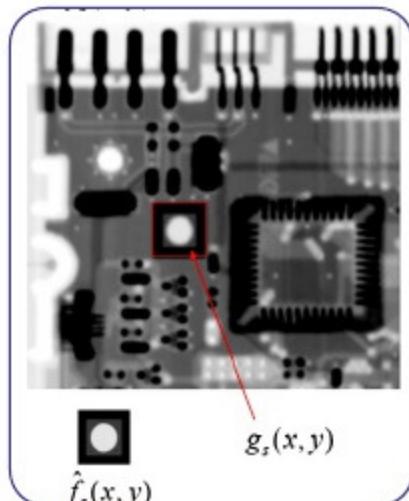
$$g(x,y) = f(x,y) * h(x,y) + \eta(x,y)$$

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

1. Estimation by Image Observation
2. Estimation by Experiment
3. Estimation by Modeling

## Observe the Image

- Select a small section of image containing simple structure (edge, point) from the degraded image
  - Reconstruct an unblurred image of the same size.
  - The degradation function can be estimated by :

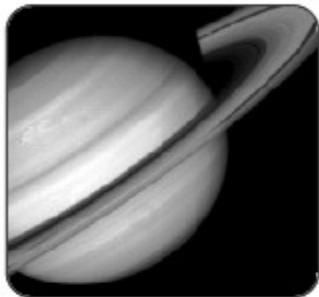


$$H(u,v) \approx H_s(u,v) = \frac{G_s(u,v)}{\hat{F}_s(u,v)}$$

# Observe the Image

## Single Image

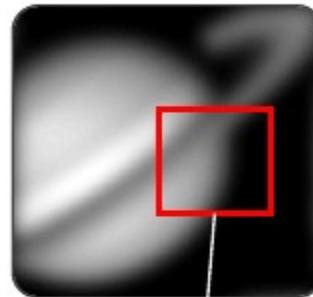
Original image (unknown)



$f(x,y)$

$$f(x,y)*h(x,y)$$

Degraded image



$g(x,y)$

Estimated Transfer function

$$H(u,v) \approx H_s(u,v) = \frac{G_s(u,v)}{\hat{F}_s(u,v)}$$

Observation

Subimage

$g_s(x,y)$



Restoration process by estimation

Reconstructed Subimage



$\hat{f}_s(x,y)$

## Estimation by Experiment

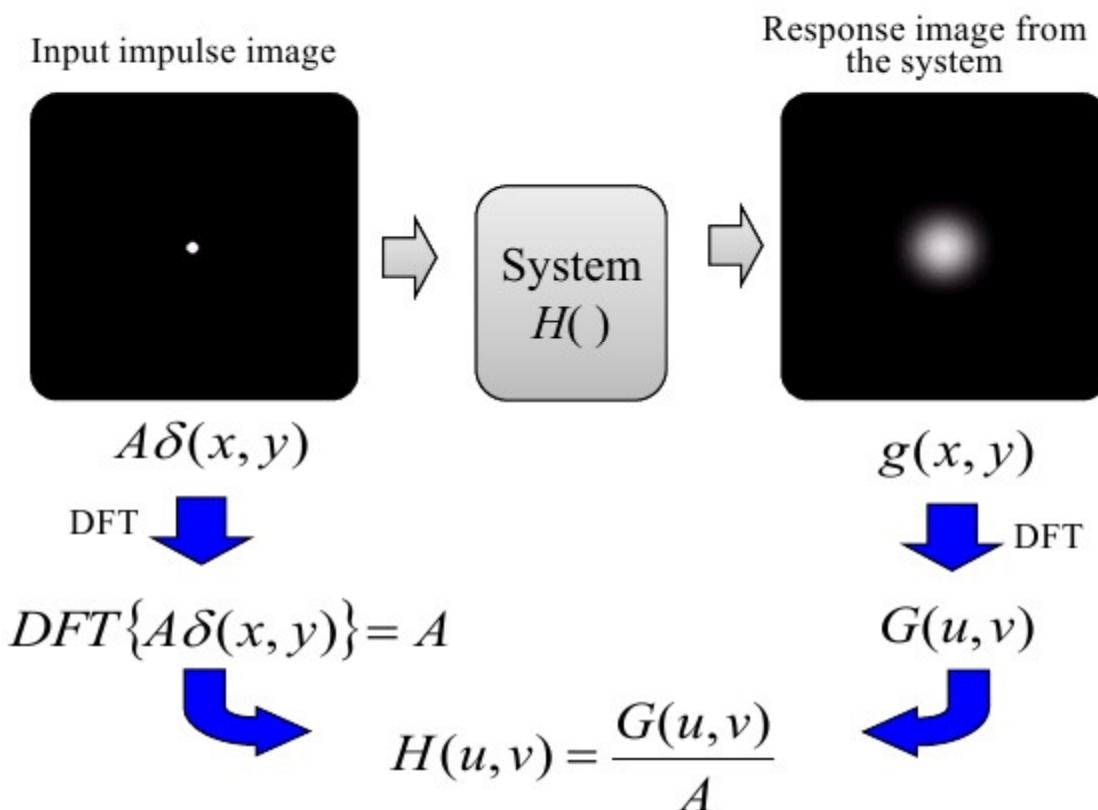
If we have the equipment used to acquire degraded image we can obtain accurate estimation of the degradation

- Obtain an impulse response of the degradation using the system setting
- A linear space-invariant system is characterized completely by its impulse response

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

# Estimation by Experiment

Used when we have the same equipment set up and can repeat the experiment.



## Estimation by Modeling

Mathematical model of degradation can be for example atmosphere turbulence

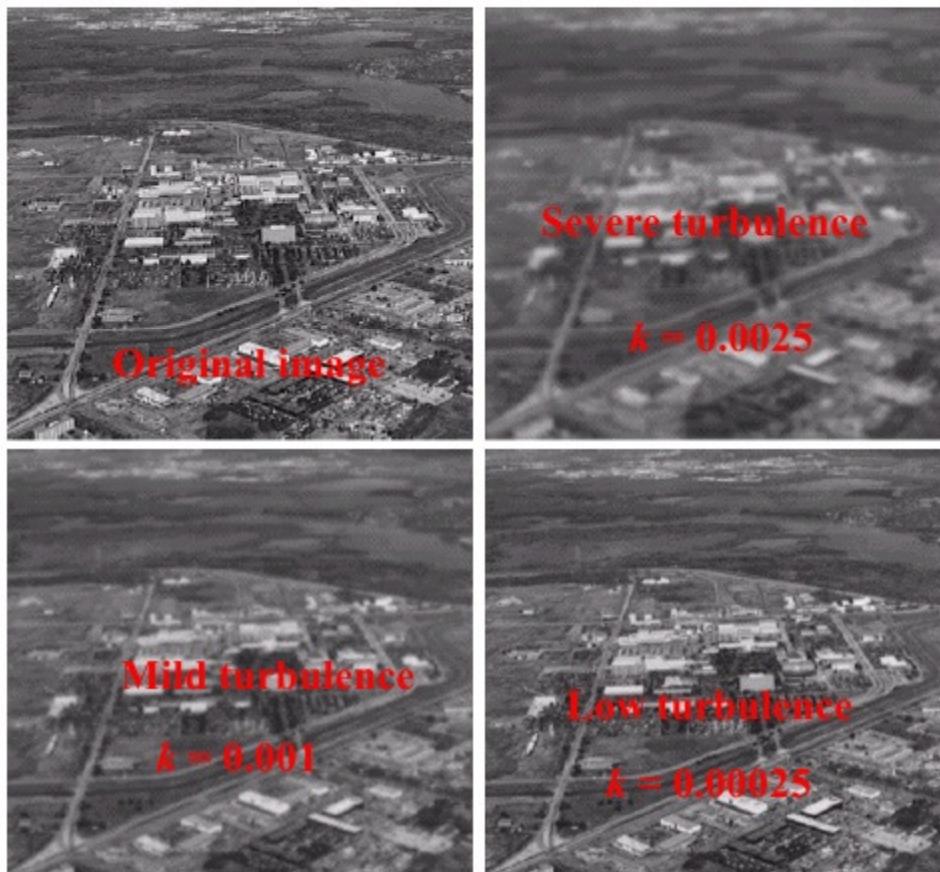
Example: Hufnagel & Stanley (1964) has established a degradation model due to atmospheric turbulence

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

K is a parameter to be determined by experiments because it changes with the nature of turbulence

## Estimation by Modeling

Used when we know physical mechanism underlying the image formation process that can be expressed mathematically.



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

## Estimation by Modeling: Motion Blurring

Assume that camera velocity is  $x_0(t)$  and  $y_0(t)$

The blurred image is obtained by

$$g(x, y) = \int_0^T f(x + x_0(t), y + y_0(t)) dt$$

where  $T$  = exposure time.

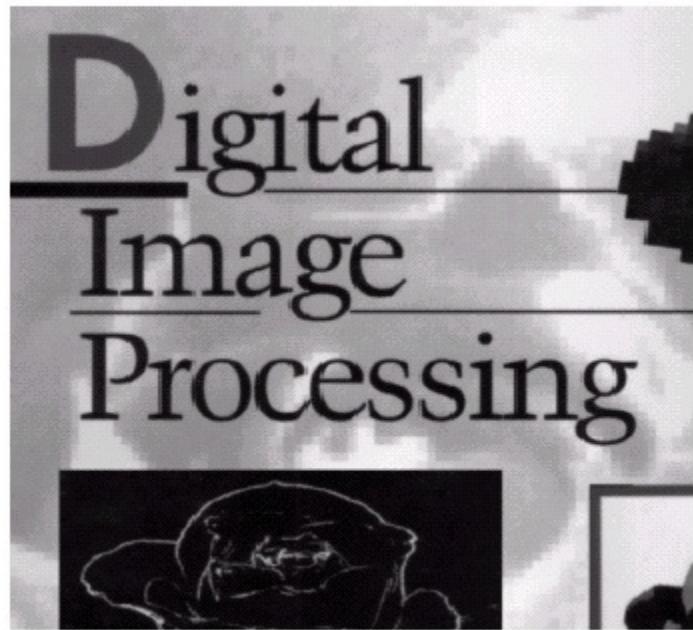
## Estimation by Modeling: Motion Blurring

$$g(x, y) = \int_0^T f(x + x_0(t), y + y_0(t)) dt$$

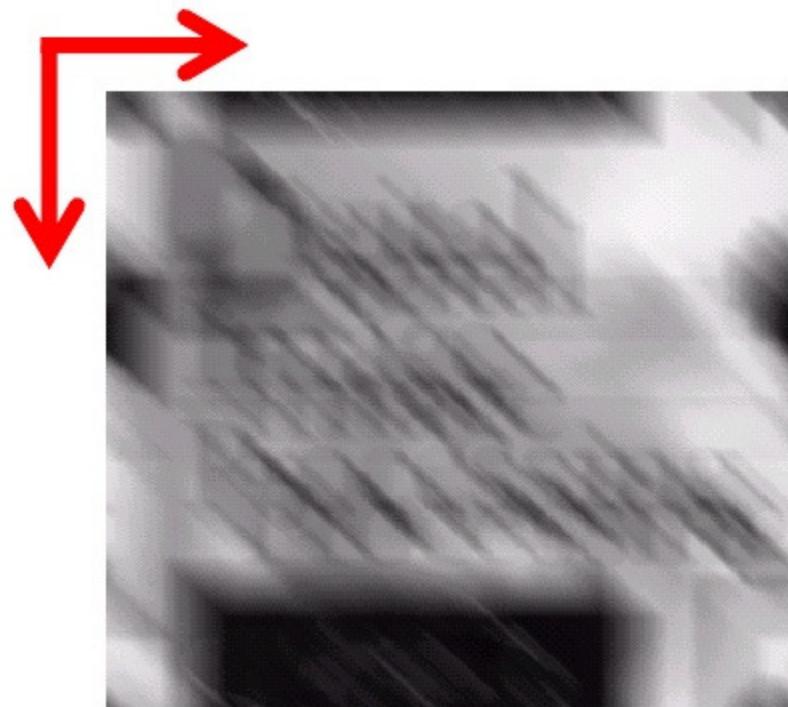
$$\begin{aligned} G(u, v) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-j2\pi(ux+vy)} dx dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[ \int_0^T f(x + x_0(t), y + y_0(t)) dt \right] e^{-j2\pi(ux+vy)} dx dy \\ &= \int_0^T \left[ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x + x_0(t), y + y_0(t)) e^{-j2\pi(ux+vy)} dx dy \right] dt = F(u, v)H(u, v) \end{aligned}$$

$$H(u, v) = \frac{1}{T} \int_0^T e^{-j2\pi(ux_0(t)+vy_0(t))} dt$$

## Estimation by Modeling: For constant motion



Original image



Motion blurred image

$$x_0(t) = y_0(t) = 0.1; T = 1$$

# Estimation by Modeling: For constant motion

Horizontal

X	X	X	X	X	

a. Horizontal PSF mask; x = nonzero term.

Vertical

		X		
		X		
		X		
		X		
		X		

b. Vertical PSF mask; x = nonzero term.

Diagonal

X				
	X			
		X		
			X	
				X

c. Diagonal PSF mask; x = nonzero term.

## Motion Blur Mask Coefficients

# Estimation by Modeling: For constant motion

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0

a. Horizontal PSF mask with uniform blur.

0	0	1	0	0
0	0	1	0	0
0	0	4	0	0
0	0	1	0	0
0	0	1	0	0

b. Vertical PSF mask with center-weighting.

1	0	0	0	0
0	2	0	0	0
0	0	4	0	0
0	0	0	2	0
0	0	0	0	1

c. Diagonal PSF mask with gaussian distribution.

0	1	1	1	0
1	2	4	2	1
1	4	8	4	1
1	2	4	2	1
0	1	1	1	0

d. Circular PSF mask with gaussian distribution.

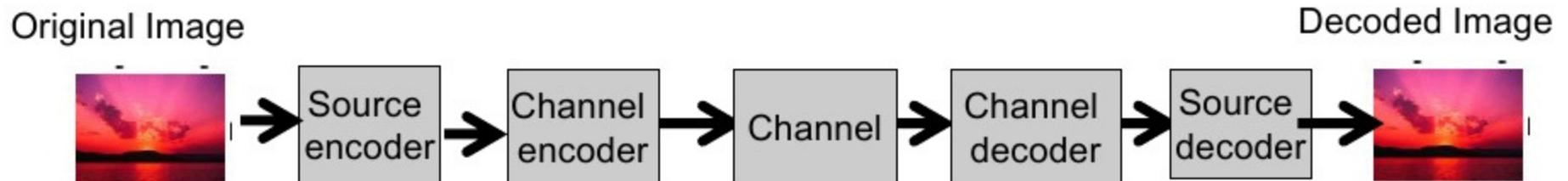
## Typical Blur Mask Coefficients

# Image Compression

Image compression is the art and science of reducing the amount of data required to represent an Image.

The compression algorithm encodes the original image with few bits.

The objective of image compression is to reduce the redundancy of the image and to store or transmit data in an efficient form.



## Image Compression

The amount of data required to represent a 2 hour *standard definition (SD) television* movie using  $720 \times 480 \times 24$  bit pixel arrays.

$$30 \frac{\text{frames}}{\text{sec}} \times (720 \times 480) \frac{\text{pixels}}{\text{frame}} \times 3 \frac{\text{bytes}}{\text{pixel}} = 31,104,000 \text{ bytes/sec}$$

$$31,104,000 \frac{\text{bytes}}{\text{sec}} \times (60^2) \frac{\text{sec}}{\text{hr}} \times 2 \text{ hrs} \cong 2.24 \times 10^{11} \text{ bytes}$$

$$= 224 \text{ G bytes}$$

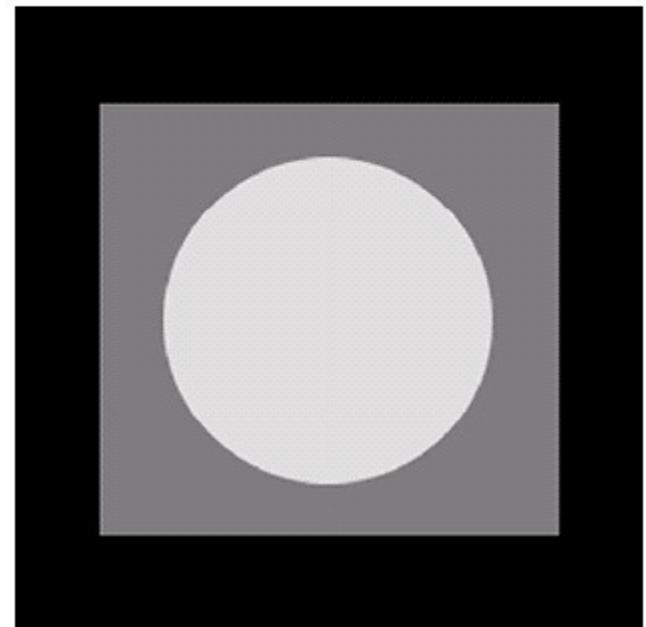
$$= 0.224 \text{ T bytes}$$

## Compression Ratio

$$C_R = \frac{n_1}{n_2}; \quad \begin{matrix} n_1 & \text{number of bits before} \\ n_2 & \text{number of bits after} \end{matrix}$$

## Relative Data Redundancy

$$R_D = 1 - \frac{1}{C_R}$$



**Data redundancy is defined** as the storing of the same **data** in less number of bits than standard form. An example of **data redundancy** is saving the same file in three bits by coding.

## Compression and Redundancy

**Compression Ratio:**  $C_R = \frac{n_1}{n_2}$ ;

- Case 1:  $n_2 = n_1$

$C_R = 1$  and  $R_D = 0$  the first data set contains no redundant information

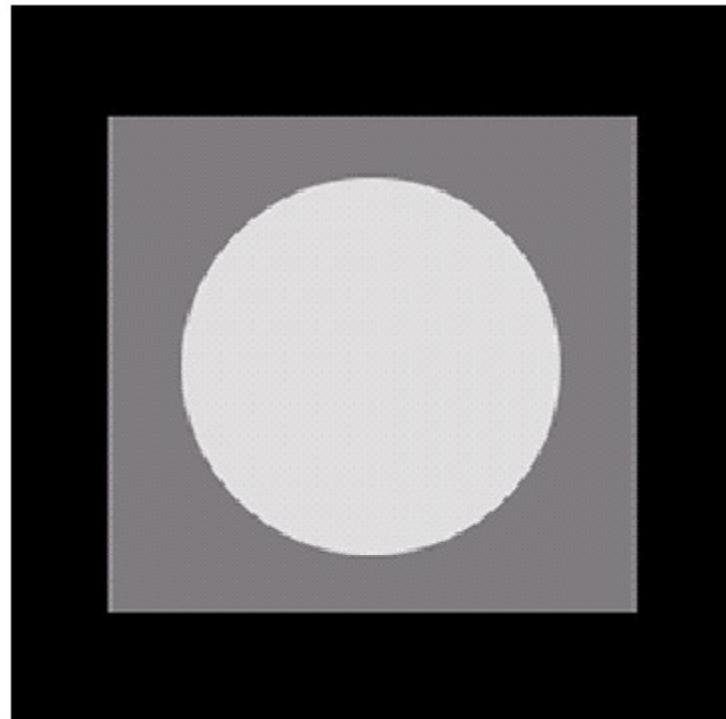
- Case 2:  $n_2 \ll n_1$

$C_R \rightarrow \infty$  and  $R_D \rightarrow 1$  highly redundant data significant compression

- Case 3:  $n_2 \gg n_1$

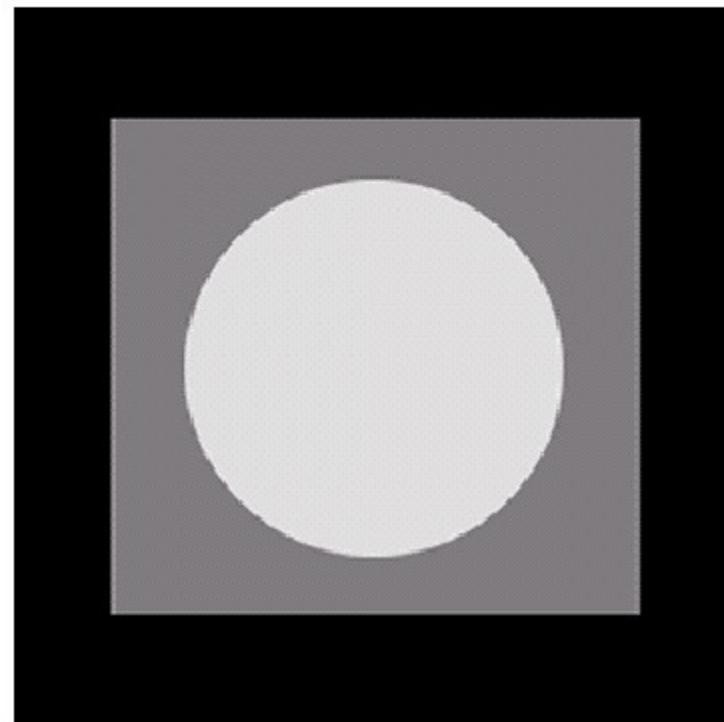
$C_R \rightarrow 0$  and  $R_D \rightarrow -\infty$  (undesirable) data expansion

# Types of Image redundancy



1. Coding redundancy
2. Interpixel redundancy
3. Psychovisual redundancy

## Coding redundancy



A gray image (512x512) pixels have three gray shades and pixel intensity can be coded two bits.

## Example: Coding Redundancy

A gray image (512x512) pixels have eight gray shades and coded by **fixed and variable length codes**. The probability of each shade is given below

<b>Gray value</b>	0	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{4}{7}$	$\frac{5}{7}$	$\frac{6}{7}$	1
P(r)	0.19	0.25	0.21	0.16	0.08	0.06	0.03	0.02

Find the memory required to store this image.

## Example: Coding Redundancy

Fixed Length Coding---

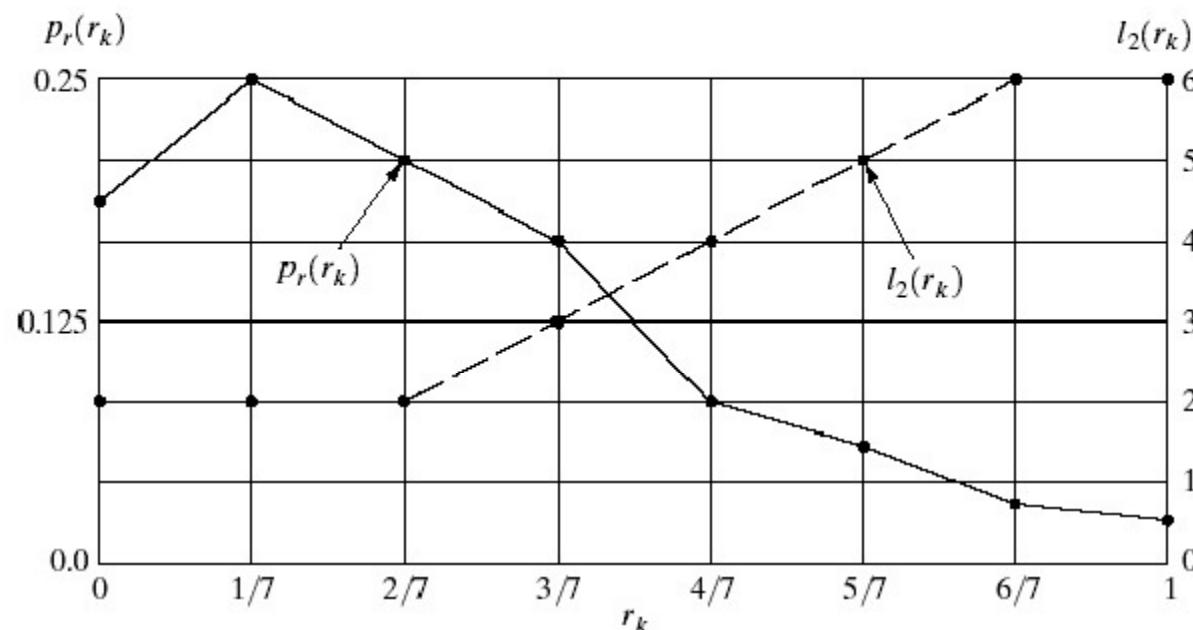
Gray value	0	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{4}{7}$	$\frac{5}{7}$	$\frac{6}{7}$	1
P(r)	0.19	0.25	0.21	0.16	0.08	0.06	0.03	0.02
codes	000	001	010	011	100	101	110	111

Average code length = 3 bits/pixel

Memory required =  $512 \times 512 \times 3 = 786432$  bits

## Example: Coding Redundancy

Variable Length Coding Concept: assign the longest code word to the symbol with the least probability of occurrence.



**FIGURE 8.1**  
Graphic representation of the fundamental basis of data compression through variable-length coding.

Gray value	0	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{4}{7}$	$\frac{5}{7}$	$\frac{6}{7}$	1
P(r)	0.19	0.25	0.21	0.16	0.08	0.06	0.03	0.02
Codes $l_i$	11	01	10	001	0001	00001	000001	000000

## Example: Coding Redundancy

Variable Length Coding:

<b>Gray value</b>	0	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{4}{7}$	$\frac{5}{7}$	$\frac{6}{7}$	1
<b>P(r)</b>	<b>0.19</b>	<b>0.25</b>	<b>0.21</b>	<b>0.16</b>	<b>0.08</b>	<b>0.06</b>	<b>0.03</b>	<b>0.02</b>
<b>Codes <math>l_i</math></b>	11	01	10	001	0001	00001	000001	000000

Average code length  $L_{avg} = \sum_i p_i l_i = 2.7$  bits/pixel

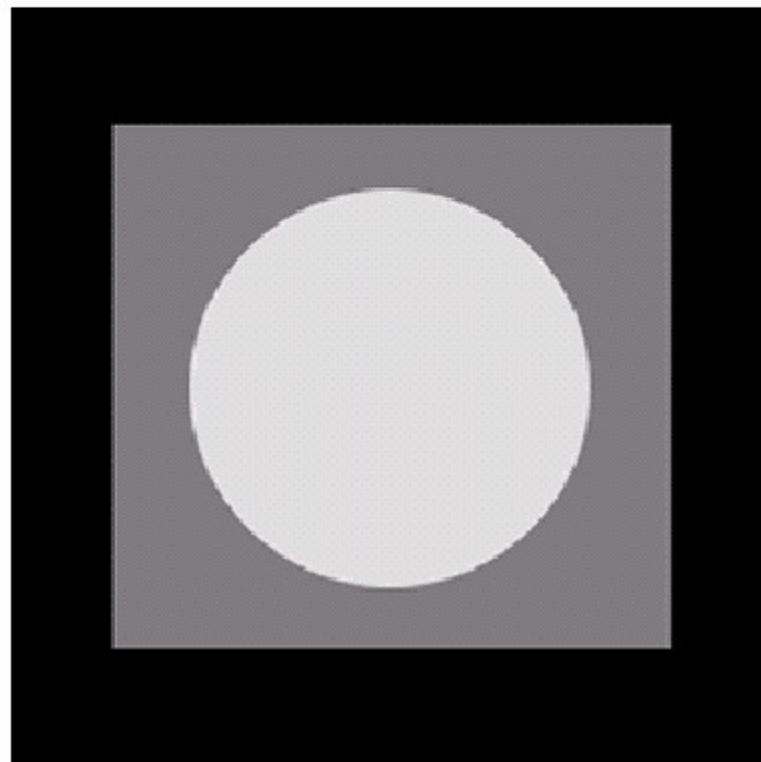
Memory required =  $512 \times 512 \times 2.7 = 707788.8$  bits

Compression ratio, relative redundancy –

$$C_R = \frac{n_1}{n_2}; \quad R_D = 1 - \frac{1}{C_R}$$

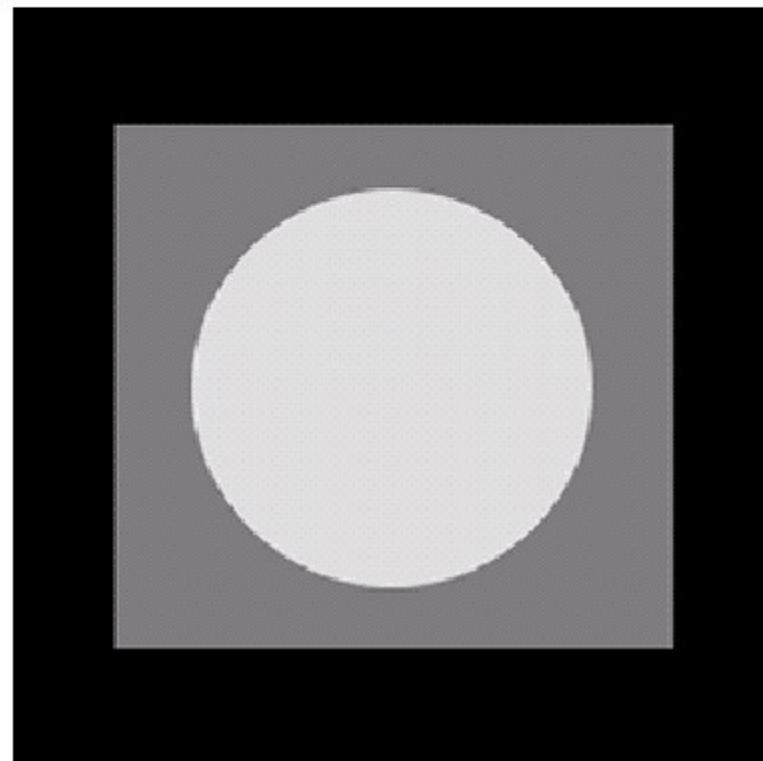
## Inter-pixel Redundancy

The pixels of most images are correlated spatially:



## Inter-pixel Redundancy

The pixels of most images are correlated spatially:

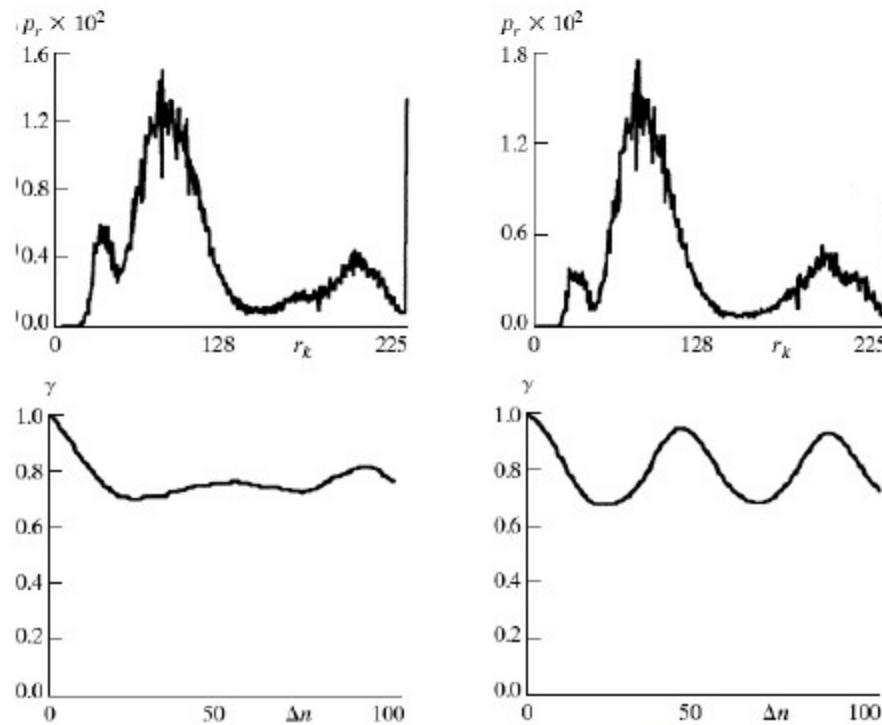
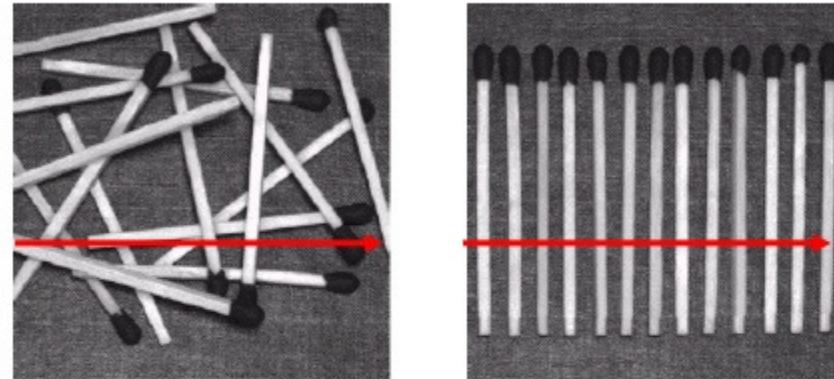


# Inter-pixel Redundancy

Parts of an image are highly correlated

Pixel intensity can be Predicted from its neighbor.

Autocorrelation in row



a  
b  
c  
d  
e  
f

FIGURE 8.2 Two images and their gray-level histograms and normalized autocorrelation coefficients along one line.

## Inter-pixel Redundancy

**Run-length encoding (RLE)** is a very simple form of data compression in which **runs of data** (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.

Example- ....000001110000001111.....

....00000        111        000000        1111.....

Code as ....(0,5),(1,3),(0,6),(1,4).....

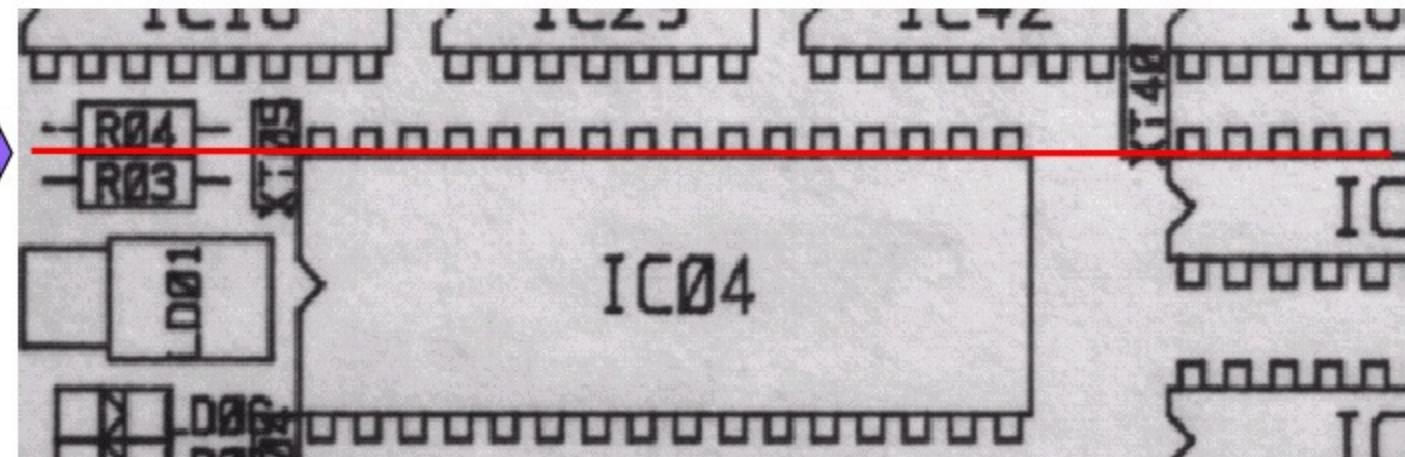
Code words – (0101), (1011), (0110), (1100) ....

Application- Binary image, Fax etc.

## Example: RL coding

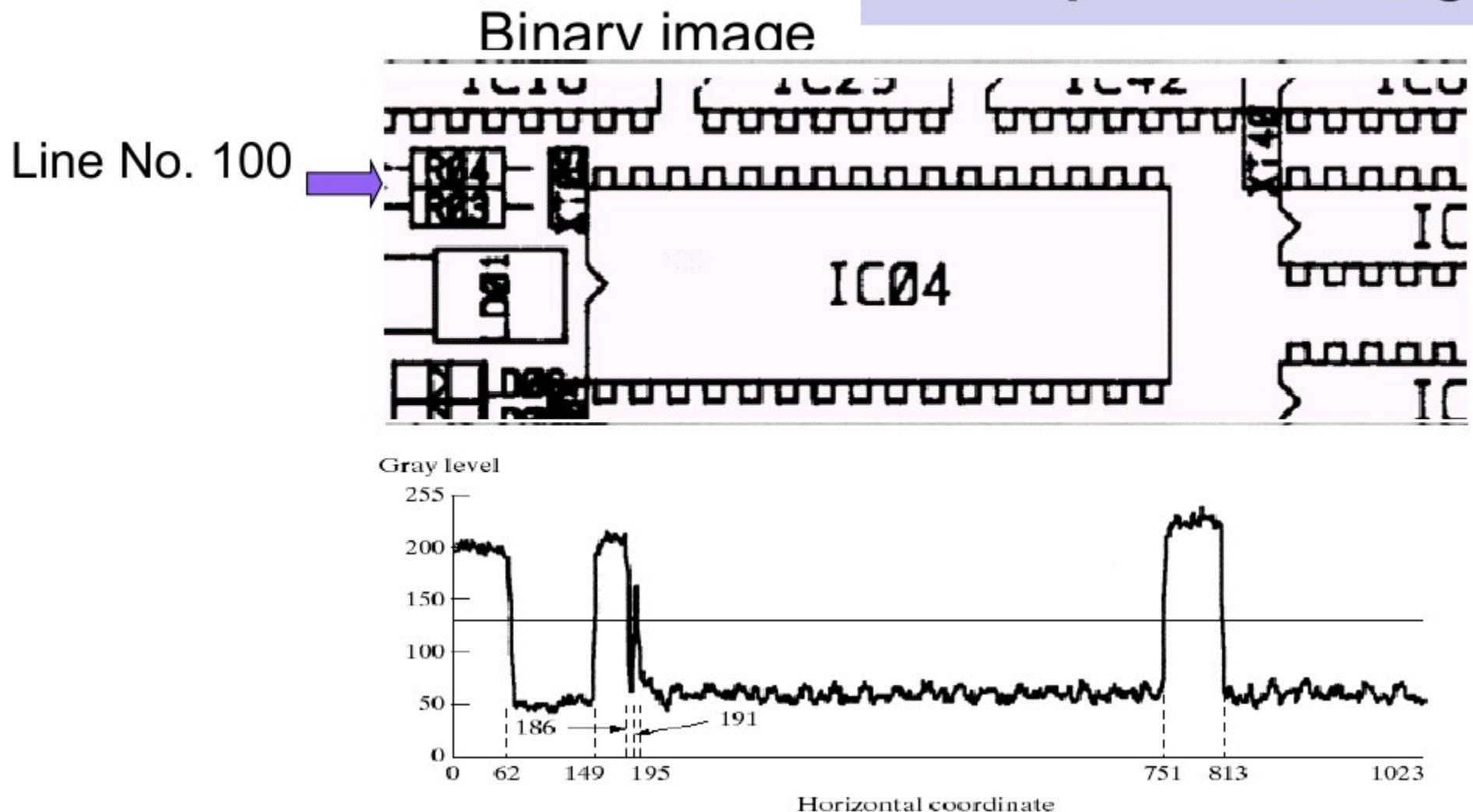
A circuit **binary image** is required to transmit. The image is coded with run length code. Calculate the number of bits required with and without coding. The size of image is 343x1024 and pixels are raster scanned. **The maximum interpixel redundant line run is given below in line 100.** Find out the compression if the total numbers of runs are 12166.

Line No. 100



Line 100: (1,63) (0,87) (1,37) (0,5) (1,4) (0,556) (1,62) (0,210)

## Example: RL coding



Line 100: (1,63) (0,87) (1,37) (0,5) (1,4) (0,556) (1,62) (0,210)

## Example: RL coding

Binary image size: 343 x 1024

Without coding -  $343 \times 1024 \times 1 = 351232$  bits

Coded by run length coding--

Total runs – 12166 and each run represent by 11 bits

**Total bits used =  $12166 \times 11 = 133826$  bits**

## Example: RL coding

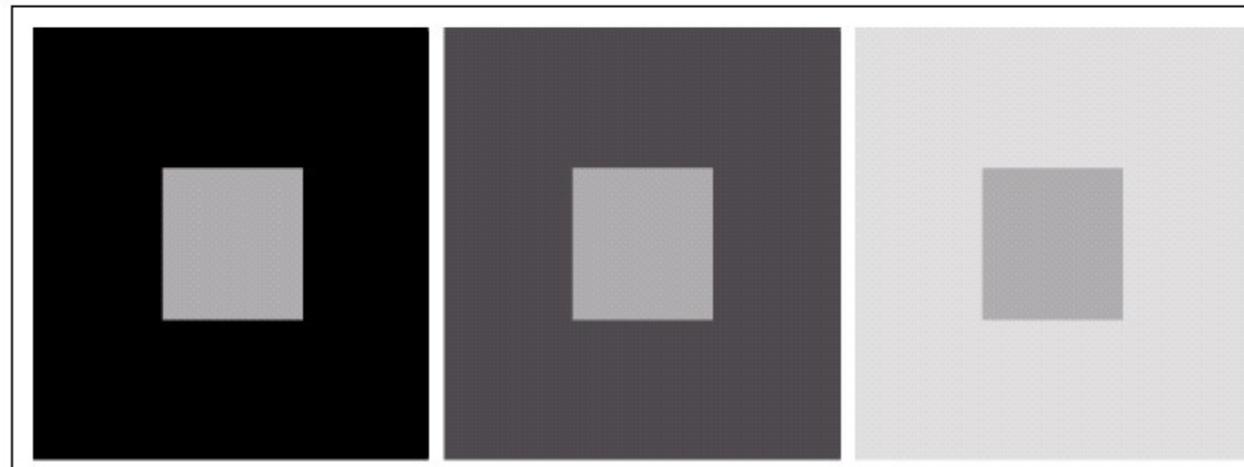
Binary image without coding = 351232 bits

Coded by run length coding = 133826 bits

$$C_R = \frac{n_1}{n_2} = 2.63$$

$$R_D = 1 - \frac{1}{C_R} = 0.62$$

# Psycho-visual Redundancy



The eye does not response linear to all visual information.

# Improved Gray Scale Quantization

## Algorithm

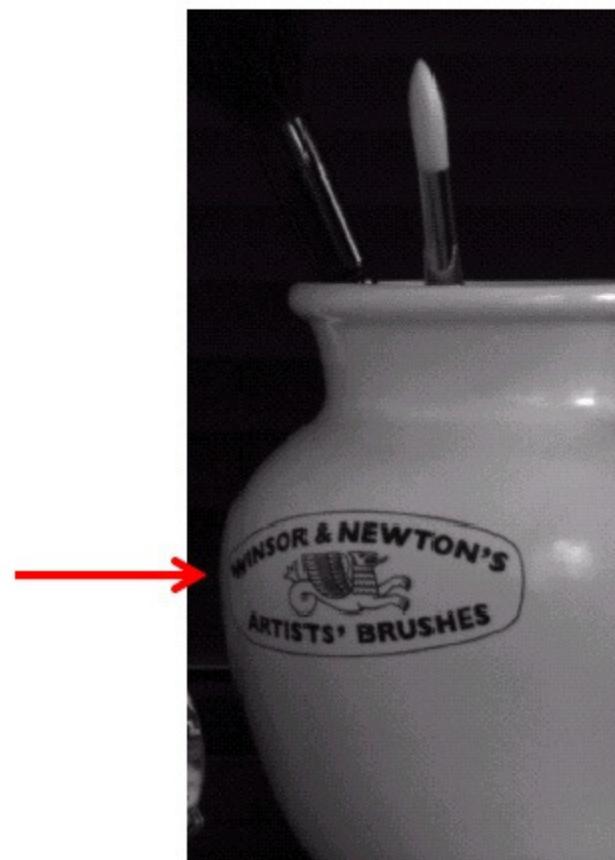
1. Add the least significant 4 bits of the previous value of Sum to the 8-bit current pixel. If the most significant 4 bit of the pixel is 1111 then add 0000 instead. Keep the result in Sum
2. Keep only the most significant 4 bits of Sum for IGS code.

## Example: IGS Quantization

The pixel values of an image is given as --

Pixel
i-1
i
i+1
i+2
i+3

Gray level
N/A
0110 1100
1000 1011
1000 0111
1111 0100



## Example: IGS Quantization

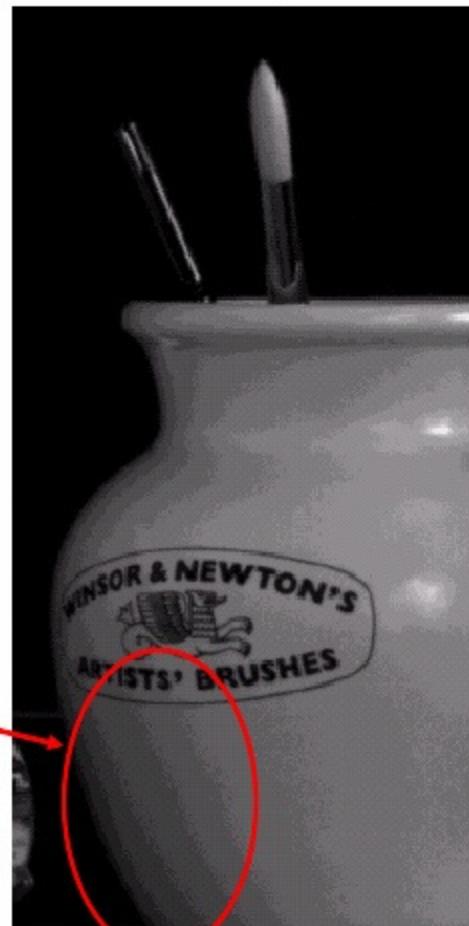
Pixel	Gray level	Sum	IGS Code
i-1	N/A	0000 0000	N/A
i	0110 1100	0110 1100	0110
i+1	1000 1011	1001 0111	1001
i+2	1000 0111	1000 1110	1000
i+3	1111 0100	1111 0100	1111

# Irrelevant Information Psycho-visual Redundancy

8-bit gray scale  
image



4-bit gray scale  
image



4-bit IGS  
image



False  
contours

The eye does not respond linear to all visual information.

## Measuring Image Information

How many bits are actually needed to represent the information in an image?

The entropy  $H(S)$  represents a fundamental limit on the **average number of bits or code length  $L$**  per source symbol, necessary to represent a discrete memoryless source (DMS).



This measure provides us with a theoretical minimum for the average number of bits per pixel that could be used to code the image.

An information source cannot be represented by a source code whose average length is less than the source entropy

## **Shannon's First Theorem: Noiseless Coding**

According to Shannon's first theorem "noiseless coding" [1948]:

**Information source is defined in terms of probability distribution.**

Entropy can also be used as a metric for judging the success of a compression method, as it is theoretically optimal

# Measuring Image Information

A random event 'E' that occurs with probability  $p(E)$  is said to contain information  $I(E)$

$$I(E) = \log_2 \frac{1}{p(E)} = -\log_2 p(E) \text{ bits}$$

$I(E)$  : is called the ***self-information of E***

- If  $p(E)=1$  then  $I(E)=0$  bit
- If  $p(E)=1/2$  then  $I(E)=1$  bit
- If  $p(E)=0$  then  $I(E)= ?$

A predictable signal conveys 'no' information.

**Source alphabet :** A  $\{a_1, a_2, \dots, a_J\}; \sum_{j=1}^J p(a_j) = 1$

**Let probabilities of each**  $\mathbf{z} = [p(a_1), p(a_2), \dots]^T$

**The average information per source output**

$$H(\mathbf{z}) = -\sum_{j=1}^J p(a_j) \log_2 p(a_j)$$

$a_1 = GS_1, a_2 = GS_2, \dots, a_J = GS_J$

## Example: Entropy

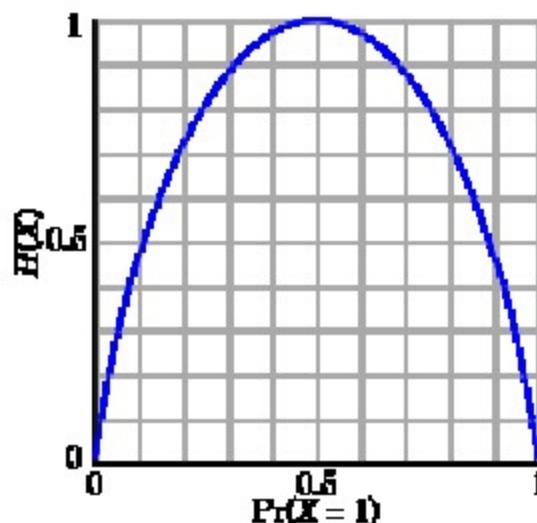
A binary Image has equiprobable black and white pixels.

Find the information content of black and white pixels.

$$\text{Information content } I(E) = -\log_2 p(E) \text{ bits}$$

$$\text{Entropy } H(\mathbf{z}) = -\sum_{j=1}^J p(a_j) \log_2 p(a_j)$$

- A binary image has black and white region. The probability of white pixel is 0.95. Find the entropy of the image.
  - 0.45 bits/pixel



## Example

Find out information content (entropy) of following 8-bit image-

21	21	95	95	169	169	243	243
21	21	95	95	169	169	243	243
21	21	95	95	169	169	243	243
21	21	95	95	169	169	243	243

$$H = -\sum_j p(a_j) \log_2(p(a_j))$$

$$H = 2 \text{ bits/pixel}$$

## Example

Find out the entropy of following 8-bit image-

$r_k$	$p_r(r_k)$
$r_{87} = 87$	0.25
$r_{128} = 128$	0.47
$r_{186} = 186$	0.25
$r_{255} = 255$	0.03

$$H = -\sum_j p(a_j) \log_2 p(a_j)$$
$$= 1.6614 \text{ bits / pixel}$$

## Example 8.10

Find out information content (entropy) of following 8-bit image-

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

$$H = -\sum_j p(a_j) \log(p(a_j))$$

$$H = 1.81 \text{ bits/pixel}$$

The highly skewed nature of the probabilities ... says that once a pixel takes on a particular gray shades (black or white), it is highly likely that the following pixels will be of the same shades. So, rather than code the shades of each pixel separately, we can simply code the run lengths of each shades. For example, if we had 190 white pixels followed by 30 black pixels, followed by another 210 white pixels, instead of coding the 430 pixels individually, we would code the blocked sequence  
190,30,210.

## Example 8.11

Find out information content (entropy) of following after differential coding scheme.

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

$$H = 1.41 \text{ bits/pixel}$$

## Fidelity Criteria

How good is the compression algorithm??

- Objective Fidelity Criterion
  - RMSE, PSNR

- Subjective Fidelity Criterion
  - Human Rating

## Objective Fidelity Criteria

**Original image**  $f(x, y)$

**Decompressed image**  $\hat{f}(x, y)$

**Error is defined in Compression :**  $e(x, y) = \hat{f}(x, y) - f(x, y)$

**Total error between two images:**

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]$$

# Objective Fidelity Criteria

## Root-mean-square error (RMSE)

$$e_{\text{rms}} = \left[ \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2 \right]^{1/2}$$

## Mean square signal-to-noise ratio:

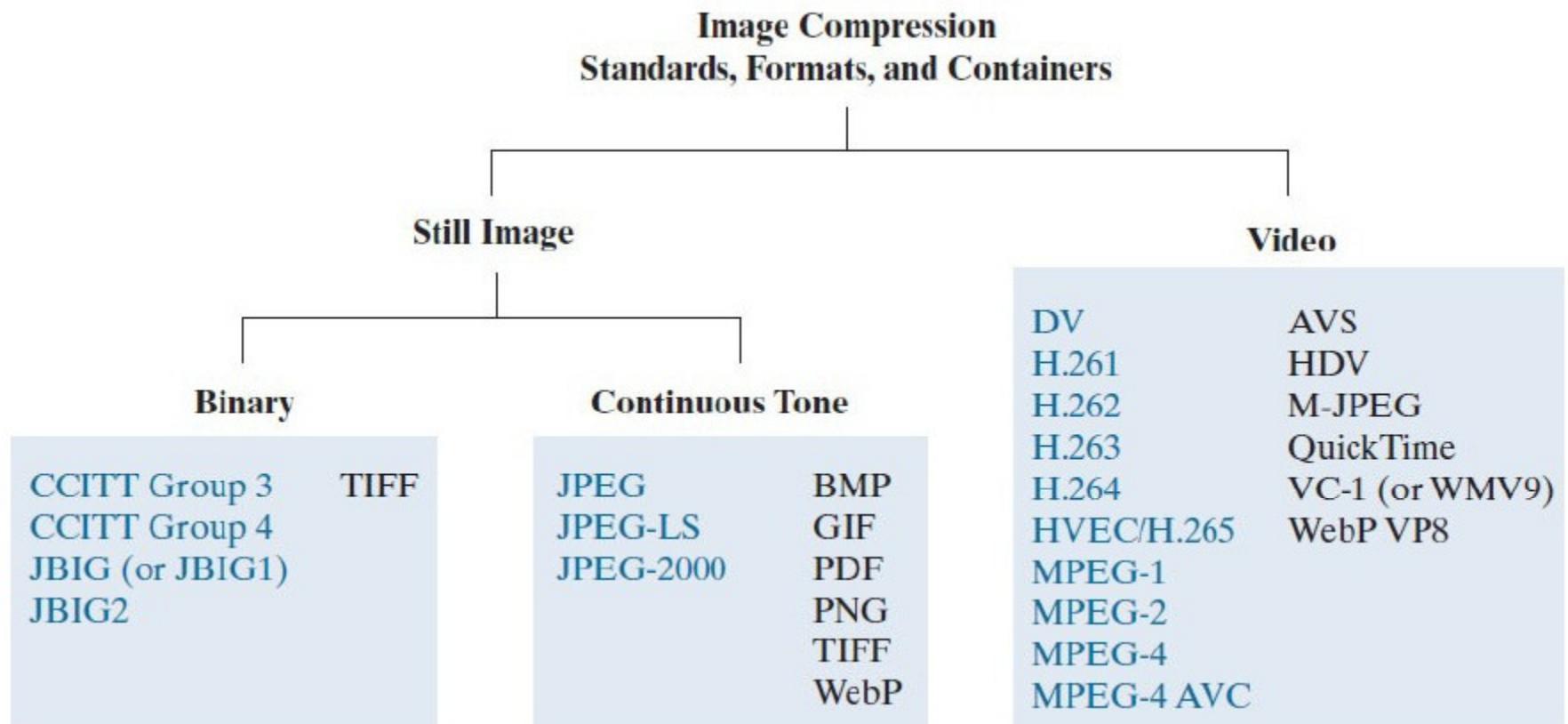
$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2}$$

# Subjective Fidelity Criterion

## Human Expert Rating

<b>Value</b>	<b>Rating</b>	<b>Description</b>
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

# Image Compression Standards



**FIGURE 8.6** Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in blue; all others are in black.

# Basic Compression Methods

Name	Organization	Description
<i>Bi-Level Still Images</i>		
CCITT Group 3	ITU-T	Designed as a facsimile (FAX) method for transmitting binary documents over telephone lines. Supports 1-D and 2-D run-length [8.6] and Huffman [8.2] coding.
CCITT Group 4	ITU-T	A simplified and streamlined version of the CCITT Group 3 standard supporting 2-D run-length coding only.
JBIG or JBIG1	ISO/IEC/ITU-T	A <i>Joint Bi-level Image Experts Group</i> standard for progressive, lossless compression of bi-level images. Continuous-tone images of up to 6 bits/pixel can be coded on a bit-plane basis [8.8]. Context-sensitive arithmetic coding [8.4] is used and an initial low-resolution version of the image can be gradually enhanced with additional compressed data.
JBIG2	ISO/IEC/ITU-T	A follow-on to JBIG1 for bi-level images in desktop, Internet, and FAX applications. The compression method used is content based, with dictionary-based methods [8.7] for text and halftone regions, and Huffman [8.2] or arithmetic coding [8.4] for other image content. It can be lossy or lossless.
<i>Continuous-Tone Still Images</i>		
JPEG	ISO/IEC/ITU-T	A <i>Joint Photographic Experts Group</i> standard for images of photographic quality. Its lossy baseline coding system (most commonly implemented) uses quantized discrete cosine transforms (DCT) on image blocks [8.9], Huffman [8.2], and run-length [8.6] coding. It is one of the most popular methods for compressing images on the Internet.
JPEG-LS	ISO/IEC/ITU-T	A lossless to near-lossless standard for continuous-tone images based on adaptive prediction [8.10], context modeling [8.4], and Golomb coding [8.3].
JPEG-2000	ISO/IEC/ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.4] and quantized discrete wavelet transforms (DWT) [8.11] are used. The compression can be lossy or lossless.

**Table 8.3: Internationally sanctioned image compression standards.**

# Basic Compression Methods

Name	Organization	Description
DV	IEC	<i>Digital Video.</i> A video standard tailored to home and semiprofessional video production applications and equipment, such as electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.9] similar to JPEG.
H.261	ITU-T	A two-way videoconferencing standard for ISDN ( <i>integrated services digital network</i> ) lines. It supports non-interlaced $352 \times 288$ and $176 \times 144$ resolution images, called CIF ( <i>Common Intermediate Format</i> ) and QCIF ( <i>Quarter CIF</i> ), respectively. A DCT-based compression approach [8.9] similar to JPEG is used, with frame-to-frame prediction differencing [8.10] to reduce temporal redundancy. A block-based technique is used to compensate for motion between frames.
H.262	ITU-T	See MPEG-2 below.
H.263	ITU-T	An enhanced version of H.261 designed for ordinary telephone modems (i.e., 28.8 Kbps) with additional resolutions: SQCIF ( <i>Sub-Quarter CIF</i> $128 \times 96$ ), 4CIF ( $704 \times 576$ ) and 16CIF ( $1408 \times 512$ ).
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, streaming, and television. It supports prediction differences within frames [8.10], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.4].
H.265 MPEG-H HEVC	ISO/IEC	<i>High Efficiency Video Coding</i> (HEVC). An extension of H.264 that includes support for macroblock sizes up to $64 \times 64$ and additional intraframe prediction modes, both useful in 4K video applications.
MPEG-1	ISO/IEC	A <i>Motion Pictures Expert Group</i> standard for CD-ROM applications with non-interlaced video at up to 1.5 Mb/s. It is similar to H.261 but frame predictions can be based on the previous frame, next frame, or an interpolation of both. It is supported by almost all computers and DVD players.
MPEG-2	ISO/IEC	An extension of MPEG-1 designed for DVDs with transfer rates at up to 15 Mb/s. Supports interlaced video and HDTV. It is the most successful video standard to date.
MPEG-4	ISO/IEC	An extension of MPEG-2 that supports variable block sizes and prediction differencing [8.10] within frames.
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 <i>Advanced Video Coding</i> (AVC). Identical to H.264.

**Table 8.4: Internationally sanctioned Video compression standards.**

# Huffman Codes

- Developed by David Huffman as part of a class assignment in 1952; the class was the first ever in the area of information theory and was taught by Robert Fano at MIT.
- It is systematic method to find codes of source events
- The probability model of source should be known
  - **Codes need not to be unique.**

Application examples: JPEG, MPEG, MP3

# Huffman Code's Procedure

1. List the source symbols in descending order
2. Merge the two least probable outputs into a composite code whose probability is the sum of the two corresponding probabilities.
3. If the number of remaining output is 2, then go to next step, otherwise go to step 2.

## Example

Find out the Huffman code for given source X

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$$

With probabilities respectively as

$$p = \{0.1, 0.4, 0.06, 0.1, 0.04, 0.3\}$$

# Example

Find out the Huffman code for given source X

$a_1$	0.1	0.04
$a_2$	0.4	0.06
$a_3$	0.06	0.1
$a_4$	0.1	0.1
$a_5$	0.04	0.3
$a_6$	0.3	0.4

## Procedure: Huffman Coding

Step 1: Source events in descending order–

Original source		Source reduction			
Symbol	Probability	1	2	3	4
$a_2$	0.4	0.4	0.4	0.4	0.6
$a_6$	0.3	0.3	0.3	0.3	0.4
$a_1$	0.1	0.1	0.2	0.3	
$a_4$	0.1	0.1	0.1		
$a_3$	0.06	0.1			
$a_5$	0.04				

# Example

Huffman code assignment procedure.

Original source			Source reduction									
Symbol	Probability	Code	1	2	3	4						
$a_2$	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0		
$a_6$	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1		
$a_1$	0.1	011	0.1	011	0.2	010	0.3	01				
$a_4$	0.1	0100	0.1	0100	0.1	011						
$a_3$	0.06	01010	0.1	0101								
$a_5$	0.04	01011										

## Example 1.12

Find out the Huffman code for given source X

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$$

With probabilities respectively as

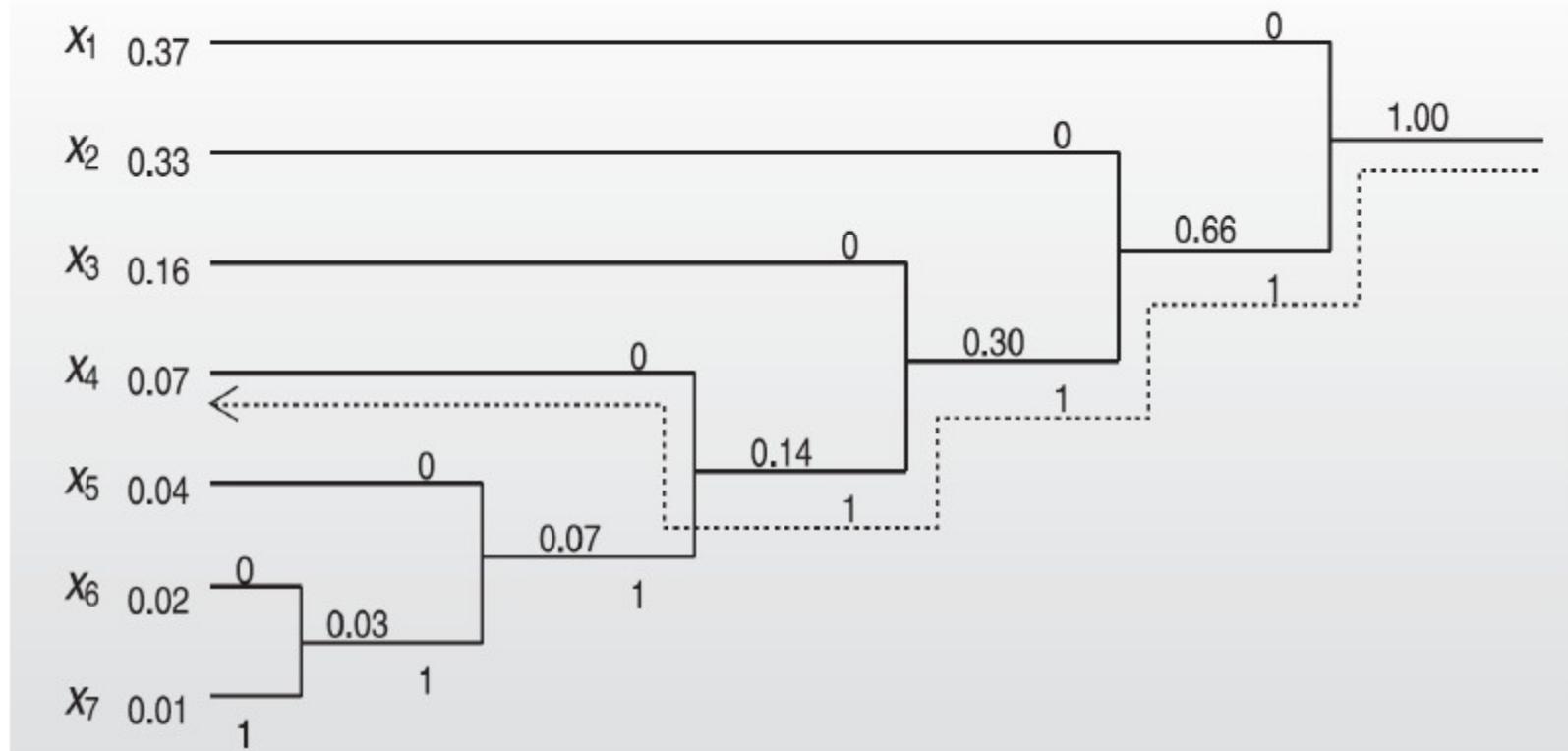
$$p = \{0.37, 0.33, 0.16, 0.07, 0.04, 0.02, 0.01\}$$

# Example

Findout the Huffman code for given source X

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$$

$$p = \{0.37, 0.33, 0.16, 0.07, 0.04, 0.02, 0.01\}$$



# Huffman Codes

Findout the Huffman code for given source X

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$$

$$p = \{0.37, 0.33, 0.16, 0.07, 0.04, 0.02, 0.01\}$$

Symbol	Probability	Self information	codeword	Code length
$x_1$	0.37	1.43	0	1
$x_2$	0.33	1.5995	10	2
$x_3$	0.16	2.6439	110	3
$x_4$	0.07	3.8365	1110	4
$x_5$	0.04	4.6439	11110	5
$x_6$	0.02	5.6439	111110	6
$x_7$	0.01	6.6439	111111	6

# Huffman Codes

- Codes

$$H(X) = 2.115 \text{ bits/symbol}$$

$$L_{avg} = \sum_i p_i l_i$$

$$L_{avg} = 2.17 \text{ bits/symbol}$$

$$\eta = H/L = 97.47 \%$$

# Extended Huffman Coding

$\alpha_i$	Source Symbols	$P(\alpha_i)$ Eq. (8.3-14)	$I(\alpha_i)$ Eq. (8.3-1)	$I(\alpha_i)$ Eq. (8.3-16)	Code Word	Code Length
<i>First Extension</i>						
$\alpha_1$	$a_1$	2/3	0.59	1	0	1
$\alpha_2$	$a_2$	1/3	1.58	2	1	1
<i>Second Extension</i>						
$\alpha_1$	$a_1 a_1$	4/9	1.17	2	0	1
$\alpha_2$	$a_1 a_2$	2/9	2.17	3	10	2
$\alpha_3$	$a_2 a_1$	2/9	2.17	3	110	3
$\alpha_4$	$a_2 a_2$	1/9	3.17	4	111	3

First extension

$$H = 0.918$$

$$L_{avg} = 1$$

Second extension

$$H = 1.83$$

$$L_{avg} = 1.89$$

## Extended Huffman Coding

First extension

$$H = 0.918$$

$$L_{avg} = 1$$

Efficiency

$$\eta_1 = \frac{0.918}{1} = 0.918$$

Second extension

$$H = 1.83$$

$$L_{avg} = 1.89$$

Efficiency

$$\eta_2 = \frac{1.83}{1.89} = 0.97$$

## Drawbacks of Huffman code

- Replacing an input pixel intensity with a code word
- Need a probability distribution of pixel intensities
- Need to store the code word table
- Minimum code word length is 1 bit
- The coding is not efficient if the probability is skewed.
- Efficiency is improved by block code.
- Hard to adapt to changing statistics

**Concept:** The entire sequences of pixel intensities are assigned a single arithmetic code word in the form of a number in an interval of real number between 0 and 1.

- It is string or sequence coding.

**Example:** Sequence of intensities  $\{a_1 ; a_2 ; a_3 ; a_3 ; a_4\}$

$a_1 \ a_1 \ a_4 \ a_4 \ a_4 \dots$

# Arithmetic Coding

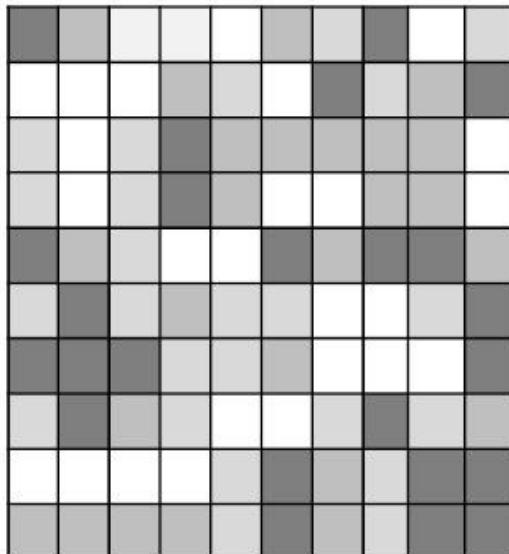
- Let us assume that all pixels are concatenations of gray shades from range of gray values.
- The function maps random intensity variables and sequences of random intensity variables into cumulative distribution function.
- This is done by using a set of interval ranges determined by the probability.
- A unique identifier or tag is generated for the intensity sequence to be coded.
- The tag corresponding to binary fraction.

# Arithmetic Coding

- The possible tag are the numbers in unit interval [0 1)
- The numbers in unit interval is infinite.
- The number in the unit interval [0, 1) is unique. In order to distinguish a sequence of intensities from another sequence of intensities, it need a unique identifier.
- In order to do this we need a function that will map sequences of intensities into the unit interval.

# Arithmetic Coding

For the given 10x10 image, the probability of intensities are given below:



Intensities	Probability
a <sub>1</sub>	0.2
a <sub>2</sub>	0.2
a <sub>3</sub>	0.4
a <sub>4</sub>	0.2



“a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> a<sub>3</sub> a<sub>4</sub> a<sub>2</sub> a<sub>3</sub> a<sub>1</sub> a<sub>4</sub> a<sub>2</sub>”

Find the Arithmetic coding sequence of intensities –  
“a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> a<sub>3</sub> a<sub>4</sub> .....”

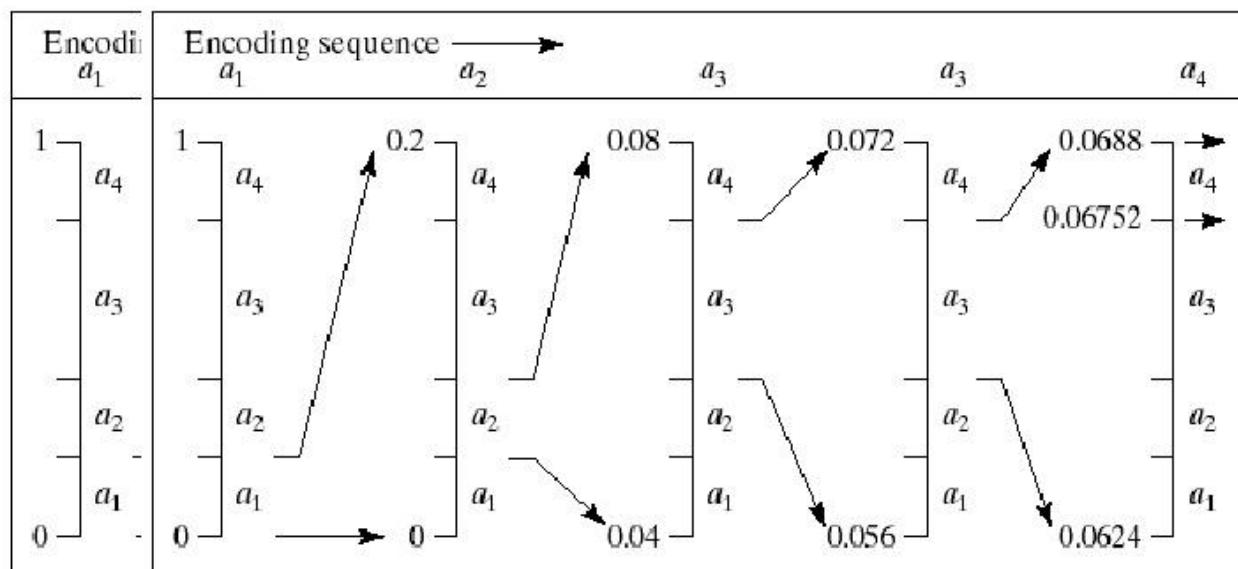


# Arithmetic Coding

Intensities      Probabilities      Sub-interval

	$a_1$	0.2	[0.0, 0.2)
	$a_2$	0.2	[0.2, 0.4)
	$a_3$	0.4	[0.4, 0.8)
	$a_4$	0.2	[0.8, 1.0)

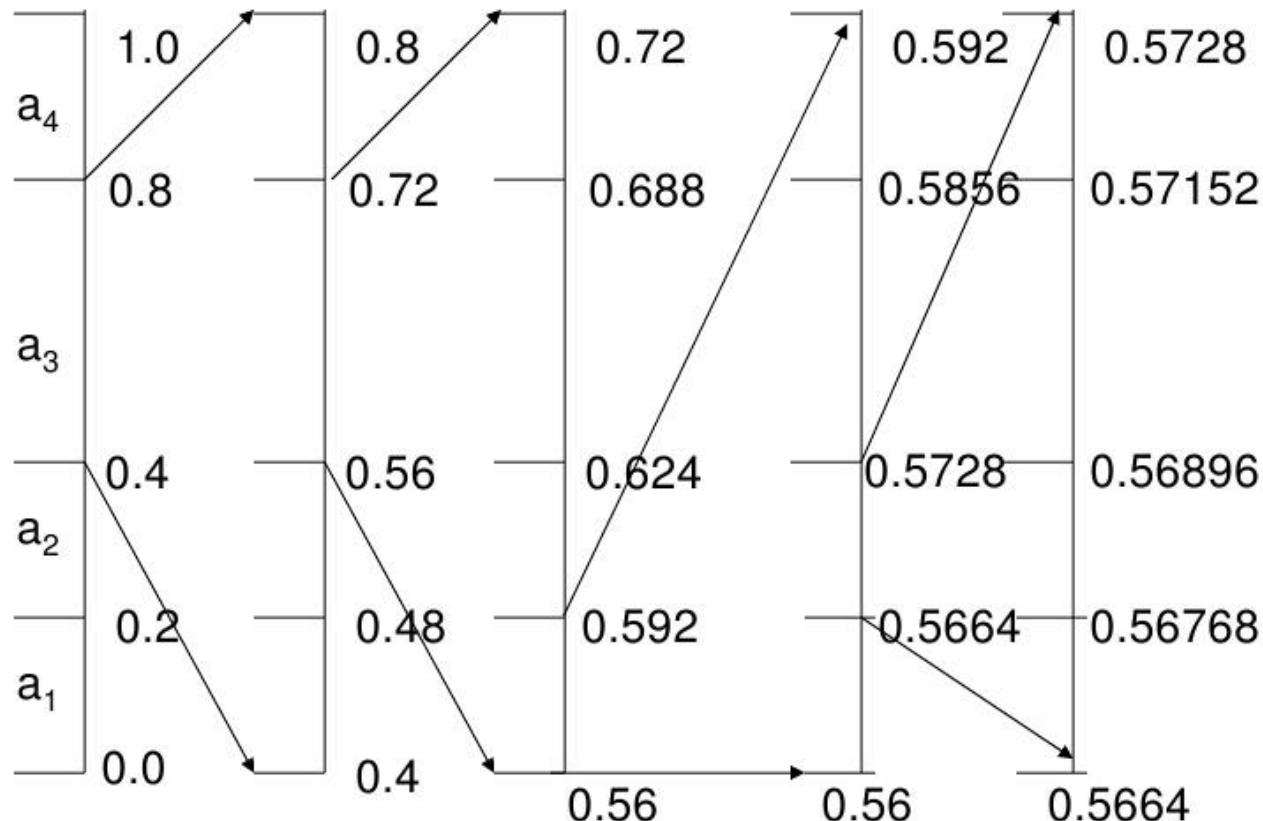
Sequence  $a_1 a_2 a_3 a_3 a_4 \dots \dots \dots$



[0.0688, 0.06752)

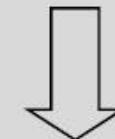
# Arithmetic Coding

Decode the sequence of 0.572.



Inten	Prob
$a_1$	0.2
$a_2$	0.2
$a_3$	0.4
$a_4$	0.2

Decode 0.572

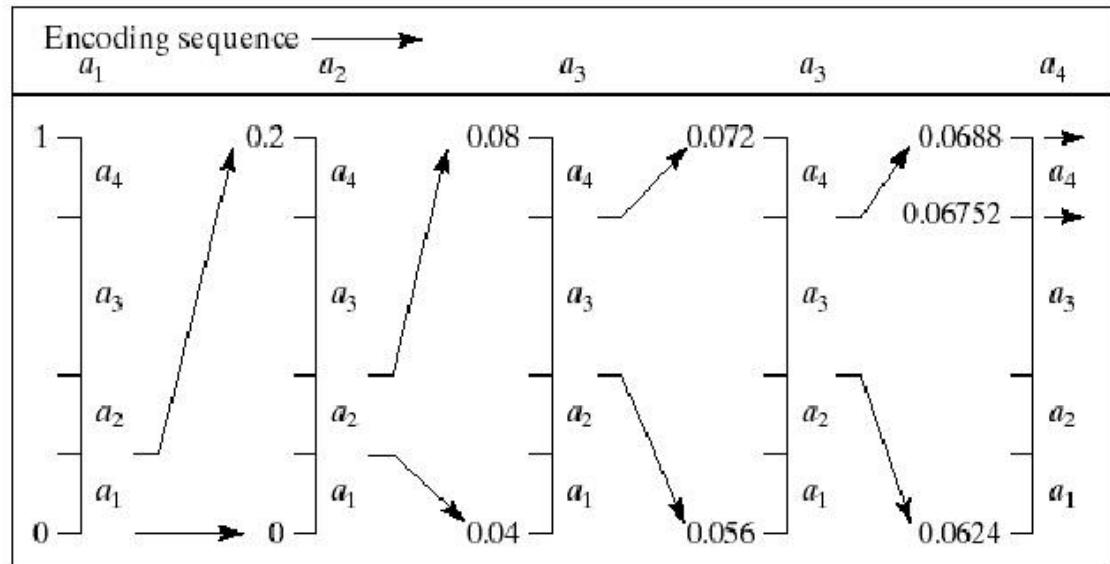


$a_3 \ a_3 \ a_1 \ a_2 \ a_4$

## Remarks: Arithmetic Coding

Input sequence: “ $a_1 a_2 a_3 a_3 a_4 \dots$ ”

- Range 1
- Range 0.2
- Range 0.04
- Range 0.016
- Range 0.0064



- Termination: Encode the lower end to signal.
- Difficulties: Shrinking of interval requires very high precision for long sequence. No output is generated until the entire sequence has been processed.

## Example

Construct a code for pixel intensities  $X = \{A, B, C\}$  with probability  $p = \{0.5, 0.25, 0.25\}$ . A sequence is generated by gray shades X as

B A C A .....

Find the Code for given gray shades sequence

Using the mapping by cumulative distribution function  $F_x(1) = 0.5$ ,  $F_x(2) = 0.75$ , and  $F_x(3) = 1$ .

# Example

Construct a code for pixel intensities  $X = \{A, B, C\}$  with probability  $p = \{0.5, 0.25, 0.25\}$ .

Using the mapping by cumulative distribution function  $F_x(1) = 0.5$ ,  $F_x(2) = 0.75$ , and  $F_x(3) = 1$ . Then find the code of sequence “B A C A.....”

## Arithmetic Coding:

- Replace the entire input with a single floating-point number
- Adaptive coding is very easy
- No need to keep and send code word table
- Fractional code word length

# The Lempel-Ziv Algorithm

- HUFFMAN'S AND ARITHMATIC coding techniques assume a source that generates a **sequence of independent symbols**
- The source coding can compressed more if we used correlation or **structural information** of the data.
- The LZ technique used **structural information** of the data.
- The technique is based on code book build a list of commonly occurring patterns and encode these patterns by transmitting their index in the book.

## Lempel-Ziv-Welch coding

It does not required source statistics.

Assign fixed length code words to variable length sequences of source symbols.

39	39	126	126	24	24	150	150
39	39	126	126	24	24	150	150
39	39	126	126	24	24	150	150
39	39	126	126	24	24	150	150

39, 39, 126, 126, 24, 24, 150, 150, .....

## LZW Coding

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Currently Recognized Sequence	Pixel Being Processed
	39
39	39
39	126
126	126
126	39
39	39
39-39	126
126	126
126-126	39
39	39
39-39	126
39-39-126	126
126	39
126-39	39
39	126
39-126	126
126	

Conversion from 2D to 1D information-

(39, 39, 126, 126, 39, 39, 126, 126, 39, 39, 126, 126, 39, 39, 126, 126,.....)

## LZW Algorithm

1. Initialize a dictionary by all possible gray values (0-255)
2. Input current pixel
3. If the current pixel combined with previous pixels  
form one of existing dictionary entries

Then

- 2.1 Move to the next pixel and repeat Step 1

Else

- 2.2 Output the dictionary location of the currently recognized sequence (which is not include the current pixel)

- 2.3 Create a new dictionary entry by appending the currently recognized sequence in 2.2 with the current pixel

- 2.4 Move to the next pixel and repeat Step 1

## Example: LZW Coding

Find the LZW code of following 4x4 image by raster scan. Assume the size of dictionary is 9-bits.

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

## Example: LZW Coding

Conversion from 2D to 1D information-

39, 39, 126, 126, 39, 39, 126, 126, 39, 39, 126, 126,  
39, 39, 126, 126,.....

Parsing the patterns to build dictionary –

(39, 39), (39, 126), (126, 126), (126, 39), (39, 39,  
126), (126, 126, 39), (39, 39, 126, 126),.....

## Example: LZW Coding

Parsing the patterns to build dictionary

(39, 39), (39, 126),  
(126, 126), (126, 39),  
(39, 39, 126), (126,  
126, 39), (39, 39, 126,  
126),.....

Location	Entry
0	0
1	1
...	...
...	...
255	255
256	39-39
257	39-126
258	126-126
259	126-39
260	39-39-126
261	126-126-39
262	39-39-126-126

## Dictionary

Location	Entry	Input pixel
0	0	
1	1	39
...	...	39
255	255	126
256	39-39	126
257	39-126	39
258	126-126	39
259	126-39	126
260	39-39-126	126
261	126-126-39	39
262	39-39-126-126	39 126 126

The output of encoder:

Currently Recognized Sequence	Pixel Being Processed	Encode Output
	39	39
39	39	39
39	126	39
126	126	126
126	39	126
39	39	
39-39	126	256
126	126	
126-126	39	258
39	39	
39-39	126	
39-39-126	126	260
126	39	
126-39	39	259
39	126	
39-126	126	257
126		126

Dictionary	
Location	Entry
0	0
1	1
...	...
...	...
255	255
256	39-39
257	39-126
258	126-126
259	126-39
260	39-39-126
261	126-126-39
262	39-39-126-126

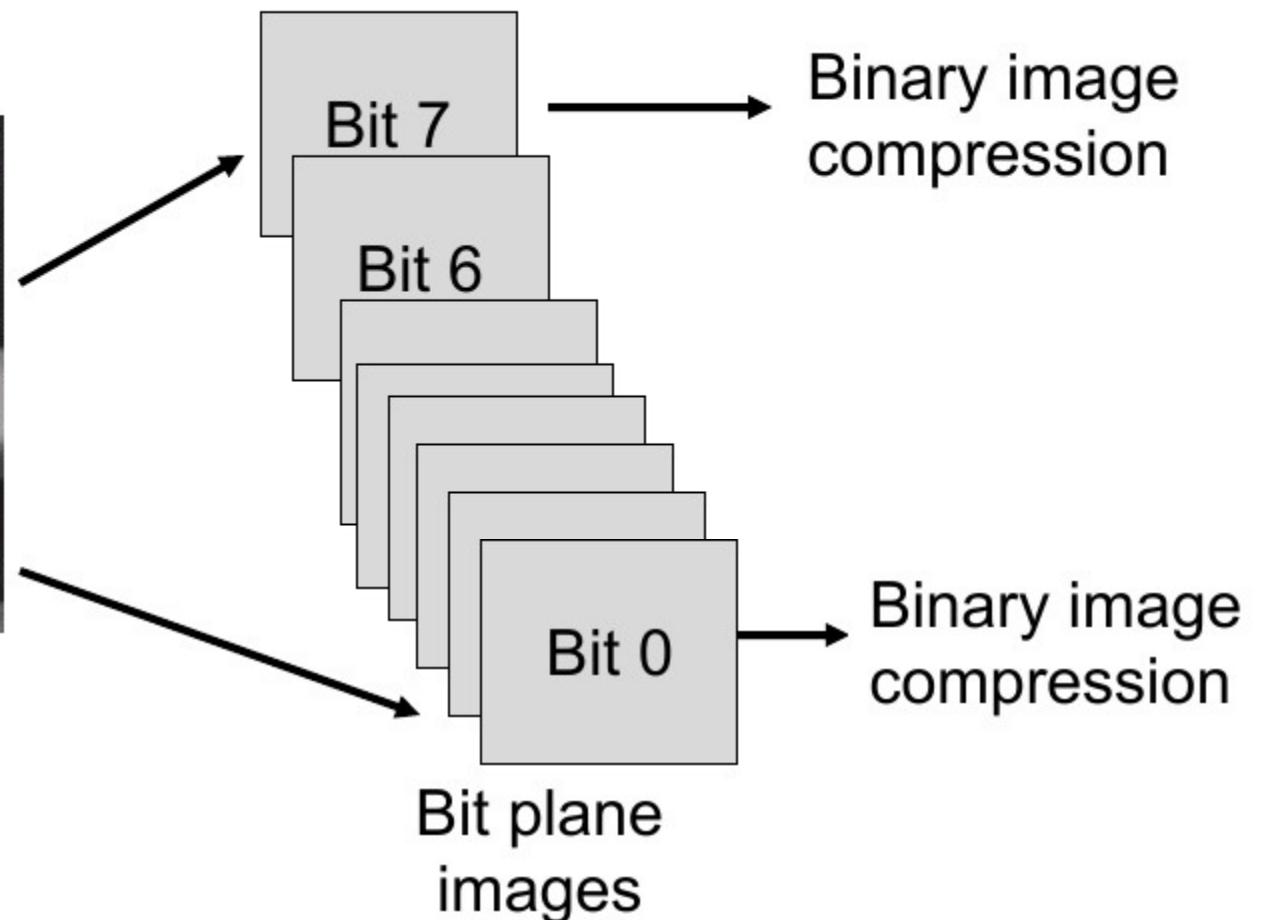
# Bit-Plane Compression

A monochrome image is decomposed into a series of binary images (bit planes), and then they are compressed by a binary compression method.

Original gray-level:  $11001 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

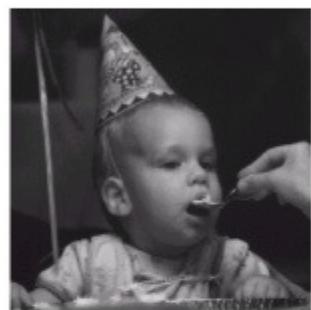


## Bit-Plane Compression



Example of binary image compression: Run length coding

## Bit-Planes



Original gray scale image

Bit 7



Bit 6



Bit 5



Bit 4



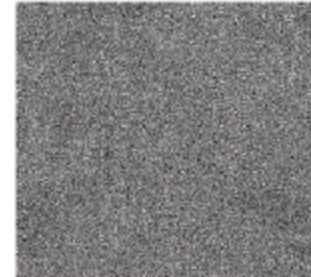
Bit 3



Bit 2



Bit 1



Bit 0

## Gray coded Bit-Planes

Original  
bit  
planes



$$g_i = a_i \otimes a_{i+1}$$

for  $0 \leq i \leq 6$

and

$$g_7 = a_7$$

**There are less transitions in grayed code bit planes.**

**Hence gray coded bit planes are more efficient for coding.**

## Gray coded Bit-Planes

Original  
bit  
planes



$$g_i = a_i \otimes a_{i+1}$$

for  $0 \leq i \leq 6$

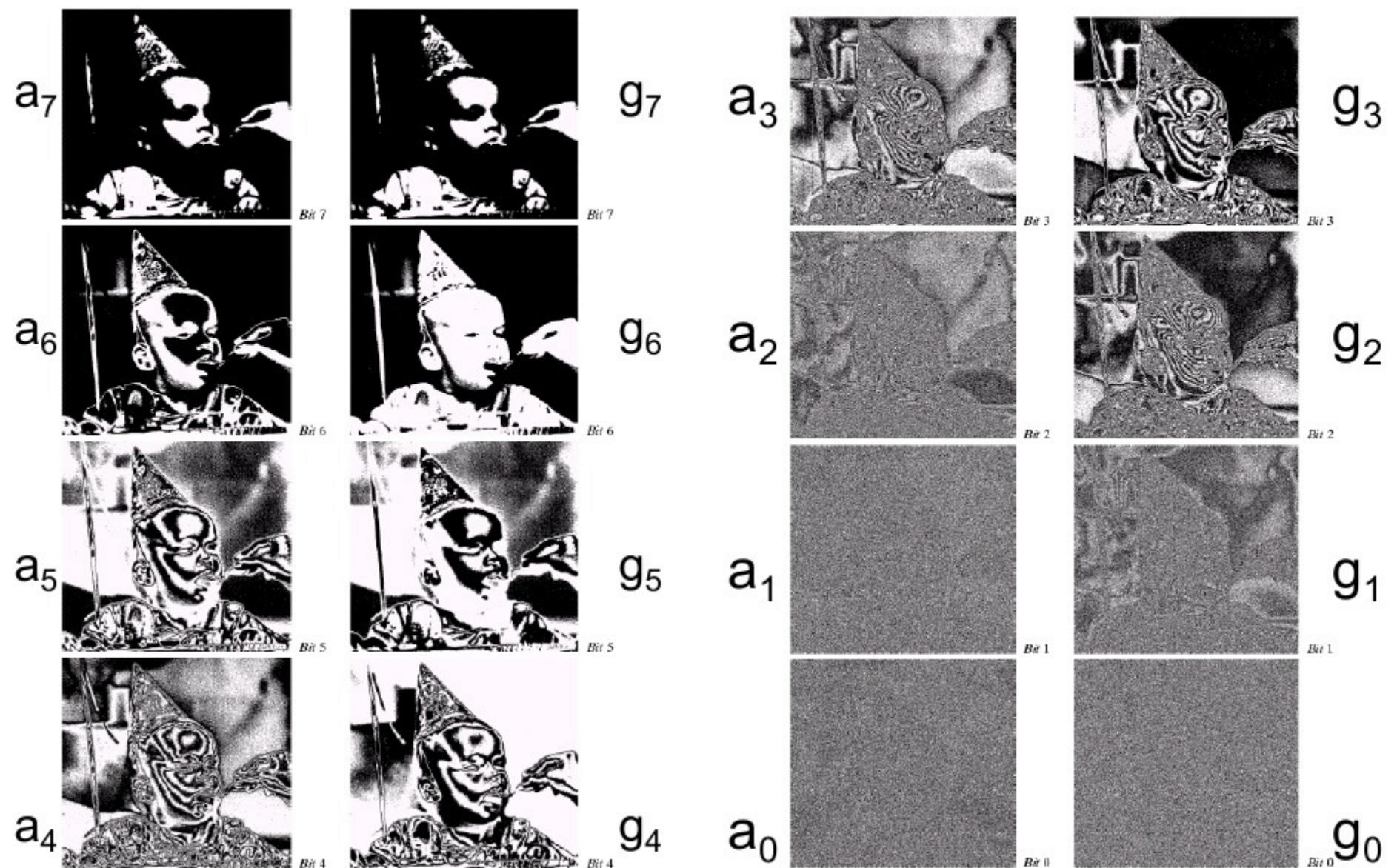
and

$$g_7 = a_7$$

**There are less transitions in grayed code bit planes.**

**Hence gray coded bit planes are more efficient for coding.**

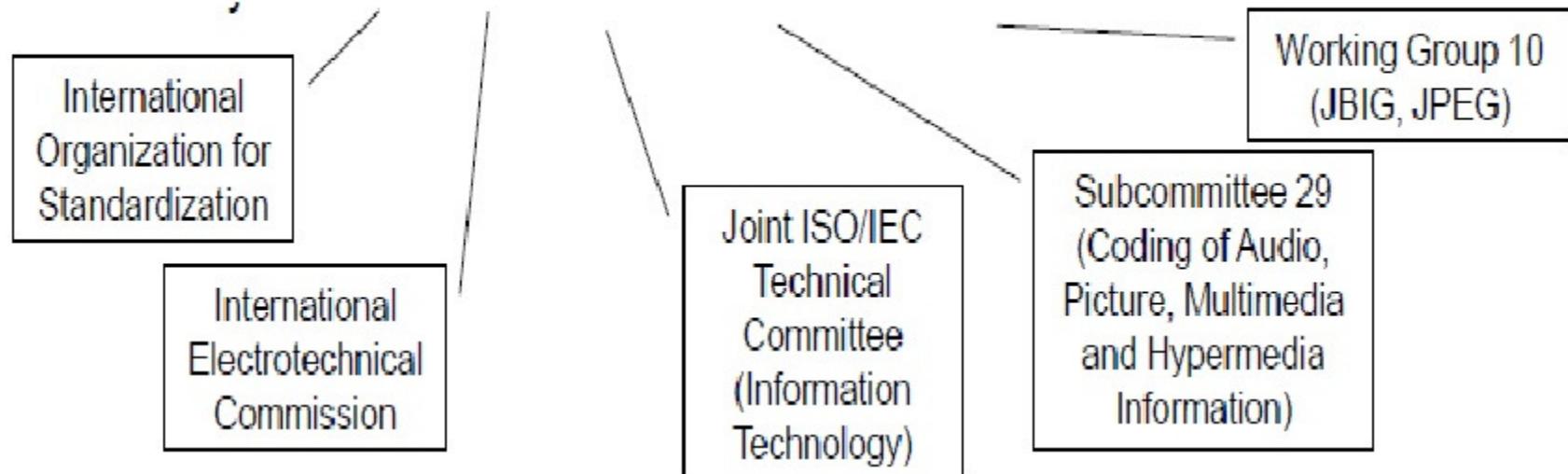
# Gray-coded Bit Planes



# Introduction to JPEG

JPEG: “Joint Photographic Experts Group”

Formally: ISO/IEC JTC1/SC29/WG10



- Joint effort with CCITT (International Telephone and Telegraph Consultative Committee, now ITU-T) Study Group VIII
  - Work commenced in 1986
- International standard ISO/IEC 10918-1 and CCITT Rec. T.81 in 1992
  - Widely used for image exchange, WWW, and digital photography
  - Motion-JPEG is de facto standard for digital video editing

# Introduction to JPEG

Many versions of JPEG standards:

**JPEG old:**

- Based on prediction rules

**JPEG base line:** lossy coding of continuous tone still images

- Based on DCT

**JPEG-LS:** lossless and nearlossless coding of continuous tone still images

- Based on predictive coding and entropy coding

**JPEG 2000:** scalable coding of continuous tone still images (from lossy to lossless)

- Based on wavelet transform

- The JPEG specification consists of several parts, including a specification for both lossless and lossy encoding.
- The lossless compression uses the predictive model
- 29 distinct coding schemes
- It provides seven predictive schemes.
- Three schemes based on one dimension.
- Four are based on two dimension prediction schemes.

- Predictive schemes-

$$1 \quad \hat{I}(i, j) = I(i - 1, j)$$

$$2 \quad \hat{I}(i, j) = I(i, j - 1)$$

$$3 \quad \hat{I}(i, j) = I(i - 1, j - 1)$$

$$4 \quad \hat{I}(i, j) = I(i, j - 1) + I(i - 1, j) - I(i - 1, j - 1)$$

$$5 \quad \hat{I}(i, j) = I(i, j - 1) + (I(i - 1, j) - I(i - 1, j - 1)) / 2$$

$$6 \quad \hat{I}(i, j) = I(i - 1, j) + (I(i, j - 1) - I(i - 1, j - 1)) / 2$$

$$7 \quad \hat{I}(i, j) = (I(i, j - 1) + I(i - 1, j)) / 2$$

- Predictive schemes-

$$1 \quad \hat{I}(i, j) = I(i - 1, j)$$

$$2 \quad \hat{I}(i, j) = I(i, j - 1)$$

$$3 \quad \hat{I}(i, j) = I(i - 1, j - 1)$$

$$4 \quad \hat{I}(i, j) = I(i, j - 1) + I(i - 1, j) - I(i - 1, j - 1)$$

$$5 \quad \hat{I}(i, j) = I(i, j - 1) + (I(i - 1, j) - I(i - 1, j - 1)) / 2.$$

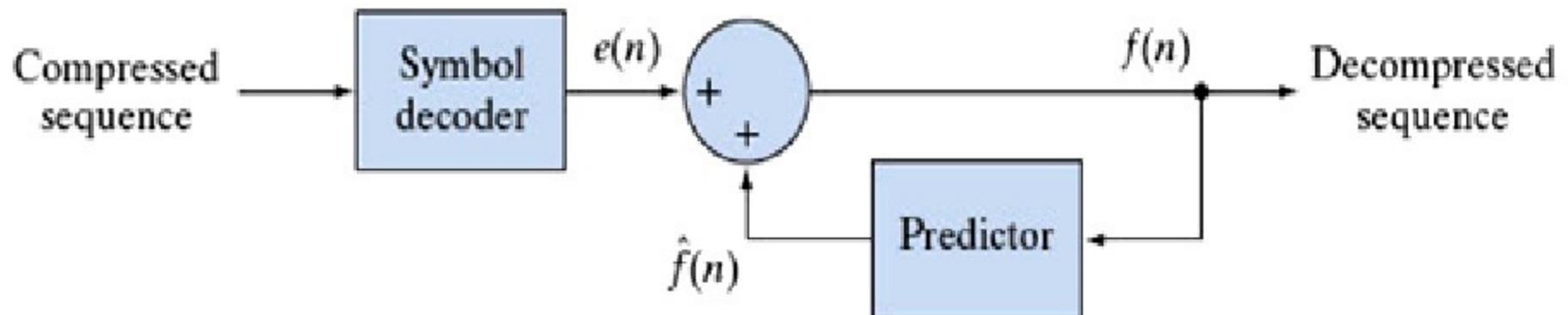
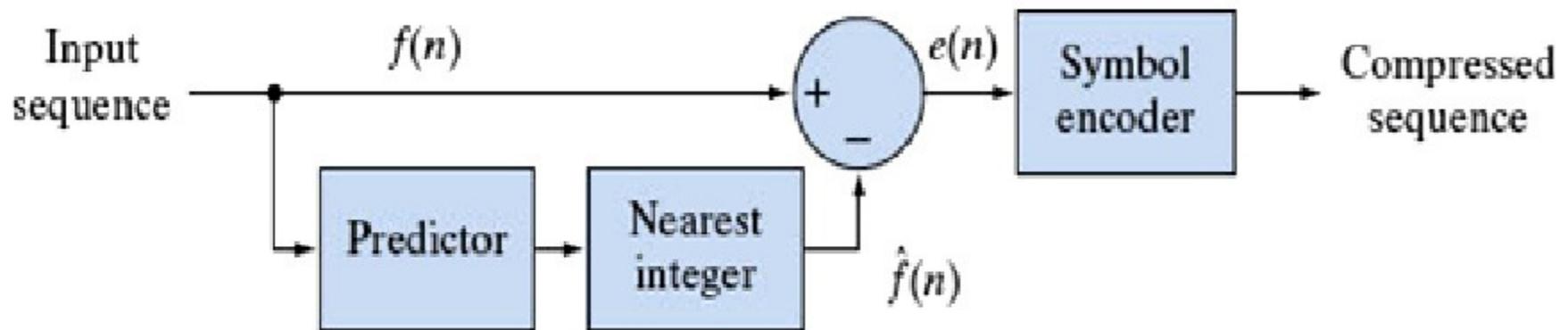
$$6 \quad \hat{I}(i, j) = I(i - 1, j) + (I(i, j - 1) - I(i - 1, j - 1)) / 2$$

$$7 \quad \hat{I}(i, j) = (I(i, j - 1) + I(i - 1, j)) / 2$$

- Three of seven predictor are one dimension
  - In off line process, all predictor are tested
- 3 bits header are add for predictor information
- The residual images are coded by adaptive arithmetic coding

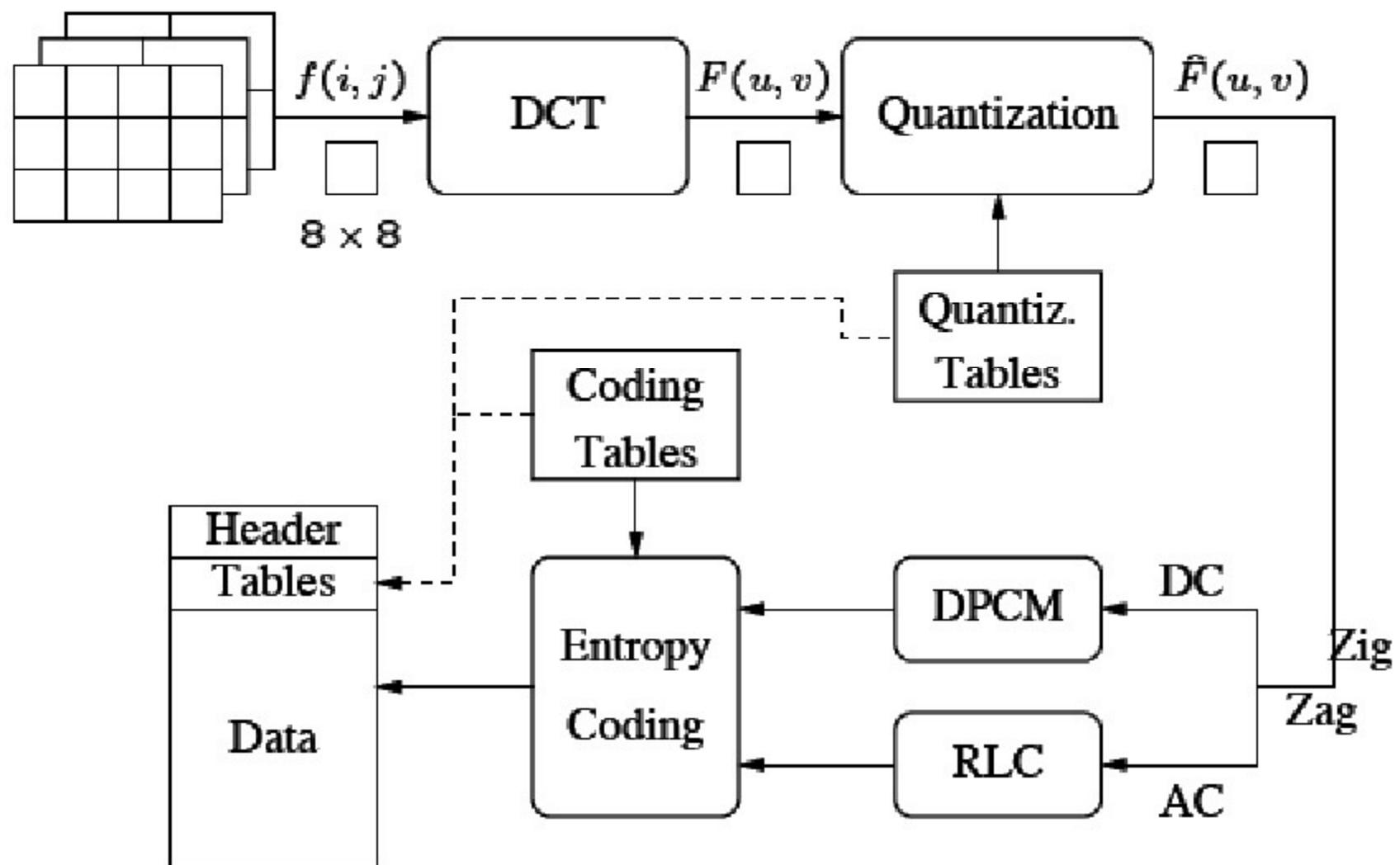
# Predictive Coding

A lossless predictive coding model: (a) encoder; (b) decoder



# Block Transform Coding

## JPEG Coder



# Block Transform Coding

JPEG is effective because of the following three observations

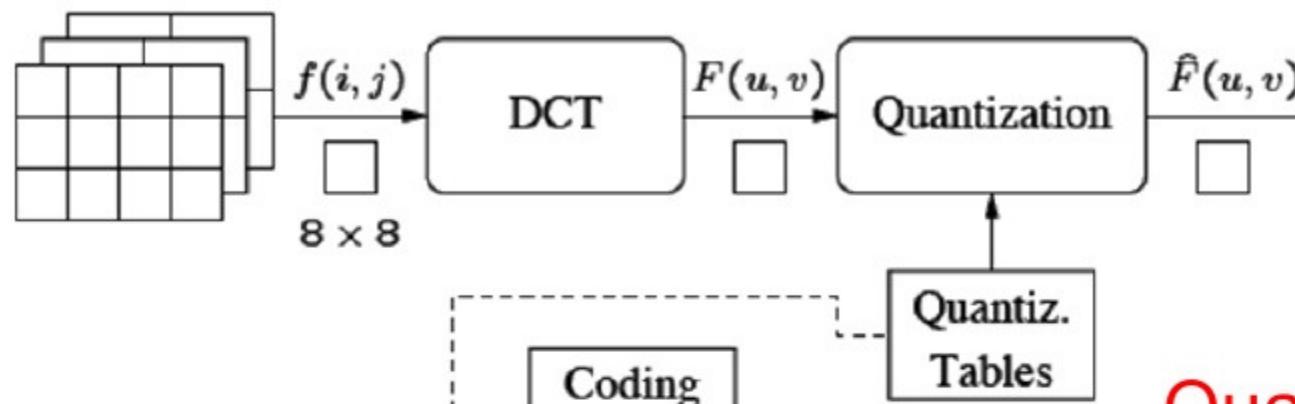
- Image data usually **changes slowly** across an image, especially within an 8x8 block
  - Therefore images contain **much redundancy**
- Experiments indicate that **humans are not very sensitive** to the high frequency data images
  - Therefore we can remove much of this data using transform coding

## Example: 8.17

The following is the **2D DCT coefficients** of 8x8 image , Using JPEG Compression Standard obtain the Final Compressed form of the image. Also obtain the Compression ratio and Redundancy.

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

## Example: 8.17



Quantization Table

$$q_{val} = \text{round} \left( \frac{dct2}{q_{mt}} \right)$$

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

## Example: 8.17

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

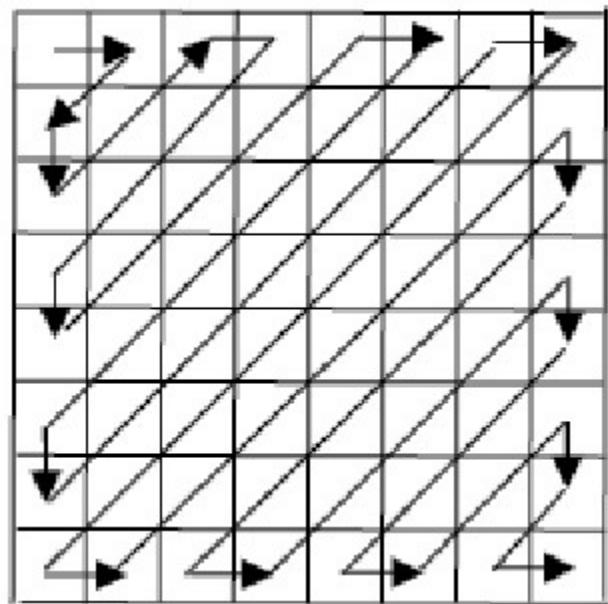
The resulting quantized values

$$q_{val} = \text{round} \left( \frac{dct2}{q_{mt}} \right)$$

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

## Example: 8.17

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



Finally we get –

**[-26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 -1 -1 EOB]**

## Example: 8.17

Finally we get –

**[-26 -31 -3 -2 -62 -41 -41 15 0 200 -1 2 0 0 0 0 0 -1 -1 EOB]**

Let the DC value of the previous 8x8 DCT block is -17

The DPCM is used to code DC coefficients.

$$- 26 - (-17) = - 9$$

From the Coding table it is coming under category 4 of the dc table.

Base code is 101 and length of the code is 7, remaining four bits will be LSBs of -9 which is 0110

Finally it is coded as **1010110**

# JPEG Coefficient coding Category Coding Table

Range	DC Difference Category	AC Category
0	0	N/A
-1, 1	1	1
-3, -2, 2, 3	2	2
-7, ..., -4, 4, ..., 7	3	3
-15, ..., -8, 8, ..., 15	4	4
-31, ..., -16, 16, ..., 31	5	5
-63, ..., -32, 32, ..., 63	6	6
-127, ..., -64, 64, ..., 127	7	7
-255, ..., -128, 128, ..., 255	8	8
-511, ..., -256, 256, ..., 511	9	9
-1023, ..., -512, 512, ..., 1023	A	A
-2047, ..., -1024, 1024, ..., 2047	B	B
-4095, ..., -2048, 2048, ..., 4095	C	C
-8191, ..., -4096, 4096, ..., 8191	D	D
-16383, ..., -8192, 8192, ..., 16383	E	E
-32767, ..., -16384, 16384, ..., 32767	F	N/A

# JPEG default DC code

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

101

0110

## Example: 8.17

The dc value coded as -

**[ -26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 -1 -1 EOB ]**

**1010110**

## Example: 8.17

The ac value code -

**[ -26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 0 -1 -1 EOB ]**

**0100**

# JPEG Coefficient coding Category

Range	DC Difference Category	AC Category
0	0	N/A
-1, 1	1	1
-3, -2, 2, 3	2	2
-7, ..., -4, 4, ..., 7	3	3
-15, ..., -8, 8, ..., 15	4	4
-31, ..., -16, 16, ..., 31	5	5
-63, ..., -32, 32, ..., 63	6	6
-127, ..., -64, 64, ..., 127	7	7
-255, ..., -128, 128, ..., 255	8	8
-511, ..., -256, 256, ..., 511	9	9
-1023, ..., -512, 512, ..., 1023	A	A
-2047, ..., -1024, 1024, ..., 2047	B	B
-4095, ..., -2048, 2048, ..., 4095	C	C
-8191, ..., -4096, 4096, ..., 8191	D	D
-16383, ..., -8192, 8192, ..., 16383	E	E
-32767, ..., -16384, 16384, ..., 32767	F	N/A

# JPEG default AC code

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	<b>1010 (= EOB)</b>	4	0/1	00	3
0/2	01	4	0/3	100	6
0/4	1011	8	0/5	11010	10
0/6	111000	12	0/7	1111000	14
0/8	1111110110	18	0/9	111111110000010	25
0/A	111111110000011	26	1/1	1100	5
1/2	111001	8	1/3	1111001	10
1/4	111110110	13	1/5	1111110110	16
1/6	111111110000100	22	1/7	111111110000101	23
1/8	111111110000110	24	1/9	111111110000111	25
1/A	111111110001000	26	2/1	11011	6
2/2	11111000	10	2/3	1111110111	13
2/4	111111110001001	20	2/5	111111110001010	21
2/6	111111110001011	22	2/7	111111110001100	23
2/8	111111110001101	24	2/9	111111110001110	25
2/A	111111110001111	26	3/1	111010	7
				B/1	11111010
					10

Run/ Category	Base Code	Length
0/0	<b>1010 (= EOB)</b>	4
0/1	00	3
0/2	01	4
0/3	100	6
0/4	1011	8
0/5	11010	10
0/6	111000	12
0/7	1111000	14
0/8	1111110110	18
0/9	111111110000010	25
0/A	111111110000011	26

**0100**

# JPEG default AC code

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	<b>1010 (= EOB)</b>	4			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	11111111000000	17
0/3	100	6	8/3	111111110110111	19
0/4	1011	8	8/4	111111110111000	20
0/5	11010	10	8/5	111111110111001	21
0/6	111000	12	8/6	111111110111010	22
0/7	1111000	14	8/7	111111110111011	23
0/8	1111110110	18	8/8	111111110111100	24
0/9	111111110000010	25	8/9	111111110111101	25
0/A	111111110000011	26	8/A	111111110111110	26

**01 00**

## Example: 8.17

Finally we get –

**[ -26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 0 -1 -1 EOB ]**

**1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001  
001 100101 11100110 110110 0110 11110100 000 1010**

## Example: 8.17

Finally we get

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001  
001 100101 11100110 110110 0110 11110100 000 1010

$$C_R = \left( \frac{(8 \times 8) \times 8}{92} \right) = 5.6$$

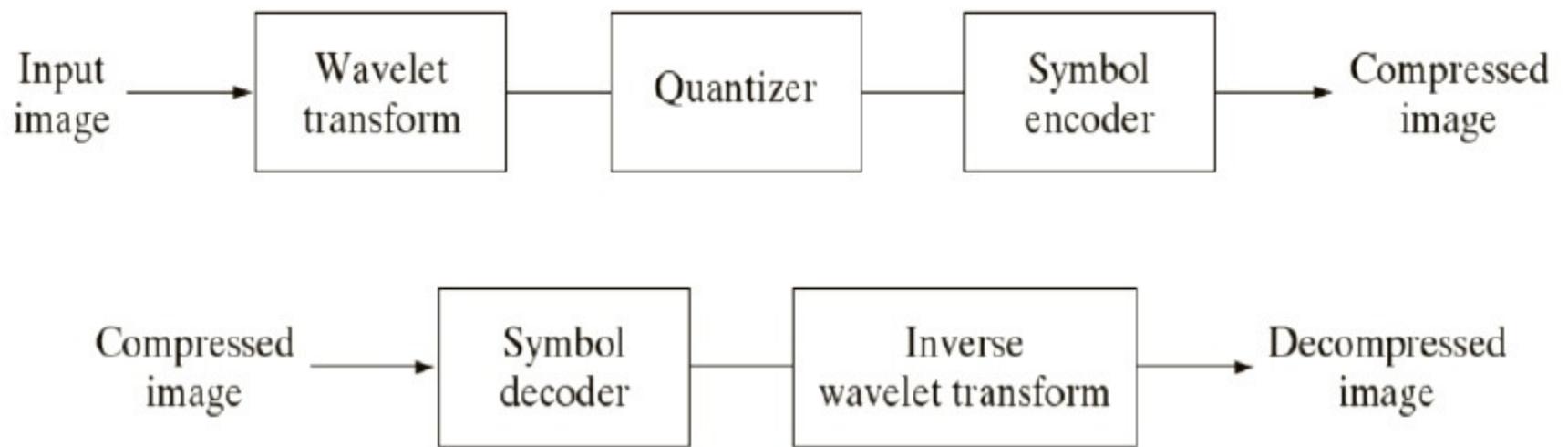
$$R_D = \left( 1 - \frac{1}{5.6} \right) = 0.82$$

JPEG2000 is the latest series of standards from the JPEG committee

- Uses wavelet Basis function
- Better compression than JPG
- Superior lossless compression
- Supports large images and images with many components
- Region-of-interest coding
- Compound documents
- Computer-generated imagery
- Other improvements over JPG

# Wavelet Compression

## Wavelet encoder and decoder

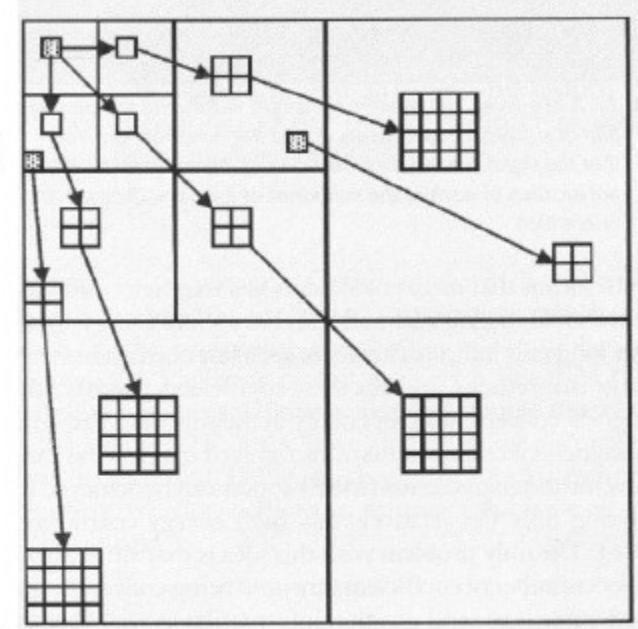


# Wavelet Compression

- “Modern” lossy wavelet coding exploits multi-resolution and self-similar nature of wavelet decomposition
  - Energy is compacted into a small number of coefficients.
  - Significant coefficients tend to cluster at the same spatial location in each frequency subband
- Two set of information to code:
  - Where are the significant coefficients?
  - What values are the significant coefficients?

# Wavelet Compression

- Parent-children relation among coeff.
  - Each parent coeff at level  $k$  spatially correlates with 4 coeff at level  $(k-1)$  of same orientation
  - A coeff at lowest band correlates with coeff.
- Coding significance map via zero-tree
  - Encode only high energy coefficients
    - Need to send location info.  
→ large overhead
  - Encode “insignificance map” w/ zero-trees
- Successive approximation quantization
  - Send most-significant-bits first and gradually refine coeff. value
  - “Embedded” nature of coded bit-stream
    - get higher fidelity image by adding extra refining bits



# Image Morphology Processing

- Morphological image processing is a collection of non-linear operations related to the shape or morphology features in an image.
- Morphological processing is constructed with operations on sets of pixels.
- Morphological techniques probe an image with a small shape or template called a **structuring element**. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighbor hood of pixels.

# Image Morphology Processing

Binary morphology is used to extract image components that are useful in the representation and description of region shape, such as

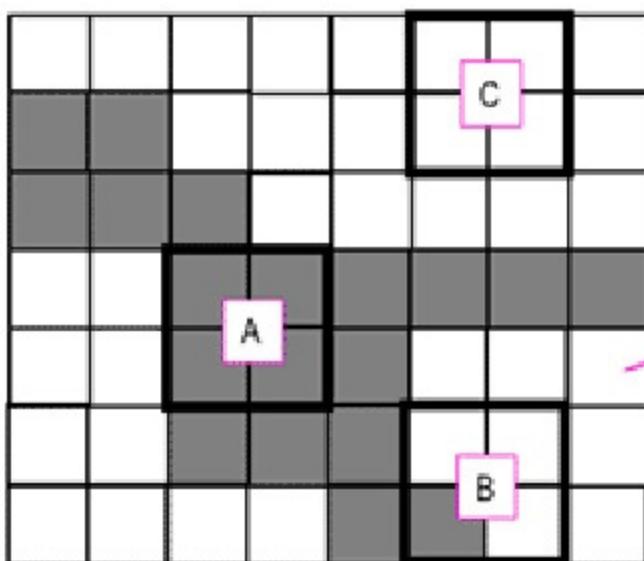
1. Boundaries extraction
2. Skeletons
3. Convex hull
4. Morphological filtering
5. Thinning
6. Pruning

# Image Morphology Processing

- It is based on set theory in algebra.
- Sets in mathematical morphology represent objects in an image.
- A tool for **extracting image components** that are useful in the representation and description of region shape, such as boundaries, skeletons etc.

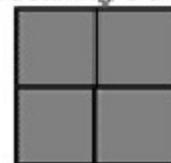
# Image Morphology Processing

- Some operations test whether the element "fits" within the neighbor hood, while others test whether it "hits" or intersects the neighbor hood



- A - the structuring element fits the image
- B - the structuring element hits (intersects) the image
- C - the structuring element neither fits, nor hits the image

Structuring element



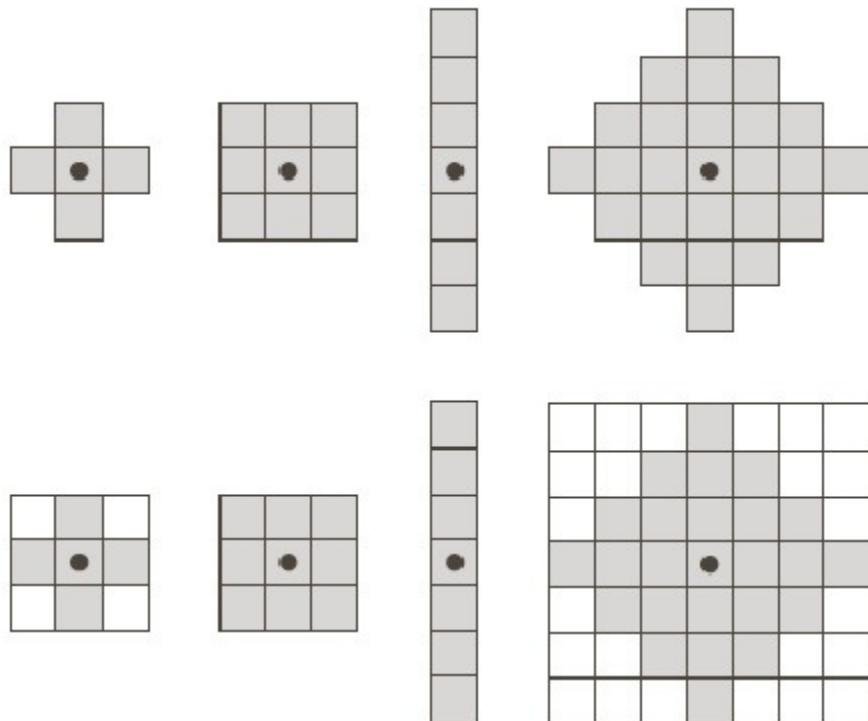
## Morphology Operator

- A **morphological operator** is therefore defined by its structuring element and the applied set operator.
- If the two sets of elements **match the condition** defined by the set operator, the pixel neighboring **the origin of the structuring element** is set to a pre-defined value (0 or 1 for binary images).

## Structure Element

The **structure element** is a small binary image, each with a value of zero or one.

- The matrix dimensions specify the size of the structuring element.
- The pattern of ones and zeros specifies the shape of the structuring element Image.



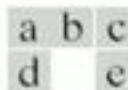
**FIGURE 9.2** First row: Examples of structuring elements. Second row: Structuring elements converted to rectangular arrays. The dots denote the centers of the SEs.

- The dilation of  $A$  by  $B$ , denoted

$$A \oplus B = \{z \mid z = x + y, x \in A, y \in B\}$$

where set  $B$  is referred to as the **structuring element**.

- The dilation of  $A$  by  $B$  is the set of all displacements  $z$ , such that  $B$  and  $A$  overlap by at least one element.



**FIGURE 9.4**

- (a) Set  $A$ .
- (b) Square structuring element (dot is the center).
- (c) Dilation of  $A$  by  $B$ , shown shaded.
- (d) Elongated structuring element.
- (e) Dilation of  $A$  using this element.

- The dilation of  $A$  by  $B$ , denoted

$$A \oplus B = \{z \mid z = x + y, x \in A, y \in B\}$$

where set  $B$  is referred to as the **structuring element**.

- The dilation of  $A$  by  $B$  is the set of all displacements  $z$ , such that  $B$  and  $A$  overlap by at least one element.

**FIGURE 9.4**

- (a) Set  $A$ .
- (b) Square structuring element (dot is the center).
- (c) Dilation of  $A$  by  $B$ , shown shaded.
- (d) Elongated structuring element.
- (e) Dilation of  $A$  using this element.

First Semester 2021-22

EEE F435

K. K. Gupta

10

- To compute the dilation of a **binary input image** by this structuring element, we consider each of the pixel in the input image in turn.
- For each, we superimpose the structuring element on top of the input image pixel so that the origin of the structuring element coincides with the input pixel position.

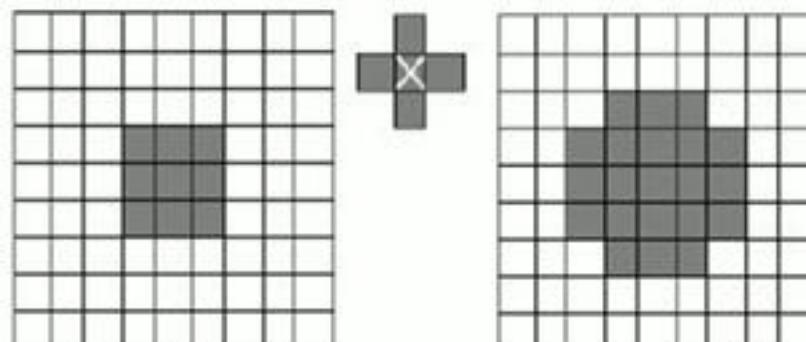
If *at least one* pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value.

If all the corresponding pixels in the image are background, however, the input pixel is left at the background value.

- To compute the dilation of a **binary input image** by this structuring element, we consider each of the pixel in the input image in turn.
- For each, we superimpose the structuring element on top of the input image pixel so that the origin of the structuring element coincides with the input pixel position.

If *at least one* pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value.

If all the corresponding pixels in the image are background, however, the input pixel is left at the background value.



**Image****Structuring element**

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

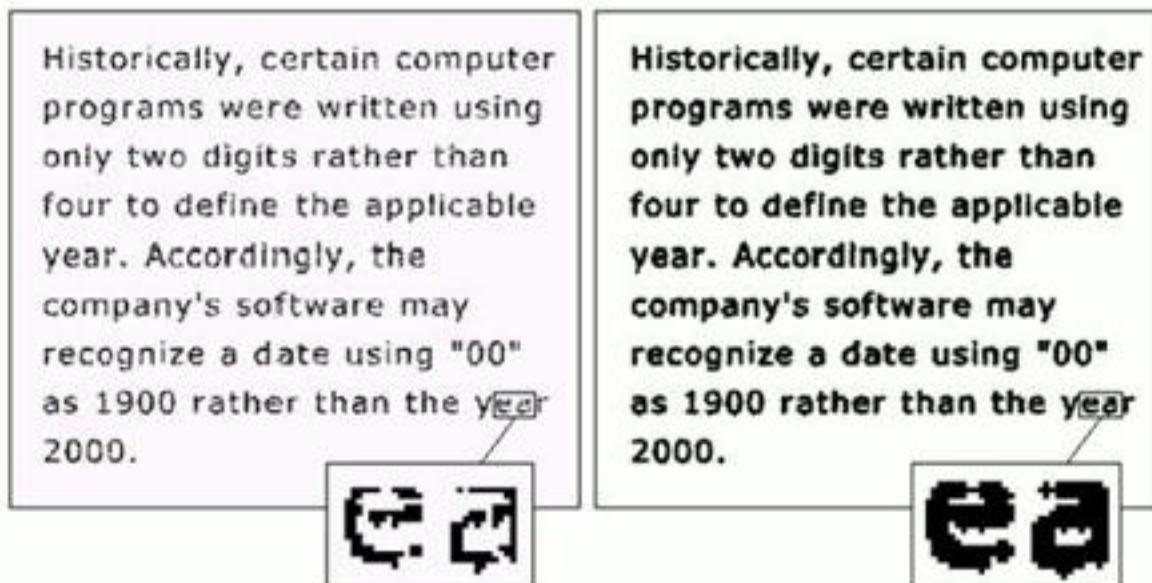
$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & \textcircled{1} & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

**Processed Image**

## Example: Bridging gaps

The maximum length of the breaks is known to be two pixels.



a      b      c

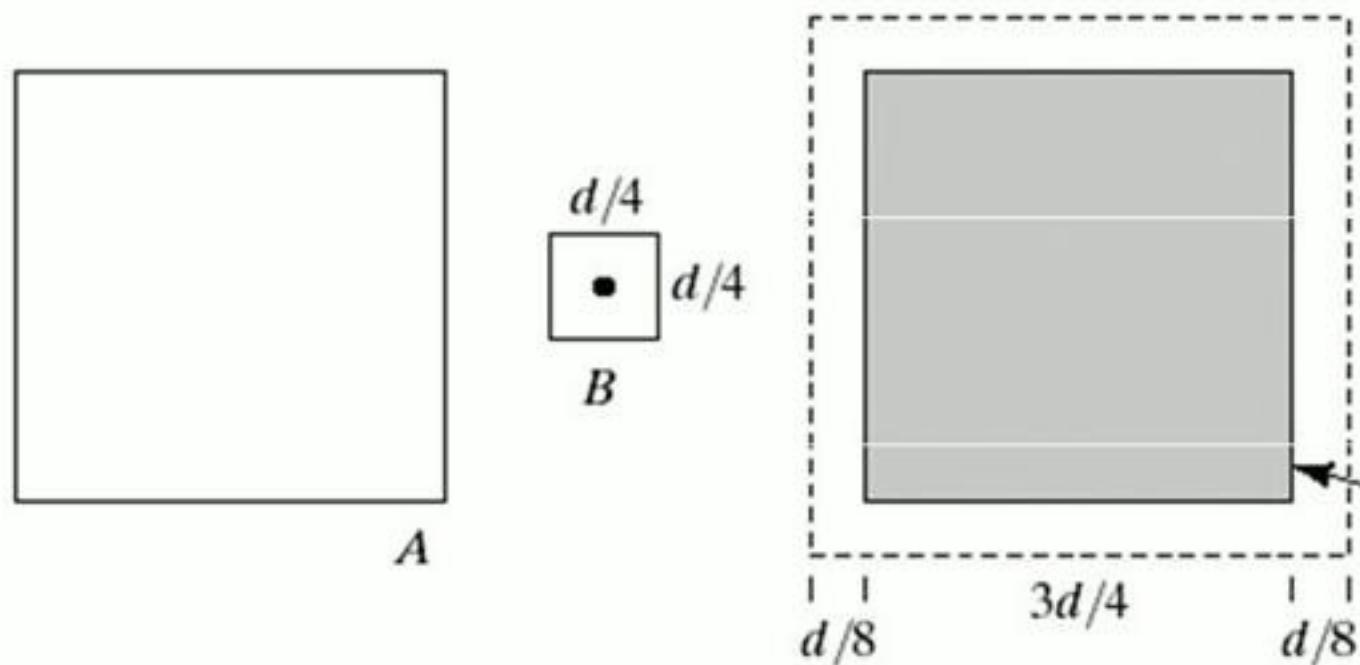
**FIGURE 9.5**  
(a) Sample text of poor resolution with broken characters (magnified view).  
(b) Structuring element.  
(c) Dilation of (a) by (b). Broken segments were joined.

- The erosion of  $A$  by  $B$ , denoted  $A \Theta B$

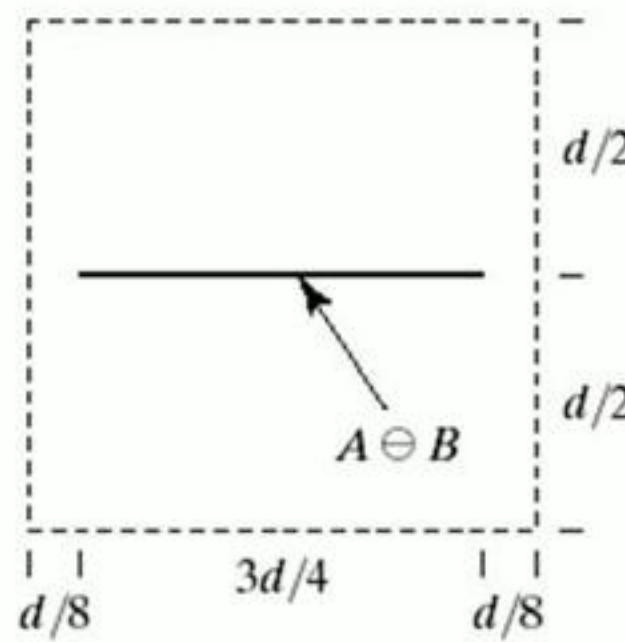
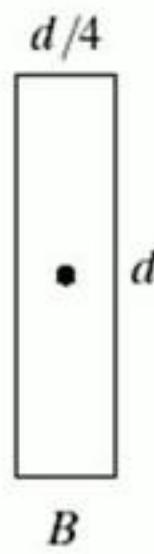
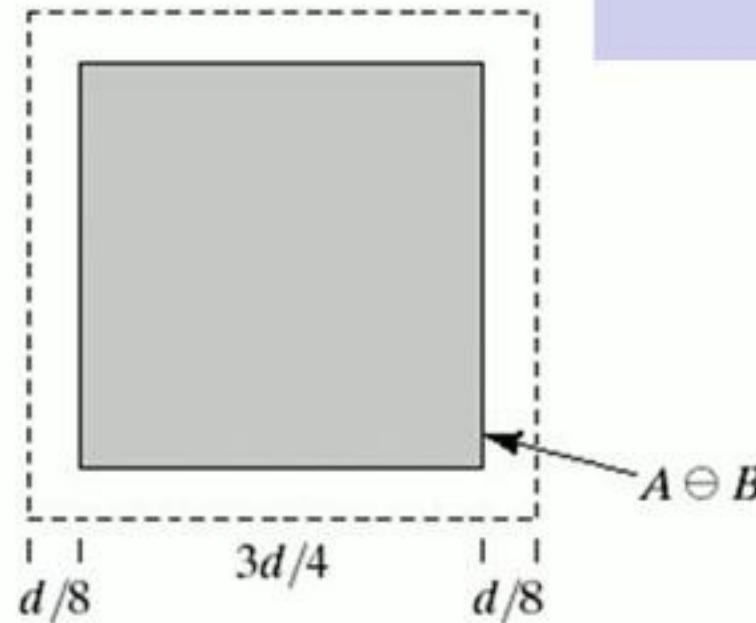
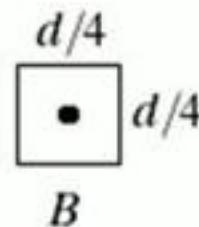
$$A \Theta B = \{z \mid z - B \subseteq A\}$$

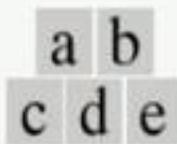
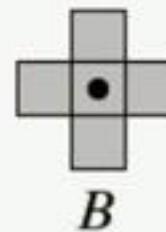
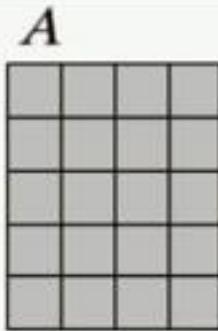
where set  $B$  is referred to as the **structuring element**.

- The erosion of  $A$  by  $B$  is the set of all points  $z$  such that  $B$ , translated by  $z$ , is contained in  $A$ .

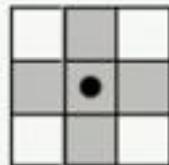
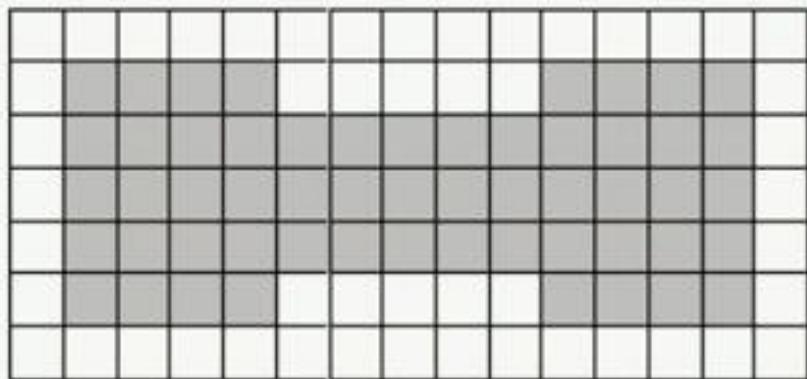
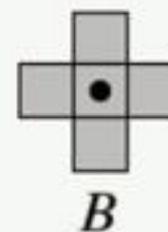
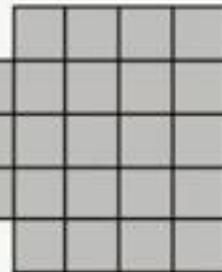
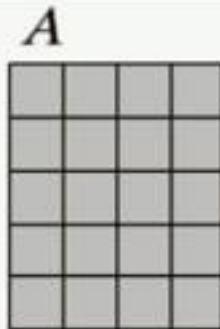


# Example





**FIGURE 9.3** (a) A set (each shaded square is a member of the set). (b) A structuring element. (c) The set padded with background elements to form a rectangular array and provide a background border. (d) Structuring element as a rectangular array. (e) Set processed by the structuring element.



a	b	
c	d	e

**FIGURE 9.3** (a) A set (each shaded square is a member of the set). (b) A structuring element. (c) The set padded with background elements to form a rectangular array and provide a background border. (d) Structuring element as a rectangular array. (e) Set processed by the structuring element.

**Image****structuring element**

0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	1	0	1	1	0	0
0	1	1	1	1	1	1	0
0	0	1	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	0	1	1	0	0
0	0	0	0	0	1	0	0

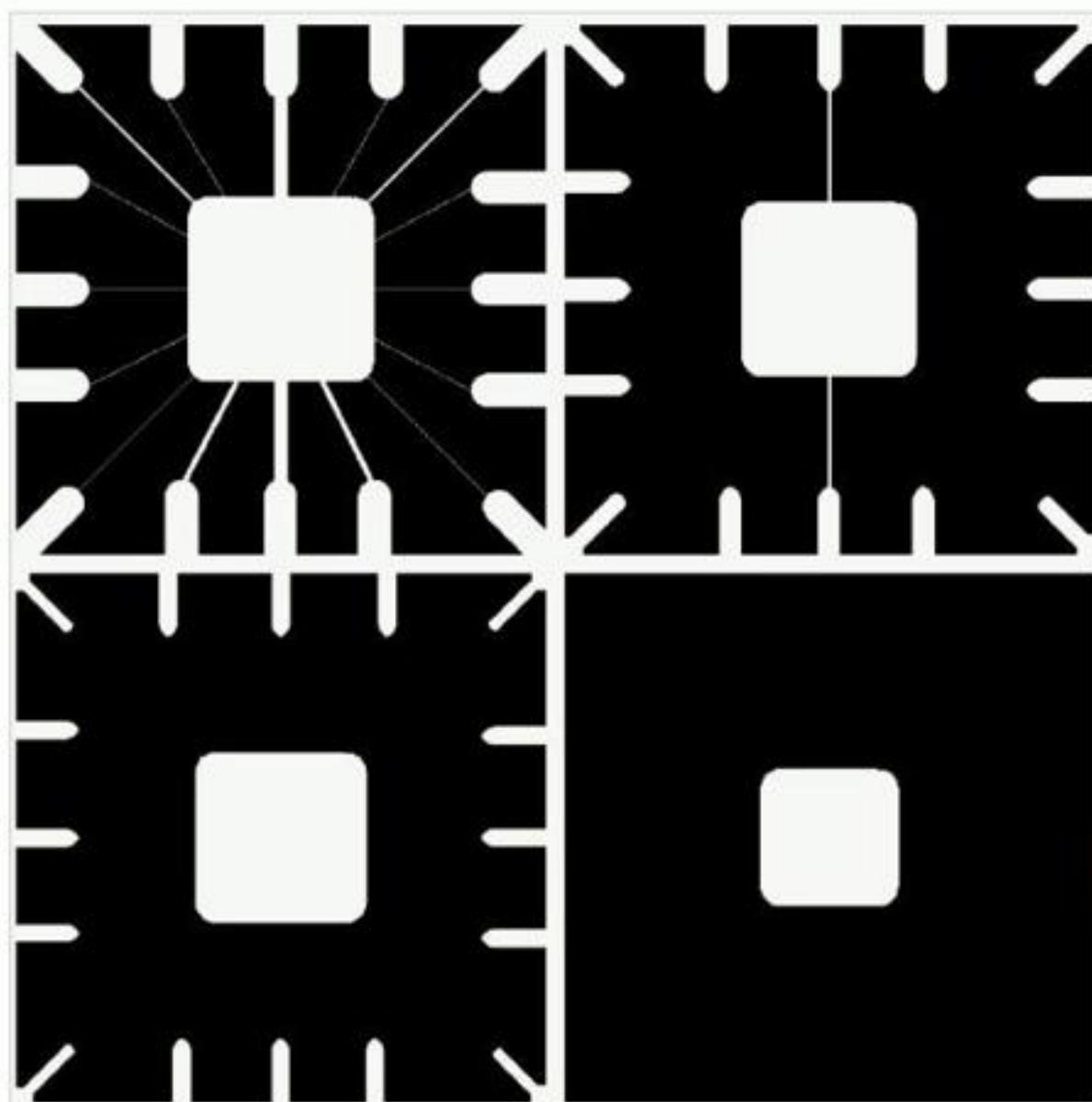
1	1	0
0	1	0
1	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0

**Processed Image**

## Eliminating irrelevant detail

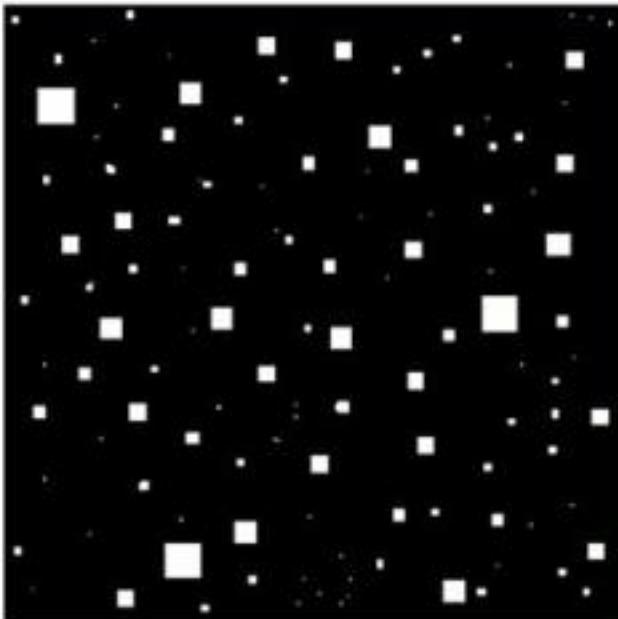
### Example: Erosion



a	b
c	d

**FIGURE 9.5** Using erosion to remove image components. (a) A  $486 \times 486$  binary image of a wire-bond mask. (b)–(d) Image eroded using square structuring elements of sizes  $11 \times 11$ ,  $15 \times 15$ , and  $45 \times 45$ , respectively. The elements of the SEs were all 1s.

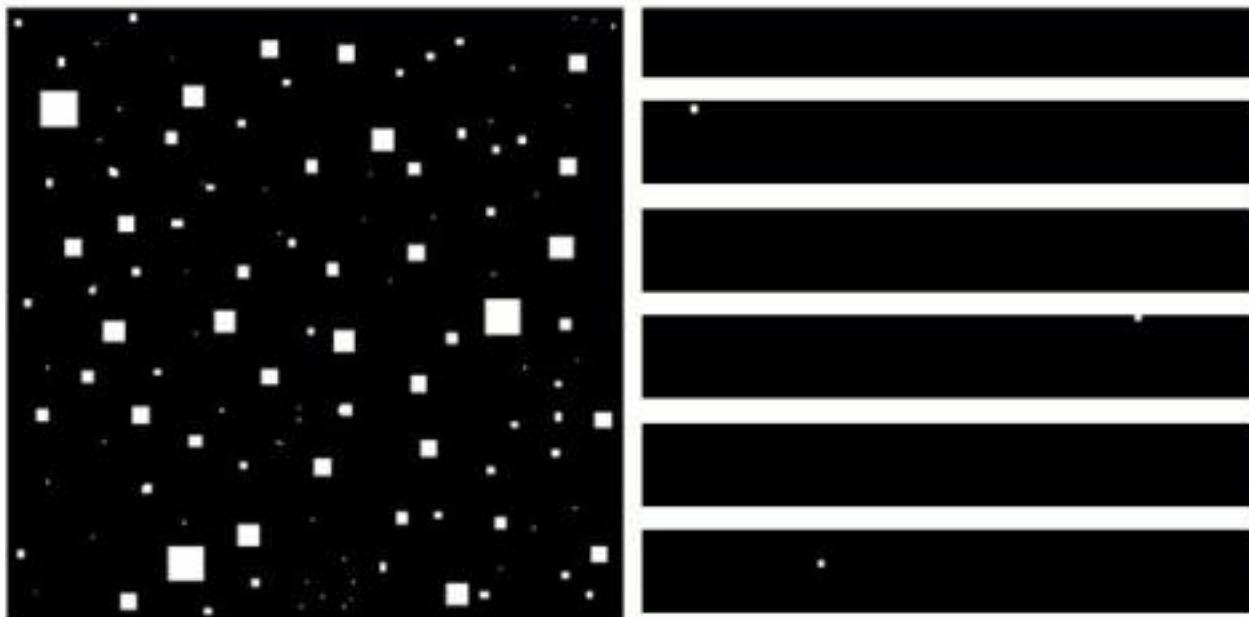
Find the total number of maximum size squares in given image



### Eliminating irrelevant detail

- a) Image of squares of size 1,3,5,7,9,15 pixels on the side

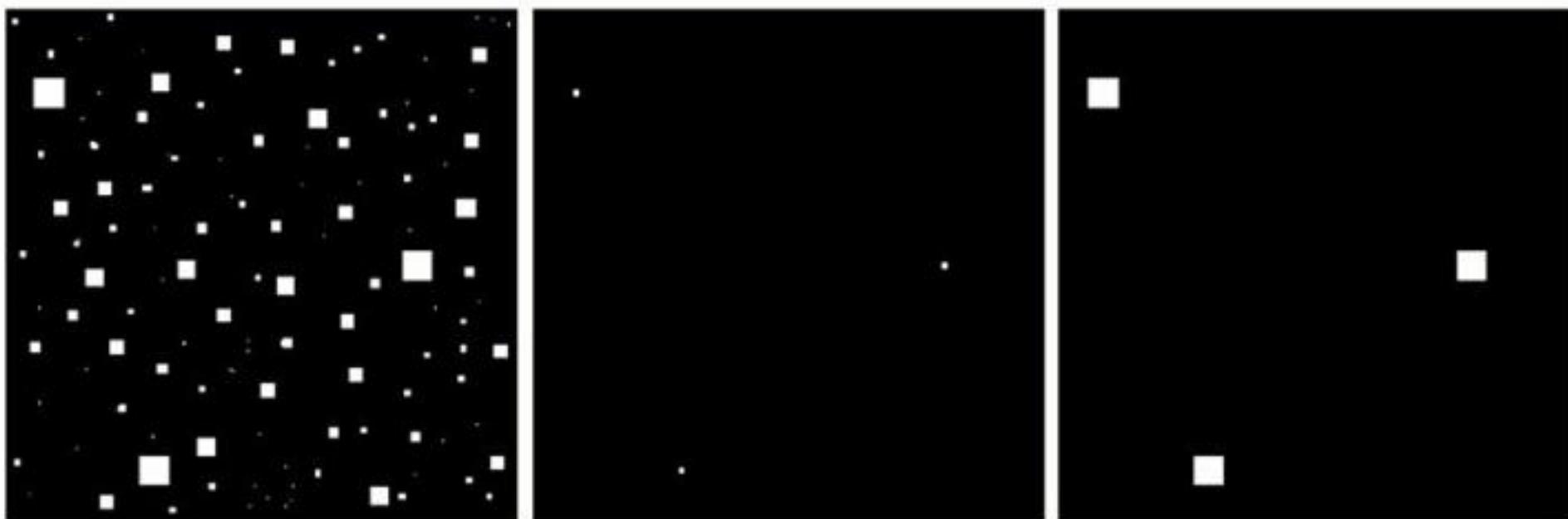
Find the total number of maximum size squares in given image



### Eliminating irrelevant detail

- a) Image of squares of size 1,3,5,7,9,15 pixels on the side
- b) Erosion of 'a' with square of elements of 1's 13 pixels

Find the total number of maximum size squares in given image



### Eliminating irrelevant detail

- a) Image of squares of size 1,3,5,7,9,15 pixels on the side
- b) Erosion of 'a' with square of elements of 1's 13 pixels
- c) Dilation of 'b' by same structural elements

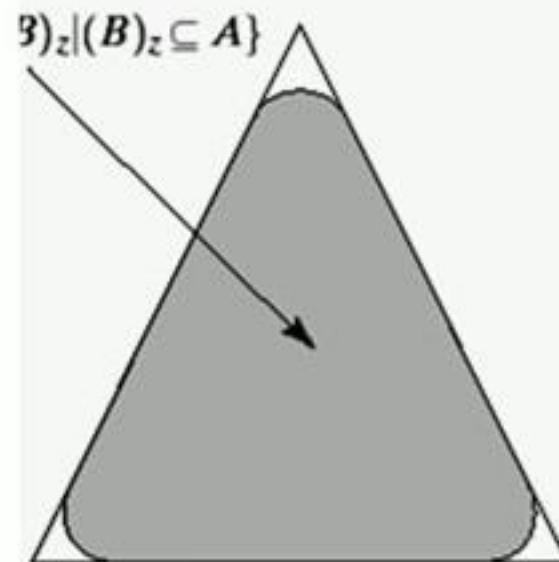
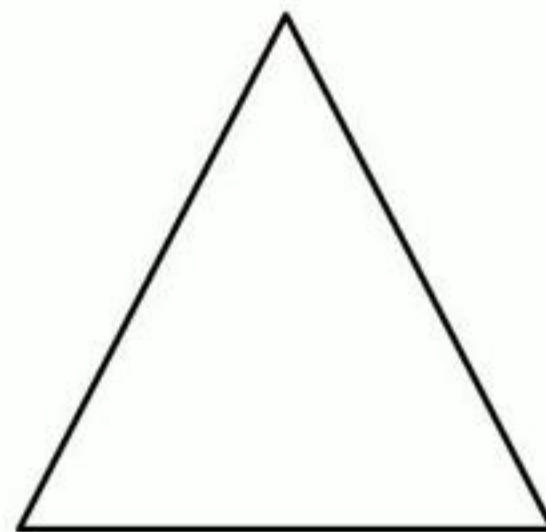
## Opening operation

The opening  $A$  by  $B$  is the erosion of  $A$  by  $B$ , followed by a dilation of the result by  $B$ .

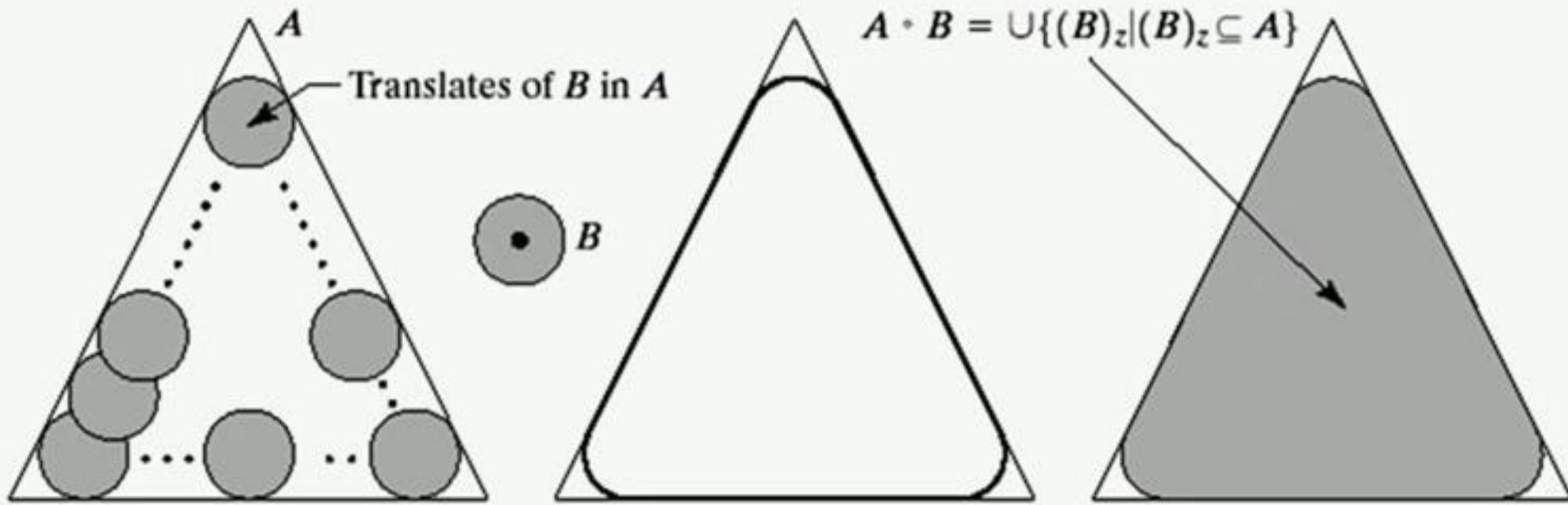
$$A \circ B = (A \ominus B) \oplus B$$

The opening operation opens small gaps between touching objects in an image.

Generally opening smoothes the contour of an object.

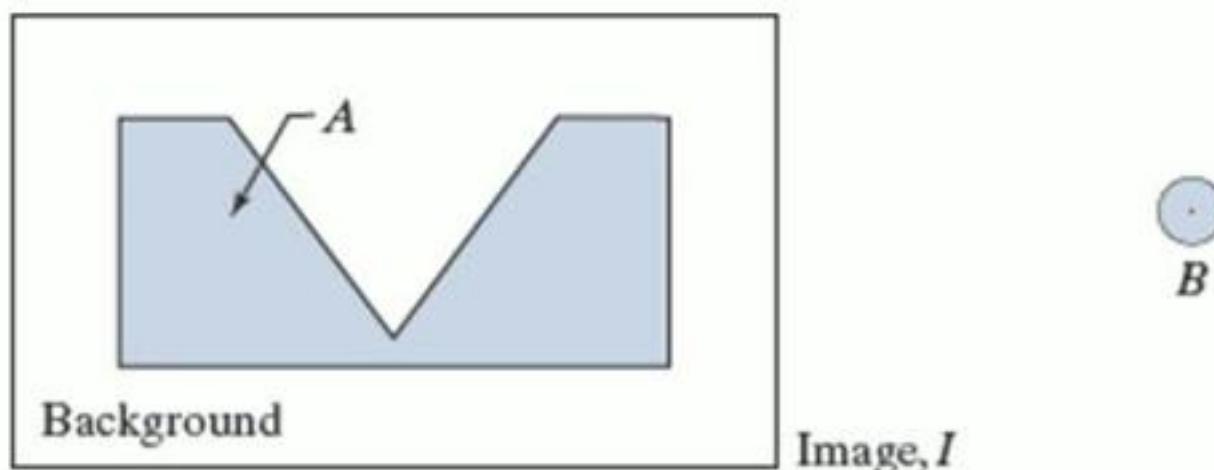


- View the structuring element  $B$  as a flat “rolling ball”
- The boundary of  $A \circ B$  is then established by the points in  $B$  that reach the farthest into the boundary of  $A$  as  $B$  is rolled around the inside of this boundary.



## Opening operation

- View the structuring element  $B$  as a flat “rolling ball”
- The boundary of  $A \circ B$  is then established by the points in  $B$  that reach the farthest into the boundary of  $A$  as  $B$  is rolled around the inside of this boundary.

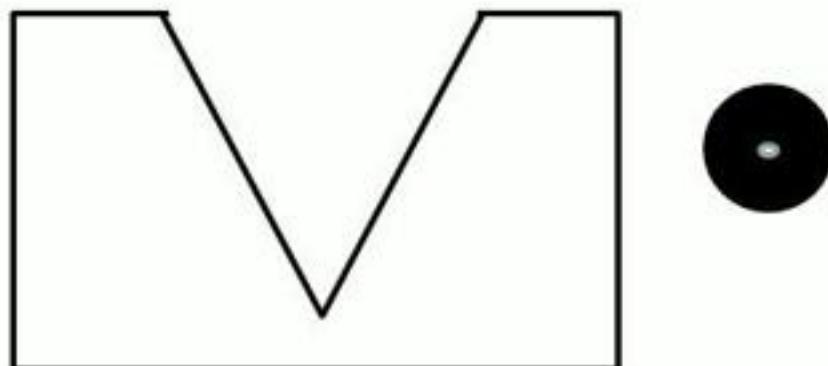


The closing of set  $A$  by structuring element  $B$ , denoted  $A \bullet B$

$$A \bullet B = (A \oplus B) \ominus B$$

**The closing of  $A$  by  $B$  is simply the dilation of  $A$  by  $B$ , followed by the erosion of the result by  $B$ .**

- Tends to smooth sections of contours, fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.

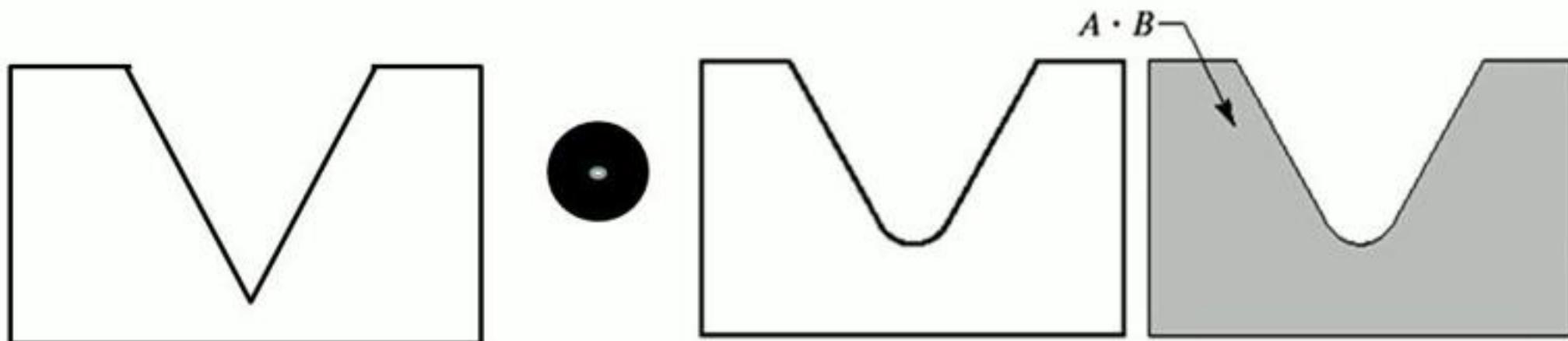


The closing of set  $A$  by structuring element  $B$ , denoted  $A \bullet B$

$$A \bullet B = (A \oplus B) \ominus B$$

**The closing of  $A$  by  $B$  is simply the dilation of  $A$  by  $B$ , followed by the erosion of the result by  $B$ .**

- Tends to smooth sections of contours, fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.

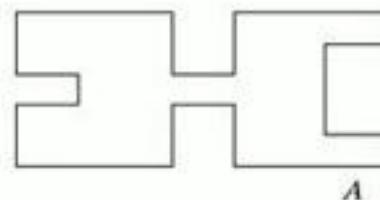


# Opening and Closing

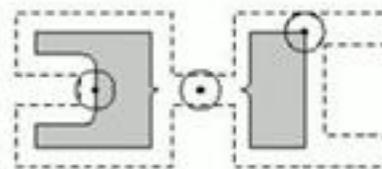
a
b c
d e
f g
h i

**FIGURE 9.10**

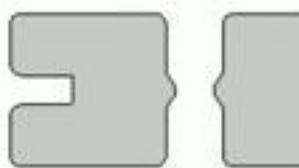
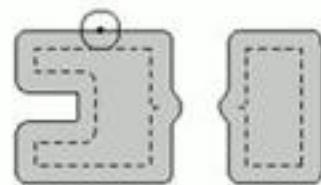
Morphological opening and closing. The structuring element is the small circle shown in various positions in (b). The dark dot is the center of the structuring element.



A

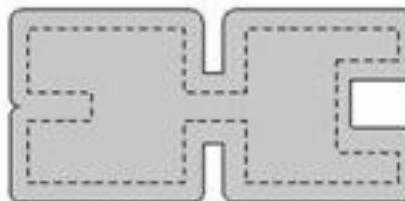


$A \ominus B$

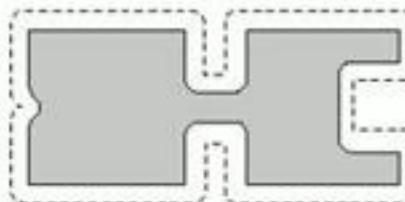


$A + B = (A \ominus B) \oplus B$

**Opening**



$A \oplus B$

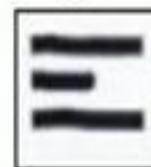


$A - B = (A \oplus B) \ominus B$

**Closing**

# Application: Fingerprint

Minutiae Include



Minutiae

ridge ending



bifurcation



dot



island (short ridge)

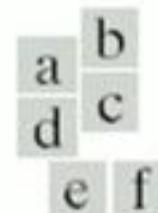
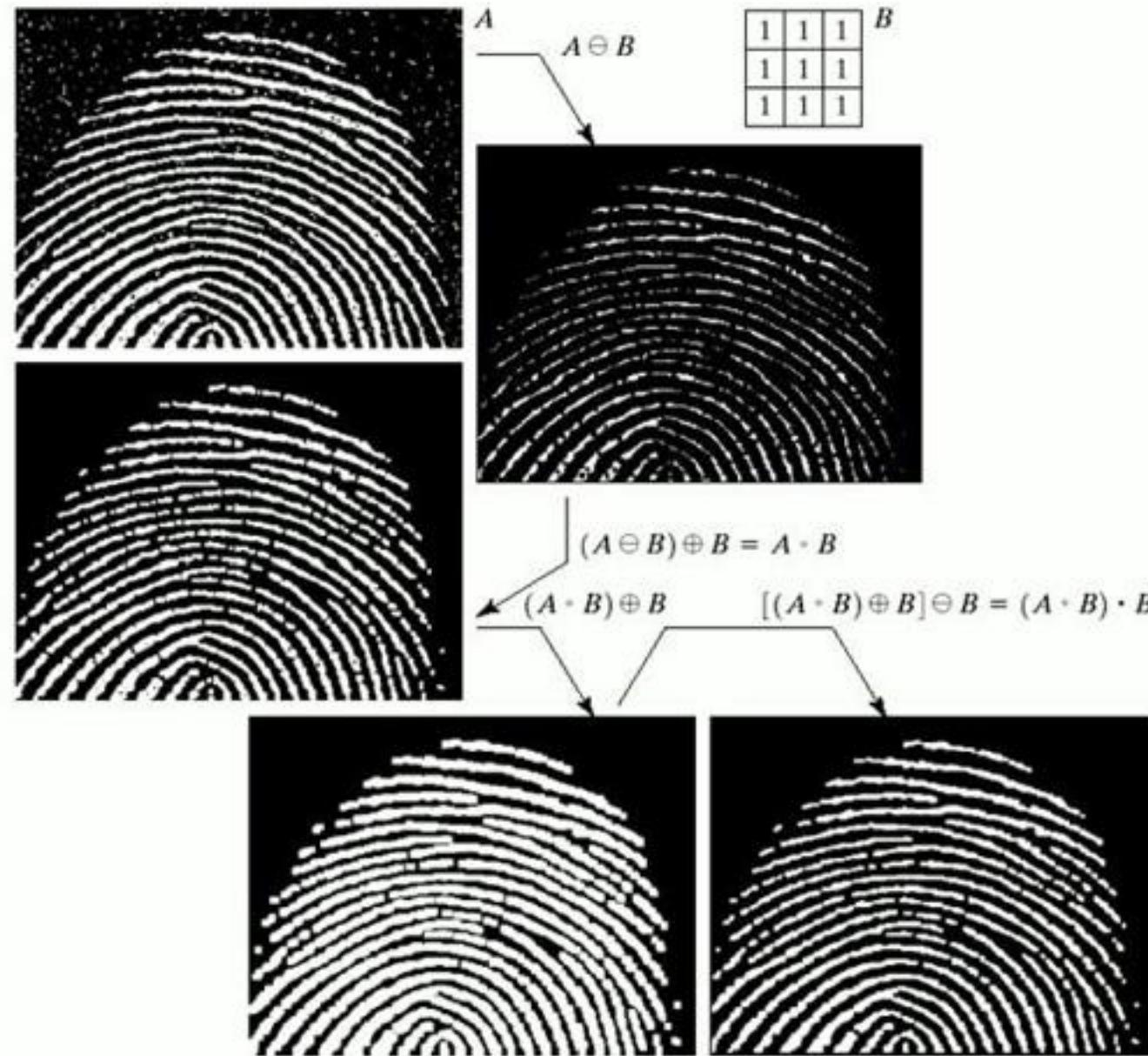


lake (enclosure)



hook (spur)

# Application: Fingerprint



**FIGURE 9.11**

(a) Noisy image.  
(c) Eroded image.  
(d) Opening of  $A$ .  
(d) Dilation of the opening.  
(e) Closing of the opening. (Original image for this example courtesy of the National Institute of Standards and Technology.)

## Properties of Opening & Closing

- $A^\circ B$  is a subset of  $A$
  - If  $C$  is a subset of  $D$ ,  
then  $C^\circ B$  is a subset of  $D^\circ B$
  - $(A^\circ B)^\circ B = A^\circ B$
- 
- $A$  is a subset of  $A \bullet B$
  - If  $C$  is a subset of  $D$ ,  
then  $C \bullet B$  is a subset of  $D \bullet B$
  - $(A \circ B) \circ B = A \circ B$

## Properties of Opening & Closing

- $A^\circ B$  is a subset of  $A$
  - If  $C$  is a subset of  $D$ ,  
then  $C^\circ B$  is a subset of  $D^\circ B$
  - $(A^\circ B)^\circ B = A^\circ B$
- 
- $A$  is a subset of  $A \bullet B$
  - If  $C$  is a subset of  $D$ ,  
then  $C \bullet B$  is a subset of  $D \bullet B$
  - $(A \bullet B) \bullet B = A \bullet B$

Multiple openings or closings of a set have no effect  
after the operator has been applied once

# Application: Feature extraction

(a)

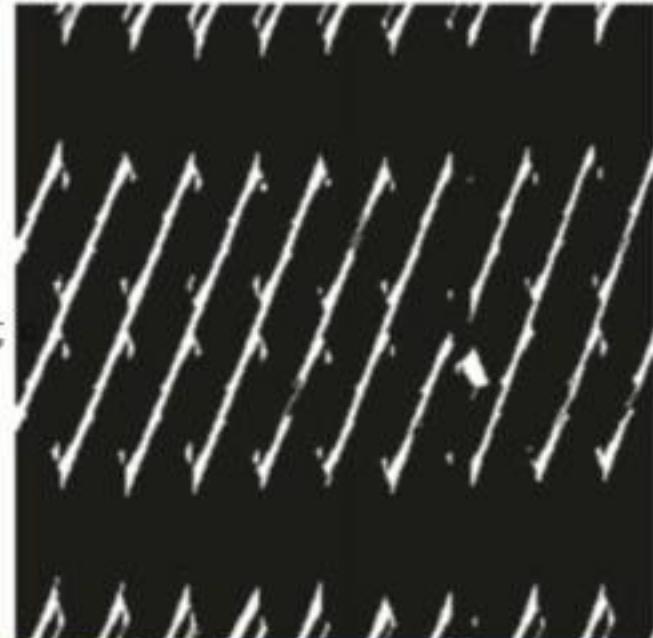


(b)



- a) A  $256 \times 256$  binary image;
- b) the output of the **opening operation** by a structuring element, which is a straight line at  $60^\circ$ .
- c) the output of the **erosion operation** alone

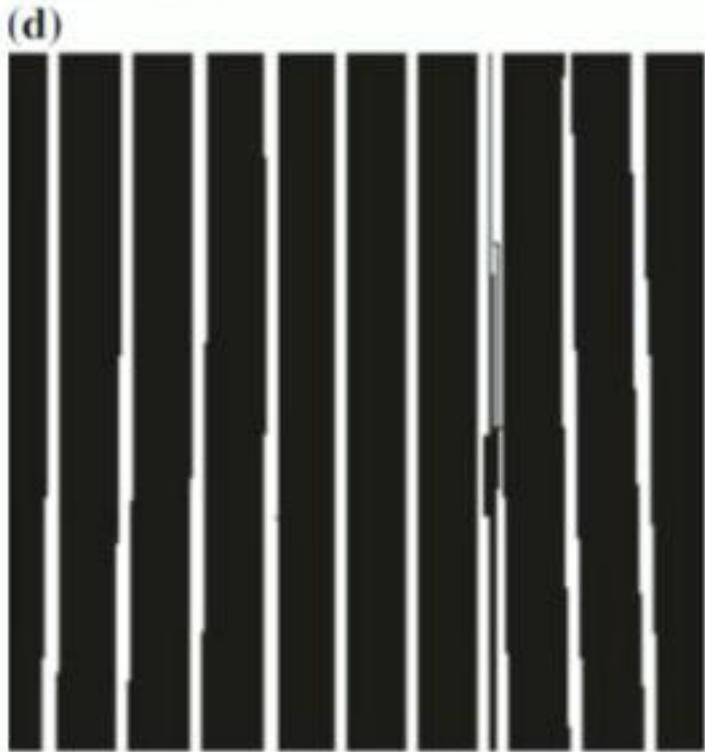
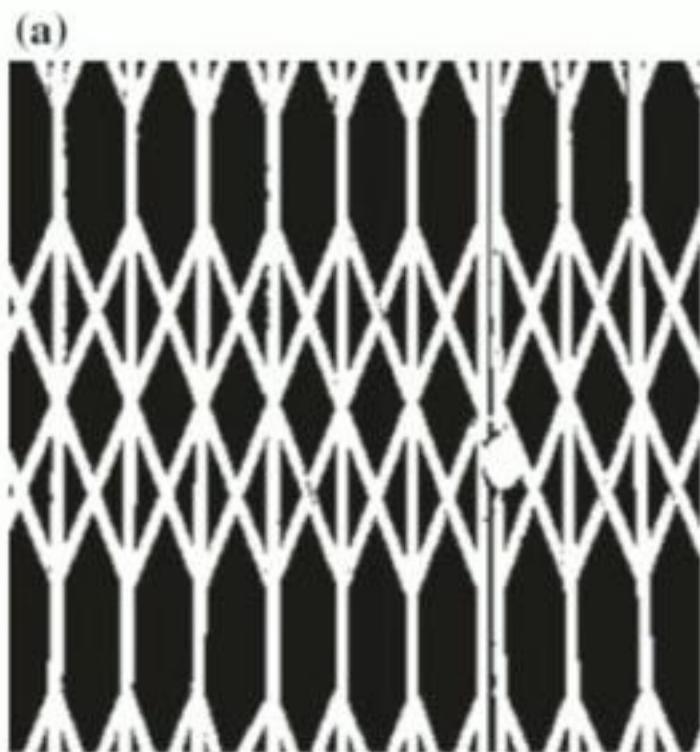
(c)



**a** A  $256 \times 256$  binary image; **b** the output of the opening operation by a structuring element, which is a straight line at  $60^\circ$ ; the output of the erosion operation alone

## Application: Feature extraction

- a) A  $256 \times 256$  binary image;  
d) the output of the opening operation by a structuring element,  
which is a vertical line;



# Morphological Watersheds

The concept of watersheds is based on visualizing an image in three dimensions:

**two spatial coordinates versus gray levels.**



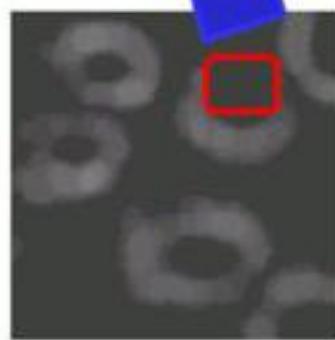


In such a topographic interpretation, we consider three types of points:

- (a) points belonging to a regional minimum.
- (b) points at which a drop of water would fall with certainty to a single minimum
- (c) points at which water would be equally likely to fall to more than one such minimum

The set of points satisfying (b) is called watershed or catchment basin and condition (c) form crest lines or watershed divide line

## Example

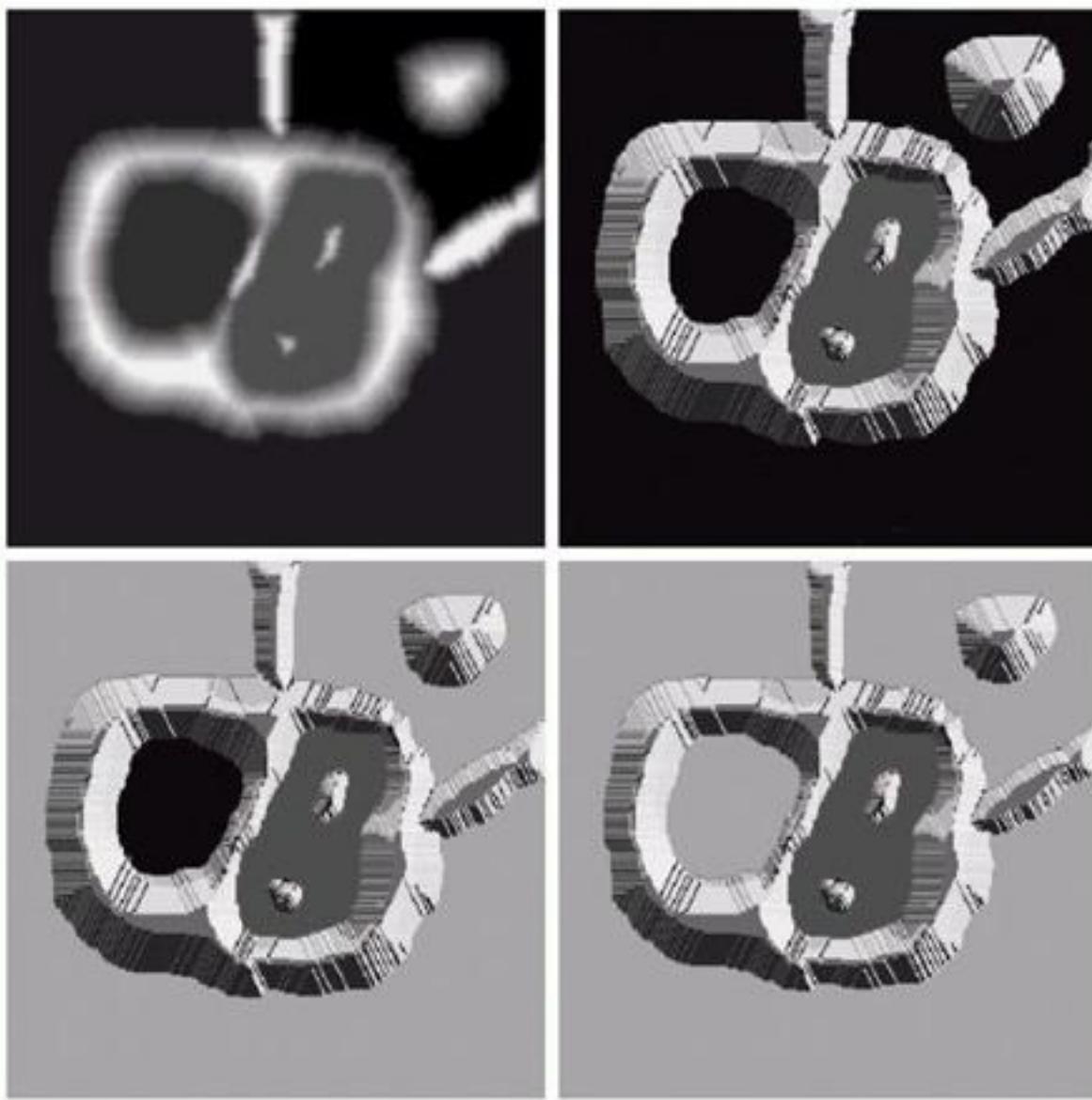


16.8	19.2	13.5	20.5	31.2	30.1
18.7	11.8	15.4	18.5	22.1	18.4
20.1	21.9	26.6	20.8	17.3	18.1
25.3	22.8	20.9	19.8	15.1	15.9
30.7	35.5	29.9	18.7	17.6	39.9
34.8	38.6	33.4	32.7	33.5	36.7

a  
b  
c  
d

**FIGURE 10.44**

(a) Original image.  
(b) Topographic view. (c)–(d) Two stages of flooding.

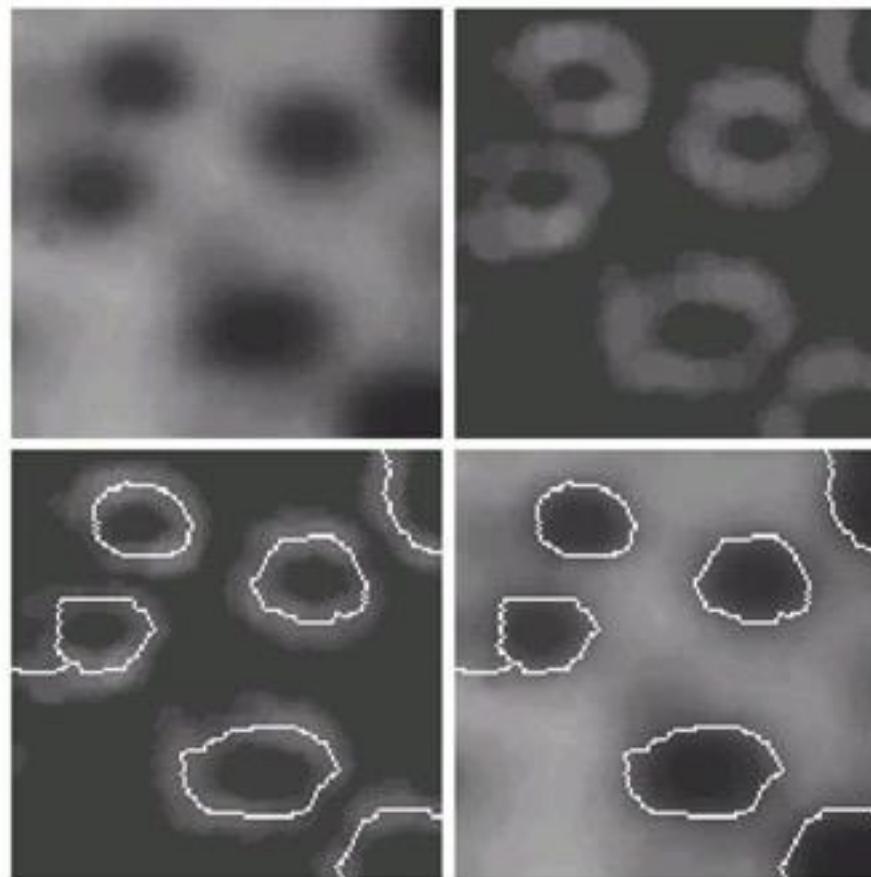


A hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a **uniform rate**.

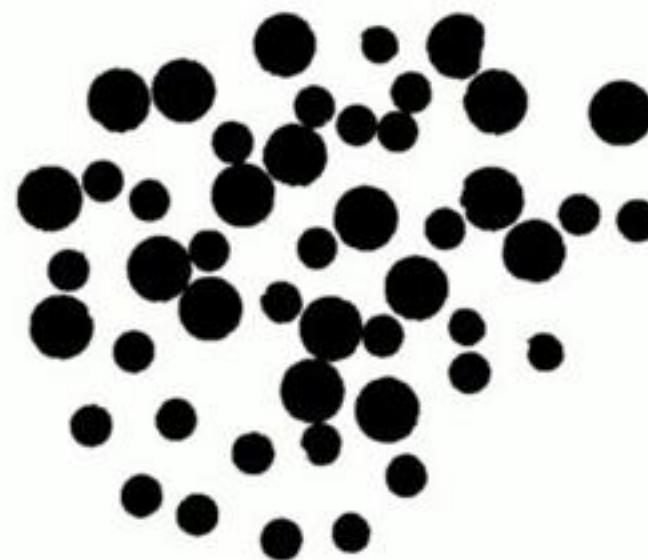
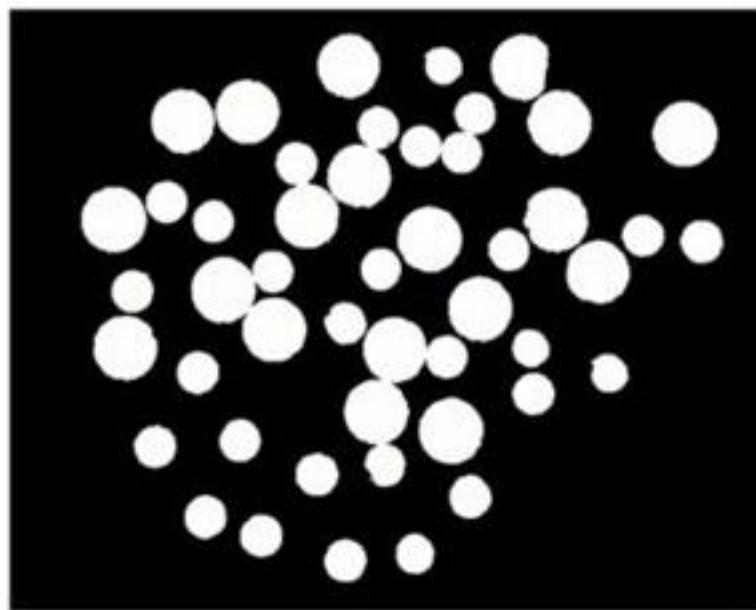
# Watershed segmentation

a  
b  
c  
d

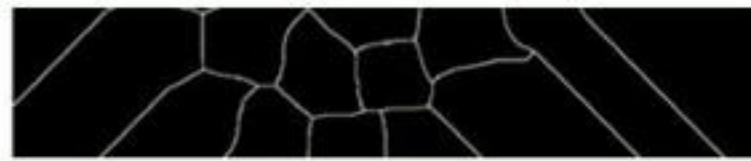
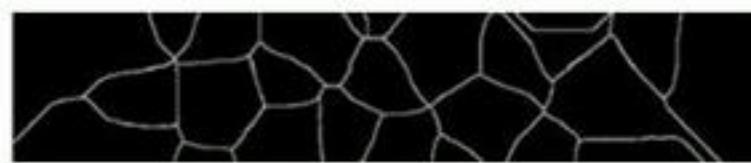
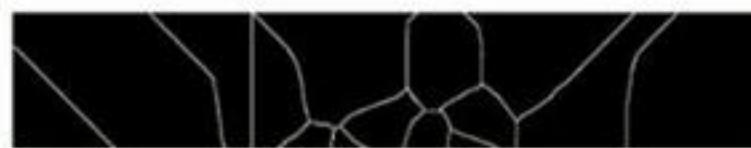
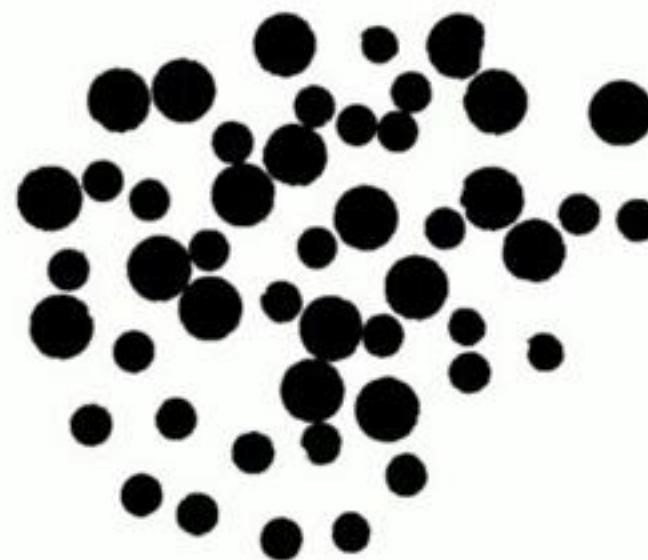
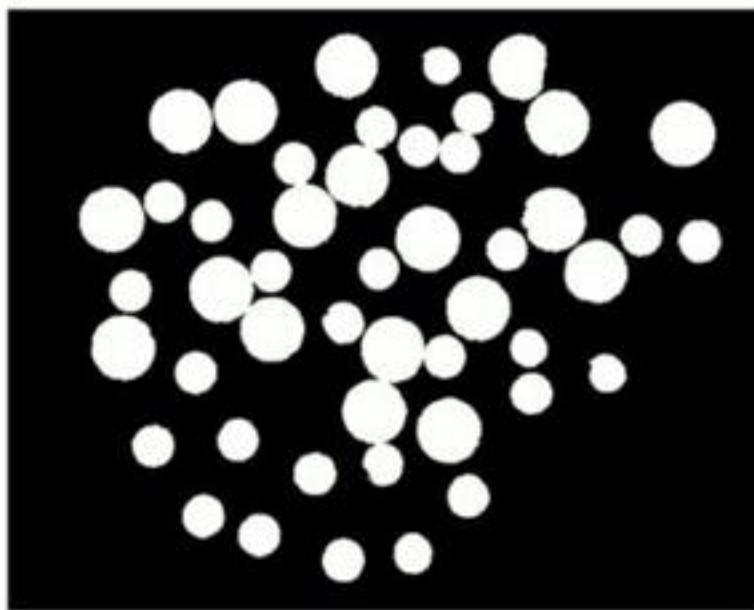
**FIGURE 10.46**  
(a) Image of blobs.  
(b) Image gradient.  
(c) Watershed lines.  
(d) Watershed lines superimposed on original image.  
(Courtesy of Dr. S. Beucher,  
CMM/Ecole des Mines de Paris.)



# Watershed segmentation



# Watershed segmentation

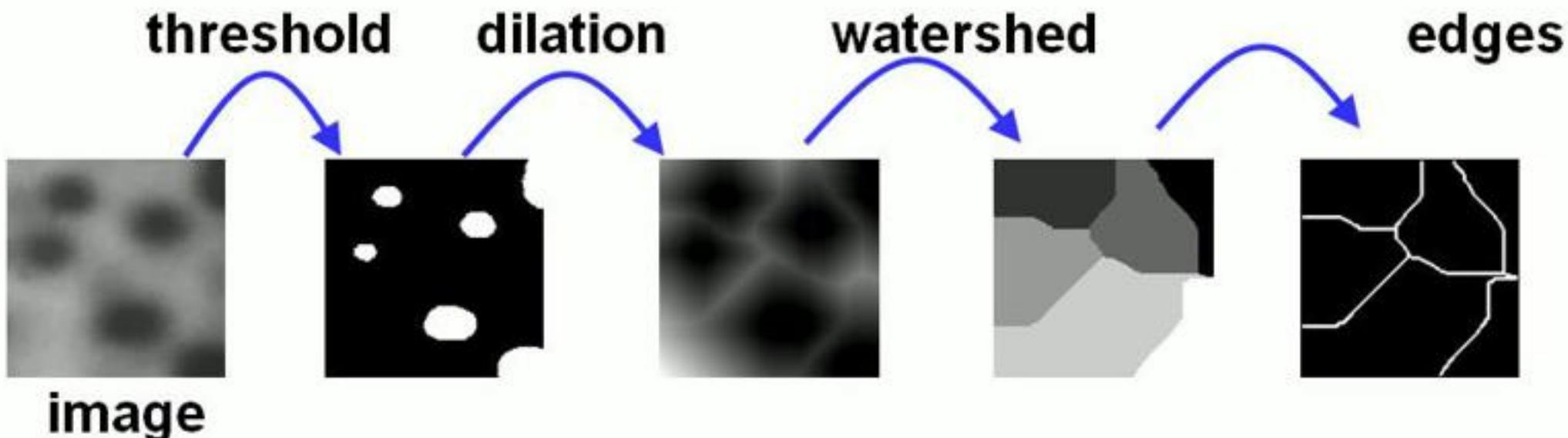


# Marker-based watershed

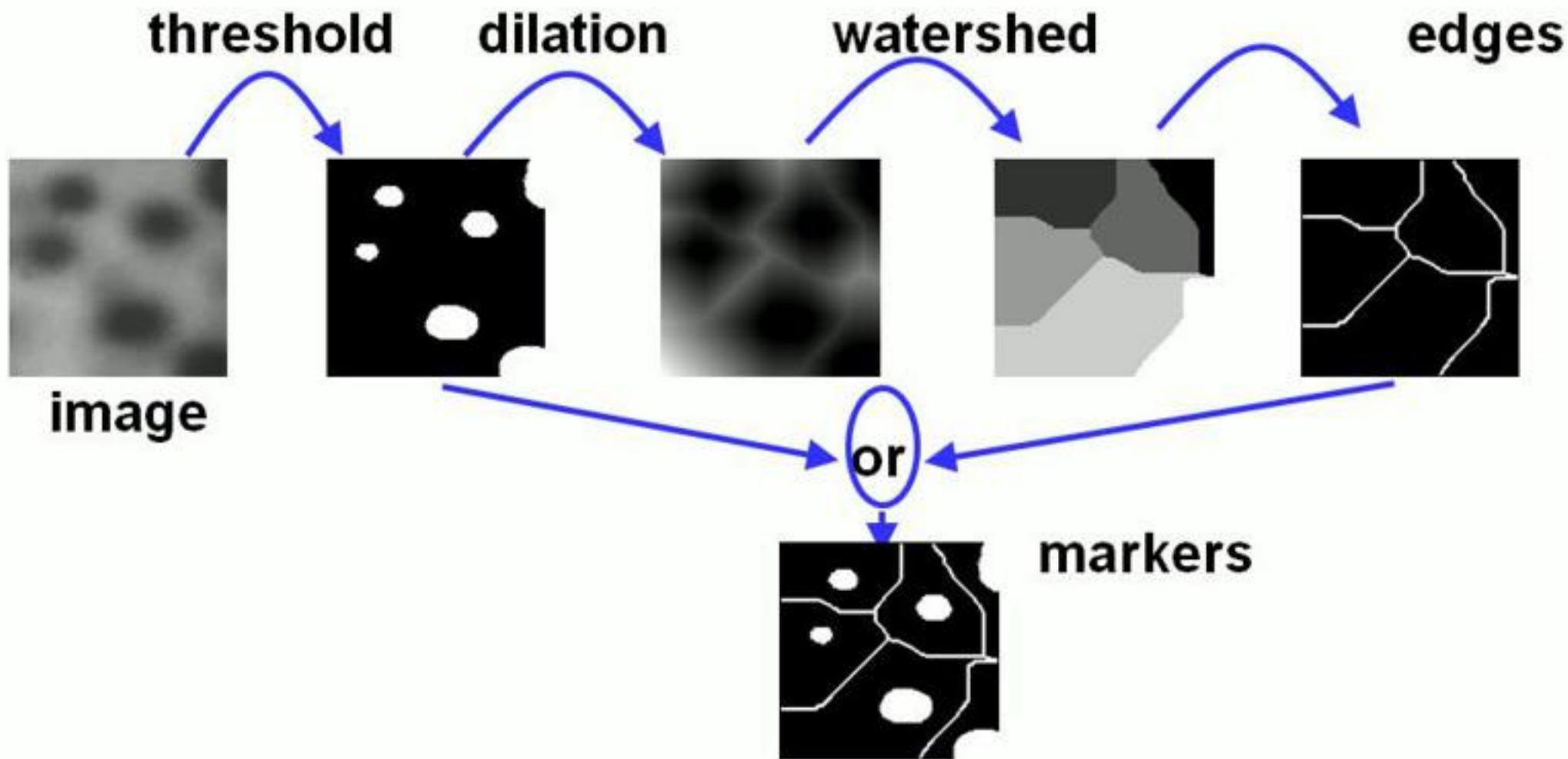


image

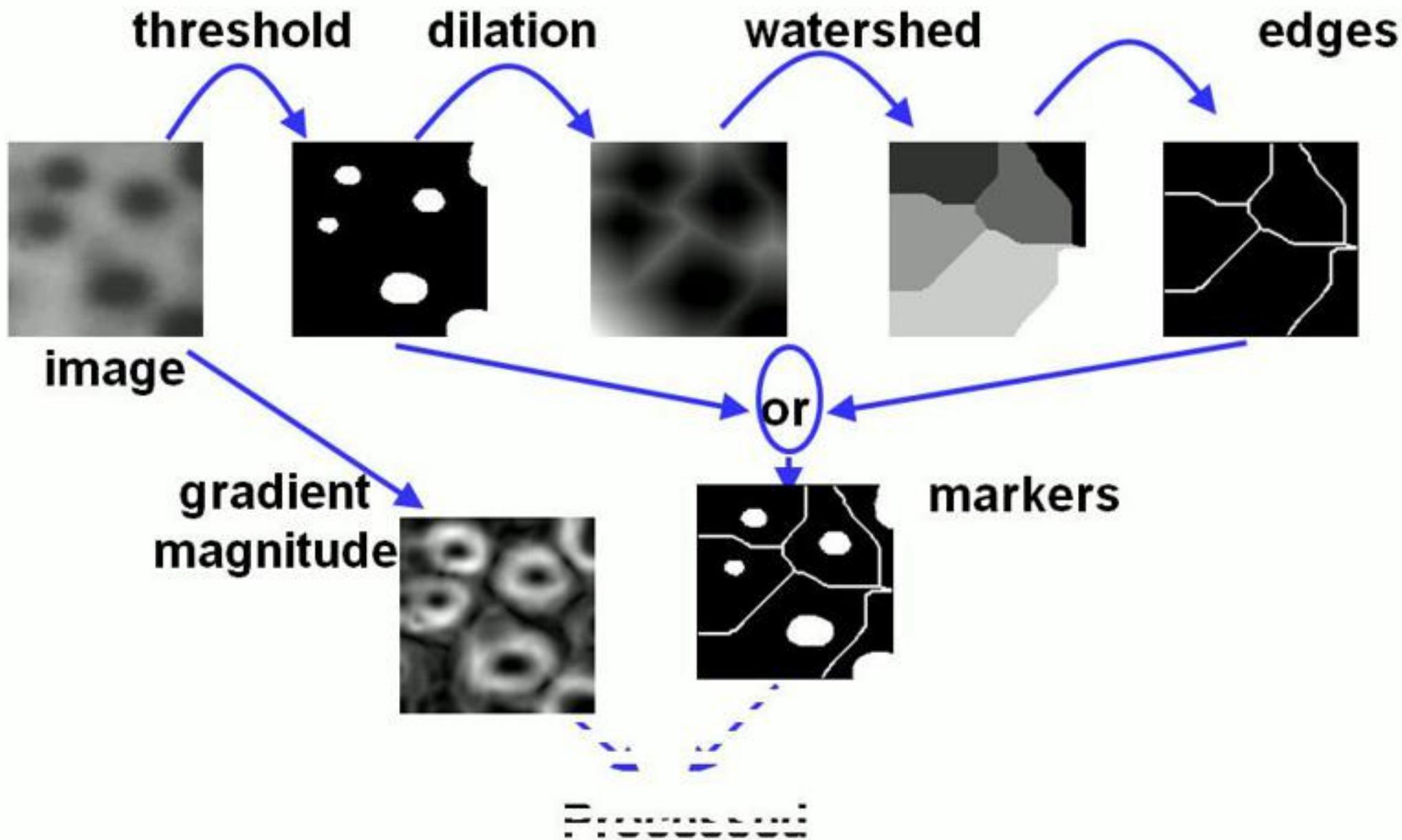
# Marker-based watershed



# Marker-based watershed



# Marker-based watershed



# Background subtraction



## Example



A. The binary image.



B. The image after closing with a disc of radius 3 pixels.

The whole is equal to the sum of its parts

-Euclid

Segmentation of an image is its partitioning into a set of N connected regions  $R(n)$ ,  $n = 0, 1, \dots, N - 1$ .

A region can be defined by its interior part as-

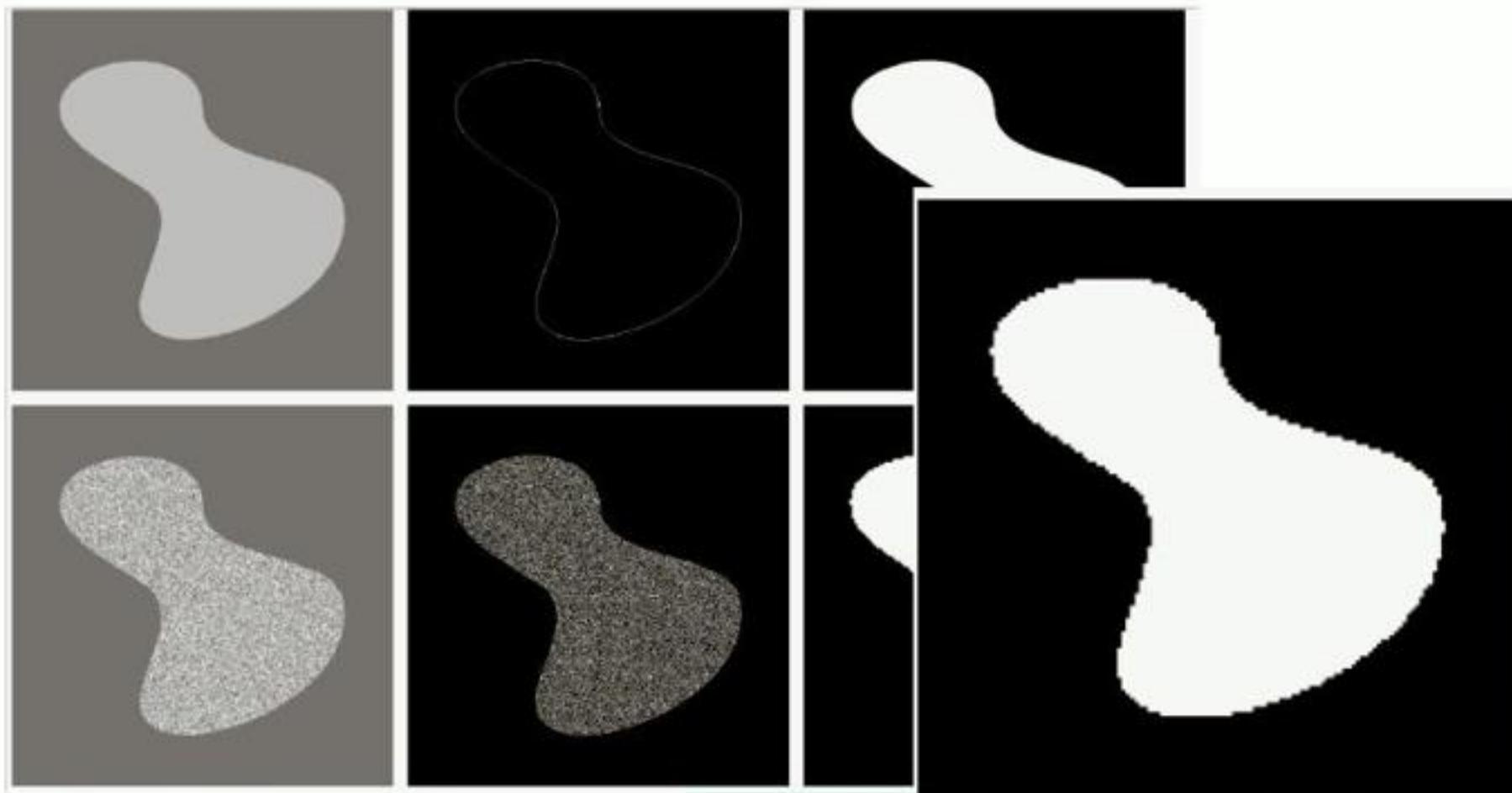
$$P(R) = \begin{cases} 1, & \text{if } f(R) \in H \\ 0, & \text{otherwise} \end{cases}$$

The process will satisfy these points-

1. The sum of all the regions is exactly equal to the image.
2. A pixel of the image belongs to only one of the regions.
3. The properties of a region are same for all its pixels.
4. The properties of adjacent regions must be different.

- Segmentation of an image is its partition into disjoint regions.
- Various segmentation methods are based on finding the interior of the region or its border.
- The border of a region can be found by edge detection.
- The interior of the region is determined by the distinct properties of the pixels comprising the region.
- It can be achieved by thresholding, region-based methods, and morphological watershed are typical segmentation algorithms

# Segmentation



**FIGURE 10.1** (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number of small edges that are connected to the original boundary, making it difficult to find a unique boundary using only edge information. (f) Result of segmentation based on region properties.

Accurate segmentation of objects in an image greatly facilitates further analysis of these objects in many applications.

## Example:

- Automation of counting objects of a certain type.
- Measure geometric properties (e.g., area, perimeter) of objects in the image.
- Study properties of an individual object (intensity, texture, etc.)

Segmentation algorithms generally are based on one of two basis properties of pixel intensity values- **discontinuity and similarity**.

Discontinuities are sudden changes (differences) in intensity:

- Point detection
- Line detection
- Edge detection
- Object boundary

Similarity may be due to pixel intensity, color or texture for finding the interior of the region-

- Threshholding
- Region growing,
- Region slitting and merging

The most common way to look for discontinuities is to scan a small mask over the image.

**The mask determines the kind of discontinuity.**

The sum of products R

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

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

The most common way to look for discontinuities is to scan a small mask over the image.

**The mask determines the kind of discontinuity.**

The sum of products R

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

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

A point has been detected at the location on which the mark is centered if

$$|R| \geq T$$

where  $T$  : non negative threshold

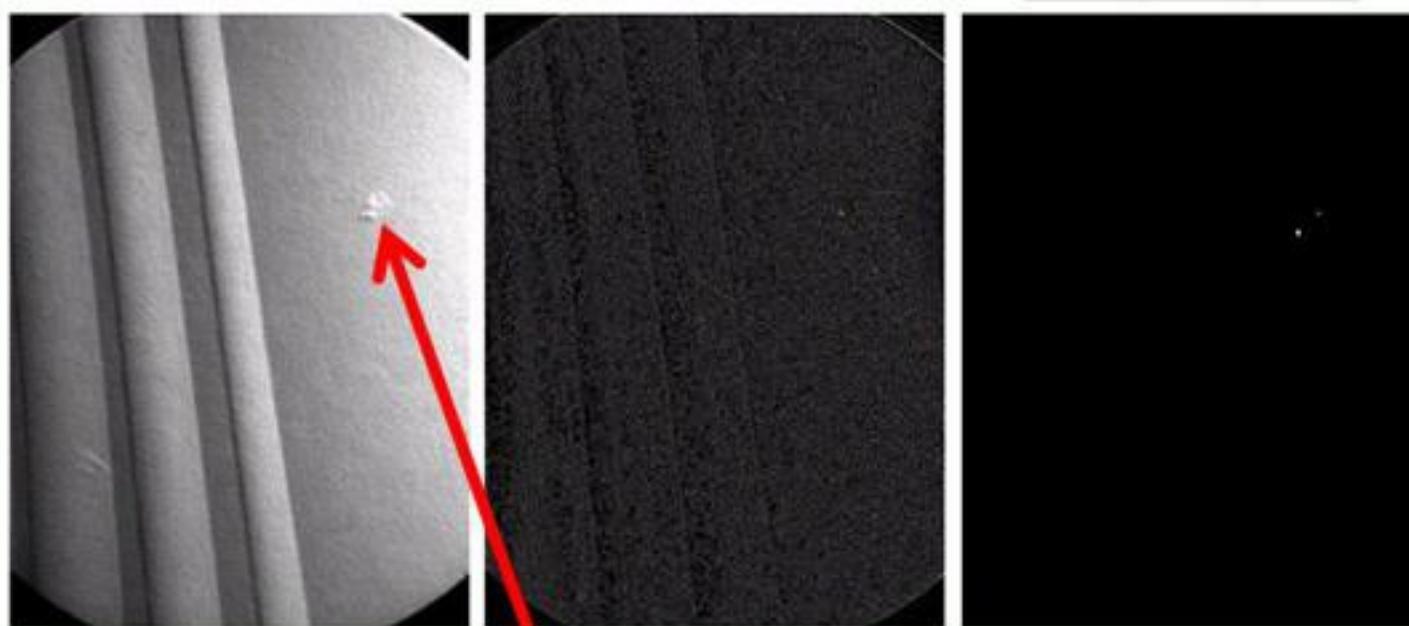
$$R = w_1z_1 + w_2z_2 + \dots + w_9z_9 = \sum_{i=1}^9 w_i z_i$$

-1	-1	-1
-1	8	-1
-1	-1	-1

- Note that the mask is the same as the Laplacian Operation
- The only differences that are considered of interest are those large enough (as determined by  $T$ ) to be considered isolated points.

# Example

-1	-1	-1
-1	8	-1
-1	-1	-1



a  
b c d

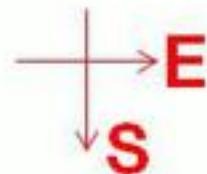
**FIGURE 10.2**

- (a) Point detection mask.
- (b) X-ray image of a turbine blade with a porosity.
- (c) Result of point detection.
- (d) Result of using Eq. (10.1-2). (Original image courtesy of X-TEK Systems Ltd.)

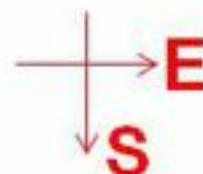
$$|R| \geq T$$

where  $T$ : a nonnegative threshold

- The masks for line detection at four directions (**E-W, NW-SE, N-S, NE-SW**).
- Horizontal mask will result with max response when a line passed through the middle row of the mask with a constant background.
- the similar idea is used with vertical and diagonal masks.
- Locate the one pixel wide line in an image.
- For digital images the only three point straight lines.



- The masks for line detection at four directions (**E-W, NW-SE, N-S, NE-SW**).
- Horizontal mask will result with max response when a line passed through the middle row of the mask with a constant background.
- the similar idea is used with vertical and diagonal masks.
- Locate the one pixel wide line in an image.
- For digital images the only three point straight lines.



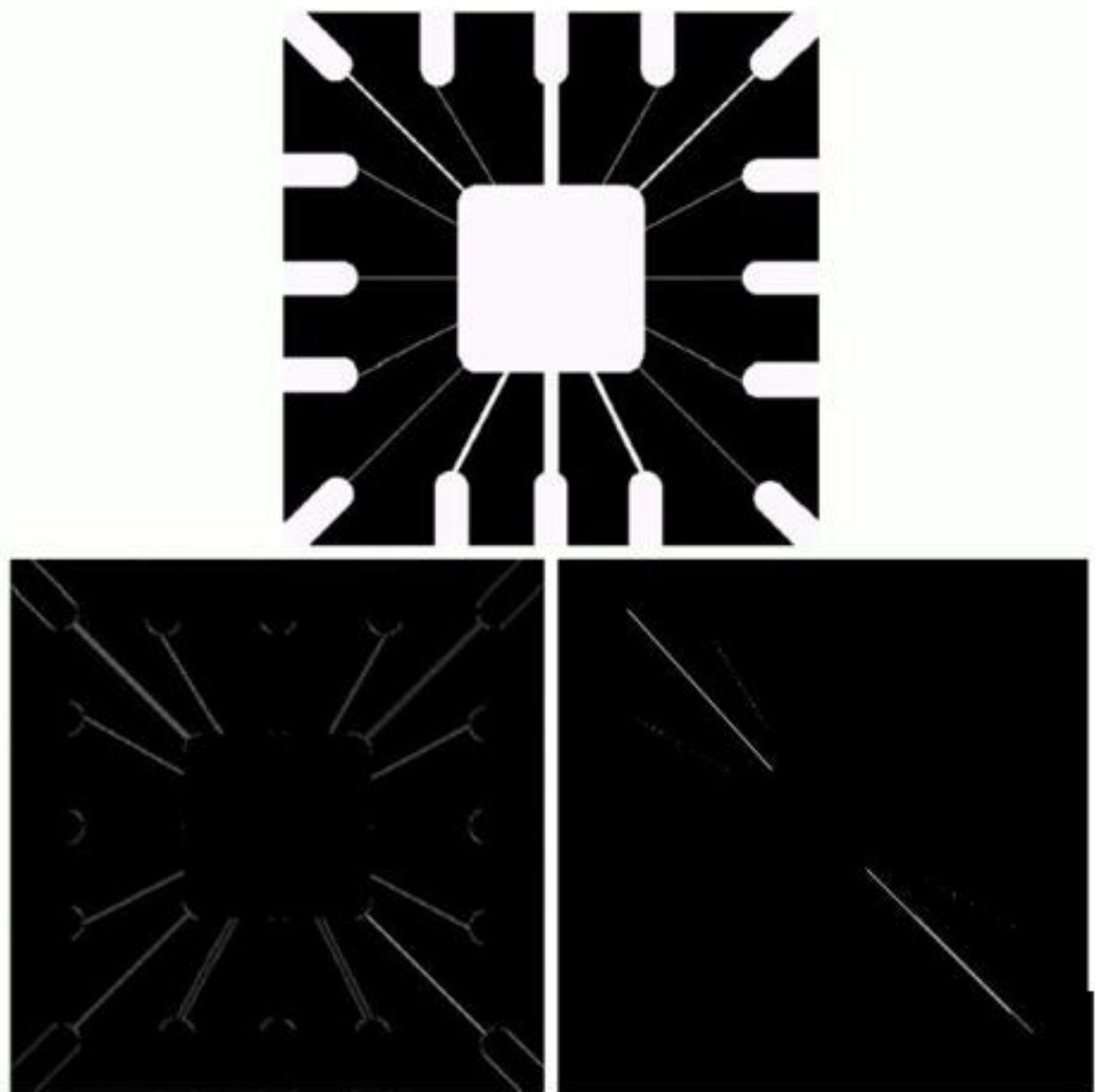
**FIGURE 10.3** Line  
masks.

-1	-1	-1	-1	-1	2	-1	2	-1	2	-1	-1
2	2	2	-1	2	-1	-1	2	-1	-1	2	-1
-1	-1	-1	2	-1	-1	-1	2	-1	-1	-1	2

- Apply every masks on the image
- let  $R_1, R_2, R_3, R_4$  denotes the response of the horizontal,  $+45^\circ$ , vertical and  $-45^\circ$  masks, respectively.
- if, at a certain point in the image
$$|R_i| > |R_j|,$$
- for all  $j$  and  $i$ , that point is said to be more likely associated with a line in the direction of mask  $i$ .

- Alternatively, if we are interested in detecting all lines in an image in the direction defined by a given mask, we simply run the mask through the image and threshold the absolute value of the result.
- The points that are left are the **strongest responses**, which, for lines **one pixel thick**, correspond closest to the direction defined by the mask.

# Line Detection



a  
b | c

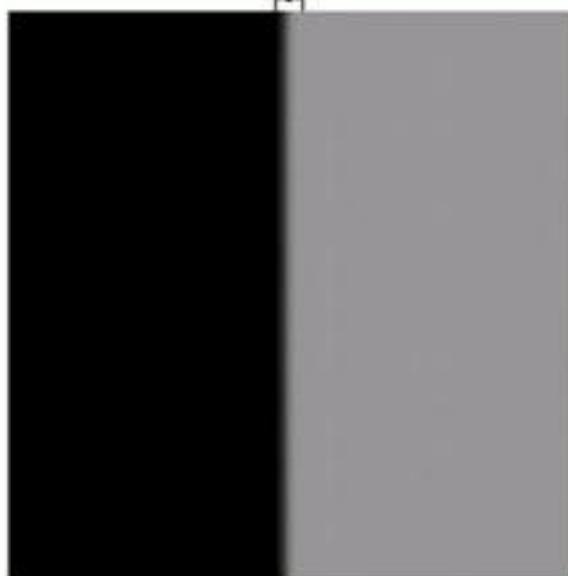
**FIGURE 10.4**  
Illustration of line detection.  
(a) Binary wire-bond mask.  
(b) Absolute value of result after processing with  $-45^\circ$  line detector.  
(c) Result of thresholding image (b).

# First and second derivative

a b

**FIGURE 10.6**

- (a) Two regions separated by a vertical edge.  
(b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.

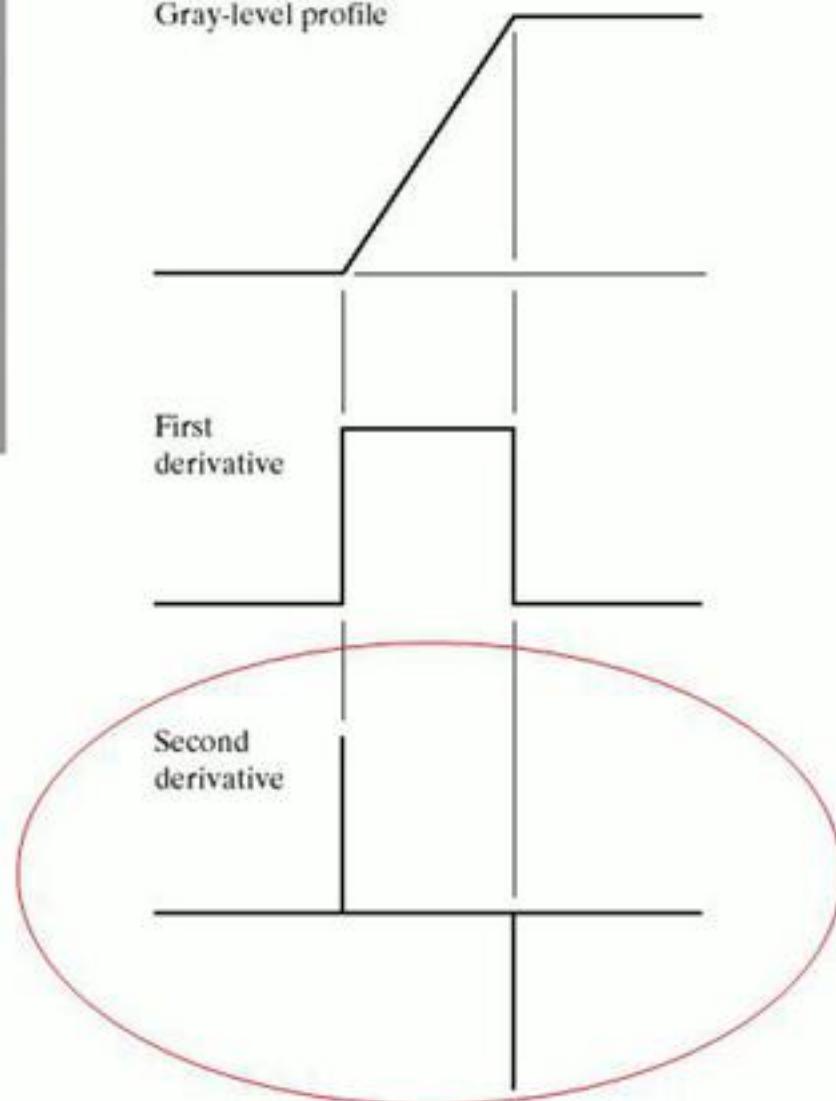


Gray-level profile

First derivative

Second derivative

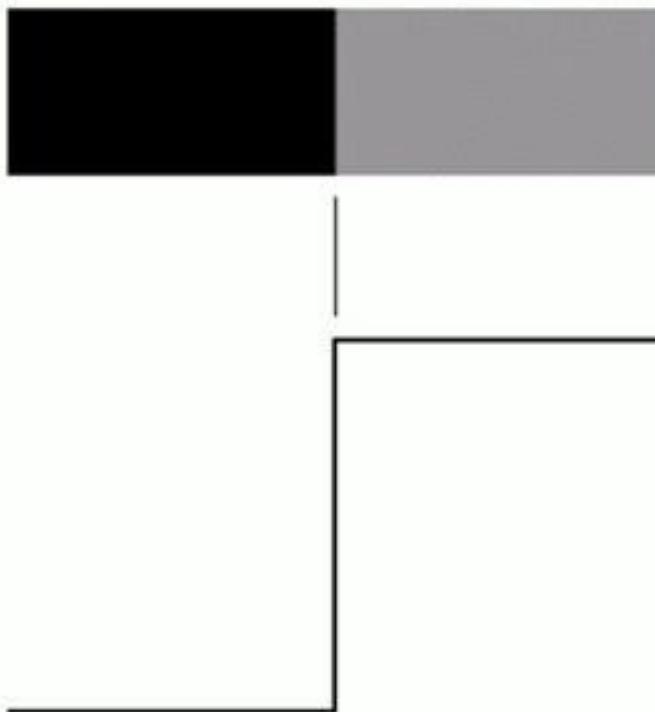
The signs of the derivatives would be reversed for an edge that transitions from light to dark



- produces two values for every edge in an image (an undesirable feature)
- an imaginary straight line joining the extreme positive and negative values of the second derivative would cross zero near the midpoint of the edge. (zero-crossing property)

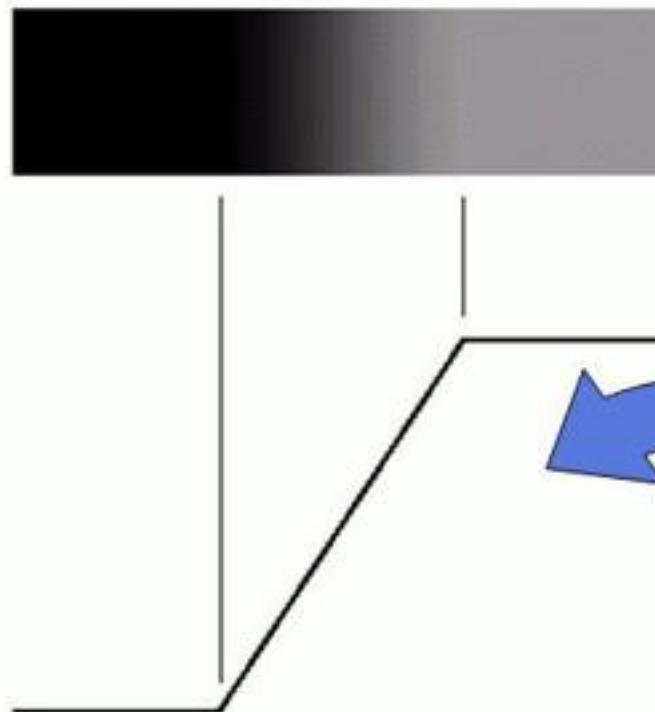
# Ideal and Ramp Edge

Model of an ideal digital edge



Gray-level profile  
of a horizontal line  
through the image

Model of a ramp digital edge



Gray-level profile  
of a horizontal line  
through the image

a b

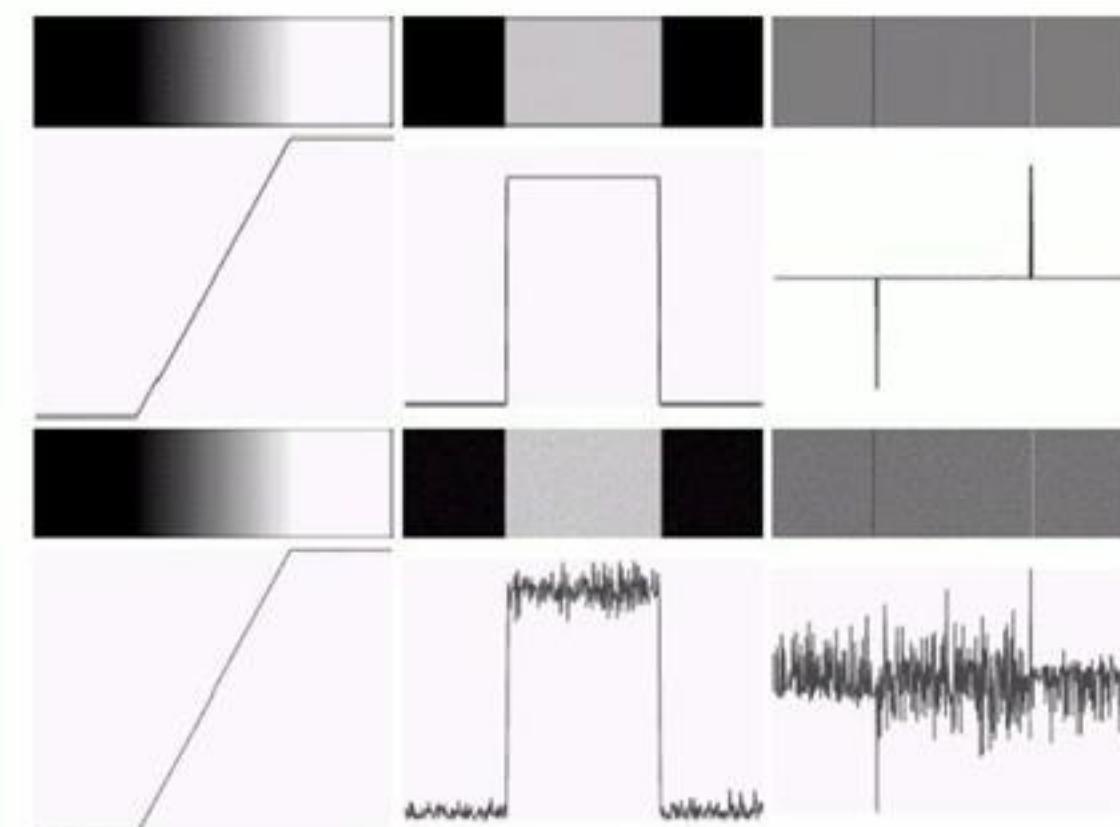
**FIGURE 10.5**

(a) Model of an ideal digital edge.  
(b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

**because of optics,  
sampling, image  
acquisition imperfection**

# Noise in Images

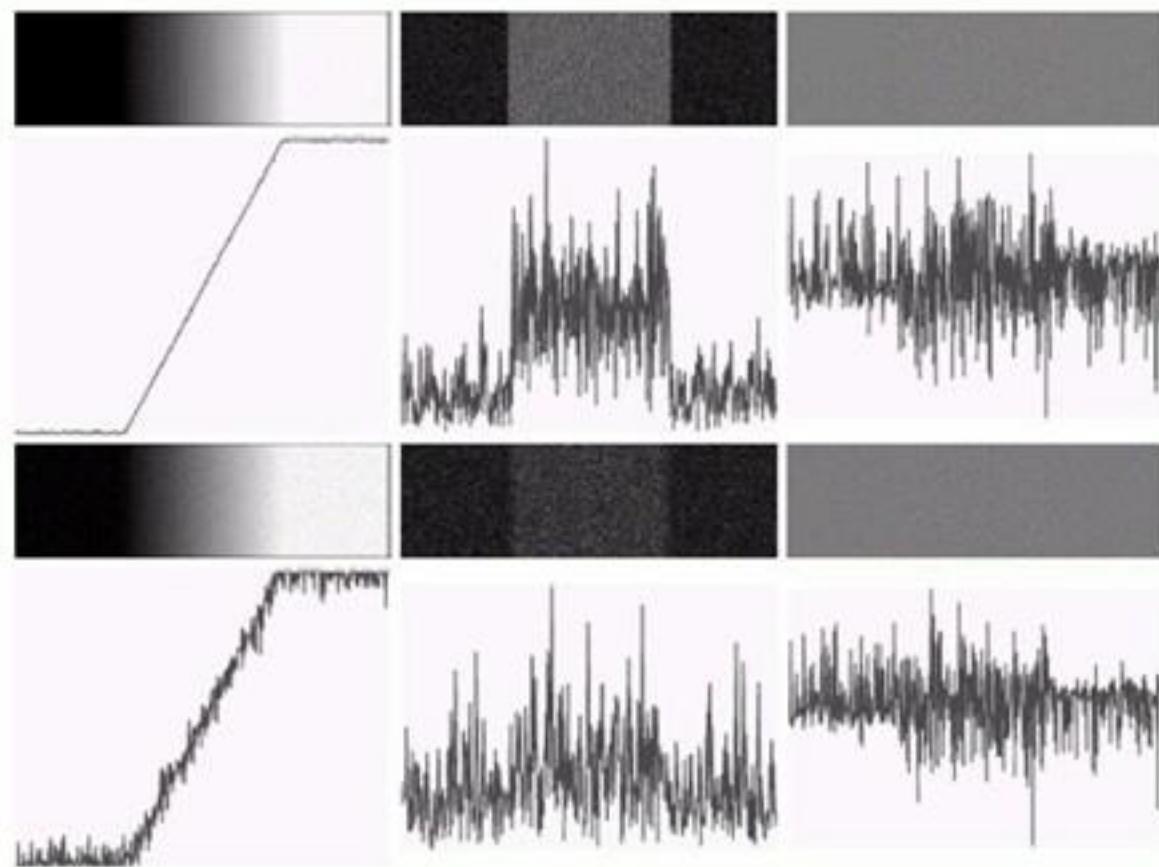
- First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and variance = 0.0, 0.1, respectively.
- Second column: first-derivative images and gray-level profiles.
- Third column : second-derivative images and gray-level profiles.



a  
b  
c  
d

**FIGURE 10.7** First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and  $\sigma = 0.0, 0.1, 1.0$ , and  $10.0$ , respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles.

# Noise in Images



- First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and variance = 1.0 and 10.0, respectively.
- Second column: first-derivative images and gray-level profiles.
- Third column : second-derivative images and gray-level profiles.

a  
b  
c  
d

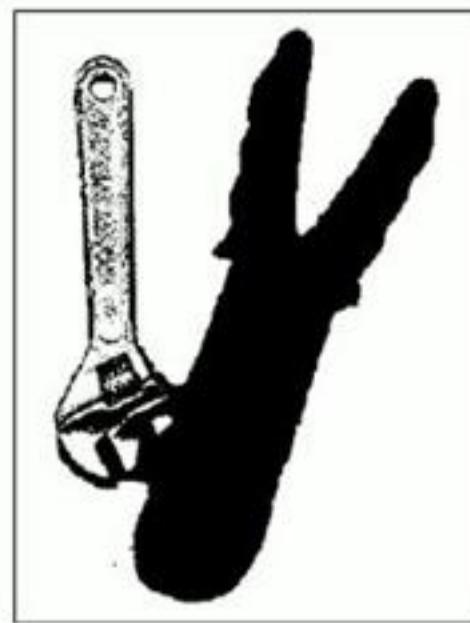
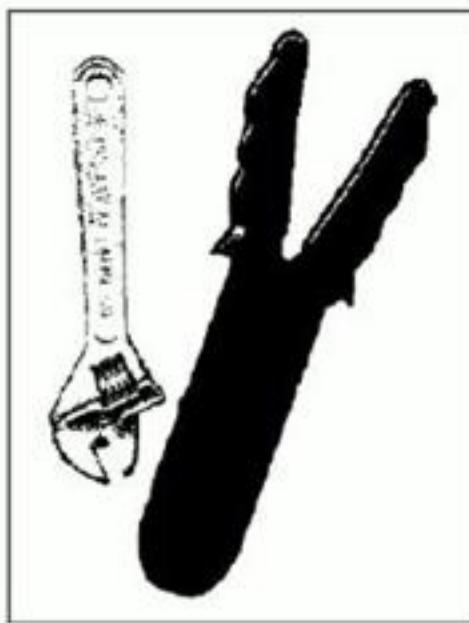
**FIGURE 10.7** First column: images and gray-level profiles of a ramp edge corrupted by random Gaussian noise of mean 0 and  $\sigma = 0.0, 0.1, 1.0$ , and 10.0, respectively. Second column: first-derivative images and gray-level profiles. Third column: second-derivative images and gray-level profiles.

- fairly little noise can have such a significant impact on the two key derivatives used for edge detection in images
- image smoothing should be serious consideration prior to the use of derivatives in applications where noise is likely to be present.

# Pixel intensity Similarity: Thresholding

All the image segmentation methods assume that:

1. the intensity values are different in different regions
2. within each region, which represents the corresponding object in a scene, the intensity values are similar.



# Thresholding

- The purpose is to subdivide an image into meaningful **non-overlapping regions**, which would be used for further analysis.
- It is hoped that the regions obtained correspond to the physical parts or objects of a scene (3-D) represented by the image (2-D).

In general, autonomous segmentation is one of the most difficult tasks in digital image processing.

Image with dark background and  
a light object

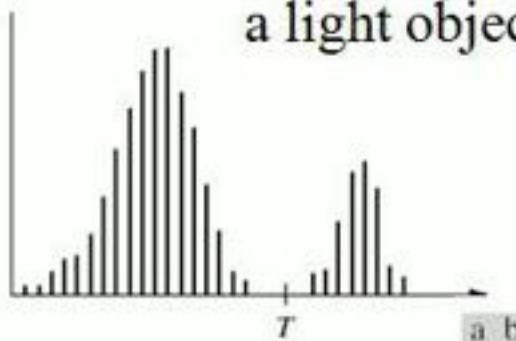


Image with dark background and  
two light objects

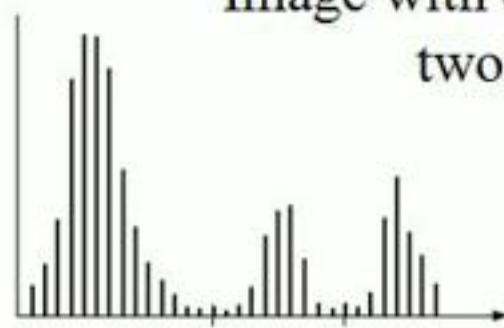


FIGURE 10.26 (a) Gray-level histograms that can be partitioned by (a) a single threshold, and (b) multiple thresholds.

# Similarity based Segmentation: Thresholding

- The purpose is to subdivide an image into meaningful non-overlapping regions.
- It is hoped that the regions obtained correspond to the physical parts or objects of a scene (3-D) represented by the image (2-D).

In general, autonomous segmentation is one of the most difficult tasks in digital image processing.

Image with dark background and a light object

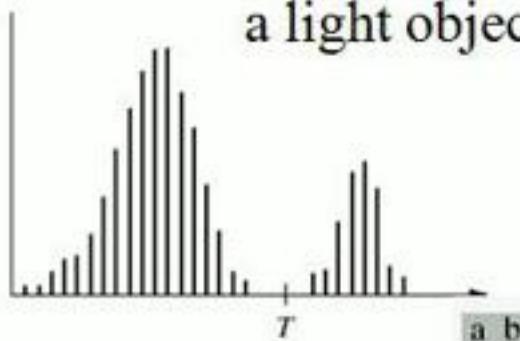


Image with dark background and two light objects

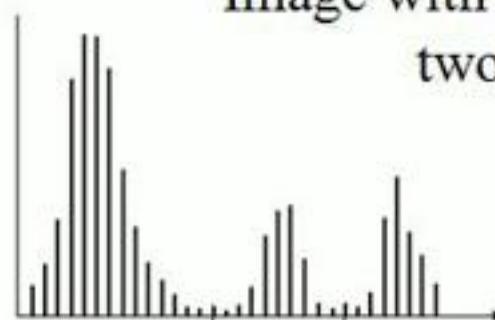
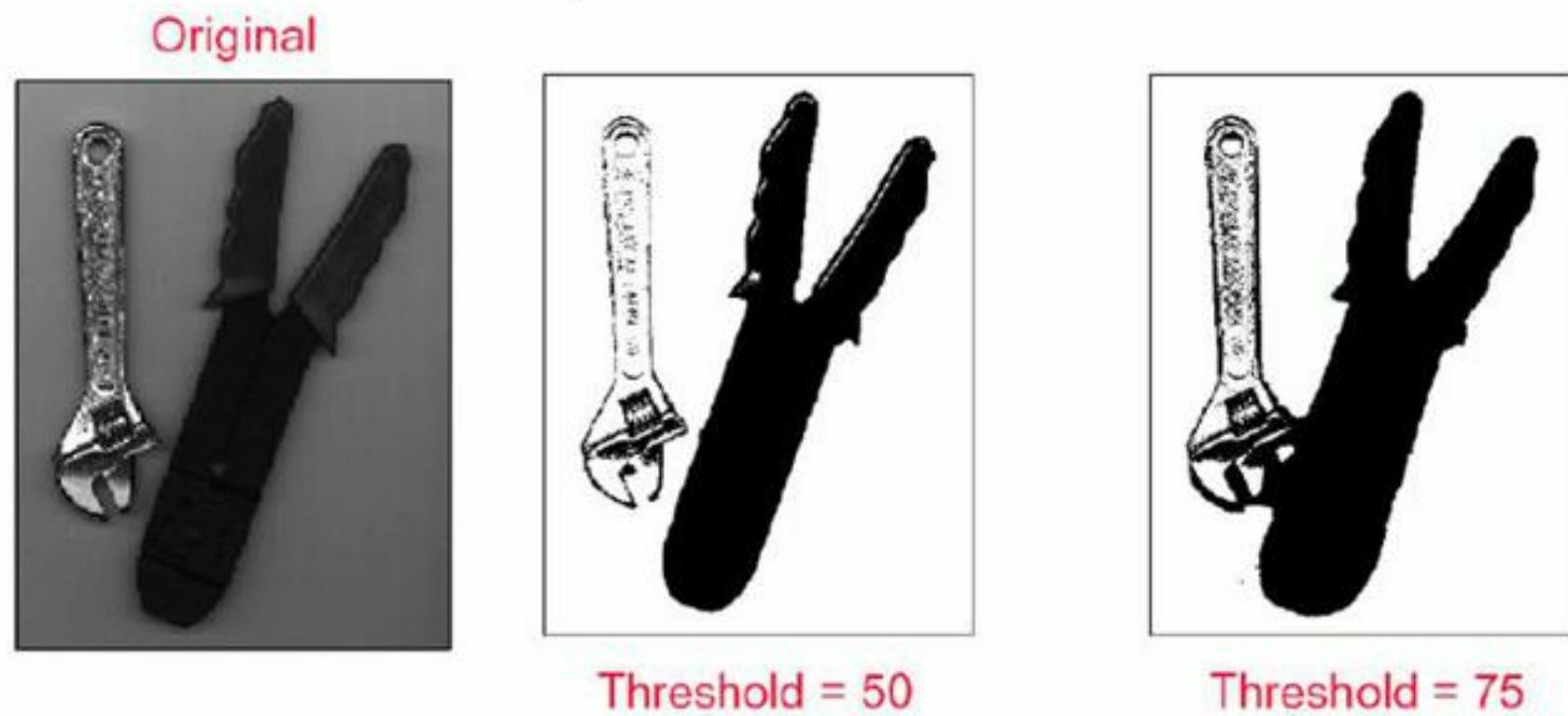


FIGURE 10.26 (a) Gray-level histograms that can be partitioned by (a) a single threshold, and (b) multiple thresholds.

# Similarity based Segmentation: Thresholding

All the image segmentation based on thresholding method assume that:

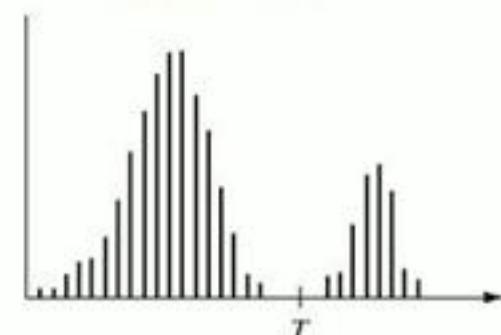
1. The intensity values are different in different regions
2. Within each region, which represents the object, the intensity values are similar.



Threshold value  $T$  depends on:

- only  $f(x, y) : \text{gray-level values} > \text{Global threshold}$
- It does assume a bimodal histogram distribution
- Easily use when object and background are separated

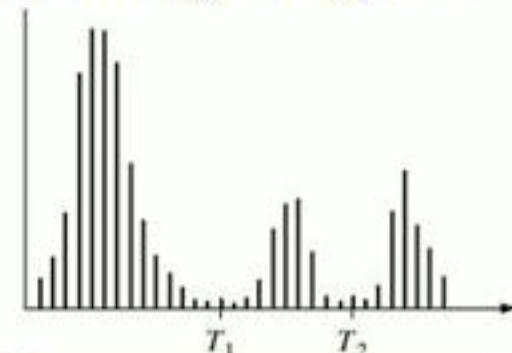
Image with dark background and a light object



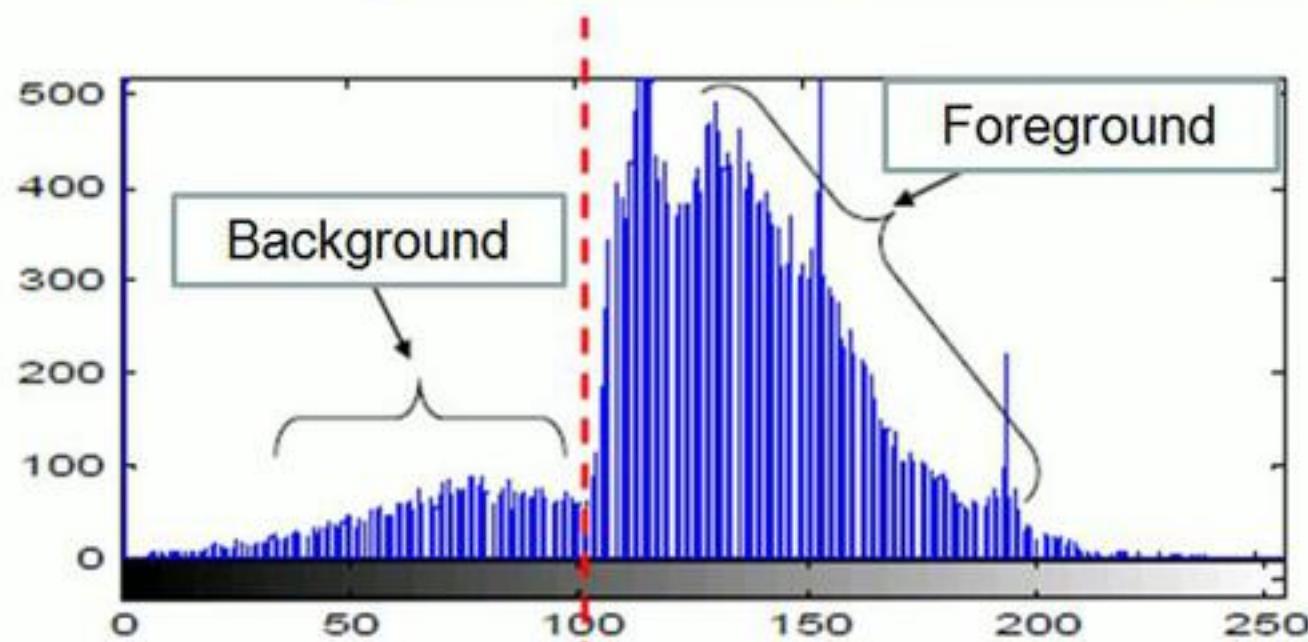
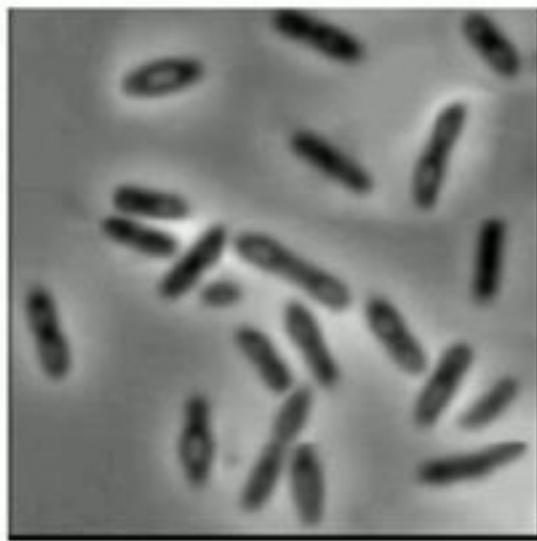
A point  $f(x, y)$  belongs to:

- to an object class if  $T_1 < f(x,y) < T_2$
- to another object class if  $f(x,y) > T_2$
- to background if  $f(x,y) < T_1$

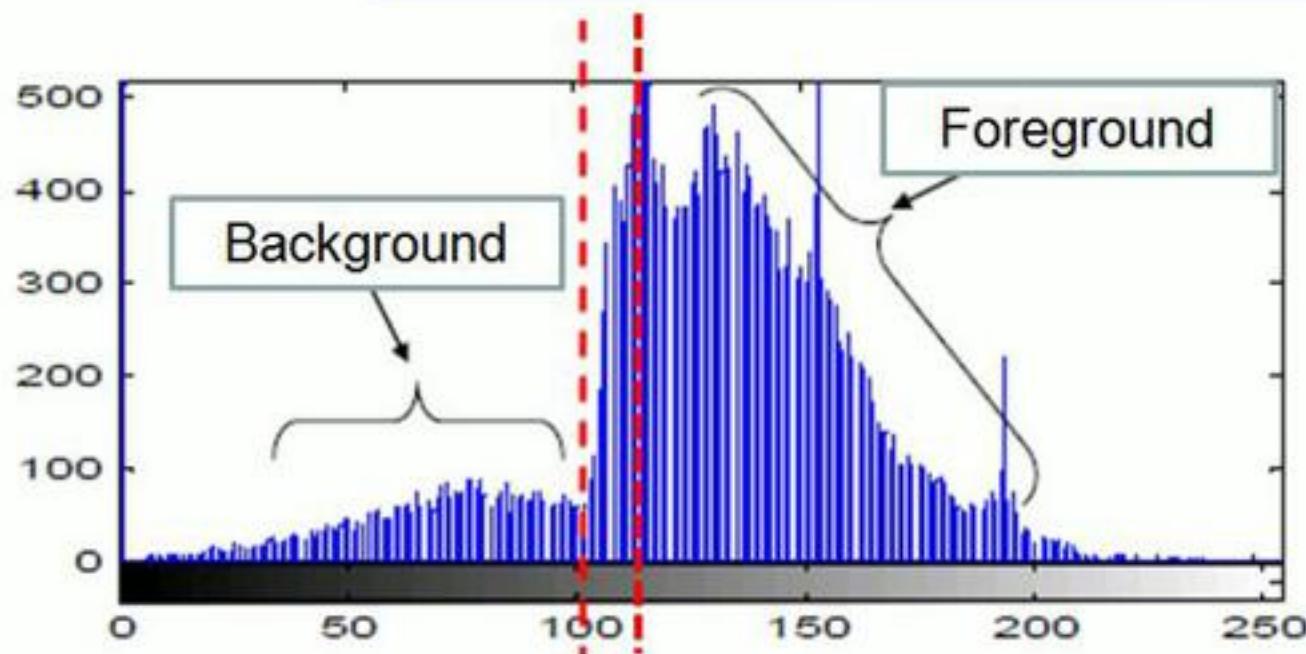
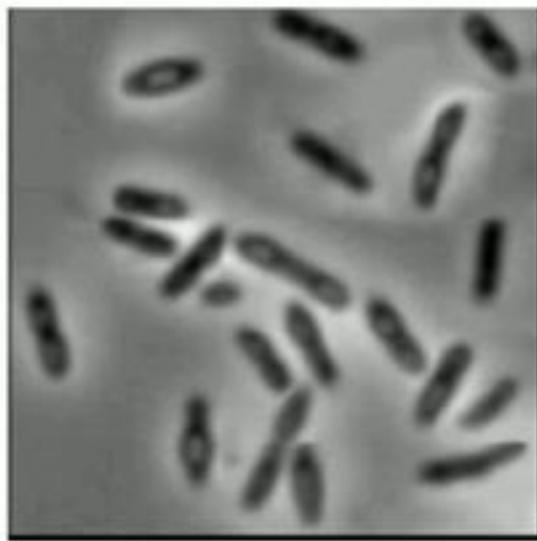
Image with dark background and two light objects



# Global Thresholding

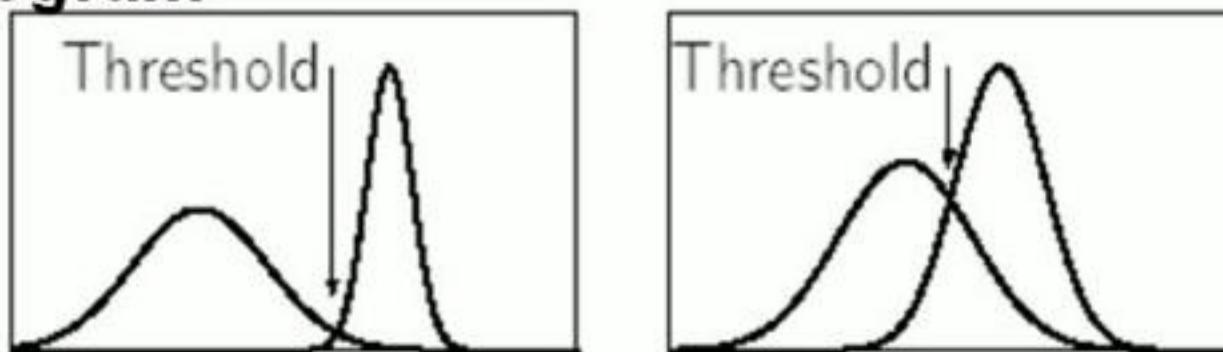


# Global Thresholding



# Automatic Threshold Level Selection

## Bimodal histogram



The major problem of intensity thresholding is to find a good threshold level. It is based on visual inspection of histogram

1. Select an initial estimate for  $T$ .
2. Segment the image using  $T$ . This will produce two groups of pixels:  $G_1$  consisting of all pixels with gray level values  $> T$  and  $G_2$  consisting of pixels with gray level values  $\leq T$
3. Compute the average gray level values  $m_1$  and  $m_2$  for the pixels in regions  $G_1$  and  $G_2$
4. Compute a new threshold value  $T = 0.5 (m_1 + m_2)$
5. Repeat steps 2 through 4 until the difference in  $T$  in successive iterations is smaller than a predefined parameter  $T_i$ .



**FIGURE 10.29**

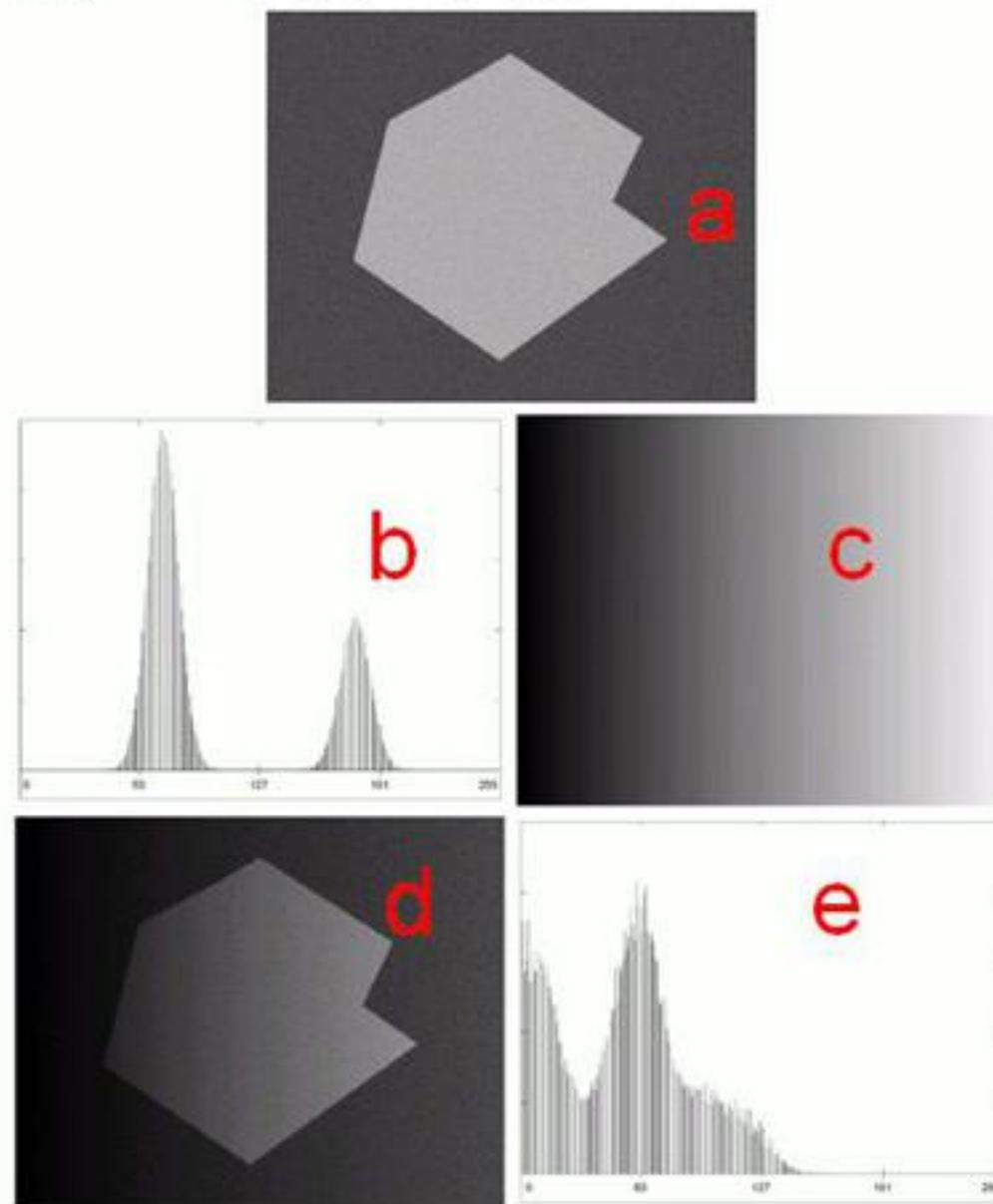
(a) Original image. (b) Image histogram.  
(c) Result of segmentation with the threshold estimated by iteration.  
(Original courtesy of the National Institute of Standards and Technology.)

Remark: the clear valley of the histogram and the effective of the segmentation between object and background

$T_0 = 0$   
**3 iterations with result  $T = 125$**

Image formation:

$$f(x,y) = i(x,y) \cdot r(x,y)$$

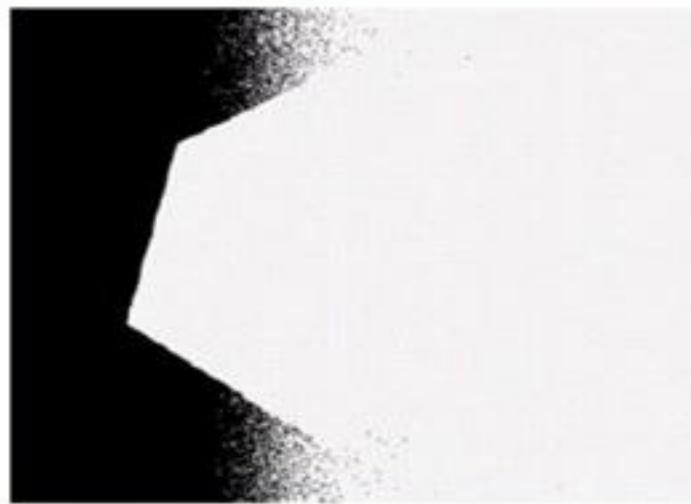


# Global Thresholding

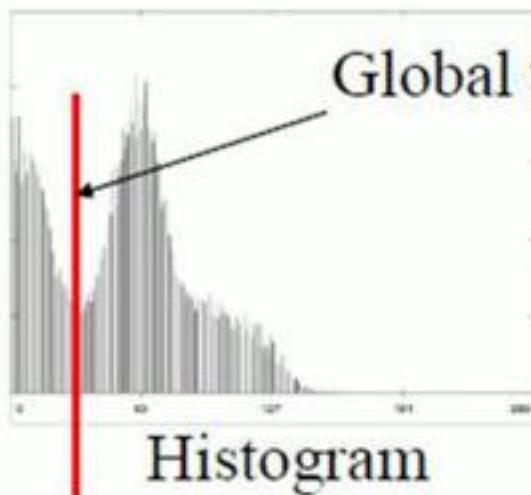
Difficult to segment

- a) computer generated reflectance function
- b) histogram of reflectance function
- c) computer generated illumination function (poor)
- d) product of a & c
- e) histogram of product image

# Global Thresholding



Global thresholding  
result



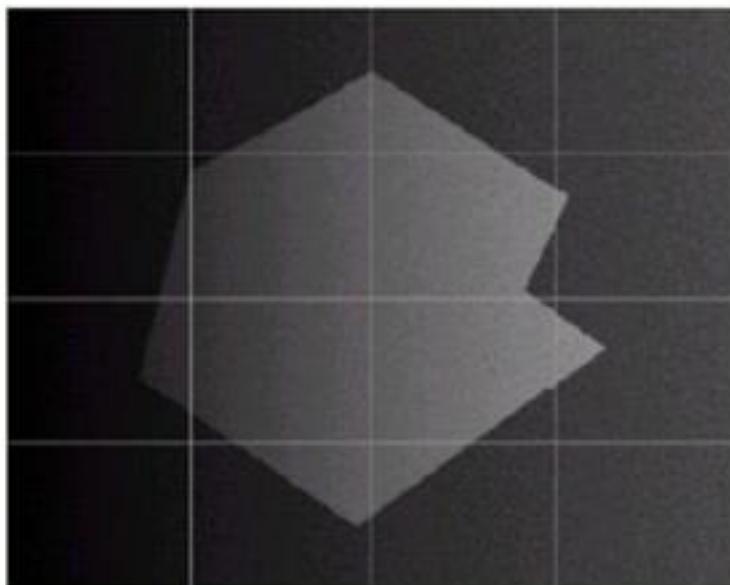
Global thresholding of nonuniform illumination image can cause huge errors!

Histogram

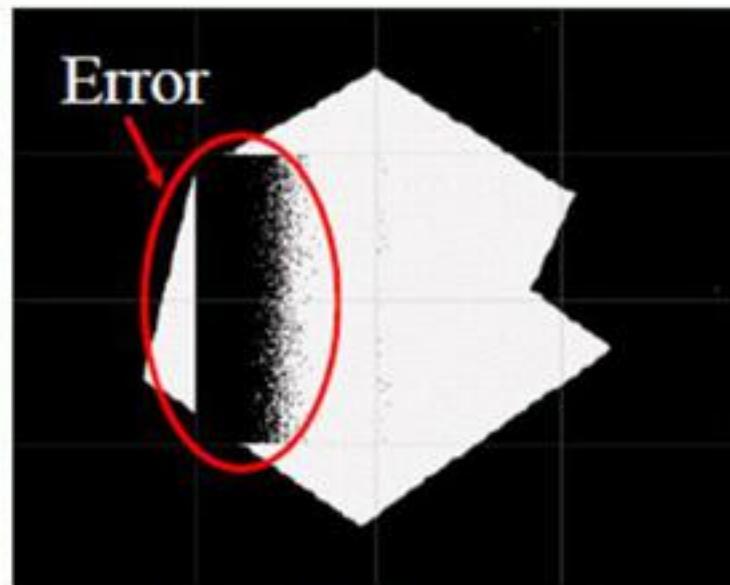
# Nonuniform Illumination and Local Thresholding

## Local thresholding:

1. Divide an image into subimages.
2. Threshold each subimage independently
  - 2.1 Compute histogram of each subimage and select a suitable threshold value for each subimage
  - 2.2 threshold each subimage using a threshold value in 2.1
  - 2.3 Combine all local thresholding results



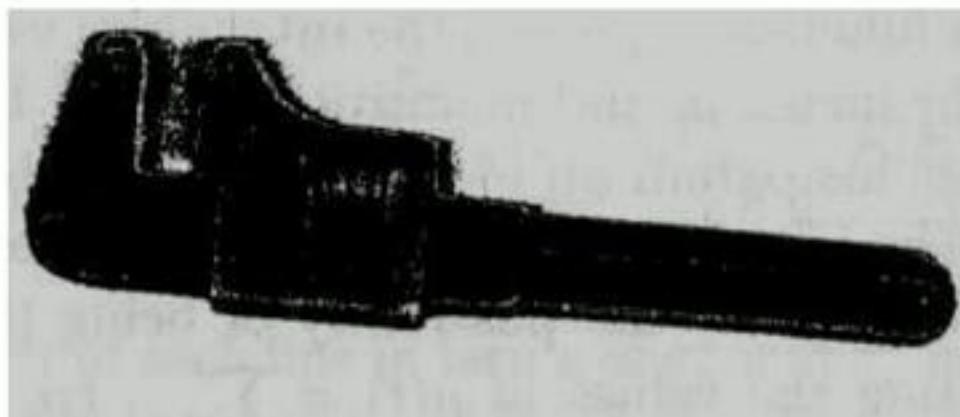
16 subimages



Result of local thresholding

## Otsu's Method (1979)

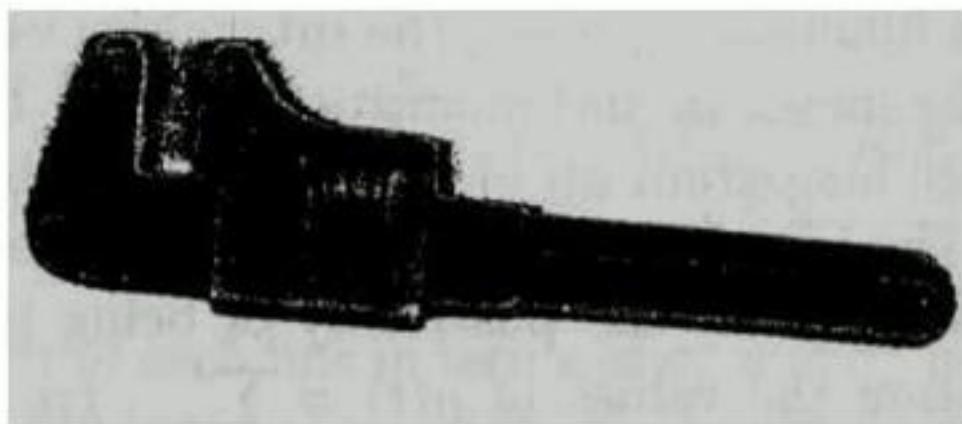
- Segmentation is based on “region homogeneity”.
  - Region homogeneity can be measured using variance.



- Otsu's thresholding chooses the threshold to minimize the intra-class variance of the thresholded black and white pixels.

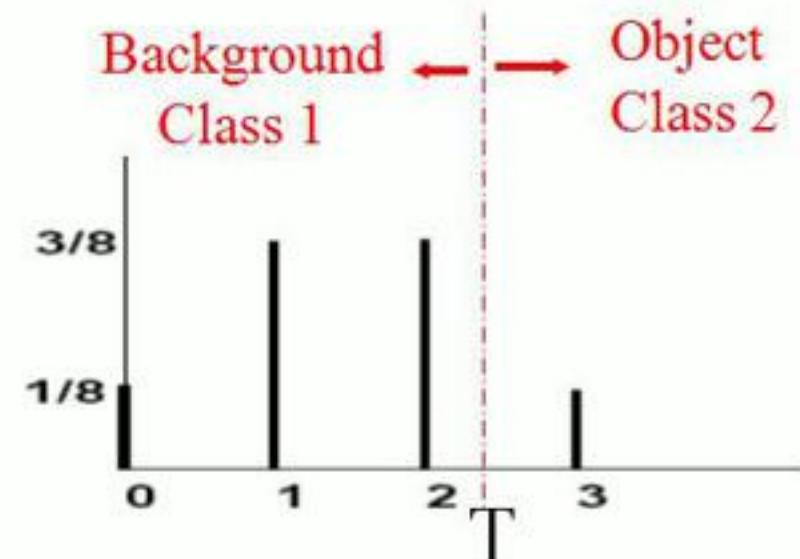
It is based on a very simple idea:

- Find the threshold that minimizes the weighted within-class variance.
- This turns out to be the same as maximizing the between-class variance.
- Operates directly on the gray level histogram so it's fast.



## Otsu's Method

- Consider an image with  $L$  gray levels and its normalized histogram.
- Assuming that we have set the threshold at  $T$ , **the normalized fraction of pixels** that will be classified as background and object will be:

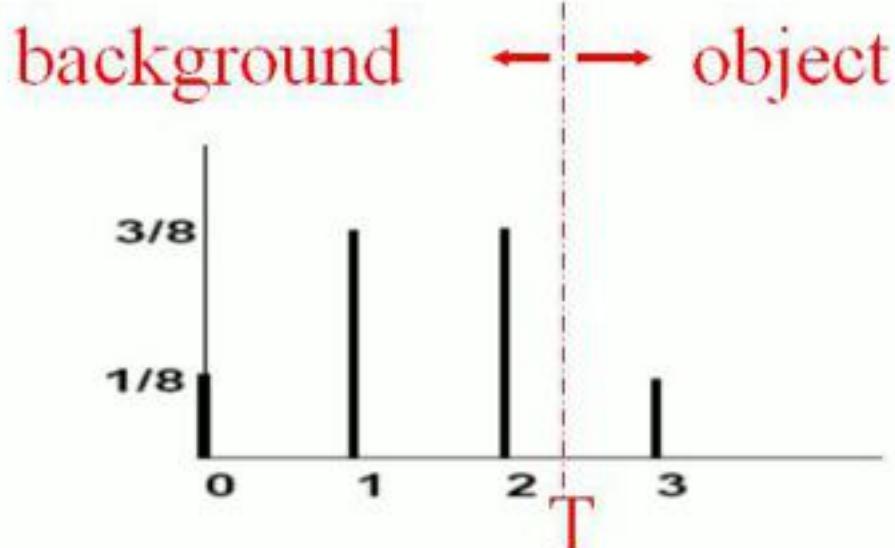


$$P_1(T) = \sum_{i=0}^T p(i) \quad \& \quad P_2(T) = \sum_{i=T+1}^{L-1} p(i)$$

$p(i)$  is the normalized frequency of intensity  $i$ .

$$P_1(T) + P_2(T) = 1$$

The individual class variances are defined as:



$$\sigma_1^2(t) = \sum_{i=1}^t [i - m_1(t)]^2 \frac{P(i)}{P_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^L [i - m_2(t)]^2 \frac{P(i)}{P_2(t)}$$

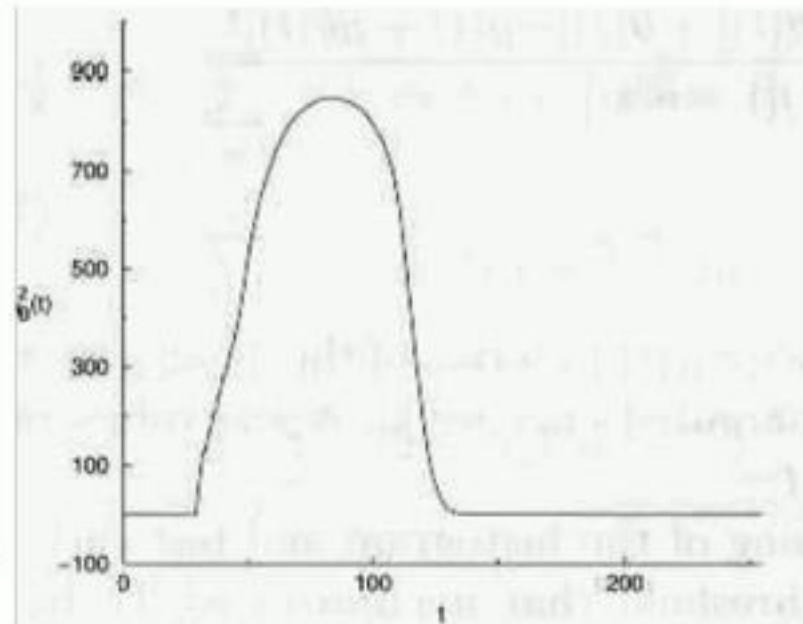
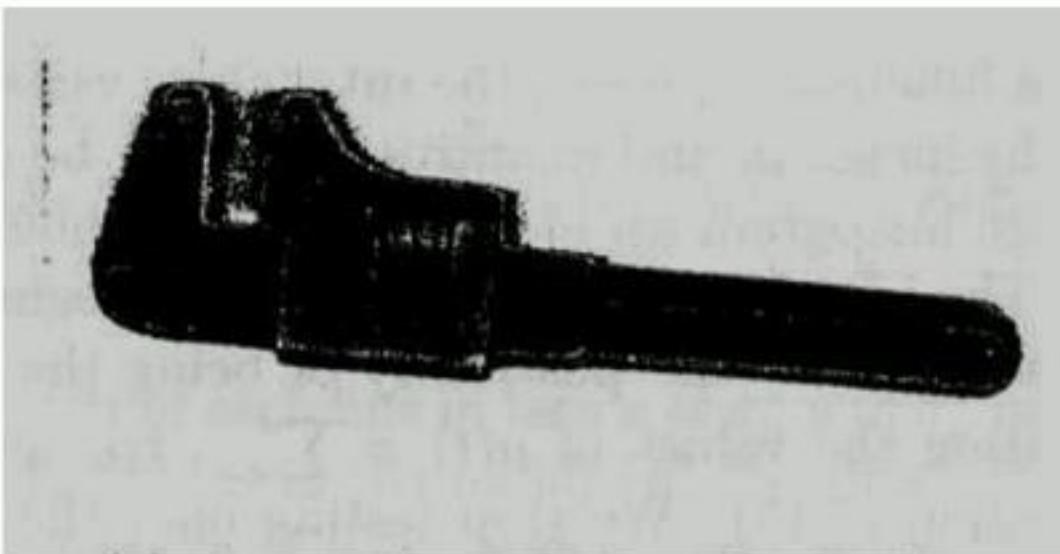
And the class means are given by:

$$m_1(t) = \sum_{i=0}^t \frac{iP(i)}{P_1(t)} \quad m_2(t) = \sum_{i=t+1}^{L-1} \frac{iP(i)}{P_2(t)}$$

$$P_1 \mu_1 + P_2 \mu_2 = m_G \Rightarrow m_G = \sum_{i=0}^{L-1} i p(i)$$

## Procedure: Otsu's Method

Start from the beginning of the histogram and test each gray level value for the possibility of being the threshold 't' that maximizes  $\sigma_B^2(t)$



(a)  $\sigma_B(t)$  versus  $t$ .

Find the segmented image of 4x4 image by Otsu algorithm

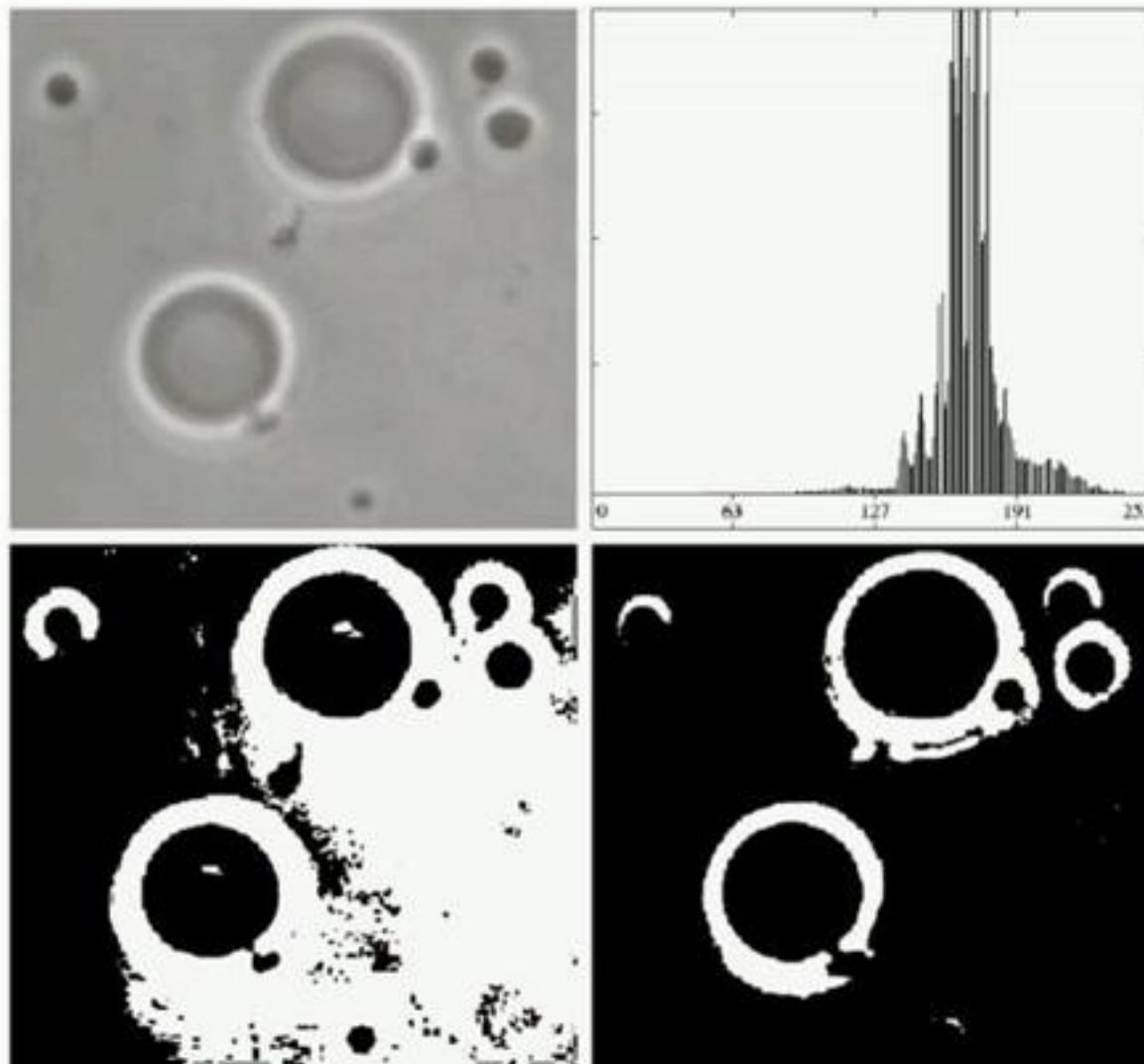
2	7	6	6
5	6	5	5
6	5	5	6
7	6	4	5

Gray level k	0	1	2	3	4	5	6	7
Between class variance	0	0	0.7594	0.7594	0.8058	0.7656	0.3772	0

## Drawbacks:

- The method assumes that the histogram of the image is bimodal (i.e., two classes).
- The method breaks down when the two classes are very unequal (i.e., the classes have very different sizes)
  - In this case,  $\sigma_B^2(t)$  may have two maxima.
  - The selected maximum is not necessarily the global one.
- The method does not work well with variable illumination.

# Application: Otsu's Method

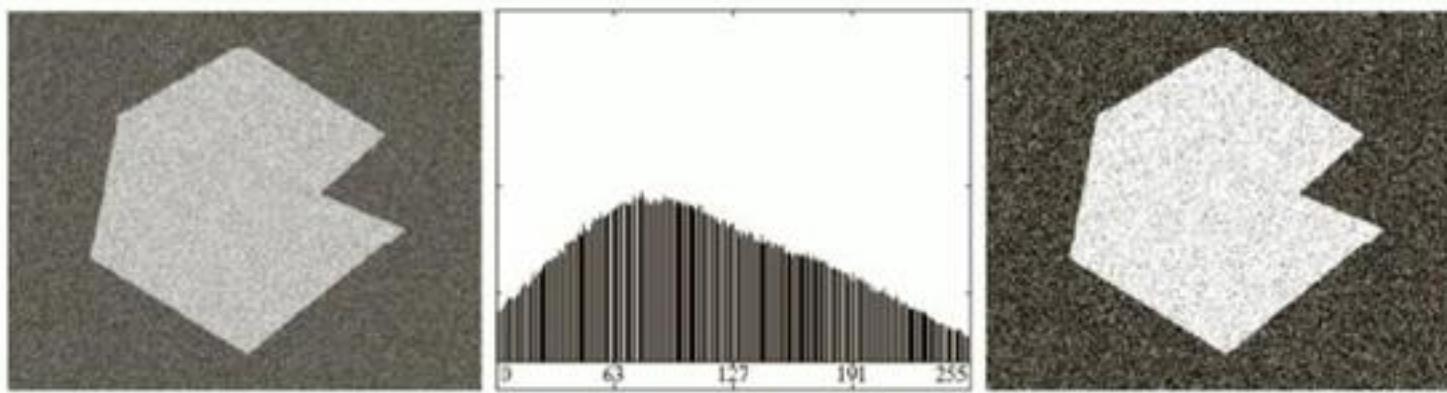


a  
b  
c  
d

**FIGURE 10.39**

(a) Original image.  
(b) Histogram (high peaks were clipped to highlight details in the lower values).  
(c) Segmentation result using the basic global algorithm from Section 10.3.2.  
(d) Result obtained using Otsu's method. (Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

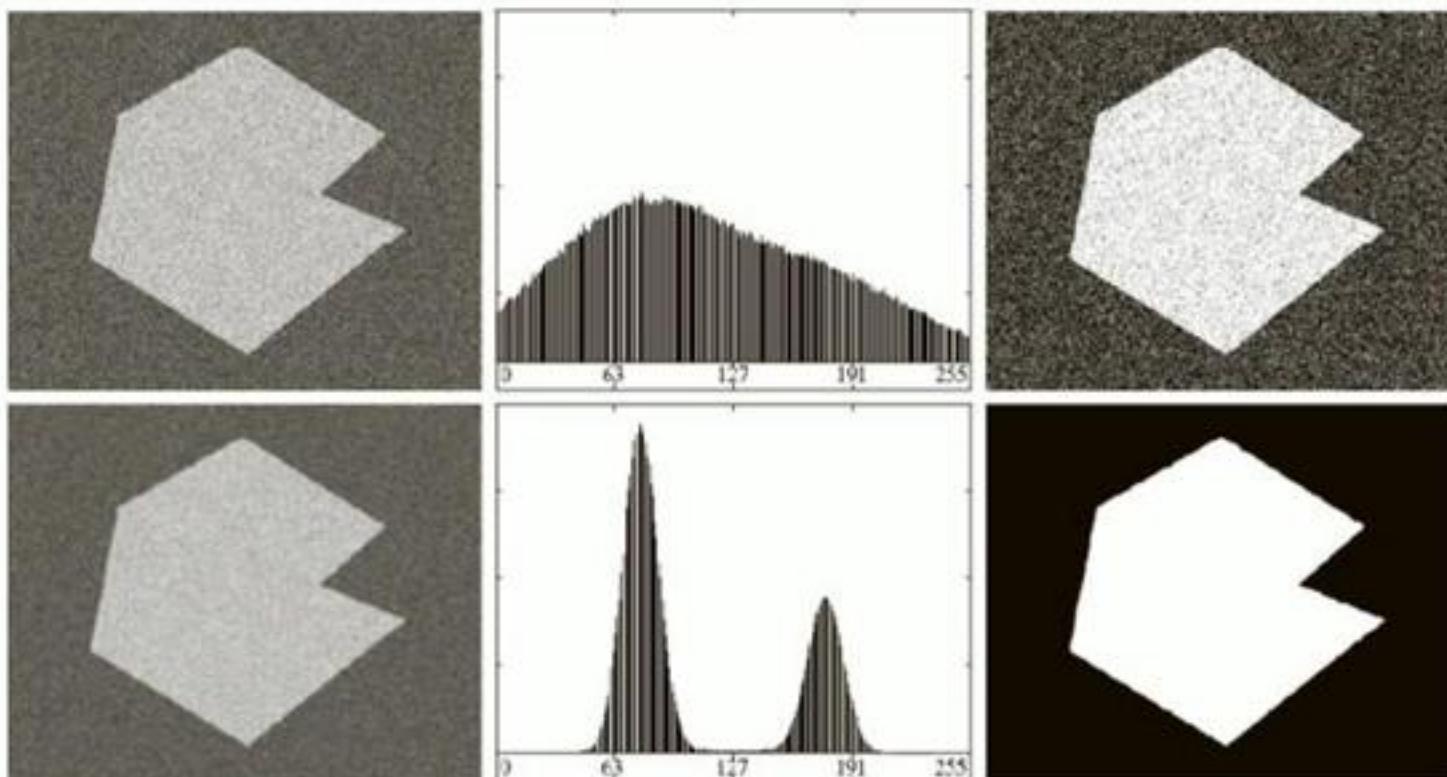
# Application: Otsu's Method



a b c  
d e f

**FIGURE 10.40** (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a  $5 \times 5$  averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

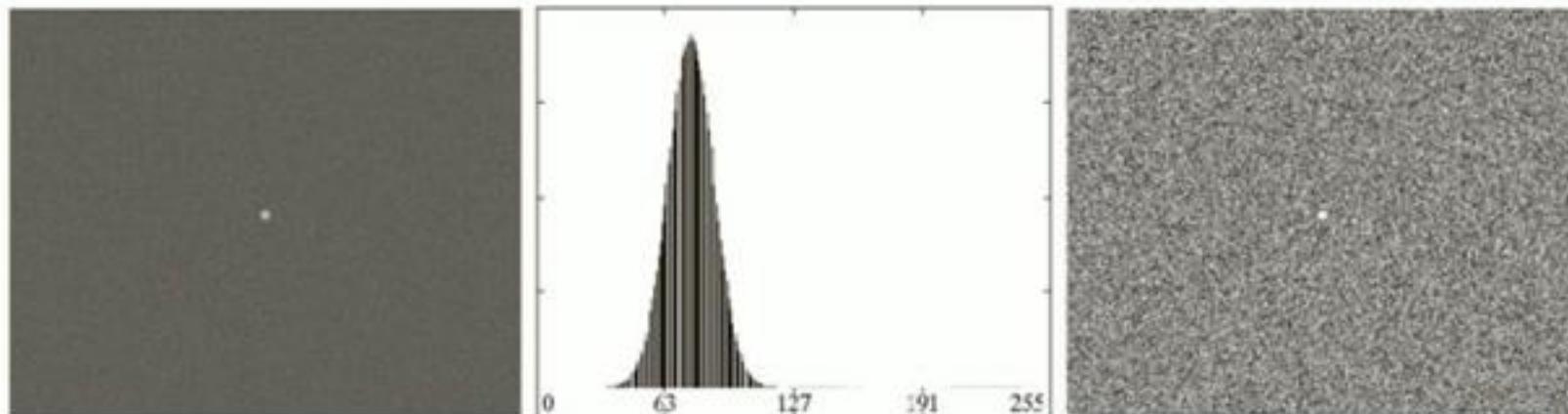
# Application: Otsu's Method



a b c  
d e f

**FIGURE 10.40** (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a  $5 \times 5$  averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

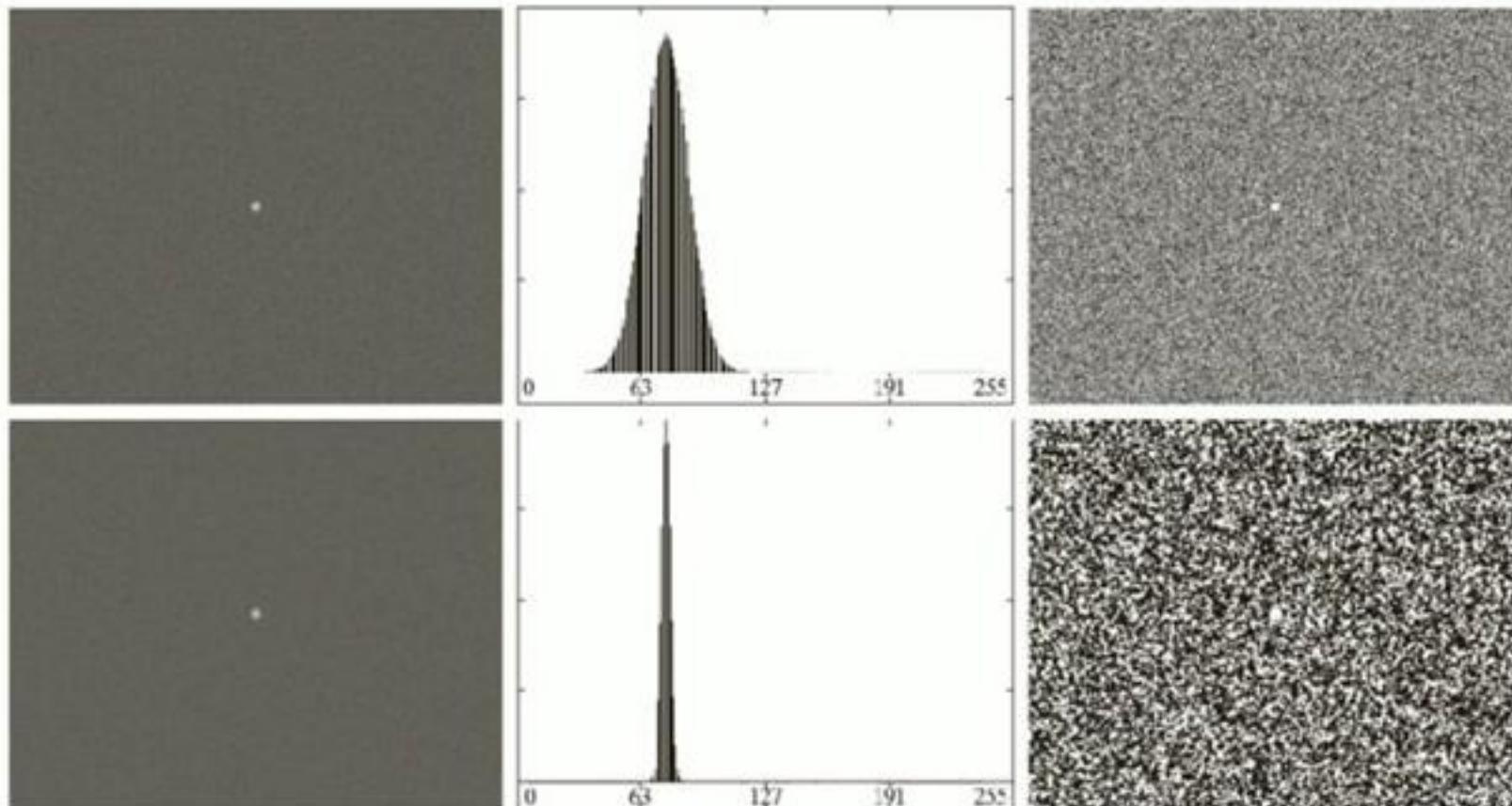
## Application: Otsu's Method



a b c  
d e f

**FIGURE 10.41** (a) Noisy image and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a  $5 \times 5$  averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method. Thresholding failed in both cases.

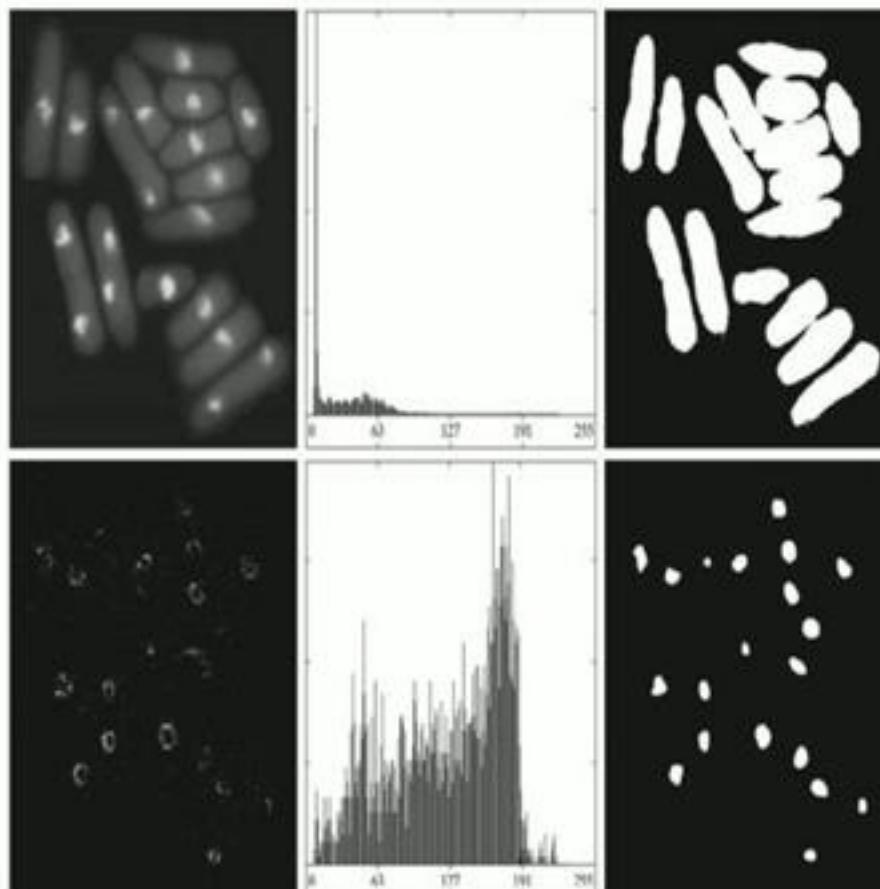
# Application: Otsu's Method



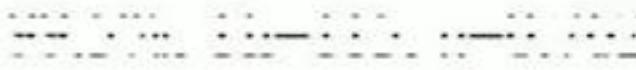
a b c  
d e f

**FIGURE 10.41** (a) Noisy image and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a  $5 \times 5$  averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method. Thresholding failed in both cases.

# Application: Otsu's Method

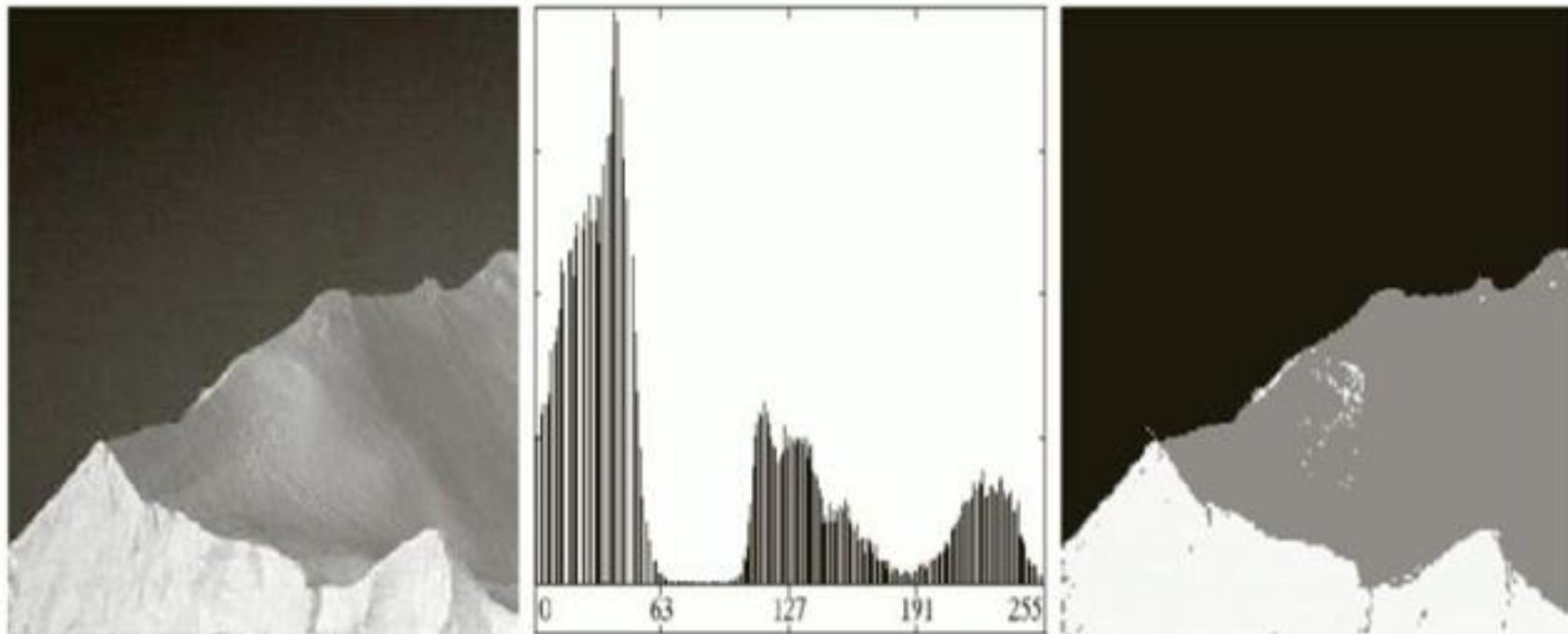


a b c  
d e f



**FIGURE 10.43** (a) Image of yeast cells. (b) Histogram of (a). (c) Segmentation of (a) with Otsu's method using the histogram in (b). (d) Thresholded absolute Laplacian. (e) Histogram of the nonzero pixels in the product of (a) and (d). (f) Original image thresholded using Otsu's method based on the histogram in (e). (Original image courtesy of Professor Susan L. Forsburg, University of Southern California.)

# Application: Otsu's Method



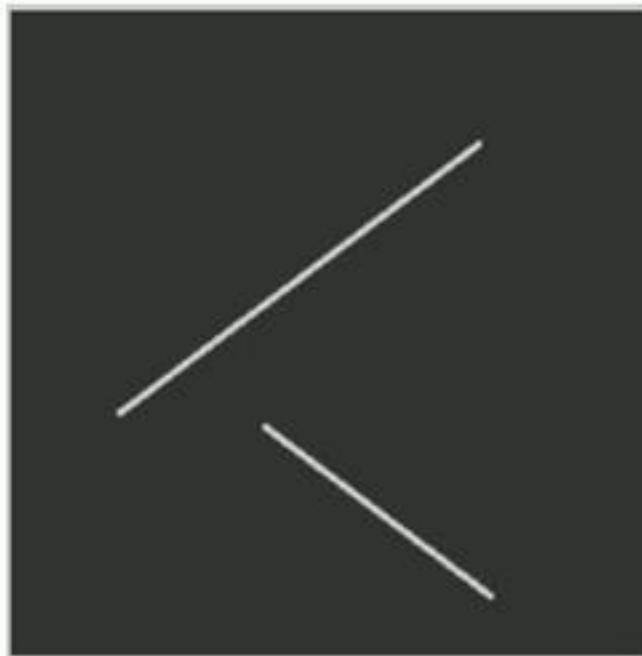
a b c

**FIGURE 10.45** (a) Image of iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)

The Hough Transform is useful to locate boundaries of regions which have a regular geometrical shape.

A method for detecting lines (or other parametric objects) in images.

It is patented by Paul Hough in 1962. The version of the method as it is used today was invented by Richard Duda and Peter Hart in 1972.



Original  
Image

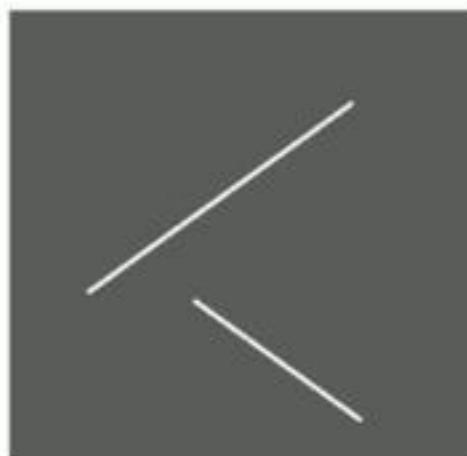


Hough Transform

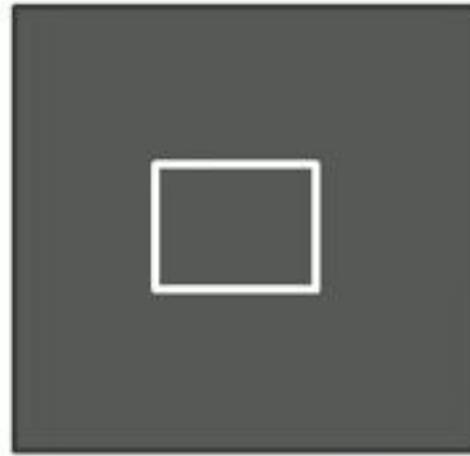
Matlab function: `hough()`

The Hough Transform is useful for detecting lines (or other parametric objects) in images.

It will locate boundaries of regions which have a regular geometrical shape.



Original Image



Matlab function  
`hough()`

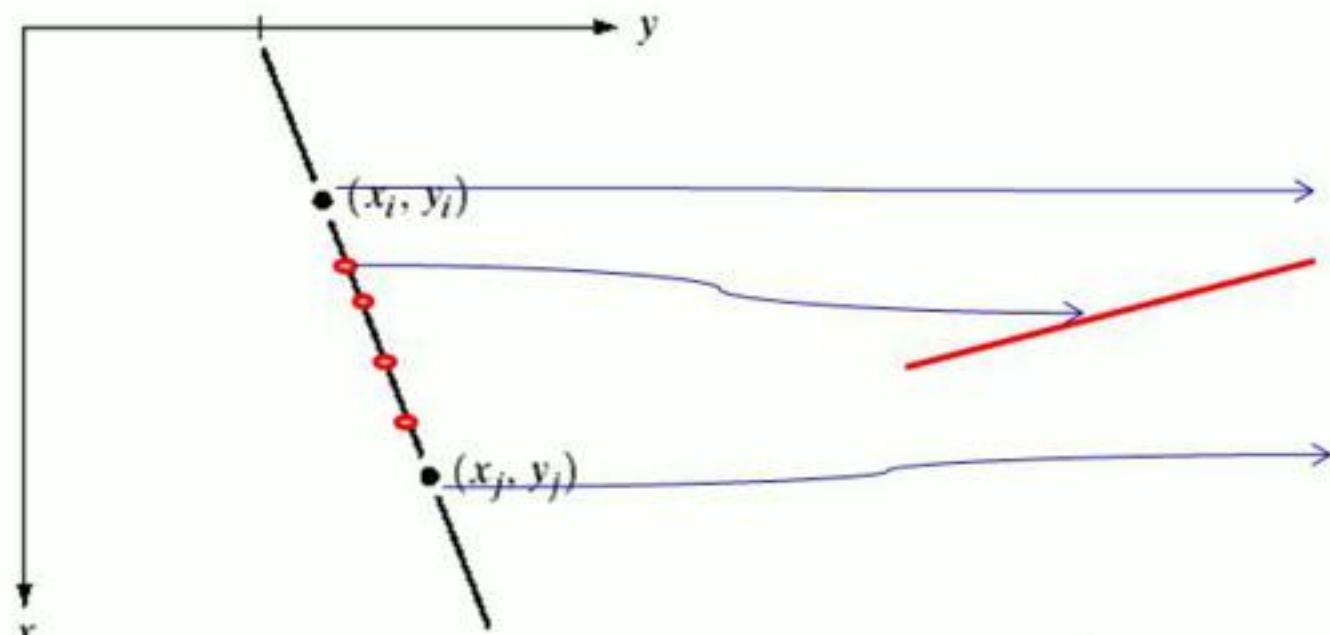


Hough Transform

It is patented by Paul Hough in 1962. The version of the method as it is used today was invented by Richard Duda and Peter Hart in 1972.

- It is used for object recognition.
- It is tolerant of gaps in edges, robust to partial deformation on shapes.
  - It is relatively less effected by noise.
  - It is also less effected by occlusion in the image.
- It can detect multiple occurrences of object in the same pass.
  - Conceptually simple.

- Infinitely many lines pass through  $(x_i, y_i)$  and  $(x_j, y_j)$ .

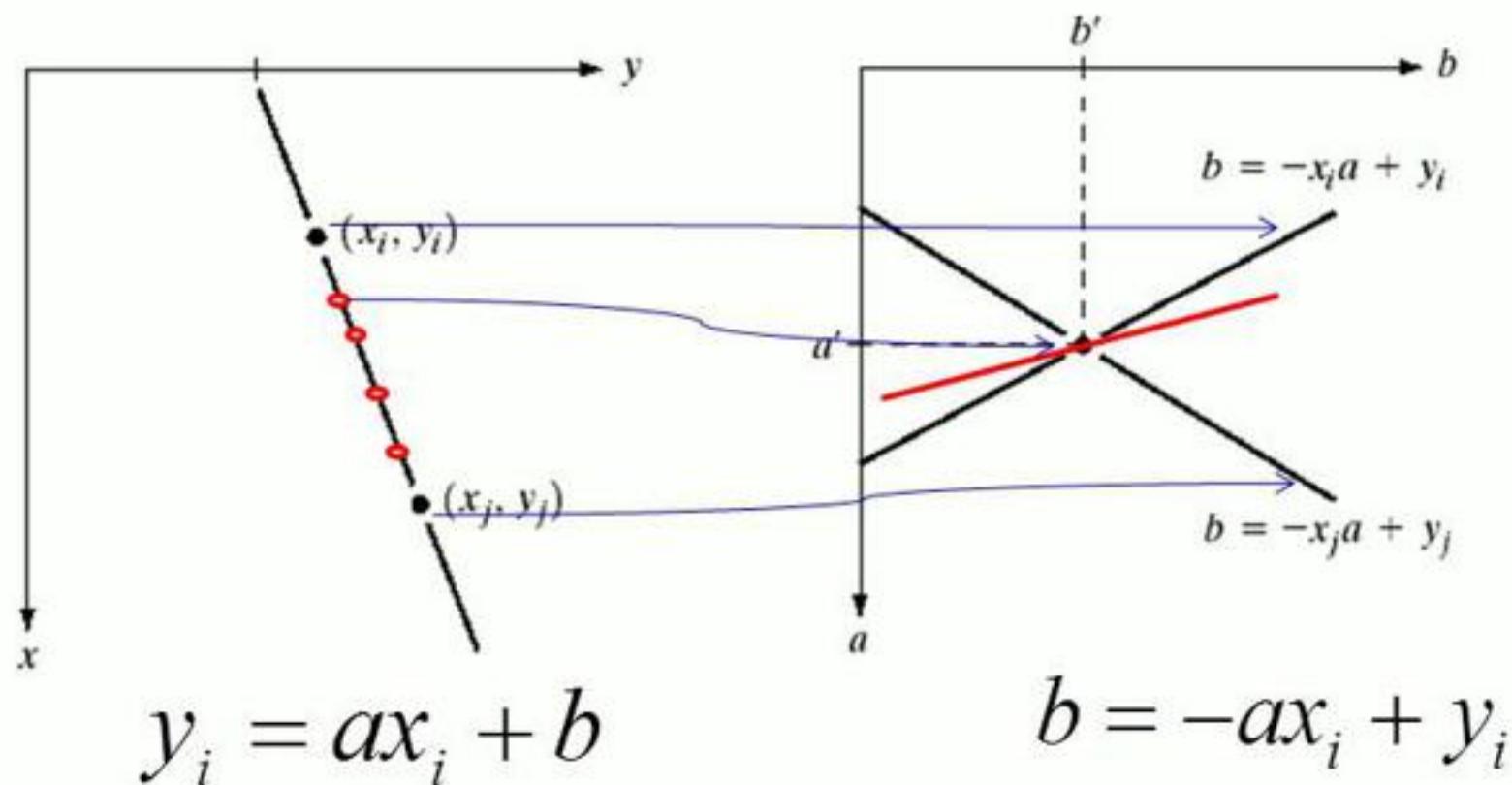


$$y_i = ax_i + b$$

$$b = -ax_i + y_i$$

# Hough Transform

- Infinitely many lines pass through  $(x_i, y_i)$  and  $(x_j, y_j)$ .



a | b

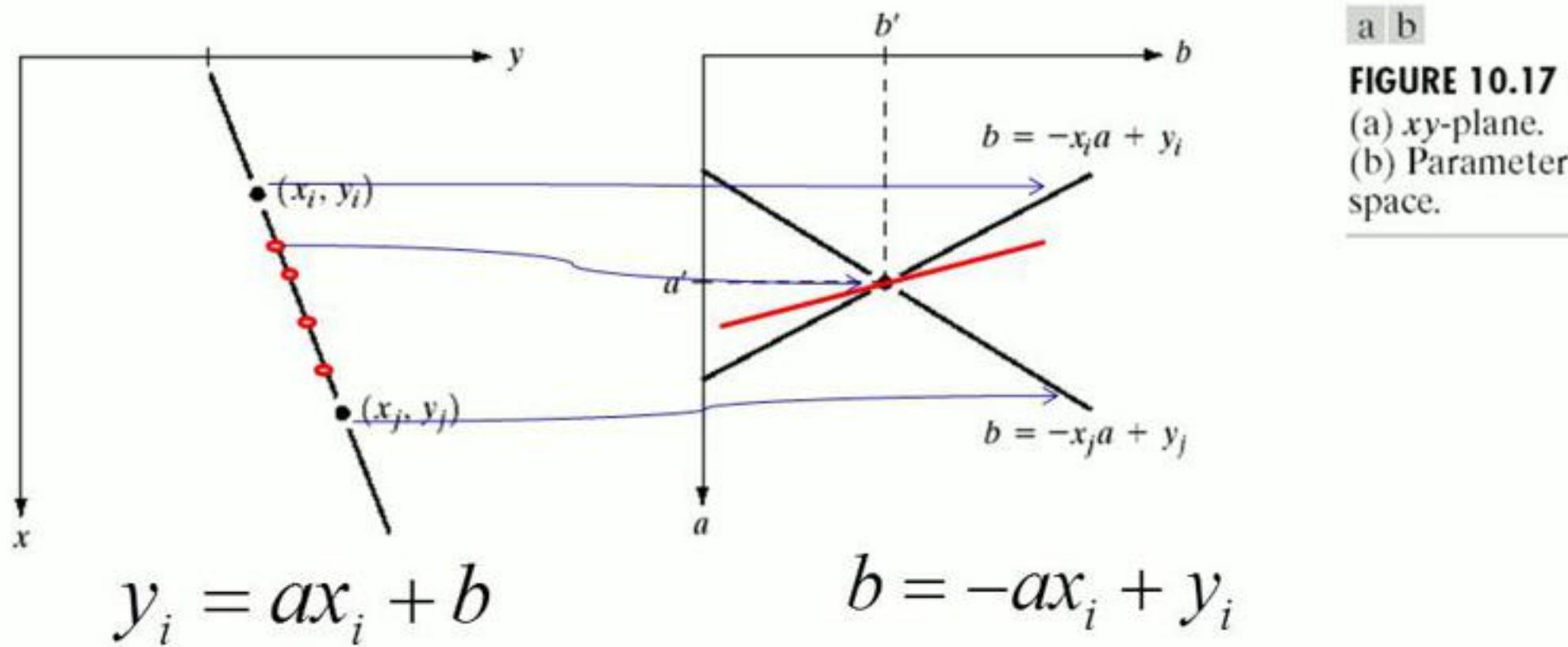
**FIGURE 10.17**

(a)  $xy$ -plane.  
(b) Parameter space.

$$y_i = ax_i + b$$

$$b = -ax_i + y_i$$

- Infinitely many lines pass through  $(x_i, y_i)$  and  $(x_j, y_j)$ .



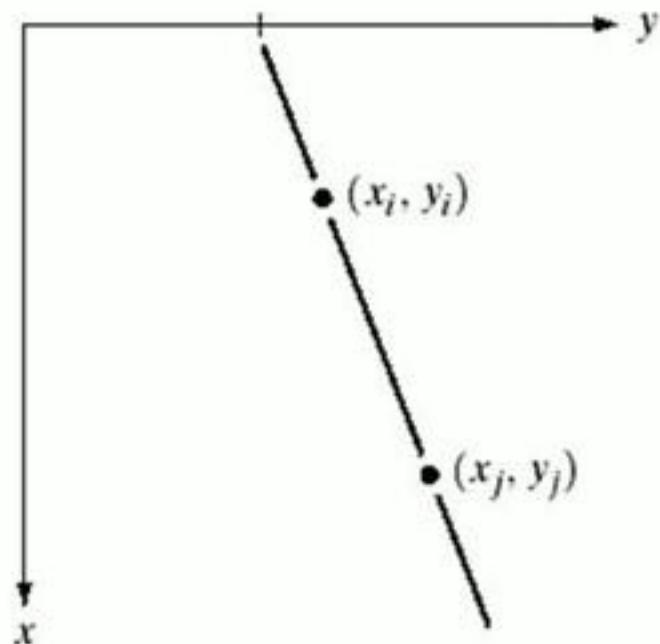
We seek that line which has the same value of  $a$  (slope) and  $b$  (intercept) for both the points.

- $xy$ -plane Vs  $ab$ -plane (parameter space)

$$y_i = ax_i + b$$

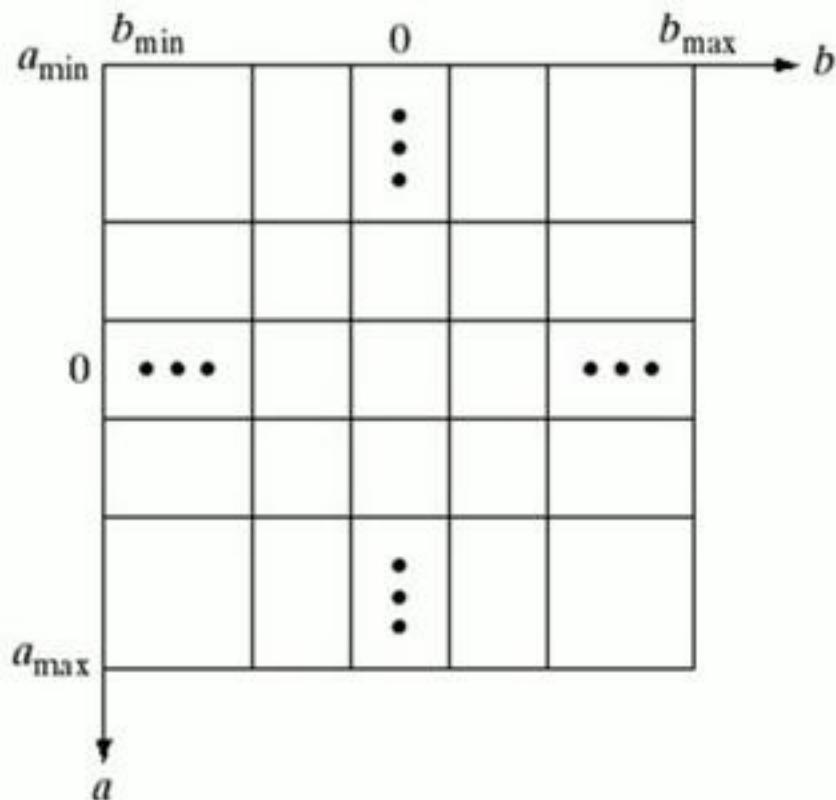
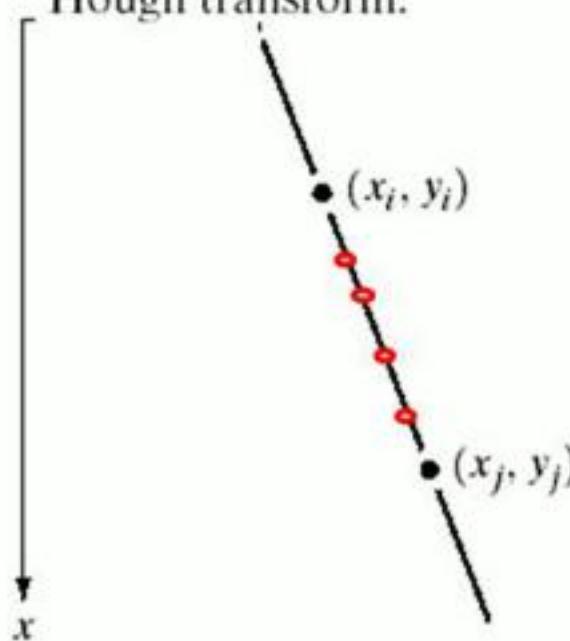
- $ab$ -plane

$$b = -x_i a + y_i$$



1. Divide the parameter space into cells.
2. Plot the **parameter space** lines corresponding to all points  $(x_k, y_k)$ .
3. Identify points in parameter space where large numbers of parameter space lines intersect.

**FIGURE 10.18**  
Subdivision of the parameter plane for use in the Hough transform.

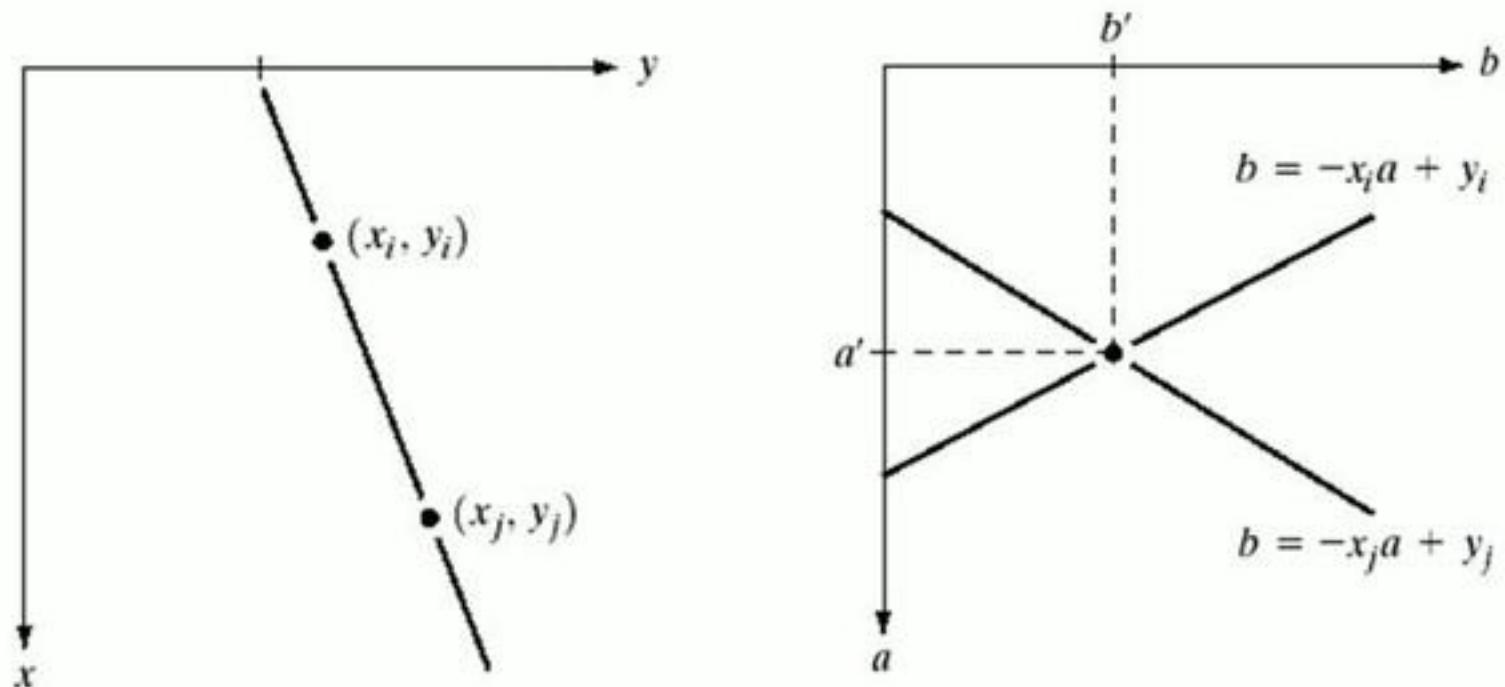


- $xy$ -plane Vs  $ab$ -plane (parameter space)

$$y_i = ax_i + b$$

- $ab$ -plane

$$b = -x_i a + y_i$$



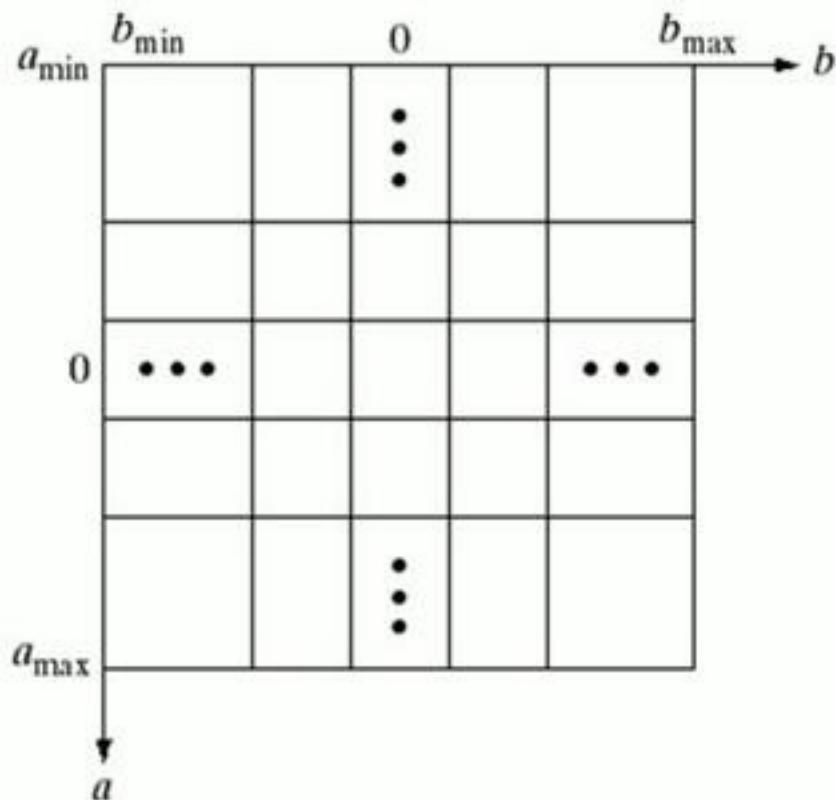
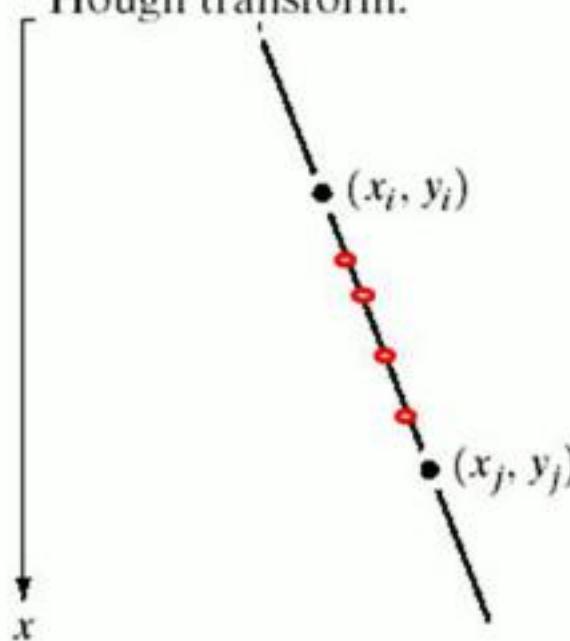
a | b

**FIGURE 10.17**  
(a)  $xy$ -plane.  
(b) Parameter  
space.

1. Divide the parameter space into cells.
2. Plot the **parameter space** lines corresponding to all points  $(x_k, y_k)$ .
3. Identify points in parameter space where large numbers of parameter space lines intersect.

**FIGURE 10.18**

Subdivision of the parameter plane for use in the Hough transform.



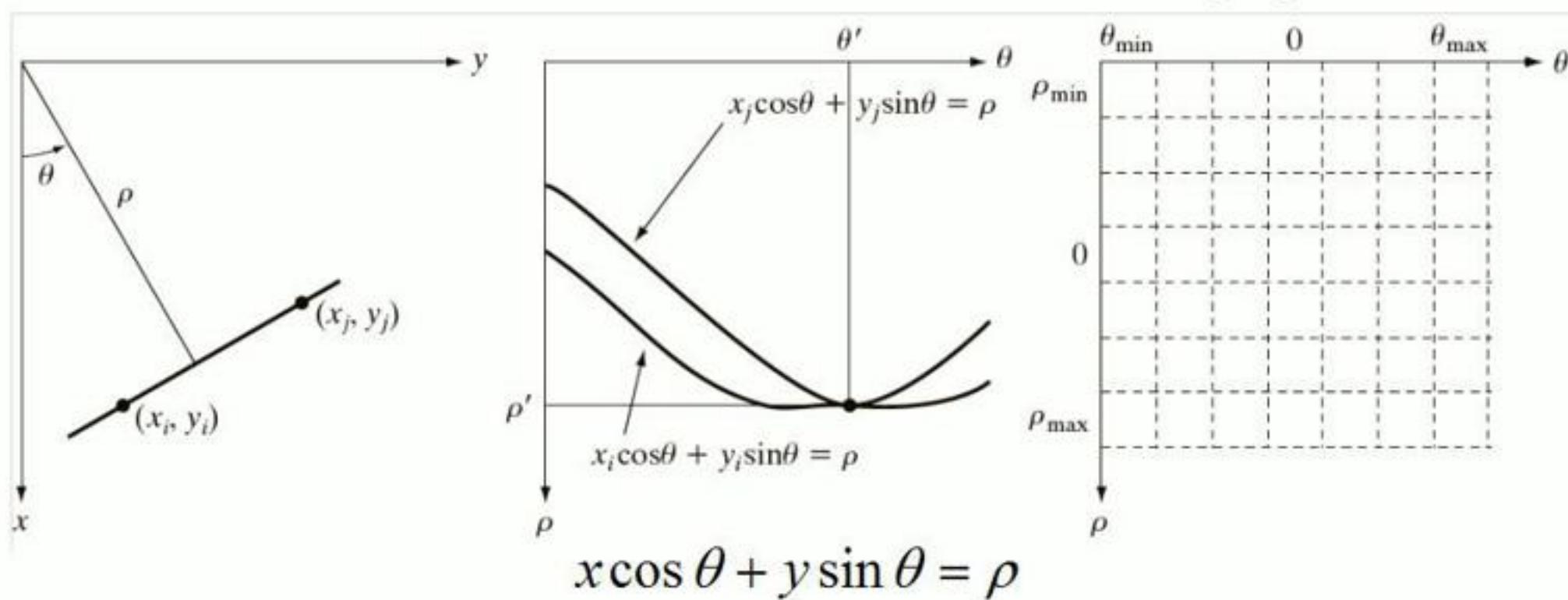
Any Practical difficulty with this approach?

*Problem- If line is vertical ( $90^\circ$ ) then slope is infinite.*

$$a = \frac{y_i - b}{x_i}$$

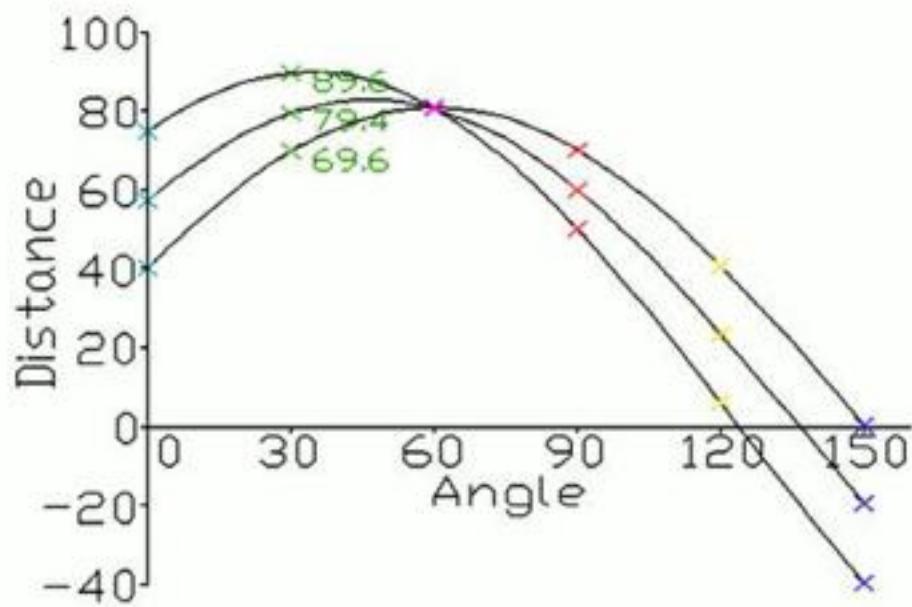
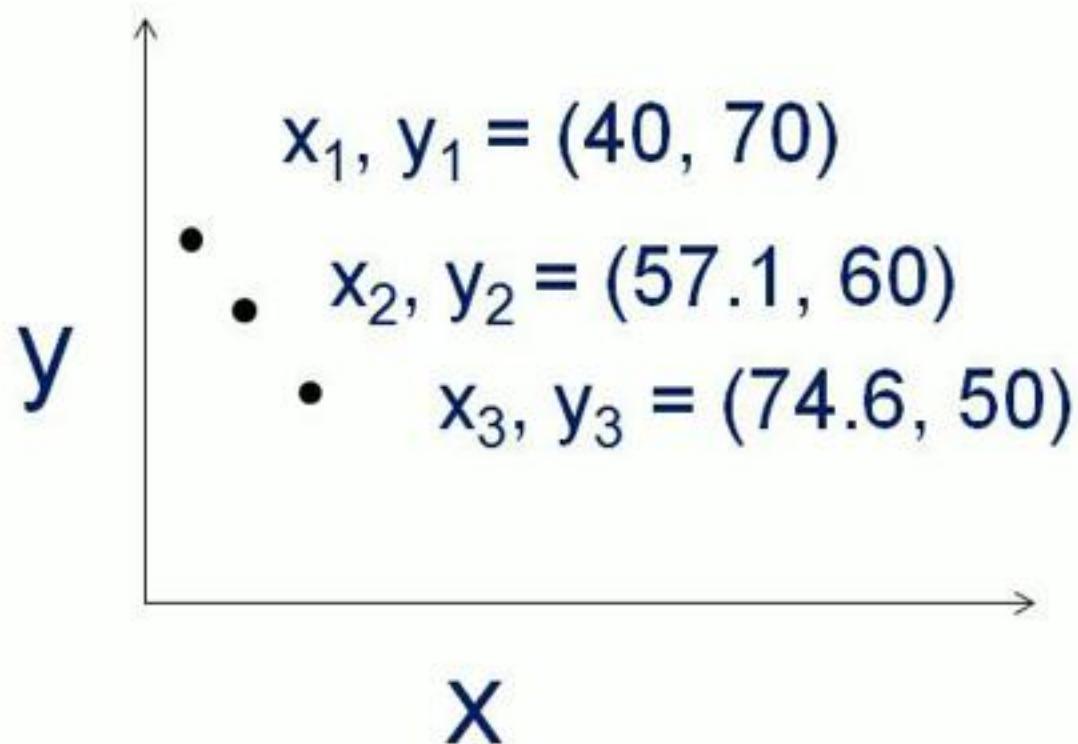
So we need to look at infinite number of cells.

- The Hough transform consists of finding all pairs of values of  $\theta$  and  $\rho$  which satisfy the equations that pass through  $(x_j, y_j)$ .
- When plotted these pairs of  $\theta$  and  $\rho$  will look like a **sine** wave. The process is repeated for all appropriate  $(x_j, y_j)$  locations.



1. Divide the  $\rho-\theta$  parameter space into accumulator cells.
2. The cell with coordinates  $(i,j)$  with accumulator value  $A(i,j)$  corresponds to the square associated with parameter space coordinates  $\rho(i,j)$ .
3. Initially, set all accumulator cells to zero.
4. For every non-background point  $(x_k, y_k)$  in the  $x-y$  plane, we let  $\theta$  span all possible values and solve for  $\rho$  using the normal equation of a line.
5. Resulting  $\rho$  values are rounded off to the nearest allowed value and the corresponding accumulator cell is incremented.

- Number of subdivisions in the  $\rho-\theta$  plane determines the accuracy of the co linearity of these points.
- At the end of the procedure a value of  $P$  in  $A(p,q)$  means that  $P$  points in the  $xy$  plane lie on the line
$$x \cos \theta_j + y \sin \theta_j = \rho_i$$
- Number of computations in the Hough Transform is of the order of  $n$  (number of non-background points)

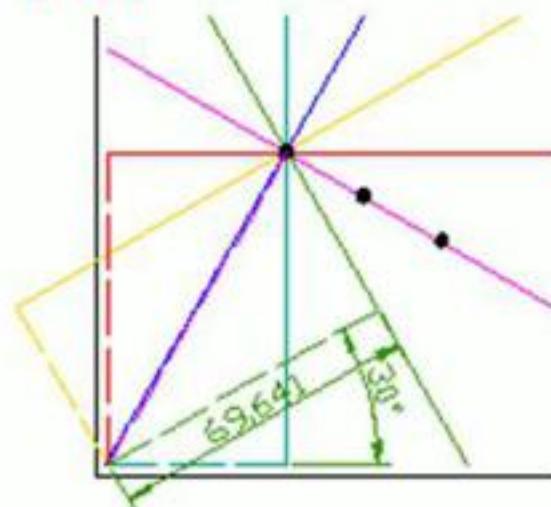


Source: [https://en.wikipedia.org/wiki/Hough\\_transform](https://en.wikipedia.org/wiki/Hough_transform)

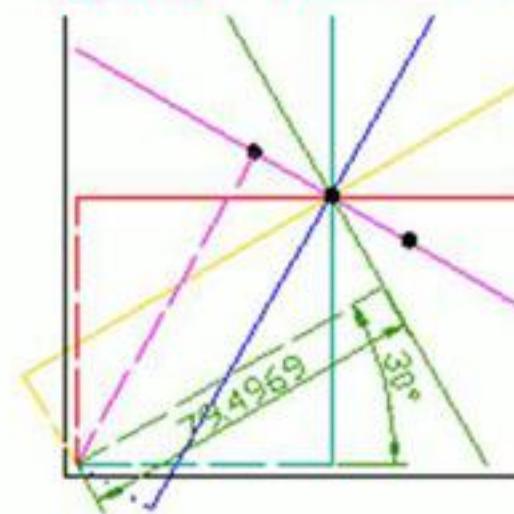
# Hough Transform

Consider the problem of find the line that contains 3 given points.

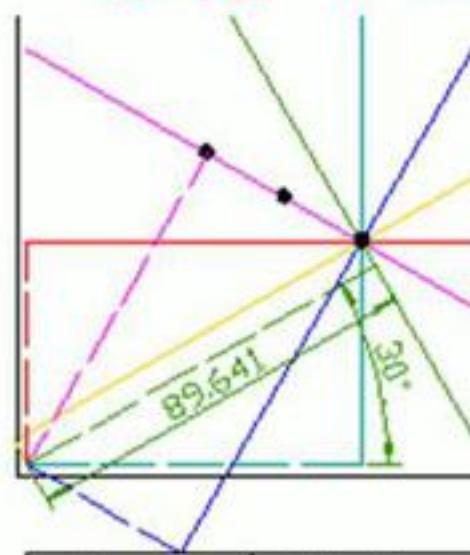
$$x_1, y_1 = (40, 70)$$



$$x_2, y_2 = (57.1, 60)$$



$$x_3, y_3 = (74.6, 50)$$

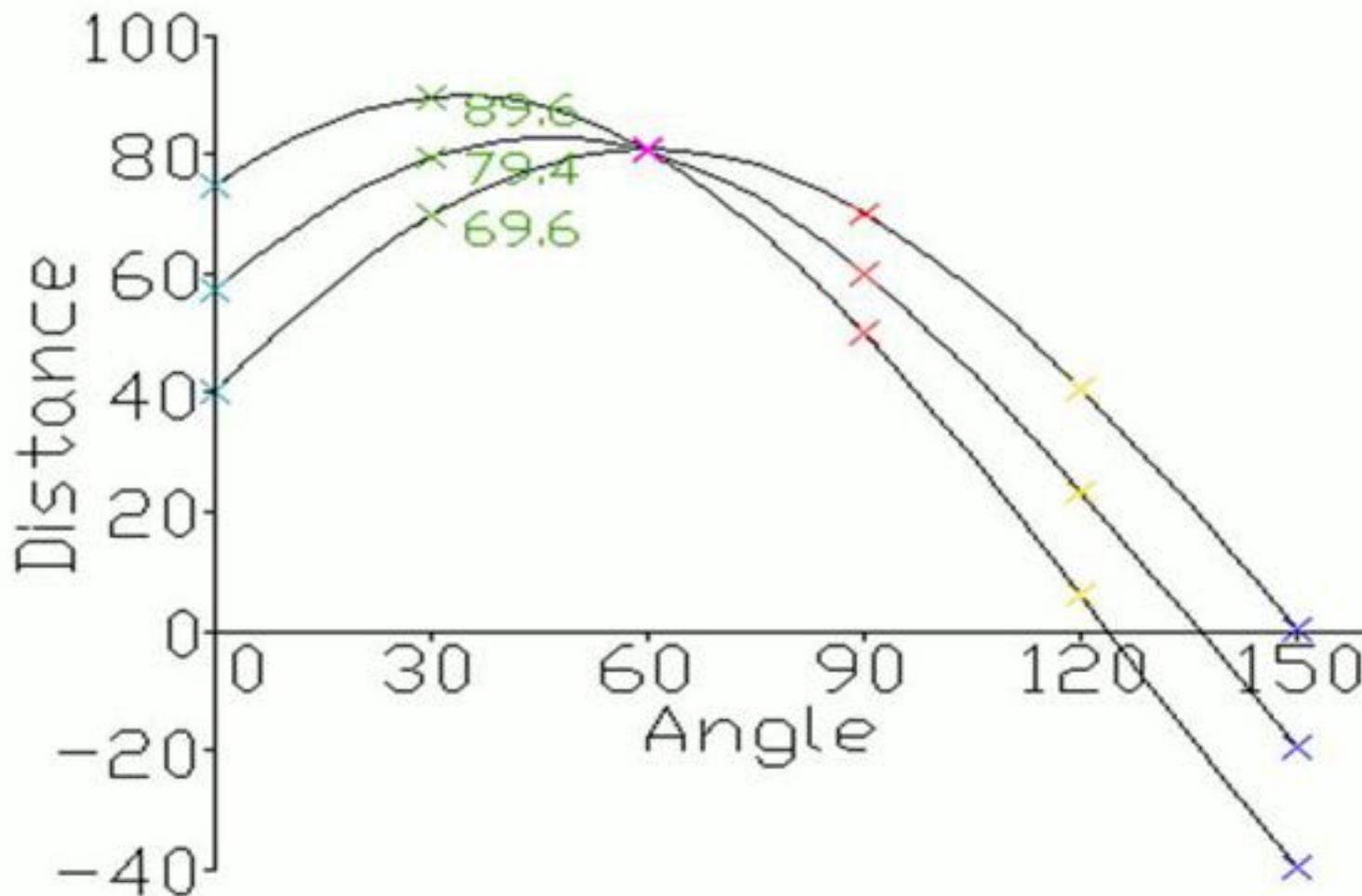


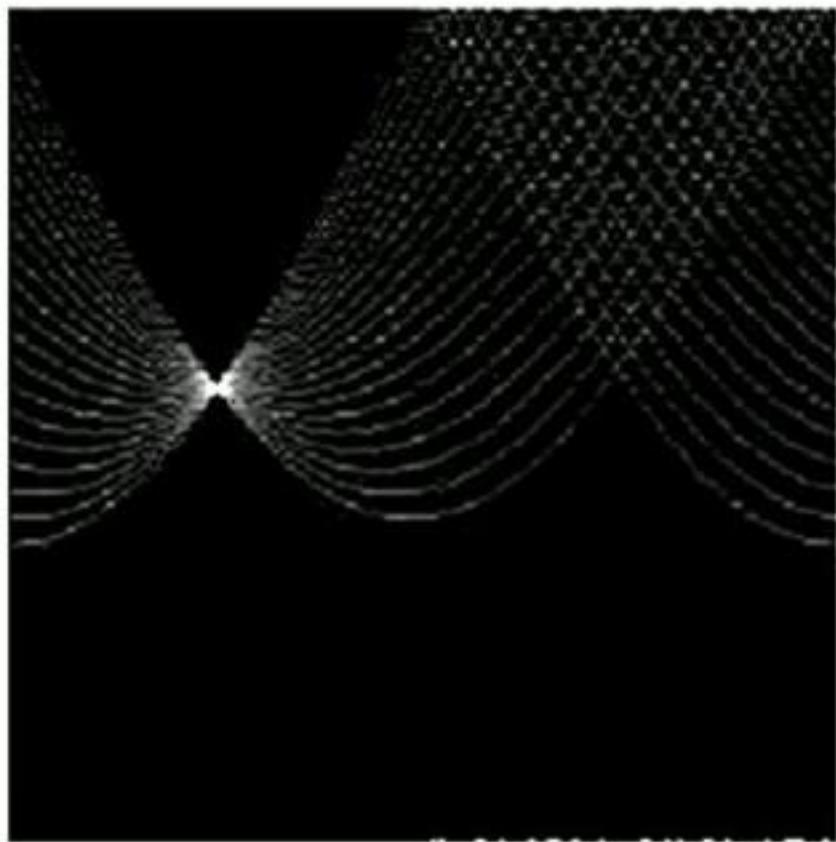
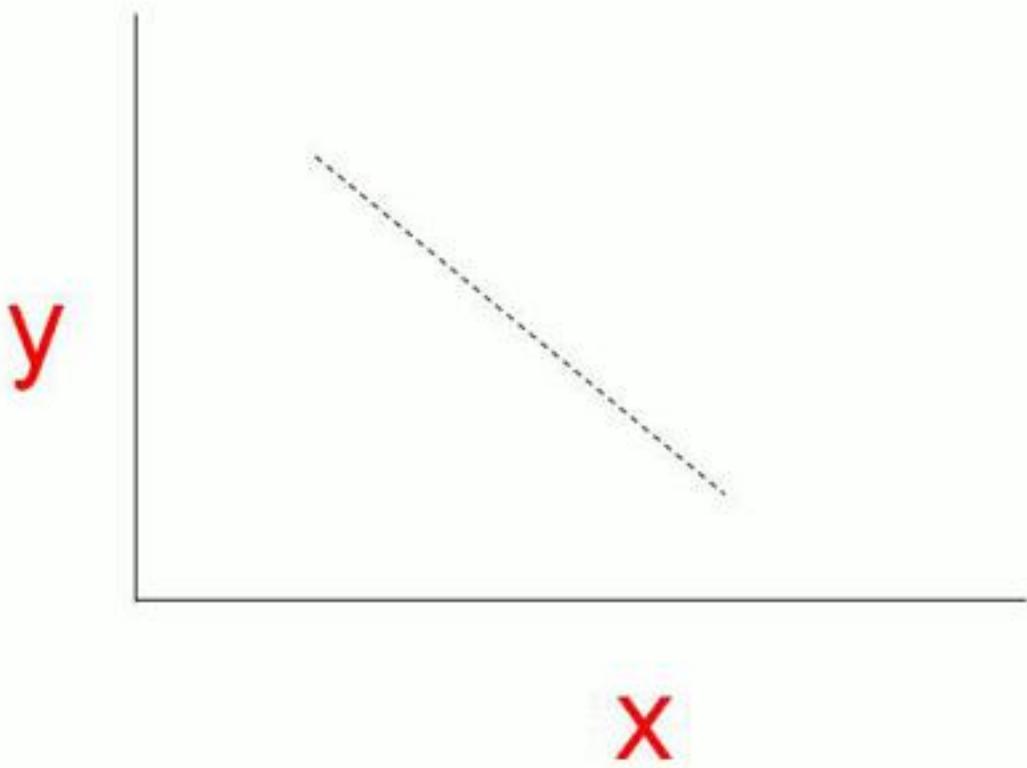
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

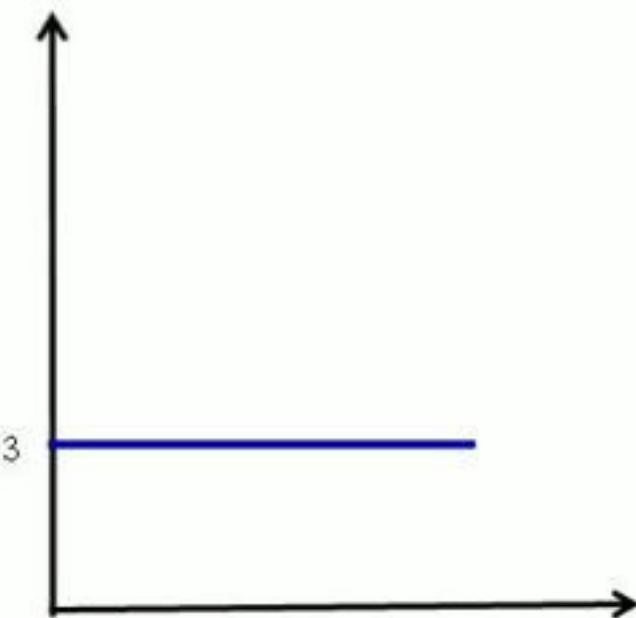
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

Consider the problem of find the line that contains 3 given points.

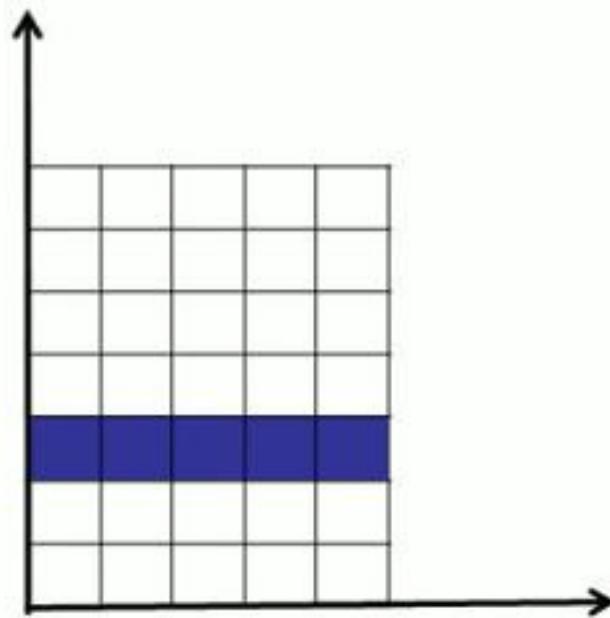
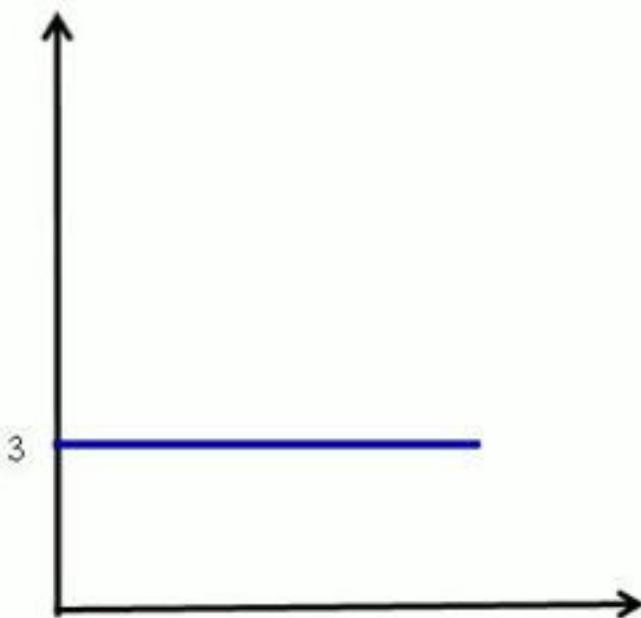


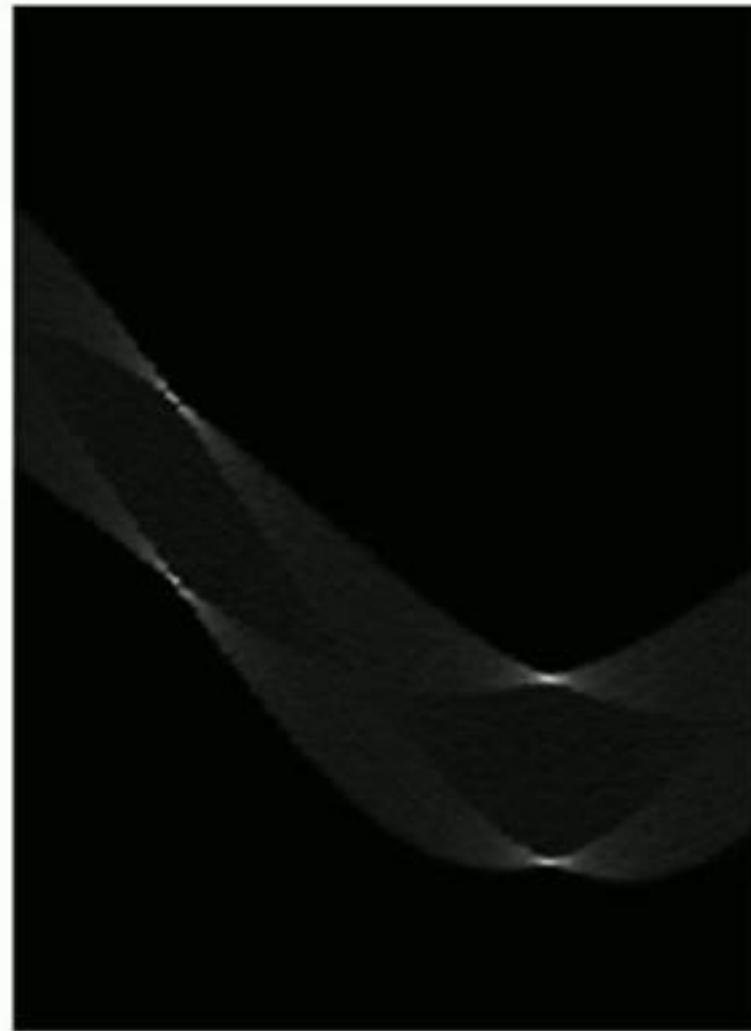


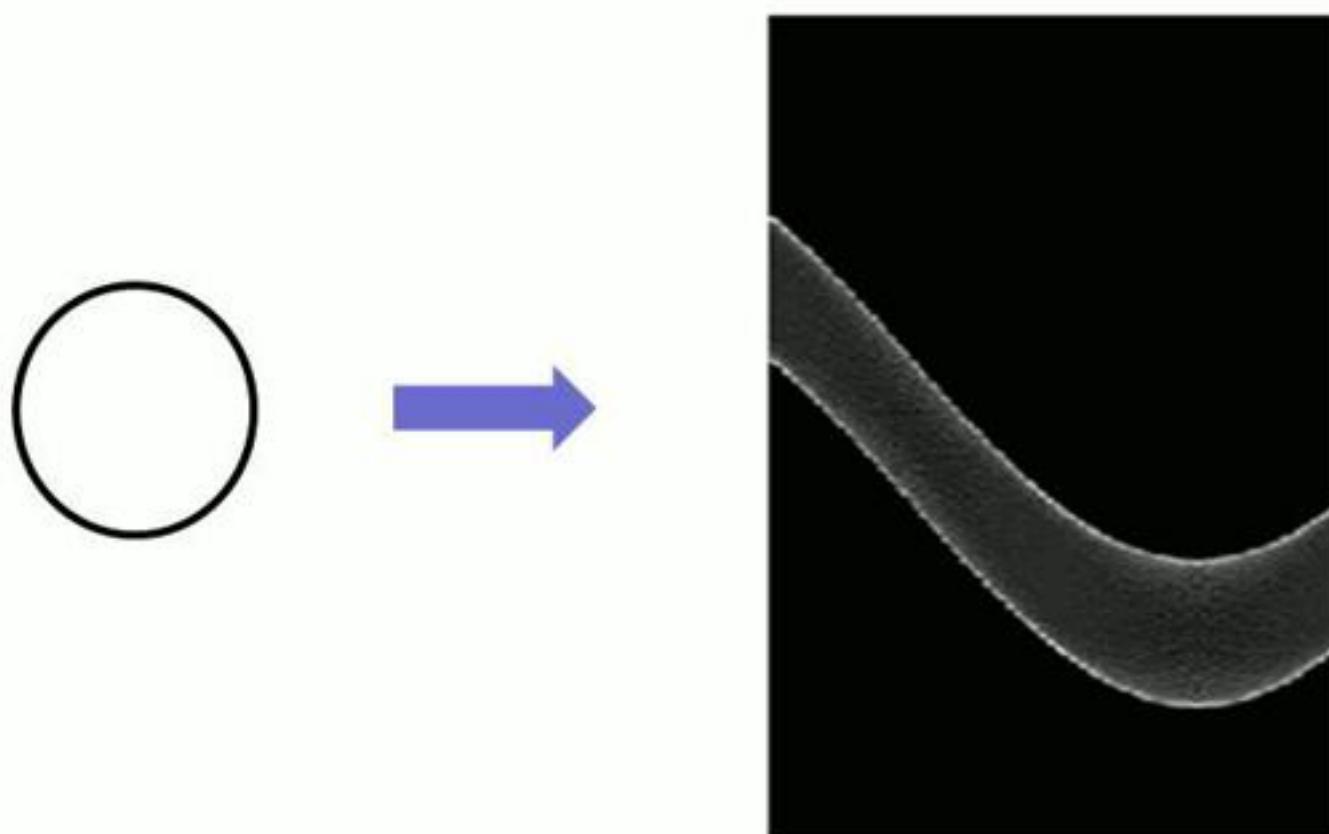
For given image, detect the line and discretize it as  $7 \times 5$  binary image  $X$  using Hough transform. Find the  $\rho - \theta$  plane values by proper resolution in distance-angle plane and plot it.

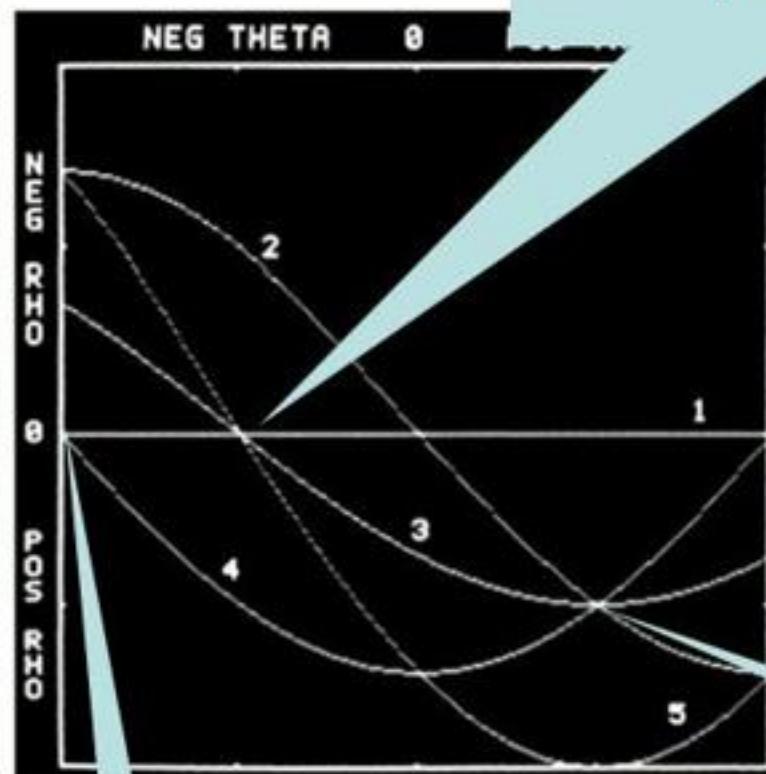
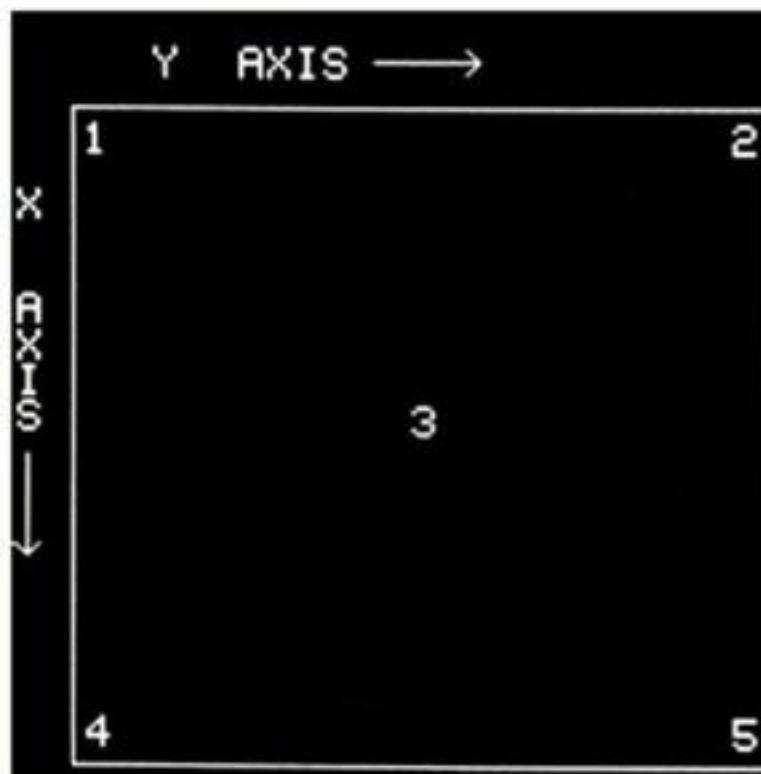


For given image, detect the line and discretize it as  $7 \times 5$  binary image  $X$  using Hough transform. Find the  $\rho - \theta$  plane values by proper resolution in distance-angle plane and plot it.









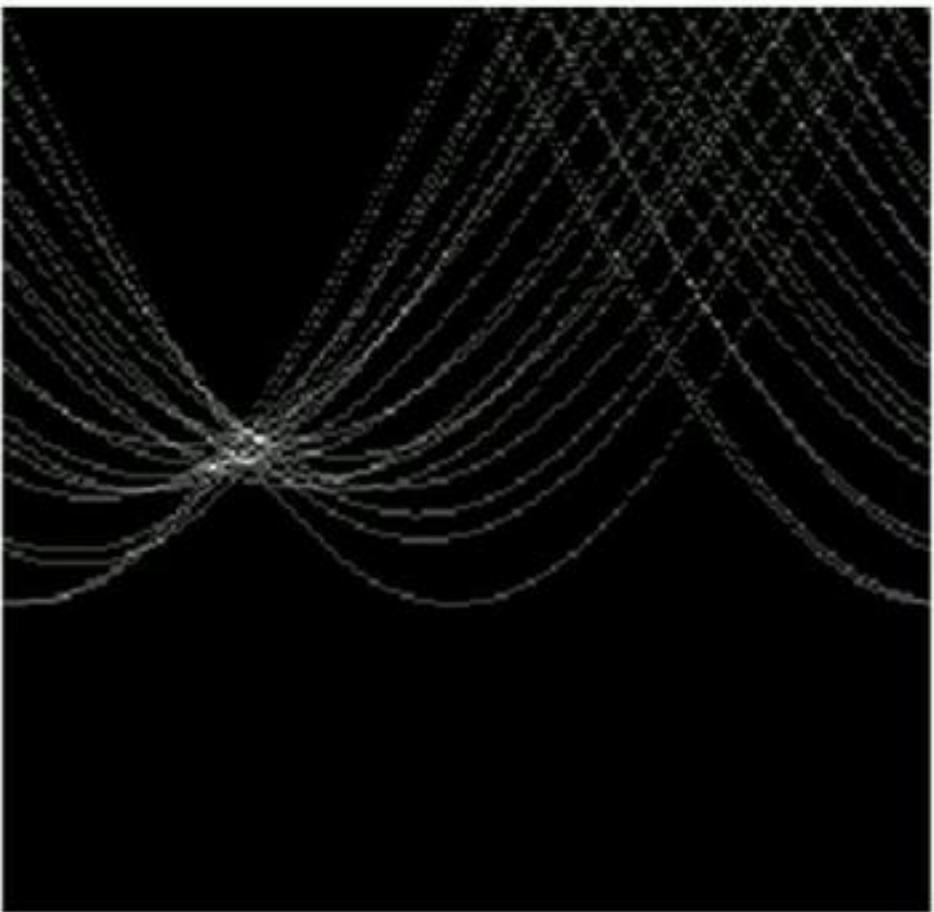
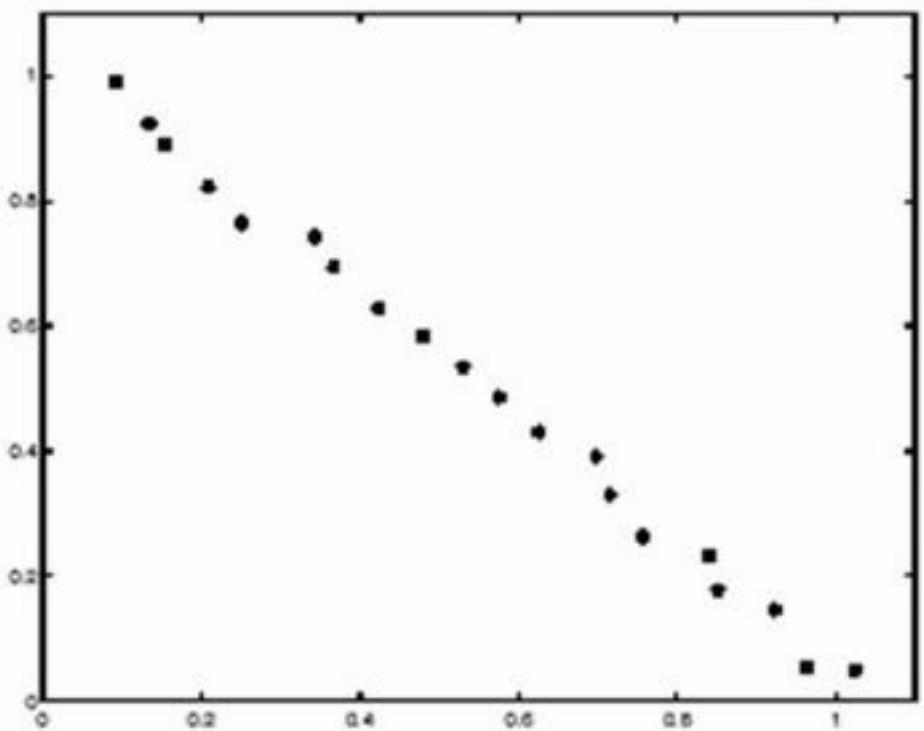
The intersection of the curves corresponding to points 1,3,5

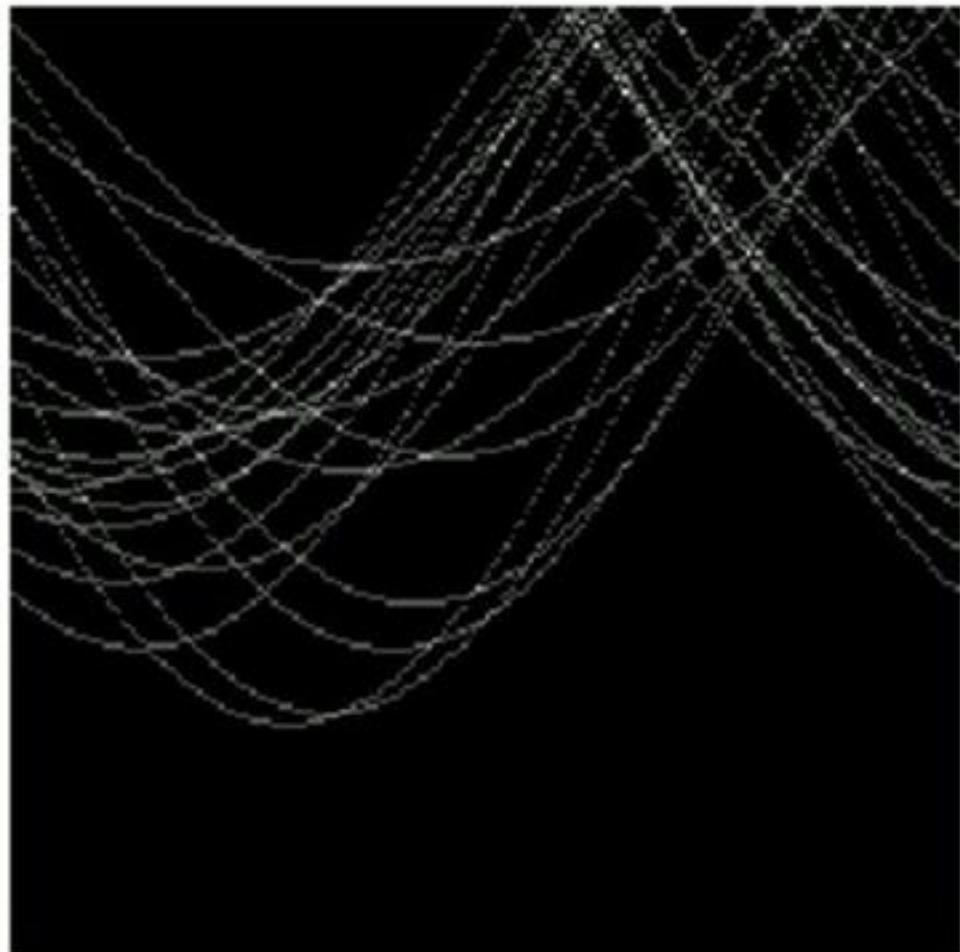
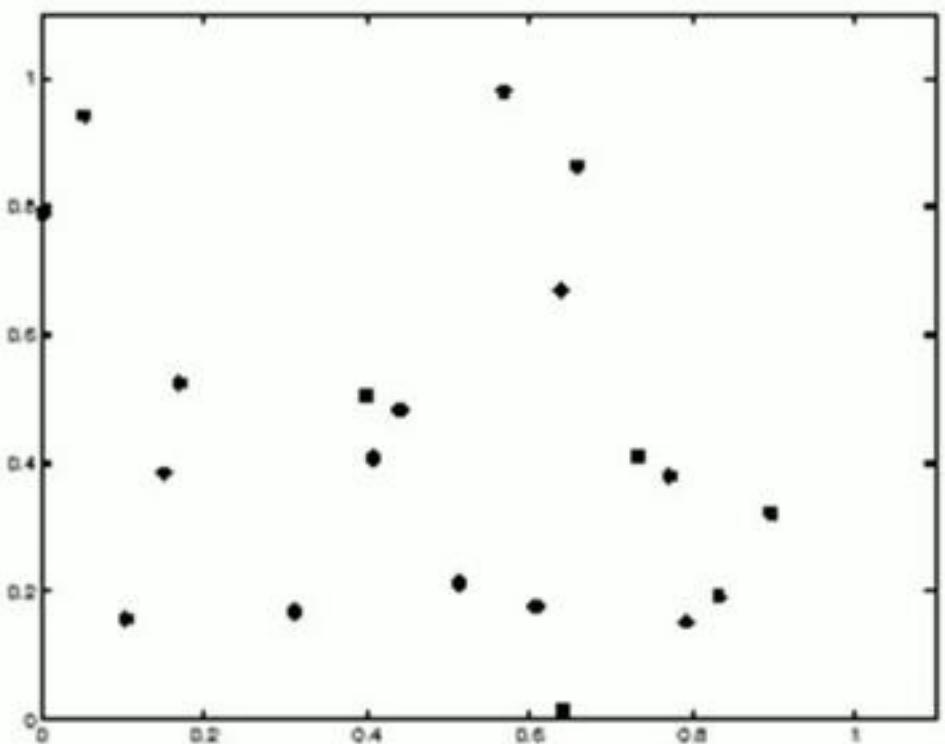
(a) Image of size  $101 \times 101$  pixels, containing five points.

(b) Corresponding parameter space.  
(The points in (a) were enlarged to make them easier

1,4

to see)



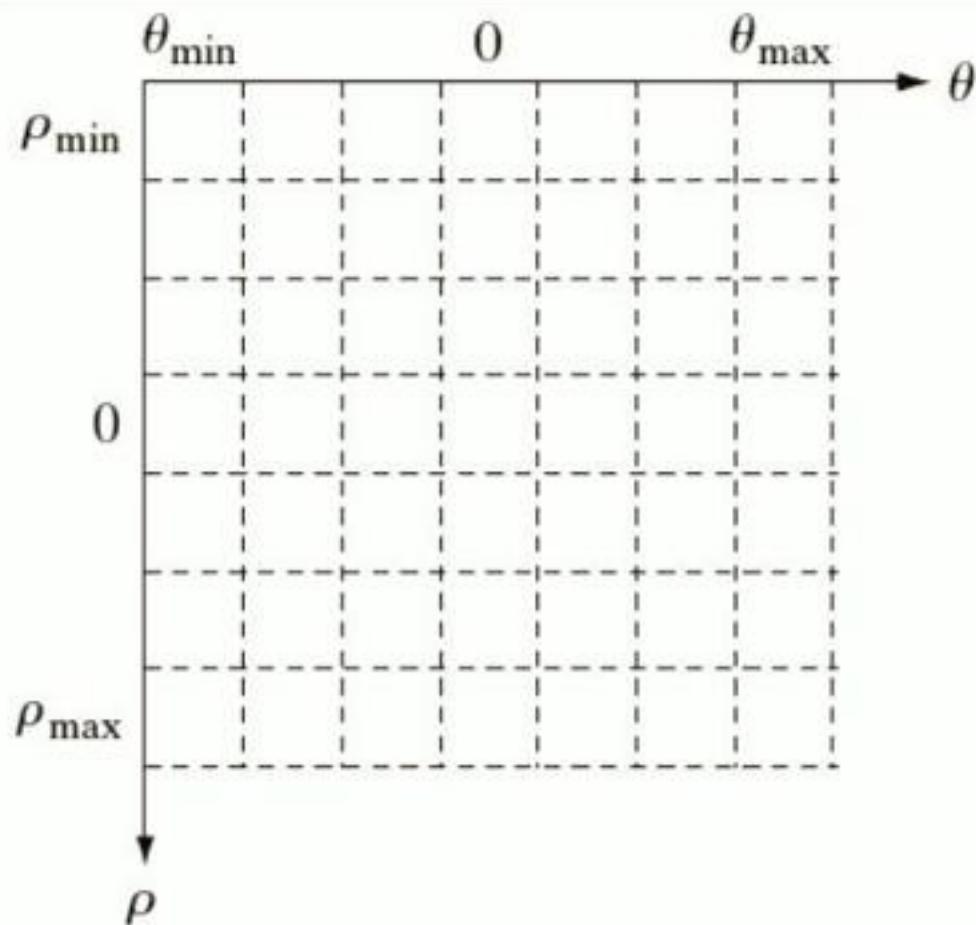


## Grid resolution tradeoff

- Too coarse: large votes obtained when too many different lines correspond to a single bucket
- Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets

## Increment neighboring bins

- Smoothing in accumulator array



The brute force approach is computationally expensive ----

If there are n points in an image:

There are pair nodes  $n(n-1)/2$  or possible straight lines.

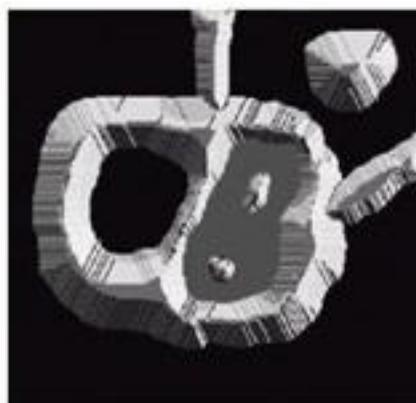
Then we have to do  $(n-2)(n(n-1))/2$  comparisons to see whether each of those points lies on some line joining two other points.

# Morphological Watersheds

The concept of watersheds is based on visualizing an image in three dimensions:

**two spatial coordinates versus gray levels.**



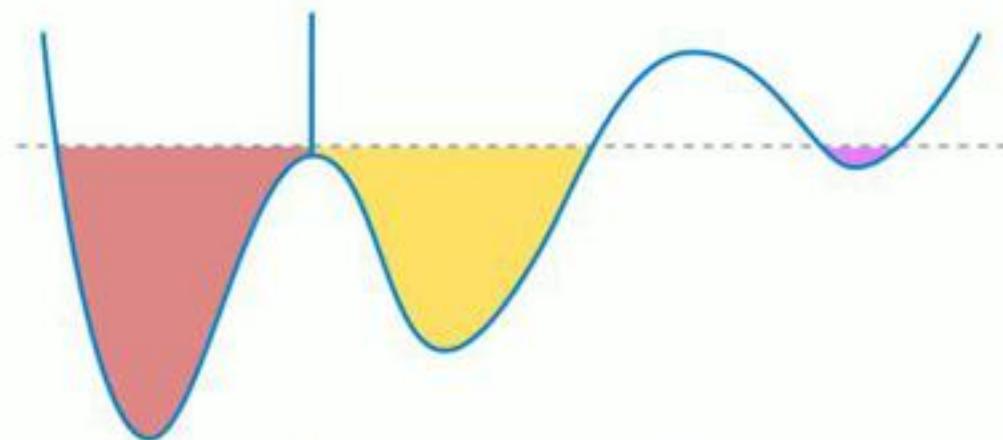


In such a topographic interpretation, we consider three types of points:

- (a) points belonging to a regional minimum.
- (b) points at which a drop of water would fall with certainty to a single minimum
- (c) points at which water would be equally likely to fall to more than one such minimum

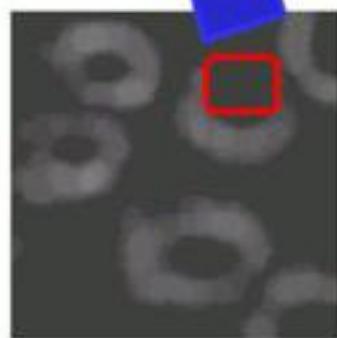
The set of points satisfying (b) is called watershed or catchment basin and condition (c) form crest lines or watershed divide line

- Watershed line form a connected path, thus giving continuous boundaries between regions.
- Watershed segmentation is in the extraction of nearly uniform (blob like) objects from the background.



- Suppose that a hole is punched in each regional minimum.
- The entire topography is flooded from below by letting water rise through the holes at a uniform rate.
- When the rising water in distinct catchment basins is about to merge, a dam is built to prevent the merging.
- The flooding will eventually reach a stage when only the tops of the dams are visible above the water line.
- These dam boundaries correspond to the divide lines of the watersheds.
  - They are the continuous boundaries extracted by a watershed algorithm.

## Example



16.8	19.2	13.5	20.5	31.2	30.1
18.7	11.8	15.4	18.5	22.1	18.4
20.1	21.9	26.6	20.8	17.3	18.1
25.3	22.8	20.9	19.8	15.1	15.9
30.7	35.5	29.9	18.7	17.6	39.9
34.8	38.6	33.4	32.7	33.5	36.7

# Watershed segmentation

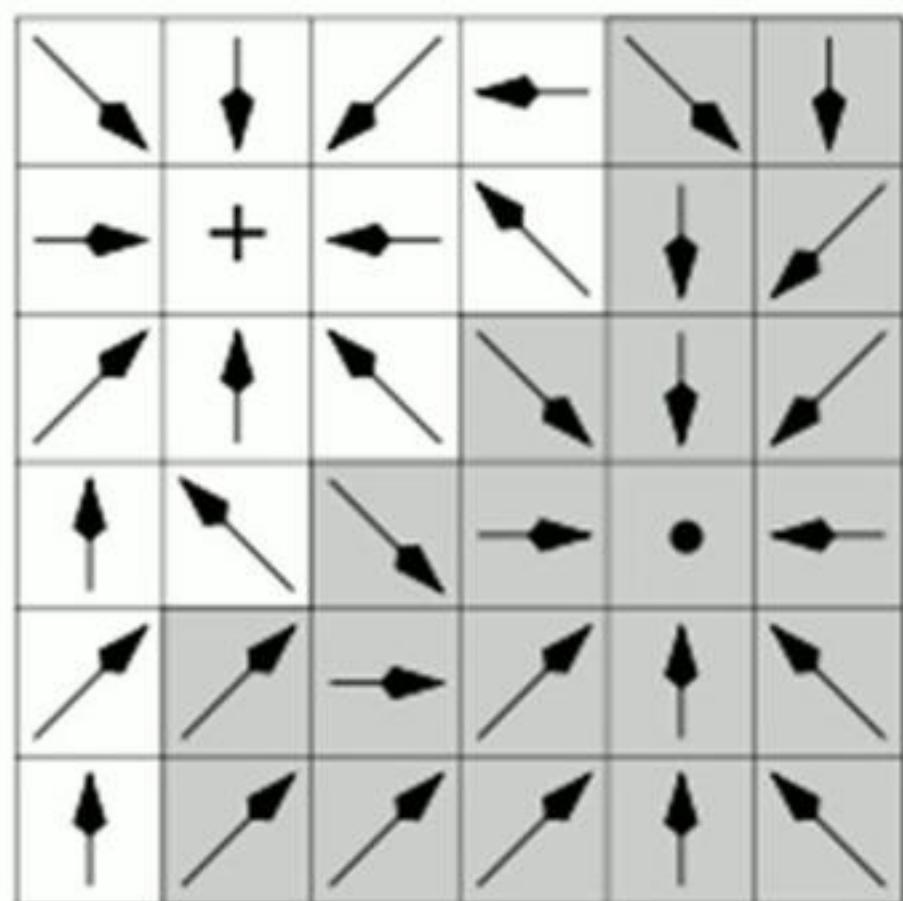
16.8	19.2	13.5	20.5	31.2	30.1
18.7	11.9	15.4	18.5	22.1	18.4
20.1	21.9	26.6	20.8	17.3	18.1
25.3	22.8	20.9	19.8	15.1	15.9
30.7	35.5	29.9	18.7	17.6	39.9
34.8	38.6	33.4	32.7	33.5	36.7

(a)

# Watershed segmentation

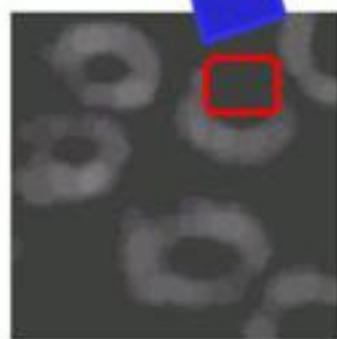
16.8	19.2	13.5	20.5	31.2	30.1
18.7	11.9	15.4	18.5	22.1	18.4
20.1	21.9	26.6	20.8	17.3	18.1
25.3	22.8	20.9	19.8	15.1	15.9
30.7	35.5	29.9	18.7	17.6	39.9
34.8	38.6	33.4	32.7	33.5	36.7

(a)



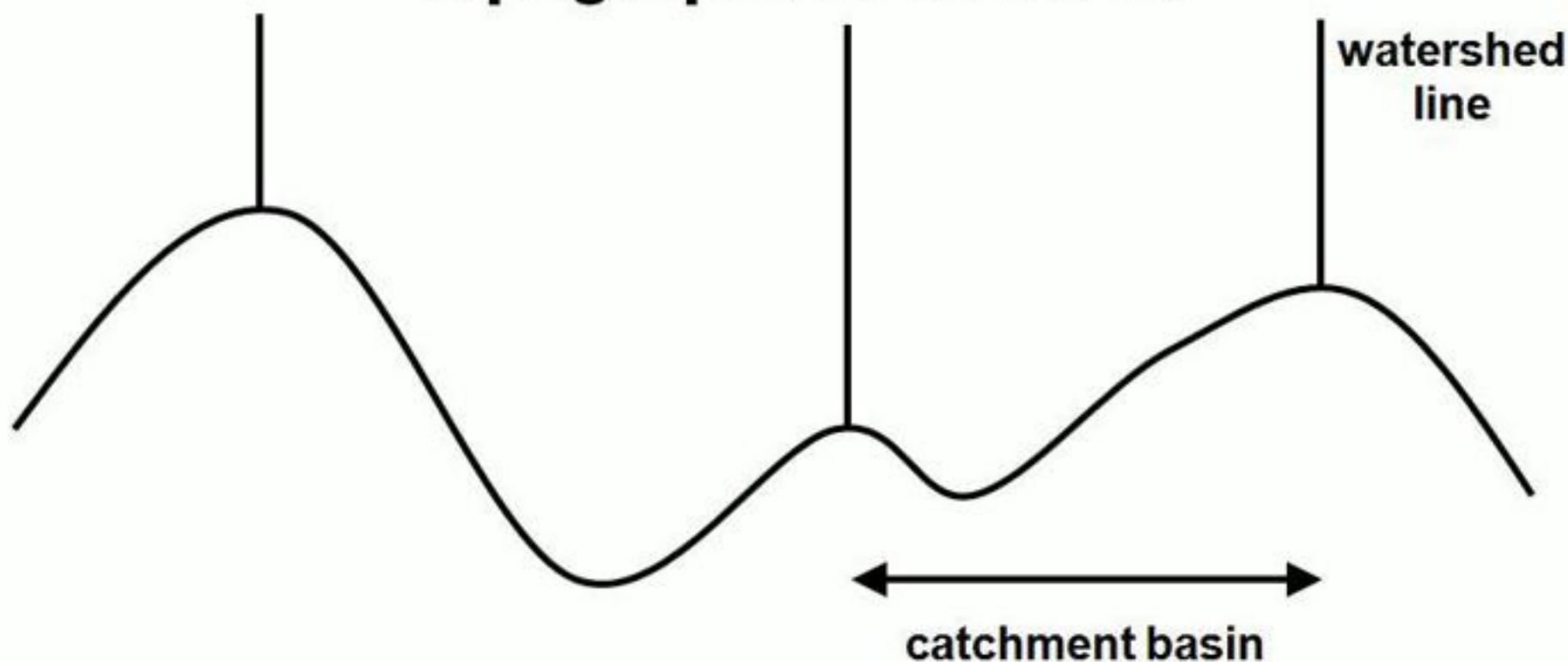
(b)

## Example



16.8	19.2	13.5	20.5	31.2	30.1
18.7	<b>11.8</b>	15.4	18.5	22.1	18.4
20.1	21.9	26.6	20.8	17.3	18.1
25.3	22.8	20.9	19.8	<b>15.1</b>	15.9
30.7	35.5	29.9	18.7	17.6	39.9
34.8	38.6	33.4	32.7	33.5	36.7

Interpret (gradient magnitude) image as topographical surface:

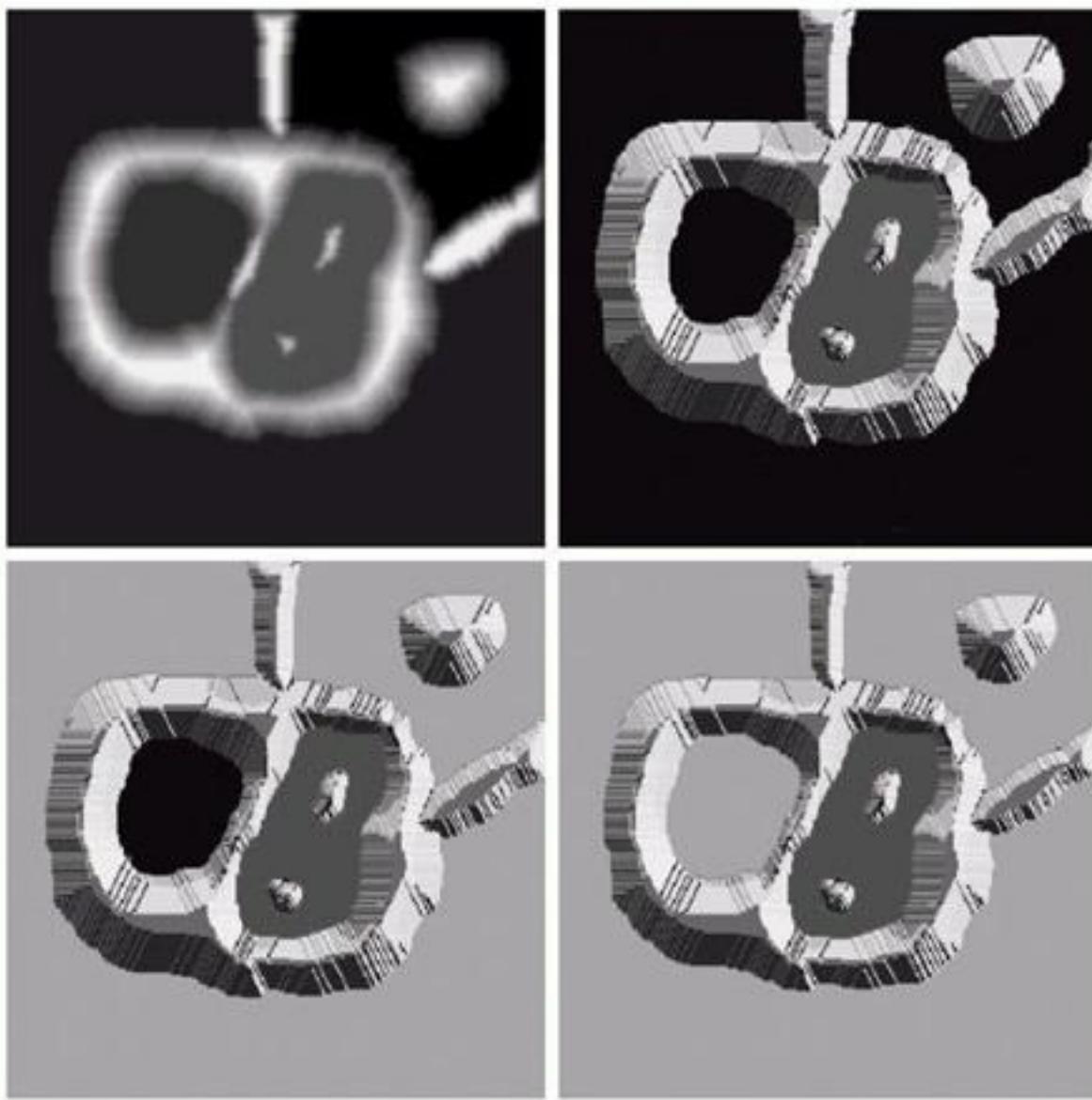


All pixels in catchment basin are connected to minimum by monotonically decreasing path

a  
b  
c  
d

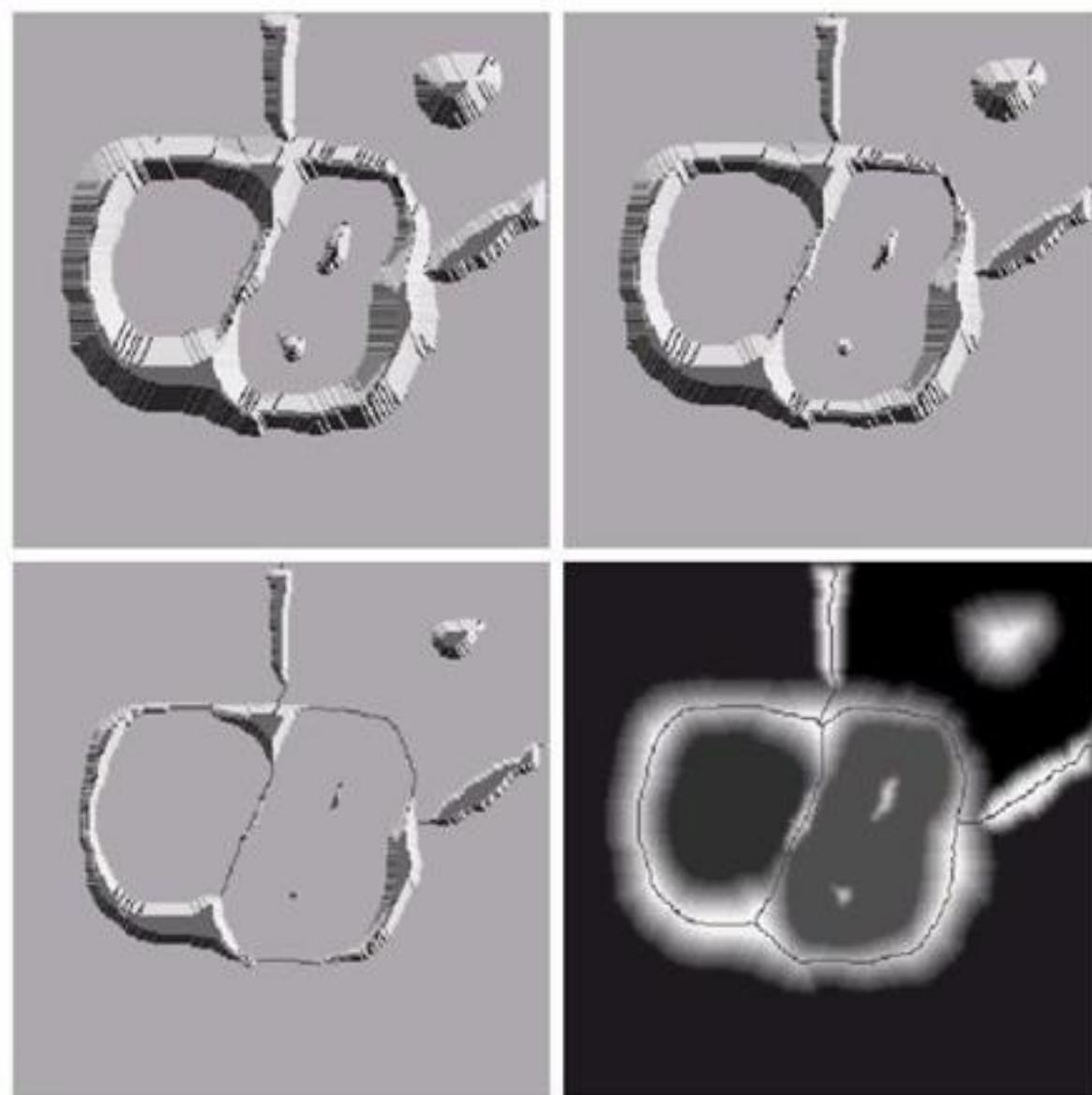
**FIGURE 10.44**

(a) Original image.  
(b) Topographic view. (c)–(d) Two stages of flooding.



A hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a **uniform rate**.

Water from the left basin actually overflowed into the basin on the right and a short dam was built to prevent water from merging at that level of flooding.



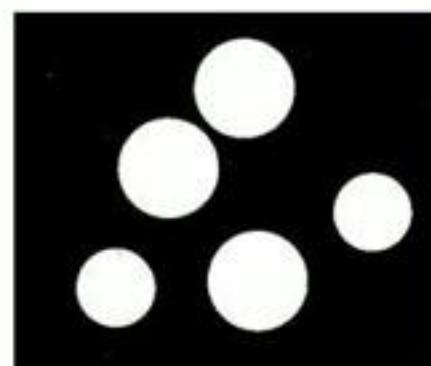
e f  
g h

**FIGURE 10.44**  
*(Continued)*  
(e) Result of further flooding.  
(f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

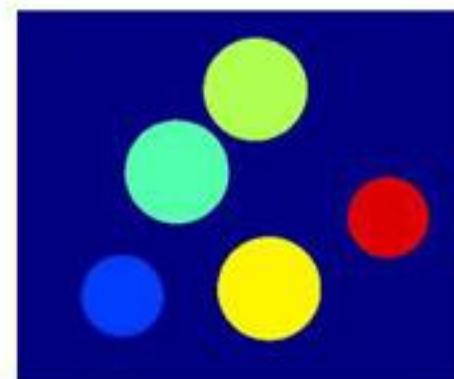
Write the MATLAB script for the given image 'a' to segment as 'b' and label with colors as 'c'. Locate the coin according to their size in ascending order.



'a'



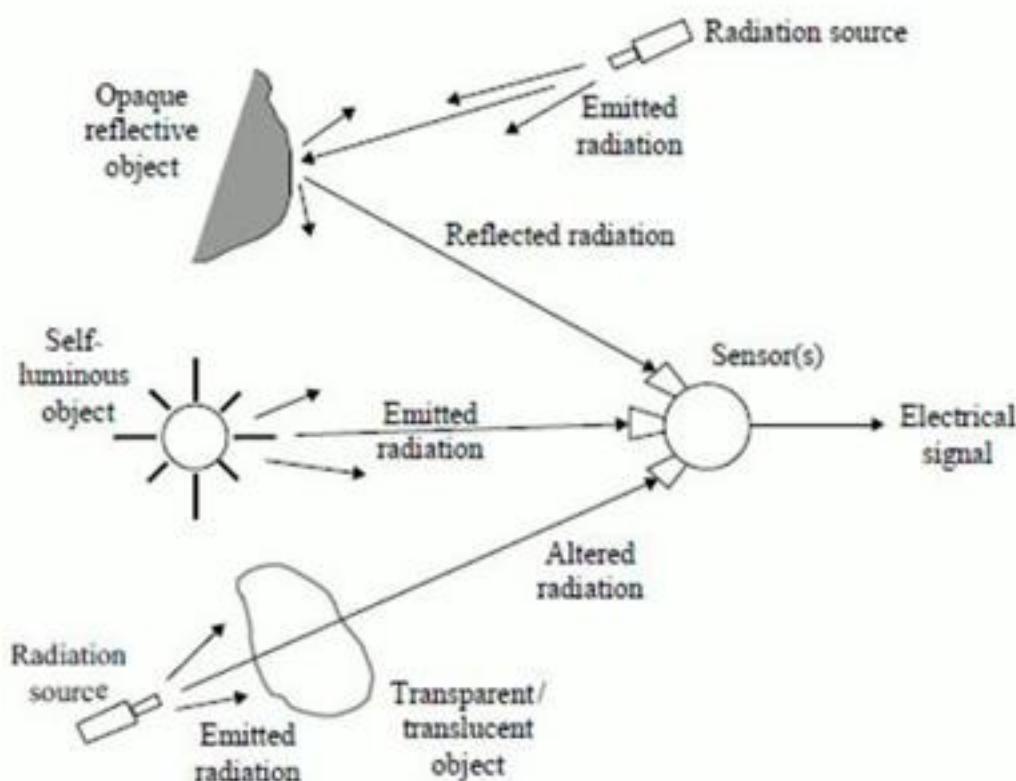
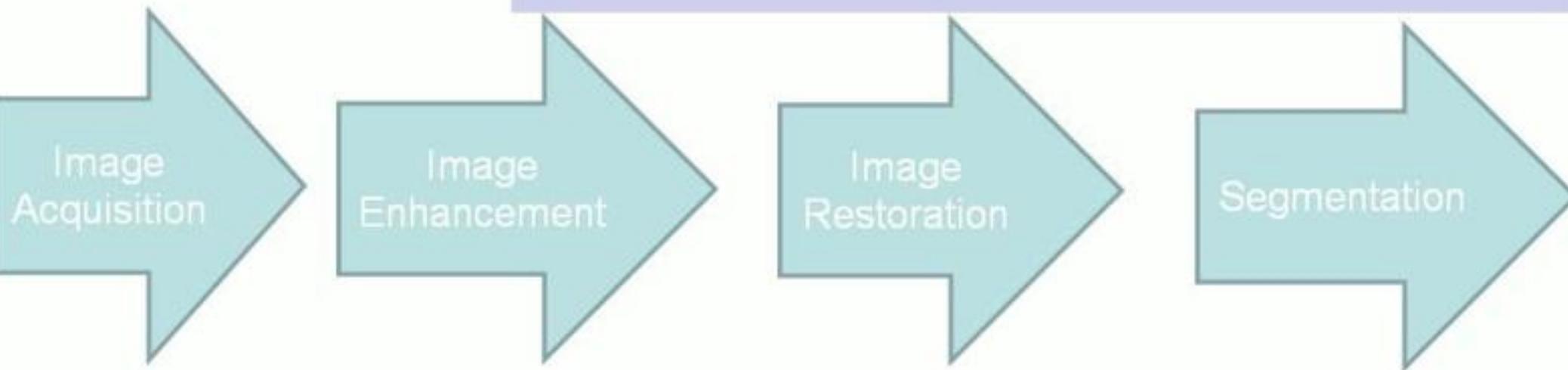
'b'



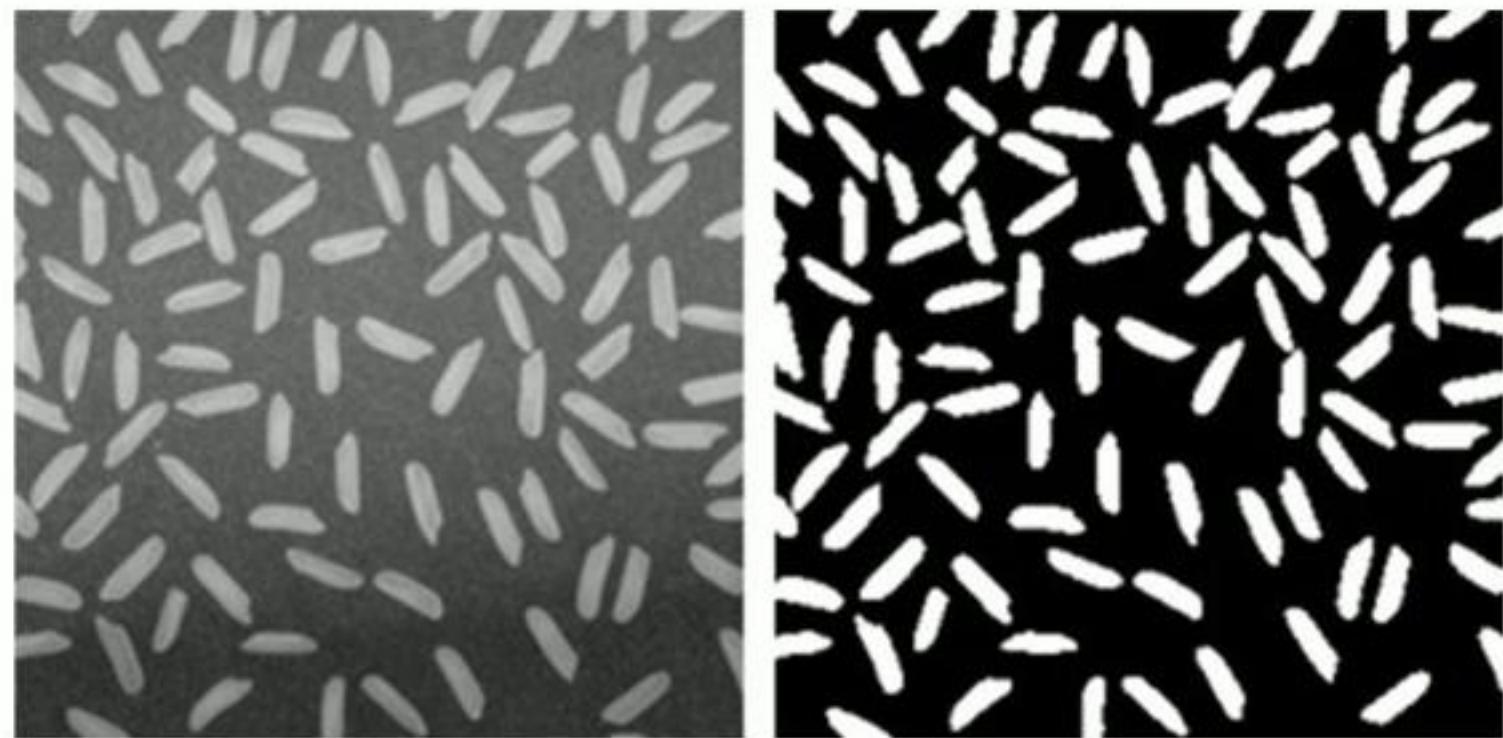
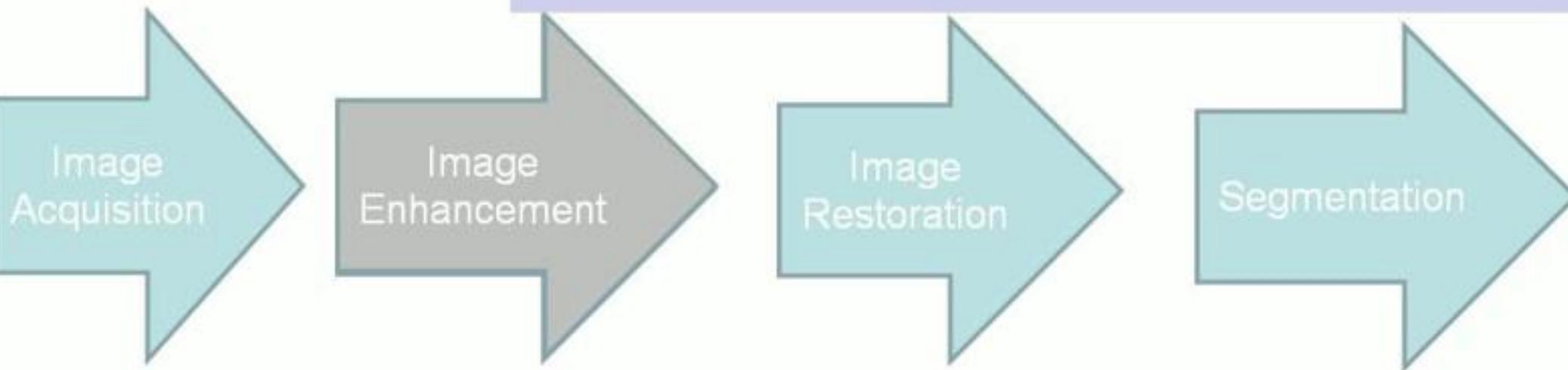
'c'

Image as different size coins

# Image Processing



# Image Processing



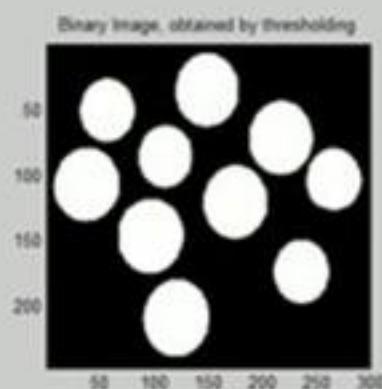
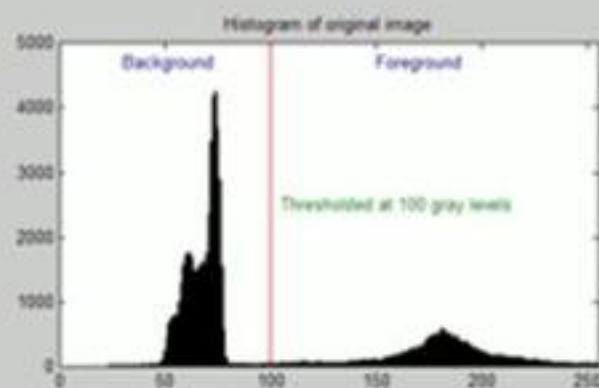
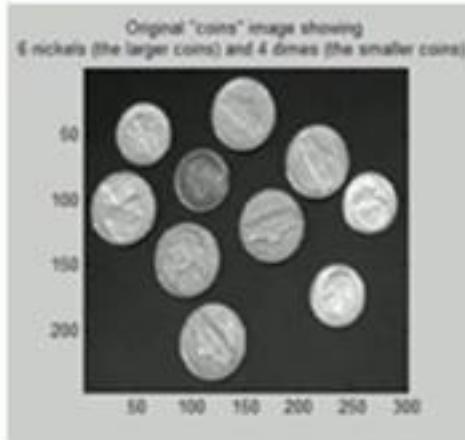
# Image Processing

Image  
Acquisition

Image  
Enhancement

Image  
Restoration

Segmentation



# Image Processing

Image Compression



# Image Processing

Image Compression



**High quality JPEG**  
**File Size: 77.9 kb**



**Medium quality JPEG**  
**File Size: 19.11 kb**

# Image Processing

Representation  
& Description

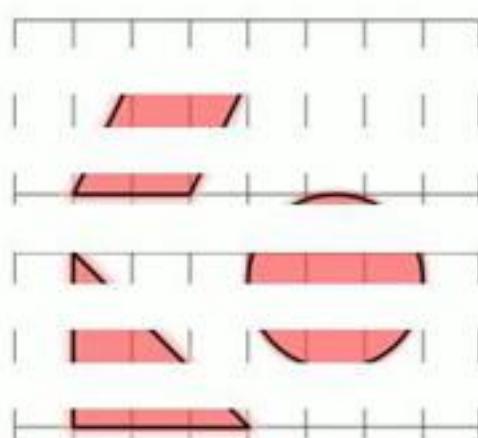
Object  
Recognition

Shape feature detectors and descriptors have been explored and applied in several domains such as biometrics technology, medical image analysis, computer vision, pattern recognition, image processing, etc

# Image Processing

Representation  
& Description

Object  
Recognition

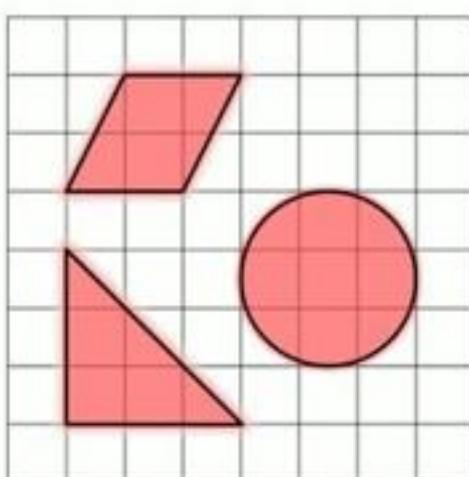


Shape feature detectors and descriptors have been explored and applied in several domains such as biometrics technology, medical image analysis, computer vision, pattern recognition, image processing, etc

# Image Processing

Representation  
& Description

Object  
Recognition



Shape feature detectors and descriptors have been explored and applied in several domains such as biometrics technology, medical image analysis, computer vision, pattern recognition, image processing, etc

## How to solve this problem?

### Sonnet for Lena

O dear Lena, your beauty is so vast.  
It is hard sometimes to describe it fast.  
I thought the entire world I would impress  
If only your portrait I could compress.  
Alas! First when I tried to use VQ  
I found that your cheeks belong to only you.  
Your silky hair contains a thousand lines  
Hard to match with sums of discrete cosines.  
And for your lips, sensual and tactful  
Thirteen Crays found not the proper fractal.  
And while these setbacks are all quite severe  
I might have fixed them with hacks here or there  
But when filters took sparkle from your eyes  
I said, 'Damn all this. I'll just digitize.'

*Thomas Collier*

## Answer: subdivision

In general, segmentation problem requiring multiple thresholds are best solved by local properties.

- When *Threshold* depends only on pixel intensity then it is called as global.
- If it depends on the spatial coordinates  $x$  and  $y$ , then threshold is called dynamic or adaptive.

## Sonnet for Lena

O dear Lena, your beauty is so vast.  
It is hard sometimes to describe it fast.  
I thought the entire world I would impress  
If only your portrait I could compress.  
Alas! First when I tried to use VQ  
I found that your cheeks belong to only you.  
Your silky hair contains a thousand lines  
Hard to match with sums of discrete cosines.  
And for your lips, sensual and tactful  
Thirteen Crays found not the proper fractal.  
And while these setbacks are all quite severe  
I might have fixed them with hacks here or there  
But when filters took sparkle from your eyes  
I said, 'Damn all this. I'll just digitize.'

*Thomas Collyer*

## Basic Adaptive Thresholding

- The thresholding may be viewed as an operation that involves tests against a function  $T$  of the form

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

where  $f(x, y)$  is the gray-level of point  $(x, y)$  and  $p(x, y)$  denotes some local property of this point.

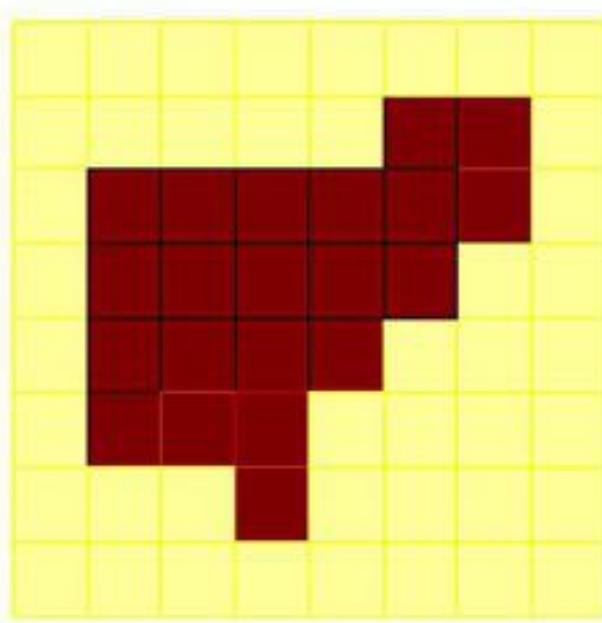
- When  $T$  depends only on  $f(x, y)$  the threshold is called global. If  $T$  depends on both  $f(x, y)$  and  $p(x, y)$ , the threshold is called local.
- A threshold image  $g(x, y)$  is defined as

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$

Thus, pixels labeled '1' correspond to objects, whereas pixels labeled '0' correspond to the background.

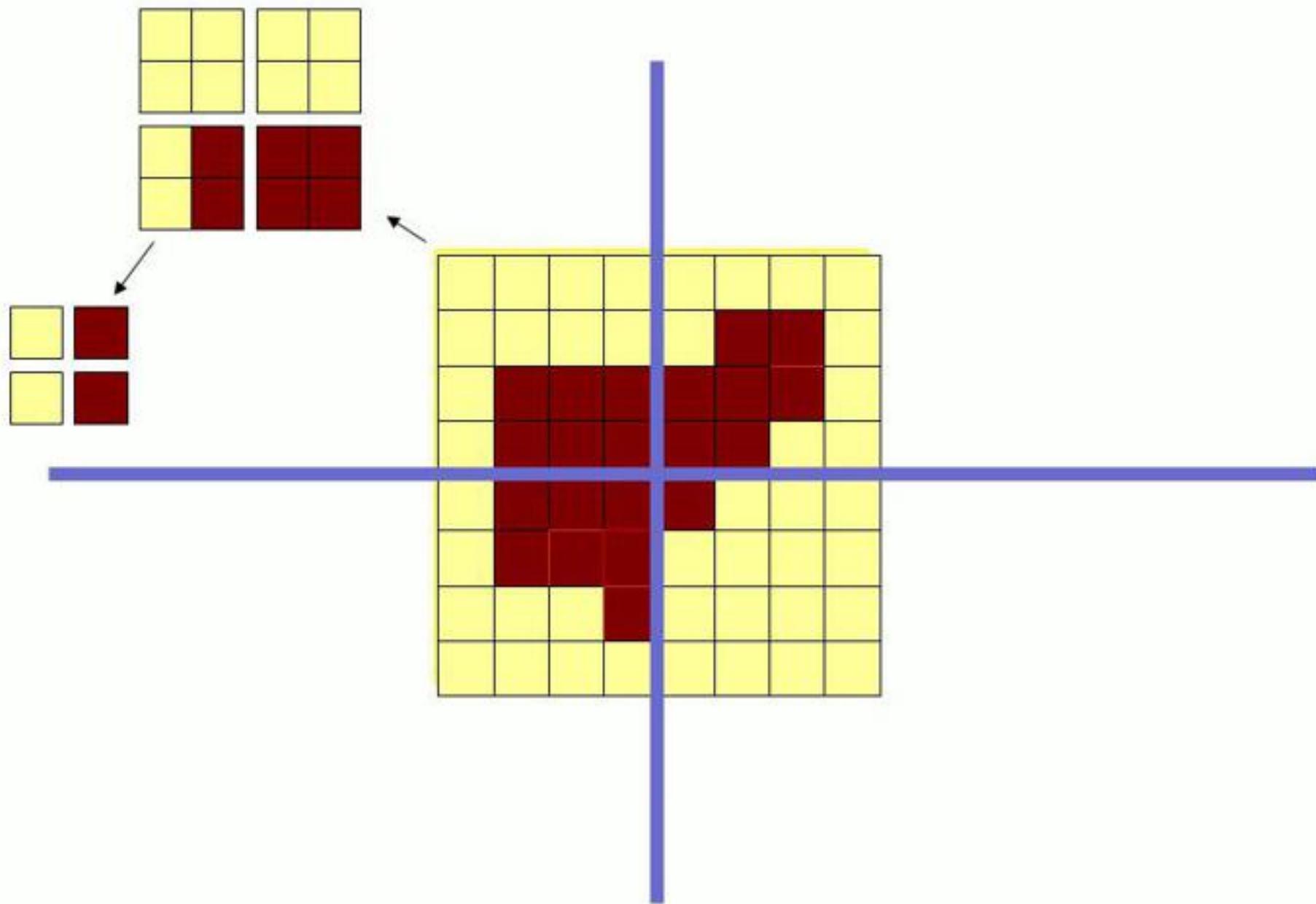
# Splitting

# Region Splitting and Merging



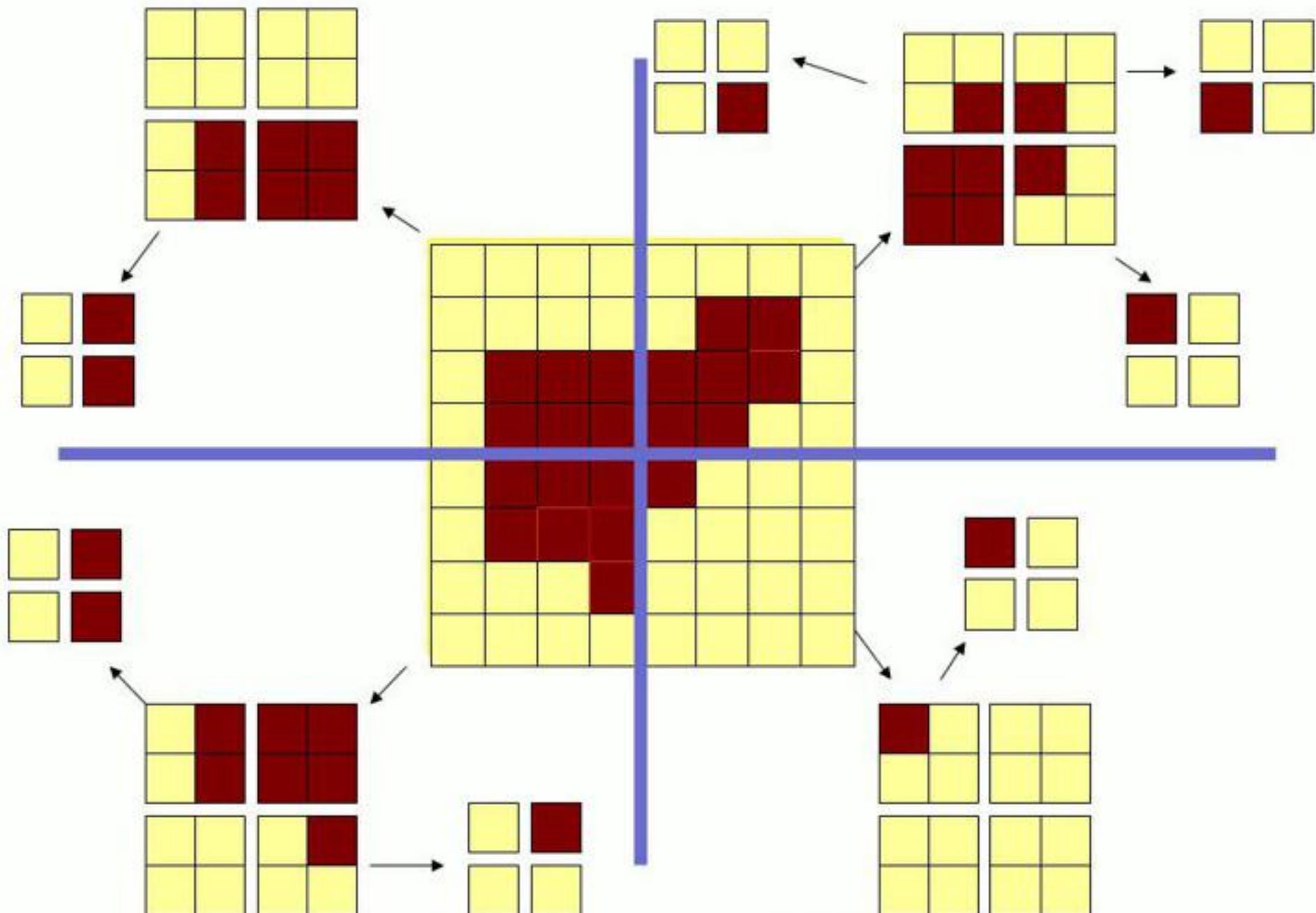
## Splitting

# Region Splitting and Merging



## Splitting

# Region Splitting and Merging

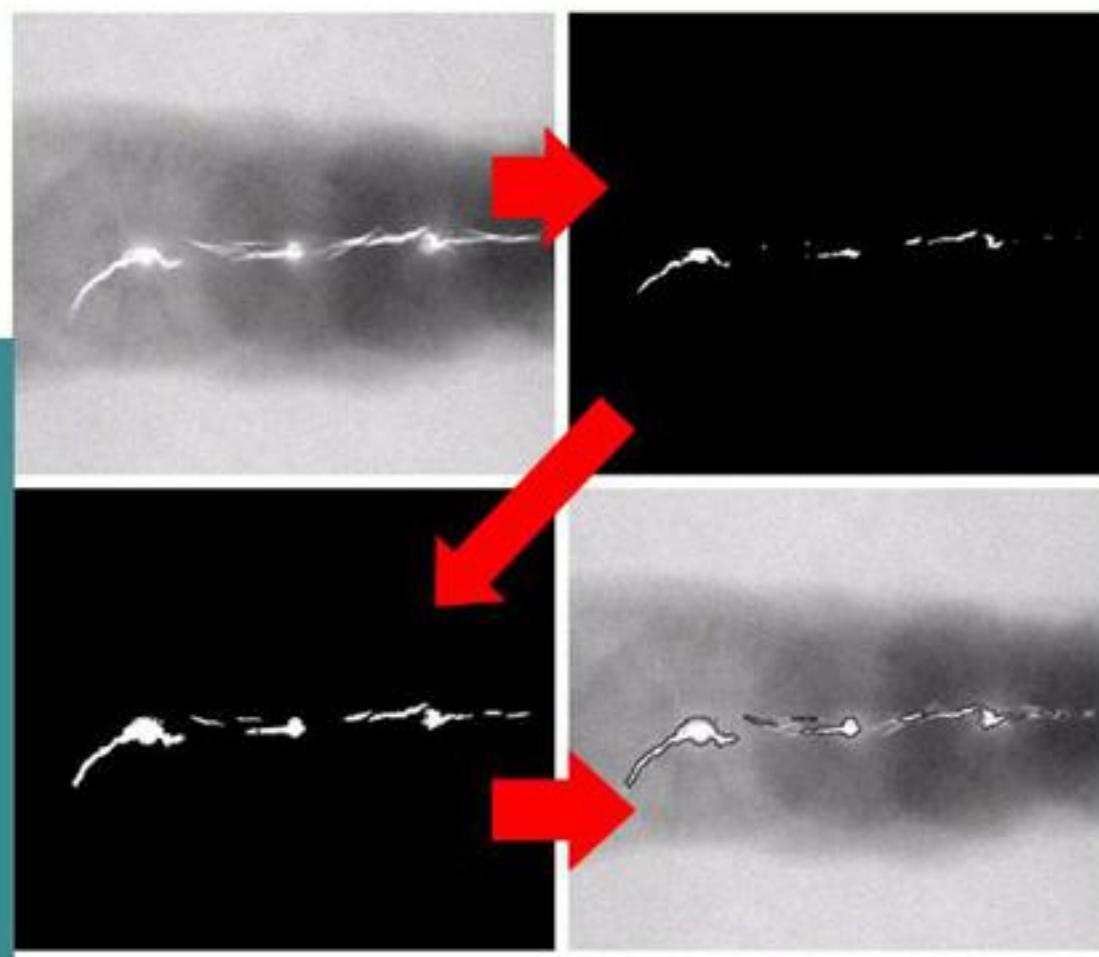


## Example: 10.16

To use region growing to segment the regions of the welding failures.

An X-ray image of a welding, the pixels of defective welds tend to have values of 255.

Criteria for region growing:  
(1) The absolute gray-level difference between any pixel and the seed had to be less than 65.  
(2) To be included in one of the regions, the pixel had to be 8-connected to at least one pixel in that region



a  
b  
c  
d

FIGURE 10.40

(a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).

# Region Splitting and Merging

We define  $P(R_i) = \text{TRUE}$  if at least 80% of the pixels in  $R_i$  have the property-

$$|z_j - m_i| \leq 2\sigma_i,$$

where  $z_j$  is the gray level of the  $j$ -th pixel in  $R_i$ ,  $m_i$  is the mean gray level of that region, and  $\sigma_i$  is the standard deviation of the gray levels in  $R_i$ .

The shading and the stem of the leaf were erroneously eliminated by the thresholding procedure.



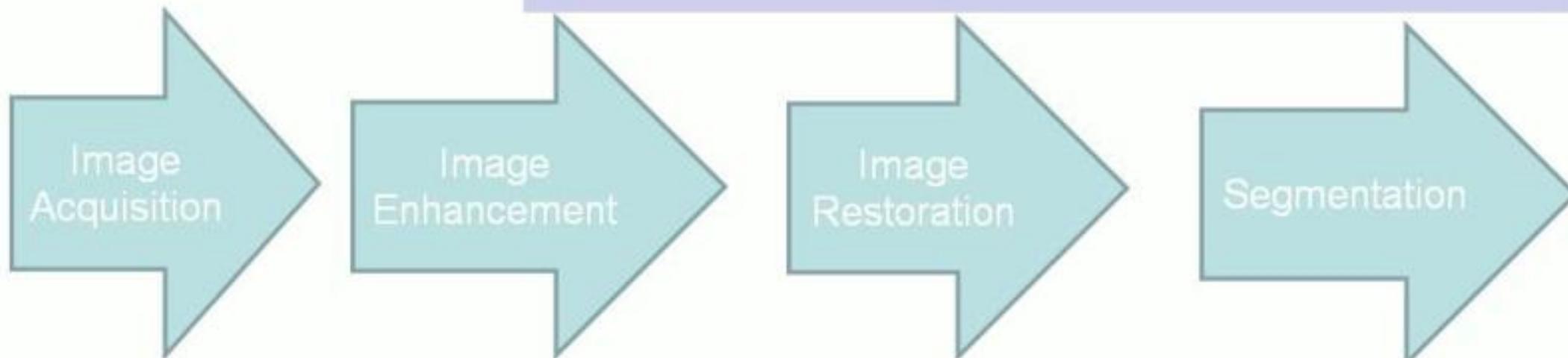
**FIGURE 10.43**

- (a) Original image. (b) Result of split and merge procedure. (c) Result of thresholding (a).

# **Chapter 11**

# **Image Description and Representation**

# Image Processing



↗ Radiation source

reflective  
object

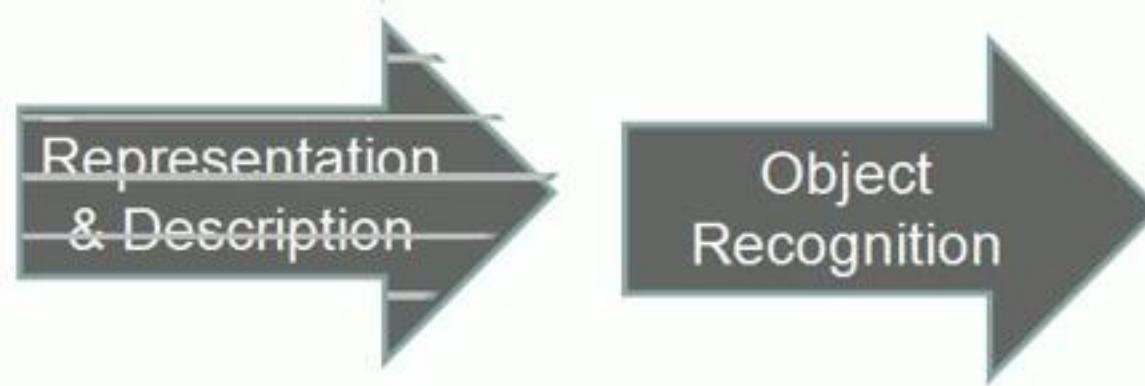
radiation

object ————— Emitted radiation —————> Electrical signal

Altered  
radiation

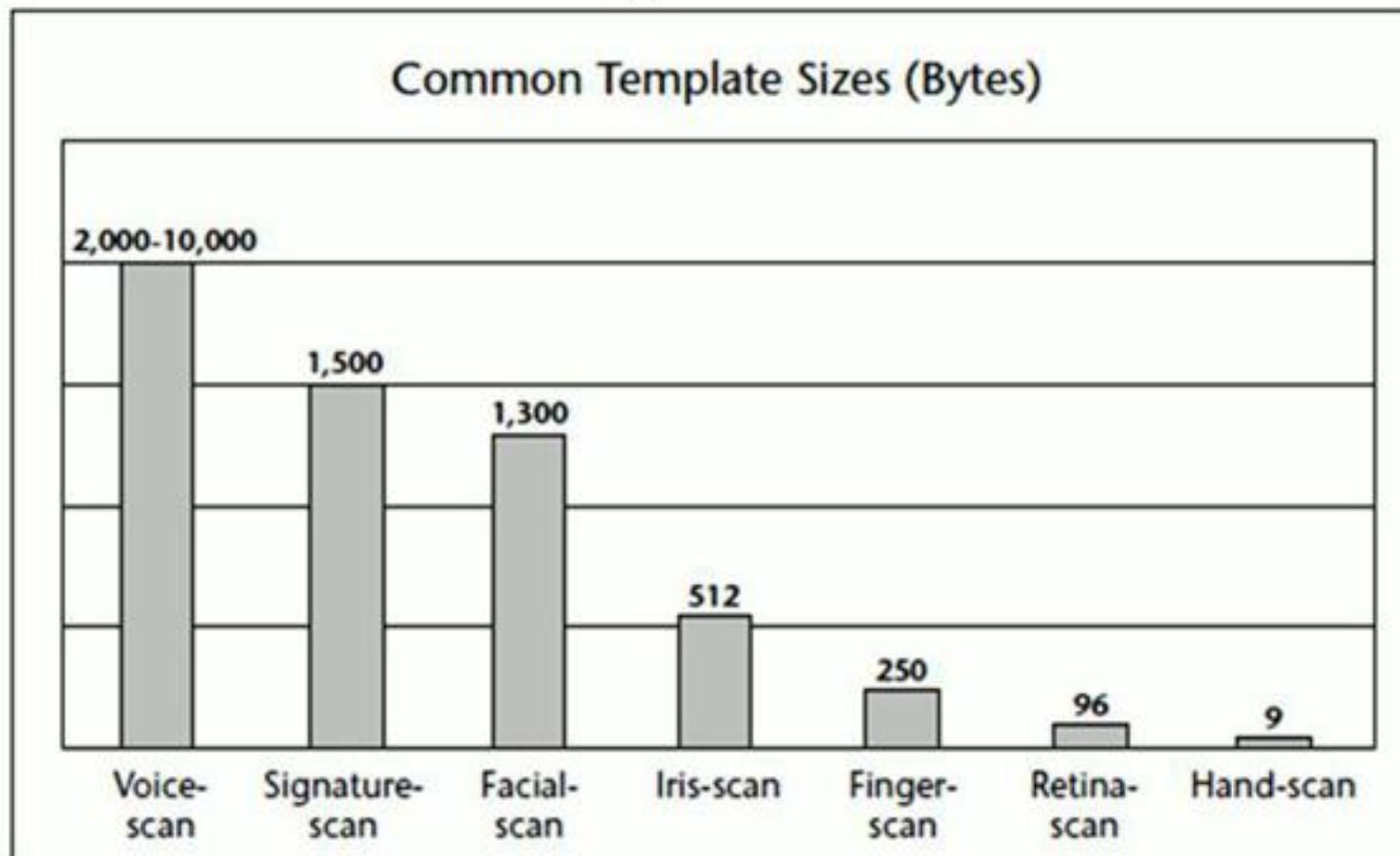
source ————— Transparent/  
translucent —————

# Image Processing



Shape feature detectors and descriptors have been explored and applied in several domains such as biometrics technology, medical image analysis, computer vision, pattern recognition, image processing, etc

## Template Sizes



# Image Representation and Description

## Objective:

To represent and describe segmented or extracted Feature information in an image in other forms that are more suitable than the image itself.

## Benefits:

- Easier to understand
- Require fewer memory, faster to be processed
- More “ready to be used”

## Features can be defined by –

- External characteristics
- Internal Characteristics
- When the feature describe the shape characteristic. It should be insensitive to variation in size, translation and rotation.
- When the feature describe enclosed regions characteristic such as color, texture etc.



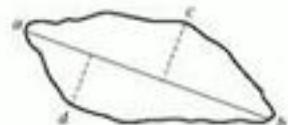
## Boundary Descriptors

- There are several simple geometric measures that can be useful for describing a boundary.
- **The length of a boundary:** the number of pixels along a boundary gives a rough approximation of its length.
- The **size of smallest circle or box** that can totally enclosing the object



## Boundary Descriptors

- **Curvature:** the rate of change of slope
- To measure a curvature accurately at a point in a digital boundary is difficult
- The difference between the slopes of adjacent boundary segments is used as a descriptor of curvature at the point of intersection of segments



# Boundary Descriptors

- There are several simple geometric measures that can be useful for describing a boundary.
  - ❖ Simple Descriptors
  - ❖ Shape numbers
  - ❖ Fourier descriptor
  - ❖ Statistical moments

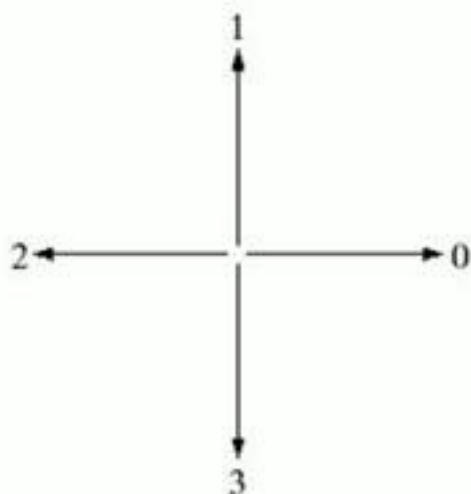
# Simple Descriptors

- ❖ Length
  - ❖ Diameter
  - ❖ Major – minor axis
  - ❖ Basic rectangle
- 
- ❖ Eccentricity
  - ❖ Curvature
  - ❖ Convex-concave
  - ❖ Corner ( $> 90^\circ$ )

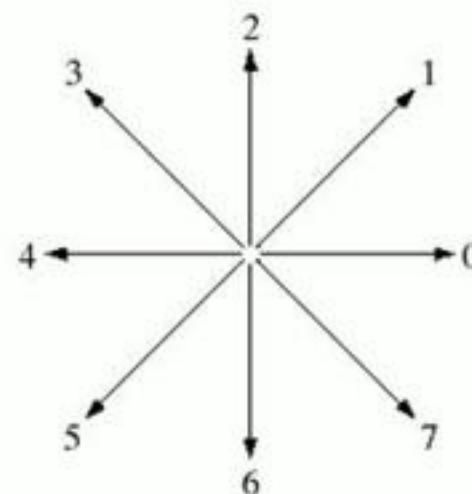


# Shape Representation by Using Chain Codes

Chain code present an object boundary by a connected sequence of straight line segments of specified length and direction.



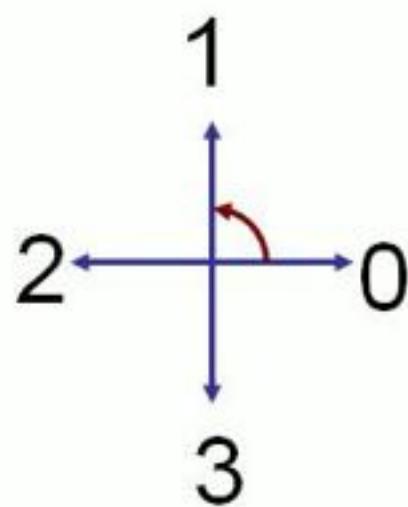
4-directional  
chain code



8-directional  
chain code

# Shape numbers and order of Feature

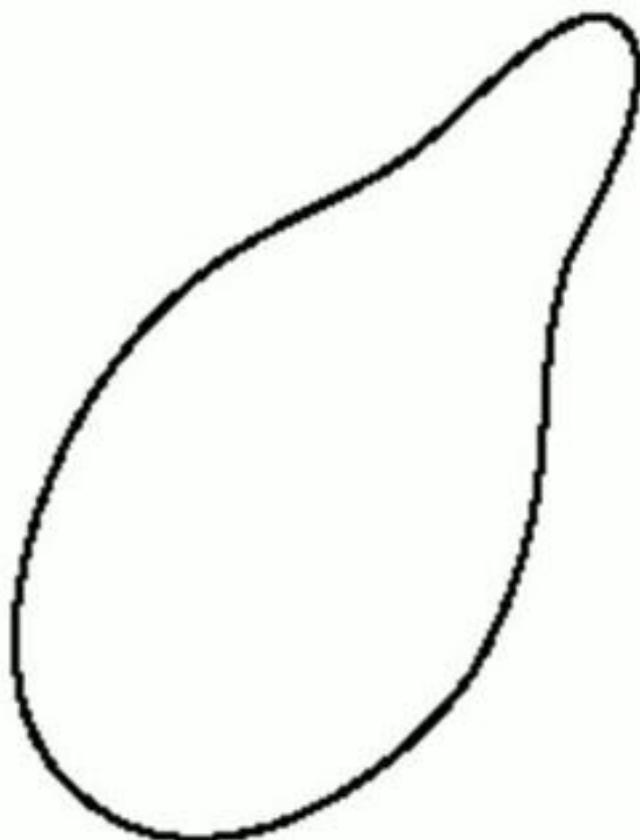
- ❖ Shape numbers - the first difference of smallest magnitude
- ❖ Order n:- the number of digits in the sequence



	Order 4	Order 6
Chain code:	0 3 2 1	0 0 3 2 2 1
Difference:	3 3 3 3	3 0 3 3 0 3
Shape no.:	3 3 3 3	0 3 3 0 3 3

## Example: 11.2

Find out the shape number and order of given figure,  
assume the major axis has length is six boxes

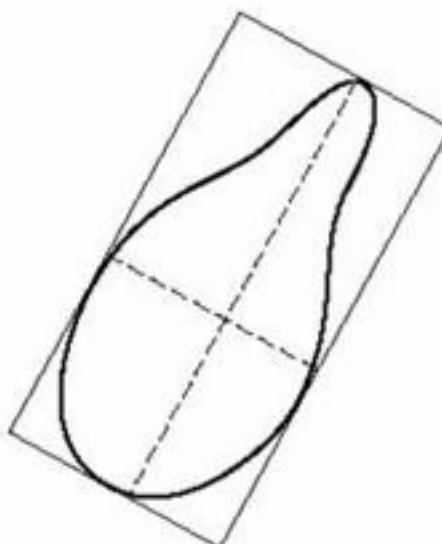


## Example: 11.2

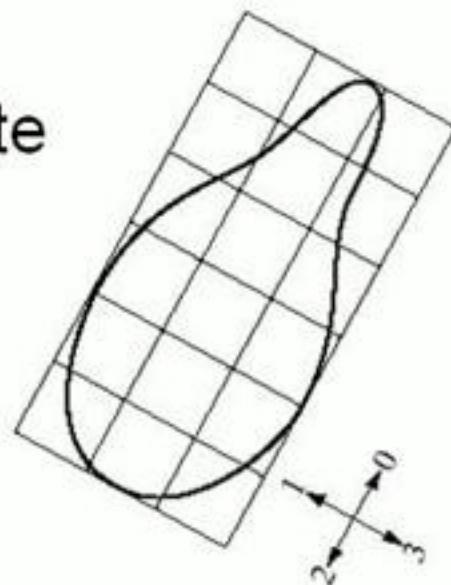
1. Original boundary



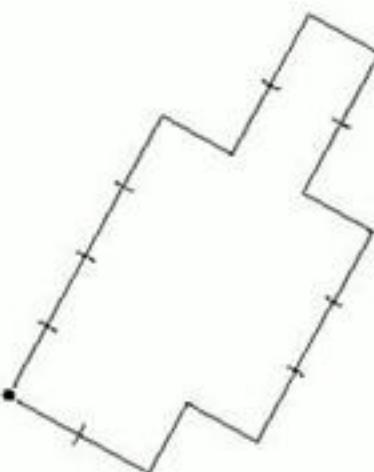
2. Find the smallest rectangle that fits the shape



3. Create grid



4. Find the nearest Grid.

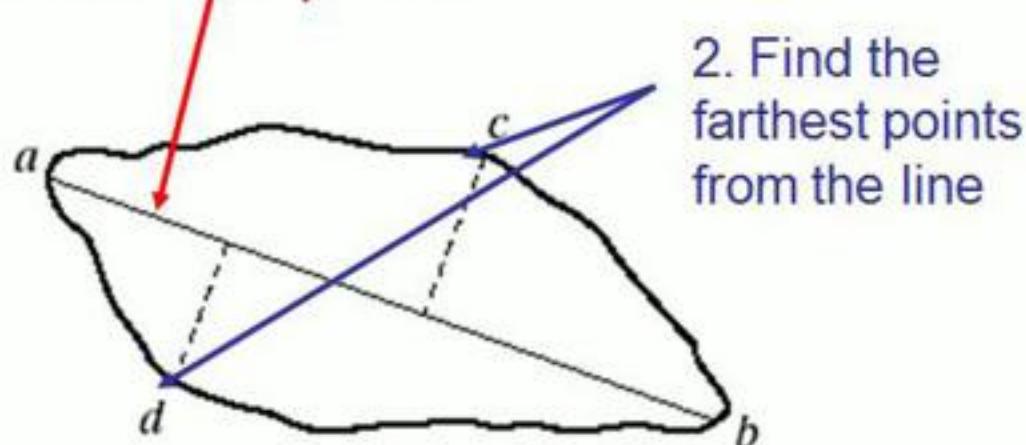


# Splitting Techniques

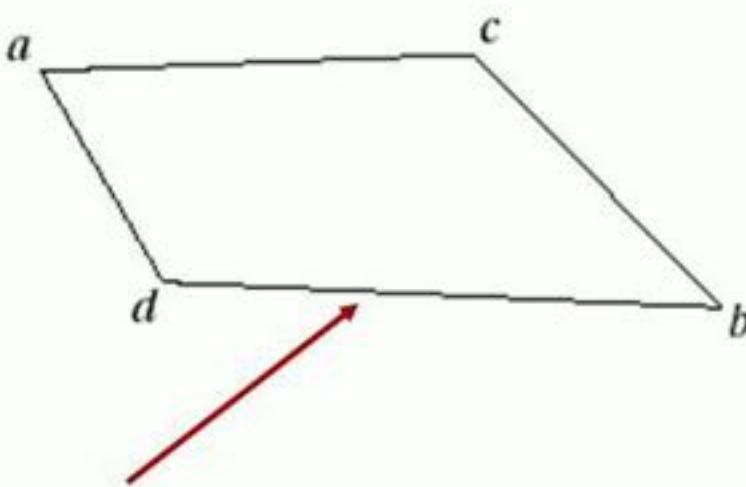
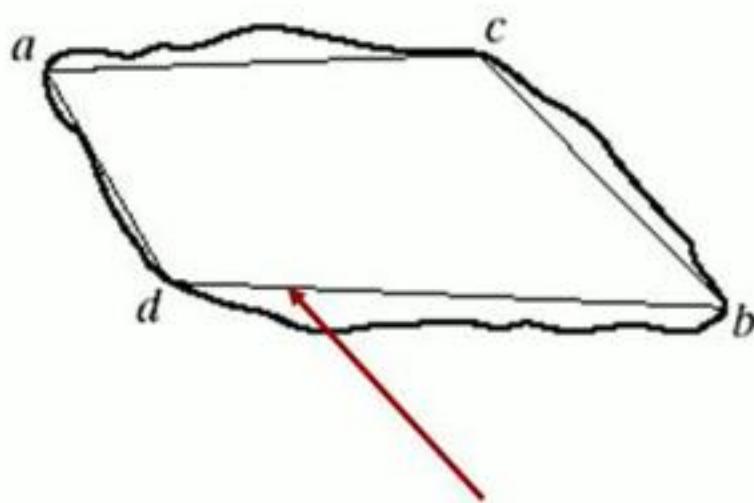
0. Object boundary



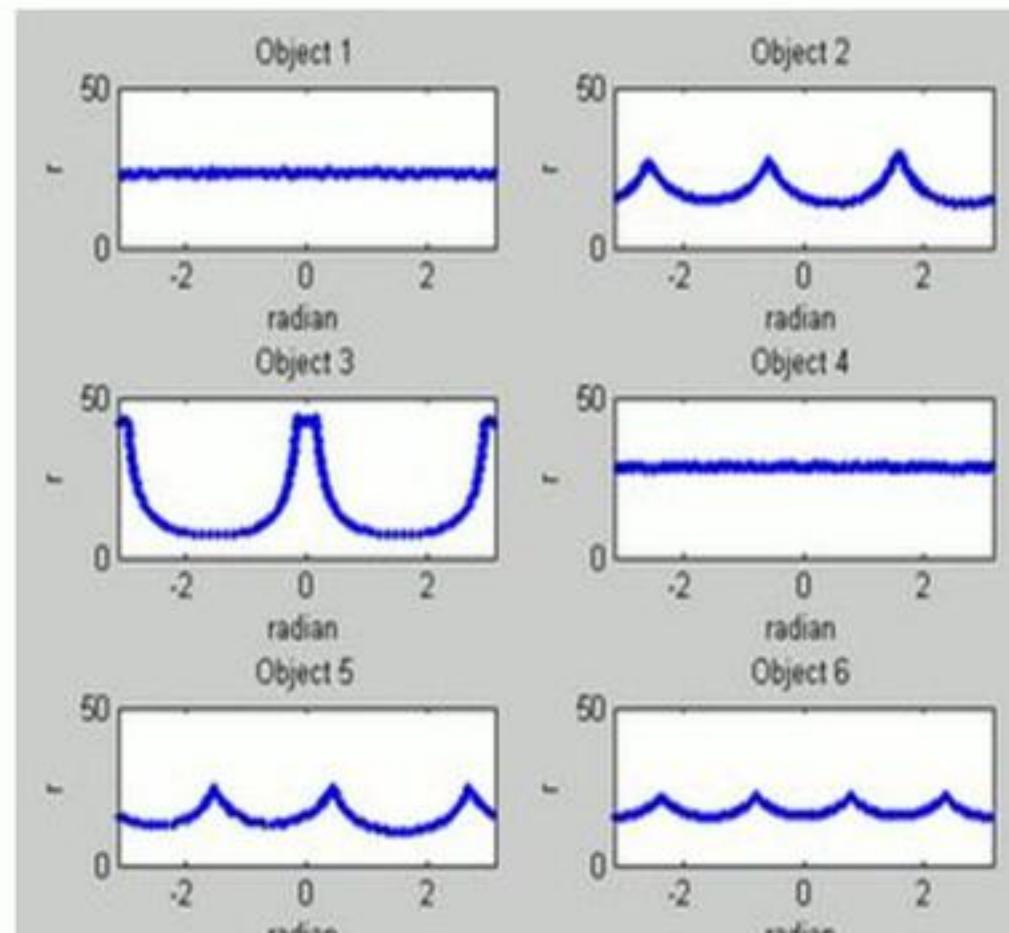
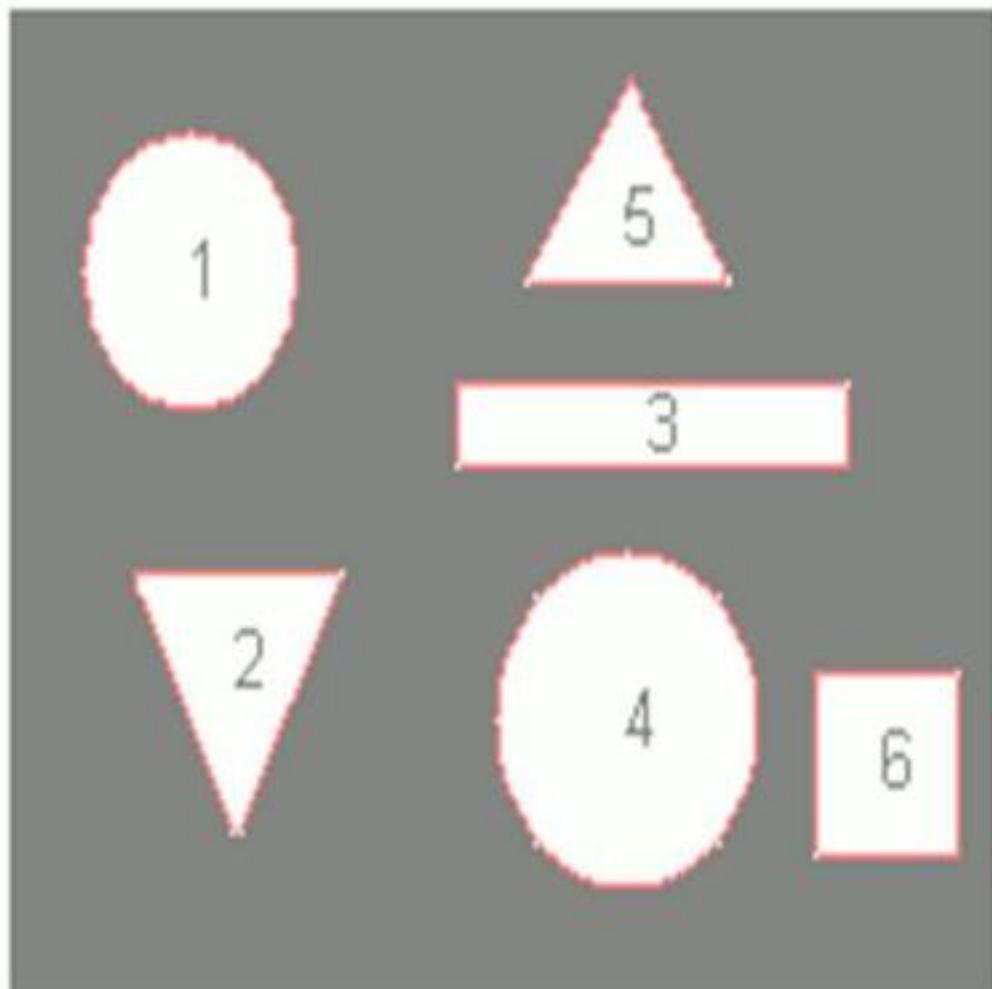
1. Find the line joining two extreme points



2. Find the farthest points from the line

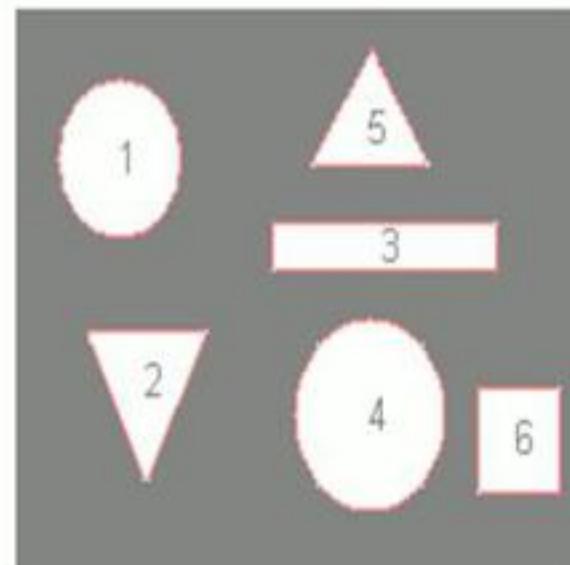
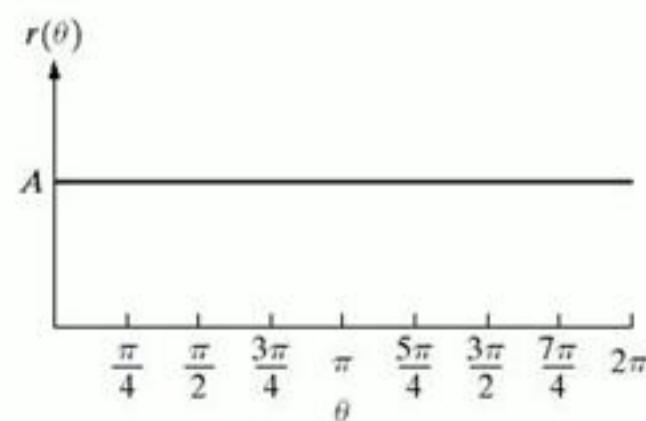
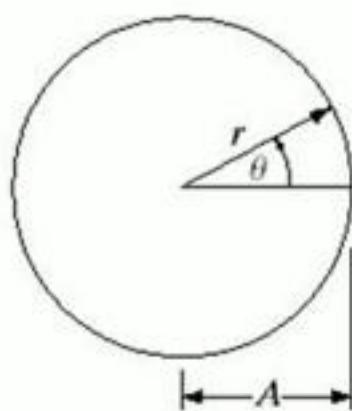


# Distance-Versus-Angle Signatures



# Image Representation and Description

## Distance-Versus-Angle Signatures



用 官 刀 三

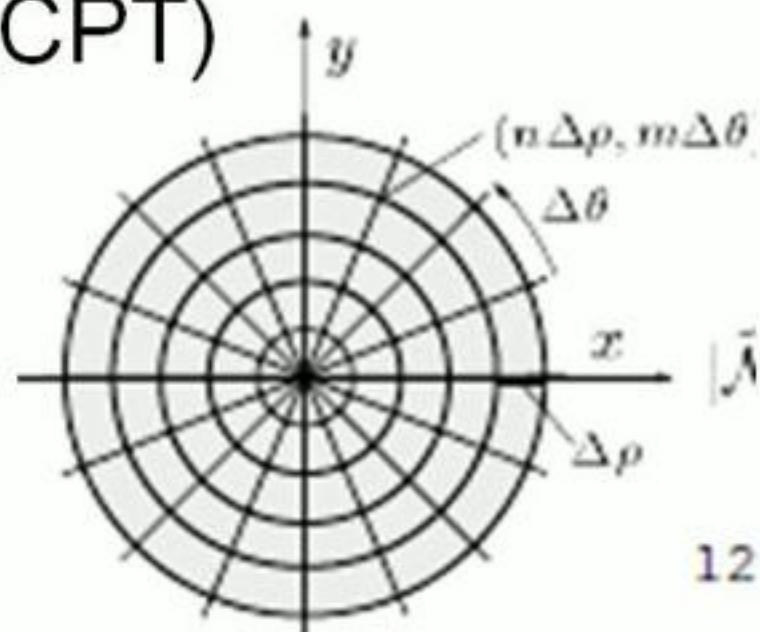
Chinese characters

## Central Projection Transform (CPT)



$$f(\theta_m) = \sum_n f(\rho_n \cos \theta_m, \rho_n \sin \theta_m)$$

where  $\theta_m \in [0, 2\pi]$ ,  $m = 1, 2, 3, \dots, 360^0$



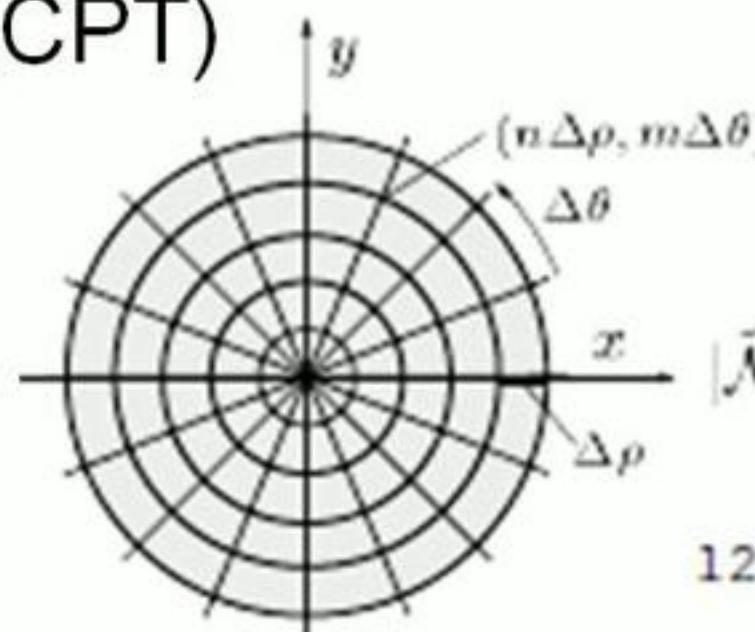
# Image Representation and Description

## Central Projection Transform (CPT)



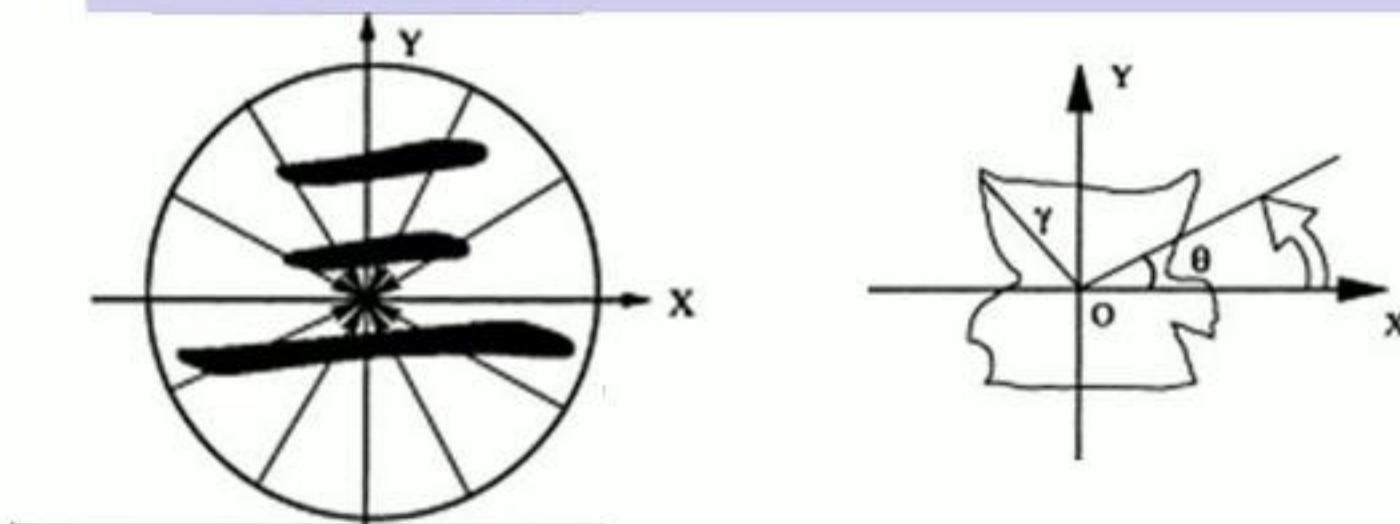
$$f(\theta_m) = \sum_n f(\rho_n \cos \theta_m, \rho_n \sin \theta_m)$$

where  $\theta_m \in [0, 2\pi]$ ,  $m = 1, 2, 3, \dots, 360^0$



# Image Representation and Description

三



Extraction of rotation invariant signature based  
on fractal geometry

Y. Tao; T.R. Ioerger; Y.Y. Tang

Proceedings 2001 International Conference on Image

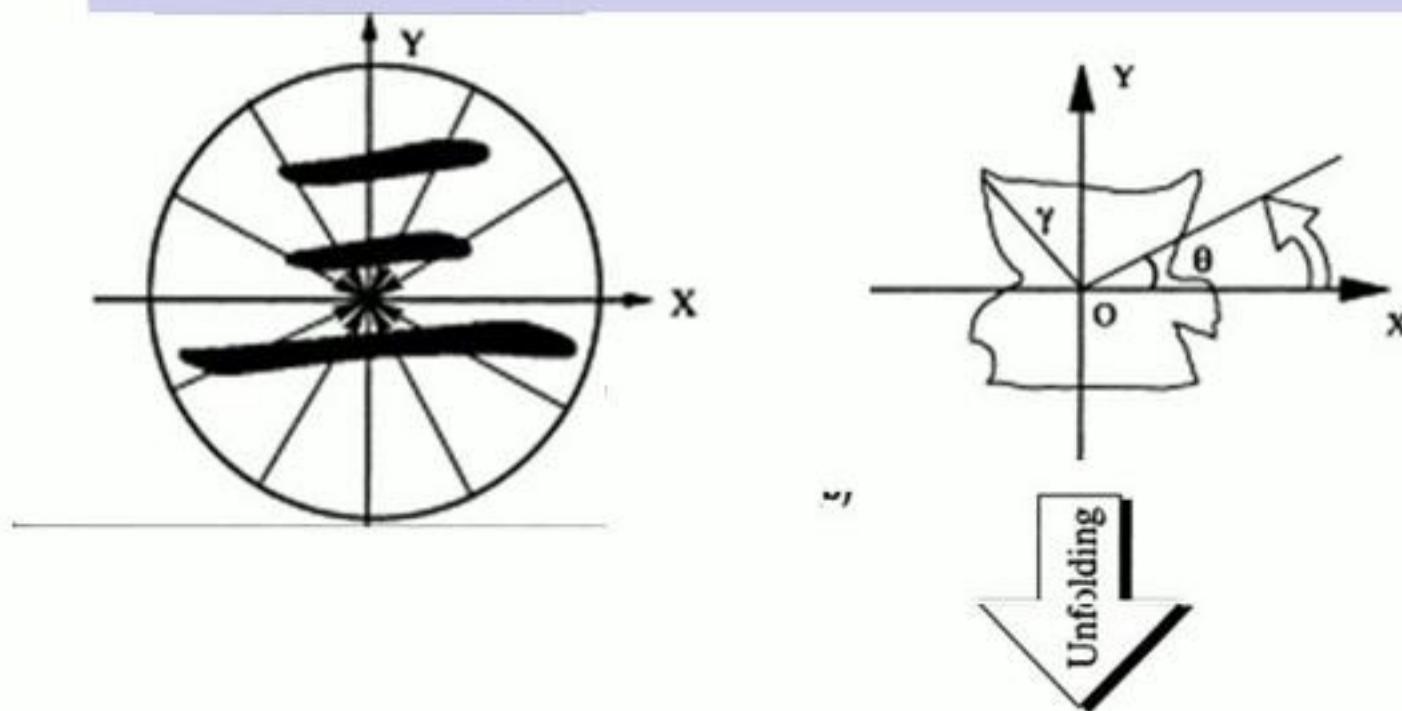
Processing (Cat. No.01CH37205)

Year: 2001 | Volume: 1 | Conference Paper |

Publisher: IEEE

# Image Representation and Description

三



Extraction of rotation invariant signature based on fractal geometry

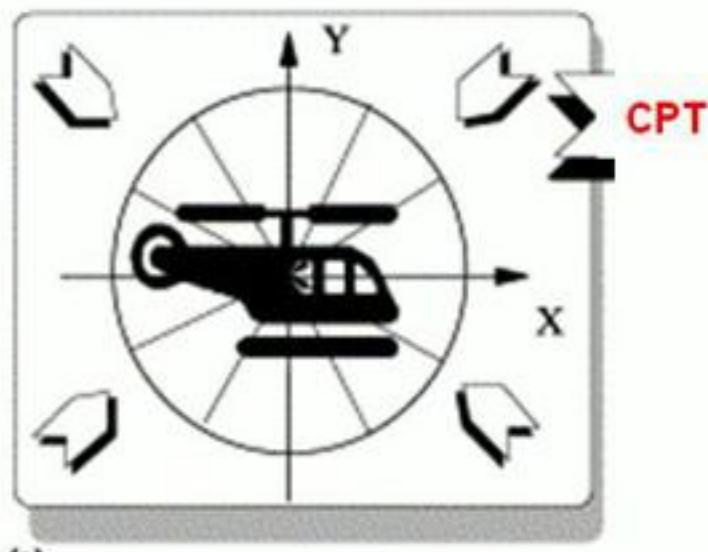
Y. Tao; T.R. Ioerger; Y.Y. Tang

Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)

Year: 2001 | Volume: 1 | Conference Paper |

Publisher: IEEE

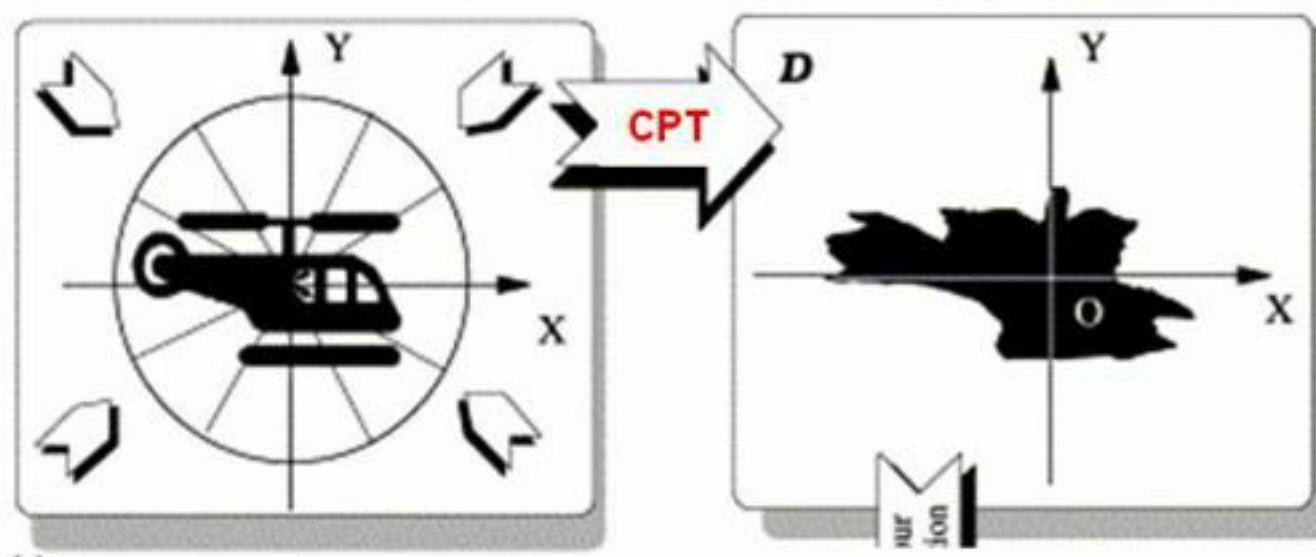
# Image Representation and Description



EXTRACTION  
OF ROTATION  
INVARIANT  
SIGNATURE  
BASED ON  
FRACTAL  
GEOMETRY

By Yuan Y.  
Tang

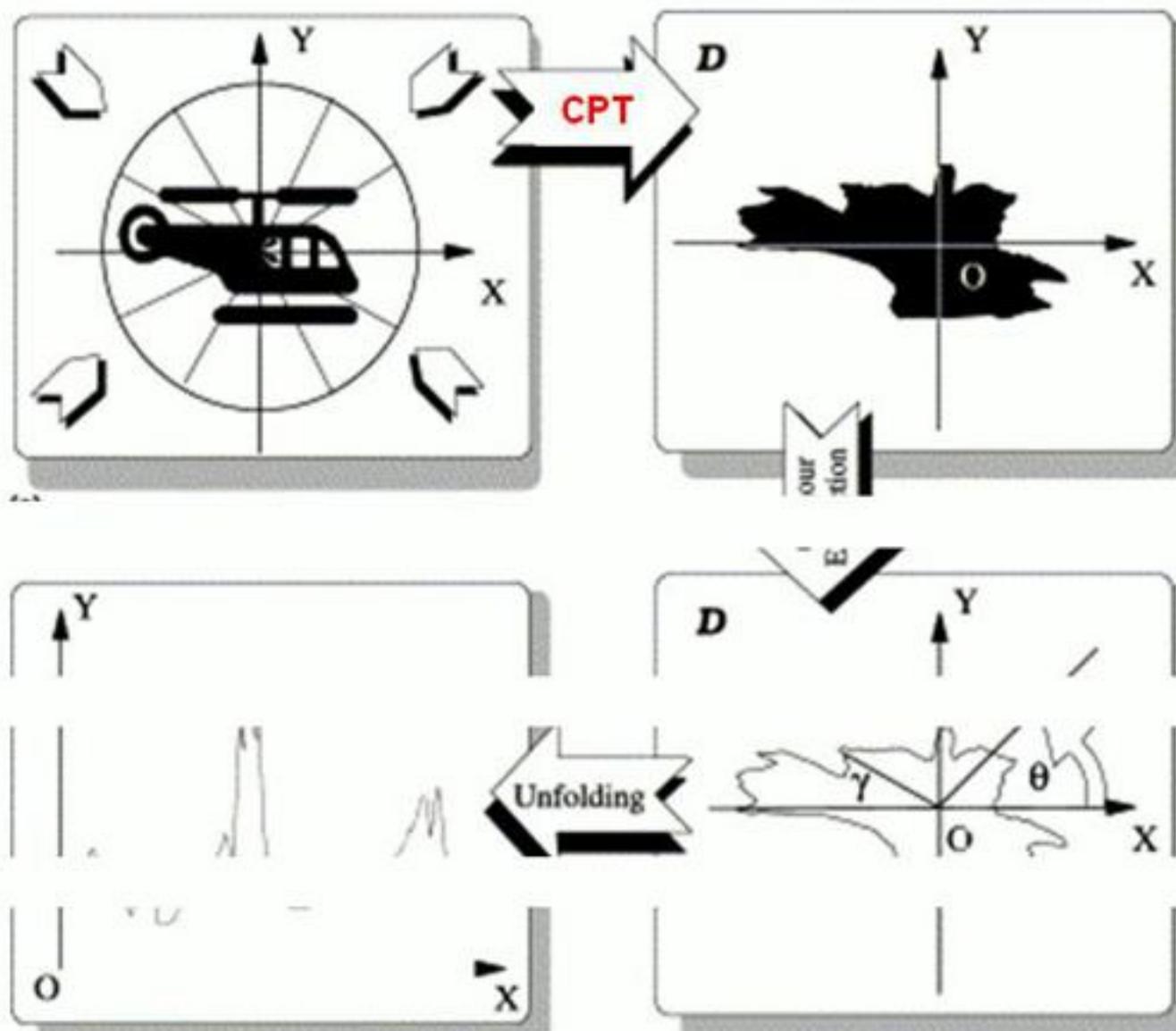
# Image Representation and Description



EXTRACTION  
OF ROTATION  
INVARIANT  
SIGNATURE  
BASED ON  
FRACTAL  
GEOMETRY

By Yuan Y.  
Tang

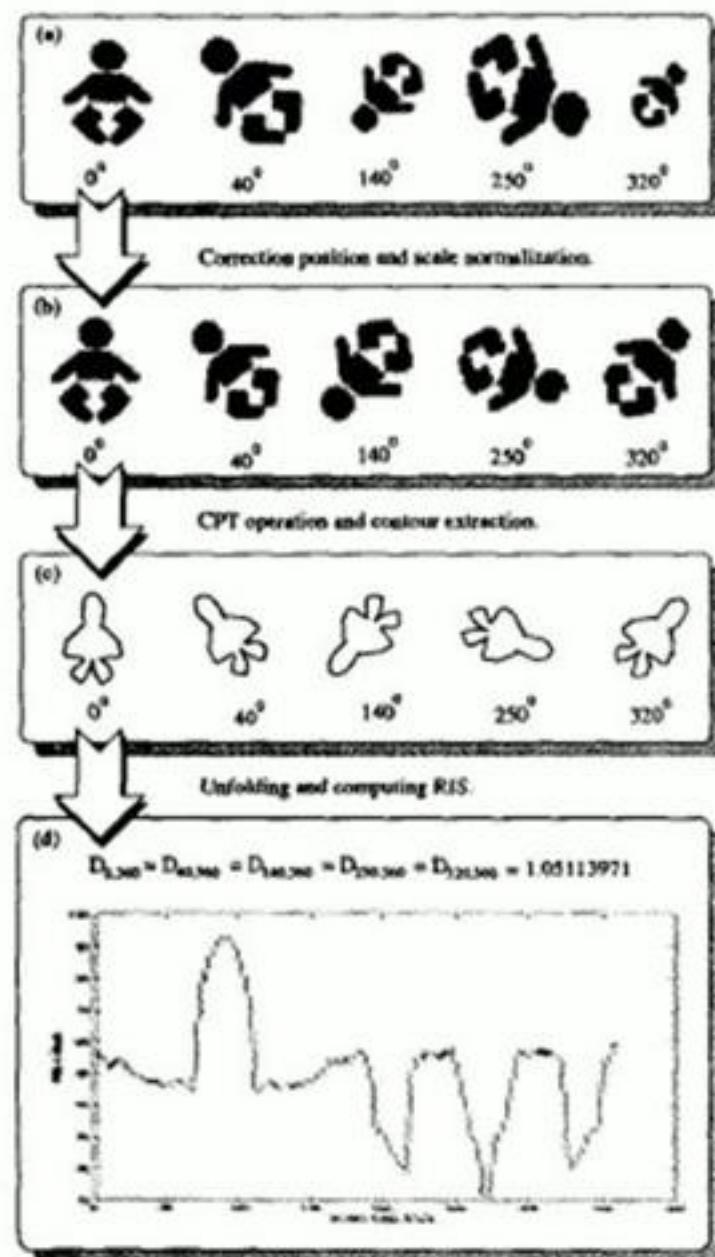
# Image Representation and Description



EXTRACTION  
OF ROTATION  
INVARIANT  
SIGNATURE  
BASED ON  
FRACTAL  
GEOMETRY

By Yuan Y.  
Tang

# Image Representation and Description



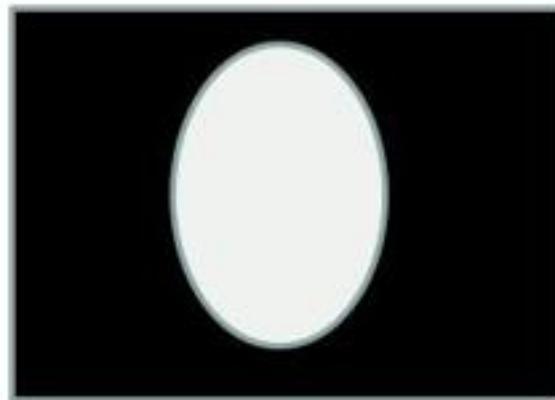
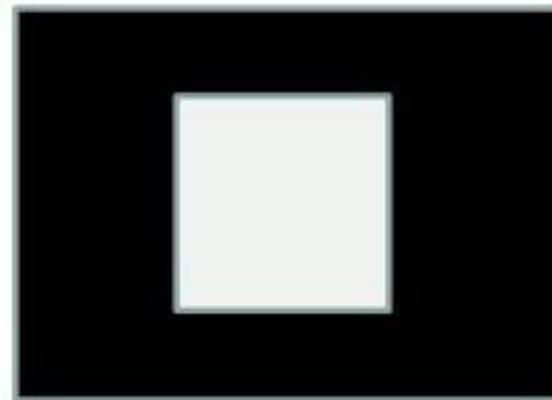
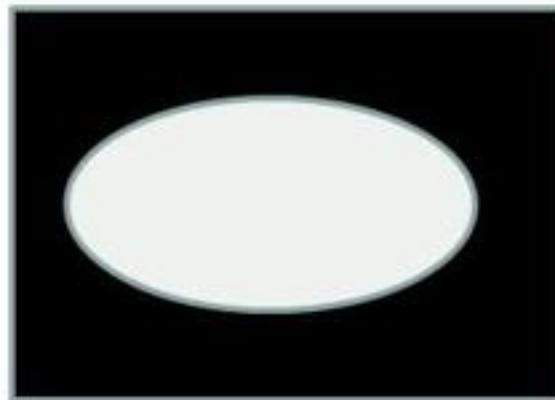
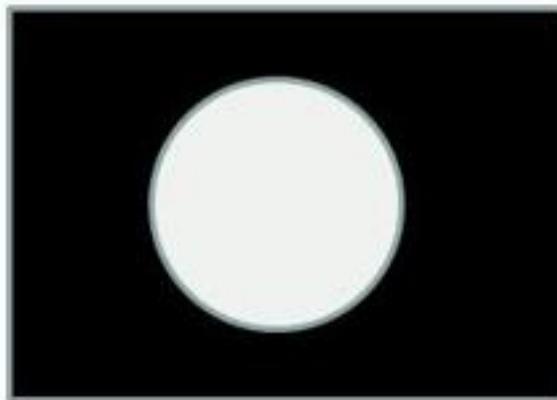
Extraction of rotation invariant signature based on fractal geometry

Y. Tao; T.R. Ioerger; Y.Y. Tang

Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)

Year: 2001 | Volume: 1 | Conference Paper |  
Publisher: IEEE

Draw these synthetic images in paint brush and read in Matlab.  
Write the MATLAB script to extract “Distance-Versus-Angle Signatures” for these geometrical shapes.



- Some simple descriptors
  - The area of a region: **the number of pixels in the region**
  - The perimeter of a region: **the length of its boundary**
  - The compactness of a region:  
**(perimeter)<sup>2</sup>/area**
  - The **mean and median** of the gray levels
  - The **minimum and maximum gray-level values**
  - The **number of pixels with values above and below the mean**

# Regional Descriptors: Statistical Properties

- The area of a region  $A(R)$ : **the number of pixels in the region**

$$A = \sum_m \sum_n x(m, n)$$

0	0	0
0	0	1
0	1	1
0	1	1

- The perimeter of a region  $P(R)$ : **the length of its boundary**

Compactness  $C = \frac{P^2(R)}{A(R)}$

where  $A(R)$  and  $P(R)$  = area and perimeter of region  $R$

# Regional Descriptors: Statistical Properties

- The area of a region  $A(R)$ : the number of pixels in the region

$$A = \sum_m \sum_n x(m, n)$$

- The perimeter of a region  $P(R)$ : the length of its boundary

$$\text{Compactness } C = \frac{P^2(R)}{A(R)}$$

where  $A(R)$  and  $P(R)$  = area and perimeter of region  $R$

Descriptor



Compactness

10.1701

42.2442

15.9836

13.2308

Find the compactness of features as '1' in image

0	0	0
0	0	1
0	1	1
0	1	1

# Fourier Descriptor

Consider each boundary pixel coordinate  $(x, y)$  as a complex number ( $x = \text{real part}$  and  $y = \text{imaginary part}$ ) then find the Fourier transform to a sequence of boundary pixels.

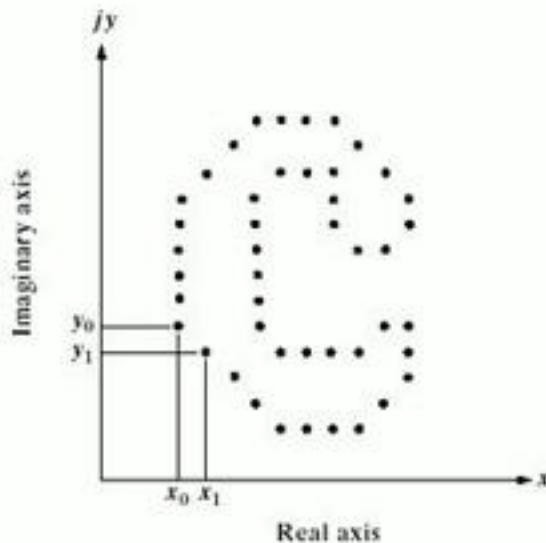
The DFT of  $s(k)$

$$a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-\frac{2\pi u k}{K}}$$

The IDFT of  $a(u)$

$$s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) e^{2\pi u k / K}$$

$$s(k) = x(k) + jy(k)$$



The approximate reconstruction of  $s(k)$  from 'P' Fourier descriptor points:

$$\tilde{s}(k) = \frac{1}{P} \sum_{u=0}^{P-1} a(u) e^{\frac{2\pi u k}{K}}$$

Draw the Fourier descriptors for given figure by number of point as  $P = 2, 4, 8, 16, 32, \dots, 64$

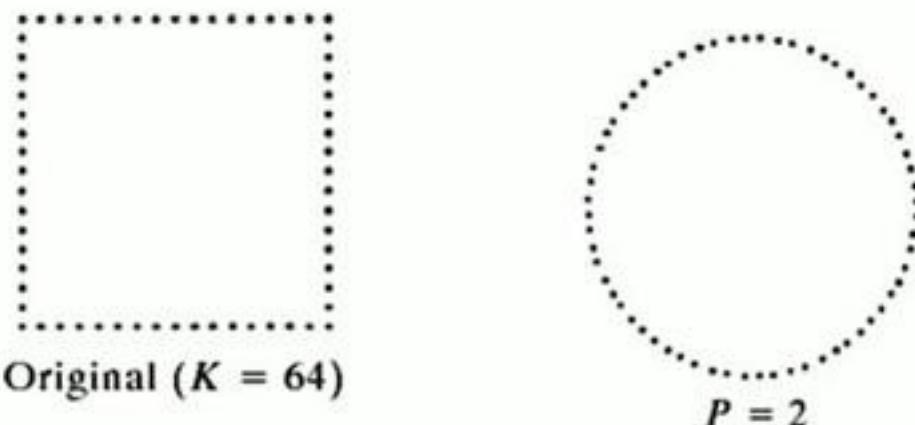


Original ( $K = 64$ )

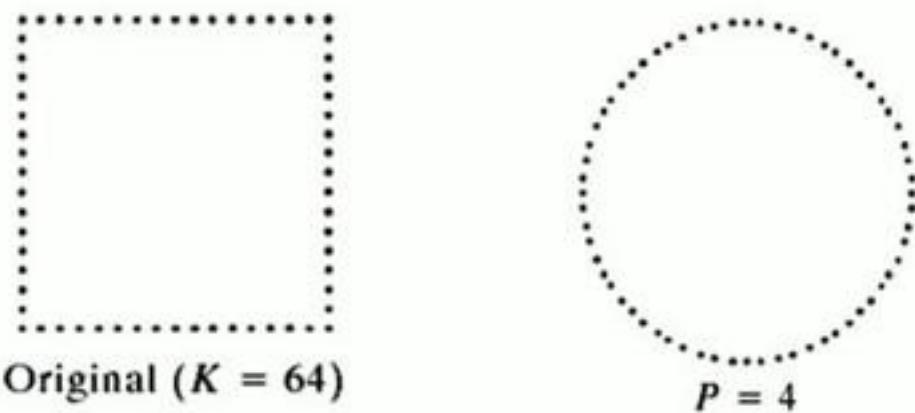
$$\tilde{s}(k) = \frac{1}{P} \sum_{u=0}^{P-1} a(u) e^{\frac{2\pi u k}{K}}$$

## Example: 11.3

$P = 2$  is the number of Fourier coefficients used to reconstruct the boundary

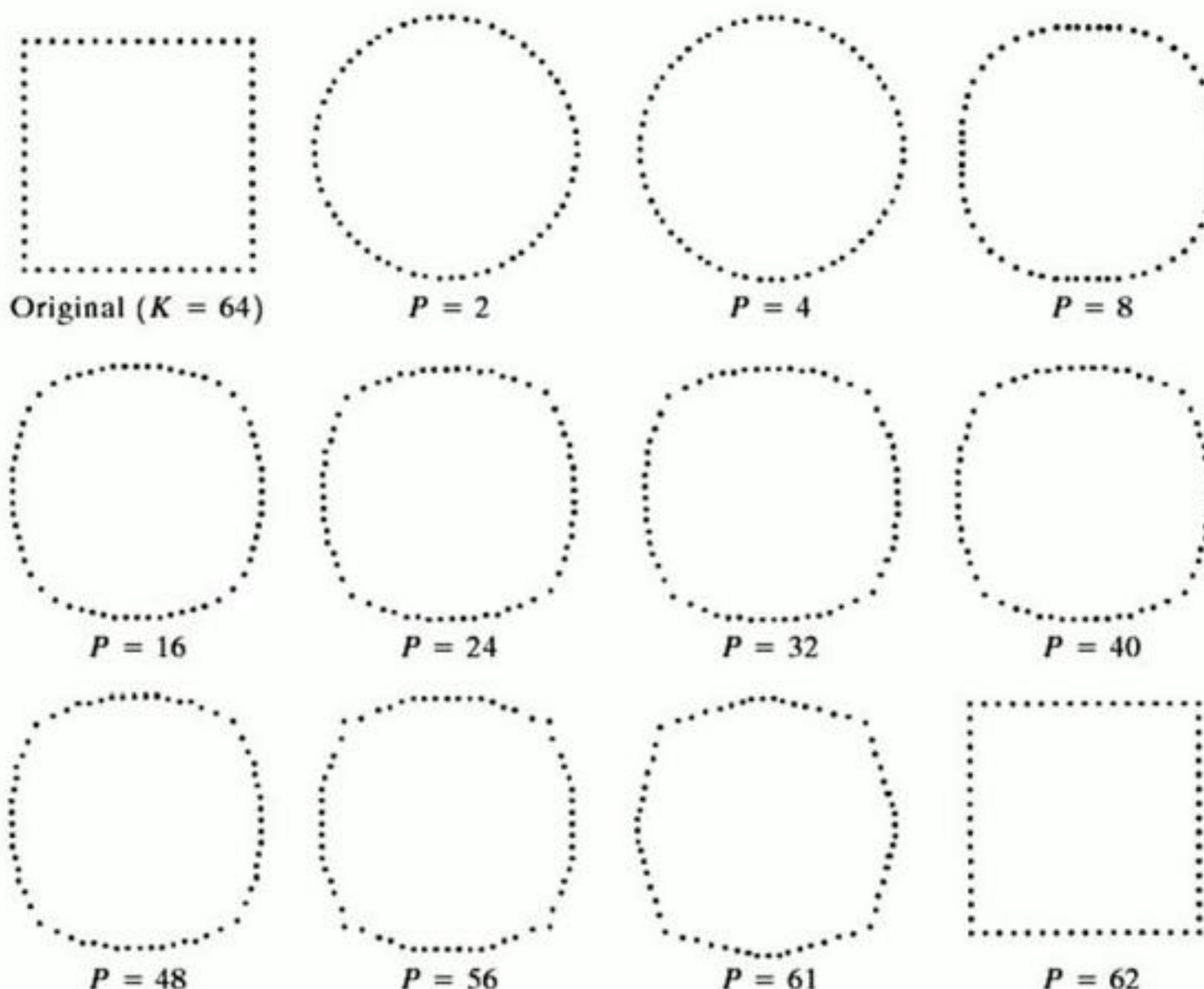


$P = 4$  is the number of Fourier coefficients used to reconstruct the boundary



# Example: 11.3

## The Fourier descriptors



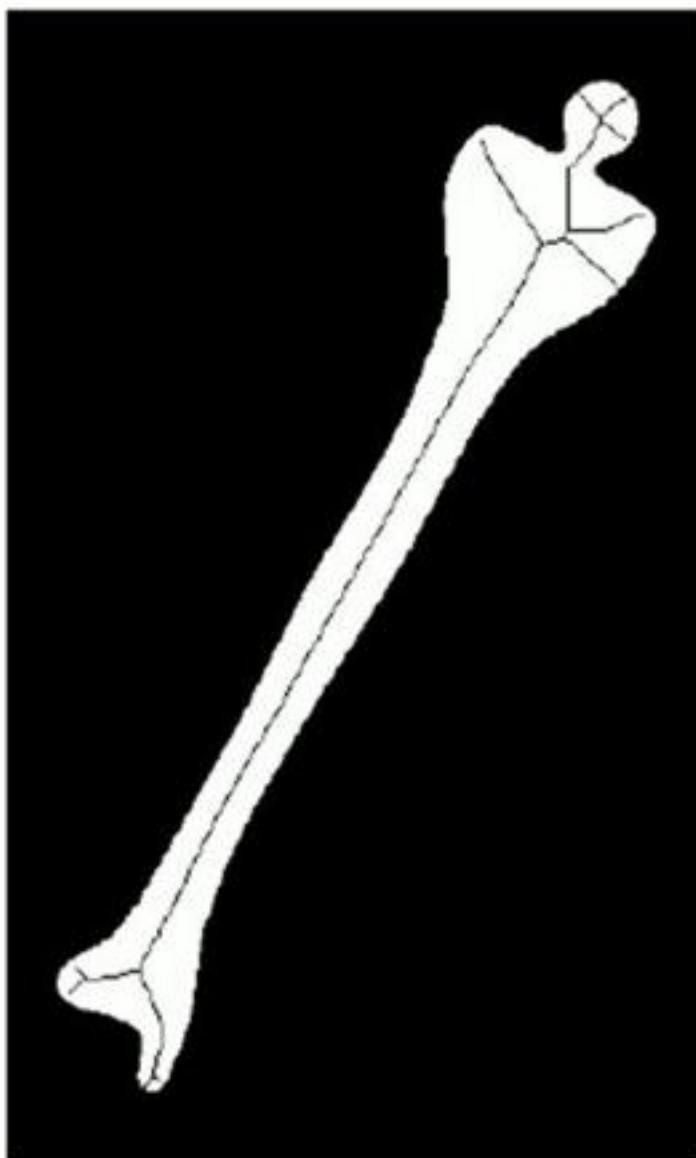
# Fourier Descriptor Properties

Transformation	Boundary	Fourier Descriptor
Identity	$s(k)$	$a(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
Translation	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$
Starting point	$s_p(k) = s(k - k_0)$	$a_p(u) = a(u)e^{-j2\pi k_0 u/K}$

- ❖ Reduces binary image objects to a set of thin strokes.
- ❖ Represent structural shape of a plane region reduce it to a graph.
- ❖ Retains important information about the shape of the original object
- ❖ Use skeletonizing algorithm via a thinning

- One application of skeletonization is for character recognition.
- A letter or character is determined by the center-line of its strokes, and is unrelated to the width of the stroke lines.

**FIGURE 11.10**  
Human leg bone  
and skeleton of  
the region shown  
superimposed.



## Medial Axis Transformation (MAT)

Medial axis is the locus of centers of maximal disks that fit within the shape.

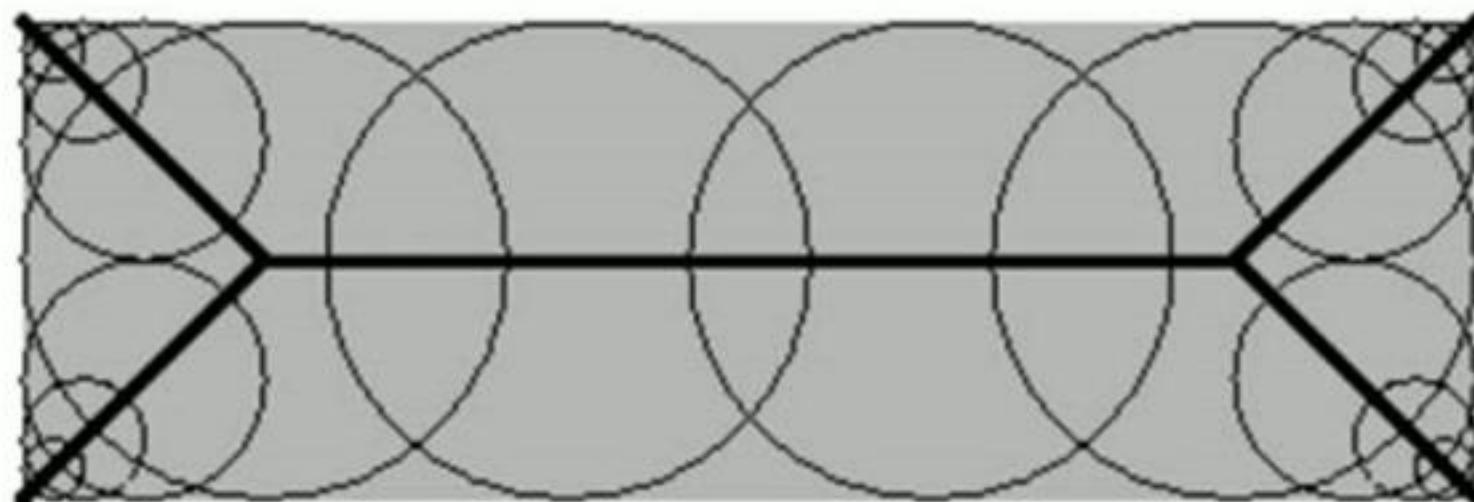


Fig. 14. Construction of the medial axis of a rectangular shape.  
(reprinted from [101]).

# Medial Axis Transformation (MAT)

Medial axis is the locus of centers of maximal disks that fit within the shape.

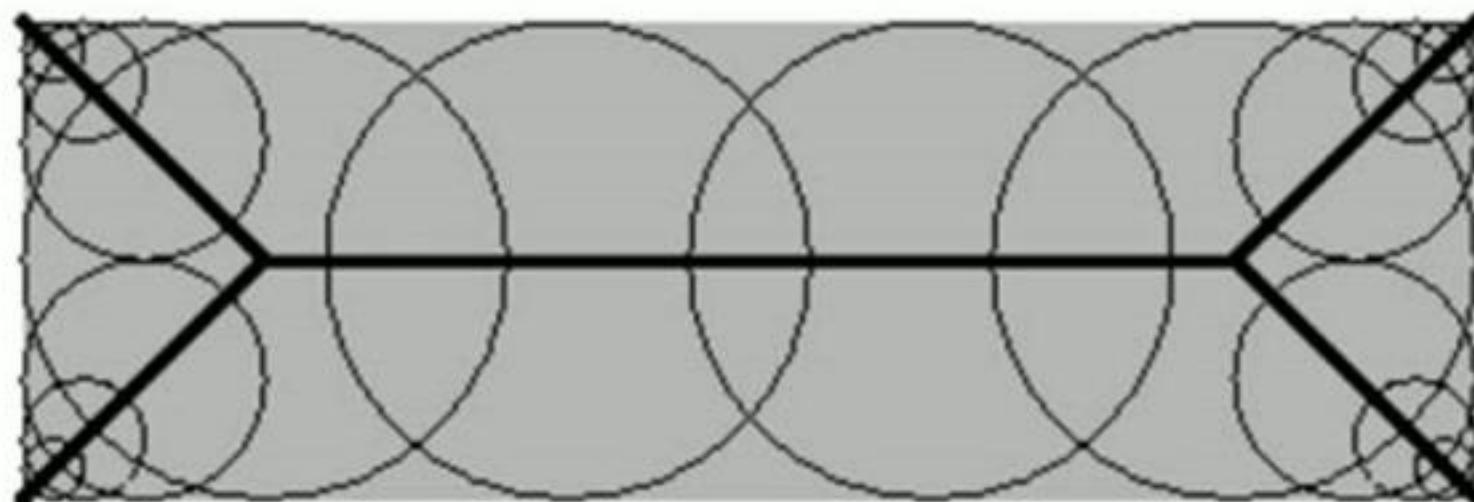
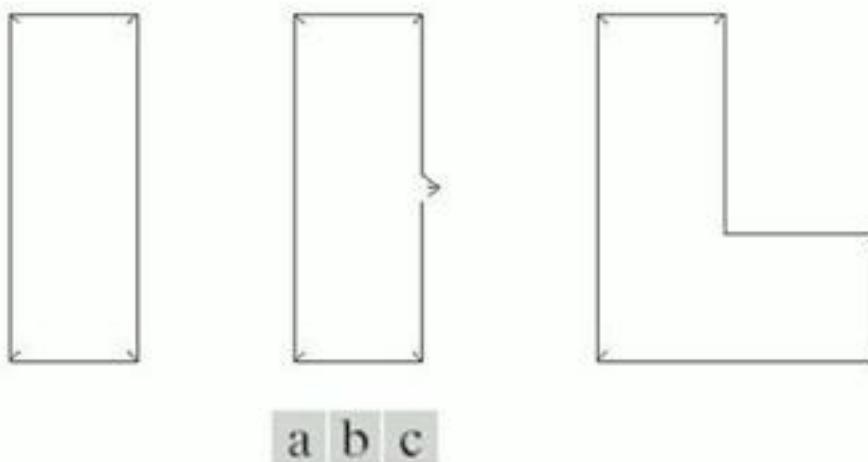


Fig. 14. Construction of the medial axis of a rectangular shape.  
(reprinted from [101]).

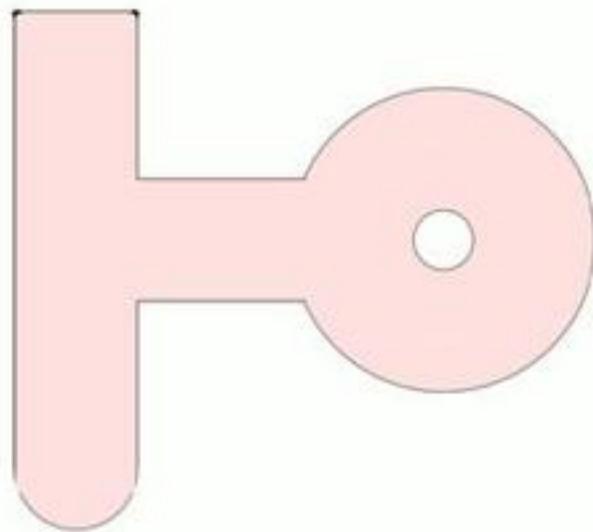
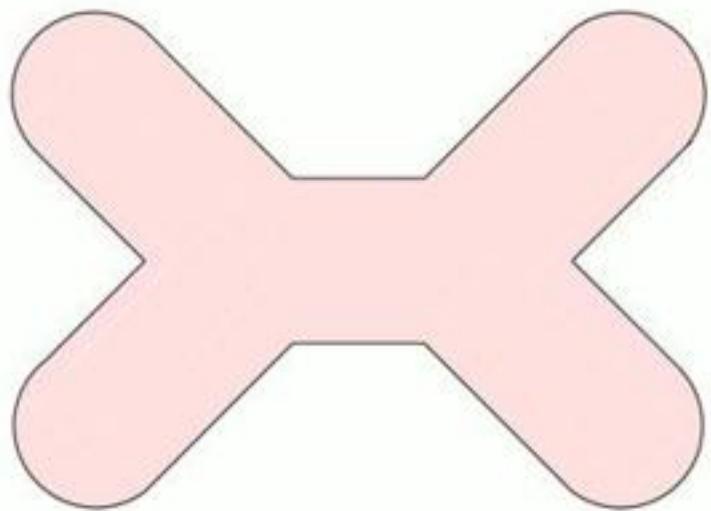
*Prairie-fire analogy:* the boundary of an object is set on fire and the skeleton is the loci where the fire fronts meet and quench each other.

# Medial Axis Transformation (MAT)

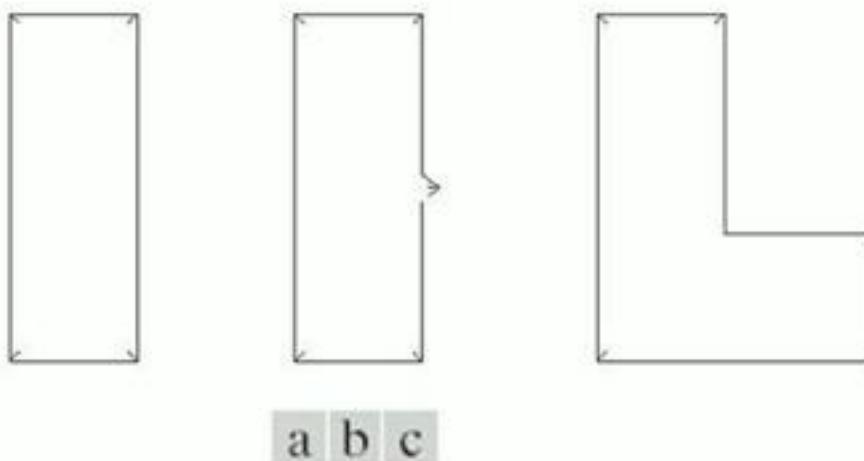


**FIGURE 11.7**  
Medial axes

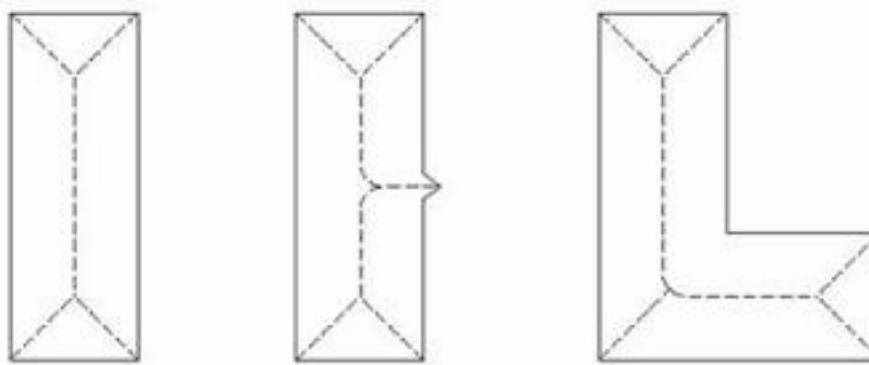
# Medial Axis Transformation (MAT)



# Medial Axis Transformation (MAT)

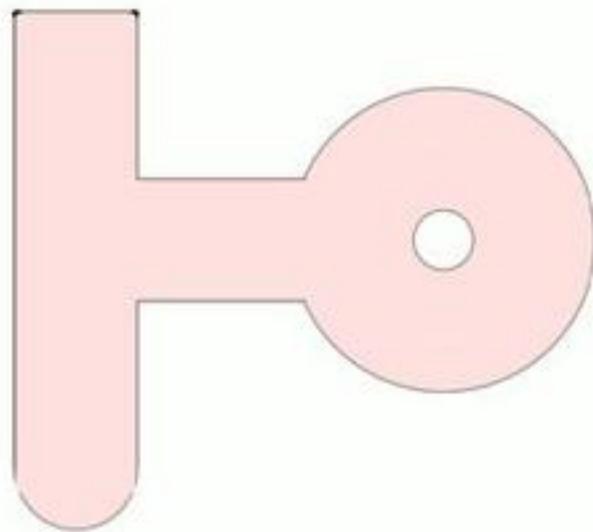
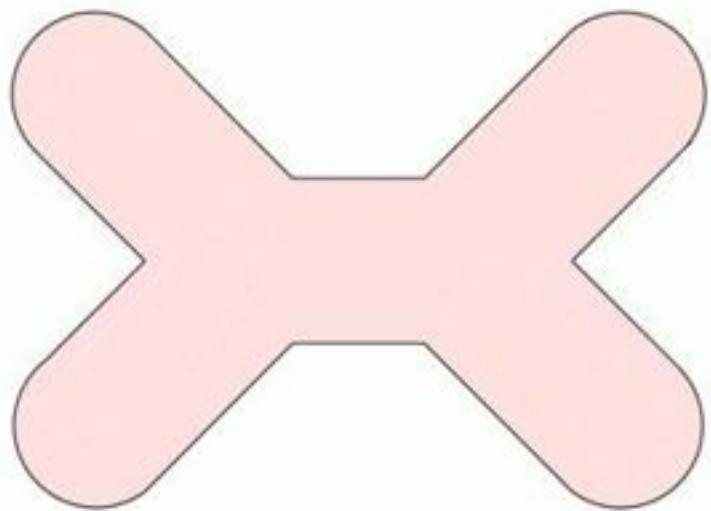


**FIGURE 11.7**  
Medial axes

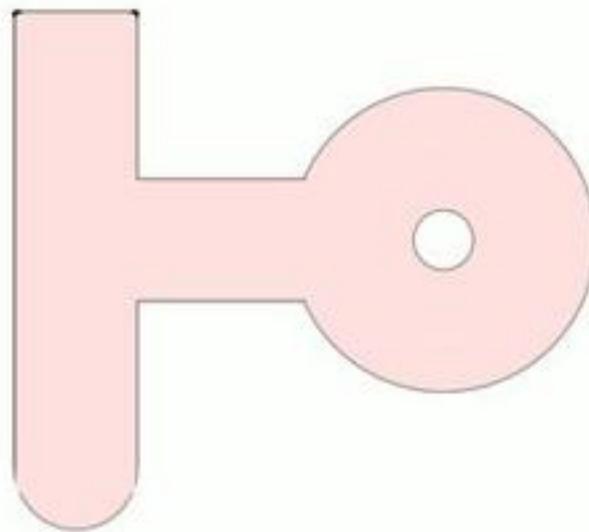
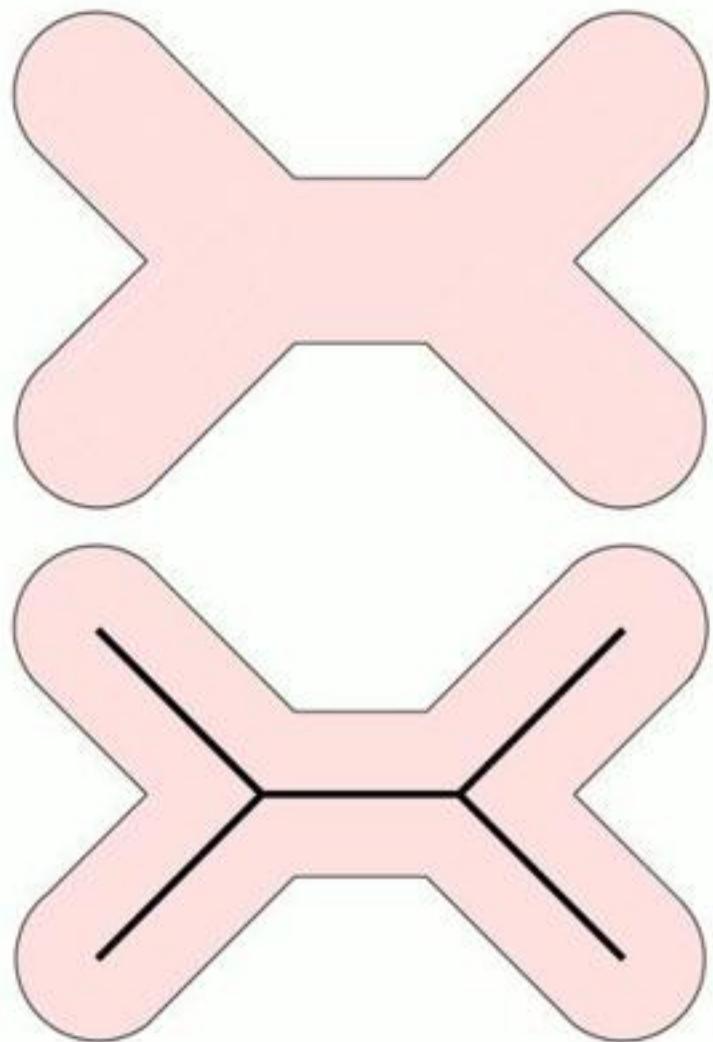


Medial axes  
(dashed) of three  
simple regions.

# Medial Axis Transformation (MAT)



# Medial Axis Transformation (MAT)



Let  $r$  be a random variable, and  $g(r_i)$  be normalized (as the probability of value  $r_i$  occurring), then the moments are –  
the  $n^{\text{th}}$  moment

$$\mu_n(r) = \sum_{i=0}^{K-1} (r_i - m)^n g(r_i)$$

where

$$m = \sum_{i=0}^{K-1} r_i g(r_i)$$

Example of moment--

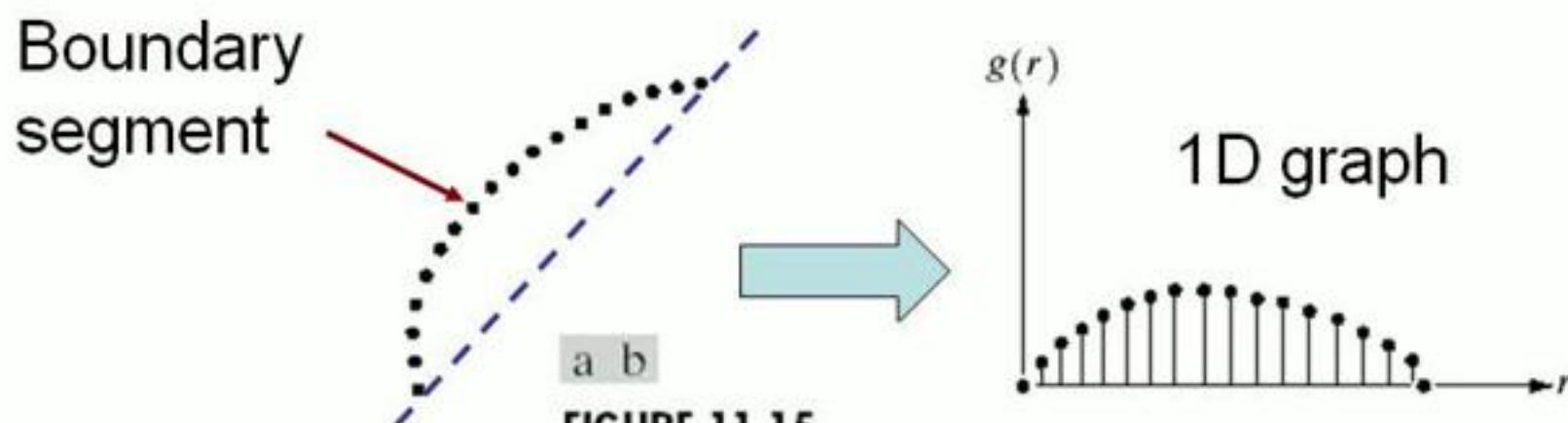
The first moment = mean

The second moment = variance

- Statistical Moments are statistical measures of data.
  - They come in integer orders.
  - Order 0 is just the number of points in the data.
  - Order 1 is the sum and is used to find the average.
  - Order 2 is related to the variance, and order 3 to the skew of the data.
  - Higher orders can also be used, but don't have simple meanings.

$$\mu_n(r) = \sum_{i=0}^{K-1} (r_i - m)^n g(r_i) \quad m = \sum_{i=0}^{K-1} r_i g(r_i)$$

1. Convert a boundary segment into 1D graph
2. View a 1D graph as a PDF function
3. Compute the  $n^{\text{th}}$  order moment of the graph



**FIGURE 11.15**  
(a) Boundary segment.  
(b) Representation as a 1-D function.

## Infrared image of America at night

White pixels represent “light of the cities”

Region no.	% of white pixels compared to the total white pixels
1	20.4%
2	64.0%
3	4.9%
4	10.7%



Use to describe holes and connected components of the region

Topological property 1: the number of holes

Topological property 2: the number of connected Components

Topological property 3:

Euler number: the number of connected components  
subtract the number of holes  $E = C - H$

$C$  = the number of connected components

$H$  = the number of holes

Use to describe holes and connected components of the region

Topological property 1:

the number of holes  $H = 2$

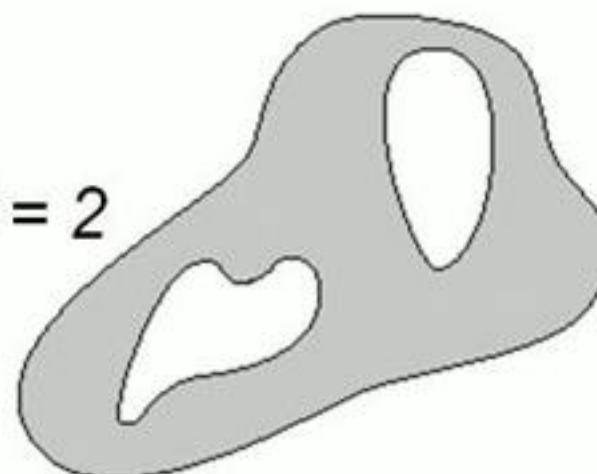


FIGURE 11.17 A region with two holes.

Topological property 2:

the number of connected Components  $C = 3$

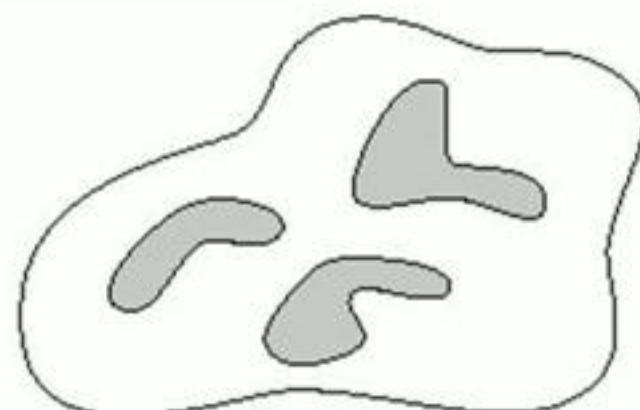


FIGURE 11.18 A region with three connected components.

# Topological Descriptors

Euler number ( $E$ ):  $E = C - H$

